Interface Sensor dan Aktuator
menggunakan Proteus, Arduino, dan LabVIEW

PRAKATA

The fear of the Lord is the beginning of wisdom; and the knowledge of the Holy One is understanding. (Proverbs)

Ucapan Terimakasih

Puji syukur kepada Tuhan YME yang telah memberikan ide, inspirasi, kekuatan dan kesehatan sehingga buku ini bisa terselesaikan. Tidak lupa penulis mengucapkan terimakasih yang sebesar-besarnya kepada:

1. Semua sivitas akademika di Politeknik Mekatronika Sanata Dharma, baik dosen, karyawan dan mahasiswa, atas kesempatan untuk belajar bersama dan dukungannya yang luar biasa.
2. Rekan-rekan saya di Univ. Sanata Dharma, Bp. Andreas Prasetyadi, Bp. Doddy Purwadianto, Bp. Rusdy Sambada, atas kesempatan yang diberikan sehingga saya bisa terlibat dalam penelitan Alat Destilasi Energi Surya, yang menjadi latar belakang buku ini.
3. Bp. S. Krisna W., Bp. Yasrof A. M., Bp. M. Hamka I., Bp. Suryo B., dan semua rekan-rekan di LabVIEW Team Indonesia, yang telah mengajarkan pemrograman LabVIEW dan memberi banyak inspirasi bagi penulis untuk menulis buku ini.
4. Penerbit Deepublishing atas bantuannya sehingga buku ini bisa diterbitkan.
5. Keluarga saya, atas kebersamaannya, yang memberi semangat sehingga penulis bisa menyelesaikan buku ini.
6. Rekan-rekan penulis yang tidak dapat disebutkan satu-persatu, yang sangat membantu penulis dalam penyelesaian buku ini.
7. Pembaca buku ini tentunya, yang sudah merelakan diri untuk membeli dan membaca buku saya, sungguh saya berterimakasih sekali. 😊

Latar Belakang Buku

Buku ini terinspirasi oleh penelitian yang sedang penulis kerjakan bersama rekan-rekan di Univ. Sanata Dharma, yang meneliti tentang pembuatan Alat Destilasi (penyuling air) dengan Energi Surya, sehingga diperoleh konstruksi yang dapat menghasilkan air suling paling banyak. Untuk menghasilkan konstruksi tersebut, tentunya dibutuhkan pengukuran berbagai faktor yang mempengaruhi, meliputi suhu, kelembaban udara, tekanan udara, intensitas cahaya, arah angin, kecepatan angin, laju aliran air, dll.

Sebelumnya, pengukuran dilakukan secara manual, dengan sensor dan alat ukur yang terbatas, yang dilakukan setiap satu jam sekali. Pengukuran secara manual ini memakan banyak waktu dan tenaga dengan hasil perolehan data yang minim. Melihat kesulitan dalam pengukuran tersebut, timbul ide untuk menggunakan LabVIEW dalam pengukuran, dengan Arduino sebagai penghubung ke sensornya. Alasan mengapa memilih LabVIEW dibandingkan software pemrograman lainnya, dikarenakan alasan kemudahan pembuatan grafik dan fasilitas instrumentasi virtual yang lengkap termasuk penyimpanan datanya. Alasan mengapa memilih Arduino dibandingkan hardware instrumentasi lainnya, dikarenakan pertimbangan open source dan tersedianya kode program Arduino, termasuk *library*-nya, untuk berbagai jenis sensor yang ada di pasaran.

Dari ide penggunaan LabVIEW dan Arduino untuk pengukuran, ditambah dengan pengalaman menggunakan LabVIEW, maka penulis telah dapat membuat alat pengukur yang otomatis menyimpan data Sensor. Pengalaman dalam pembuatan itulah yang penulis bagikan dalam buku ini.

Ringkasan Isi Buku

Sesuai dengan judul buku ini, yaitu “Interface Sensor dan Aktuator menggunakan Proteus, Arduino, dan LabVIEW” maka buku ini membahas tentang Interface antara Sensor dan Aktuator dengan Komputer. Istilah “interface” di sini penulis artikan sebagai “menghubungkan”, yaitu menghubungkan Sensor dan Aktuator dengan komputer, dan alat penghubung yang digunakan di buku ini adalah hardware Arduino dan software LabVIEW. Software Proteus di sini digunakan

untuk mensimulasikan hardware Arduino, Sensor dan Aktuator. Penulis sengaja memasukkan software Proteus, agar pembaca yang tidak memiliki hardware Arduino, Sensor dan Aktuator, tetap dapat mempelajari buku ini dan bisa mempraktekkannya. 5 Bab pertama dari buku ini menggunakan Simulasi Proteus, sedangkan 2 Bab terakhir menggunakan hardware Arduino (Arduino Nano), Sensor dan Aktuator, serta modul komunikasi berupa Bluetooth HC-05, dan NRF24L01. Berikut ini gambaran singkat dari masing-masing Bab dalam buku ini.

1. Pembagi Tegangan.

   Bab ini berisi tentang ide sederhana mengenai rangkaian pembagi tegangan dan penerapannya untuk mendeteksi mana dari 4 buah tombol yang sedang ditekan. Software Proteus mulai dikenalkan di Bab ini untuk menunjukkan kemampuan Proteus, baik untuk simulasi rangkaian maupun untuk pengukuran arus dan tegangan pada rangkaian.

2. Pembaca Sinyal Analog.

   Bab ini berisi tentang pembacaan tegangan keluaran dari rangkaian pembagi tegangan menggunakan ADC Arduino dan pemrogramannya, serta bagaimana menampilkan datanya di komputer melalui Serial Monitor. Semua hal tersebut disimulasikan dengan software Proteus dan diprogram dengan software IDE Arduino.

3. Penampil Data.

   Bab ini berisi tentang simulasi rangkaian sensor arah angin, dan bagaimana menghubungkan simulasi tersebut dengan Serial Monitor Arduino dan LabVIEW. Dengan LabVIEW, data rangkaian tidak hanya ditampilkan dalam bentuk teks, tetapi juga bisa dalam bentuk gambar.

4. Grafik, Tabel dan Penyimpanan Data.

   Tidak hanya berupa teks dan gambar, dengan LabVIEW data Sensor dapat ditampilkan dalam bentuk grafik dan tabel. Ada 3 jenis Grafik yang dibahas, yaitu Grafik Waveform Chart, Waveform Graph dan 2D Compass. Di samping tampilan grafik dan tabel, data Sensor tersebut juga dapat disimpan ke dalam file secara berkala disertai dengan catatan waktu penyimpanan. File tempat penyimpanan data tersebut juga dapat dibuka sewaktu-waktu, dan ditampilkan isinya dalam bentuk teks maupun grafik.

5. Interface Sensor dan Aktuator.

   Bab ini berisi tentang simulasi Interface Sensor dan Aktuator menggunakan Proteus, Arduino dan LabVIEW. Ada 3 Sensor dan 3 Aktuator yang dihubungkan, yang semuanya dapat dimonitor, dikontrol dan diatur hubungannya di komputer melalui Arduino dan LabVIEW.
6. Hardware Sensor dan Aktuator.

   Apabila di Bab 5 menggunakan Proteus untuk mensimulasikan Interface Sensor dan Aktuator, maka di Bab 6 ini menggunakan hardware Arduino, Sensor dan Aktuator secara riil untuk mengimplementasikan semua program yang disimulasikan di Bab 5. Di bab ini juga dilakukan pembuatan hardware Sensor Arah Angin dengan 4 buah Reed Switch dan Magnet.
7. Komunikasi Data secara Nirkabel.

   Menghubungkan sesuatu tanpa bersentuhan langsung dan tanpa melalui kabel tentu menjadi hal yang sangat menarik. Bab terakhir ini menyajikan penambahan Bluetooth HC-05 dan NRF24L01 untuk membuat Interface Sensor dan Aktuator dengan Komputer secara nirkabel. Di samping lebih praktis, juga kebutuhan di lapangan untuk memantau dan mengendalikan banyak *nodes* (simpul) dapat diakomodasi melalui komunikasi nirkabel.

Keterbatasan dan Harapan Penulis

Pemrograman Arduino dan LabVIEW dalam buku ini sangat sederhana, dan belum memenuhi standar pemrograman Arduino dan LabVIEW yang baik. Namun demikian, dalam buku ini, penulis telah berupaya semaksimal mungkin untuk menyajikan pengalaman penulis dalam mempelajari keduanya. Rangkaian dan program dalam buku ini juga telah penulis praktekkan dengan hasil yang cukup baik. Apabila pembaca dalam mempraktekkan buku ini, mengalami kesulitan, mohon kiranya pembaca dapat menyampaikan hal tersebut kepada penulis. Untuk informasi lebih jauh, pertanyaan, diskusi, saran, komentar dan kritik, dapat pembaca kirimkan melalui email ke: **dian.artanto@pmsd.ac.id**, atau silahkan mengunjungi blog penulis di **interfacesensordanaktuator.blogspot.co.id**. Terimakasih.

DAFTAR ISI

BAB 1
PEMBAGI TEGANGAN

Target Materi:

- Menyusun rangkaian pembagi tegangan.
- Menyusun rangkaian yang bisa mendeteksi apakah tombol sedang ditekan atau tidak.
- Menggunakan software Proteus untuk mensimulasikan rangkaian yang dapat mendeteksi manakah tombol yang sedang ditekan.

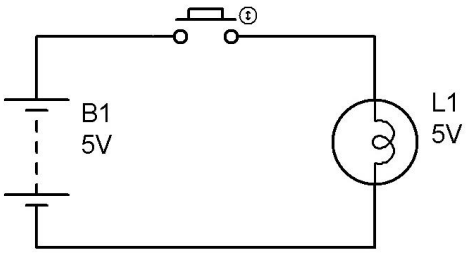
Persoalan:

Buatlah rangkaian yang praktis dan sederhana, yang dapat mendeteksi mana dari 4 buah tombol yang sedang ditekan.

Uraian Materi:

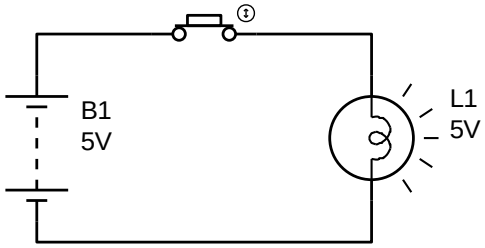
1.1 Pendeteksian Penekanan 1 buah Tombol

Perhatikan rangkaian sederhana berikut ini:



Gambar 1.1 Rangkaian sebuah Batere 5V, Tombol dan Lampu 5V

Pada gambar rangkaian di atas, ketika tombol ditekan, maka Lampu L1 akan menyala, seperti ditunjukkan pada gambar berikut ini.



Gambar 1.2 Lampu menyala ketika Tombol ditekan

Rangkaian sederhana di atas dapat digunakan untuk mendeteksi penekanan sebuah tombol. Ketika tombol ditekan, L1 menyala, ketika tombol dilepas, L1 padam. Jadi bila L1 menyala, berarti tombol sedang ditekan.

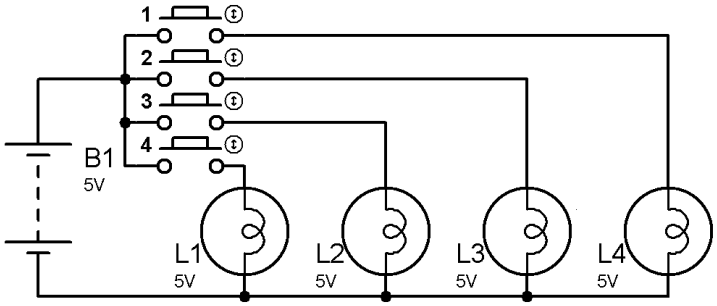
Catatan: Gambar skematik rangkaian di atas dibuat menggunakan software Proteus. Selain bisa digunakan untuk menggambar skematik dan layout rangkaian, software Proteus ini dapat digunakan untuk mensimulasikan kerja rangkaian tersebut. Pembaca disarankan memiliki software Proteus untuk bisa mensimulasikan materi dalam buku ini. Sekedar informasi, semua gambar skematik rangkaian yang ditampilkan dalam buku ini, dapat pembaca ambil di CD yang disertakan dalam buku ini, dalam bentuk file Proteus, dengan nama file sesuai nomor gambarnya.

1.2 Pendeteksian Penekanan 4 buah Tombol

Telah ditunjukkan bahwa rangkaian pada Gambar 1.1 di atas dapat mendeteksi penekanan sebuah tombol. Bila tombolnya lebih banyak, yaitu 4 buah tombol, seperti apakah rangkaian pendeteksiannya? Tentu saja ada banyak solusi rangkaian yang bisa diberikan. Dua di antaranya adalah sebagai berikut:

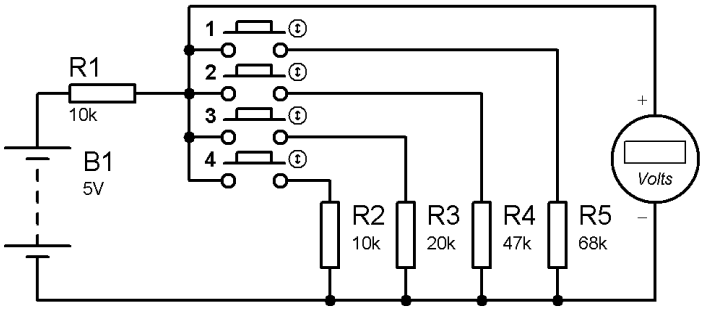
1. Solusi pertama adalah seperti rangkaian pada Gambar 1.1, namun dengan lebih banyak lampu, yaitu 4 lampu untuk 4 tombol. Masing-masing tombol akan dihubungkan dengan sebuah lampu, yang disusun seperti rangkaian

pada Gambar 1.3 berikut. Jadi bila tombol ditekan, lampu yang terhubung dengan tombol tersebut akan menyala.



Gambar 1.3 Rangkaian solusi pertama dengan 4 lampu untuk 4 tombol

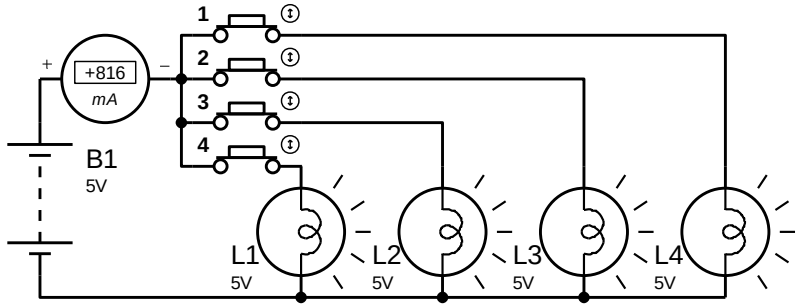
2. Solusi kedua adalah dengan membuat rangkaian pembagi tegangan, yang disusun dengan 5 buah resistor; 1 resistor 10 kohm dan 4 resistor dengan nilai hambatan yang bervariasi, yang masing-masing terhubung dengan sebuah tombol, dan sebuah Voltmeter yang dipasang pada titik temu keempat tombol, seperti ditunjukkan pada Gambar 1.4 berikut ini.



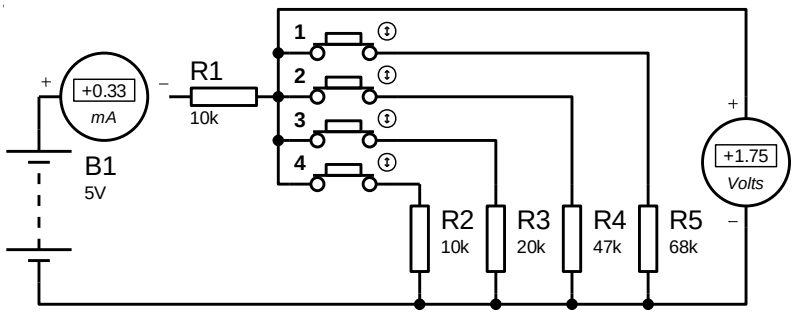
Gambar 1.4 Rangkaian solusi kedua dengan rangkaian pembagi tegangan

Dua solusi rangkaian di atas masing-masing memiliki kelebihan dan kekurangan masing-masing. Kelebihan rangkaian solusi pertama dibandingkan solusi kedua adalah tombol yang ditekan dapat langsung diketahui dari lampu yang menyala. Sedangkan untuk solusi kedua, tombol yang ditekan tidak dapat langsung diketahui dari Voltmeter. Voltmeter hanya menampilkan nilai tegangan. Untuk mengetahui mana tombol yang ditekan, nilai tegangan tersebut perlu

"diterjemahkan" dulu. Namun demikian, kelebihan dari rangkaian solusi kedua adalah hemat konsumsi daya dibandingkan dengan rangkaian solusi pertama. Untuk membuktikannya, tambahkan alat ukur arus (Ampermeter) pada rangkaian solusi pertama maupun solusi kedua. Perhatikan arus yang mengalir pada rangkaian ketika keempat tombol ditekan. Berikut ini hasil simulasi rangkaian menggunakan software Proteus untuk kedua solusi rangkaian:



Gambar 1.5 Ampermeter menunjukkan arus yang mengalir 816 mA



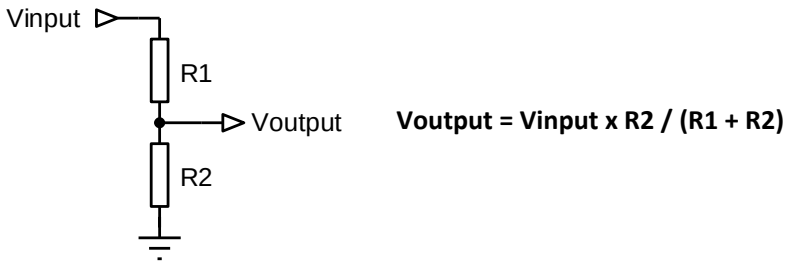
Gambar 1.6 Ampermeter menunjukkan arus yang mengalir 0,33 mA

Tampak dari Gambar 1.5 arus yang mengalir pada rangkaian solusi pertama adalah sebesar 816 mA. Dengan sumber tegangan sebesar 5V, maka konsumsi daya setiap detiknya adalah sebesar 5 x 816 = 4080 mWatt. Bandingkan dengan rangkaian solusi kedua di Gambar 1.6, arus yang mengalir hanya sebesar 0,33 mA. Dengan sumber tegangan sebesar 5V, maka konsumsi daya rangkaian solusi kedua setiap detiknya hanyalah sebesar 5 x 0,33 = 1,65 mWatt, atau 0,0004 kali dari konsumsi daya rangkaian solusi pertama.

Dengan konsumsi daya yang jauh lebih hemat dari rangkaian solusi pertama, maka rangkaian solusi kedua menjadi pilihan solusi yang terbaik. Meskipun untuk mengetahui mana tombol yang ditekan, rangkaian solusi kedua perlu "diterjemahkan" dulu, namun itu tidak menjadi soal. Sub Bab berikut akan menunjukkan cara "menerjemahkan" nilai tegangan dalam kaitannya dengan tombol yang ditekan.

1.3 Rangkaian Pembagi Tegangan

Rangkaian solusi kedua menggunakan rangkaian pembagi tegangan. Apa itu rangkaian pembagi tegangan? Sesuai dengan namanya, rangkaian pembagi tegangan adalah rangkaian yang membagi tegangan dengan nilai tertentu, yang sebanding dengan nilai resistor-resistor yang menyusunnya. Jadi umumnya rangkaian pembagi tegangan tersusun dari minimal 2 buah resistor yang dirangkai seri, seperti ditunjukkan pada Gambar 1.7 berikut ini. Nilai tegangan output pada rangkaian tersebut dapat dihitung dengan rumus berikut:



Gambar 1.7 Rangkaian pembagi tegangan 2 resistor dan rumusnya

Dengan menggunakan rumus pembagi tegangan tersebut, akan dapat diperoleh nilai tegangan untuk semua kombinasi penekanan keempat tombol di rangkaian solusi kedua. Sebagai contoh, untuk 4 buah kondisi tombol berikut ini:.

- Kondisi 1: Tombol pertama saja yang ditekan.
- Kondisi 2: Tombol pertama dan kedua yang ditekan.
- Kondisi 3: Tombol pertama, kedua dan ketiga yang ditekan.
- Kondisi 4: Semua tombol ditekan.

Berikut ini perhitungan untuk keempat kondisi di atas:

Perhitungan untuk Kondisi 1: Tombol pertama saja yang ditekan.

- Ketika tombol 1 ditekan, maka resistor-resistor yang menyusun rangkaian pembagi tegangan adalah resistor R1 (10k ohm) dan R5 (68k ohm). Dengan memasukkan nilai R1 dan R5 pada rumus di atas, menghasilkan nilai tegangan output sebesar 5 x 68k/(10k+68k) = 4,36V.

Perhitungan untuk Kondisi 2: Tombol pertama dan kedua yang ditekan.

- Ketika tombol 1 dan 2 ditekan, maka resistor-resistor yang menyusun rangkaian pembagi tegangan adalah resistor R1 (10k ohm), R4 (47k ohm) dan R5 (68k ohm). Resistor R4 dan R5 tersusun secara paralel. Nilai gabungan 2 buah resistor paralel tersebut adalah sebesar (R4xR5)/(R4+R5) = (68k x 47k)/(68k + 47k) = 27791,3. Dengan memasukkan nilai R1 (10k ohm) dan nilai paralel R4 dan R5 (27791,3 ohm) pada rumus pembagi tegangan di atas, menghasilkan nilai tegangan output sebesar 5 x 27791,3/(10k+27791,3) = 3,68V.

Perhitungan untuk Kondisi 3: Tombol pertama, kedua dan ketiga yang ditekan.

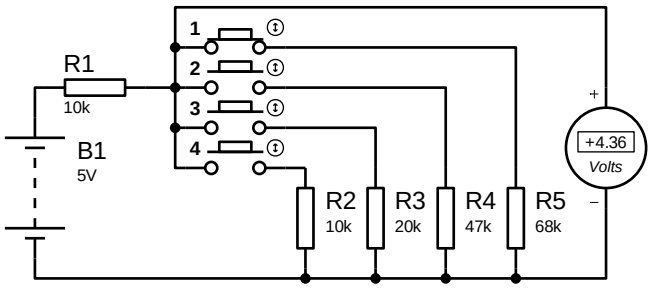
- Ketika tombol 1, 2 dan 3 ditekan bersama, maka resistor-resistor yang menyusun rangkaian pembagi tegangan adalah resistor R1 (10k ohm), R3 (20k ohm), R4 (47k ohm) dan R5 (68k ohm). Resistor R3, R4 dan R5 tersusun secara paralel. Nilai gabungan 3 buah resistor paralel tersebut adalah sebesar (R3xR4R5)/(R3+R4R5) = (20k x 27791,3)/(20k + 27791,3) = 11630,3. Dengan memasukkan nilai R1 (10k ohm) dan nilai gabungan R3, R4 dan R5 (11630,3 ohm) pada rumus di atas, menghasilkan nilai tegangan output sebesar 5 x 11630,3/(10k + 11630,3) = 2,69V.

Perhitungan untuk Kondisi 4: Semua tombol ditekan.

- Ketika semua tombol ditekan bersama, maka rangkaian pembagi tegangan terbentuk dari resistor R1, dan paralel dari resistor R2, R3, R4 dan R5. Nilai gabungan R2, R3, R4 dan R5 adalah sebesar (10k x 11630,3)/(10k + 11630,3) = 5376,86 ohm. Dengan nilai paralel sebesar itu, dan R1 sebesar 10k, akan menghasilkan nilai tegangan output yang terbaca sebesar 5 x 5376,86/(10k + 5376,86) = 1,75V.

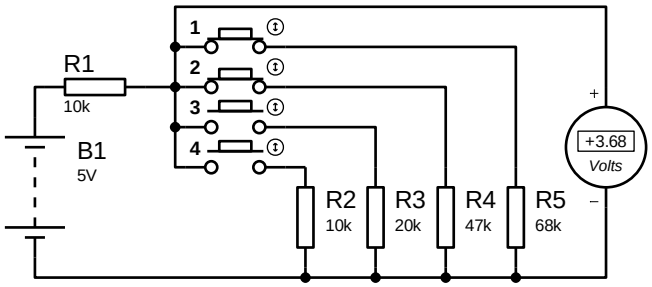
Untuk membuktikan bahwa hasil perhitungan di atas benar, pembaca dapat mensimulasikan keempat kondisi tombol di atas dengan software Proteus, seperti ditunjukkan dalam gambar berikut:

1. Simulasi Kondisi 1: Tombol pertama saja yang ditekan.



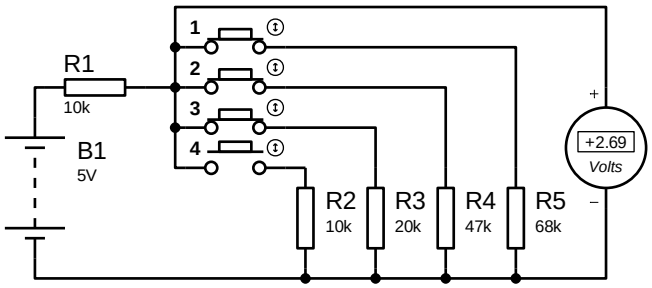
Gambar 1.8 Hasil simulasi menunjukkan tegangan output 4,36V

2. Simulasi Kondisi 2: Tombol pertama dan kedua yang ditekan.



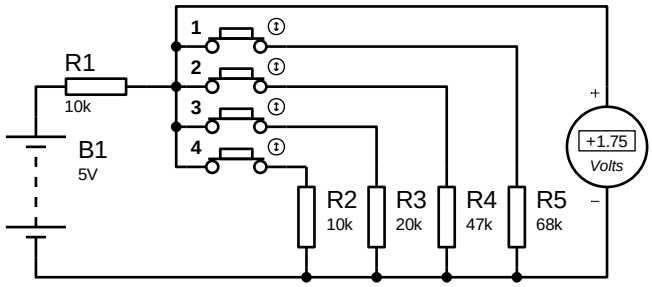
Gambar 1.9 Hasil simulasi menunjukkan nilai output 3,68V

3. Simulasi Kondisi 3: Tombol pertama, kedua dan ketiga yang ditekan.



Gambar 1.10 Hasil simulasi menunjukkan tegangan output 2,69V

4. Simulasi Kondisi 4: Semua tombol ditekan.



Gambar 1.11 Hasil simulasi menunjukkan tegangan output 1,75V

Tampak dari Gambar 1.8, 1.9, 1.10 dan 1.11, nilai tegangan output dari hasil simulasi dengan software Proteus menunjukkan nilai yang sama dengan hasil perhitungan. Untuk kombinasi penekanan tombol yang lain, penulis berikan sebagai latihan bagi pembaca untuk menemukannya (lihat Soal Latihan No. 2). Dengan diperolehnya nilai tegangan output untuk semua kombinasi penekanan tombol, maka "penerjemahan" nilai tegangan output dalam kaitannya dengan mana tombol yang ditekan, akan dapat dilakukan dengan mudah.

1.4 Soal Latihan

1. Rangkaian solusi kedua pada Gambar 1.4 menggunakan nilai resistor R2, R3, R4 dan R5 yang tidak sama nilainya. Mengapa nilai resistor-resistor tersebut tidak sama? Apa akibatnya bila nilai resistor-resistor tersebut dibuat sama?

2. Lengkapi tabel berikut dengan nilai tegangan output dari hasil perhitungan dan dari simulasi software Proteus sesuai kondisi tombol pada rangkaian solusi kedua Gambar 1.4, dengan nilai resistor R1=10kΩ, R2 = 10kΩ, R3 = 20kΩ, R4 = 47kΩ dan R5 = 68kΩ.

No.	Kondisi tombol	Nilai Tegangan Output (V)	
		Perhitungan	simulasi Proteus
1.	Tidak ada tombol ditekan.	...	...
2.	Tombol 1 ditekan.	...	...

No.	Kondisi tombol	Nilai Tegangan Output (V)	
		Perhitungan	simulasi Proteus
3.	Tombol 2 ditekan.	...	...
4.	Tombol 3 ditekan.	...	...
5.	Tombol 4 ditekan.	...	...
6.	Tombol 1 dan 2 ditekan.	...	...
7.	Tombol 2 dan 3 ditekan.	...	...
8.	Tombol 3 dan 4 ditekan.	...	...
9.	Tombol 4 dan 1 ditekan.	...	...
10.	Tombol 1 dan 3 ditekan.	...	...
11.	Tombol 2 dan 4 ditekan.	...	...
12.	Tombol 1, 2 & 3 ditekan.	...	...
13.	Tombol 1, 2 & 4 ditekan.	...	...
14.	Tombol 1, 3 & 4 ditekan.	...	...
15.	Tombol 2, 3 & 4 ditekan.	...	...
16.	Semua tombol ditekan.	...	...

3. Berdasarkan hasil nilai tegangan output pada Tabel di Soal No. 2 di atas, terjemahkanlah kondisi tombol untuk nilai tegangan output pada Tabel berikut ini.

No.	Nilai Tegangan (V)	Kondisi Tombol
1.	1,75	...
2.	1,84	...
3.	1,89	...
4.	2	...
5.	2,12	...
6.	2,26	...
7.	2,33	...

No.	Nilai Tegangan (V)	Kondisi Tombol
8.	2,50	...
9.	2,69	...
10.	2,92	...
11.	3,04	...
12.	3,33	...
13.	3,68	...
14.	4,12	...
15.	4,36	...
16.	5	...

BAB 2
PEMBACA SINYAL ANALOG

Target Materi:

- Menyusun rangkaian untuk membaca sinyal analog.
- Memprogram Arduino untuk membaca sinyal analog.
- Menggunakan software Proteus untuk mensimulasikan rangkaian pembaca sinyal analog.

Persoalan:

Buatlah rangkaian yang praktis dan sederhana, yang dapat membaca sinyal analog yang dihasilkan oleh rangkaian pendeteksi penekanan tombol.

Uraian Materi:

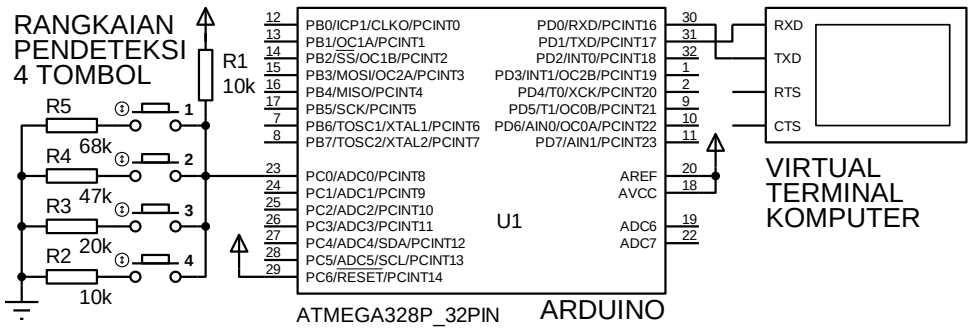
2.1 Rangkaian Pembaca Sinyal Analog

Dari 2 rangkaian pendeteksi penekanan tombol yang diusulkan di Bab 1, telah dipilih rangkaian kedua sebagai solusi terbaik karena konsumsi dayanya cukup rendah. Namun demikian, rangkaian kedua ini membutuhkan Voltmeter untuk membaca nilai tegangan output. Penggunaan Voltmeter jelas tidak praktis, karena selain ukurannya biasanya cukup besar, juga untuk mengetahui mana tombol yang ditekan, masih harus "menerjemahkan", yaitu dengan cara mencarinya dari tabel hubungan antara nilai tegangan output dengan kondisi tombol yang telah dibuat sebelumnya. Untuk itu perlu pengganti Voltmeter yang dapat membaca nilai tegangan output. Untuk itu perlu rangkaian pembaca sinyal analog.

Apa itu sinyal analog? Lawan dari sinyal analog adalah sinyal digital. Apabila sinyal digital adalah sinyal yang hanya memiliki 2 kondisi, yaitu kondisi High dan Low saja, maka sinyal analog adalah sinyal yang memiliki lebih dari 2 kondisi, bisa 1V, 2V, 3V dan seterusnya. Sinyal seperti itulah yang dihasilkan oleh rangkaian pendeteksi penekanan tombol.

Untuk membuat rangkaian pembaca sinyal analog, cara paling mudah adalah menggunakan Arduino. Apa itu Arduino? Arduino adalah sebuah nama dari sistem minimum dengan mikokontroler di dalamnya, yang dilengkapi juga dengan software pemrograman untuk mikrokontroler tersebut.

Mengapa Arduino? Karena Arduino memiliki rangkaian ADC (*Analog to Digital Converter*) di dalamnya, yaitu rangkaian yang dapat mengubah sinyal analog menjadi data digital. Data digital tersebut kemudian dapat dibaca dan diolah serta bisa dikirimkan ke komputer melalui komunikasi serial. Lebih jelasnya mengenai rangkaian pembaca sinyal analog menggunakan Arduino, dapat dilihat pada gambar skematik berikut ini:



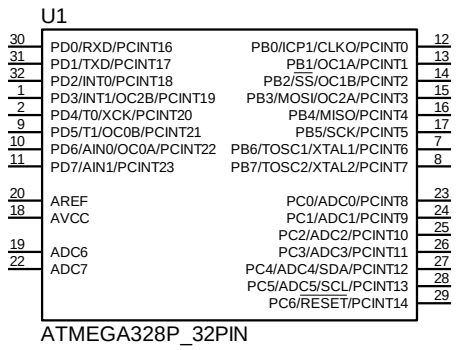
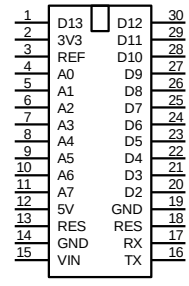
Gambar 2.1 Rangkaian pembaca sinyal analog dengan Arduino

Dari gambar 2.1 di atas, tampak rangkaian terdiri dari 3 bagian, yaitu:

1. Bagian pertama adalah rangkaian pendeteksi 4 buah tombol (rangkaian yang sama dengan Gambar 1.6).
2. Bagian kedua adalah Arduino, yang dihadirkan dalam bentuk komponen mikrokontroler ATmega328_32pin. Mengapa? Karena mikrokontroler pada Arduino Nano adalah ATmega328 dengan 32 kaki.
3. Bagian ketiga adalah komputer, yang diwakili dengan Virtual Terminal.

Berikut ini tabel hubungan kaki ATmega328_32pin dengan kaki Arduino Nano:

Tabel 2.1 Persamaan kaki ATmega328_32pin dengan Arduino Nano

ATmega328_32pin	Arduino Nano	Persamaan Kaki	
		PD0=RX	PB3=D11
		PD1=TX	PB4=D12
		PD2=D2	PB5=D13
		PD3=D3	PC0=A0
		PD4=D4	PC1=A1
		PD5=D5	PC2=A2
		PD6=D6	PC3=A3
		PD7=D7	PC4=A4
		PB0=D8	PC5=A5
		PB1=D9	ADC6=A6
		PB2=D10	ADC7=A7

Catatan: Penulis memilih Arduino jenis Nano karena harga Arduino Nano relatif lebih murah dibandingkan jenis lain, dengan ukuran yang cukup kecil, serta bisa dipasang di *Breadboard*. Sebenarnya ada juga jenis Arduino yang lebih murah, yaitu Arduino Mini. Hanya saja, Arduino Mini tidak dilengkapi dengan IC konverter USB ke TTL, sehingga untuk menghubungkannya ke komputer harus menambahkan komponen tersebut. Apabila pembaca menggunakan Arduino jenis lain, program di buku ini tetap dapat diterapkan, hanya saja pembaca perlu melakukan beberapa penyesuaian, terutama pada kaki IO yang digunakan.

Dari Gambar 2.1 di atas, pembaca perlu memperhatikan sambungan antar ketiga bagian tersebut, yang dijelaskan sebagai berikut:

1. Output rangkaian pendeteksi tombol terhubung dengan kaki PC0 ATmega328, atau sama dengan kaki A0 Arduino.
2. Virtual Terminal komputer, yang merepresentasikan komputer, terhubung dengan kaki PD0 dan PD1 ATmega328 atau sama dengan kaki RX dan TX Arduino. Kaki RX dan TX Arduino ini terhubung ke USB Konverter, sehingga akan terhubung ke port serial komputer setiap kali kabel USB Arduino dihubungkan ke komputer.

3. Tanda panah ke atas pada rangkaian pendeteksi 4 tombol menunjukkan simbol tegangan 5V, dan 3 garis mendatar membentuk segitiga menunjukkan simbol Ground.

Dengan rangkaian di atas, setiap kali salah satu tombol, atau beberapa tombol ditekan, akan menghasilkan sinyal analog dengan tegangan berkisar antara 0 – 5V. Sinyal analog ini kemudian dibaca oleh kaki ADC Arduino, untuk kemudian diubah menjadi data digital dan ditampilkan di Serial Monitor di Komputer, dalam bentuk angka yang memiliki jangkauan nilai antara 0 – 1023. Mengapa nilainya bisa menjadi 0 – 1023? Ini dikarenakan resolusi ADC yang dimiliki Arduino adalah 10 bit, dengan jangkauan mulai dari 00 0000 0000 hingga 11 1111 1111, yang bila diubah menjadi angka desimal menjadi 0 – 1023.

Dengan resolusi sebesar 10 bit tersebut, ADC yang dimiliki Arduino memiliki kepekaan pembacaan sinyal analog paling kecil hingga 5/1023 = 0,0048V. Sehingga setiap kali ada perubahan input sinyal analog sekecil-kecilnya 4,8mV sudah bisa membuat perubahan 1 angka pada output data digitalnya.

2.2 Pemrograman ADC Arduino

Agar bisa membaca sinyal analog dan mengubahnya ke data digital, tentunya Arduino harus diprogram. Berikut ini program untuk membaca sinyal analog, mengubahnya ke data digital dan menampilkan datanya di Komputer.

Program_2_1.ino

```
1.  void setup() {
2.    Serial.begin(9600);
3.  }
4.
5.  void loop() {
6.    int a = analogRead(A0);
7.    Serial.println(a);
8.    delay(1000);
9.  }
```

Program di atas dimulai dengan **void setup**. Sekedar informasi, program Arduino selalu melibatkan 2 buah void, yaitu **void setup** dan **void loop**.

Apa itu **void setup**? **void setup** adalah sebuah blok program, yang berisi instruksi program yang hanya dijalankan sekali di awal, yaitu setiap kali Arduino dihidupkan. Beberapa instruksi yang ditaruh di **void setup** adalah instruksi untuk kode komunikasi, penugasan kaki IO, inisialisasi program, dll. Seperti terlihat di baris kedua, instruksi yang ditempatkan di dalam blok **void setup** ini adalah **Serial.begin(9600)**. Instruksi tersebut digunakan untuk mengaktifkan komunikasi serial dengan nilai baudrate atau kecepatan komunikasi sebesar 9600 bit/detik.

Program berikutnya di baris kelima adalah **void loop**. Sama seperti **void setup**, **void loop** juga sebuah blok program, hanya bedanya, **void loop** berisi instruksi program yang akan dijalankan terus-menerus sampai Arduino dimatikan. Seperti terlihat di baris ke 6 hingga 8, ada 3 instruksi program, yaitu:

1. Instruksi di baris ke-6 untuk membaca sinyal analog di kaki **A0**, mengubahnya menjadi data digital, dan kemudian menyimpannya ke dalam variabel **a** dengan tipe data integer.
2. Instruksi di baris ke-7 untuk menampilkan data digital variabel **a** tersebut di komputer melalui komunikasi serial.
3. Instruksi di baris ke-8 untuk menunggu selama 1000 mili detik.

Karena ketiga instruksi tersebut berada di dalam blok **void loop**, maka ketiga instruksi tersebut akan dijalankan secara berurutan dari atas ke bawah, dan kemudian diulang dari atas kembali terus-menerus, hingga Arduino dimatikan.

Agar program Arduino di atas bisa ditanamkan ke memori program Arduino, program di atas harus dijadikan kode mesin. Untuk mengubah dari bahasa program Arduino ke dalam kode mesin, program Arduino harus dikompilasi. Namun sebelum proses pengubahan tersebut, bahasa program Arduino akan diperiksa atau diverifikasi, apakah ada kesalahan pada program atau tidak. Proses verifikasi dan kompilasi ini akan dijalankan software IDE Arduino apabila tombol ***verify***, atau tombol bergambar tanda centang di Toolbar paling kiri ditekan.

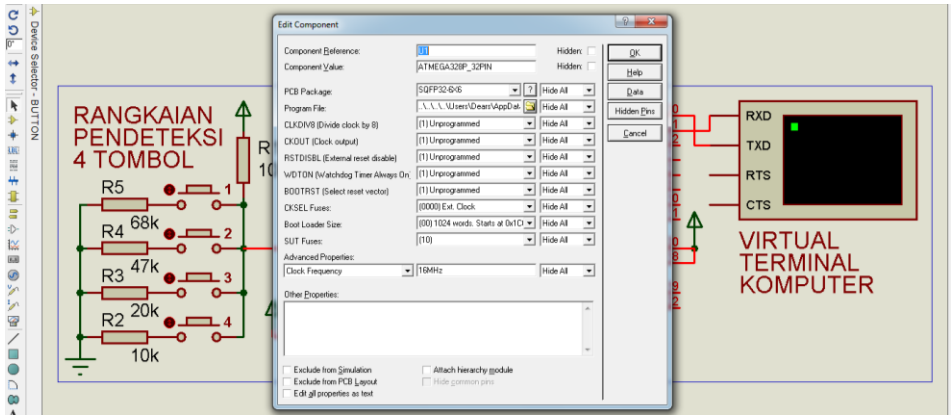
Apabila proses kompilasi telah berhasil, ditunjukkan dengan munculnya pesan ***Done Compiling***, langkah selanjutnya adalah proses ***upload*** atau proses penanaman kode mesin ke memori program Arduino. Agar software IDE Arduino bisa menjalankan proses ***upload***, tekan tombol bergambar tanda panah ke kanan. Apabila proses ***upload*** berhasil, maka akan muncul pesan ***Done Uploading***.

Proses ***upload*** hanya berhasil apabila saluran port yang dipilih dan jenis Arduinonya benar sesuai dengan yang digunakan. Untuk mengganti pilihan jenis Arduino dan saluran port, pembaca bisa menggantinya di menu ***Tools***, di pilihan ***Board*** dan ***Port***. Untuk mengetahui saluran port yang sedang digunakan, pembaca dapat melihatnya di ***Device Manager***, di dalam kategori ***Ports (COM & LPT)***. Untuk membuka ***Device Manager***, pembaca dapat membukanya dengan cara mengetikkan kata "device manager" di kolom ***search programs & files***, di bagian pojok bawah ***Start Menu***.

2.3 Penggunaan Simulator Proteus

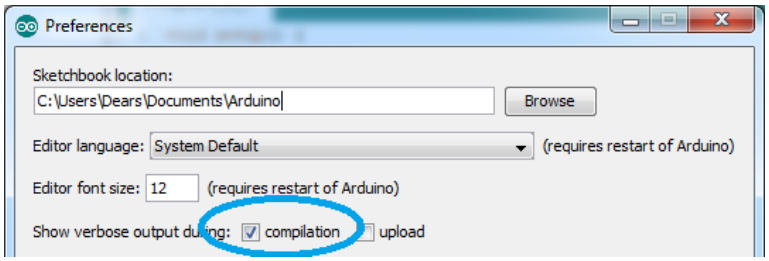
Apabila pembaca belum memiliki hardware Arduino, maka langkah terbaik adalah menggunakan software Proteus. Software Proteus dapat digunakan untuk mensimulasikan rangkaian elektronika, termasuk Arduino, secara tepat sama seperti unjuk kerja sebenarnya. Berikut langkah-langkah penggunaan Proteus untuk mensimulasikan rangkaian pembaca sinyal analog Gambar 2.1:

1. Buka software Proteus, dan buat rangkaian Gambar 2.1 di atas.
2. Agar rangkaian dapat bekerja, mikrokontroler ATmega328 harus diisi program. Pengisian program dilakukan dengan cara meng-klik 2 kali komponen ATmega328 tersebut, hingga muncul kotak Edit Component seperti terlihat pada Gambar 2.2 berikut ini.



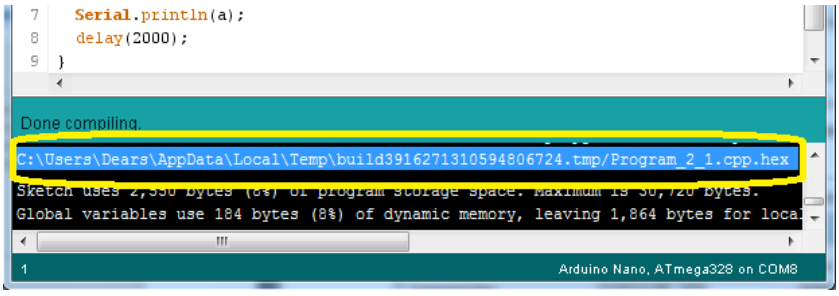
Gambar 2.2 Kotak Edit Component muncul ketika ATmega328 diklik 2 kali

3. Setelah kotak Edit Component muncul, pada kolom Program File, isi kolom tersebut dengan lokasi file kode mesin. Untuk mendapatkan lokasi file kode mesin tersebut, berikut langkah-langkahnya:
   a. Di software IDE Arduino, pilih menu File, dan pilih Preferences.
   b. Pada kotak Preferences yang muncul, pada bagian Show verbose output during, klik pada pilihan compilation, seperti ditunjukkan pada Gambar 2.3 berikut ini. Kemudian tutup kotak Preferences.



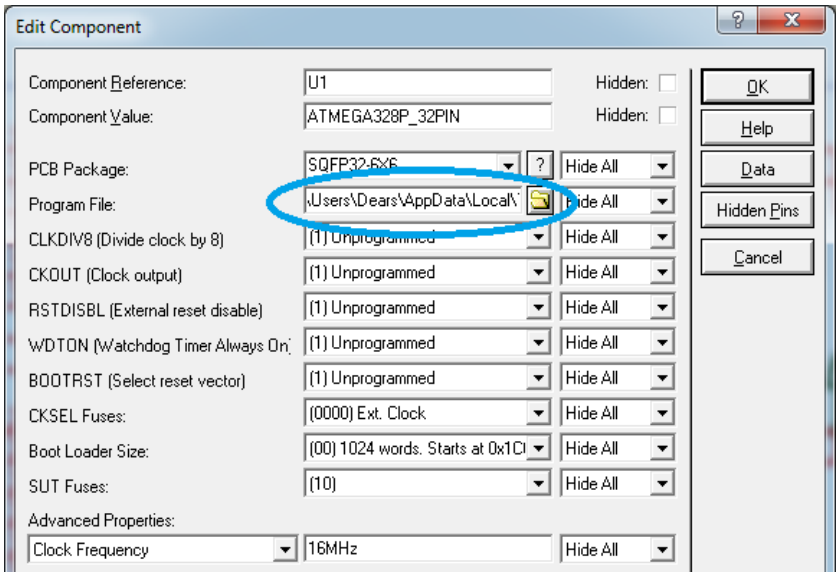
Gambar 2.3 Show verbose output untuk memunculkan file kode mesin

   c. Kompilasi Program_2_1.ino dengan menekan tombol Verify (tanda centang). Namun sebelum melakukan kompilasi, pastikan jenis Arduino yang dipilih adalah Arduino Nano. Untuk itu buka menu Tools, pilih Board, pilih Arduino Nano.
   d. Apabila kompilasi berhasil, ditandai dengan munculnya pesan Done Compiling, maka lokasi file kode mesin tersebut dapat ditemukan di bagian kotak informasi (kotak pesan berlatar belakang hitam), dengan posisi baris umumnya di baris ketiga dari bawah, dengan nama file berekstensi hex, seperti ditunjukkan pada Gambar 2.4 berikut ini.



Gambar 2.4 Lokasi file kode mesin ditemukan di bagian kotak informasi proses

4. Setelah lokasi file ditemukan, *copy* teks lokasi tersebut dengan cara pilih teks, dan tekan tombol Control dan tombol C bersamaan, kemudian tempelkan pada kolom **Program File** di kotak **Edit Component**.



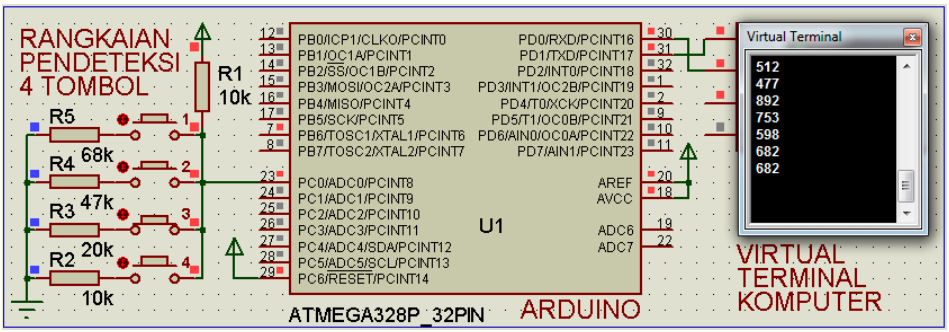
Gambar 2.5 Tempelkan lokasi file kode mesin pada kolom Program File

5. Atur isian kolom-kolom yang lain, sehingga sesuai seperti tabel berikut.

Tabel 2.2 Isian kotak Edit Component

Component Reference:	U1
Component Value:	ATMEGA328P_32PIN
PCB Package:	SQFP32-6X6
Program File:	Lokasi file program_2_1.hex
CLKDIV8:	[1] Unprogrammed
CKOUT:	[1] Unprogrammed
RSTDISBL:	[1] Unprogrammed
WDTON:	[1] Unprogrammed
BOOTRST:	[1] Unprogrammed
CKSEL Fuses:	[0000] Ext. Clock
BootLoader Size:	[00] 1024 words. Starts at 0x1C00
SUT Fuses:	[10]
Advances Properties	Clock Frequency: 16MHz

6. Jalankan simulasi Proteus dengan menekan tombol Play (bergambar panah ke kanan) yang berada di pojok kiri bawah. Gambar 2.6 berikut ini menunjukkan hasil simulasi Proteus untuk rangkaian pembaca sinyal analog. Lakukan penekanan tombol dan perhatikan perubahan nilai ADC yang ditampilkan pada kotak Virtual Terminal.



Gambar 2.6 Hasil simulasi Proteus untuk rangkaian pembaca sinyal analog

2.4 Soal Latihan

1. Lengkapi tabel berikut dengan mengisi nilai tegangan output dan nilai ADC untuk berbagai kombinasi penekanan tombol pada rangkaian Gambar 2.1, baik melalui perhitungan maupun melalui simulasi Proteus dengan nilai resistor R1 = 10kΩ, R2 = 10kΩ, R3 = 20kΩ, R4 = 47kΩ dan R5 = 68kΩ. Untuk nilai ADC, nilai ADC dapat dihitung dari persamaan:

Nilai ADC = nilai tegangan output x 1023/5.

No.	Kondisi tombol	Nilai Tegangan Output		Nilai ADC	
		Hasil hitungan	Hasil simulasi	Hasil hitungan	Hasil simulasi
1.	Tidak ada tombol yang ditekan.	...	...	...	...
2.	Tombol 1 ditekan.	...	...	...	...
3.	Tombol 2	...	...	...	...

No.	Kondisi tombol	Nilai Tegangan Output		Nilai ADC	
		Hasil hitungan	Hasil simulasi	Hasil hitungan	Hasil simulasi
	ditekan.				
4.	Tombol 3 ditekan.	...	...	...	...
5.	Tombol 4 ditekan.	...	...	...	...
6.	Tombol 1 dan 2 ditekan.	...	...	...	...
7.	Tombol 2 dan 3 ditekan.	...	...	...	...
8.	Tombol 3 dan 4 ditekan.	...	...	...	...
9.	Tombol 4 dan 1 ditekan.	...	...	...	...
10.	Tombol 1 dan 3 ditekan.	...	...	...	...
11.	Tombol 2 dan 4 ditekan.	...	...	...	...
12.	Tombol 1, 2 & 3 ditekan.	...	...	...	...
13.	Tombol 1, 2 & 4 ditekan.	...	...	...	...
14.	Tombol 1, 3 & 4 ditekan.	...	...	...	...
15.	Tombol 2, 3 & 4 ditekan.	...	...	...	...
16.	Semua tombol ditekan	...	...	...	...

2. Berdasarkan nilai ADC pada Tabel di Soal No. 1 di atas, terjemahkanlah kondisi tombol untuk nilai ADC pada Tabel berikut ini.

No.	Nilai ADC	Kondisi Tombol

No.	Nilai ADC	Kondisi Tombol
1.	358	...
2.	376	...
3.	387	...
4.	409	...
5.	434	...
6.	462	...
7.	477	...
8.	512	...
9.	550	...
10.	597	...
11.	622	...
12.	681	...
13.	753	...
14.	843	...
15.	892	...
16.	1023	...

3. Menurut Anda, apa yang terjadi pada rangkaian ketika nilai ADC yang ditampilkan di Serial Monitor adalah angka 0. Sebutkan kemungkinan-kemungkinan yang terjadi pada rangkaian ketika hal tersebut terjadi.

BAB 3
PENAMPIL DATA

Target Materi:

- Menggunakan software Proteus untuk mensimulasikan rangkaian sensor arah angin.
- Menampilkan data sensor arah angin dalam bentuk teks dan gambar di komputer dengan bantuan LabVIEW.
- Mensimulasikan komunikasi serial antara Arduino dengan LabVIEW menggunakan Proteus.

Persoalan:

Simulasikan rangkaian sensor arah angin yang datanya dapat ditampilkan di komputer, baik dalam bentuk teks maupun gambar.

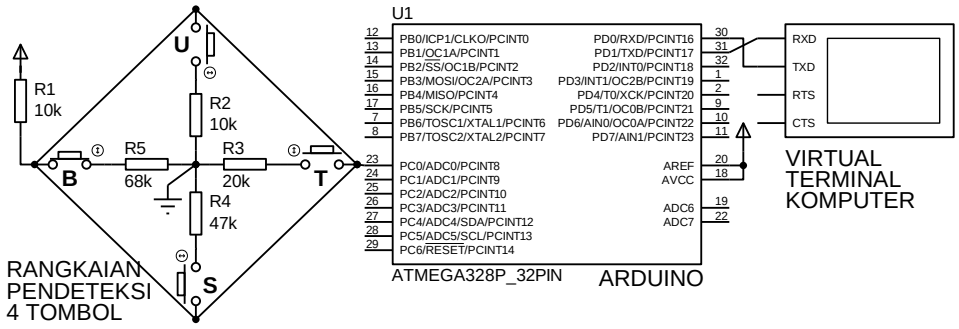
Uraian Materi:

3.1 Simulasi Rangkaian Sensor Arah Angin

Diinginkan sebuah rangkaian sensor arah angin, dapat disimulasikan dan ditampilkan datanya di Komputer. Berikut langkah-langkahnya:

1. Dengan menggunakan software Proteus, modifikasi rangkaian pendeteksi 4 tombol di Gambar 2.1 sedemikian rupa, sehingga keempat tombol ditempatkan dalam posisi seperti ditunjukkan pada Gambar 3.1.
2. Beri nama tombol sesuai posisi arah mata angin, yaitu U (utara), T (timur), S (selatan) dan B (barat), seperti ditunjukkan pada Gambar 3.1.

3. Pastikan bahwa file Hex hasil kompilasi Program_2_1.ino masih ada sesuai alamat di kolom Program File. Hal ini perlu dilakukan karena file Hex yang dihasilkan oleh software IDE Arduino ini bersifat sementara (*temporary*), sehingga ketika ada program lain yang dikompilasi di software IDE Arduino, maka file Hex Program_2_1.ino tersebut akan terhapus. Apabila sudah terhapus - ditandai dengan munculnya pesan Error ketika Proteus dijalankan - maka kompilasi lagi program tersebut di software IDE Arduino. Setelah kompilasi berhasil, jalankan simulasi Proteus dan amati nilai ADC yang dihasilkan untuk kondisi tombol yang ditekan mengikuti Tabel 3.1, dan bandingkan nilai ADC yang dihasilkan dari simulasi dengan nilai ADC di Tabel 3.1 tersebut.



Gambar 3.1 Keempat tombol menunjuk ke Utara, Timur, Selatan dan Barat

Tabel 3.1 Pengamatan Nilai ADC untuk 8 kondisi tombol

No.	Tombol yang ditekan	Indikasi Arah	Nilai ADC
1.	Tombol U ditekan	Utara	512
2.	Tombol U dan T ditekan	Timur Laut	410
3.	Tombol T ditekan	Timur	682
4.	Tombol T dan S ditekan	Tenggara	598
5.	Tombol S ditekan	Selatan	844
6.	Tombol S dan B ditekan	Barat Daya	753
7.	Tombol B ditekan	Barat	892
8.	Tombol B dan U ditekan	Barat Laut	477

Catatan: Perhatikan, apabila keempat tombol pada rangkaian Gambar 3.1 di atas diganti dengan sensor yang bisa diaktifkan tanpa kontak langsung, seperti contohnya sensor Reed Switch yang dapat menyambung ketika ada medan magnet di dekatnya dan putus ketika tidak ada medan magnet, maka jika sebuah poros yang ditempeli magnet, diletakkan di tengah-tengah empat Reed Switch, dan poros tersebut dapat berputar ketika tertiup angin, maka pembaca telah dapat membuat rangkaian sensor arah angin, yang dapat menunjukkan 8 buah arah angin, yaitu arah Utara, Timur Laut, Timur, Tenggara, Selatan, Barat Daya, Barat dan Barat Laut. Lebih lanjut mengenai pembuatan hardware rangkaian sensor arah angin dapat pembaca pelajari di Bab 6. Untuk sementara, di Bab 3 ini, pembaca diminta membayangkan bahwa apabila tombol U ditekan, maka angin sedang bertiup ke arah Utara, begitu pula ketika tombol U dan T ditekan, maka anggap bahwa angin sedang bertiup ke arah Timur Laut, dan seterusnya.

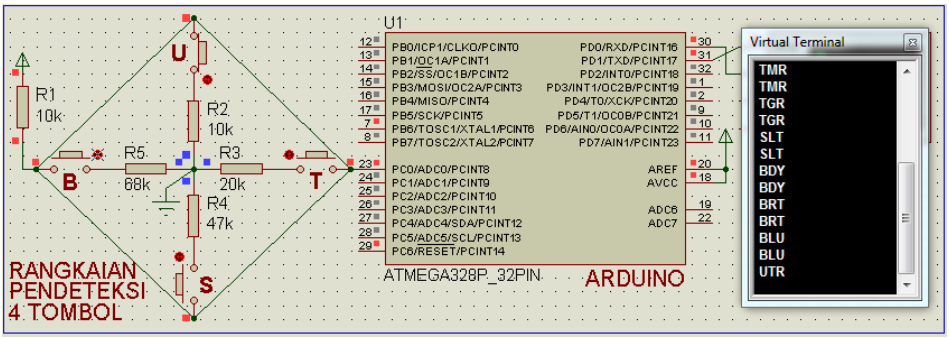
4. Dari Tabel 3.1 di atas, diperoleh hubungan antara arah angin dengan nilai ADC. Berdasarkan Tabel 3.1 tersebut, maka dapat dibuat program untuk menerjemahkan nilai ADC yang dihasilkan oleh rangkaian pendeteksi 4 tombol menjadi arah angin yang ditampilkan di Komputer melalui Serial Monitor IDE Arduino. Jadi ketika tombol U ditekan, nilai ADC yang dihasilkan adalah 512, maka di Serial Monitor akan menampilkan UTR. Begitu pula untuk penekanan tombol mengikuti Tabel 3.1, yang menghasilkan nilai ADC berturut-turut sebesar 410, 682, 598, 844, 753, 892, 477, maka seharusnya di Serial Monitor akan menampilkan arah angin berturut-turut TLU, TMR, TGR, SLT, BDY, BRT, BLU. Tampilan arah angin sengaja dibuat 3 huruf untuk menyeragamkan jumlah data. Berikut ini program untuk menerjemahkan nilai ADC menjadi arah angin 3 huruf.

Program_3_1.ino

```
1.  void setup() {
2.    Serial.begin(9600);
3.  }
4.  void loop() {
5.    int a = analogRead(A0);
6.    if (a == 512) Serial.println("UTR");
7.    if (a == 410) Serial.println("TLU");
```

```
8.    if (a == 682) Serial.println("TMR");
9.    if (a == 598) Serial.println("TGR");
10.   if (a == 844) Serial.println("SLT");
11.   if (a == 753) Serial.println("BDY");
12.   if (a == 892) Serial.println("BRT");
13.   if (a == 477) Serial.println("BLU");
14.   delay(1000);
15. }
```

5. Dengan menggunakan Proteus, simulasikan Program_3_1.ino pada rangkaian Gambar 3.1 di atas. Kompilasi Program_3_1.ino tersebut pada software IDE Arduino, kemudian masukkan lokasi file Hex hasil kompilasi pada kolom Program File ATmega328 (lihat Gambar 2.5). Setelah itu jalankan Proteus, dan amati data arah angin yang ditampilkan di Serial Monitor setiap kali tombol tertentu ditekan, seperti ditunjukkan pada Gambar 3.2 berikut ini. Bandingkan hasilnya dengan Tabel 3.1.

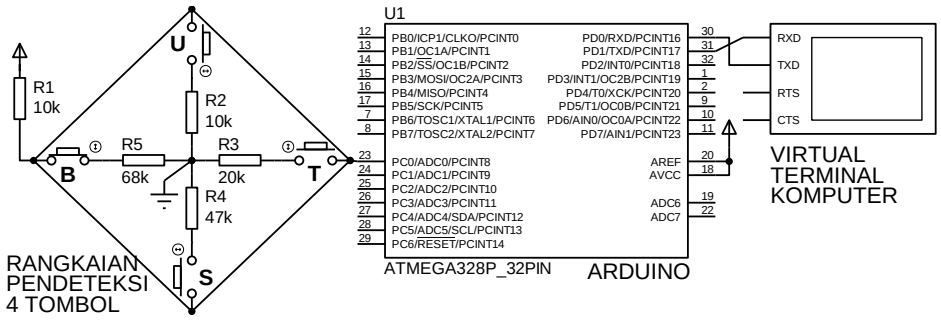


Gambar 3.2 Hasil simulasi Proteus untuk rangkaian sensor arah angin

3.2 Simulasi Proteus dan Serial Monitor IDE Arduino

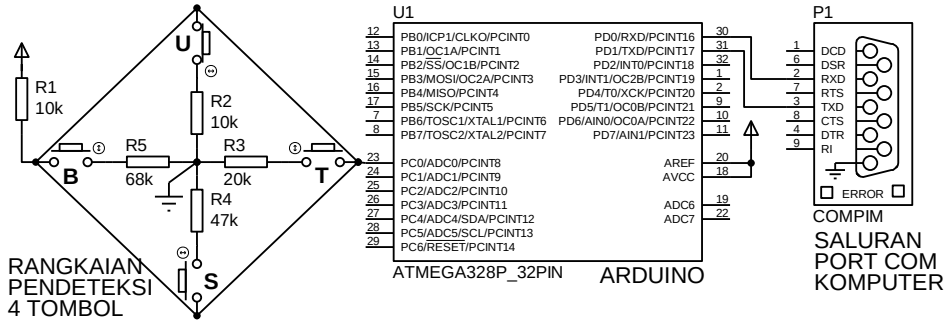
Software Proteus tidak hanya mampu mensimulasikan cara kerja rangkaian dan mikrokontroler di dalam softwarenya, tetapi juga mampu mensimulasikan komunikasi data dari mikrokontroler dengan software lain di luar Proteus. Singkatnya, diinginkan Virtual Terminal yang menampilkan data arah angin di Gambar 3.2 di atas digantikan dengan Serial Monitor di software IDE Arduino. Jadi nantinya, data arah angin hasil simulasi Proteus di atas ditampilkan di Serial Monitor IDE Arduino. Untuk membuat hal itu, berikut ini langkah-langkahnya:

1. Buka kembali rangkaian Gambar 3.1 di Proteus.



Gambar 3.3 Membuka kembali Gambar 3.1

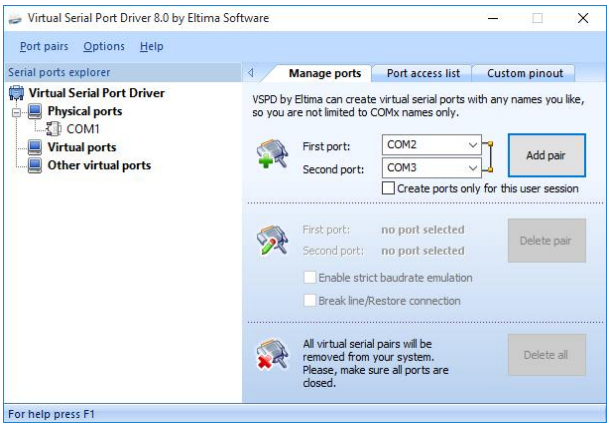
2. Ganti komponen Virtual Terminal dengan COMPIM. COMPIM adalah model interface fisik di Proteus untuk menghadirkan serial port secara riil. Hubungkan kaki RX dan TX Arduino dengan RX dan TX COMPIM. Berbeda dengan sambungan Virtual Terminal yang bersilangan (*cross*), sambungan antara Arduino dengan COMPIM lurus (*straight*), yaitu RX Arduino terhubung dengan RX COMPIM, dan TX Arduino dengan TX COMPIM.



Gambar 3.4 Komponen Virtual Terminal diganti dengan COMPIM

3. Penggunaan COMPIM membutuhkan Emulator Serial Port. Untuk itu instal software VSPD (*Virtual Serial Port Driver*) buatan Eltima. Pembaca bisa mendapatkan software VSPD ini di dalam CD Pendukung. Walaupun software VSPD ini hanya *shareware* (waktu penggunaan dibatasi selama 14 hari saja), namun konfigurasi emulator serial port yang dihasilkan oleh software ini akan berfungsi seterusnya. Berikut cara konfigurasinya:

a. Buka software VSPD:

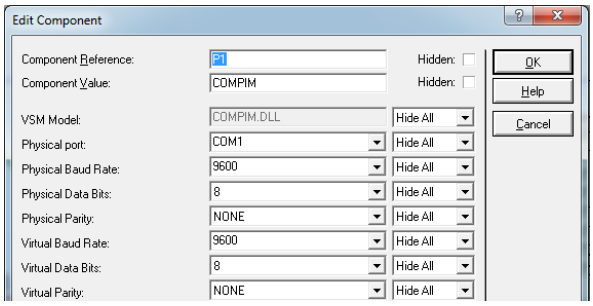


Gambar 3.5 Konfigurasi Emulator Serial Port dengan VSPD

b. Pada Tab Manage Ports, pilih 2 buah COM yang berurutan yang belum digunakan untuk First Port dan Second Port. Contoh, bila COM1 dan COM2 belum digunakan, maka atur First Port pada COM1 dan Second Port pada COM2. Setelah itu tekan tombol Add pair.

c. Konfigurasi selesai. Tutup software VSPD. Untuk seterusnya, kedua COM tersebut dapat digunakan untuk simulasi komunikasi serial antara 2 software. Software pertama akan menggunakan COM yang satu, dan software kedua akan menggunakan COM pasangannya.

2. Apabila COM1 dan COM2 yang dipasangkan, maka gunakan COM1 untuk software Proteus dan COM2 untuk software IDE Arduino. Klik kanan komponen COMPIM hingga muncul kotak Edit Component berikut ini:

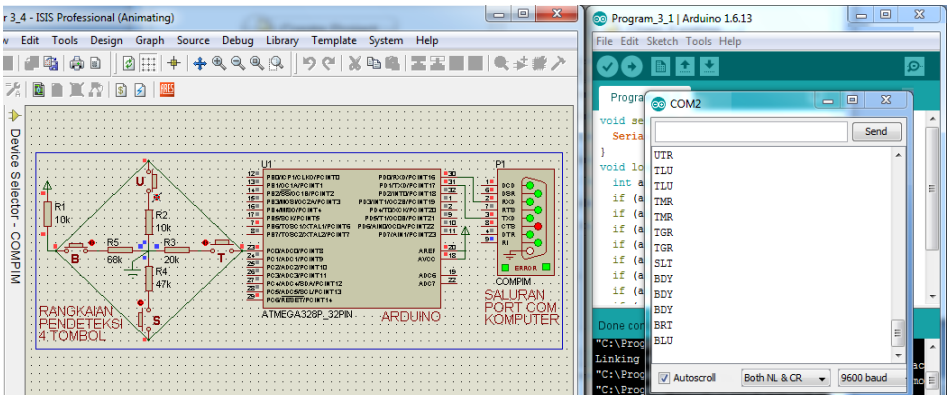


Gambar 3.6 Edit Component COMPIM, isi COM1 dan isi Baud Rate 9600

Tabel 3.2 Isian kotak Edit Component COMPIM

Component Reference:	P1
Component Value:	COMPIM
VSM Model:	COMPIM.dll
Physical Port:	COM1
Physical Baud Rate:	9600
Physical Data Bits:	8
Physical Parity:	NONE
Virtual Baud Rate:	9600
Virtual Data Bits:	8
Virtual Parity:	NONE

3. Setelah itu jalankan Proteus. Apabila terjadi error, ada kemungkinan file Hex hasil kompilasi tidak ditemukan. Untuk itu, lakukan lagi kompilasi Program 3_1.ino dengan menekan tombol Verify di software IDE Arduino. Setelah kompilasi selesai, pastikan lokasi file Hex yang muncul di kotak informasi, sesuai dengan yang ada di kolom Program File, atau agar lebih pasti, salin (*copy*) lokasi tersebut dan tempelkan di kolom Program File. Kemudian jalankan kembali Proteus, seharusnya tidak muncul error.
4. Sementara Proteus masih berjalan, di software IDE Arduino, pada menu Tools, pada pilihan Port, pilih saluran port pasangan, dalam hal ini COM2 (lihat langkah no. 2). Setelah itu klik tombol bergambar kaca pembesar untuk membuka Serial Monitor. Seharusnya di Serial Monitor akan muncul data arah angin dari Proteus, seperti ditunjukkan gambar berikut.



Gambar 3.7 Data arah angin dari simulasi Proteus ditampilkan di Serial Monitor

3.3 Simulasi Proteus dan LabVIEW

Apabila Sub Bab 3.2 di atas menampilkan simulasi data arah angin Proteus di Serial Monitor IDE Arduino, maka di Sub Bab 3.3 ini simulasi data arah angin Proteus tersebut ditampilkan di LabVIEW. Namun sebelum mengetahui langkah pembuatannya, pembaca perlu mengenal LabVIEW dulu.

Apa itu LabVIEW? Mungkin itu pertanyaan yang muncul khususnya bila pembaca belum pernah menggunakan LabVIEW. LabVIEW adalah sebuah software pemrograman komputer, dengan bahasa pemrograman berbentuk gambar. LabVIEW merupakan singkatan dari *Laboratory Virtual Instrument Engineering Workbench*. Dari namanya dapat diketahui bahwa LabVIEW dibuat untuk keperluan teknik instrumentasi dan laboratorium virtual.

Mengapa menggunakan LabVIEW? Software LabVIEW telah dibuat sedemikian rupa untuk membuat pemrograman menjadi mudah dan cepat karena berbentuk gambar visual, dengan instruksi berupa icon. LabVIEW juga telah dilengkapi dengan banyak *library* atau fungsi untuk keperluan akuisisi data, pengolahan sinyal, filter, matematika, statisika, analisis data, objek grafis, dll. LabVIEW juga telah dilengkapi dengan driver untuk interface ke alat-alat ukur, alat-alat instrumentasi, kamera, mikrokontroler dan alat kontrol lainnya.

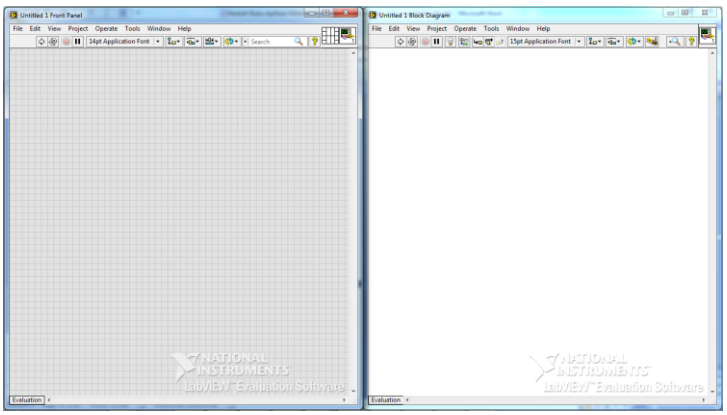
Software LabVIEW di sini digunakan untuk menggantikan Serial Monitor. Data yang bisa ditampilkan di Serial Monitor, akan dapat ditampilkan di LabVIEW juga. Di LabVIEW, data tersebut tidak hanya ditampilkan dalam bentuk teks, tetapi juga bisa ditampilkan dalam bentuk gambar, tabel dan grafik, sehingga lebih informatif dan menarik, bahkan data tersebut dapat disimpan secara otomatis berkala.

Agar pembaca lebih mengenal LabVIEW, akan lebih baik bila langsung menggunakannya. Berikut langkah-langkah menampilkan data Sensor Arah Angin di LabVIEW dalam bentuk teks dan gambar:

1. Instal software LabVIEW. Pembaca dapat mendownload software LabVIEW versi evaluasi di alamat www.ni.com/download-labview. Jalankan installer hasil download tersebut. Ketika diminta untuk mengisi kode lisensi, abaikan dan tekan tombol Next. Kemudian ketika muncul Instal Hardware Support for LabVIEW, pilih Decline Support.

Catatan: Hal yang menarik di sini adalah, sekalipun versi evaluasi, namun fitur yang dimiliki adalah sama seperti versi profesional. Tidak hanya itu, sekalipun masa penggunaan versi evaluasi dibatasi hanya 7 hari, namun masa penggunaan tersebut bisa diperpanjang dengan cara membuat waktu di komputer mundur ke belakang setiap kali membuka software LabVIEW, sehingga masa *expired* selalu tidak pernah terlewati. Namun demikian, saran penulis, sebaiknya pembaca membeli versi yang resmi, karena pembaca akan mendapat layanan dan suport yang lebih baik dari pihak National Instruments, perusahaan pembuat LabVIEW.

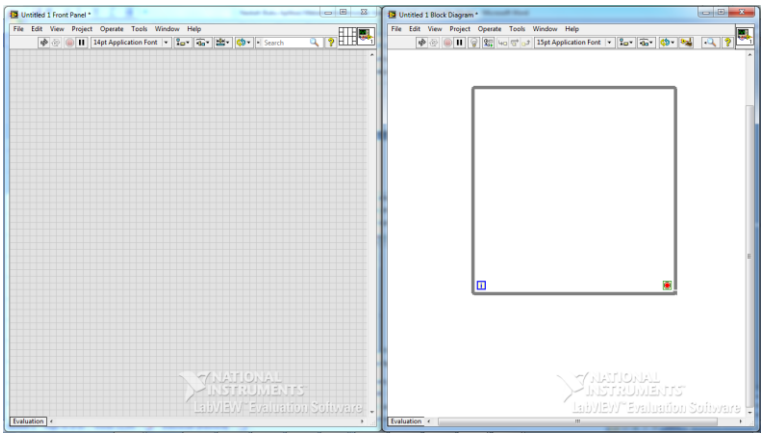
2. Instal software VISA, yaitu software driver untuk komunikasi serial. Pembaca dapat mendownload software driver VISA ini di alamat www.ni.com/download/ni-visa-15.0.1/5693/en.
3. Buka software LabVIEW, maka akan muncul 2 jendela, yaitu Front Panel dan Block Diagram. Front Panel adalah jendela tampilan program, sedangkan Block Diagram adalah tempat pembuatan program. Kode program berbentuk icon-icon, yang memiliki kaki input output yang bisa dihubungkan. Agar jendela Front Panel dan Block Diagram ditampilkan bersisian dalam satu layar, tekan tombol Control dan T bersamaan.



Gambar 3.8 Jendela Front Panel dan Block Diagram

4. Klik kanan jendela Block Diagram, maka akan muncul kotak Palet Functions. Pada kategori Programming, pilih Structures, kemudian pilih

While Loop. Untuk menempatkan While Loop pada jendela Block Diagram, klik dan tarik mouse membentuk kotak seperti gambar berikut.



Gambar 3.9 Menempatkan kotak While Loop di jendela Block Diagram

Catatan: Apabila program Arduino menggunakan void setup() untuk menempatkan program yang hanya sekali dijalankan di awal, dan void loop() untuk menempatkan program yang terus-menerus dijalankan, maka LabVIEW menggunakan kotak While Loop. Semua kode program atau icon yang ditempatkan di luar While Loop akan dijalankan sekali, sedangkan semua icon di dalam While Loop akan dijalankan terus-menerus.

5. Program LabVIEW untuk menerima data dari Arduino yang dikirim melalui komunikasi serial dapat dibuat dengan 3 icon, yaitu: VISA Configure Serial Port, VISA Read dan Visa Close. Berikut ini keterangan masing-masing:
   - Icon VISA Configure Serial Port mirip seperti instruksi Serial.begin() di Arduino, yaitu digunakan untuk mengatur komunikasi serial. Secara default, kecepatan komunikasi yang digunakan adalah 9600 bps. Icon VISA Configure Serial Port ini hanya perlu dijalankan sekali di awal.
   - Icon VISA Read digunakan untuk membaca data yang masuk sebanyak jumlah byte yang ditentukan, kecuali ada karakter terminasi (*termination char*). Secara default, karakter terminasi VISA Read adalah karakter ASCII 10 atau New Line. Setiap kali VISA Read

menerima karakter ini, maka data berikutnya setelah karakter tersebut dianggap data yang baru, yang akan dibaca pada loop berikutnya. Sebagai contoh, ketika jumlah byte yang ditentukan adalah 4, ternyata pada byte atau karakter kedua adalah karakter terminasi, maka 2 byte berikutnya akan dibaca pada loop berikutnya. Karena digunakan untuk membaca data yang masuk, maka Icon VISA Read ini perlu dijalankan terus-menerus.

- Icon VISA Close digunakan untuk menutup komunikasi. Icon VISA Close ini hanya perlu dijalankan sekali di akhir program.

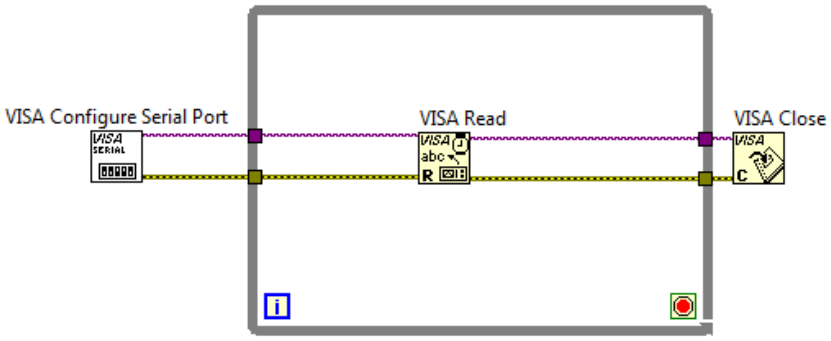
Catatan: Dari keterangan ketiga icon di atas, maka dapat diketahui bahwa hanya VISA Read saja yang harus ditempatkan di dalam kotak While Loop, sedangkan VISA Configure Serial Port dan VISA Close ditempatkan di luar While Loop.

6. Untuk membuat icon VISA Configure Serial Port dijalankan lebih dulu, kemudian baru VISA Read dan diakhiri dengan VISA Close, dapat dilakukan dengan menggunakan aliran data, yaitu garis data harus keluar dari VISA Configure Serial Port, diteruskan ke VISA Read, diteruskan ke VISA Close.

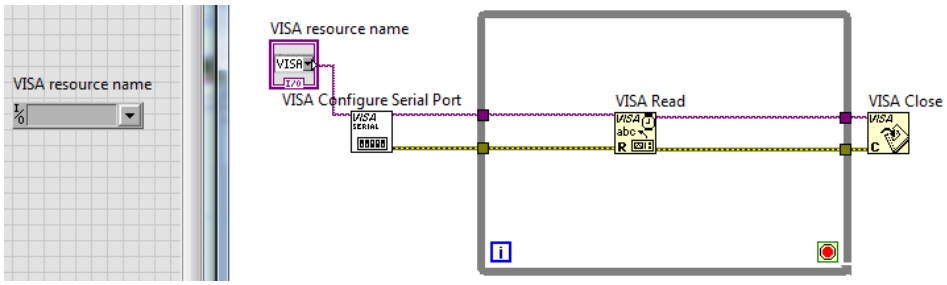
Catatan: LabVIEW menggunakan prinsip aliran data (*dataflow*) untuk menentukan urutan eksekusi. Aliran data ditunjukkan dengan garis. Icon yang mengeluarkan garis dari sisi kanannya menjadi sumber data, sehingga dijalankan lebih dulu, sedangkan icon yang di sisi kirinya mendapat garis, adalah penerima data, yang akan dijalankan setelah sumber data selesai dijalankan.

7. Sama seperti di Arduino, untuk melakukan komunikasi serial, pembaca perlu memilih saluran Port yang sedang digunakan. Pengaturan saluran Port ini ada di kaki input VISA Configure Serial Port, yaitu di kaki input paling atas, yang dinamai VISA resource name. Klik kanan kaki tersebut, dan pilih Create diikuti Control. Maka akan muncul icon VISA Resource Name di Block Diagram, dan juga di Front Panel muncul objek berupa kotak drop-down VISA Resource Name.

8. Dengan kotak drop-down ini, pengguna bisa memilih saluran Port yang sedang digunakan dari daftar, seperti terlihat pada Gambar 3.11.



Gambar 3.10 Aliran data dari VISA Configure → VISA Read → VISA Close

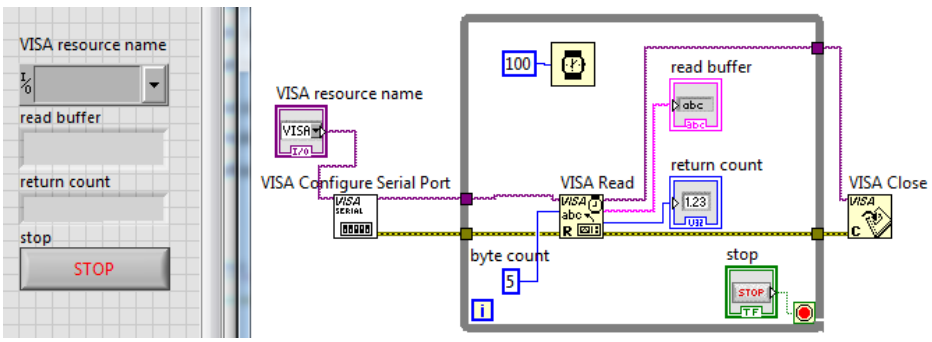


Gambar 3.11 Menambahkan VISA resource name untuk memilih saluran COM

Catatan: Perhatikan bahwa ketika pembaca memilih Create, maka tersedia 3 pilihan, yaitu Constant, Control dan Indicator. Berikut keterangan masing-masing:

- Apabila pembaca membutuhkan input nilai untuk icon yang isinya dapat diubah oleh pengguna, maka pembaca bisa memilih Control. Objek Control ini akan ditampilkan di jendela Front Panel.
- Apabila pembaca membutuhkan input nilai untuk icon yang isinya selalu tetap dan tidak dapat diubah oleh pengguna, maka pembaca bisa memilih Constant. Objek untuk icon Constant ini tidak ditampilkan di jendela Front Panel.
- Apabila pembaca membutuhkan output nilai untuk icon yang dapat terlihat oleh pengguna, maka pembaca bisa memilih Indicator. Objek Indicator ini akan ditampilkan di jendela Front Panel.

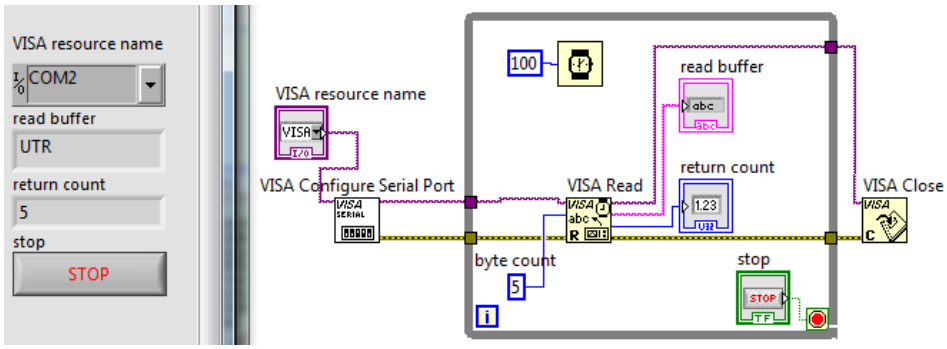
9. Program_3_1.ino di Sub Bab 3.1 di atas, akan mengirim data arah angin yang terdiri dari 3 huruf, yaitu: UTR, TLU, TMR, TGR, SLT, BDY, BRT, BLU. Karena instruksi pengiriman menggunakan Serial.println(), maka data 3 huruf tersebut diikuti dengan karakter *Carriage Return* (ASCII 13 atau '\r') dan *New Line* (ASCII 10 atau '\n'). Jadi total data yang dikirimkan adalah 5 karakter, atau 5 byte. Karena 5 byte, isi input byte count pada VISA Read dengan angka 5. Untuk itu klik kanan input byte count di VISA Read, pilih Create, pilih Constant. Ganti angka di kotak Constant dari 0 menjadi 5.
10. Untuk mengetahui isi data yang diterima, klik kanan pada kaki output read buffer VISA Read, pilih Create, pilih Indicator, maka akan muncul icon read buffer di Block Diagram dan kolom read buffer di Front Panel.
11. Untuk mengetahui jumlah byte yang diterima, klik kanan pada kaki output byte count VISA Read, pilih Create, pilih Indicator, maka akan muncul icon return count di Block Diagram dan kolom return count di Front Panel.
12. Agar program LabVIEW tidak menyita seluruh kemampuan dari CPU komputer, maka tambahkan delay waktu, dengan mengambil icon Wait (ms), yang ada di palet Functions, di kategori Programming, di kategori Timing. Klik kanan pada kaki input Wait (ms), pilih Create, pilih Constant. Ganti nilai angka di dalam kotak Constant dari 0 menjadi 100.
13. Terakhir, klik kanan pada kaki input terminal Loop Condition di pojok kiri bawah kotak While Loop, dan pilih Create, pilih Control, maka akan muncul icon stop di Block Diagram dan tombol stop di Front Panel. Gambar 3.12 berikut menunjukkan hasil akhir program:



Gambar 3.12 Program LabVIEW untuk menerima data serial dan menampilkannya

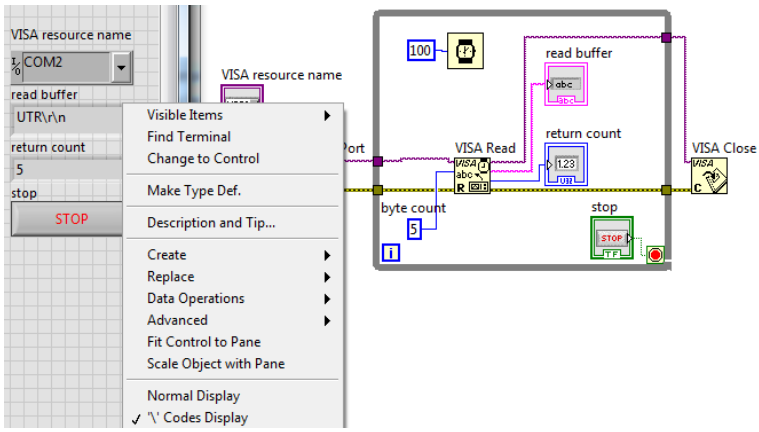
Catatan: Untuk merapikan tampilan objek di Front Panel, silahkan menggunakan tombol Align Object untuk membuat semua objek rata kanan, rata kiri atau rata tengah, dan tombol Distribute Object untuk membuat jarak antar objek menjadi seragam. Sedangkan untuk merapikan susunan icon di Block Diagram, silahkan menggunakan tombol Clean Up Diagram, maka susunan icon akan menjadi rapi.

14. Untuk menguji program LabVIEW pada Gambar 3.12 di atas, jalankan program LabVIEW tersebut bersama dengan simulasi rangkaian sensor arah angin Gambar 3.4 di Proteus. Gunakan sepasang COM virtual yang dibuat dengan VSPD. Apabila sepasang COM tersebut COM1 dan COM2, maka gunakan COM1 untuk COMPIM di Proteus dan COM2 untuk VISA Resource Name di LabVIEW.
15. Apabila tidak ada pesan error, baik di Proteus maupun LabVIEW, tekan tombol U di Proteus, maka seharusnya kotak read buffer di Front Panel menampilkan teks "UTR" seperti ditunjukkan dalam Gambar 3.13.



Gambar 3.13 Kolom read buffer menampilkan data UTR dan return count berisi 5

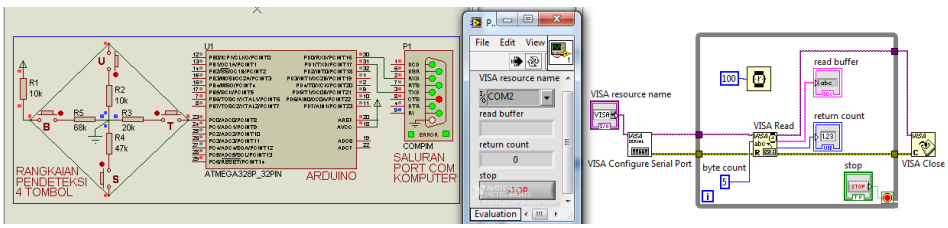
16. Perhatikan bahwa isi kolom return count di Front Panel adalah 5, yang berarti VISA Read membaca ada 5 byte data. Di kolom read buffer hanya ada 3 huruf saja, berarti ada 2 karakter ASCII yang tidak terlihat. Pembaca dapat melihat 2 karakter yang tak terlihat tersebut dengan mengganti mode kotak read buffer dari Normal Display menjadi '\'Codes Display. Caranya dengan menghentikan dulu program LabVIEW, klik kanan kotak read buffer dan pilih pilihan '\'Codes Display (lihat Gambar 3.14).



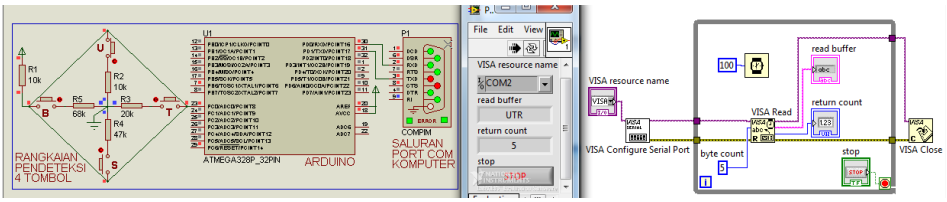
Gambar 3.14 Pilihan '\' Codes Display menampilkan 2 karakter yang tak terlihat

Catatan: Untuk menghentikan program LabVIEW, gunakan tombol Stop. Jangan sekali-kali menggunakan tombol Abort Execution yang ada di Toolbar, karena akan membuat semua program berhenti seketika, yang juga membuat VISA Close tidak sempat dijalankan. VISA Close harus dijalankan agar dapat menutup komunikasi. Apabila komunikasi belum tertutup, namun program telah berhenti, maka muncul pesan *error*. Pembaca perlu menutup LabVIEW dan membukanya kembali untuk menghilangkan pesan *error* tersebut.

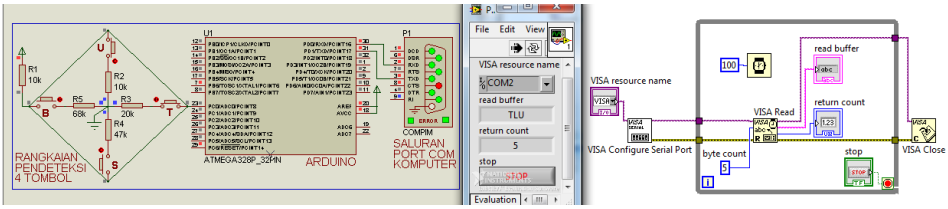
17. Jalankan kembali program LabVIEW, maka seharusnya akan muncul tambahan karakter \r\n seperti ditunjukkan dalam Gambar 3.14 di atas. Agar tampilan di kotak read buffer tidak membingungkan pengguna, ubah kembali mode '\'Codes Display menjadi Normal Display.
18. Berikut ini berturut-turut diperlihatkan tampilan LabVIEW untuk simulasi kombinasi penekanan tombol mengikuti Tabel 3.1.



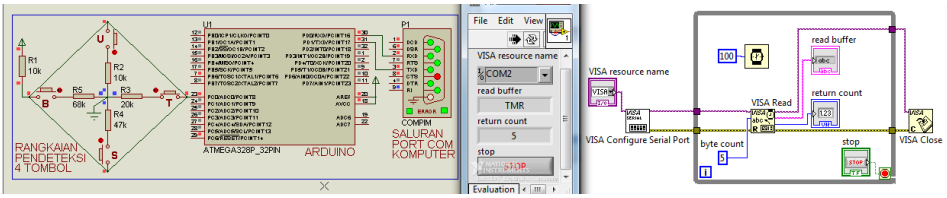
Gambar 3.15 Ketika tidak ada tombol yang ditekan, read buffer kosong



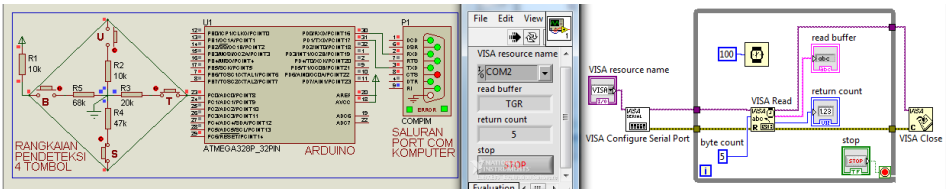
Gambar 3.16 Ketika tombol U ditekan, read buffer menampilkan UTR



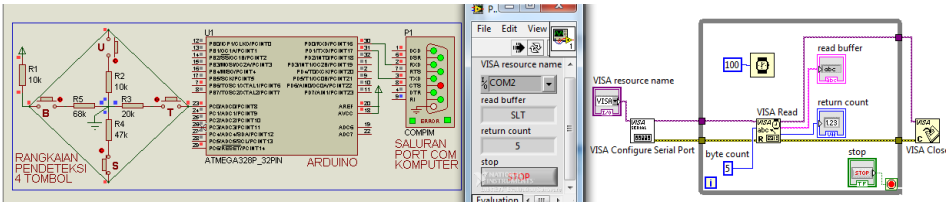
Gambar 3.17 Ketika tombol U dan T ditekan, read buffer menampilkan TLU



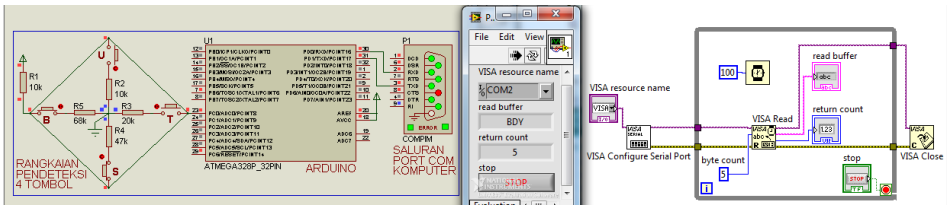
Gambar 3.18 Ketika tombol T ditekan, read buffer menampilkan TMR



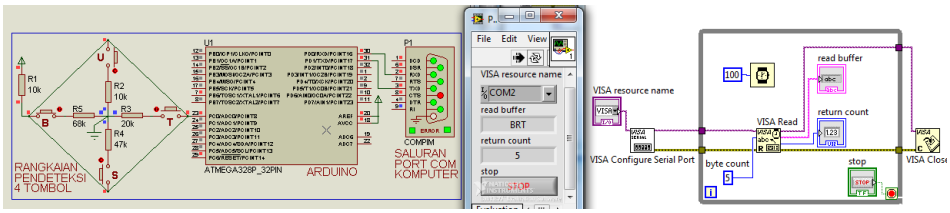
Gambar 3.19 Ketika tombol T dan S ditekan, read buffer menampilkan TGR



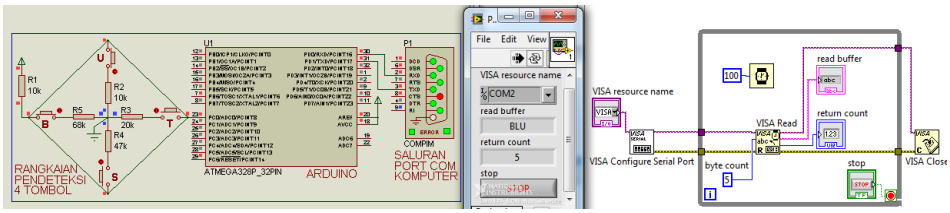
Gambar 3.20 Ketika tombol S ditekan, read buffer menampilkan SLT



Gambar 3.21 Ketika tombol S dan B ditekan, read buffer menampilkan BDY



Gambar 3.22 Ketika tombol B ditekan, read buffer menampilkan BRT



Gambar 3.23 Ketika tombol B dan U ditekan, read buffer menampilkan BLU

3.4 Menampilkan Gambar Arah Angin di LabVIEW

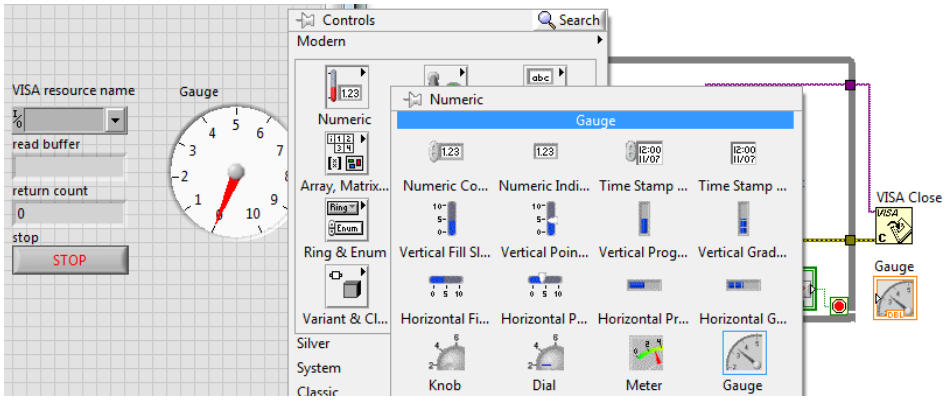
Setelah Sub Bab sebelumnya menyajikan bagaimana LabVIEW menampilkan data Sensor Arah Angin dalam bentuk teks, maka Sub Bab 3.4 ini akan menyajikan bagaimana menampilkan data Sensor Arah Angin dalam bentuk gambar. Ada 2 jenis gambar yang bisa digunakan di LabVIEW, yaitu:

a. Gambar objek dari Palet Controls LabVIEW.
b. Gambar objek dari luar dengan Picture Ring.

3.4.1 Gambar Objek dari Palet Controls LabVIEW

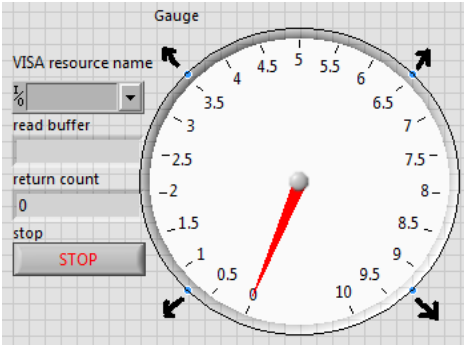
Berikut ini langkah-langkah menampilkan gambar data sensor arah angin menggunakan objek dari Palet Controls LabVIEW:

1. Sebelumnya, simpan dulu program LabVIEW yang telah dibuat pada Sub Bab 3.3 di atas dengan nama Program_3_3.vi. Setelah itu simpan lagi (*Save as*) Program_3_3.vi tersebut dengan nama Program_3_4_a.vi.
2. Klik kanan halaman Front Panel, pada kategori Modern, pilih Numeric, pilih Gauge, tempatkan di samping kotak read buffer di Front Panel.



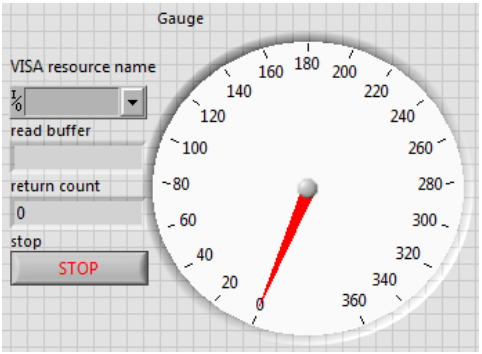
Gambar 3.24 Mengambil Gauge dari palet Controls, di Modern, di Numeric

3. Perbesar ukuran Gauge dengan menarik keluar titik tepi lingkaran.

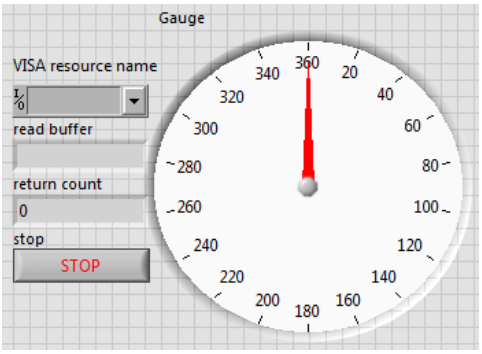


Gambar 3.25 Memperbesar Gauge dengan menarik keluar titik tepi lingkaran

4. Ubah angka 10 menjadi 360 seperti ditunjukkan pada Gambar 3.26.
5. Agar angka 360 bisa ditempatkan di puncak lingkaran, posisikan mouse pada garis strip 360 hingga pointer mouse menjadi garis lengkung. Kemudian seret strip 360 tersebut searah jarum jam hingga angka 360 berada di puncak lingkaran seperti ditunjukkan pada Gambar 3.27.



Gambar 3.26 Mengubah angka 10 menjadi 360



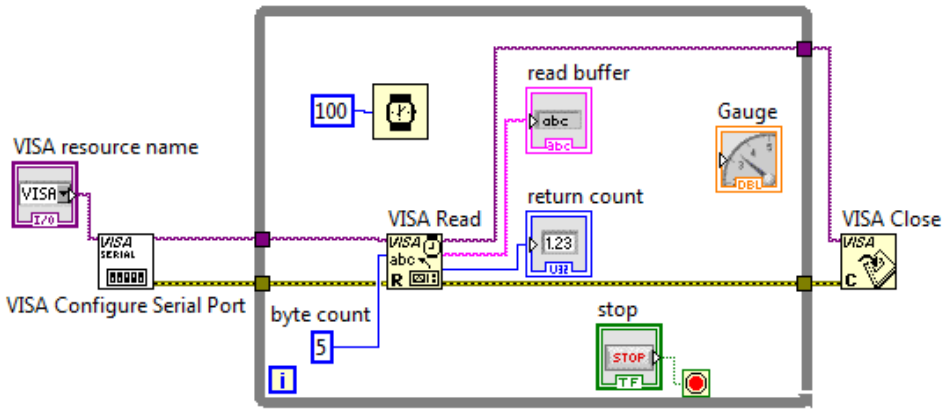
Gambar 3.27 Memposisikan angka 360 di puncak lingkaran Gauge

6. Berikutnya adalah menghubungkan data teks di read buffer dengan posisi jarum Gauge, seperti terlihat dalam tabel berikut:

Tabel 3.3 Hubungan Data Teks dengan Jarum Gauge

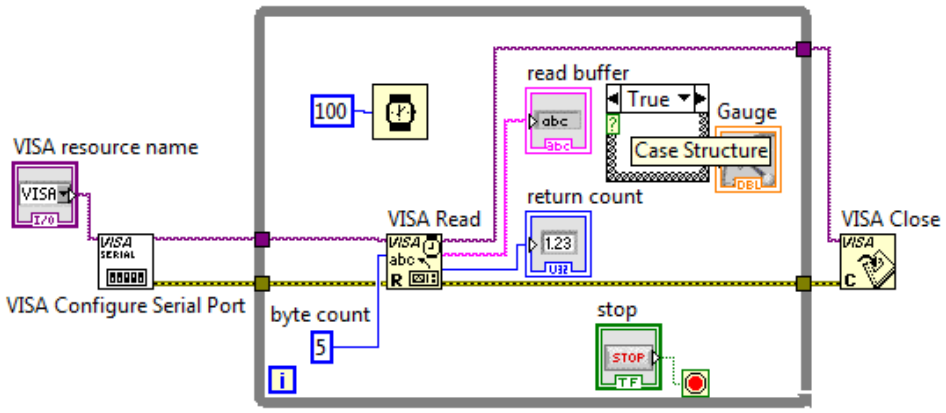
No	Data teks di read buffer	Keterangan arah	Posisi jarum di Gauge
1.	UTR	Utara	0
2.	TLU	Timur Laut	45
3.	TMR	Timur	90
4.	TGR	Tenggara	135
5.	SLT	Selatan	180
6.	BDY	Barat Daya	225
7.	BRT	Barat	270
8.	BLU	Barat Laut	315

7. Di Block Diagram, tempatkan icon Gauge di dalam kotak While Loop.



Gambar 3.28 Menempatkan icon Gauge di dalam kotak While Loop

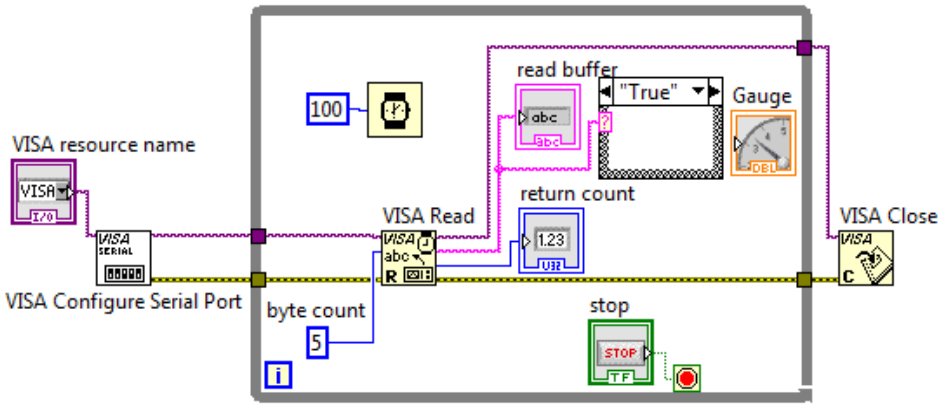
8. Klik kanan Block Diagram untuk memunculkan palet Functions. Pada palet Functions, pilih Structures, pilih Case Structure. Perhatikan pointer mouse berubah menjadi sebuah kotak. Gerakkan mouse membentuk kotak, maka sebuah Case Structure akan terbentuk (lihat Gambar 3.29).



Gambar 3.29 Menempatkan kotak Case Structure di antara read buffer dan Gauge

9. Hubungkan terminal Case Selector (bergambar tanda tanya) pada kotak Case Structure dengan garis data read buffer (lihat Gambar 3.30). Tampak kata True pada Selector Label berubah menjadi "True". Dengan diberi petik dua, menandakan bahwa kotak Case Structure memiliki input

bertipe data String. Secara default, kotak Case Structure memiliki input bertipe data Boolean, yaitu True dan False, yang juga tampak pada Selector Label (untuk berpindah ke bagian False, klik pada tanda panah di pojok kiri atau kanan). Namun ketika terminal Case Selector dihubungkan dengan garis data yang bertipe data selain Boolean, maka secara otomatis Selector Label Case Structure akan berubah sesuai garis data tersebut.

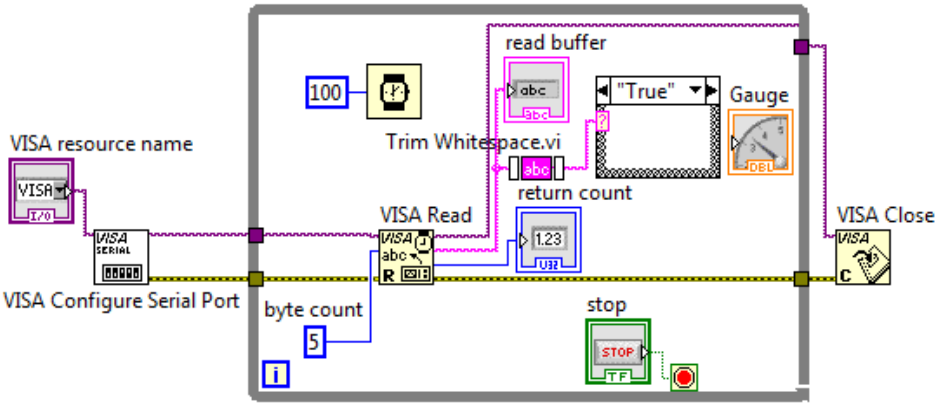


Gambar 3.30 Menghubungkan input Case Structure dengan garis data read buffer

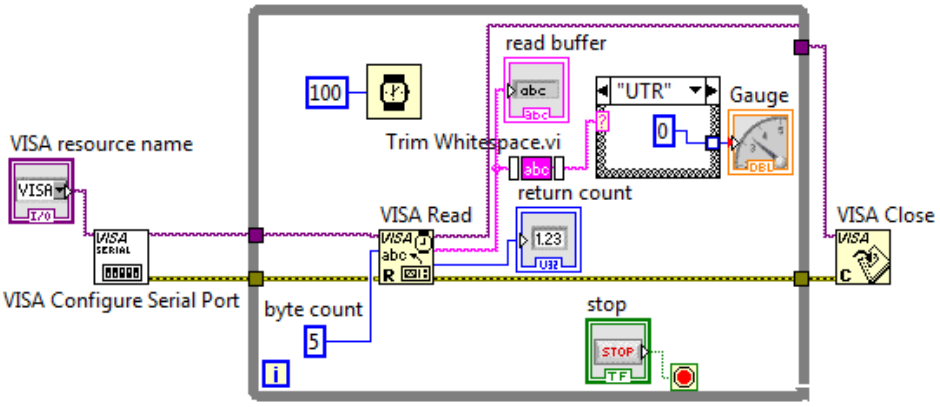
Catatan: Tipe data di LabVIEW ditunjukkan dengan warna. Warna hijau menunjukkan tipe data Boolean. Warna merah muda menunjukkan tipe data String. Warna biru menunjukkan tipe data Integer (bilangan bulat). Warna oranye menunjukkan tipe data Floating (bilangan pecahan).

10. Agar Selector Label pada Case Structure tidak perlu melibatkan karakter yang tak terlihat, yaitu Carriage Return (\r) dan New Line (\n), maka sisipkan icon Trim Whitespace pada garis data Case Structure dengan cara meng-klik kanan garis data tersebut, dan kemudian pilih Insert, pilih All palletes, pilih Programming, pilih String, pilih Trim Whitespace, seperti ditunjukkan pada Gambar 3.31.
11. Isi data di Case Structure sama dengan di read buffer, yaitu salah satu dari kedelapan data teks (UTR, TLU, TMR, TGR, SLT, BDY, BRT, atau BLU). Ketika isi data UTR, maka sesuai tabel di langkah no. 6, Gauge harusnya

berisi 0, ketika isi data TLU, maka Gauge harusnya berisi 45, dan seterusnya. Untuk itu, ubah kata “True” menjadi “UTR”, dan tempatkan sebuah angka 0 di kotak Case Structure. Angka 0 bisa diperoleh dari palet Functions di Block Diagram, pilih Programming, pilih Numeric, pilih Numeric Constant. Atau angka 0 bisa juga diperoleh dengan cara meng-klik kanan kaki input icon Gauge, dan memilih Create, pilih Constant, seperti ditunjukkan pada Gambar 3.32.



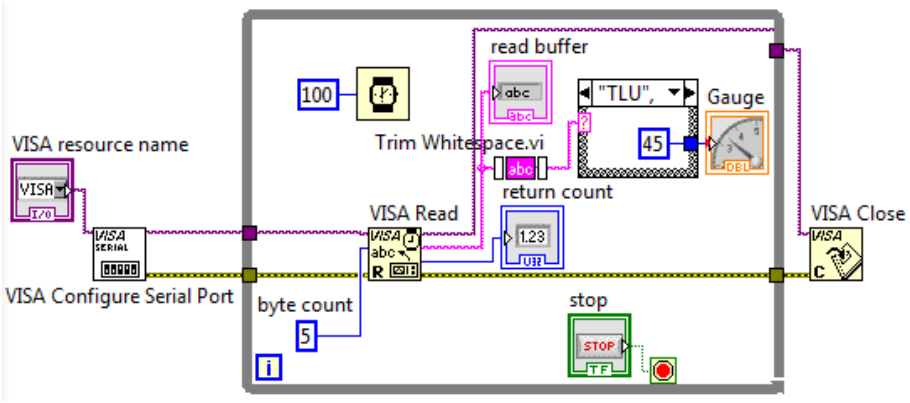
Gambar 3.31 Menyisipkan icon Trim Whitespace di garis data Case Structure



Gambar 3.32 Selector Label “UTR” berisi angka 0 untuk input Gauge

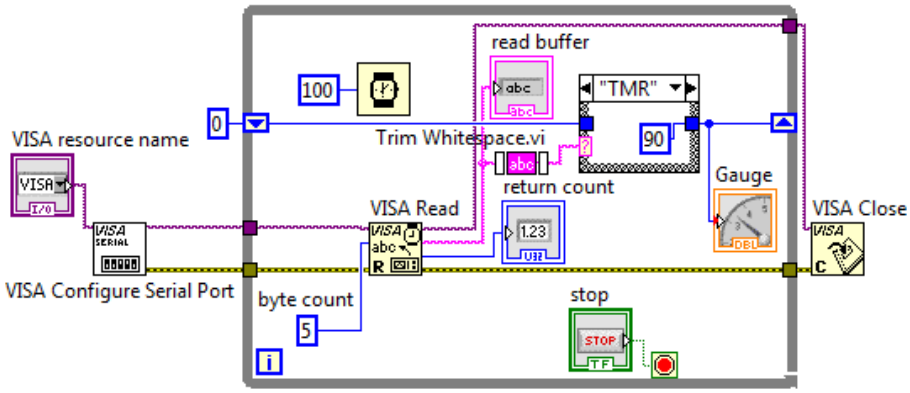
12. Klik tanda panah di pojok kiri Case Structure untuk pindah ke Selector Label “False, Default”. Ubah kata “False” menjadi “TLU”. Kemudian klik

kanan pada kotak kecil kosong di dinding kanan Case Structure, dan pilih Create, pilih Constant. Ganti angka 0 menjadi angka 45.



Gambar 3.33 Selector Label "TLU" berisi angka 45 untuk input Gauge

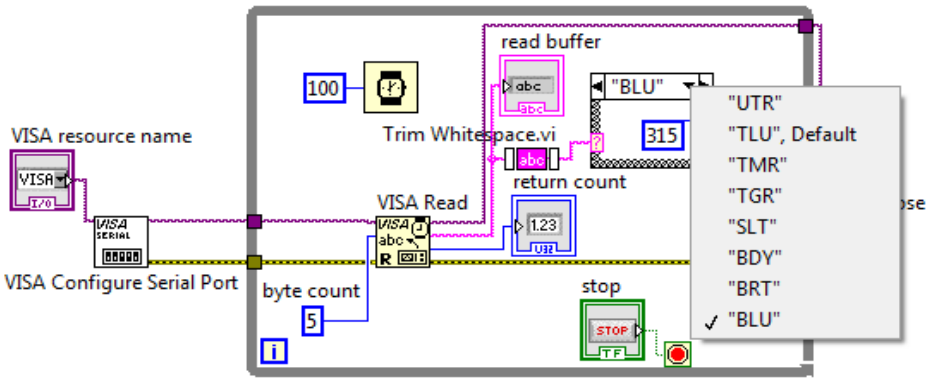
13. Klik kanan pada Selector Label, pilih Add Case After, maka muncul Case yang baru. Pada Selector Label untuk Case yang baru, ketikkan "TMR". Pada Case "TMR" ini, input Gauge harus mendapat angka 90. Untuk itu klik kanan kotak kecil kosong di dinding kanan Case Structure, dan pilih Create, pilih Constant. Ganti angka Numeric Constant, dari 0 menjadi 90.



Gambar 3.34 Selector Label "TMR" berisi angka 90 untuk input Gauge

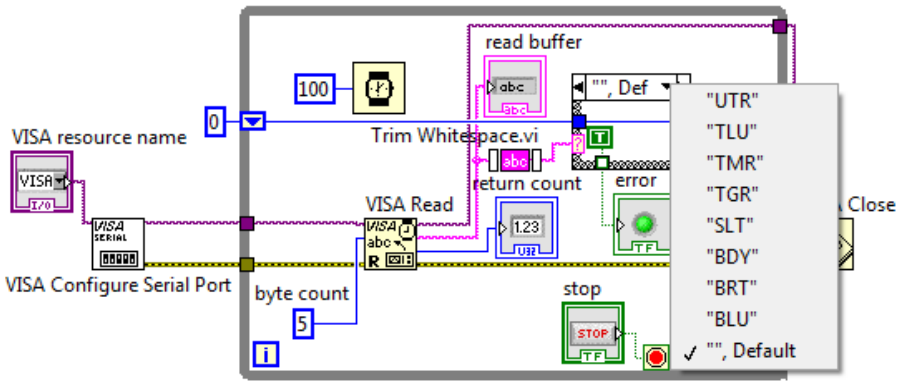
14. Ulangi langkah no. 13 di atas, sehingga muncul Case dengan Selector Label berturut-turut: "TGR", "SLT", "BDY", "BRT", "BLU". Isi setiap Case dengan angka Numeric Constant berturut-turut 135, 180, 225, 270, 315.

15. Kemudian klik pada tanda panah ke bawah di Selector Label, maka tampak daftar Case Selector Label seperti ditunjukkan gambar berikut ini.



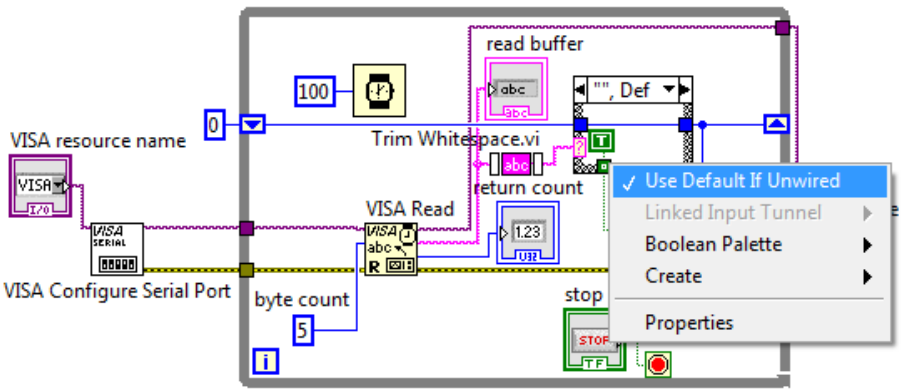
Gambar 3.35 Daftar Selector Label Case Structure

16. Perhatikan, bahwa pada Label "TLU", terdapat tambahan kata Default. Label Case dengan tambahan kata Default ini menjadi Case yang satu-satunya dipilih, apabila dari kedelapan Label yang tersedia, tidak satupun yang sama dengan data yang dibaca oleh VISA Read. Padahal, ada kemungkinan data yang dikirim Arduino adalah data kosong, yaitu ketika tidak ada tombol di rangkaian yang ditekan, atau ketika semua tombol di rangkaian ditekan. Apabila kata Default itu tetap berada di Label "TLU", maka ketika kondisi data kosong terjadi, LabVIEW akan menampilkan "TLU", padahal tombol U dan T tidak ditekan, sehingga mengakibatkan kesalahan pembacaan. Untuk itu tambahkan sebuah Case lagi, dengan meng-klik kanan Selector Label, pilih Add Case After, dan ketik sepasang tanda petik dua, "", yang artinya kosong. Kemudian pada Label yang baru ini, klik kanan, pilih "Make This The Default Case".
17. Kemudian agar ada tanda bahwa terjadi error, maka tambahkan di Front Panel, objek Round LED, yang diambil di Palet Controls, di kategori Boolean. Ganti nama Boolean menjadi error. Di Block Diagram, tempatkan icon error ini di dekat Case Structure. Pada kotak Case Default, tempatkan icon True Constant, yang diambil dari Palet Functions, kategori Boolean. Hubungkan icon error dengan True Constant seperti Gambar 3.36.

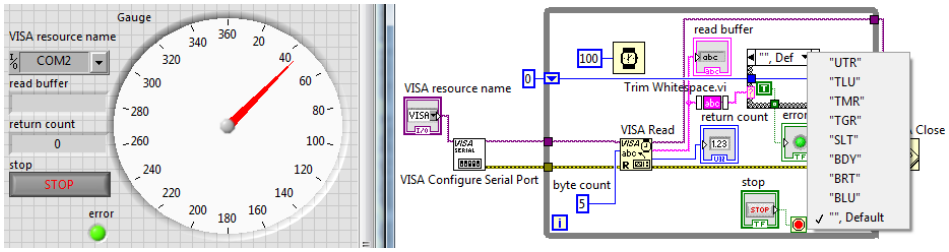


Gambar 3.36 Menambahkan indikator error, yang menyala ketika Case Default

18. Tampak pada Gambar 3.36, True Constant di dalam kotak Case Default dihubungkan dengan icon Round LED error. Perlu diketahui, bahwa setiap kali sebuah garis data keluar kotak Case, maka akan muncul kotak kecil kosong (warna putih) di dinding Case. Kotak kecil ini tetap berwarna putih atau tetap kosong bila tidak semua Case memiliki data yang terhubung dengan kotak tersebut. Apabila kotak kecil tersebut masih kosong, maka program akan error, sehingga tidak bisa dijalankan. Agar tidak perlu membuka setiap Case untuk membuat kotak kecil tersebut terisi, maka klik kanan pada kotak kecil tersebut, pilih "Use Default If Unwired". Dengan "Use Default If Unwired" ini, akan membuat semua Case selain Case Default, mengisi data False pada kotak kecil tersebut.

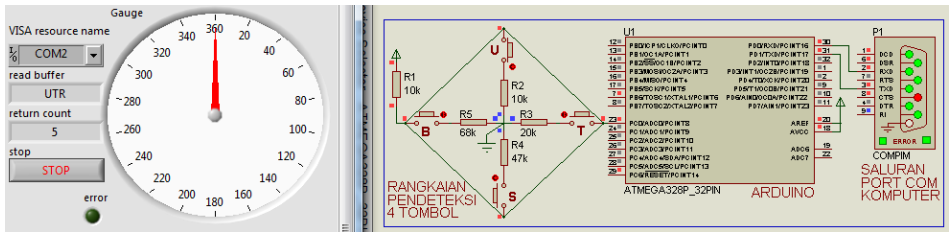


Gambar 3.37 Use Default If Unwired untuk mengisi False ke Case yang lain

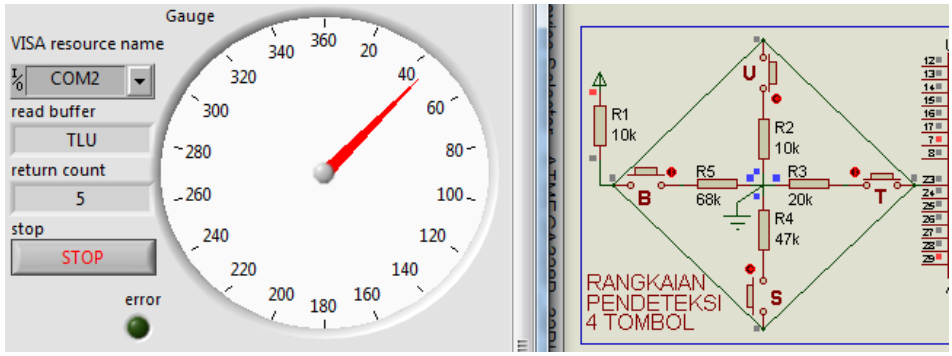


Gambar 3.38 Front Panel dan Block Diagram untuk program menampilkan Data Arah Angin menggunakan objek Gauge yang diambil dari Palet Control

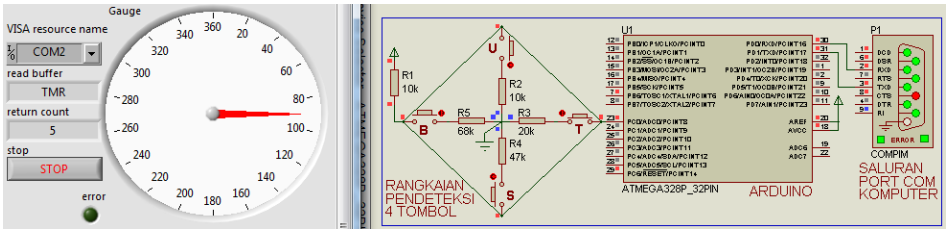
19. Jalankan simulasi Proteus dengan rangkaian seperti Gambar 3.4, dan program LabVIEW seperti Gambar 3.38 di atas, dan amati tampilan Gauge di LabVIEW setiap kali tombol di Proteus ditekan/dilepas. Berikut ini berturut-turut diperlihatkan tampilan Gauge LabVIEW untuk simulasi kombinasi penekanan tombol mengikuti Tabel 3.1, dan ketika kondisi error terjadi, yaitu ketika tidak ada tombol yang ditekan atau ketika semua tombol ditekan.



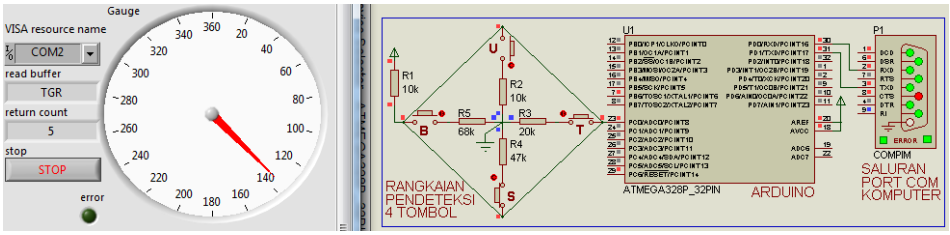
Gambar 3.39 Jarum Gauge menunjuk ke 360^0 atau 0^0 ketika tombol U ditekan



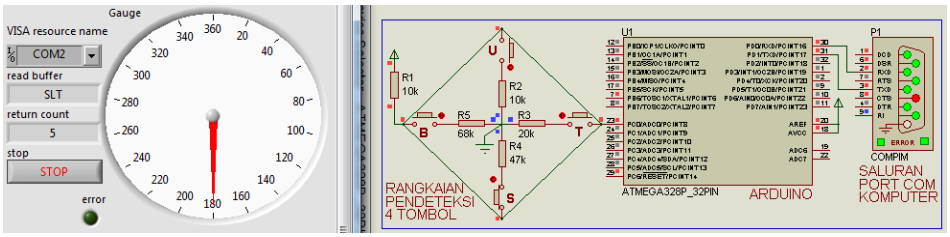
Gambar 3.40 Jarum Gauge menunjuk ke 45^0 ketika tombol U dan T ditekan



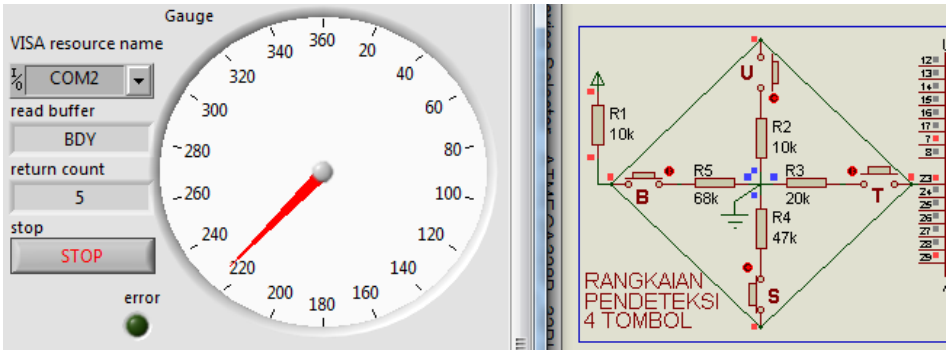
Gambar 3.41 Jarum Gauge menunjuk ke 90^0 ketika tombol T ditekan



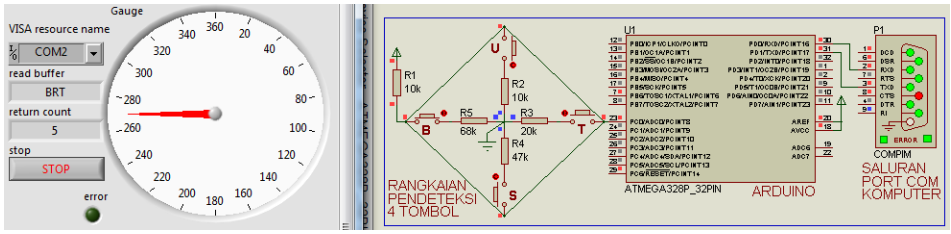
Gambar 3.42 Jarum Gauge menunjuk ke 135^0 ketika tombol T dan S ditekan



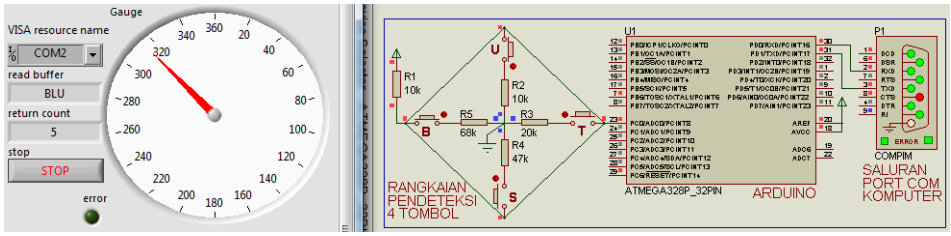
Gambar 3.43 Jarum Gauge menunjuk ke 180^0 ketika tombol S ditekan



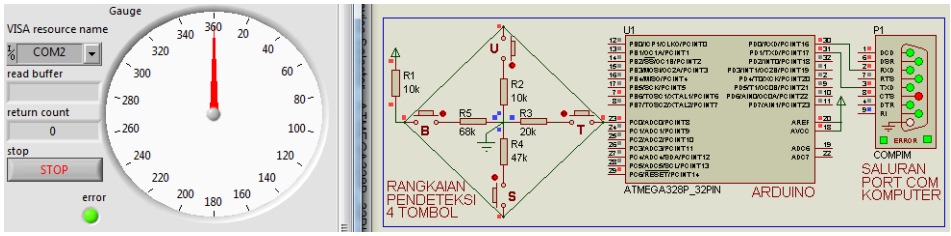
Gambar 3.44 Jarum Gauge menunjuk ke 225^0 ketika tombol S dan B ditekan



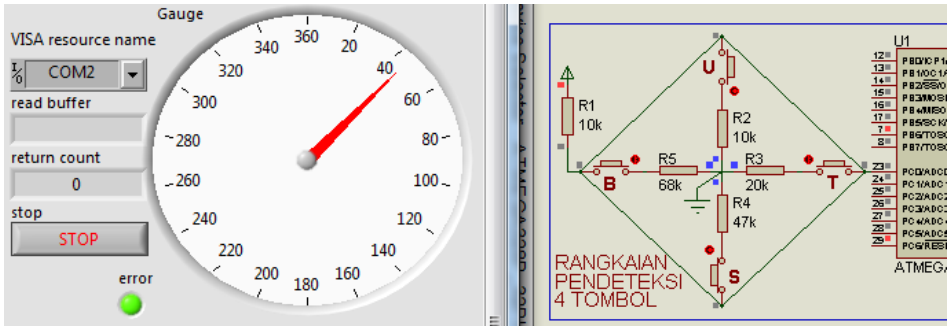
Gambar 3.45 Jarum Gauge menunjuk ke 270^0 ketika tombol B ditekan



Gambar 3.46 Jarum Gauge menunjuk ke 270^0 ketika tombol B ditekan



Gambar 3.47 Indikator LED Error menyala ketika tidak ada tombol yang ditekan. Nilai yang ditunjuk oleh Jarum Gauge diabaikan ketika LED Error menyala.

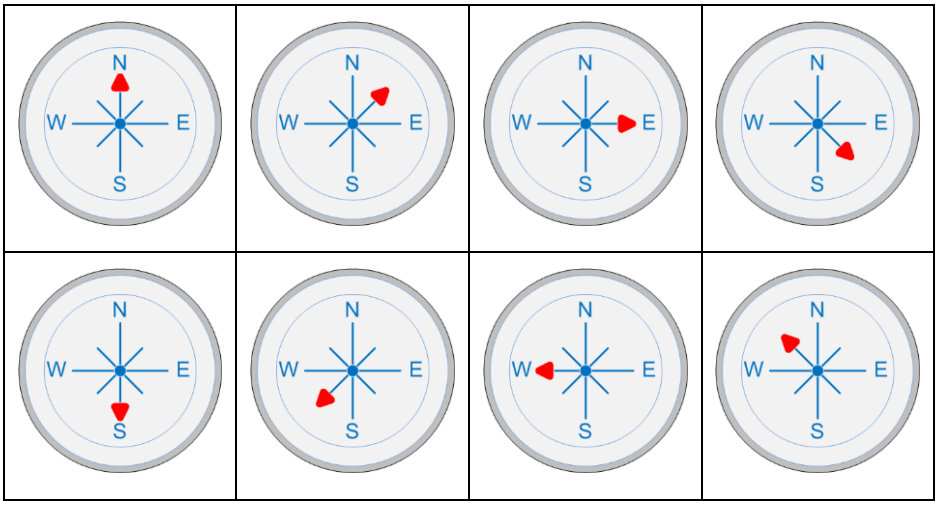


Gambar 3.48 Indikator LED Error menyala ketika semua tombol ditekan. Nilai yang ditunjuk oleh Jarum Gauge diabaikan ketika LED Error menyala.

3.4.2 Gambar objek dari luar dengan Picture Ring

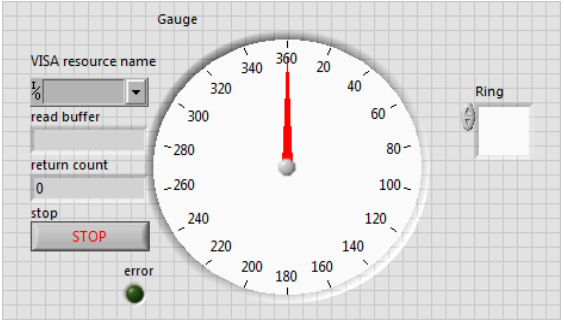
Apabila Sub Bab 3.4.1 menggunakan objek yang diambil dari Palet Control, berikut ini langkah-langkah menampilkan data arah angin dengan objek yang diambil dari gambar di luar Palet Control, yang diimport dan ditempatkan di Picture Ring:

1. Ambil gambar sensor arah angin di internet atau gunakan gambar berikut ini. Pembaca bisa mendapatkan file gambar sensor arah angin ini di CD pendukung dengan nama arah1.png, arah2.png, dan seterusnya):



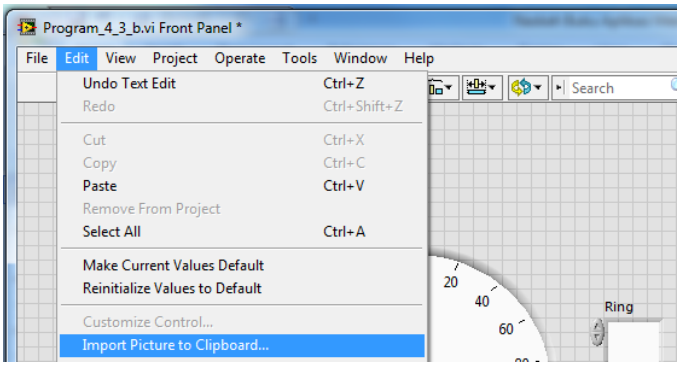
Gambar 3.49 Contoh gambar 8 buah sensor arah mata angin

2. Simpan (*Save As*) Program_3_3_a.vi dengan nama Program_3_3_b.vi.
3. Ambil objek Pict Ring (Picture Ring) dari Palet Controls di Front Panel, di kategori Modern, di Ring & Enum, tempatkan di samping Gauge.



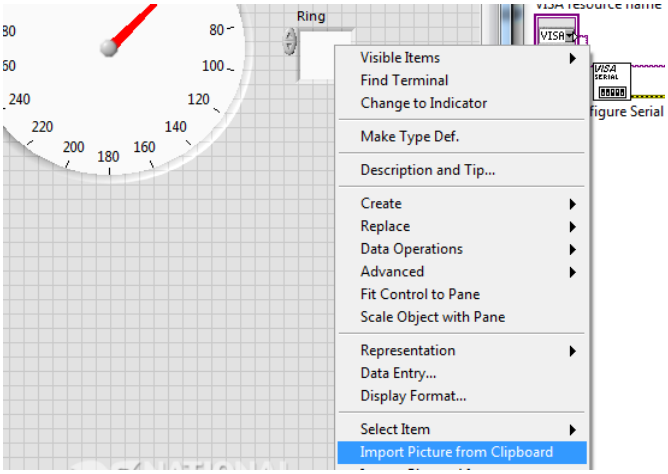
Gambar 3.50 Objek Picture Ring ditempatkan di samping Gauge

4. Pada langkah no. 1 di atas, simpan file gambar tersebut di lokasi tertentu. Kemudian buka menu Edit di LabVIEW, pilih Import Picture to Clipboard. Arahkan pencarian file ke lokasi tempat penyimpanan file gambar. Agar gambar bisa berurutan, pilih file pertama, yaitu arah1.png.



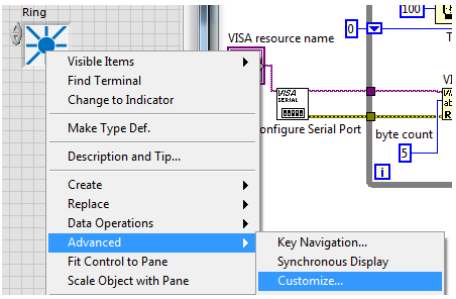
Gambar 3.51 Memasukkan gambar dari luar dengan Import Picture to Clipboard

5. Berikutnya, klik kanan pada objek Pict Ring, dan pilih Import Picture from Clipboard, untuk meletakkan gambar dari Clipboard ke dalam Pict Ring.



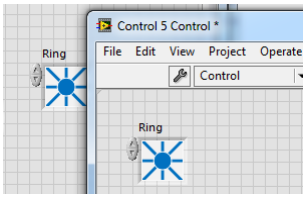
Gambar 3.52 Import Picture from Clipboard untuk menaruh gambar ke Pict Ring

6. Perhatikan bahwa gambar sensor arah angin muncul di Pict Ring, tetapi gambar belum tampak sepenuhnya. Untuk membuat gambar tampak dan menyatu dengan tampilan, klik kanan, pilih Advanced, pilih Customize.



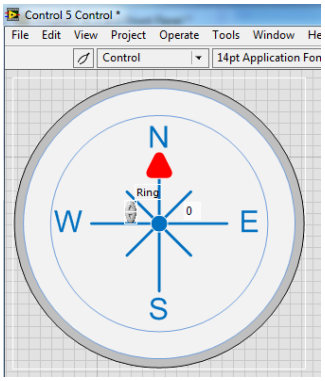
Gambar 3.53 Pilih Advanced, Customize untuk menampilkan gambar secara utuh

7. Pilihan Customize akan memunculkan jendela Customize Control. Ada 2 mode yang bisa digunakan, yaitu Mode Edit untuk mengedit teks dan Mode Customize untuk memodifikasi gambar. Awalnya, jendela Customize Control ini berada pada Mode Edit, dengan gambar tombol berbentuk Kunci pas. Untuk mengubahnya ke Mode Customize, tekan tombol bergambar Kunci pas sehingga berubah menjadi gambar Tang.



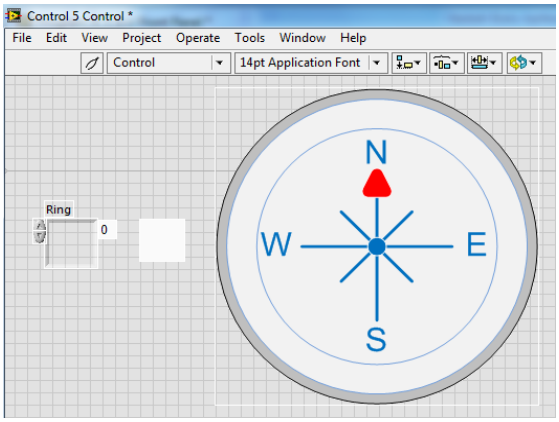
Gambar 3.54 Jendela Customize Control pada Mode Edit (gambar Kunci pas)

8. Ubah Mode ke Mode Customize (tombol Mode bergambar Tang)



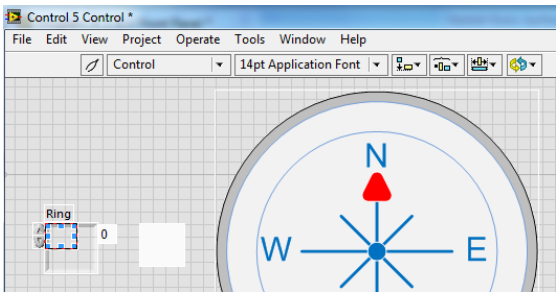
Gambar 3.55 Jendela Customize Control pada Mode Customize (gambar Tang)

9. Saat Mode Customize ini, objek Pict Ring dapat dipisahkan ke dalam 3 bagian. Bagian pertama berupa frame/pigura, bagian kedua berupa media penampil gambar dan bagian ketiga berupa gambar yang ingin ditampilkan di Pict Ring. Berikut 3 bagian Pict Ring yang telah dipisahkan:



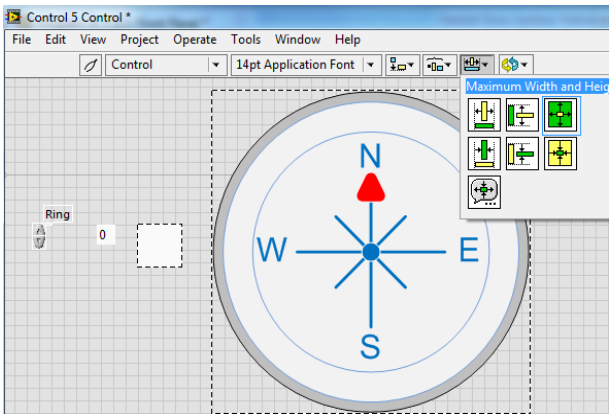
Gambar 3.56 Pict Ring terdiri dari 3 bagian: Frame, Media Penampil, dan Gambar

10. Berhubung Frame tidak perlu ditampilkan, maka buat ukurannya sekecil mungkin, dengan cara klik ujungnya dan tarik ke dalam hingga mengecil.



Gambar 3.57 Memperkecil ukuran Frame sekecil mungkin agar tak terlihat

11. Berikutnya, agar gambar dapat ditampilkan sepenuhnya, maka buat ukuran media penampil gambar (objek yang tengah) sebesar ukuran Gambar. Untuk itu sambil menekan tombol **Shift**, klik pada bagian penampil dan klik pada bagian Gambar untuk memilih keduanya, dan gunakan Tool Maximum Width and Height pada kategori Resize Objects, maka ukuran bagian penampil akan sebesar Gambar.

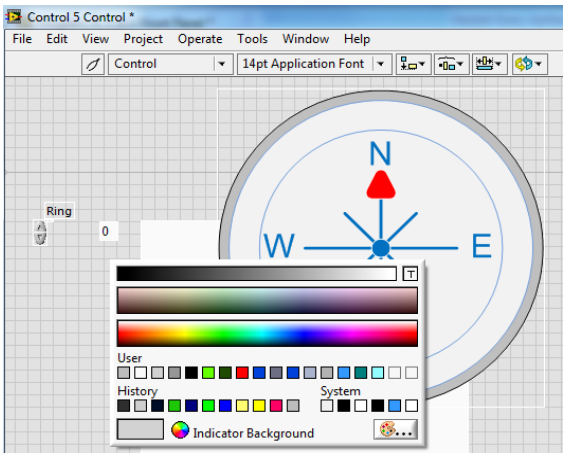


Gambar 3.58 Memperbesar bagian penampil hingga sebesar ukuran Gambar

12. Berikutnya, agar Gambar terlihat menyatu dengan Front Panel, maka buat agar bagian media penampil yang semula berwarna putih, menjadi berwarna transparan. Untuk memunculkan Tool yang mengubah warna, tekan tombol Shift dan klik kanan, maka akan muncul kotak Palet Tool. Pilih pengatur warna dengan meng-klik tool bergambar kuas (yang paling bawah), maka pointer berubah menjadi kuas. Sentuhkan kuas tersebut ke bagian penampil, maka akan muncul kotak warna. Pilih warna transparan dengan menyentuhkan kuas pada kotak dengan huruf T di pojok kanan, maka warna bagian penampil akan berubah menjadi transparan.

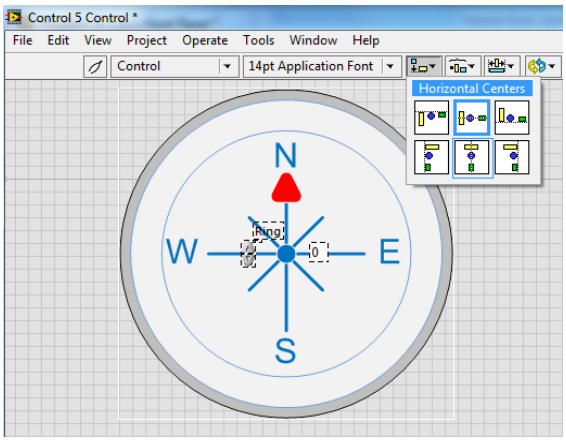
Gambar 3.59 Kotak Palet Tool yang berisi pointer kuas pengubah warna

13. Untuk mengembalikan pointer kuas menjadi pointer otomatis, buka kotak Palet Tool dengan menekan kembali tombol Shift dan klik kanan. Kemudian klik kotak LED agar LED menyala kembali untuk mode otomatis.



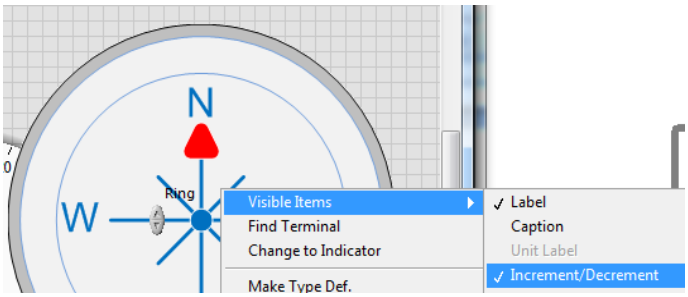
Gambar 3.60 Mengubah warna bagian penampil dari putih menjadi transparan

14. Buat agar ketiga bagian, baik bagian Frame, Media Penampil dan Gambar, menjadi satu, saling bertumpukan, dengan menggunakan Tool Top Edges dan Left Edges di kategori Align Objects.



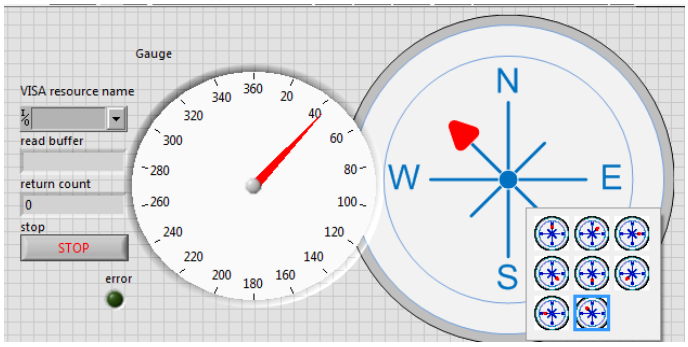
Gambar 3.61 Menyatukan ketiga bagian dengan Tool Align Objects

15. Tutup jendela Customize Control. Pilih Save untuk pertanyaan Save changes before Closing? Beri nama file Control tersebut.
16. Pada jendela Front Panel, hilangkan Label dan Increment/Decrement pada Pict Ring dengan meng-klik kanan Pict Ring, pilih Visible Items, dan hilangkan tanda centang pada Label dan Increment/Decrement.



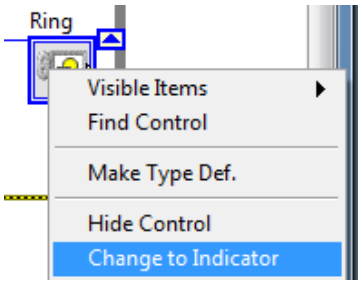
Gambar 3.62 Menghilangkan Label dan Increment/Decrement Pict Ring

17. Setelah Pict Ring dapat menampilkan gambar pertama secara penuh sesuai gambar di langkah no. 1, maka berikutnya adalah menambahkan 7 gambar lain secara berurutan sesuai posisinya. Untuk itu buka menu Edit, pilih Import Picture to Clipboard, arahkan ke file gambar kedua (arah2.png). Kemudian klik kanan pada Pict Ring, pilih Add Item After.
18. Ulangi langkah no. 17 di atas hingga semua gambar di langkah no. 1 berhasil ditambahkan secara berurutan. Setelah semua gambar selesai ditambahkan, jika Pict Ring di-klik seharusnya terlihat *thumbnail* semua gambar yang diletakkan secara berurutan seperti Gambar 3.63.



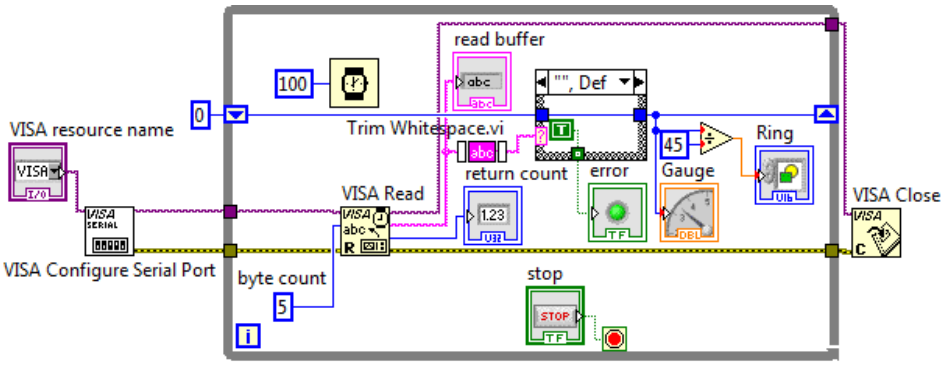
Gambar 3.63 Klik pada objek Pict Ring akan memunculkan 8 thumbnail gambar

19. Di Block Diagram, pastikan icon Pict Ring berfungsi sebagai Indicator, yang ditandai dengan adanya kaki input di sisi kiri. Untuk mengubahnya, klik kanan pada icon Pict Ring, pilih Change to Indicator (lihat Gambar 3.64).



Gambar 3.64 Klik kanan icon Pict Ring di Block Diagram, pilih Change to Indicator

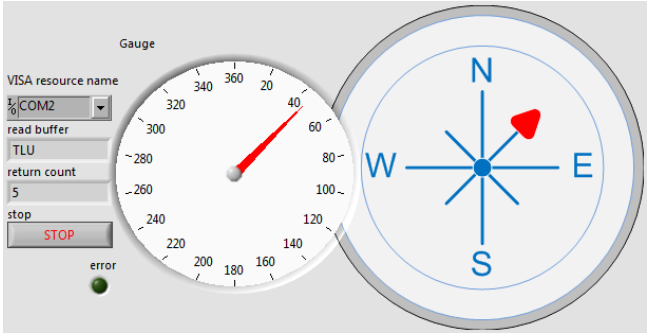
20. Berhubung objek Pict Ring hanya memiliki input angka dari 1 sampai dengan 8, sedangkan objek Gauge memiliki input angka dari 0 sampai 360, maka agar input Gauge bisa digunakan sebagai input Pict Ring, diperlukan pembagian input Gauge dengan 45, seperti Gambar berikut.



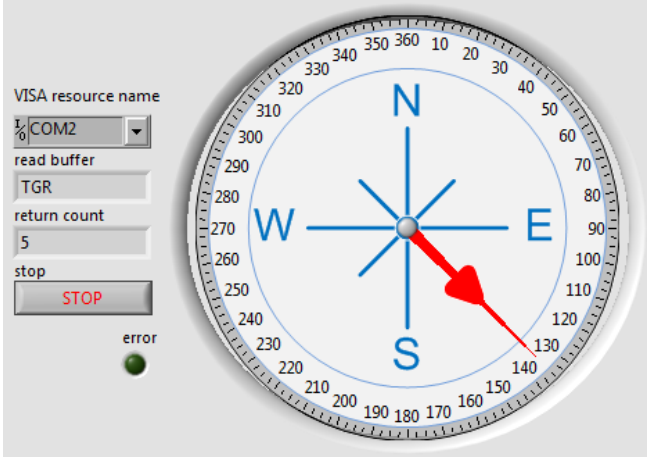
Gambar 3.65 Data input Gauge dibagi 45 untuk dijadikan input Pict Ring

21. Jalankan simulasi Proteus rangkaian Gambar 3.4 dan program LabVIEW di atas. Seharusnya tanda panah di objek Pict Ring akan menunjuk arah yang sama dengan jarum di objek Gauge, seperti ditunjukkan Gambar 3.66.
22. Agar tampilan lebih menarik, satukan objek Gauge dan objek Pict Ring. Tentunya agar warna putih di Gauge tidak menutupi warna Pict Ring, ubah warna Gauge menjadi transparan. Caranya adalah dengan meng-klik kanan Gauge, pilih Advanced, pilih Customize. Pada jendela Customize Control, buat Mode Customize untuk memisahkan bagian-bagian Gauge. Warnai bagian Media Penampil yang berwarna putih dengan warna

transparan. Satukan kembali bagian-bagian tersebut dan tutup jendela Customize Control. Agar lebih menarik, tumpuk Gauge dan Pict Ring. Perbesar objek Gauge sehingga garis strip skala di Gauge dapat tepat di slot Pict Ring, seperti ditunjukkan pada Gambar 3.67.

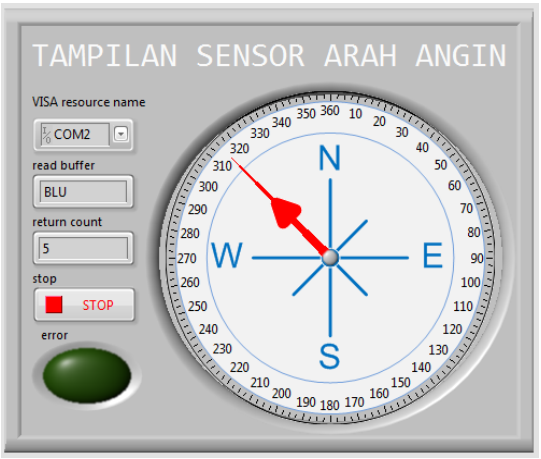


Gambar 3.66 Gauge dan Pict Ring menunjukkan arah yang sama saat dijalankan



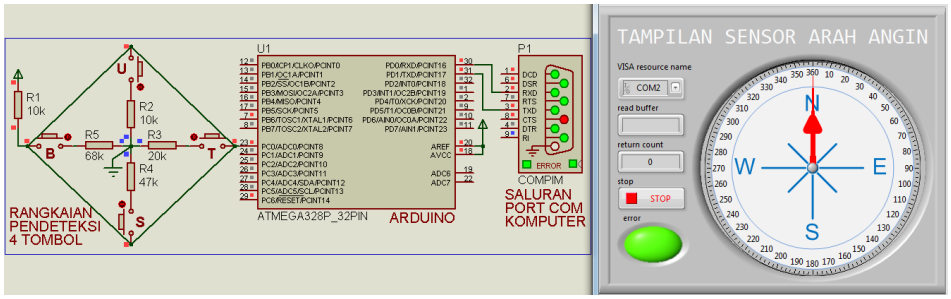
Gambar 3.67 Menumpuk Gauge di atas Pict Ring agar tampilan lebih menarik

23. Terakhir tambahkan Decorations, yang ada di Palet Controls, di Modern. Ambil Vertical Smooth Box dan Thick Lowered Box dan letakkan di bawah semua objek. Untuk menempatkan objek di bawah atau di atas objek lain, gunakan Tool Reorder, di samping Tool Resize Objects. Untuk membuat tulisan, gunakan Tool Text Edit (tombol A) di kotak Palet Tool, yang dimunculkan dengan cara menekan tombol Shift dan klik kanan.

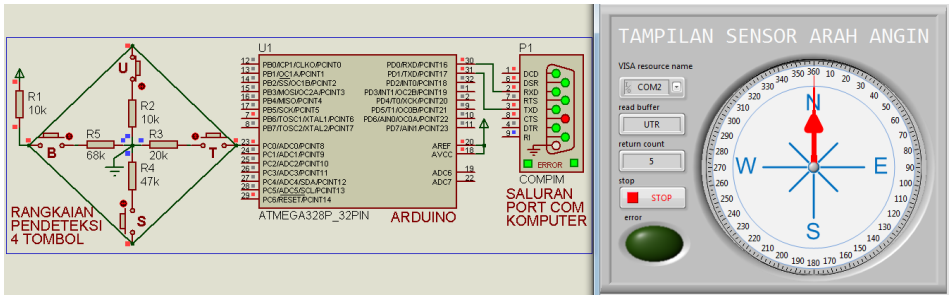


Gambar 3.68 Tampilan Gambar Data Arah Angin program LabVIEW

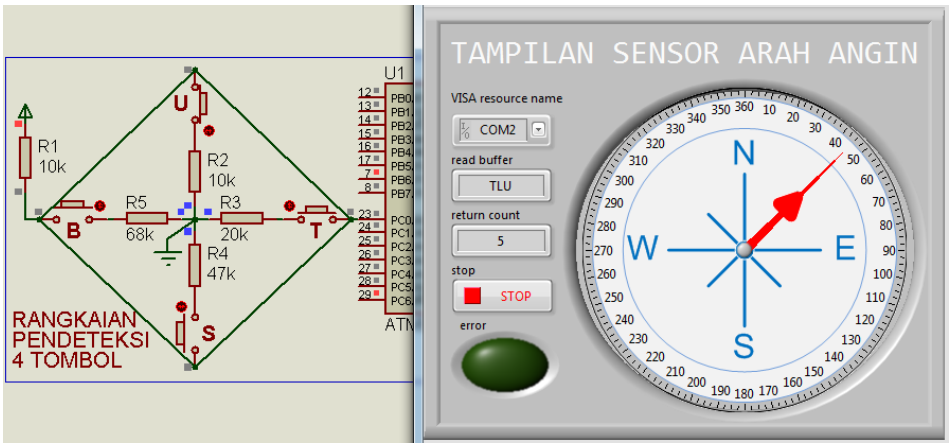
20. Jalankan kembali simulasi Proteus rangkaian Gambar 3.4 dan program LabVIEW di atas. Berikut ini berturut-turut diperlihatkan tampilan Gauge LabVIEW untuk kombinasi penekanan tombol mengikuti Tabel 3.1.



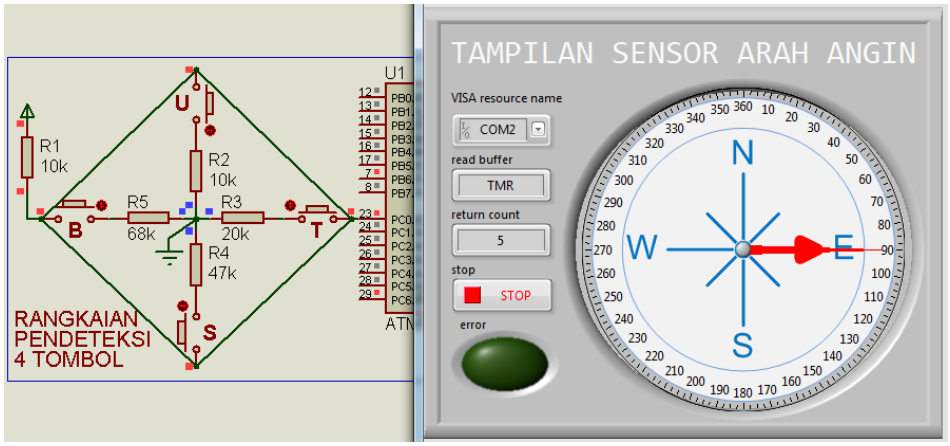
Gambar 3.69 Ketika tidak ada tombol ditekan, data kosong, LED Error menyala



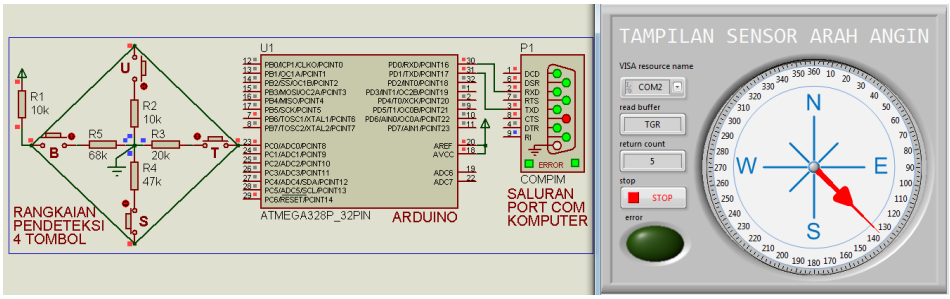
Gambar 3.70 Ketika tombol U ditekan, jarum menunjuk ke arah N atau 0^0



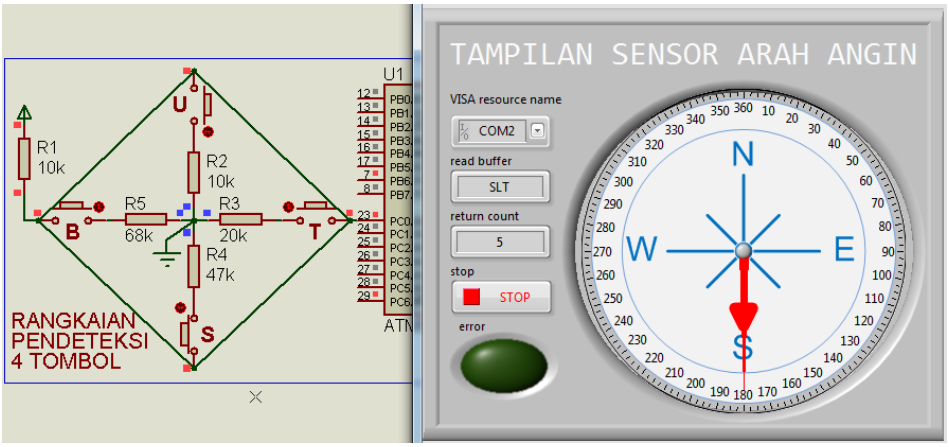
Gambar 3.71 Ketika tombol U dan T ditekan, jarum menunjuk ke arah NE atau 45⁰



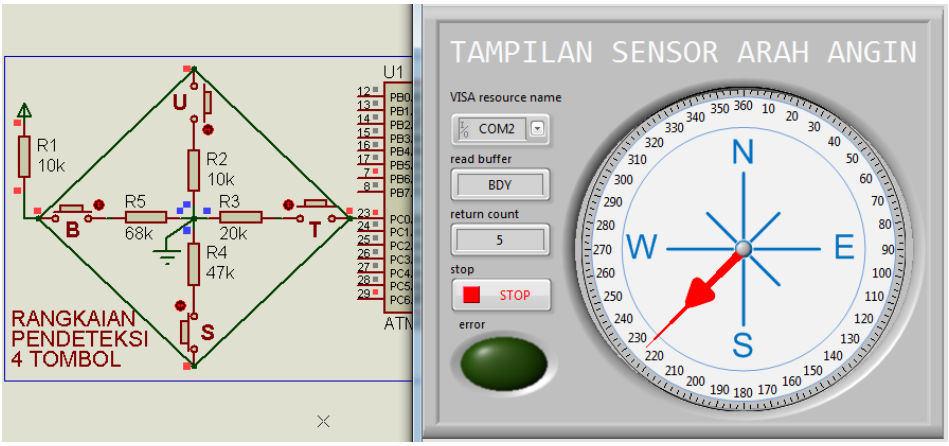
Gambar 3.72 Ketika tombol T ditekan, jarum menunjuk ke arah E atau 90⁰



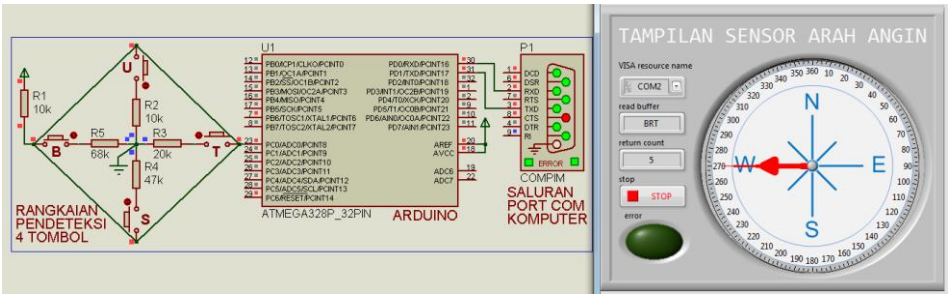
Gambar 3.73 Ketika tombol T dan S ditekan, jarum menunjuk ke arah SE atau 135⁰



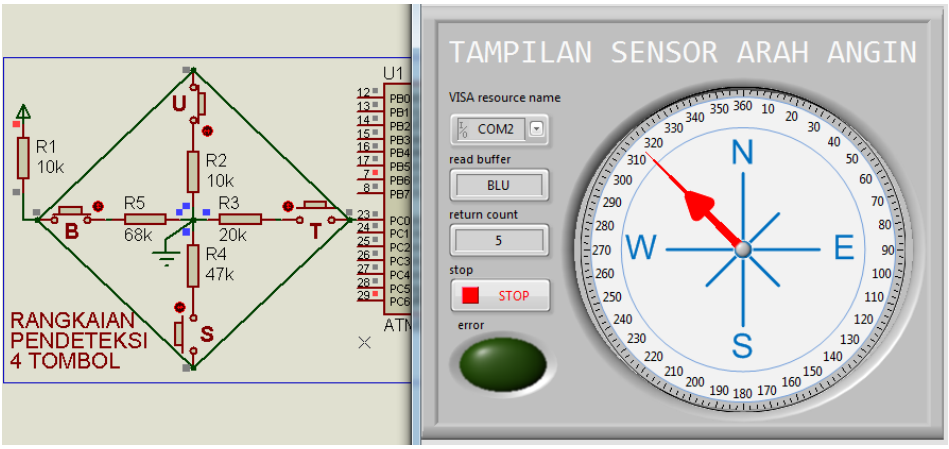
Gambar 3.74 Ketika tombol S ditekan, jarum menunjuk ke arah S atau 180^0



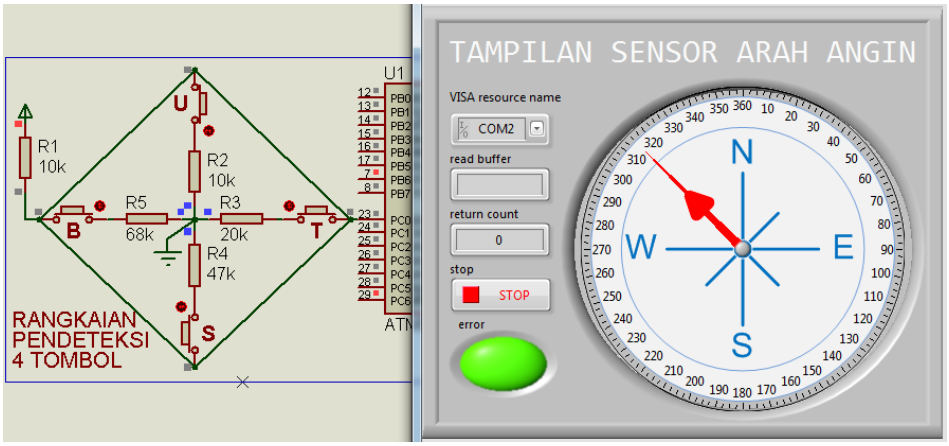
Gambar 3.75 Ketika tombol S & B ditekan, jarum menunjuk ke arah SW atau 225^0



Gambar 3.76 Ketika tombol B ditekan, jarum menunjuk ke arah W atau 270^0



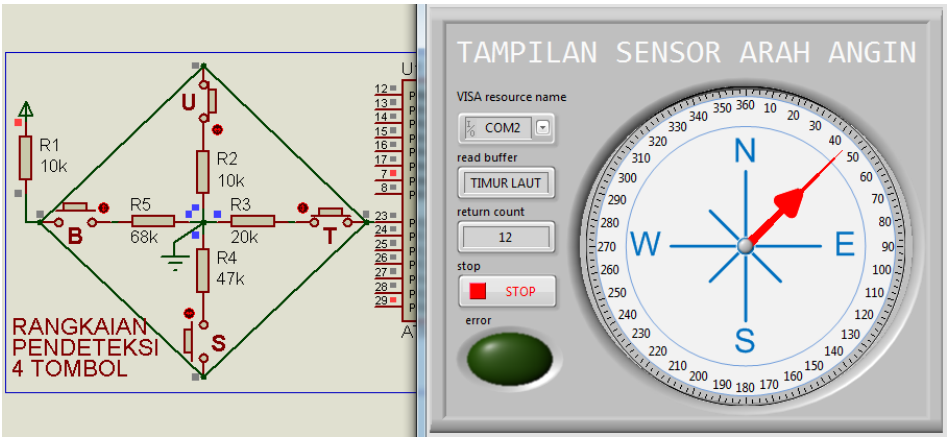
Gambar 3.77 Ketika tombol B & U ditekan, jarum menunjuk ke arah NW atau 315^0



Gambar 3.78 Ketika semua tombol ditekan, data kosong, LED Error menyala

3.5 Soal Latihan

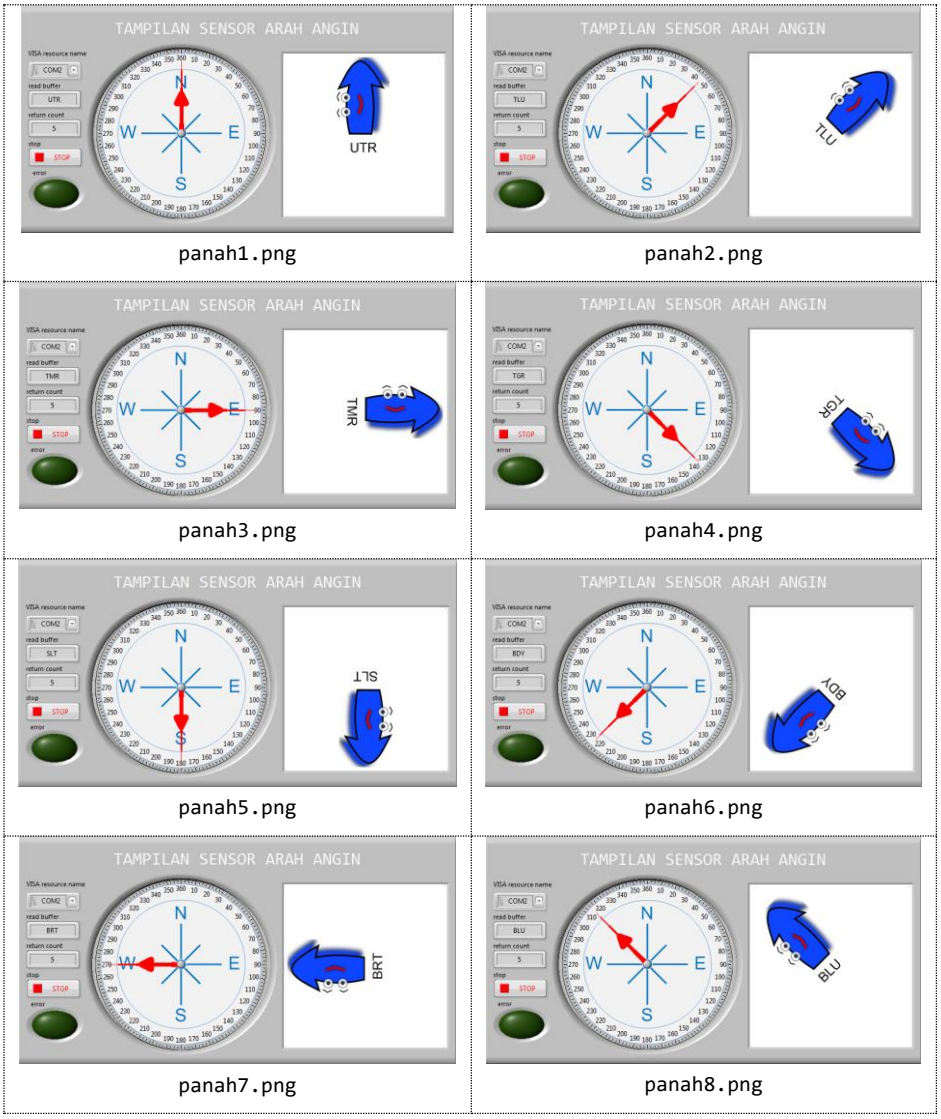
1. Tampilan data teks yang semula UTR, TLU, TMR, TGR, SLT, BDY, BRT dan BLU, diinginkan ditampilkan sesuai namanya, yaitu UTARA, TIMUR LAUT, TIMUR, TENGGARA, SELATAN, BARAT DAYA, BARAT dan BARAT LAUT. Buat program Arduino dan LabVIEW untuk menampilkan data teks sesuai 8 nama tersebut.



Gambar 3.79 Read buffer menampilkan data teks TIMUR LAUT, bukan TMR

Bantuan: Di Arduino, modifikasi Program_3_1.ino dengan mengganti kata UTR, TLU, TMR, TGR, SLT, BDY, BRT, dan BLU menjadi UTARA, TIMUR LAUT, TIMUR, TENGGARA, SELATAN, BARAT DAYA, BARAT dan BARAT LAUT. Di LabVIEW, isi byte count dengan nilai byte pada kata yang terpanjang ditambah dengan karakter akhiran. Untuk nama-nama arah di atas, jumlah karakter terbanyak adalah 10. Karena 1 karakter sama dengan 1 byte, maka 10 karakter sama dengan 10 byte. Bila ditambah dengan 2 karakter akhiran, yang berupa Carriage Return dan New Line, yang dihasilkan dari instruksi Serial.println(), maka total byte untuk byte count adalah 12. Perhatikan, bahwa sekalipun byte count diisi 12, untuk kata yang terpendek, seperti TIMUR, data teks tetap menampilkan utuh, tidak tersambung dengan kata yang lain, karena adanya karakter akhiran, yang menghentikan pembacaan tanpa perlu mencapai 12. Terakhir, Label Case Structure yang semula bertuliskan UTR, TLU, TMR, TGR, SLT, BDY, BRT, BLU, ubah menjadi UTARA, TIMUR LAUT, TIMUR, TENGGARA, SELATAN, BARAT DAYA, BARAT dan BARAT LAUT.

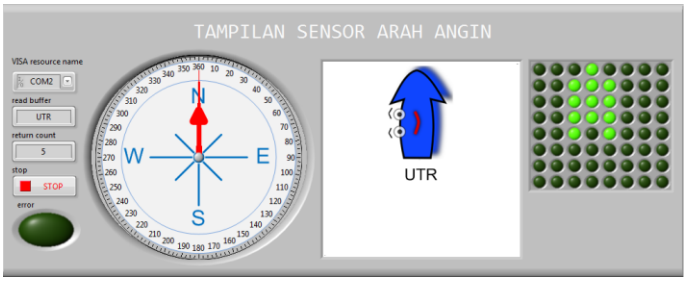
2. Untuk memperjelas tampilan arah angin, tambahkan gambar tanda panah yang menunjuk arah yang sama dengan jarum Gauge. File gambar tanda panah disediakan dalam CD pendukung buku ini. Tersedia 8 buah file gambar, dengan nama file panah1.png hingga panah8.png.



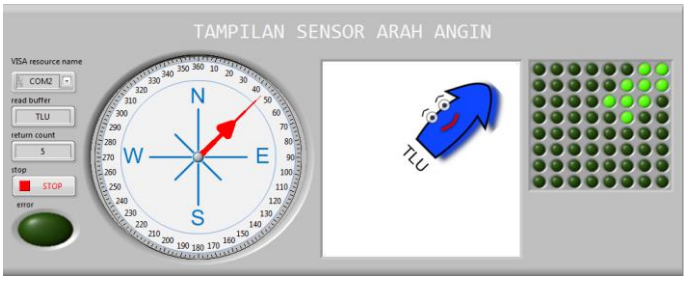
Gambar 3.80 Tampilan tanda panah menunjuk arah yang sama dengan Gauge

3. Diinginkan nantinya alat dilengkapi dengan hardware berupa matriks LED 8x8, yang dapat menampilkan tanda panah sensor arah angin di tempat gelap atau di malam hari. Agar nantinya matriks LED 8x8 tersebut dapat diatur dari LabVIEW, maka matriks LED 8x8 tersebut ditampilkan di LabVIEW dengan bentuk tanda panah seperti ditunjukkan pada gambar berikut ini:

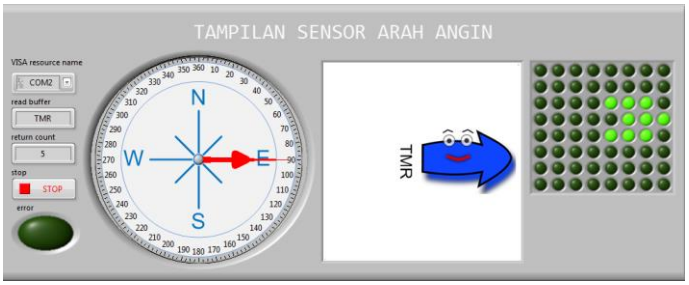
1. Arah Utara



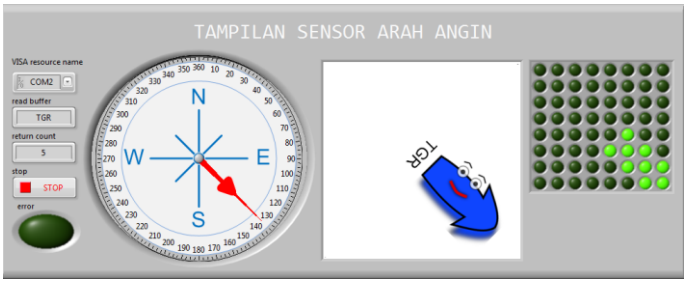
2. Arah Timur Laut



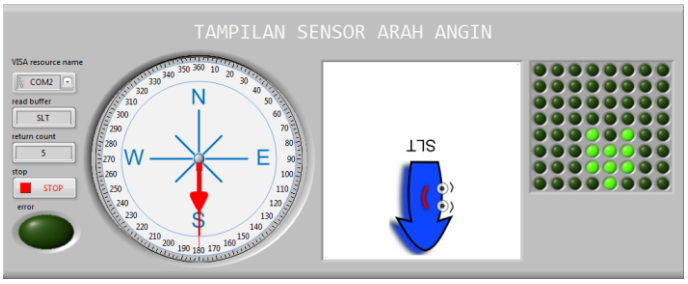
3. Arah Timur



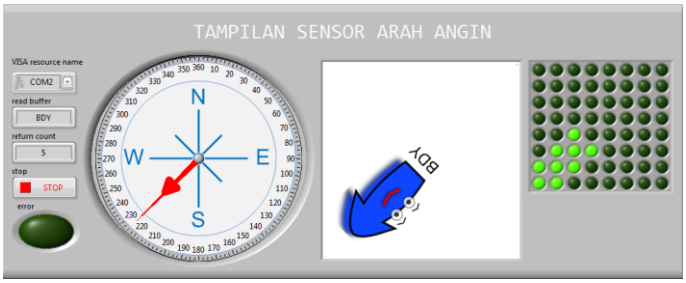
4. Arah Tenggara



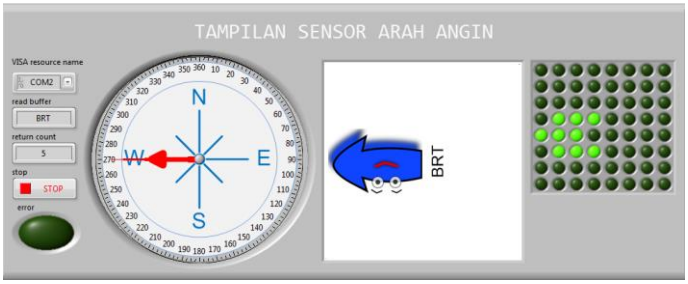
5. Arah Selatan



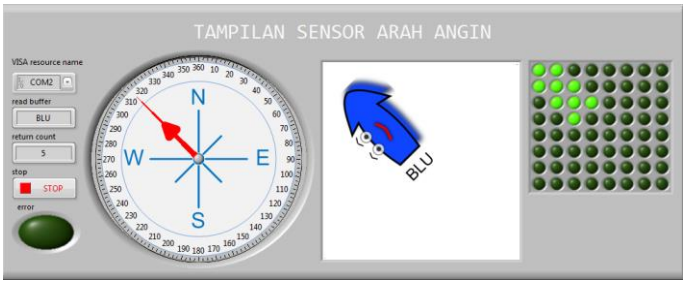
6. Arah Barat Daya



7. Arah Barat



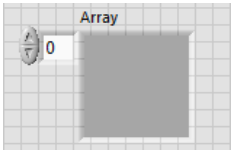
8. Arah Barat Laut



Gambar 3.81 Tanda panah pada matriks LED 8X8 menunjuk kedelapan arah angin

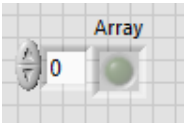
Bantuan: Untuk menampilkan matriks LED 8x8 di LabVIEW seperti gambar di atas, berikut ini langkah-langkahnya:

1. Ambil objek Array, dari Palet Controls, di kategori Modern, di kategori Array, Matrix & Clusters, dan tempatkan di Front Panel.



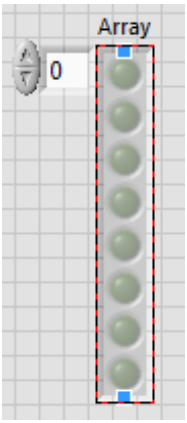
Gambar 3.82 Objek Array kosong

2. Ambil objek Round LED, dari Palet Controls, di kategori Modern, di kategori Boolean, dan tempatkan di dalam kotak objek Array.



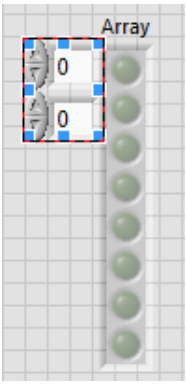
Gambar 3.83 Objek Array diisi dengan objek Round LED

3. Tarik kotak Array yang berisi LED ke bawah hingga muncul 8 buah LED.



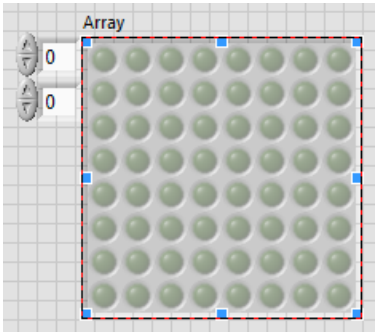
Gambar 3.84 Tarik objek Array ke bawah hingga muncul 8 buah LED

4. Untuk membuat Matriks 8x8, maka Array 1 Dimensi pada Gambar 3.84 di atas harus dibuat menjadi Array 2 Dimensi. Caranya adalah dengan menarik ke bawah kotak Index Display sehingga dari 1 kotak Index menjadi 2 kotak Index seperti gambar berikut.



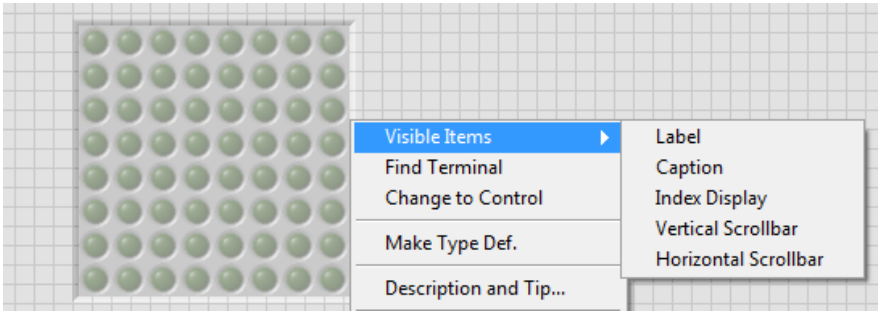
Gambar 3.85 Tarik kotak Index Display hingga menjadi 2 kotak

5. Setelah Index Display menjadi 2 kotak, tarik kotak yang berisi 8 LED ke kanan hingga kotak berisi 8x8 LED, seperti gambar berikut.



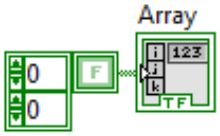
Gambar 3.86 Tarik kotak berisi 8 LED ke kanan hingga berisi 8x8 LED

6. Agar Label Array dan kotak Index Display tidak terlihat, klik kanan objek Matriks 8x8 LED tersebut, dan pada Visible Items, hilangkan tanda centang pada Label dan Index Display, sehingga seperti berikut.



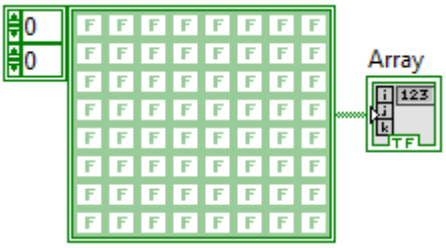
Gambar 3.87 Hilangkan kotak Index Display dan Label

7. Untuk membuat satu atau beberapa LED menyala, maka buka jendela Block Diagram. Cari icon objek Array tersebut, dan kemudian klik kanan pada kaki input dan pilih Create Constant, seperti gambar berikut.



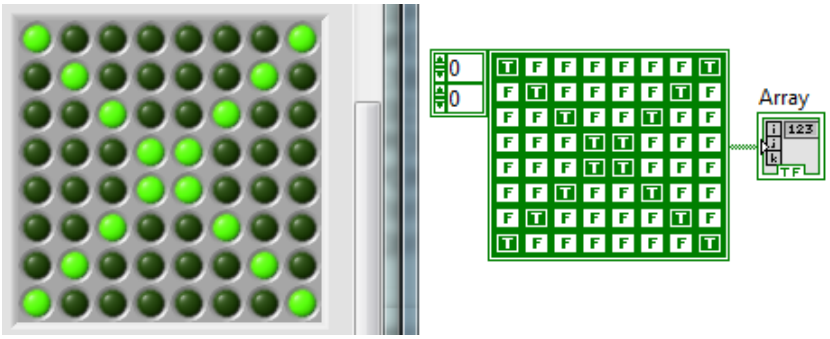
Gambar 3.88 Create Constant pada kaki input Array

8. Tarik kotak berisi **F** di Gambar 3.88 di atas, sehingga menjadi 8x8 kotak **F**.



Gambar 3.89 Tarik kotak berisi F sehingga menjadi 8x8 kotak F

9. Untuk membuat LED menyala, ubah nilai F menjadi T. Contoh untuk membuat tulisan X, maka buat beberapa kotak bernilai T seperti berikut.



Gambar 3.90 Ubah nilai F menjadi T untuk menyalakan LED

10. Dengan cara mengubah beberapa nilai F menjadi nilai T sehingga membentuk pola tanda panah yang menunjuk utara, timur laut, timur, tenggara, selatan, barat daya, barat dan barat laut, dan menempatkan setiap pola yang berbeda tersebut ke dalam Case Structure sesuai Casenya, maka pembaca dapat membuat tampilan sensor arah angin menggunakan matriks LED.

BAB 4
GRAFIK, TABEL DAN PENYIMPANAN DATA

Target Materi:

- Memprogram Arduino untuk membaca Sensor dan mengirimkan datanya ke komputer melalui komunikasi Serial.
- Memprogram LabVIEW untuk menampilkan data Sensor dalam bentuk teks, grafik dan tabel beserta catatan waktunya.
- Memprogram LabVIEW untuk menyimpan data Sensor ke dalam file beserta catatan waktunya.
- Membuat fitur pemanggilan file yang berisi data Sensor dan menampilkan isi file tersebut, baik dalam bentuk teks maupun grafik.

Persoalan:

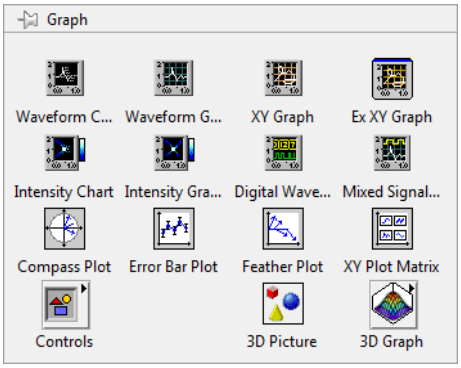
Buat agar data Sensor dapat ditampilkan dalam bentuk Grafik dan Tabel, serta disimpan secara otomatis dan berkala.

Uraian Materi:

4.1 Menampilkan Grafik

Bab 4 ini merupakan kelanjutan dari Bab 3, yang membahas mengenai penyajian data sensor arah angin. Apabila Bab 3 hanya menampilkan data dalam bentuk teks dan gambar, maka di Bab ini, data ditampilkan dalam bentuk grafik dan tabel. Sama seperti di bab sebelumnya, sensor arah angin di bab ini disimulasikan

dengan rangkaian Gambar 3.4 menggunakan Proteus. Perlu diketahui bahwa LabVIEW memiliki keunggulan dalam hal pembuatan Grafik. Jenis Grafik yang disediakan LabVIEW cukup banyak dan bervariasi. Pembaca dapat menemukannya di Palet Controls, di kategori Modern, di Graph, seperti ditunjukkan Gambar 4.1.



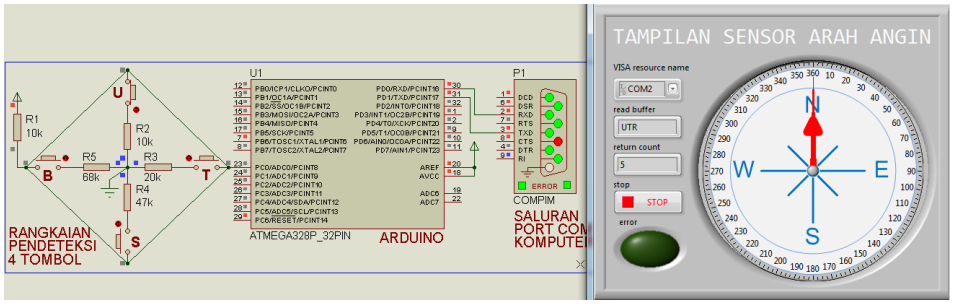
Gambar 4.1 Jenis Grafik yang disediakan LabVIEW

Dari sekian jenis grafik yang disediakan LabVIEW, di sini hanya akan dibahas penggunaan 3 jenis grafik saja, yaitu Waveform Chart, Waveform Graph dan 2D Compass. Berikut ini langkah-langkah penggunaan ketiga Grafik untuk aplikasi pemantauan data Sensor Arah Angin.

4.1.1 Penggunaan Waveform Chart

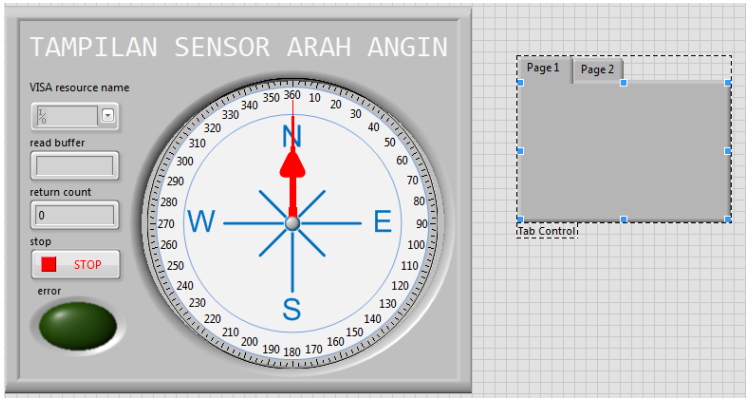
Berikut ini langkah-langkah penggunaan Waveform Chart:

1. Buka kembali rangkaian simulasi Sensor Arah Angin di software Proteus (Gambar 3.4) dan tampilannya di LabVIEW (Program_3_3_c) seperti ditunjukkan pada gambar berikut ini:



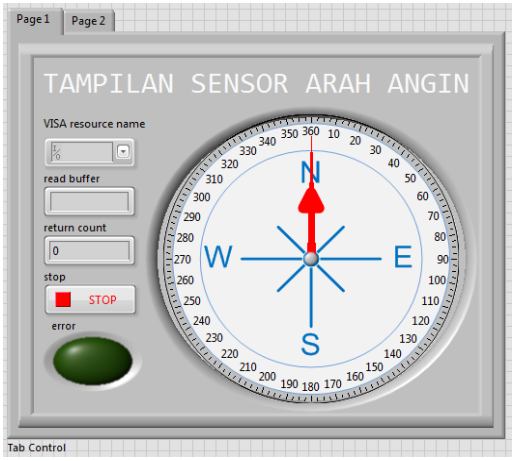
Gambar 4.2 Rangkaian Sensor di Proteus dan Tampilannya di LabVIEW

2. Di LabVIEW, di jendela Front Panel, tambahkan sebuah objek Tab Control, yang diambil dari Palet Controls, kategori Modern, kategori Containers, seperti ditunjukkan gambar berikut:



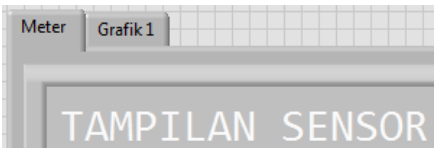
Gambar 4.3 Menambahkan objek Tab Control di Front Panel

3. Perbesar ukuran Tab Control sehingga dapat memuat kotak Tampilan Sensor Arah Angin. Kemudian pindahkan kotak Tampilan Sensor Arah Angin tersebut ke dalam Tab Control (Page 1). Namun agar urutan tumpukan objek (*order*) di kotak Tampilan tidak berubah, sebelum memindahkan kotak tersebut, pilih semua objek, kemudian klik pilihan Group di Menu Toolbar Reorder (bergambar 2 panah melingkar).



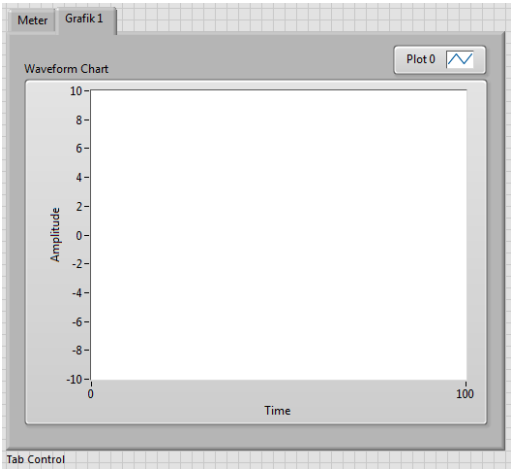
Gambar 4.4 Pilih kotak Tampilan, di-Group-kan, pindahkan ke dalam Tab Control

4. Ubah nama Page 1 menjadi Meter dan Page 2 menjadi Grafik 1.



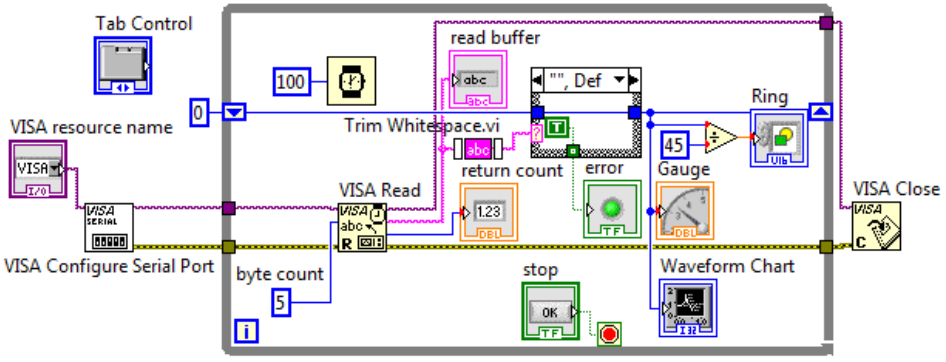
Gambar 4.5 Mengubah nama halaman Tab Control menjadi Meter dan Grafik 1

5. Klik halaman Grafik 1. Tempatkan Waveform Chart yang diambil dari palet Controls, di Silver, di Graph, di tengah-tengah kotak Tab Control Grafik 1.



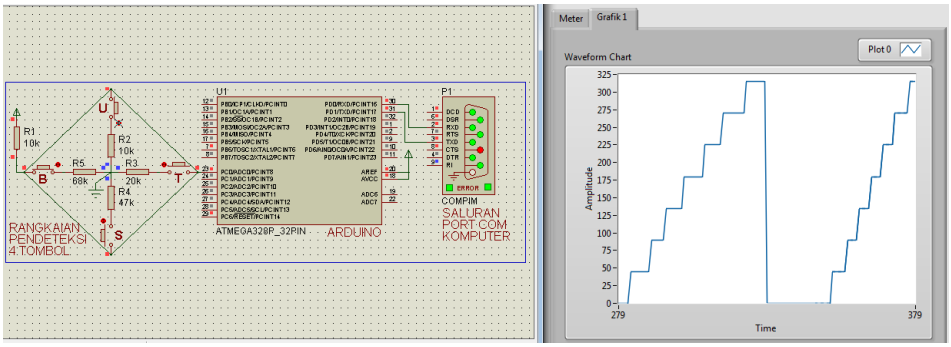
Gambar 4.6 Tempatkan Waveform Chart (Silver) di Tab Control Grafik 1

6. Di Block Diagram, hubungkan icon Waveform Chart dengan icon Gauge.



Gambar 4.7 Menghubungkan icon Waveform Chart dengan icon Gauge

7. Jalankan simulasi rangkaian sensor arah angin di Proteus (pastikan kolom Program File telah terisi dengan file hex hasil kompilasi Program_3_1.ino), dan tekan tombol Run di LabVIEW. Pastikan saluran COM yang digunakan, baik di Proteus maupun di LabVIEW sudah benar (gunakan sepasang COM yang dibuat dengan VSPD Eltima). Seharusnya Waveform Chart dapat menampilkan grafik seperti ditunjukkan pada gambar berikut.

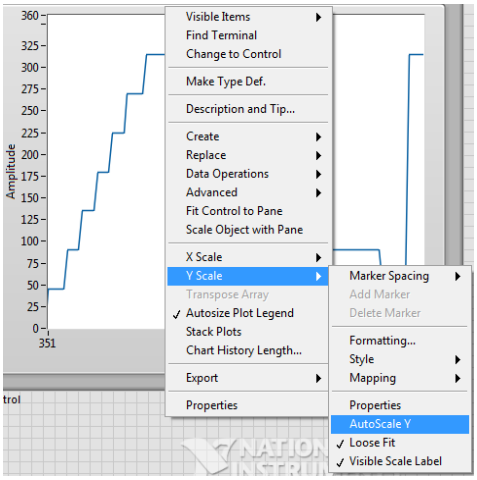


Gambar 4.8 Waveform Chart (Silver) menampilkan grafik data sensor

Catatan: Setiap kali menghubungkan simulasi rangkaian di Proteus dengan program LabVIEW, pastikan bahwa LED Indikator Error di halaman Meter tidak menyala. Apabila LED tersebut menyala, maka ada 2 kemungkinan kesalahan, yaitu tombol di rangkaian pendeteksi 4 tombol tidak ada yang ditekan, atau karena komunikasi kedua softare belum berhasil. Untuk kemungkinan kedua, atasi dengan cara menghidupkan simulasi di Proteus dulu, baru diikuti LabVIEW. Sedangkan cara menghentikan programnya, selalu mulai dari mematikan LabVIEW dulu baru Proteus. Dalam hal menghentikan program LabVIEW, jangan sekali-kali menghentikannya dengan menekan tombol Abort Execution yang ada di menu Toolbar, karena akan membuat VISA Close tidak dijalankan sehingga sambungan tidak ditutup dengan baik. Untuk itu sebaiknya selalu hentikan program LabVIEW dengan menekan tombol Stop yang ada di halaman program (di Tab Meter).

8. Perhatikan bahwa pada grafik di atas, skala sumbu Y akan berubah mengikuti tinggi grafik. Apabila diinginkan skala sumbu Y tidak berubah dapat dilakukan dengan menghilangkan tanda centang pada pilihan

AutoScale Y. Untuk memunculkan pilihan AutoScale Y ini, klik kanan pada grafik, pilih Y scale pada kotak yang muncul, dan kemudian hilangkan tanda centang pada pilihan AutoScale Y.



Gambar 4.9 Membuat sumbu Y tetap dengan menghilangkan centang AutoScale Y

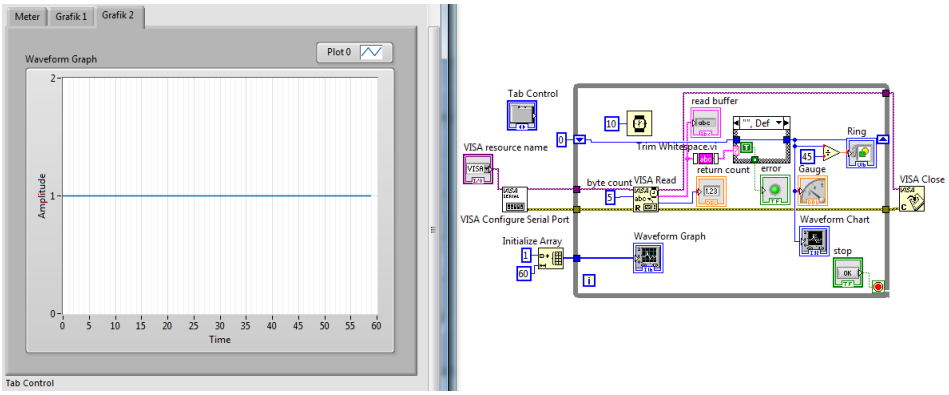
Catatan: Ketika AutoScale Y dimatikan, pastikan bahwa grafik tidak terpotong, yaitu dengan membuat nilai maksimum skala sumbu Y lebih tinggi dari nilai data maksimum. Untuk mengubah angka pada skala sumbu Y, klik 2 kali pada angka di ujung garis sumbu dan ketik angka yang baru.

4.1.2 Penggunaan Waveform Graph

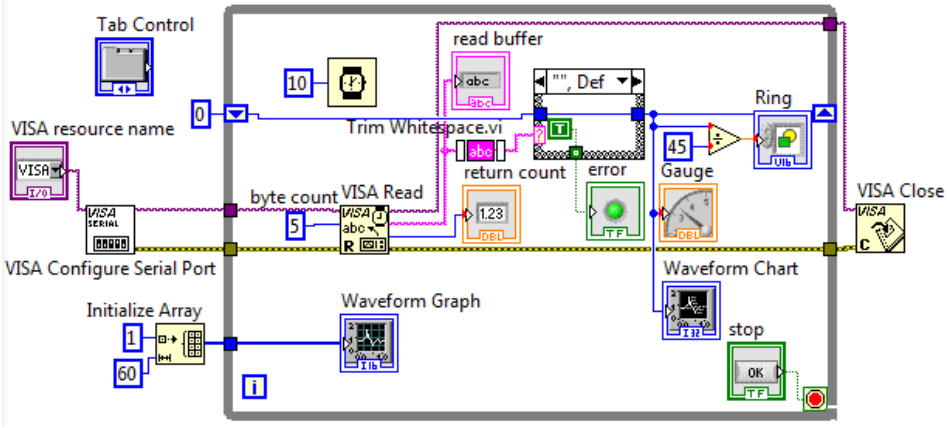
Berikut ini langkah-langkah penggunaan Waveform Graph:

1. Pada kotak Tab Control, klik kanan pada Tab Grafik 1, kemudian pilih Add Page After. Maka akan muncul Tab Grafik 2.
2. Pada Tab Grafik 2, tempatkan Waveform Graph (Silver) yang diambil dari palet Controls, di kategori Silver, di kategori Graph.
3. Berbeda dengan input data Waveform Chart yang bisa berupa data tunggal, input data Waveform Graph harus berupa data Array. Dengan data Array ini menunjukkan bahwa pada Waveform Graph memerlukan data lengkap dari titik awal hingga titik akhir garis kurva grafik.

4. Sebagai contoh untuk menampilkan garis datar dengan tinggi 1 (skala sumbu Y) dari titik 0 hingga 59 di sumbu X, berikut ini tampilannya di Front Panel dan programnya di Block Diagram.



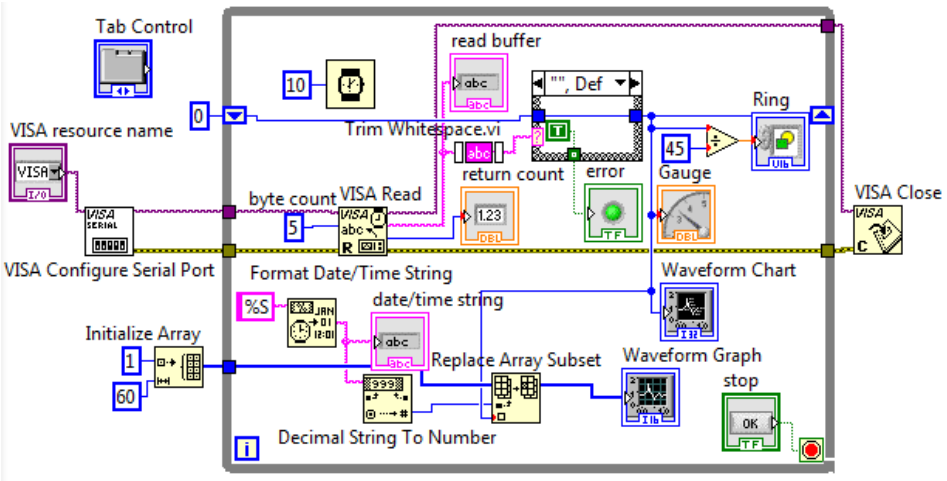
Gambar 4.10 Menampilkan garis datar dengan tinggi 1 sepanjang 0-59 sumbu X



Gambar 4.11 Block Diagram Gambar 4.10

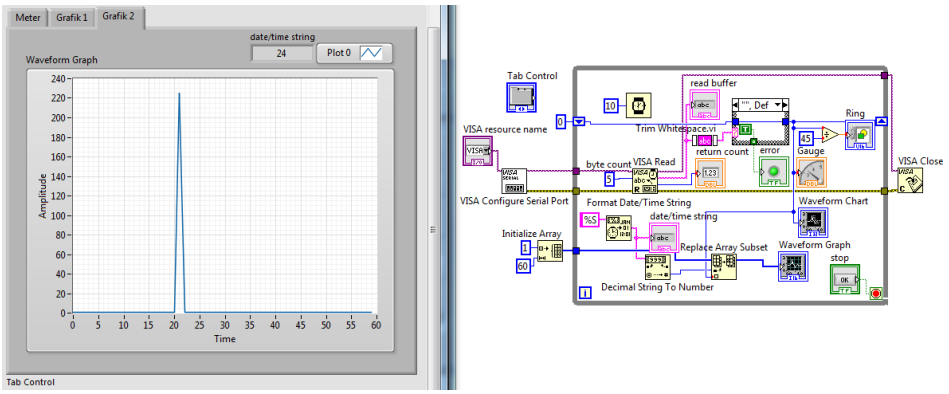
5. Tampak di Block Diagram, icon Waveform Graph mendapat input dari icon Initialize Array. Dengan memberi input element = 1 dan dimension = 60, icon Initialize Array tersebut akan menghasilkan sebuah array atau larik dengan anggota berupa angka 1 sebanyak 60 buah. Karena index array selalu dimulai dari 0, maka garis grafik hanya sampai pada titik sumbu x 59. Tampak ketika titik sumbu x mencapai 60 garis grafik hilang.

6. Diinginkan bahwa sumbu x menunjukkan waktu dalam detik. Apabila data ditampilkan perdetik, maka untuk waktu 1 menit, akan ada 60 buah data sensor arah angin. Untuk bisa menampilkan data tersebut pada Waveform Graph, gunakan icon Replace Array Subset untuk menggantikan data yang dihasilkan oleh Initialize Array dengan data dari sensor arah angin. Untuk itu sisipkan icon *Replace Array Subset* ini di tengah garis antara icon Initialize Array dengan icon *Waveform Graph* seperti ditunjukkan dalam gambar berikut ini:



Gambar 4.12 Replace Array Subset untuk mengganti data Initialize Array

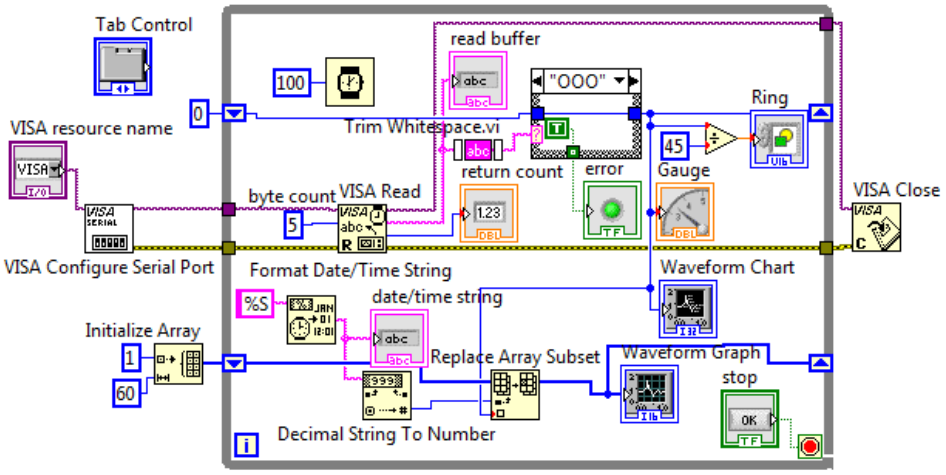
7. Pada Gambar 4.12 di atas, tampak bahwa icon *Replace Array Subset* memiliki 3 kaki input dan 1 kaki output. Input pertama adalah input dari data Array yang akan diganti. Input kedua dan ketiga adalah input *index* dan input *new element*. Input *new element* adalah input data baru untuk menggantikan data lama, yang posisi data lama yang akan digantikan tersebut ditentukan oleh data *index*. Dengan membuat data *index* berisi nilai waktu dalam detik (dari 0 – 59), dan data *new element* diisi data sensor arah angin, maka akan dihasilkan output data array baru yang berisi data sensor arah angin per detik. Data array baru tersebut dapat ditampilkan dalam bentuk grafik di Waveform Graph. Jalankan Proteus dan LabVIEW, maka hasilnya seperti ditunjukkan dalam Gambar 4.13.



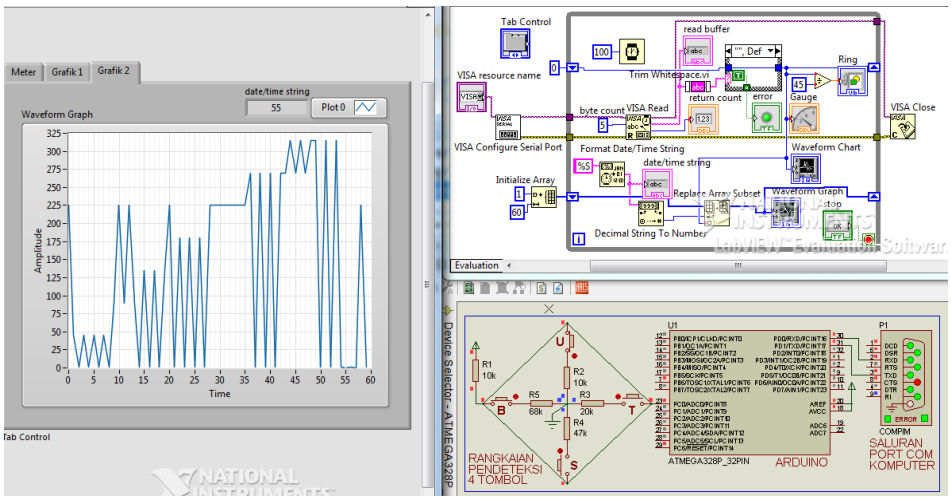
Gambar 4.13 Hasil grafik data sensor angin perdetik di Waveform Graph

Catatan: Lihat Gambar 4.12. Untuk memberikan data waktu dalam detik ke kaki input index, gunakan icon Format Date/Time String, dengan kaki input time format string untuk icon tersebut diberi %S. (untuk memunculkan input %S ini, klik kanan pada kaki input, pilih Create, pilih Constant, dan kemudian pada kotak yang muncul, ketikkan %S). Kemudian untuk menampilkan nilai detik tersebut, klik kanan pada kaki output icon Format Date/Time String, pilih Create pilih Indicator. Berhubung nilai detik yang dihasilkan tersebut bertipe string, sedangkan kaki input index membutuhkan tipe data integer, maka ubah tipe data string menjadi integer dengan menyisipkan icon Decimal String to Number. Icon ini dapat diambil dari palet Functions, di Programming, di String, di kategori Number/String Conversion.

8. Perhatikan bahwa grafik yang dihasilkan masih salah, yaitu data arah angin yang lama, tidak tersimpan. Agar data arah angin yang lama tersimpan, tambahkan Shift Register, dengan meng-klik kanan dinding kiri While Loop yang dilintasi garis data dari Initialize Array, dan memilih Replace with Shift Register. Maka akan muncul terminal Shift Register di dinding kiri While Loop, sedangkan untuk terminal Shift Register yang lain, tempatkan di dinding kanan While Loop. Kemudian hubungkan garis data icon Waveform Graph dengan terminal Shift Register di dinding kanan, seperti ditunjukkan dalam Gambar 4.14.



Gambar 4.14 Menambahkan Shift Register untuk menyimpan data array



Gambar 4.15 Hasil grafik di Waveform Graph setelah Shift Register ditambahkan

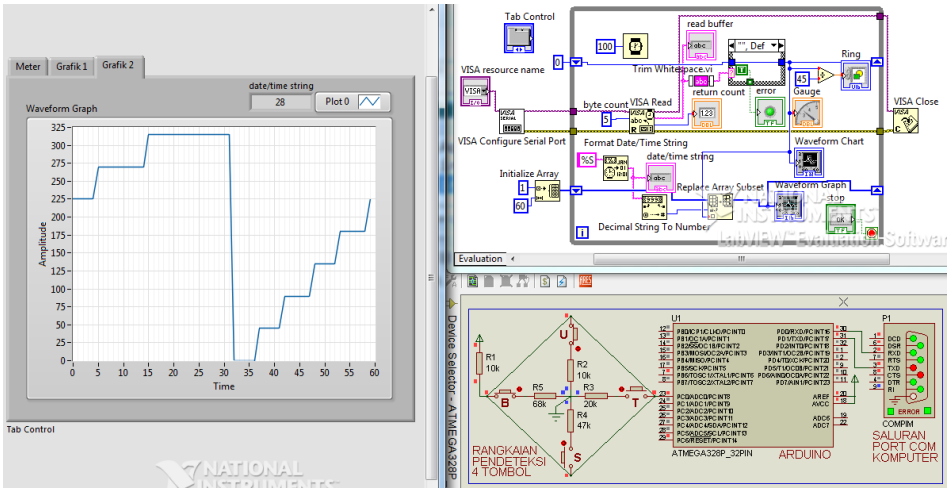
9. Sekalipun sudah bisa menyimpan data yang lama, namun grafik yang dihasilkan masih salah, yaitu data arah angin yang seharusnya di 45 derajat, ternyata sempat ke 0 derajat, begitu juga untuk arah angin yang lain, membuat grafik berbentuk gigi gergaji. Hal ini disebabkan karena tidak sinkronnya antara kecepatan pengiriman dan penerimaan data. Arduino mengirimkan data setiap detik, sedangkan LabVIEW menantikan

data setiap 0,1 detik, mengakibatkan ketika tidak ada data diterima, dianggap datanya 0. Untuk membuat data tidak sempat ke 0, salah satu caranya adalah dengan mengubah delay pada Program_3_1.ino, dari 1000 milidetik, menjadi 100 milidetik, seperti berikut:

Program_4_1.ino

```
1.  void setup() {
2.    Serial.begin(9600);
3.  }
4.  void loop() {
5.    int a = analogRead(A0);
6.    if (a == 512) Serial.println("UTR");
7.    if (a == 410) Serial.println("TLU");
8.    if (a == 682) Serial.println("TMR");
9.    if (a == 598) Serial.println("TGR");
10.   if (a == 844) Serial.println("SLT");
11.   if (a == 753) Serial.println("BDY");
12.   if (a == 892) Serial.println("BRT");
13.   if (a == 477) Serial.println("BLU");
14.   delay(100);
15. }
```

10. Kompilasi program di atas, dan jalankan kembali simulasi Proteus dan program LabVIEW, maka hasil grafiknya seharusnya seperti berikut:



Gambar 4.16 Hasil grafik di Waveform Graph setelah kecepatan pengiriman dari Arduino dan penerimaan data di LabVIEW disamakan sebesar 100 milidetik

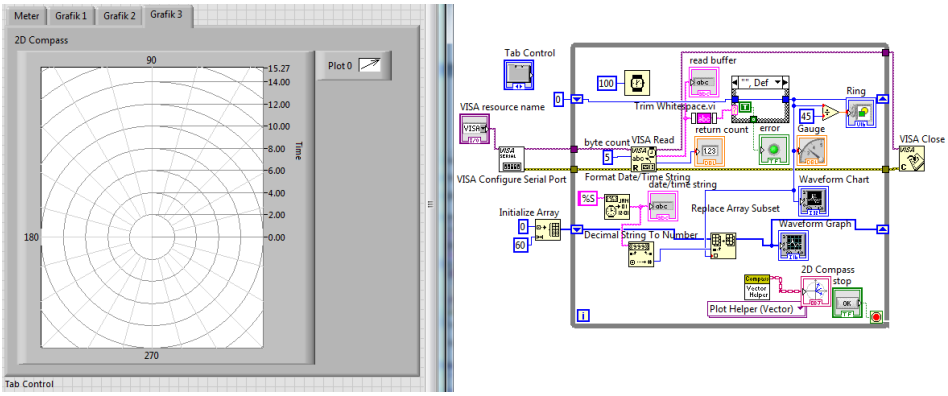
4.1.3 Penggunaan 2D Compass

Grafik 2D Compass cukup menarik untuk ditambahkan. Ada 3 hal yang bisa diperoleh dari Grafik 2D Compass ini, yaitu:

1. Data posisi sudut arah angin.
2. Data kapan posisi sudut tersebut terjadi.
3. Berapa kali posisi sudut tersebut terjadi dalam satu periode waktu.

Berikut langkah-langkah penggunaan 2D Compass:

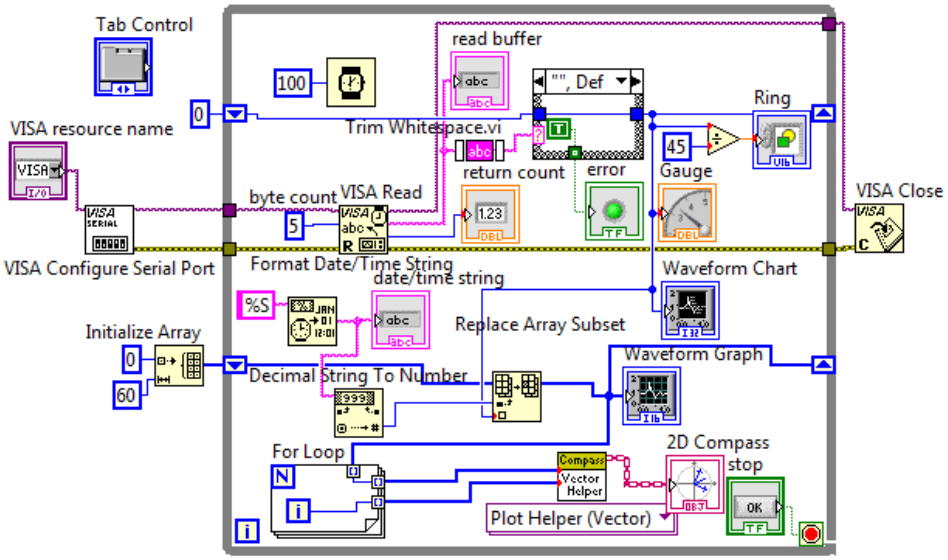
1. Di jendela Front Panel, pada kotak Tab Control, klik kanan pada Tab Grafik 2, kemudian pilih Add Page After, maka akan muncul Tab Grafik 3.
2. Ambil Grafik 2D Compass di Palet Controls, di Modern, di Graph, dan tempatkan di Tab Grafik 3.



Gambar 4.17 Menempatkan 2D Compass pada Tab Grafik 3

3. Di Block Diagram, tempatkan icon 2D Compass di dalam While Loop.
4. Karena nilai element di Initialize Array akan berpengaruh terhadap nilai Time di grafik 2D Compass, maka agar nilai Time tersebut bisa dimulai dari 0, ubah angka 1 di input element Initialize Array menjadi 0.
5. Hubungkan garis data Waveform Graph ke kaki input theta vector icon Compass Vector Helper.
6. Agar grafik dapat menampilkan data perdetik selama 1 menit, maka dibutuhkan data waktu yang nilainya dimulai dari 0 hingga 59. Untuk menghasilkan array nilai waktu tersebut, tambahkan sebuah kotak For

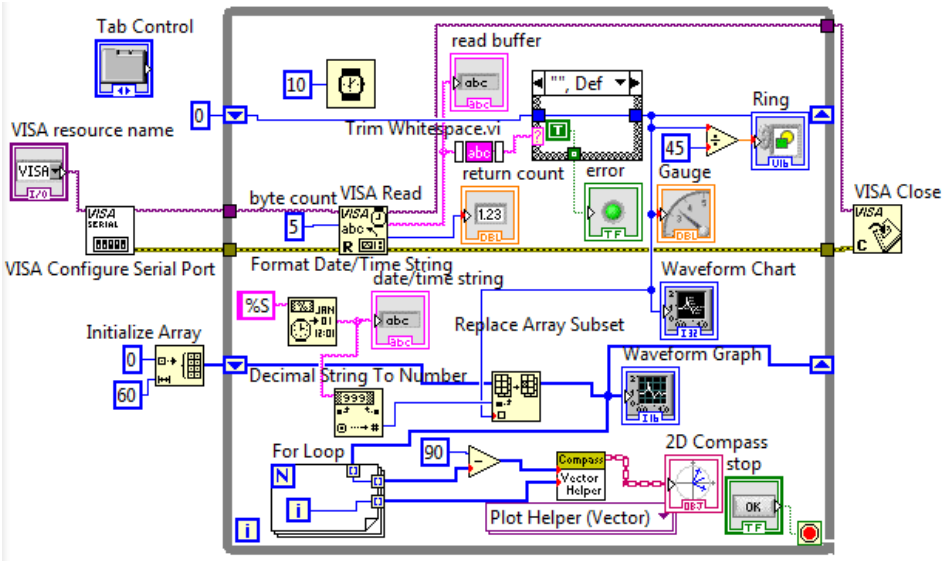
Loop. Lewatkan garis data Waveform Graph ke dalam kotak For Loop sebelum garis data tersebut masuk ke kaki input theta vector. Kemudian hubungkan garis data dari terminal iterasi (i) kotak For Loop ke kaki input radius vector Compass seperti ditunjukkan pada Gambar 4.18 berikut ini.



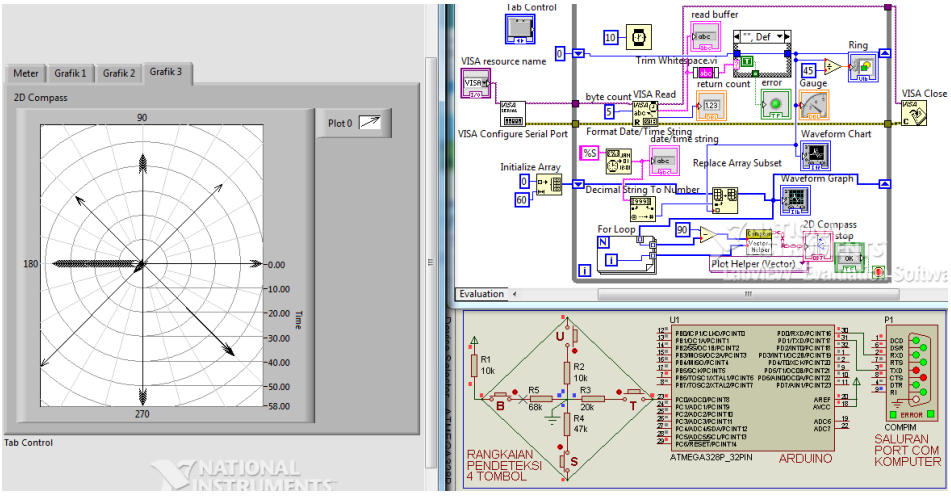
Gambar 4.18 Data Waveform Graph untuk theta dan iterasi untuk radius vector

Catatan: For Loop pada program di atas digunakan untuk menghasilkan data array yang dipakai sebagai input radius vector. Data array pada input radius vector ini akan ditampilkan sebagai nilai Time pada grafik 2D Compass. Untuk itu data array tersebut harus berisi angka yang dimulai dari 0 hingga 59. Untuk menghasilkan data array tersebut, dilakukan dengan melewatkan garis data Waveform Graph ke dalam kotak For Loop. Dengan melewatkan garis data tersebut, For Loop akan melakukan pencacahan (indexing) data Waveform Graph. Karena jumlah data Waveform Graph sebanyak 60, maka secara otomatis For Loop akan melakukan perulangan sebanyak 60 kali, yang membuat terminal iterasi (i) mengeluarkan angka yang berurut mulai dari 0 hingga 59. Perhatikan bahwa kaki input Loop Count (huruf N), tidak perlu diberi nilai, karena ketika sebuah array dilewatkan pada dinding kotak For Loop, secara otomatis For Loop akan melakukan proses pencacahan sebanyak jumlah anggota array.

7. Terakhir, agar arah panah pada grafik 2D Compass sama dengan arah simulasi sensor arah angin, yaitu arah panah menunjuk ke atas ketika tombol U ditekan, arah panah menunjuk ke samping kanan ketika tombol T ditekan, dan seterusnya, maka tambahkan pengurangan dari 90 untuk data theta, seperti Block Diagram berikut:



Gambar 4.19 Data theta dikurangkan dari 90 agar sesuai dengan arah angin



Gambar 4.20 Hasil grafik 2D Compass, makin sering muncul makin banyak tanda panah

Setelah pembaca menggunakan ketiga jenis grafik, yaitu Waveform Chart, Waveform Graph dan 2D Compass, maka dapat diketahui fungsi khusus masing-masing grafik, seperti ditunjukkan dalam Tabel berikut ini.

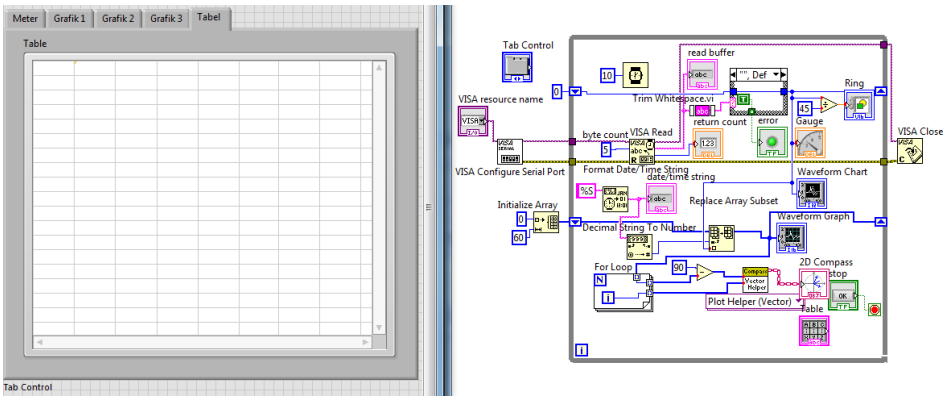
Tabel 4.1 Perbandingan Waveform Chart, Graph & 2D Compass

No	Item	Waveform Chart	Waveform Graph	2D Compass
1.	Input Data	Data Tunggal	Data Array	Data Array
2.	Histori Data	Sudah dilengkapi Buffer	Tidak ada Buffer, memerlukan Shift Register	Tidak ada Buffer, memerlukan Shift Register
3.	Fungsi Khusus	Pengamatan data Sensor secara langsung dan kontinyu	Pengamatan data Sensor dengan jumlah data dan interval tertentu	Pengamatan data Sensor dengan jumlah data dan interval tertentu serta sudut dan frekuensi

4.2 Menampilkan Tabel Data

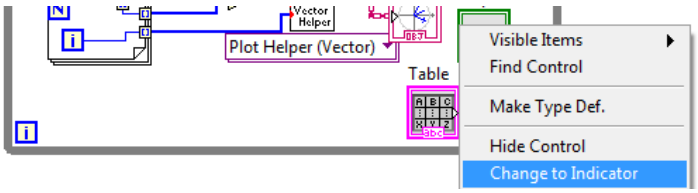
Berikut langkah-langkah untuk menampilkan tabel data sensor arah angin beserta dengan waktunya:

1. Pada kotak Tab Control, klik kanan Tab Grafik 3, kemudian pilih Add Page After, maka akan muncul Tab Grafik 4. Ubah nama Grafik 4 menjadi Tabel.
2. Ambil objek Table (Silver) di Palet Controls, di Silver, di List, Table & Tree, dan tempatkan di Tab Tabel.



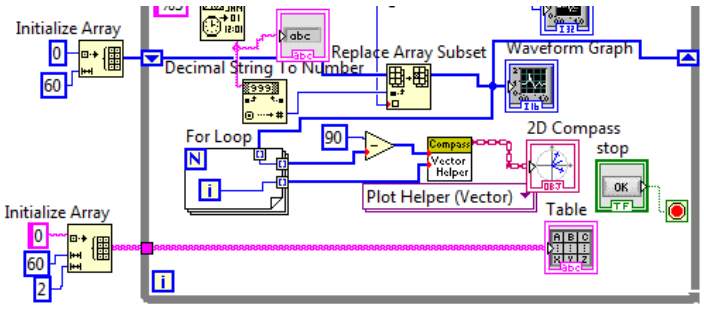
Gambar 4.21 Menempatkan objek Table pada Tab Tabel

3. Secara default, objek Table berupa Control (kaki di sisi kanan icon, atau kaki output). Karena objek Table di sini digunakan sebagai penampil data, maka harus diubah menjadi Indicator (kaki di sisi kiri icon, atau kaki input). Untuk mengubah Control menjadi Indicator, klik kanan objek Table, dan pilih Change to Indicator.



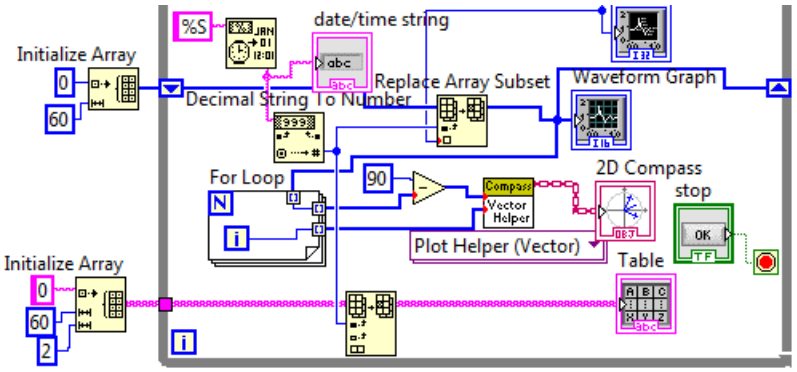
Gambar 4.22 Mengubah objek Table dari Control menjadi Indicator

4. Objek Table sebenarnya sama seperti Array 2D (Array dengan 2 index, yaitu baris dan kolom), dengan tipe data String. Untuk menampilkan data pada Table, perlu ditentukan dulu, berapa banyak baris dan kolom yang dibutuhkan. Apabila data yang ditampilkan adalah data waktu dan data sensor arah angin, maka dibutuhkan 60 buah baris (karena nilai detik dari 0 – 59) dan 2 buah kolom. Untuk menyiapkan 60 buah baris dan 2 buah kolom, gunakan Initialize Array dengan elemen diisi String Constant 0 dan dimension size diisi Numeric Constant 2. Agar menghasilkan Array 2D, Initialize Array harus memiliki 2 buah dimension size. Untuk memunculkan dimension size yang kedua, tarik kotak Initialize Array ke bawah. Setelah itu, hubungkan output Initialize Array dengan input icon Table, seperti ditunjukkan pada gambar berikut ini:



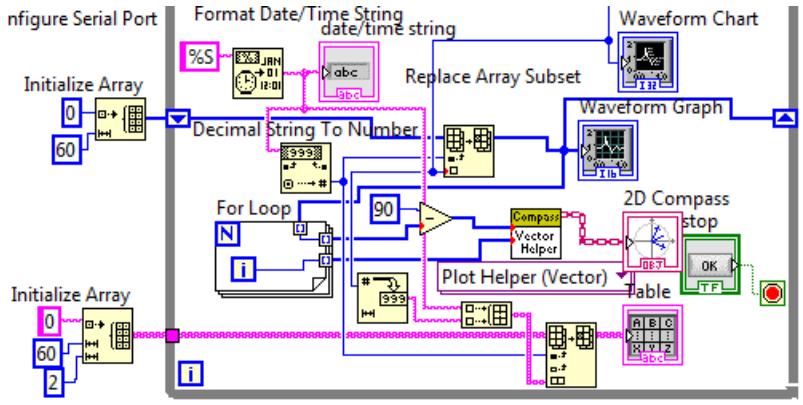
Gambar 4.23 Menyiapkan 60 baris dan 2 kolom dengan Initialize Array

5. Apabila program di atas dijalankan, maka pada Table akan muncul angka 0 di 2 kolom dari kiri ke kanan dan di 60 baris dari atas ke bawah. Untuk mengubah angka 0 tersebut menjadi data waktu di kolom pertama dan data sensor di kolom kedua, sisipkan icon Replace Array Subset.



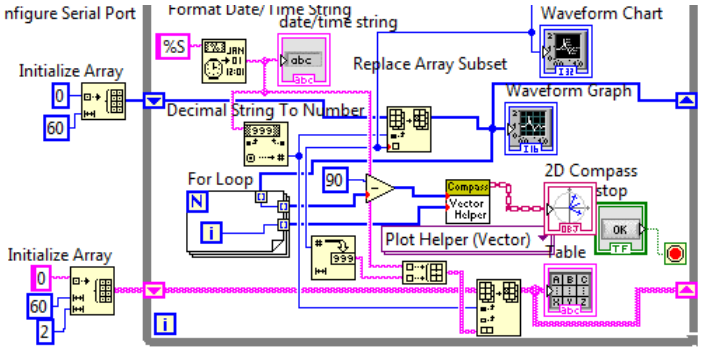
Gambar 4.24 Menyisipkan Replace Array Subset antara Table dan Initialize Array

6. Hubungkan kaki input index (row) dengan garis data waktu (Format Date/Time String) yang telah diubah ke desimal melalui icon Decimal String to Number. Untuk kaki input new element, tanda 2 kotak kecil menunjukkan bahwa input yang dibutuhkan adalah data array (bila 1 kotak kecil, inputnya adalah data tunggal). Untuk membuat data array, dapat dilakukan dengan menggunakan icon Build Array.



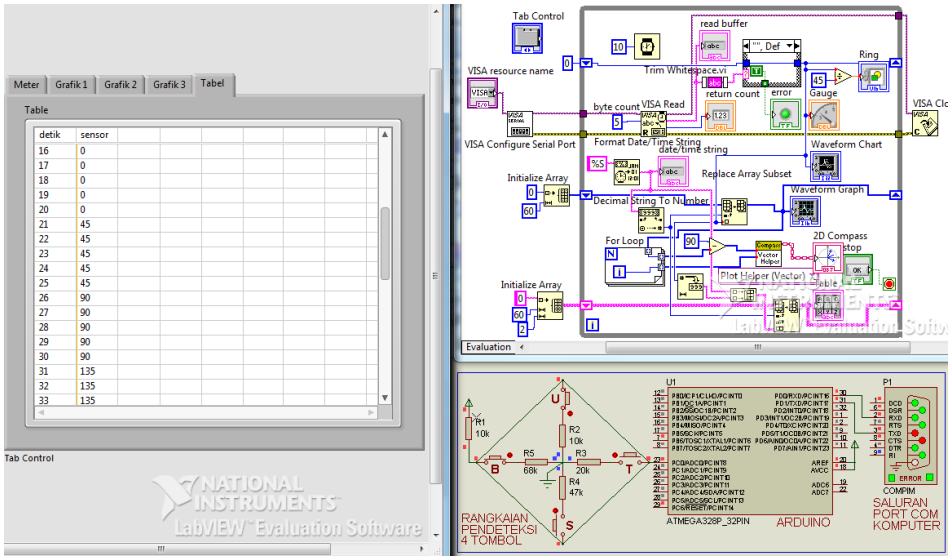
Gambar 4.25 Memberi input index (row) dan input new element dengan data array

7. Hubungkan kedua input icon Build Array dengan data waktu dan data sensor, dari Waveform Chart yang diubah ke String dengan icon Decimal String to Number. Data waktu dan data sensor tersebut akan ditempatkan di kolom pertama dan kedua, dengan urutan baris sesuai data waktu.



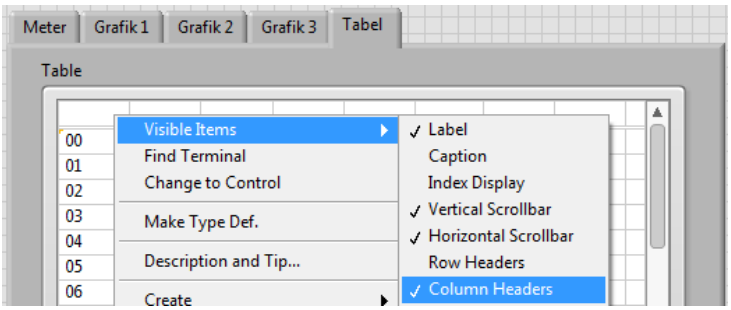
Gambar 4.26 Memberi input Build Array dari data waktu dan data sensor

8. Tekan tombol Run, maka pada Table akan muncul data waktu di kolom pertama dan data sensor di kolom kedua, dengan urutan data sesuai nilai waktu, yaitu nilai 0 di baris paling atas, dan nilai 59 di baris paling bawah.



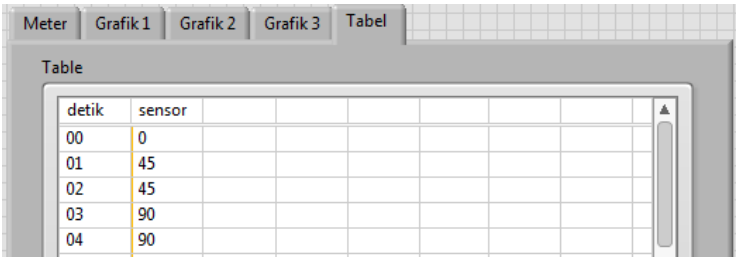
Gambar 4.27 Data waktu dan data sensor ditampilkan di Table

9. Untuk membuat tabel lebih jelas, perlu adanya kolom judul. Untuk itu klik kanan Table, pilih Visible Items, centang pada pilihan Column Headers.



Gambar 4.28 Centang Column Headers untuk menampilkan kolom judul

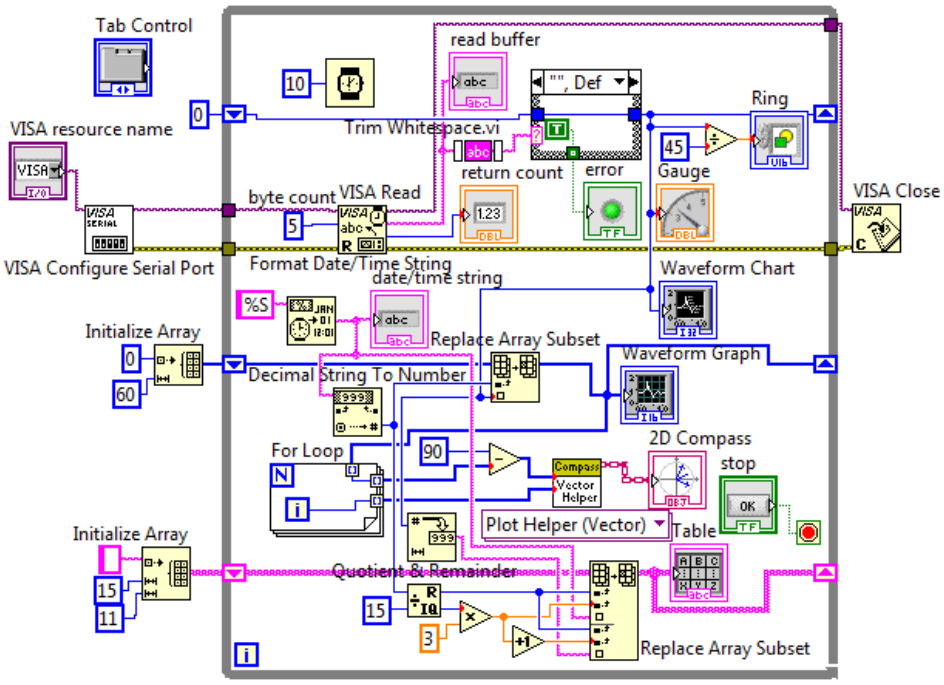
10. Beri judul pada kolom pertama dan kedua dengan “detik” dan “sensor”.



Gambar 4.29 Memberi judul pada kolom pertama dan kedua

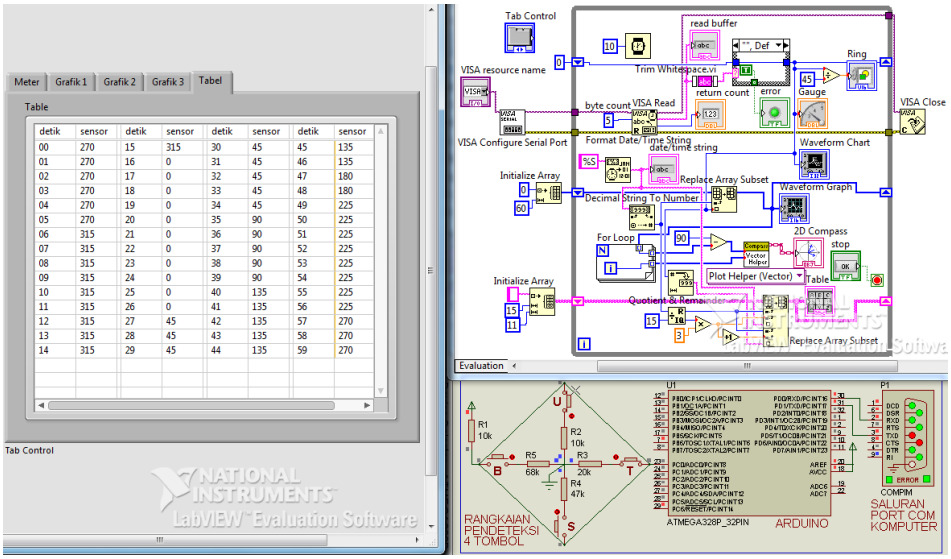
11. Jalankan program kembali, dan perhatikan bahwa setiap baris pada Table akan diupdate secara otomatis ketika nomor baris sesuai detik saat itu.
12. Tampilan tabel pada Gambar 4.29 adalah memanjang ke bawah. Untuk melihat data yang lain, perlu menggeser Scrollbar. Diinginkan tampilan tabel yang bisa menampilkan data secara penuh tanpa perlu menggeser Scrollbar, yaitu dengan cara memanfaatkan kolom yang kosong, sehingga 60 buah baris data dapat dibagi dan ditempatkan pada beberapa kolom yang kosong. Agar pembagian merata dan semua data terlihat, maka 60 buah baris data tersebut dibagi menjadi 4 bagian, dengan setiap kolomnya memiliki 15 baris data. Bagian pertama akan menempati kolom pertama dan kedua, bagian kedua akan menempati kolom keempat dan kolom kelima (satu kolom di antara kedua bagian sengaja dikosongkan), dan

seterusnya. Untuk membuat 60 baris menjadi 4 x 15 baris, maka perlu icon Quotient & Remainder. Ambil icon ini dari kategori Numeric, di Programming, di palet Functions. Hubungkan data waktu detik ke kaki input x, dan Create Constant di kaki input y. Ganti angka 0 dengan 15. Kemudian hubungkan kaki output R dengan index (row), dan kaki output IQ dengan index (col), yang sebelumnya dikalikan 3. Mengapa dikali 3? Karena ketika baris telah mencapai 15 dan kelipatannya, maka data berikutnya akan ditempatkan pada 3 kolom dari kolom sebelumnya. Ketika index (row) dan index (col) diisi, input new element akan berubah dari 2 kotak menjadi 1 kotak, yang berarti input datanya harus data tunggal bukan array. Padahal ada 2 data yang harus dimasukkan, yaitu data detik dan data sensor. Agar kedua data tersebut dapat dimasukkan, tarik ke bawah kotak Replace Array Subset, sehingga muncul 3 kaki input lagi, yaitu index (row), index (col) dan new element kedua. Hubungkan index (row) kedua dengan output R.



Gambar 4.30 Replace Array Subset untuk membuat data ditempatkan pada Table

13. Berikutnya, hubungkan output IQ yang telah dikali 3 dengan index (col) kedua, yang sebelumnya disisipi +1. Dengan penambahan 1 angka ini akan membuat kolom pada data kedua selalu ditempatkan 1 kolom di samping kanan data pertama. Untuk new element pertama, hubungkan dengan data detik dari icon Format Date/Time String, dan untuk new element kedua, hubungkan dengan data sensor dari Waveform Chart yang telah diubah ke dalam String dengan icon Number to Decimal String.
14. Setelah selesai, jalankan simulasi Proteus dan LabVIEW. Maka data akan dibagi menjadi 4 bagian, masing-masing 15 baris dan 2 kolom dengan 1 kolom pemisah antar bagian, agar semua data terlihat pada satu Table.

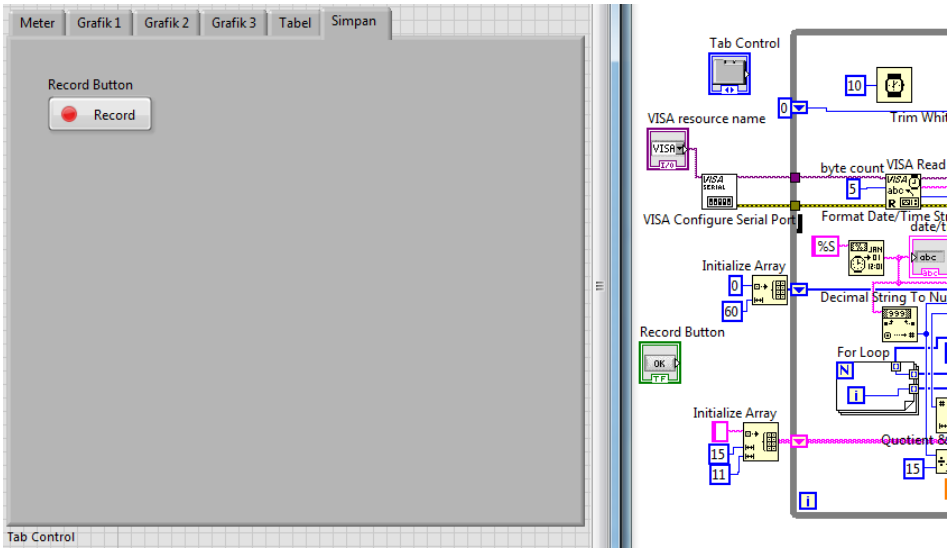


Gambar 4.31 Membuat data tertampil dalam 4 bagian per 15 baris dalam Table

4.3 Menyimpan Data

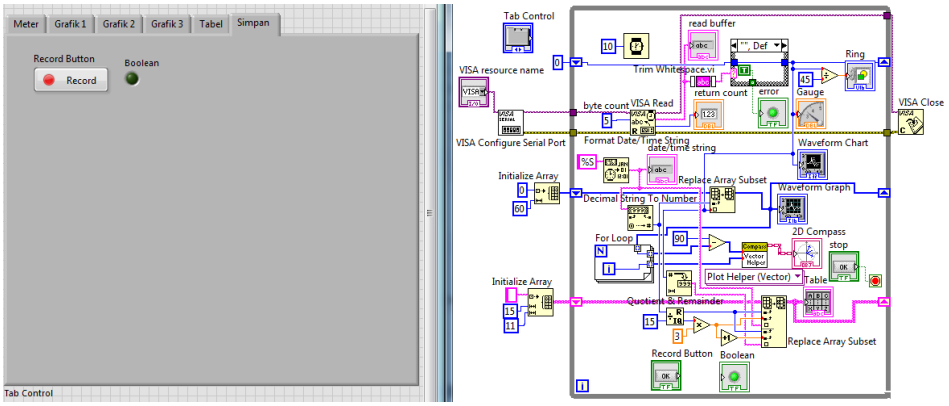
Data yang ditampilkan dalam bentuk grafik dan tabel di atas akan hilang begitu program LabVIEW dimatikan. Agar data tidak hilang ketika LabVIEW dimatikan, perlu program yang dapat menyimpan data. Berikut ini langkah-langkahnya:

1. Buat Tab Simpan di samping Tab Tabel. Kemudian tempatkan tombol Record Button, yang diambil dari palet Controls, di kategori Silver, Boolean, Buttons, dan Record Button.



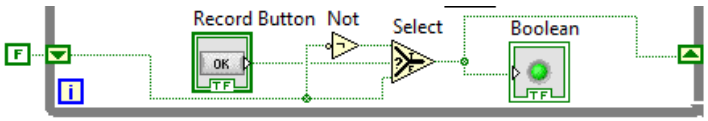
Gambar 4.32 Menempatkan tombol Record Button di Tab Simpan

2. Tambahkan Round LED dari palet Controls, Modern, Boolean, Round LED. Diinginkan, ketika tombol Record Button ditekan, maka penyimpanan dilakukan. Kemudian ketika tombol Record Button ditekan lagi, maka penyimpanan dihentikan. Demikian seterusnya.



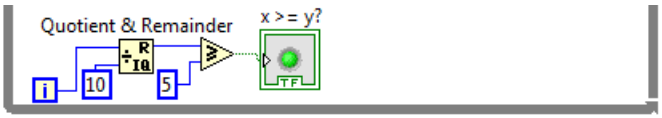
Gambar 4.33 Tempatkan Round LED sebagai indikator Tombol Record Button

3. Untuk membuat penyimpanan dijalankan dan dihentikan dengan satu tombol, maka tambahkan Shift Register. Beri nilai awal False (ambil dari kategori Boolean, yaitu di palet Functions, Programming, Boolean, False Constant). Kemudian tempatkan icon Not (ambil dari kategori Boolean) dan icon Select (ambil dari kategori Comparison). Hubungkan seperti Gambar 4.34. Arti dari program tersebut adalah, ketika tombol Record ditekan, maka Select akan memilih nilai dari input yang atas, sedangkan ketika tombol Record tidak ditekan, maka Select memilih nilai input yang bawah. Input atas akan selalu me-Not-kan kondisi pada Shift Register. Jadi jika Shift Register saat itu adalah False, maka nilai input atas adalah True. Sebaliknya ketika Shift Register saat itu adalah True, maka nilai input atas adalah False. Jadi saat tombol Record ditekan, LED Boolean akan dibalik kondisinya, dan dikunci hingga penekanan tombol Record berikutnya.



Gambar 4.34 Tombol Record akan me-NOT-kan kondisi setiap kali ditekan

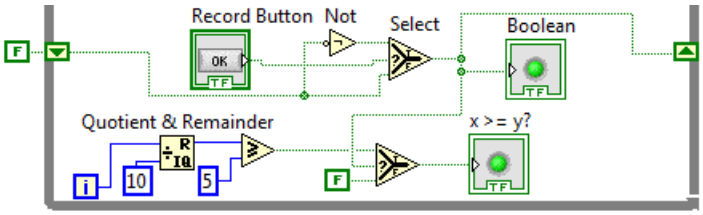
4. Agar saat proses penyimpanan benar-benar diketahui, diinginkan ada sebuah tambahan LED lagi yang akan berkedip ketika proses penyimpanan sedang berlangsung. Untuk membuat LED berkedip setiap detik, tambahkan icon Quotient & Remainder. Hubungkan input x dengan terminal i dan input y dengan 10. Mengapa 10? Karena While Loop akan melakukan perulangan setiap 100 milidetik. Agar bisa 1 detik, maka perlu 10 kali perulangan, yaitu 10 x 100 milidetik = 1 detik. Karena input y = 10 membuat nilai output R akan berkisar antara 0 sampai dengan 9. Agar bisa berkedip setiap detik, maka nilai R harus dibagi menjadi 2 bagian, yaitu ketika nilainya lebih besar atau sama dengan 5, LED menyala, dan ketika nilainya kurang dari 5, LED padam. Untuk itu, tambahkan icon Greater or Equal, dan hubungkan inputnya dengan output R dan angka 5, untuk membandingkan nilai output R dengan 5. Kemudian klik kanan pada kaki output icon Greater or Equal dan pilih Create Indicator agar muncul LED.



Gambar 4.35 Membuat LED berkedip setiap detik dengan Quotient & Remainder

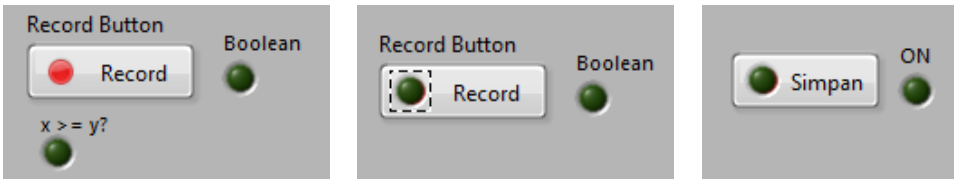
Catatan: Perhatikan bahwa waktu perulangan sebesar 100 milidetik pada While Loop tidak diperoleh dari icon Wait (ms). Karena icon Wait (ms) diisi dengan nilai 10 (lihat Gambar 4.32). Waktu perulangan sebesar 100 milidetik diperoleh dari icon VISA Read yang menunggu data dari komunikasi serial. Karena data dari komunikasi serial dikirim setiap 100 mili detik (lihat program Arduino), maka perulangan akan dilakukan setiap kali icon VISA Read telah selesai menerima data.

5. Agar LED berkedip hanya ketika proses penyimpanan dijalankan dan akan padam bila proses penyimpanan dihentikan, sisipkan icon Select di antara output icon Greater or Equal dan LED x>=y. Berikan output icon Greater or Equal pada kaki input atas (t) dan False Constant pada kaki input bawah (f), sedangkan kaki tengah (s) hubungkan dengan output LED Boolean, yang merupakan indikator tombol Record.



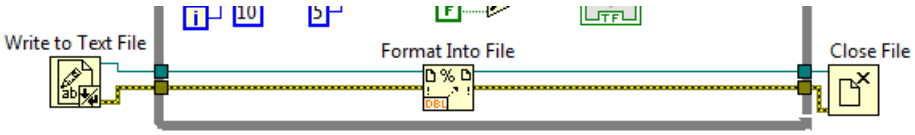
Gambar 4.36 Membuat LED berkedip hanya ketika LED Boolean menyala

6. Tempatkan LED berkedip di atas tombol Record, tepatnya menutupi bulatan merah pada tombol Record. Kemudian hilangkan Label LED berkedip dengan menghilangkan tanda centang pada Label, yaitu di Visible Items. Begitu juga untuk tombol Record, hilangkan Labelnya. Kemudian ubah kata Record menjadi simpan, dan kata Boolean menjadi ON, seperti ditunjukkan pada Gambar 4.37 berikut.



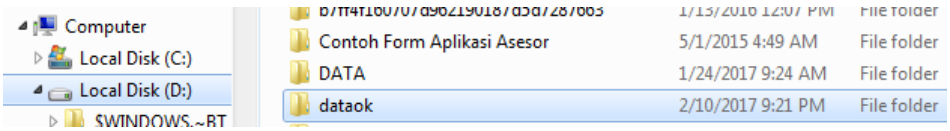
Gambar 4.37 Dari kiri ke kanan: LED kedip ditempatkan di tombol Record/Simpan

7. Seharusnya ketika tombol Record/Simpan ditekan, LED ON akan menyala, sedangkan LED di kiri kata Simpan akan berkedip. Kemudian ketika tombol Record/Simpan ditekan lagi, seharusnya LED ON akan padam, dan LED kedip juga padam. Proses berulang kembali ketika tombol ditekan lagi.
8. Setelah tombol Simpan dan indikatornya selesai dibuat, langkah berikutnya adalah menempatkan instruksi penyimpanan. Ambil 3 buah icon, yaitu Write to Text File, Format Into File dan Close File, yang ketiganya diambil dari palet Functions, kategori File I/O. Tempatkan seperti gambar berikut, dan hubungkan ketiganya.



Gambar 4.38 Menempatkan icon Write to Text File, Format Into File dan Close File

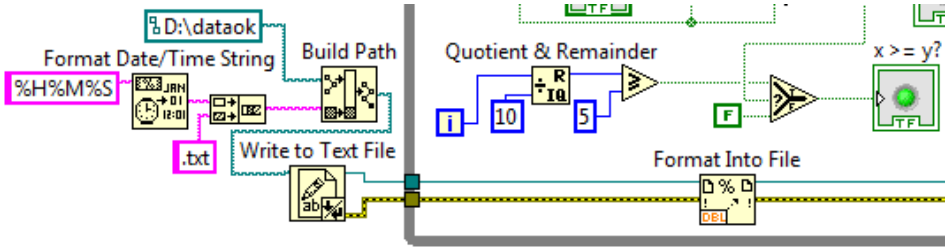
9. Agar semua file hasil penyimpanan terlokalisir, maka buat folder baru di Drive Data, contoh Drive D, dan beri nama folder dataok seperti berikut:



Gambar 4.39 Membuat folder dataok untuk tempat data hasil penyimpanan

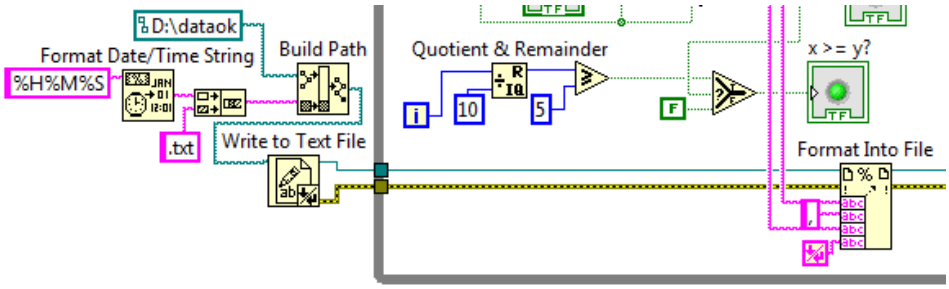
10. Agar nama file penyimpanan yang dihasilkan setiap kali bisa berbeda, maka sebaiknya nama file diambil dari nilai waktu jam, menit dan detik saat itu. Untuk itu pada kaki input File di Write to Text File, hubungkan dengan icon Format Date/Time String, dengan kaki input Time Format diberi karakter Constant (Create Constant) %H%M%S. Kemudian agar tipe

file berbentuk text file, tambahkan .txt pada nama file, yang disatukan melalui icon Concatenate String. Kemudian agar file hasil penyimpanan tersebut dapat ditempatkan dalam folder dataok di Drive D, tambahkan icon Build Path, dan klik kanan kaki input base path, pilih Create Constant, dan ketikkan kata D:\dataok, dan hubungkan output Concatenate String dengan kaki input name path, seperti ditunjukkan pada gambar berikut.



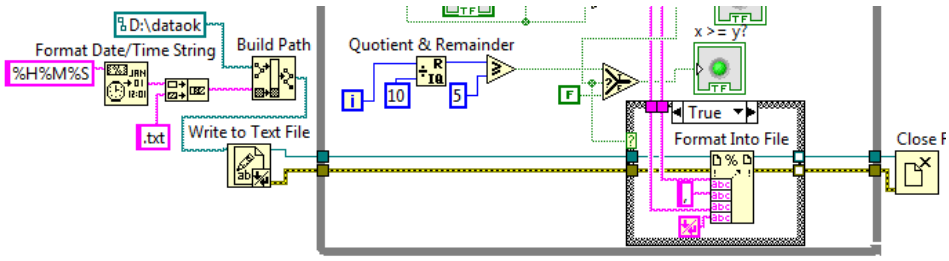
Gambar 4.40 Mengatur lokasi penyimpanan dan nama file (dibuat sesuai waktu)

11. Setelah pengaturan lokasi dan nama file, langkah berikutnya adalah mengisi file dengan data. Karena data yang akan disimpan ada 2 data, dan di tengahnya diberi tanda koma, dan di akhir data diberi Enter, maka buat agar icon Format Into File memunculkan 3 buah kaki input lagi, yaitu dengan cara menarik kotak icon tersebut ke bawah. Kemudian kaki input 1, hubungkan dengan garis data waktu detik (String), dan kaki input 2 Create Constant, isi dengan tanda koma. Kemudian kaki input 3, hubungkan dengan garis data sensor (String), dan terakhir kaki input 4, hubungkan dengan End of Line Constant (karakter Enter), yang diambil dari palet Functions, di Programming, di kategori String.



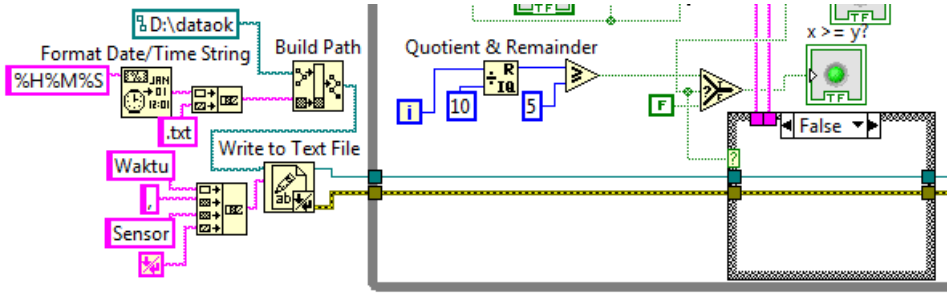
Gambar 4.41 Masukkan data waktu, koma, sensor dan Enter, ke Format Into File

12. Agar proses penyimpanan dilakukan hanya ketika LED ON hidup, tambahkan Case Structure pada icon Format Into File, dan hubungkan terminal selector (?) ke garis data LED ON. Seharusnya icon Format Into File berada pada kondisi True dari Case Structure. Kemudian pada kondisi False, teruskan garis data refnum dan error dari icon Write to Text File ke icon Close File seperti ditunjukkan pada Gambar 4.42.



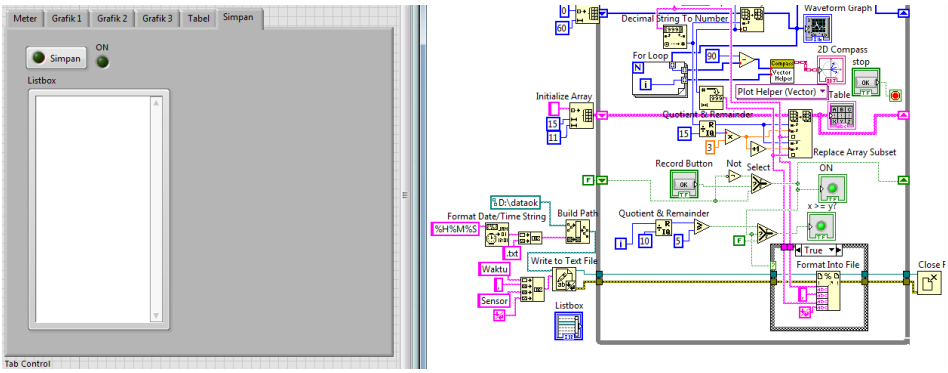
Gambar 4.42 Tambahkan Case Structure pada Format Into File

13. Agar isi file memiliki header/judul pada awalan, tambahkan kata Waktu, koma, Sensor dan Enter, dan gabungkan dengan icon Concatenate String. Kemudian hubungkan output Concatenate String dengan kaki input text pada icon Write to Text File.

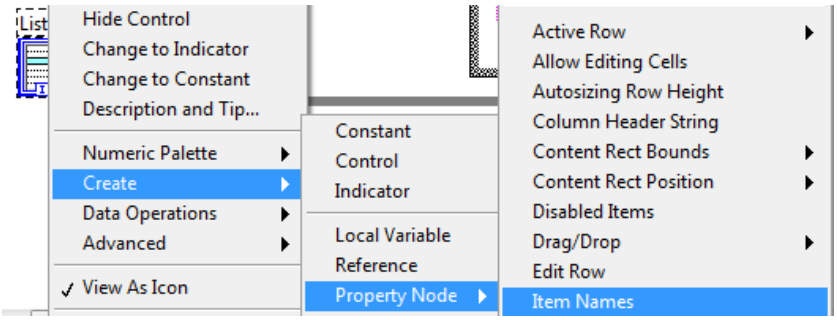


Gambar 4.43 Menambahkan judul di isi file dengan Waktu, koma, Sensor, Enter

14. Kemudian agar isi data yang disimpan ke file dapat diketahui saat proses penyimpanan berlangsung, tambahkan sebuah Listbox (Silver), yang diambil dari Palet Controls, Silver, di kategori List Table & Tree.
15. Di Block Diagram, klik kanan icon Listbox, pilih Create, pilih Property Node, pilih Item Names, seperti ditunjukkan pada Gambar 4.45.

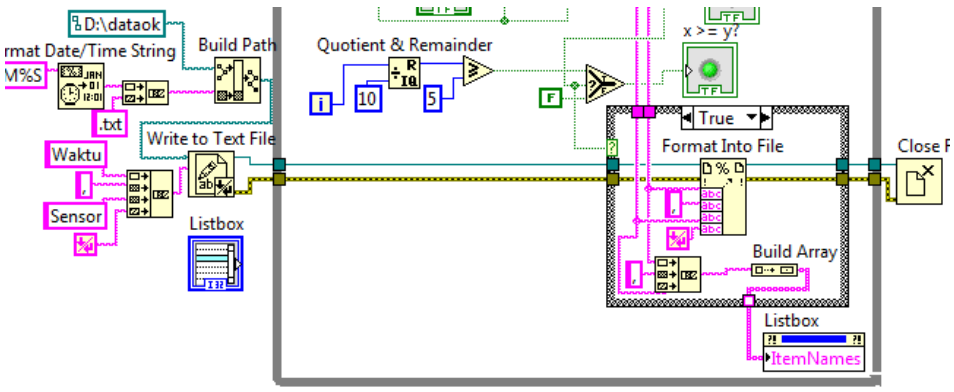


Gambar 4.44 Menambahkan Listbox untuk menampilkan data yang disimpan



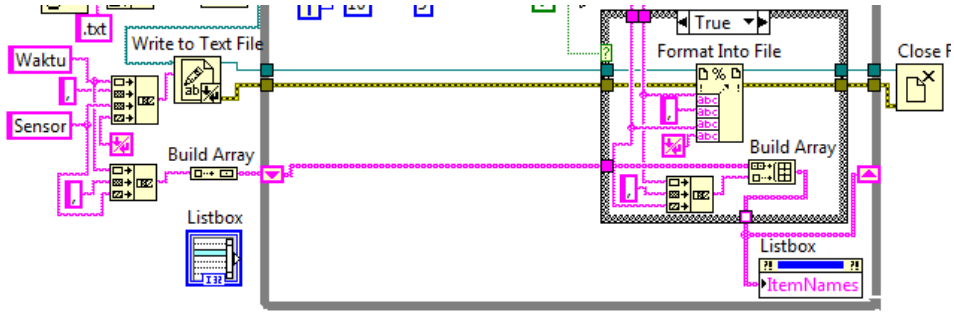
Gambar 4.45 Klik kanan Listbox, Create Property Node, pilih Item Names

16. Icon Item Names berisi data array. Untuk menampilkan data Waktu dan Sensor pada Listbox, masukkan data tersebut ke dalam icon Item Names. Berhubung secara default Item Names memiliki kaki output, agar dapat diberi input (memiliki kaki input), ubah Item Names dengan Change to Write. Kemudian pada kondisi True di Case Structure, gabungkan data Waktu, koma dan Sensor dengan Concatenate String, dan ubah data String tersebut menjadi Array. Untuk mengubah data dari String menjadi Array, gunakan icon Build Array. Hubungkan output Build Array ke input icon Item Names, seperti ditunjukkan pada Gambar 4.46.
17. Agar Listbox dapat menyimpan data sebelumnya, tambahkan Shfit Register. Beri nilai awal pada Shift Register tersebut dengan gabungan kata Waktu, koma dan Sensor oleh Concatenate String, yang kemudian diubah ke dalam data Array dengan icon Build Array.



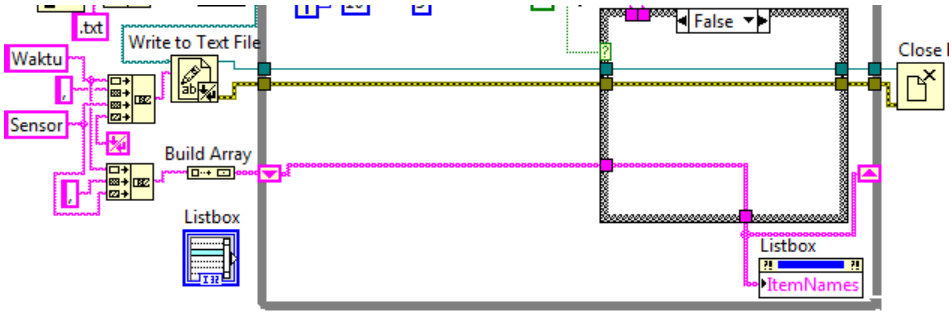
Gambar 4.46 Menampilkan data Waktu dan Sensor ke Listbox melalui Item Names

18. Pada icon Build Array di dalam kotak Case Structure, tarik kotak tersebut ke atas agar muncul kaki input yang baru (kaki input yang baru di atas kaki input yang terhubung dengan Concatenate String). Hubungkan kaki input yang baru tersebut dengan terminal Shift Register kiri. Kemudian hubungkan garis data Listbox dengan terminal Shift Register kanan.

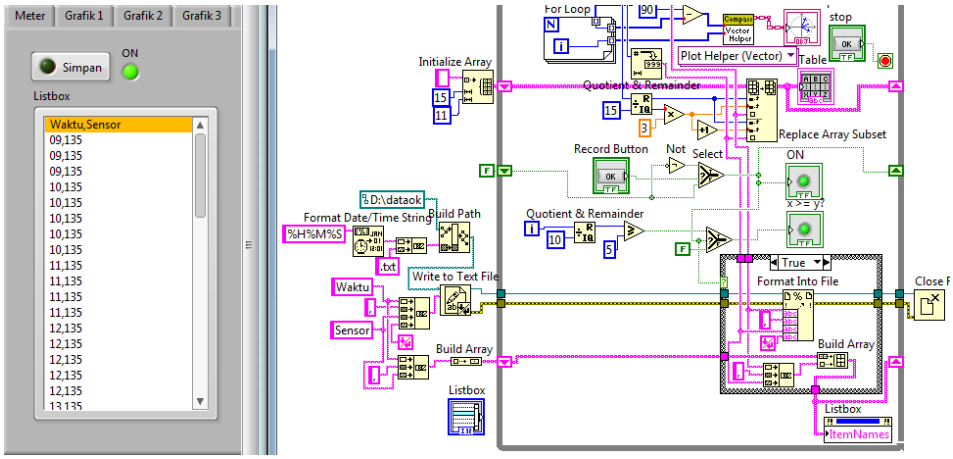


Gambar 4.47 Tambahkan Shift Register pada Listbox agar dapat menyimpan data

19. Pada kondisi False di Case Structure, sambungkan garis data dari terminal Shift Register kiri dengan kaki input Item Names Listbox.
20. Jalankan simulasi Proteus, dan tekan tombol Run LabVIEW. Buka Tab Simpan, dan tekan tombol Simpan. Seharusnya LED ON menyala dan LED di tombol Simpan berkedip. Juga di Listbox seharusnya menampilkan data dengan header/judul bertuliskan “Waktu, Sensor”, diikuti data Waktu dan Sensor di baris berikutnya seperti ditunjukkan pada Gambar 4.48.



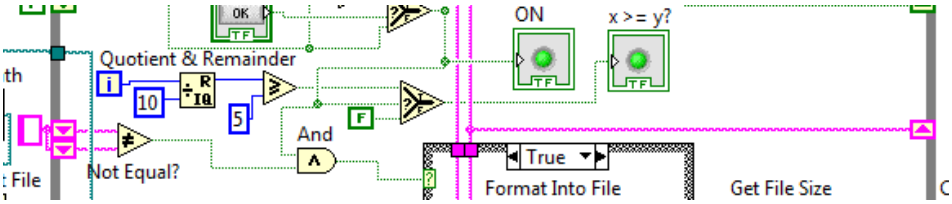
Gambar 4.48 Sambungkan garis data dari Shift Register kiri dengan Item Names



Gambar 4.49 Listbox menampilkan data Waktu & Sensor yang disimpan

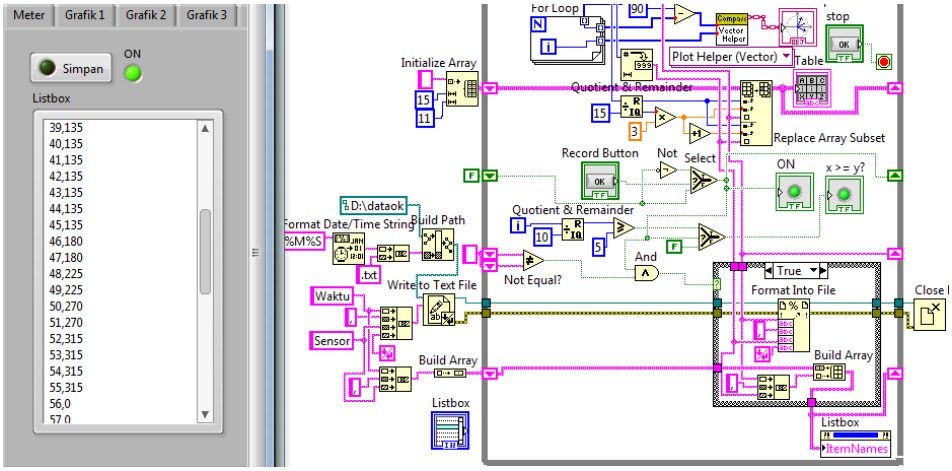
21. Terlihat dari Listbox Gambar 4.49 di atas, bahwa data yang disimpan ke dalam file tidak konsisten waktunya, kadang dalam 1 detik yang sama, ada 5 data dan kadang 4 data. Agar data waktu lebih konsisten, dan juga agar ukuran file yang dihasilkan tidak terlalu besar, maka diinginkan penyimpanan dilakukan per detik. Untuk membuat hal itu, tambahkan Shift Register untuk data waktu. Hubungkan data waktu detik dengan terminal Shift Register di kanan. Kemudian pada terminal Shift Register kiri, tarik kotak terminal tersebut ke bawah, hingga muncul 2 kotak terminal. Masing-masing kotak terminal, beri nilai awal String Constant (klik kanan terminal dan Create Constant). Kemudian tambahkan icon Not Equal untuk membandingkan data yang baru dengan data sebelumnya.

Apabila data detik yang baru tidak sama dengan data detik yang sebelumnya, dan LED ON menyala, maka proses penyimpanan dijalankan. Sebaliknya apabila detik yang baru dengan detik yang sebelumnya masih sama, atau LED belum ON, maka proses penyimpanan akan berhenti.



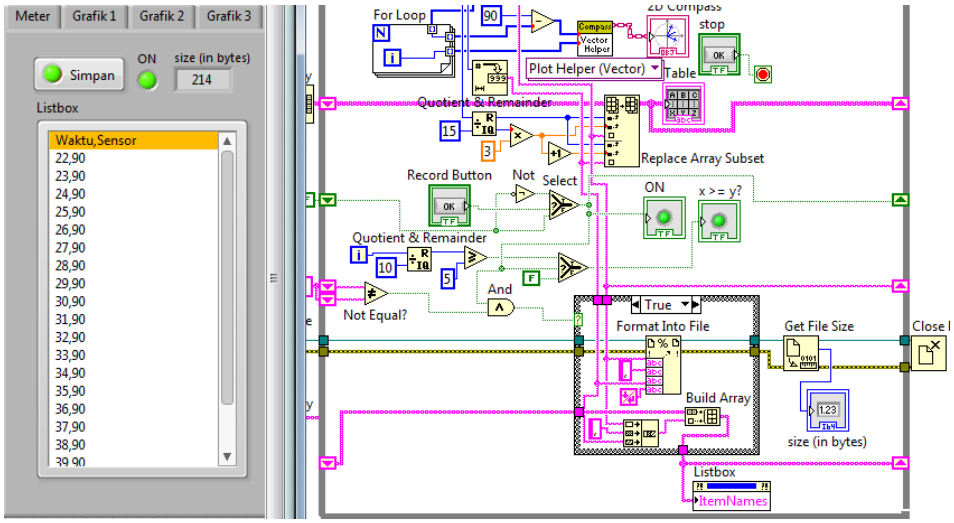
Gambar 4.50 Membuat penyimpanan dijalankan bila detik berubah dan LED ON

22. Jalankan simulasi Proteus dan LabVIEW. Tekan tombol Simpan, maka LED ON akan menyala dan LED di tombol Simpan berkedip. Seharusnya Listbox menampilkan data Waktu dan data Sensor perdetik, seperti Gambar 4.51.



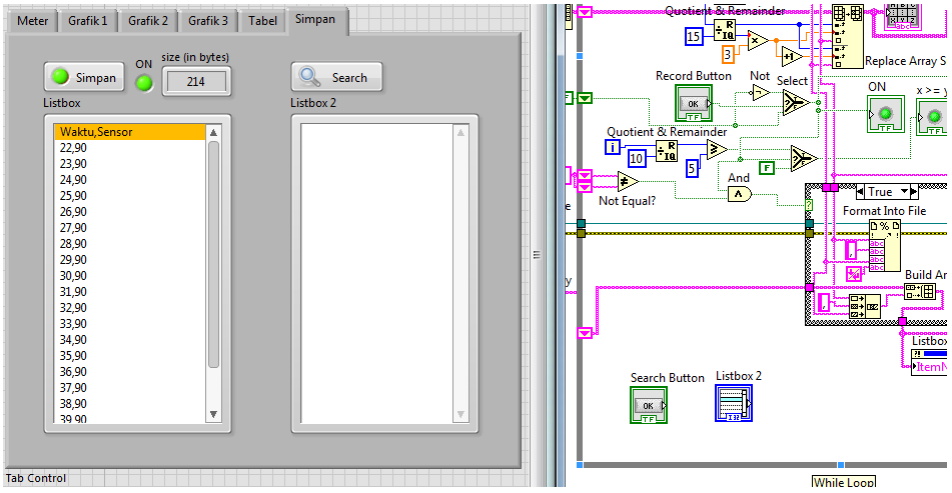
Gambar 4.51 Tampilan data di Listbox menunjukkan data Waktu & Sensor perdetik

23. Diinginkan pada saat proses penyimpanan, ukuran file hasil penyimpanan (dalam byte) dapat diketahui. Untuk itu sisipkan pada garis data refnum dan error setelah keluar dari Case Structure, icon Get File Size. Kemudian pada kaki output icon tersebut, klik kanan, pilih Create Indicator.



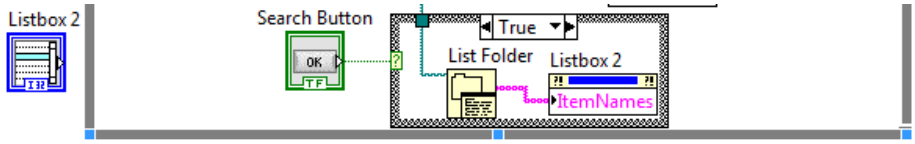
Gambar 4.52 Menyisipkan icon Get File Size untuk menampilkan ukuran file

24. Berikutnya, diinginkan daftar nama-nama file hasil penyimpanan dapat ditampilkan, dan sewaktu-waktu file tersebut dapat dibuka dengan hanya meng-klik 2 kali pada nama file yang ditampilkan tersebut. Untuk itu pada Front Panel, di Tab Simpan, tambahkan objek Listbox kedua, dan sebuah tombol Search Button, yang diambil dari Palet Controls, kategori Silver, kategori Boolean, kategori Buttons, pilih Search Button.



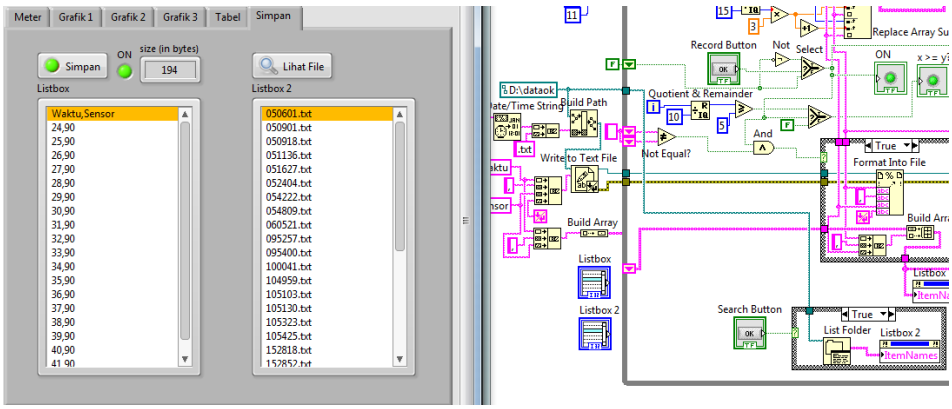
Gambar 4.53 Menambahkan Listbox kedua untuk menampilkan daftar nama file

25. Untuk menampilkan daftar nama file pada Listbox kedua, caranya adalah dengan memasukkan daftar nama file ke Item Names Listbox 2. Untuk mendapatkan daftar nama file, ambil icon List Folder, dari palet Functions, Programming, File I/O, Advanced File Functions, dan pilih List Folder. Hubungkan kaki input Path List Folder dengan nama path di Drive D, yaitu D:\dataok. Kemudian pada kaki output Filenames List Folder, hubungkan dengan Item Names Listbox 2. Untuk memunculkan Item Names, klik kanan pada Listbox 2, pilih Create, pilih Property Node, pilih Item Names. Agar dapat diberi input, klik kanan Item Names, dan pilih Change to Write. Agar Listbox 2 tidak menampilkan daftar nama terus-menerus, maka tambahkan Case Structure, dan hubungkan terminal selector (?) dengan icon Search Button. Dengan tambahan Case Structure ini, Listbox2 hanya menampilkan daftar nama apabila tombol Search Button ditekan.



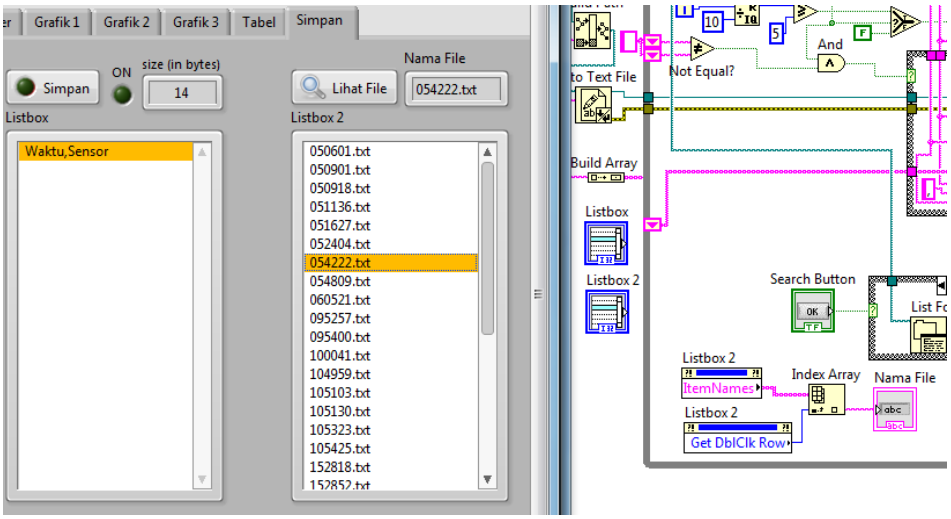
Gambar 4.54 Listbox2 menampilkan daftar nama file bila Search Button ditekan

26. Ubah nama tombol Search menjadi Lihat File. Jalankan simulasi Proteus dan program LabVIEW. Buka Tab Simpan, tekan tombol Lihat File, maka seharusnya pada Listbox 2 muncul daftar nama file hasil penyimpanan.



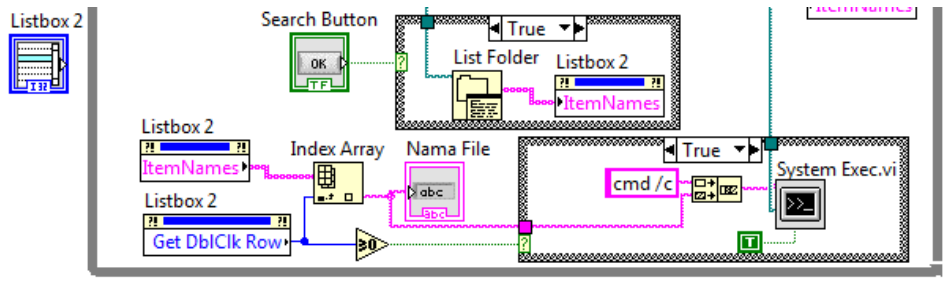
Gambar 4.55 Muncul daftar nama file pada Listbox2 saat tombol Lihat File ditekan

27. Setelah daftar nama file dapat ditampilkan di Listbox 2, langkah berikutnya adalah membuat file tersebut dapat dibuka dengan hanya meng-klik 2 kali namanya di Listbox 2. Untuk itu tambahkan Get Double-Clicked Row Listbox 2, yang dimunculkan dengan cara meng-klik kanan icon Listbox 2 di Block Diagram, dan pilih Create, pilih Invoke Node, pilih Get Double-Clicked Row. Icon Get Double-Clicked Row ini akan menghasilkan sebuah angka yang sesuai dengan nomor baris pada nama file yang di-klik 2 kali. Sebagai contoh, pada Gambar 4.56 berikut, untuk nama file 054222.txt, yang berada pada nomor baris ketujuh. Ketika nama file 054222.txt di Listbox 2 ini diklik 2 kali, icon Get Double-Clicked Row Listbox 2 akan menghasilkan angka 6 (karena baris pertama dimulai dengan angka 0). Dengan mengetahui bahwa icon Get Double-Clicked Row menghasilkan nomor baris, maka untuk mendapatkan nama file yang di-klik 2 kali, tambahkan Item Names Listbox 2 (klik kanan Listbox 2, pilih Create, pilih Property Node, pilih Item Names). Dengan menambahkan icon Index Array, dan menghubungkan kaki input array dengan output Item Names, dan kaki index dengan output Get Double-Clicked Row, akan dihasilkan output Index Array berupa nama file. Klik kanan pada output Index Array, pilih Create Indicator, maka akan muncul String Indicator.



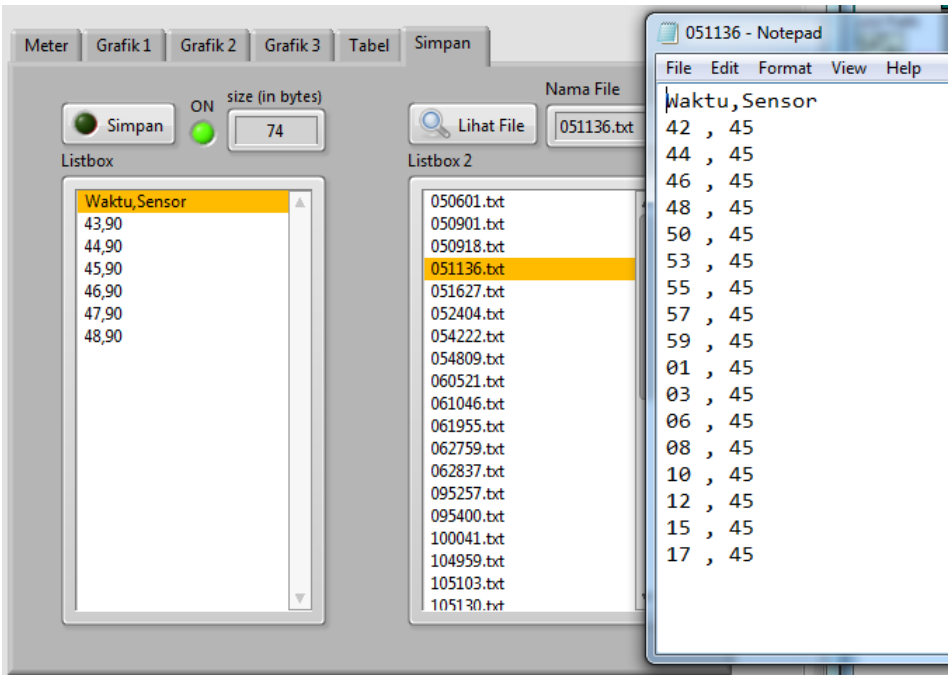
Gambar 4.56 Nama file muncul di String Indicator ketika namanya di-klik 2 kali

28. Ubah label String Indicator menjadi Nama File. Jalankan simulasi Proteus dan program LabVIEW. Buka Tab Simpan dan tekan tombol Lihat File, maka seharusnya muncul daftar nama file. Klik 2 kali pada salah satu nama file, maka seharusnya nama file tersebut muncul pada String Indicator Nama File. Untuk dapat memanggil dan membuka file tersebut, tambahkan icon System Exec.vi, yang diambil dari Palet Functions, pilih Connectivity, pilih Libraries & Executables, pilih System Exec.vi. Pada kaki input command line di icon System Exec, berikan String constant yang berisi "cmd /c" diikuti dengan nama file yang akan dibuka. Gunakan icon Concatenate String untuk menggabungkan keduanya. Pada kaki input working direktctory, hubungkan dengan path di Drive D, yaitu D:\dataok. Kemudian agar jendela command tidak tampak, berikan nilai True pada kaki input run minimized. Terakhir, System Exec.vi hanya memanggil dan membuka file ketika nama file di Listbox di-klik 2 kali, yaitu ketika icon Get Double-Clicked Row bernilai lebih dari atau sama dengan 0. Untuk itu tambahkan Case Structure, dan hubungkan terminal Selector (?) dengan icon Greater or Equal yang mendapat data dari Get Double-Clicked Row.

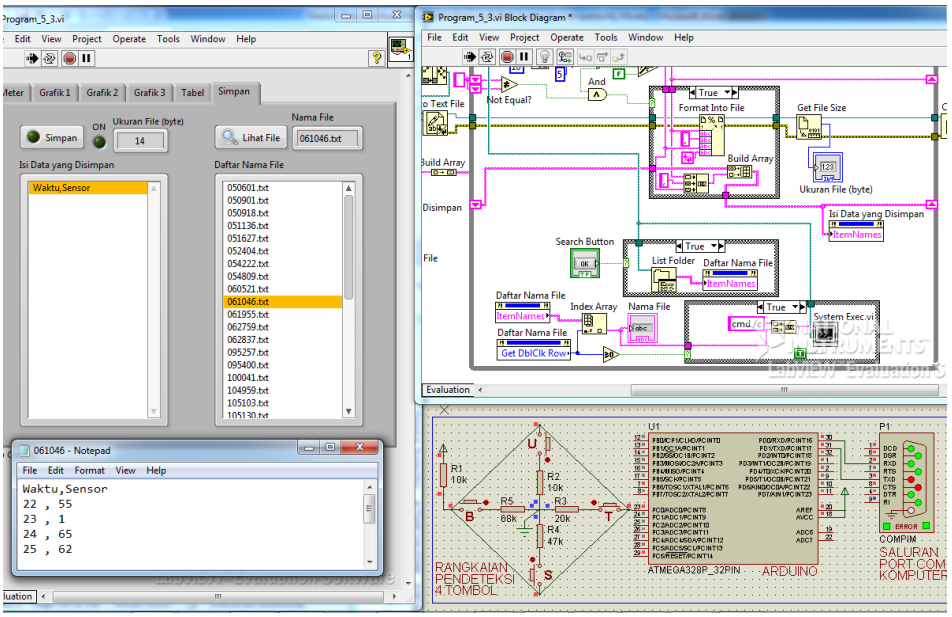


Gambar 4.57 Menambahkan System Exec.vi untuk memanggil dan membuka file

29. Jalankan simulasi Proteus dan program LabVIEW, tekan tombol Lihat File. Kemudian dari daftar nama file yang muncul di Listbox 2, klik 2 kali salah satu nama file di Listbox 2 tersebut. Seharusnya file tersebut terbuka seperti ditunjukkan pada Gambar 4.58. Tampak bahwa file 051136.txt terbuka setelah namanya di Listbox 2 di-klik2 kali. Ubah nama Listbox menjadi Isi Data yang Disimpan dan Listbox2 menjadi Daftar Nama File.



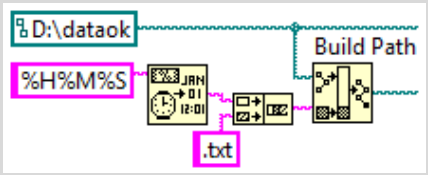
Gambar 4.58 Membuka file 051136.txt dengan meng-klik 2x namanya di Listbox2



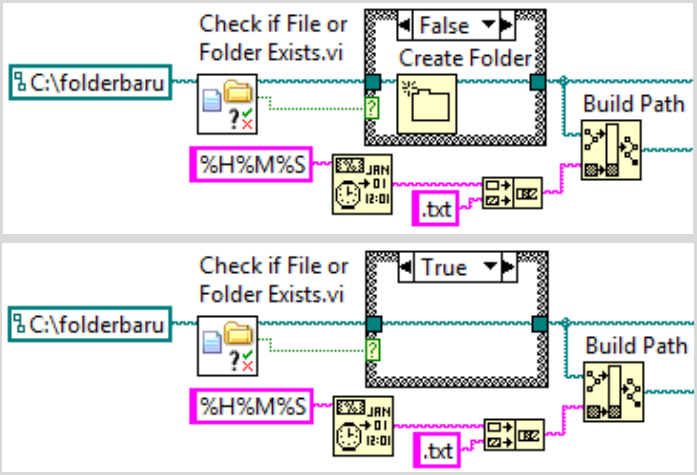
Gambar 4.59 Mengubah nama Listbox dg Isi Data dan Listbox2 dg Daftar Nama

Catatan: Pada langkah no. 9 di atas, pembaca diminta untuk membuat folder baru bernama dataok di Drive D, untuk lokasi file data tersimpan. Tentu saja hal ini cukup merepotkan. Untuk itu, agar program dapat menciptakan folder baru sendiri secara otomatis, berikut langkah-langkah modifikasinya:

1. Buka jendela Block Diagram, dan perhatikan icon Build Path di luar While Loop, yang terhubung dengan nama path D:\ dataok, seperti gambar berikut:



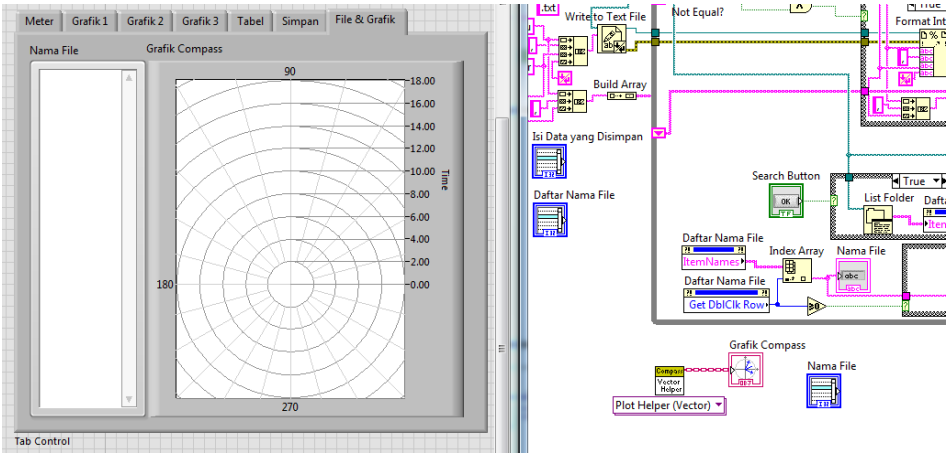
2. Agar secara otomatis program dapat membuat folder sendiri, sisipkan icon Create Folder antara nama path dengan icon Build path. Icon Create Folder ini dapat pembaca ambil dari palet Functions, Programming, File I/O, Advanced File Functions. Hanya saja, ketika program ini dijalankan untuk yang kedua kali, akan muncul error yang mengatakan bahwa terjadi duplikasi path karena path atau folder dengan nama tersebut sudah pernah dibuat. Untuk mengatasi hal ini, tambahkan sebuah fungsi yang menanyakan, apakah sebuah path atau folder sudah ada atau belum. JIka belum ada, maka Create Folder, jika sudah ada, jangan Create Folder. Untuk itu gunakan icon Check if File or Folder Exists dan Case Structure, seperti ditunjukkan gambar berikut:



4.4 Menampilkan Grafik dari File

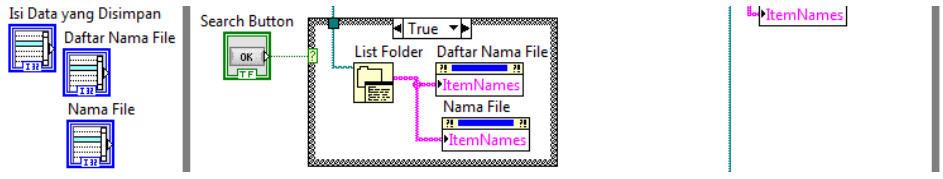
Setelah data dari Sensor dapat disimpan dalam bentuk File, diinginkan data di dalam File tersebut tidak hanya dapat dibuka dan ditampilkan dalam bentuk teks, tetapi dapat ditampilkan dalam bentuk grafik. Berikut ini langkah-langkahnya:

1. Buat sebuah Tab baru di samping Tab Simpan, dan beri nama Tab tersebut File & Grafik. Tempatkan pada Tab File & Grafik tersebut sebuah Listbox dan 2D Compass. Ubah nama Listbox menjadi Nama File dan 2D Compass menjadi Grafik Compass. Hilangkan Plot Legend pada 2D Compass dengan menghilangkan centang Plot Legend pada Visible Items.



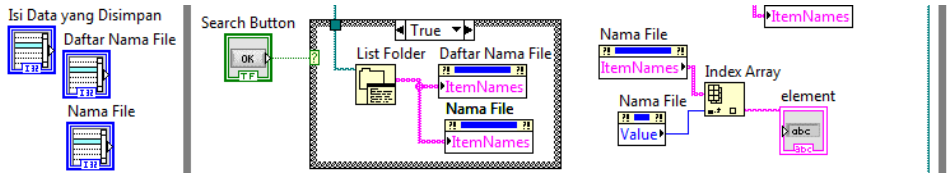
Gambar 4.60 Tambahkan Tab File & Grafik, isi dengan Listbox dan 2D Compass

2. Di Block Diagram, klik kanan icon Nama File, pilih Create, pilih Property Node, pilih Item Names. Agar Item Names dapat diberi input, klik kanan Item Names, pilih Change to Write. Tempatkan Item Names di dalam Case Structure Search Button, dan hubungkan dengan output List Folder.



Gambar 4.61 Hubungkan Item Names Nama File dengan output List Folder

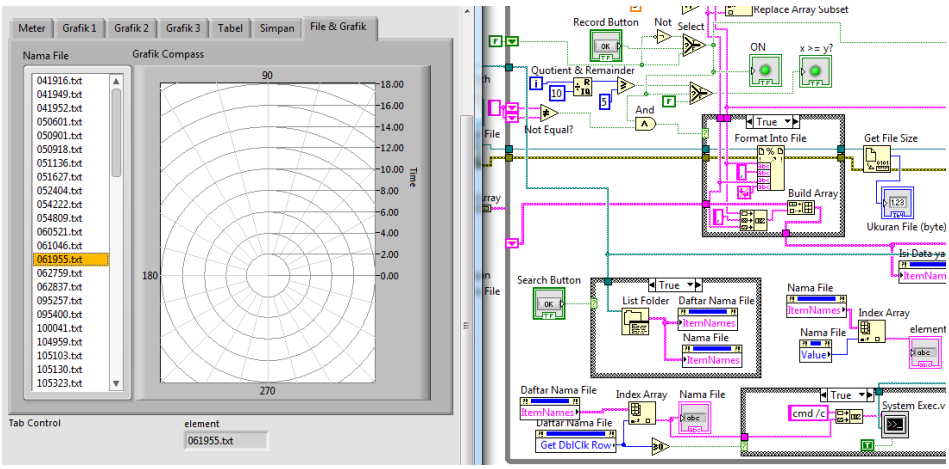
3. Maksud dari program pada langkah no. 2 di atas adalah, ketika tombol Search Button (Lihat File) ditekan, maka tidak hanya Listbox Daftar Nama File, Listbox Nama File di Tab File & Grafik juga terisi nama-nama file yang sama dengan Listbox Daftar Nama File di Tab Simpan. Diinginkan nantinya, Grafik 2D Compass langsung menampilkan data dari file yang dipilih di Listbox Nama File. Pemilihan dilakukan tidak perlu dengan meng-klik 2 kali, cukup sekali klik, atau hanya dengan menggerakkan cursor ke atas ke bawah dengan tombol panah atas bawah di keyboard. Untuk itu tambahkan lagi Item Names Listbox Nama File, kemudian Index Array. Hubungkan kaki input array Index Array dengan output Item Names. Kemudian munculkan Property Node Value Listbox Nama File, dengan cara meng-klik kanan Listbox Nama File, pilih Create, pilih Property Node, pilih Value. Hubungkan output Value dengan kaki index Index Array. Untuk menampilkan nama file yang sedang dipilih di Listbox Nama File, klik kanan pada kaki output Index Array, dan pilih Create, pilih Indicator.



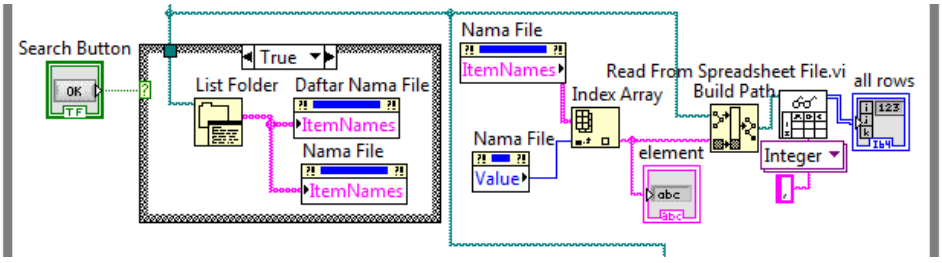
Gambar 4.62 Menampilkan nama file yang sedang dipilih dari Listbox Nama File

4. Jalankan simulasi Proteus dan tekan tombol Run LabVIEW. Buka Tab File & Grafik, tampak bahwa Listbox Nama File masih kosong, karena belum diisi. Untuk mengisinya dengan nama-nama file, buka Tab Simpan, tekan tombol Lihat File. Buka kembali Tab File & Grafik, maka seharusnya Listbox Nama File sudah terisi nama-nama file. Klik pada salah satu nama file di Listbox Nama File, dan perhatikan tampilan pada String Indicator element. Pindahkan cursor pilihan ke atas ke bawah dengan tombol panah di keyboard, seharusnya element menampilkan nama file pada cursor pilihan seperti ditunjukkan pada Gambar 4.63.
5. Setelah String Indicator element dapat menampilkan nama file yang dipilih, maka langkah berikutnya adalah menampilkan grafik dari data

pada file yang dipilih tersebut. Untuk itu tambahkan icon Read From Spreadsheet File.vi, yang diambil dari palet Functions, Programing, File I/O. Hubungkan kaki input file path icon Read From Spreadsheet File.vi tersebut dengan alamat file yang dipilih, yaitu dengan bantuan icon Build Path. Hubungkan kaki input base path icon Build Path dengan path di Drive D, yaitu D:\dataok, dan kaki input name icon Build Path dengan nama file di icon element. Berhubung data pada file dipisahkan dengan tanda koma, beri input pada kaki delimiter icon Read From Spreadsheet File.vi dengan String Constant, dan isi atau ketikkan tanda koma. Ubah juga pilihan tipe data dari Double menjadi Integer. Klik kanan kaki output all rows (yang paling atas) dan pilih Create, pilih Indicator, maka akan muncul icon array all rows, seperti ditunjukkan pada Gambar 4.64.

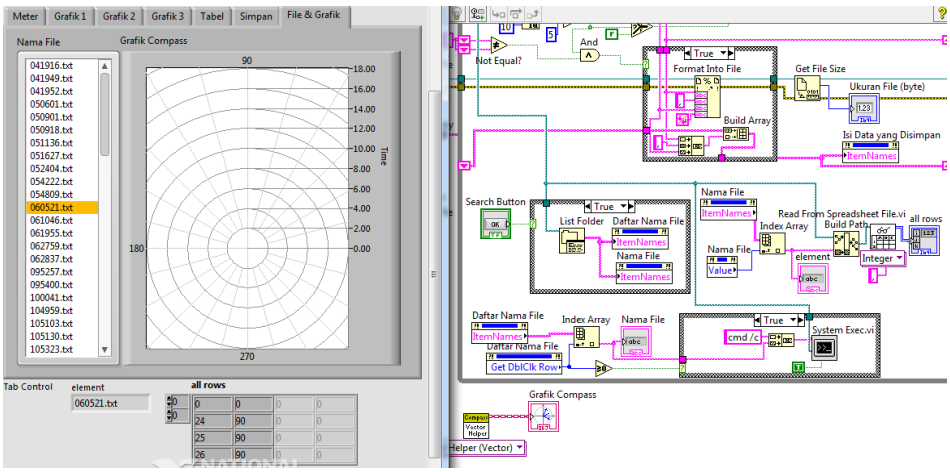


Gambar 4.63 String Indicator element menampilkan nama file yang sedang dipilih



Gambar 4.64 Read From Spreadsheet File.vi untuk membaca data dari sebuah File

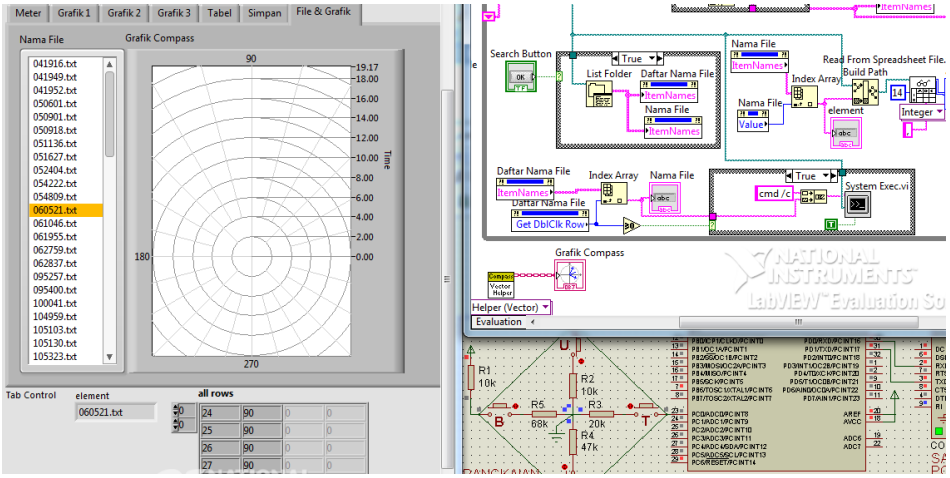
6. Jalankan simulasi Proteus dan tekan tombol Run LabVIEW. Buka Tab File & Grafik, pilih salah satu nama file di Listbox Nama File dan perhatikan isi array all rows. Seharusnya array all rows akan menampilkan isi File, yang ditampilkan dalam 2 kolom, dengan baris yang bisa lebih dari 1. Perhatikan bahwa baris pertama selalu menampilkan angka 0. Mengapa? Karena tipe data yang bisa ditampilkan oleh array all rows hanyalah integer, sedangkan data pada baris pertama semua File adalah String, yaitu bertuliskan Waktu, koma, Sensor, diikuti Enter.



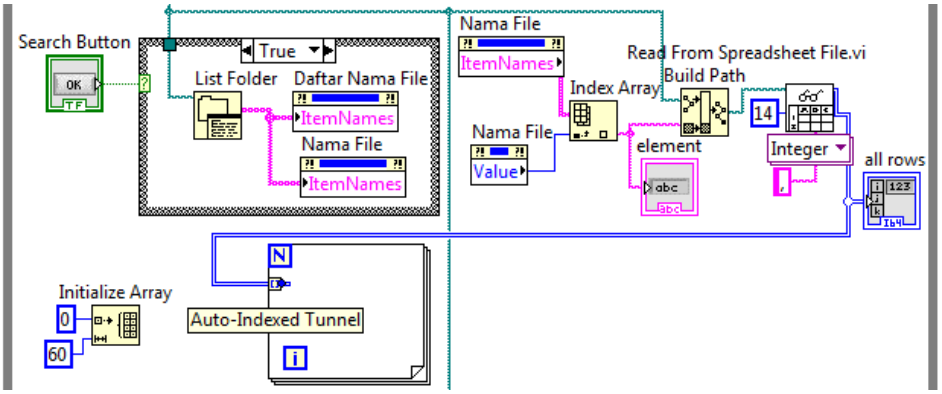
Gambar 4.65 Array all rows menampilkan data dari isi file 060521.txt

7. Untuk membuat baris pertama pada file tidak dibaca, tambahkan pada kaki input start of read offset icon Read Spreadsheet From File.vi sebuah Numeric Constant dengan angka 14. Mengapa 14? Karena jumlah karakter atau jumlah byte dari Waktu, koma, Sensor dan Enter, total adalah 14 karakter atau 14 byte (Waktu = 5, koma = 4, Sensor = 6, Enter = 2). Dengan memasukkan nilai offset sebesar 14, maka data yang akan ditampilkan di array all rows akan dimulai dari baris kedua, seperti ditunjukkan pada Gambar 4.66.
8. Langkah berikutnya adalah membuat data di all rows dipetakan ke dalam Grafik 2D Compass. Diinginkan Grafik 2D Compass menunjukkan data Sensor untuk nilai waktu dari detik 0 hingga 59. Namun demikian,

terdapat masalah, yaitu data di all rows tidak selalu dimulai dari 0, dan baris datanya selalu bervariasi, ada yang kurang dari 60 baris, ada juga yang lebih. Contohnya seperti ditunjukkan pada Gambar 4.66, di mana isi dari file 060521.txt dimulai dari angka 24, dengan baris data yang belum diketahui. Untuk memetakan data file 060521.txt ini ke dalam Grafik 2D Compass, mula-mula tambahkan sebuah Initialize Array dengan input element diisi 0 dan input dimension diisi 60. Kemudian gunakan For Loop untuk menguraikan data tiap baris di all rows dengan bantuan Auto-Indexed Tunnel seperti ditunjukkan pada Gambar 4.67.

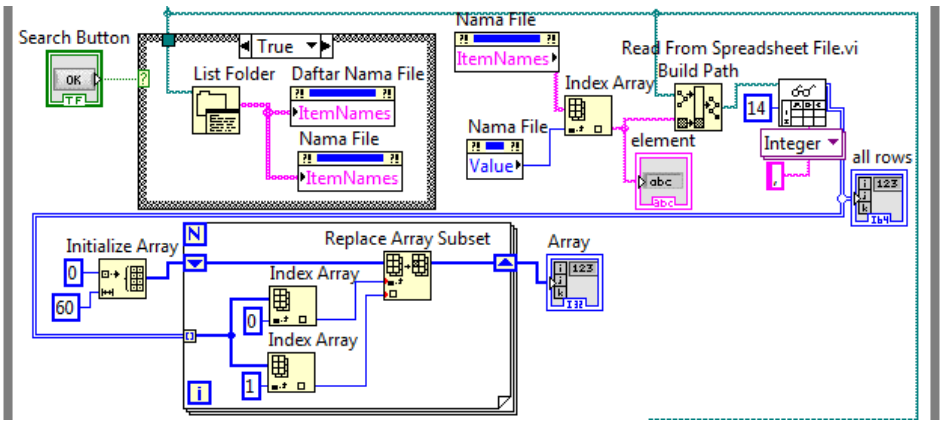


Gambar 4.66 Baris pertama file 060521.txt tidak dibaca karena offset diisi 14



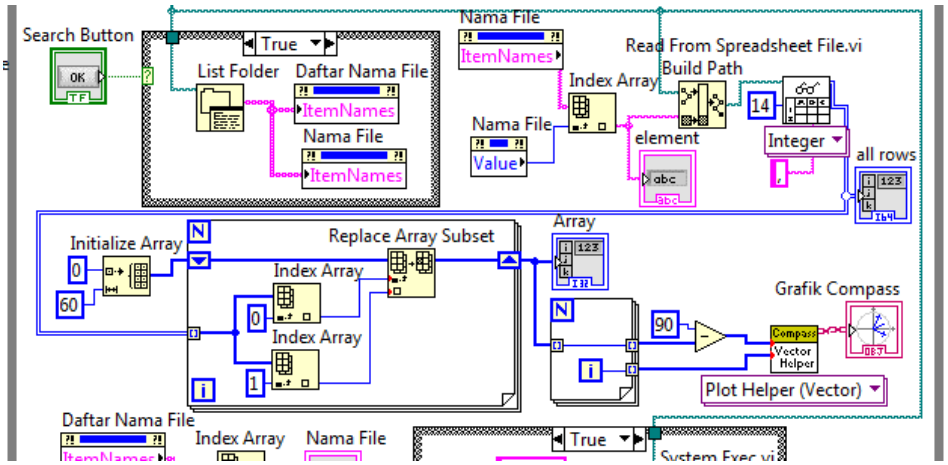
Gambar 4.67 Menggunakan Auto-Index For Loop untuk menguraikan all rows

9. Setelah data terpisah per baris, gunakan Index Array untuk memisahkan data tiap kolom. Index 0 untuk data di kolom pertama, yaitu data waktu detik, dan Index 1 untuk data di kolom kedua, yaitu data sensor. Data di kedua kolom ini digunakan untuk menggantikan data Initialize Array dengan bantuan Replace Array Subset. Tambahkan Shift Register pada For Loop untuk mengakumulasi semua penggantian data tersebut. Klik kanan pada terminal Shift Register di kanan, dan pilih Create Indicator. Maka akan muncul output Array. Output Array inilah hasil pemetaan all rows.

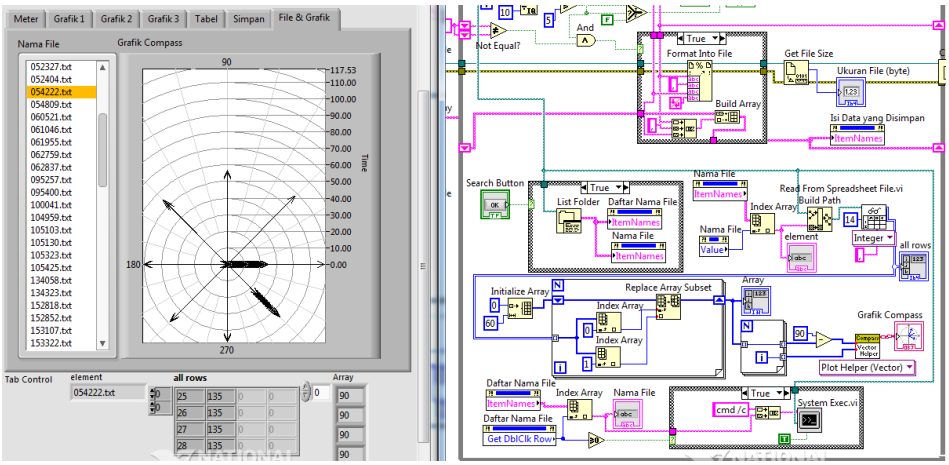


Gambar 4.68 Replace Array Subset untuk memetakan all rows ke Array 60 baris

10. Output Array memiliki jumlah baris data 60, yang isinya diperoleh dari pemetaan file hasil penyimpanan. Output Array ini akan menjadi data theta pada 2D Compass. Sedangkan data radius 2D Compass, yang harus bernilai antara 0-59, diperoleh dengan bantuan For Loop, yaitu dihasilkan dari terminal iterasi. Untuk membuat jumlah iterasi sama dengan jumlah data theta, cara paling mudah adalah dengan bantuan Auto-Indexed Tunnel, yaitu dengan melewatkan data theta ke dalam For Loop. Agar posisi 0 Compass tidak menunjuk ke samping kanan, tetapi menunjuk ke atas, maka data theta perlu dikurangkan dengan 90.
11. Jalankan simulasi Proteus, dan tekan tombol Run LabVIEW. Buka Tab File & Grafik, dan pilih salah satu file, maka 2D Compass akan menampilkan grafik data dari file tersebut.



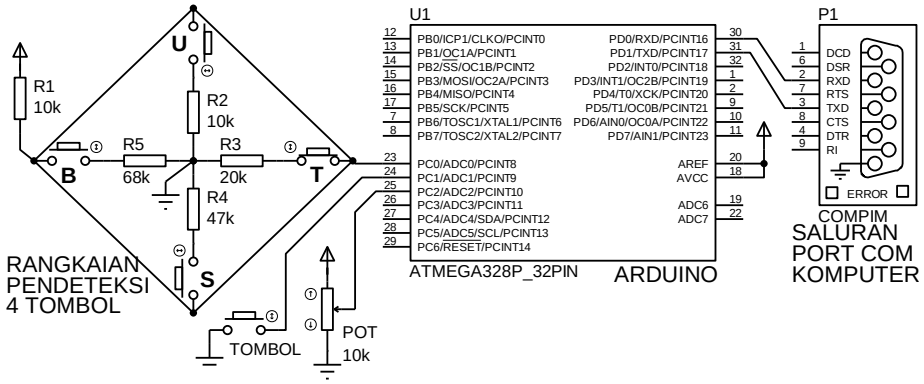
Gambar 4.69 Menghasilkan data radius dan theta dengan sumber data dari File



Gambar 4.70 2D Compass menampilkan grafik data dari File yang dipilih

4.5 Soal Latihan

1. Diinginkan tampilan Grafik dari 3 buah Sensor, yang terdiri dari Sensor Arah Angin, Tombol dan Potensio 10k ohm. Tombol terhubung dengan kaki A1 Arduino dan Potensio 10k ohm terhubung dengan kaki A2, seperti ditunjukkan pada gambar skematik Gambar 4.71 berikut ini.



Gambar 4.71 Rangkaian 3 buah Sensor: Sensor Arah Angin, Tombol dan Potensio

Tampilkan data perdetik dari ketiga buah sensor tersebut dalam Grafik Waveform Chart, Waveform Graph dan 2D Compass, dengan rentang waktu 1 menit, atau pada skala Time 0 - 59.

2. Diinginkan tampilan Tabel data perdetik dengan rentang waktu 1 menit (0 – 59) dari 3 buah Sensor seperti Soal no. 1.
3. Tambahkan penyimpanan data ke dalam file untuk ketiga buah Sensor di atas, yang nama filenya ditampilkan dalam daftar dan isi datanya dapat diperlihatkan, baik dalam bentuk teks maupun grafik.

BAB 5
INTERFACE SENSOR DAN AKTUATOR

Target Materi:

- Mengulangi pembuatan aplikasi di Bab 4 tetapi untuk lebih dari satu Sensor, yaitu menampilkan data beberapa Sensor dalam bentuk teks, grafik dan tabel, dan menyimpan datanya secara berkala.
- Memprogram LabVIEW untuk mengirimkan instruksi ke Arduino untuk mengendalikan satu atau lebih Aktuator, baik melalui teks maupun objek Control.
- Memprogram Arduino untuk menerima instruksi dari LabVIEW melalui komunikasi Serial, dan menerjemahkannya untuk mengendalikan satu atau lebih Aktuator.
- Memprogram LabVIEW untuk mengatur hubungan antara beberapa Sensor dengan beberapa Aktuator.
- Mensimulasikan aplikasi interface Sensor dan Aktuator di atas menggunakan Proteus.

Persoalan:

Program Arduino dan LabVIEW untuk menghubungkan 3 buah Sensor dan 3 buah Aktuator ke komputer, dan pengaturan hubungan antara keduanya termasuk pemantauan dan pengendaliannya.

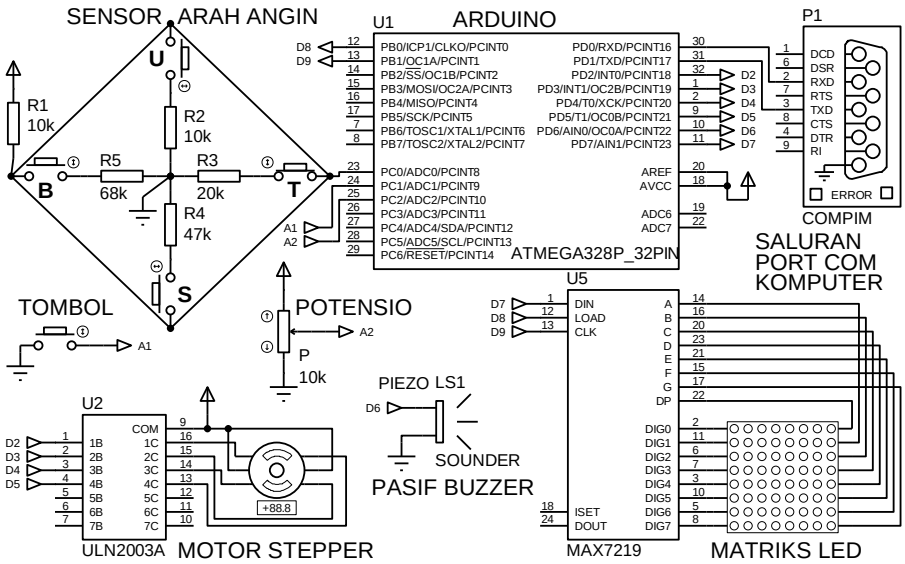
Uraian Materi:

5.1 Membaca Sensor Mengendalikan Aktuator

Membicarakan Sensor, tentunya tidak bisa lepas dari Aktuator. Apa itu Aktuator? Aktuator adalah sebuah Transduser yang bekerja berlawanan dengan Sensor. Apabila Sensor mengubah energi non-listrik menjadi listrik, maka Aktuator mengubah energi listrik menjadi non-listrik. Energi non-listrik dalam hal ini bisa berupa gerakan, cahaya, bunyi, dll. Lalu apa itu Transduser? Transduser adalah alat yang dapat mengubah sinyal dari satu bentuk energi ke bentuk energi lain.

Pembacaan Sensor dan Pengendalian Aktuator menjadi materi yang dibahas di Sub Bab ini. Secara praktis, berikut ini diberikan contoh Sensor yang akan dibaca dan Aktuator yang akan dikendalikan. Ada 3 Sensor yang digunakan, yaitu Sensor Arah Angin, Tombol dan Potensio, dan ada 3 Aktuator yang digunakan, yaitu Motor Stepper, Buzzer dan Matriks LED. Diinginkan data ketiga Sensor tersebut dapat dibaca dan ditampilkan di Serial Monitor software IDE Arduino, dan ketiga Aktuator tersebut dapat dikendalikan dari Serial Monitor. Berikut langkah-langkahnya:

1. Buat gambar skematik rangkaian berikut ini di Proteus.



Gambar 5.1 Rangkaian 3 sensor dan 3 aktuator dengan pengolah Arduino

Tabel 5.1 Sambungan kaki Sensor dan Aktuator dengan Arduino

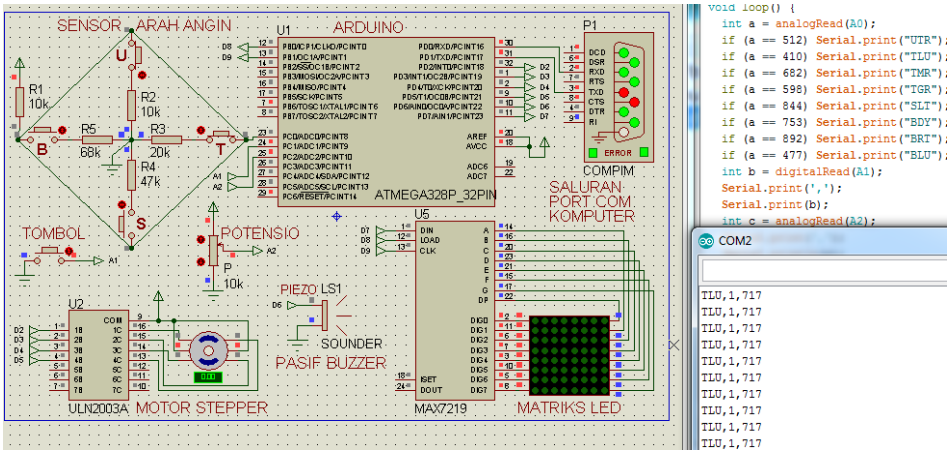
Komponen	Kaki Komponen	Kaki Arduino
Sensor Arah Angin	kaki 1, 2	A0 , Gnd
Resistor 10k ohm	kaki 1, 2	5V, A0
Tombol	kaki 1, 2	A1, Gnd
Potensio	kaki 1, 2, 3	5V, A2, Gnd
Rangkaian Motor Stepper	IN1, IN2, IN3, IN4, COM	D2, D3, D4, D5, 5V
Pasif Buzzer (Piezo)	kaki 1, 2	D6, Gnd
Rangkaian Matriks LED	VCC, GND, DIN, CS, CLK	5V, Gnd, D7, D8, D9

2. Setelah gambar skematik 5.1 di atas selesai dibuat, langkah berikutnya adalah memprogram Arduino untuk membaca data Sensor Arah Angin, Tombol dan Potensio, dan menampilkan ketiga data Sensor tersebut di Serial Monitor. Berikut ini program Arduinonya.

Program_5_1.ino

```
1.  void setup() {
2.    Serial.begin(9600);
3.    pinMode(A1,INPUT_PULLUP);
4.  }
5.  void loop() {
6.    int a = analogRead(A0);
7.    if (a == 512) Serial.print("UTR");
8.    if (a == 410) Serial.print("TLU");
9.    if (a == 682) Serial.print("TMR");
10.   if (a == 598) Serial.print("TGR");
11.   if (a == 844) Serial.print("SLT");
12.   if (a == 753) Serial.print("BDY");
13.   if (a == 892) Serial.print("BRT");
14.   if (a == 477) Serial.print("BLU");
15.   int b = digitalRead(A1);
16.   Serial.print(‘,’);
17.   Serial.print(b);
18.   int c = analogRead(A2);
19.   Serial.print(‘,’);
20.   Serial.println(c);
21.   delay(100);
22. }
```

3. Masukkan alamat file Hex hasil kompilasi Program_5_1.ino di atas pada komponen ATmega328_32pin di Proteus, dan jalankan simulasi serta buka Serial Monitor. Seharusnya di Serial Monitor menampilkan data arah angin, kondisi tombol dan potensio seperti ditunjukkan pada Gambar 5.2.
4. Setelah ketiga Sensor dapat dibaca, berikutnya adalah mengendalikan Aktuator. Dimulai dari Aktuator pertama, yaitu Motor Stepper. Diinginkan arah putaran dan Posisi Sudut Motor Stepper dapat dikendalikan dari Serial Monitor. Cara pengendalian dilakukan dengan mengetikkan nilai 1 atau 2 untuk arah putaran (cw atau ccw), dan nilai 0 – 360 untuk posisi poros pada kolom Send di Serial Monitor, yang keduanya dipisahkan dengan tanda koma, dan dikirim dengan menekan tombol Send.



Gambar 5.2 Serial Monitor menampilkan data ketiga Sensor setiap 100 milidetik

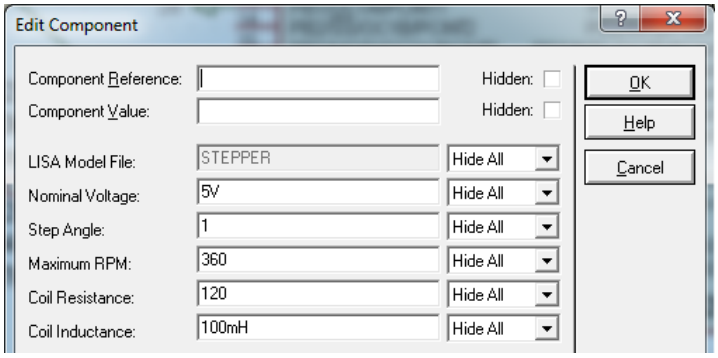
5. Berikut program pengendalian Aktuator Motor Stepper, yang dapat mengatur arah putaran Motor (nilai 1 untuk cw = searah jarum jam, dan nilai 2 untuk ccw = berlawanan jarum jam), dan juga mengatur posisi sudut poros Motor Stepper, dari 0 derajat hingga 360 derajat.

Program_5_2.ino

```
1.  #include <Stepper.h>
2.  const int stepsPerRevolution = 360;
3.  Stepper myStepper(stepsPerRevolution, 2, 3, 4, 5);
4.  void setup() {
5.    Serial.begin(9600);
```

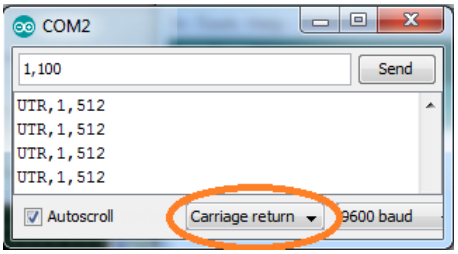
```
  pinMode(A1, INPUT_PULLUP);
  myStepper.setSpeed(2);
}
void loop() {
  int a = analogRead(A0);
  if (a == 512) Serial.print("UTR");
  if (a == 410) Serial.print("TLU");
  if (a == 682) Serial.print("TMR");
  if (a == 598) Serial.print("TGR");
  if (a == 844) Serial.print("SLT");
  if (a == 753) Serial.print("BDY");
  if (a == 892) Serial.print("BRT");
  if (a == 477) Serial.print("BLU");
  int b = digitalRead(A1);
  Serial.print(',');
  Serial.print(b);
  int c = analogRead(A2);
  Serial.print(',');
  Serial.println(c);
  delay(100);
}
void serialEvent() {
  while (Serial.available()) {
    int d = Serial.parseInt(); //arah, 1=cw, 2=ccw
    int e = Serial.parseInt(); //posisi
    if (Serial.read() == char(13)) {
      if (d == 1) myStepper.step(-e);
      if (d == 2) myStepper.step(e);
}}}
```

6. Kompilasi program di atas, dan masukkan lokasinya ke dalam Program File Component ATmega328. Namun sebelum simulasi tersebut dijalankan, Edit Component Motor Stepper di Proteus, dan isi seperti gambar berikut.



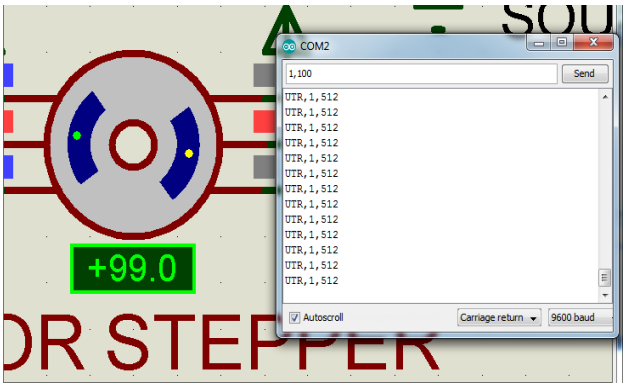
Gambar 5.3 Edit Component Nominal Voltage=5V, Step Angle=1 dan Maximum RPM=360

7. Jalankan simulasi, dan buka Serial Monitor. Isi pada kolom Send nilai 1,100 dan tekan tombol Send atau tombol Enter di Keyboard. Namun sebelum menekan tombol tersebut, atur kotak drop down di pojok kanan bawah pada pilihan Carriage Return. Mengapa memilih Carriage Return? Carriage Return dipilih karena pada Program_5_2.ino di atas, yaitu pada baris ke 31, program menanyakan apakah ada char(13) atau Carriage Return yang dibaca. Jika ada, berarti data yang dikirim sudah lengkap. Jadi kotak drop down pada Serial Monitor digunakan untuk menambahkan akhiran pada data yang dikirim ke Arduino. Apabila pembaca mengganti char(13) dengan char(10) pada baris program 31 di atas, maka pembaca dapat memilih New Line pada kotak drop down di Serial Monitor.



Gambar 5.4 Pilih Carriage Return sebagai akhiran untuk data yang dikirim

8. Seperti Gambar 5.4 di atas, ketika angka 1,100 dikirimkan, maka Motor Stepper akan berputar cw dan berhenti berputar saat poros mencapai sudut 99 derajat, seperti ditunjukkan pada gambar berikut.



Gambar 5.5 Motor Stepper berputar cw 99 derajat saat data 1,100 dikirimkan

9. Setelah Aktuator yang menghasilkan gerakan dapat dikendalikan, langkah berikutnya adalah mengendalikan Aktuator yang menghasilkan bunyi. Aktuator bunyi salah satunya adalah Buzzer. Ada 2 jenis Buzzer, yaitu Buzzer tipe Aktif dan tipe Pasif. Buzzer tipe Aktif memiliki rangkaian pengondisi sinyal di dalamnya, sedangkan Buzzer tipe Pasif tidak. Karena tidak memiliki rangkaian pengondisi sinyal, maka Buzzer tipe Pasif memerlukan input sinyal khusus, yaitu sinyal yang berbentuk pulsa agar bisa berbunyi. Sedangkan Buzzer tipe Aktif, input yang diberikan tidak perlu berbentuk pulsa, cukup tegangan dc biasa. Namun kelemahan dari Buzzer tipe Aktif, bunyi yang dihasilkan tidak dapat diatur nadanya. Sebaliknya untuk Buzzer tipe Pasif, nada bunyinya dapat diatur sesuai dengan frekuensi pulsa inputnya, semakin tinggi frekuensinya, semakin tinggi nadanya. Buzzer tipe Pasif juga disebut Piezoelektrik. Di Proteus, Buzzer tipe Pasif diberi nama Piezo Sounder. Berikut program untuk membuat Piezo Sounder menghasilkan bunyi nada dari do rendah hingga do tinggi untuk nilai 1 - 8 yang dikirimkan dari Serial Monitor.

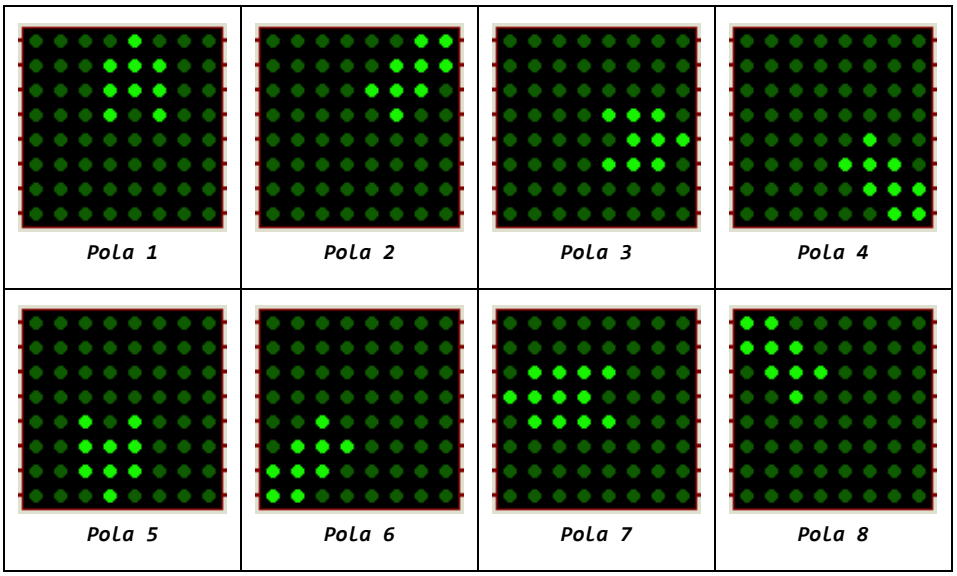
Program_5_3.ino

```
#include <Stepper.h>
#define do1  1047
#define re  1175
#define mi  1319
#define fa  1397
#define sol  1568
#define la  1760
#define si  1976
#define do2  2093
int nada[] = {do1, re, mi, fa, sol, la, si, do2};
const int stepsPerRevolution = 360;
Stepper myStepper(stepsPerRevolution, 2, 3, 4, 5);
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  myStepper.setSpeed(2);
}
void loop() {
  int a = analogRead(A0);
  if (a == 512) Serial.print("UTR");
  if (a == 410) Serial.print("TLU");
  if (a == 682) Serial.print("TMR");
  if (a == 598) Serial.print("TGR");
```

```
  if (a == 844) Serial.print("SLT");
  if (a == 753) Serial.print("BDY");
  if (a == 892) Serial.print("BRT");
  if (a == 477) Serial.print("BLU");
  int b = digitalRead(A1);
  Serial.print(',');
  Serial.print(b);
  int c = analogRead(A2);
  Serial.print(',');
  Serial.println(c);
  delay(100);
}
void serialEvent() {
  while (Serial.available()) {
    int d = Serial.parseInt(); //arah, 1=cw, 2=ccw
    int e = Serial.parseInt(); //posisi
    int f = Serial.parseInt(); //nada
    if (Serial.read() == char(13)) {
      tone(6, nada[f-1], e * 100);
      if (d == 1) myStepper.step(-e);
      if (d == 2) myStepper.step(e);
      noTone(6);
}}}
```

10. Perhatikan pada program di atas, instruksi untuk membunyikan Buzzer adalah tone, yang memiliki 3 parameter, yaitu kaki IO, frekuensi dan durasi bunyi. Tampak pada progam, durasi bunyi diisi e*100, ini dimaksudkan agar Buzzer berbunyi selama Motor Stepper berputar. Kompilasi program di atas, dan masukkan lokasi file Hex yang dihasilkan ke dalam Program File Component ATmega328.
11. Jalankan simulasi Proteus dan buka Serial Monitor. Pada kolom Send, masukkan angka 1 atau 2 untuk arah putaran, 0 – 360 untuk Posisi Sudut Motor Stepper dan 1 – 8 untuk nada Buzzer. Beri tanda koma untuk memisahkan ketiga data, kemudian tekan Enter atau tombol Send. Sebagai contoh 1,100,3.
12. Setelah Aktuator yang menghasilkan gerakan dan bunyi berhasil dikendalikan, berikutnya adalah Aktuator yang menghasilkan cahaya, yaitu Matriks LED. Pengendalian Matriks LED di sini cukup mudah, karena adanya bantuan IC Shift Register MAX 7219. Sama seperti Aktuator yang lainnya, diinginkan Matriks LED di sini dapat dibuat menampilkan 8 pola

yang berbeda, sesuai angka yang dimasukkan dalam Serial Monitor. Buat kedelapan pola tersebut menunjukkan arah angin seperti gambar berikut.



Gambar 5.6 Kedelapan pola Matriks LED di Proteus

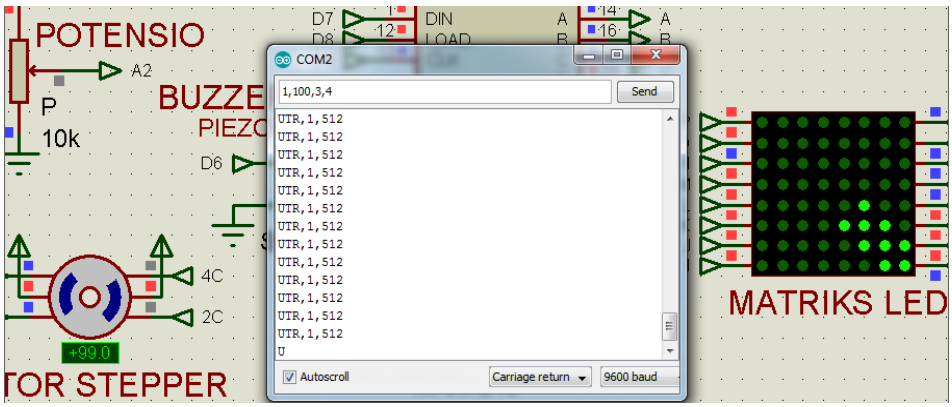
13. Menggunakan bantuan library LedControl, berikut program untuk mengendalikan Matriks LED agar menampilkan pola sesuai Gambar 5.6 di atas, mengikuti angka 1 – 8 yang dikirimkan dari Serial Monitor. Sekedar informasi, pembaca dapat mendownload library ledControl di Internet, atau mengambilnya di CD pendukung buku ini. Jadi ada 4 data yang harus dimasukkan di Serial Monitor, yaitu arah putaran Motor Stepper, Posisi Sudut Motor Stepper, nada Buzzer Pasif/Piezo dan pola Matriks LED.

Program_5_4.ino

```
1.  #include <Stepper.h>
2.  #include "LedControl.h"
3.  #define do1  1047
4.  #define re  1175
5.  #define mi  1319
6.  #define fa  1397
7.  #define sol  1568
8.  #define la  1760
9.  #define si  1976
10. #define do2  2093
11. int nada[] = {do1, re, mi, fa, sol, la, si, do2};
```

```
const int stepsPerRevolution = 360;
byte pola[8][8] = {
{B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},//1 UTR
{B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},//2 TLU
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},//3 TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},//4 TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},//5 SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},//6 BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},//7 BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} //8 BLU
};
Stepper myStepper(stepsPerRevolution, 2, 3, 4, 5);
LedControl lc = LedControl(7, 9, 8, 1);
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  myStepper.setSpeed(2);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);}
void loop() {
  int a = analogRead(A0);
  if (a == 512) Serial.print("UTR");
  if (a == 410) Serial.print("TLU");
  if (a == 682) Serial.print("TMR");
  if (a == 598) Serial.print("TGR");
  if (a == 844) Serial.print("SLT");
  if (a == 753) Serial.print("BDY");
  if (a == 892) Serial.print("BRT");
  if (a == 477) Serial.print("BLU");
  int b = digitalRead(A1);
  Serial.print(',');
  Serial.print(b);
  int c = analogRead(A2);
  Serial.print(',');
  Serial.println(c);
  delay(100);}
void serialEvent() {
  while (Serial.available()) {
    int d = Serial.parseInt(); //arah, 1=cw, 2=ccw
    int e = Serial.parseInt(); //posisi Motor
    int f = Serial.parseInt(); //nada Buzzer
    int g = Serial.parseInt(); //pola Matriks
    if (Serial.read() == char(13)) {
      for (int i = 0; i < 8; i++) {
        lc.setRow(0, i, pola[g - 1][i]);}
      tone(6, nada[f - 1], e * 100);
      if (d == 1) myStepper.step(-e);
      if (d == 2) myStepper.step(e);
      noTone(6);
}}}
```

14. Kompilasi program di atas, masukkan lokasi file Hex yang dihasilkan ke dalam Program File Component ATmega328. Jalankan simulasi Proteus dan buka Serial Monitor. Pada kolom Send, masukkan angka 1 atau 2 untuk arah putaran, 0 – 360 untuk Posisi Sudut Motor Stepper, angka 1 – 8 untuk nada Buzzer, dan angka 1 – 8 untuk pola Matriks LED. Beri tanda koma untuk memisahkan keempat data tersebut, kemudian tekan Enter atau tombol Send. Berikut ini tampilan Proteus untuk angka 1,100,3,4, yang membuat posisi sudut Motor Stepper 99^0, Buzzer berbunyi pada nada ketiga (mi) dan Matriks LED menampilkan pola keempat.



Gambar 5.7 Motor Stepper cw 99^0, Buzzer berbunyi nada ketiga, Matriks LED pola 4

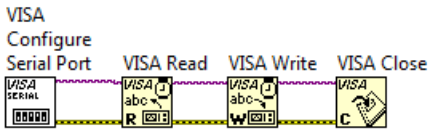
5.2 *Menghubungkan (Interface) Sensor dan Aktuator melalui LabVIEW*

Apabila pada Sub Bab 5.1 di atas, Serial Monitor digunakan untuk menampilkan data Sensor yang dibaca oleh Arduino dan mengirimkan instruksi ke Arduino untuk mengendalikan Aktuator, maka pada Sub Bab 5.2 ini, baik Sensor maupun Aktuator, akan dihubungkan atau di-interface melalui LabVIEW. Keuntungan interface dengan LabVIEW adalah, fungsi-fungsi pengolahan data Sensor dan pengendalian Aktuator, dan hubungan antara keduanya dapat diatur dengan mudah. Ada 2 bagian pada Sub Bab 5.2 Interface Sensor dan Aktuator ini, yaitu Menggantikan Serial Monitor, dan Menambahkan Objek Interaktif.

5.2.1 Menggantikan Serial Monitor

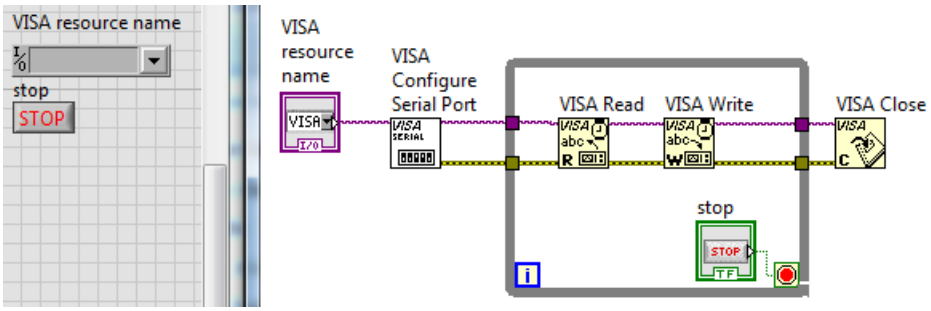
Apabila Serial Monitor dapat digunakan untuk menampilkan data Sensor dan mengirimkan instruksi ke Arduino, maka LabVIEW juga dapat melakukan fungsi Serial Monitor tersebut. Berikut ini langkah-langkahnya:

1. Di jendela Block Diagram, tempatkan icon VISA Configure Serial Port, VISA Read, VISA Write dan VISA Close, dan hubungkan seperti gambar berikut:



Gambar 5.8 Icon VISA Configure, VISA Read, VISA Write dan VISA Close

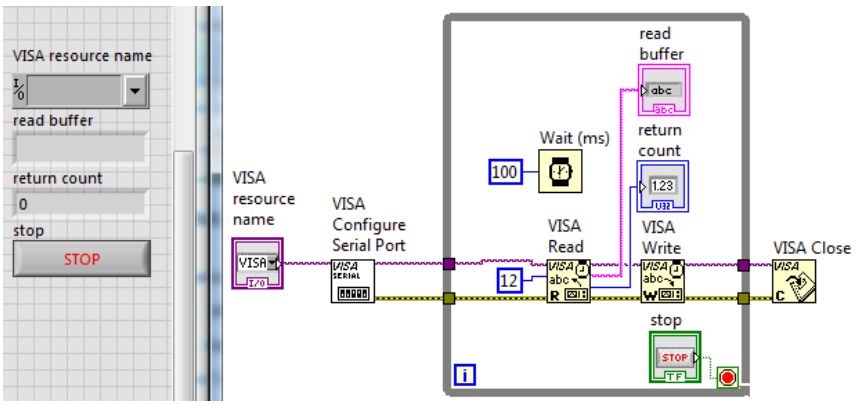
2. Tambahkan kotak While Loop pada VISA Read dan VISA Write. Klik kanan pada terminal Loop Condition While Loop dan pilih Create Control. Kemudian klik kanan kaki VISA Resource Name di VISA Configure, pilih Create Control, maka akan muncul kotak drop down VISA Resource Name dan tombol Stop di jendela Front Panel seperti gambar berikut:



Gambar 5.9 Kotak dropdown VISA Resource Name dan tombol Stop di Front Panel

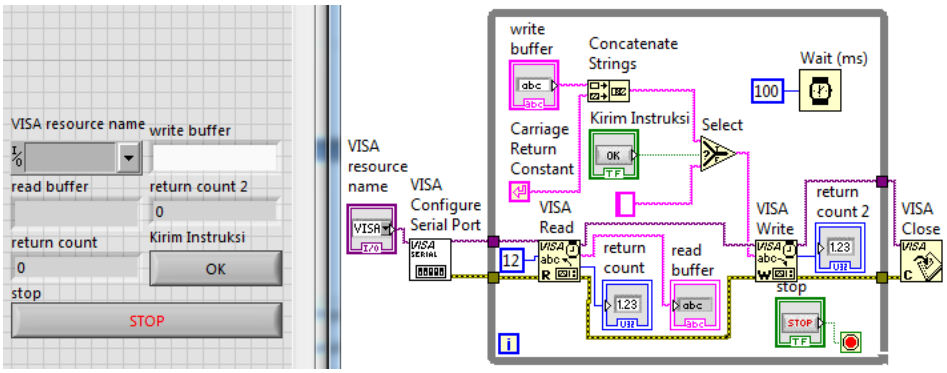
3. Klik kanan kaki byte count VISA Read, pilih Create Constant, dan isi nilainya dengan 12. Mengapa 12? Karena jumlah karakter yang dikirimkan dari Arduino total maksimum sebesar 12 byte, yang terdiri dari data arah angin 3 byte, tanda koma 1 byte, kondisi tombol 1 byte, tanda koma kedua 1 byte, nilai potensio dari 0 – 1023 (maksimum 4 karakter) = 4 byte, karakter akhiran CR dan NL 2 byte, sehingga 3+1+1+1+4+2 = 12 byte.

4. Untuk mengetahui karakter yang dibaca VISA Read, klik kanan kaki read buffer VISA Read, dan pilih Create Indicator. Untuk mengetahui jumlah byte yang dibaca, klik kanan kaki return count VISA Read, dan pilih Create Indicator. Agar CPU komputer tidak bekerja terlalu keras, tambahkan icon Wait (ms), dan isi nilainya dengan 100.



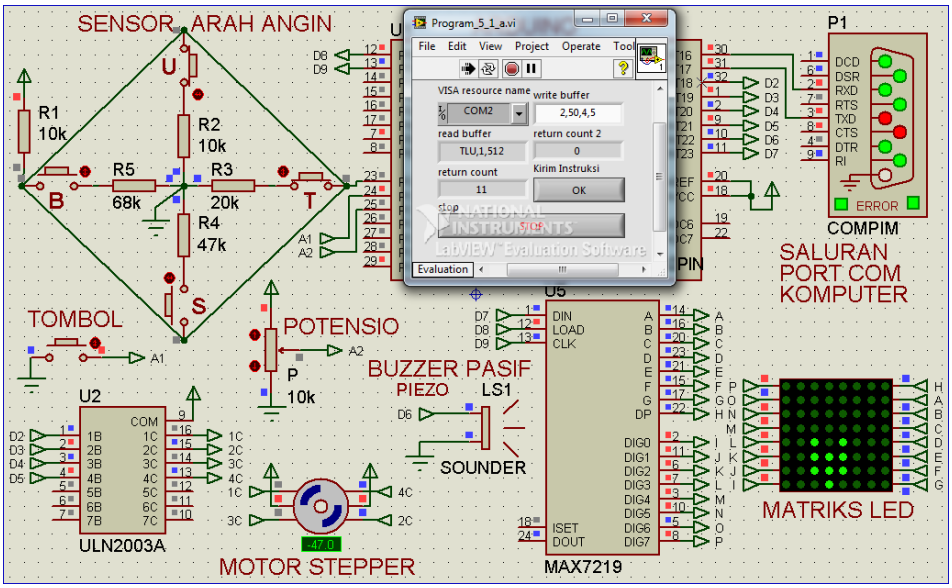
Gambar 5.10 VISA Read dengan byte count 12, read buffer dan return count

5. Meniru kolom Send pada Serial Monitor, klik kanan kaki write buffer VISA Write, dan pilih Create Control. Agar muncul karakter akhiran, sisipkan sebuah Concatenate String di antara write buffer dan VISA Write, dan tarik ke bawah icon Concatenate String tersebut sehingga muncul satu kaki input. Klik kanan kaki tersebut, pilih String Pallete, pilih Carriage Return Constant.
6. Berikutnya, meniru tombol Send di Serial Monitor, tambahkan sebuah OK Button (Boolean, Modern), dan beri nama Kirim Instruksi. Tambahkan icon Select. Hubungkan tombol Kirim tersebut ke kaki s icon Select dan output Concatenate String ke kaki t Select. Kemudian klik kanan kaki f Select, pilih Create Constant, maka akan muncul String kosong. Dengan hubungan tersebut, jika tombol Kirim tidak ditekan, tidak ada karakter yang dikirimkan, sedangkan ketika tombol Kirim ditekan, maka isi write buffer dikirim bersama akhiran CR. Terakhir, untuk mengetahui jumlah byte yang dikirimkan, klik kanan return count VISA Write, dan pilih Create Indicator. Kemudian simpan program ini dengan nama Program_5_1.vi.



Gambar 5.11 VISA Write dengan write buffer dan akhiran CR serta tombol Kirim

7. Dengan menggunakan rangkaian Gambar 5.1 di Proteus, dan Program_5_4.ino pada komponen Arduino (ATmega328), jalankan simulasi Proteus dan Program_5_1.vi LabVIEW Gambar 5.11 tersebut.
8. Gambar 5.12 berikut ini menunjukkan isi read buffer (TLU, 1, 512) untuk data yang dikirimkan dari Arduino ke LabVIEW, dan tampilan Proteus untuk data 2,50,4,5 CR yang dikirimkan dari LabVIEW ke Arduino.

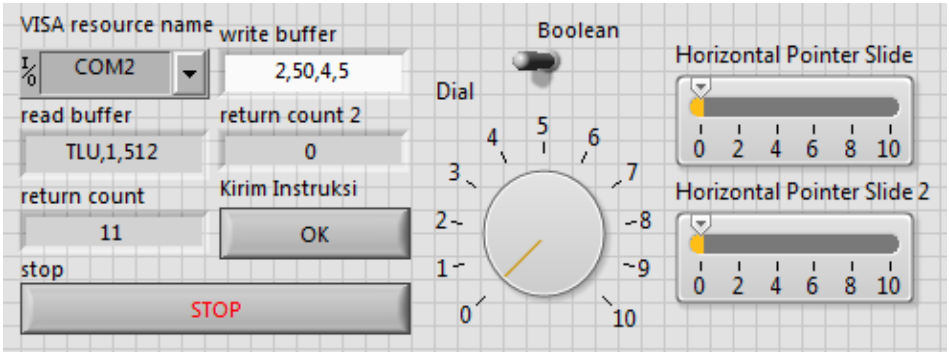


Gambar 5.12 Isi read buffer di LabVIEW (TLU,1,512) yang diterima dari Arduino dan tampilan Proteus untuk data 2,50,4,5 yang dikirimkan dari LabVIEW

5.2.2 *Menambahkan Objek Interaktif*

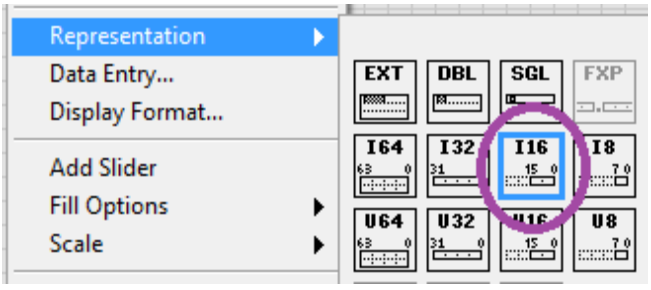
Seperti telah disebutkan di awal, bahwa keuntungan dari LabVIEW adalah tersedianya fungsi-fungsi untuk pengolahan data input dan output, dan pengaturan hubungan antara keduanya yang mudah dan interaktif. Untuk itu, agar pengaturan hubungan input output di program sebelumnya menjadi lebih interaktif, tambahkan objek-objek yang diambil dari Pallet Control seperti berikut:

1. Di jendela Front Panel, tempatkan 4 objek, berturut-turut: Horizontal Toggle Switch (Boolean, Modern), Dial (Numeric, Silver) dan 2 buah Horizontal Pointer Slide (Numeric, Silver).



Gambar 5.13 Tambahkan 4 objek: Boolean, Dial, 2 buah Horizontal Pointer Slide

2. Buat agar objek Dial memiliki skala 0-360, dan kedua Horizontal Pointer Slide memiliki skala 1-8. Agar nilainya bulat, tidak ada pecahan, maka ubah tipe data ketiga objek tersebut dari Double menjadi Integer 16 bit, dengan cara klik kanan objek, pilih Representation, pilih I16.

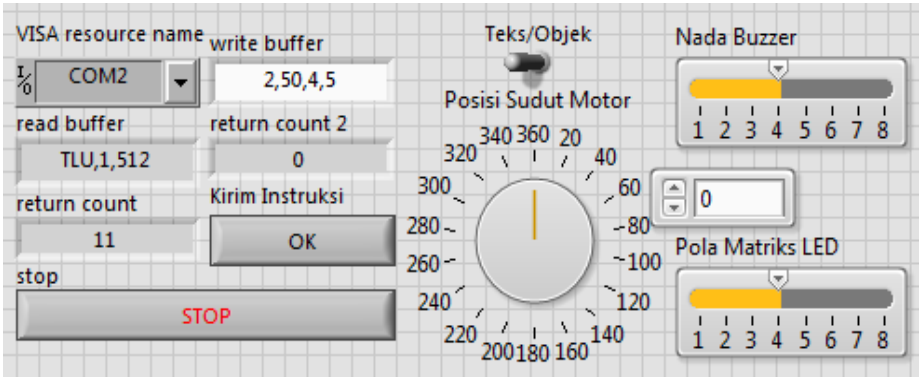


Gambar 5.14 Mengubah tipe data dengan klik kanan, pilih Representation, pilih I16

3. Berikutnya ubah nama Boolean menjadi Teks/Objek, Dial menjadi Posisi Sudut Motor, Horizontal Pointer Slide pertama menjadi Nada Buzzer dan Horizontal Pointer Slide kedua menjadi Pola Matriks LED. Berhubung nilai yang ditunjuk oleh jarum pada objek Dial tidak terlihat jelas, maka tambahkan Digital Display, dengan cara meng-klik kanan objek Dial, pilih Visible Items, kemudian pilih Digital Display, seperti Gambar 5.15.

Catatan:

1. Toggle Switch Teks/Objek digunakan untuk memilih, apakah data yang dikirimkan ke Arduino, diambil dari teks atau objek. Apabila memilih teks, maka data harus diketik pada write buffer dan kemudian menekan tombol Kirim Instruksi untuk mengirimkan data. Apabila memilih Objek, maka data langsung terkirim setiap kali nilai salah satu dari ketiga objek diubah.
2. Ada 2 kontrol sudut poros motor, yaitu kontrol sudut relatif dan kontrol sudut absolut. Kontrol sudut relatif membuat poros Motor Stepper akan berputar sebesar nilai sudut yang diberikan. Contoh, pengiriman nilai 100 sebanyak 2 kali, akan membuat poros Motor Stepper berputar sejauh 200 derajat. Berbeda dengan kontrol sudut relatif, kontrol sudut absolut membuat poros Motor Stepper berada pada posisi sesuai nilai sudut yang diberikan. Contoh, pengiriman nilai 100 sebanyak 2 kali, akan membuat poros Motor Stepper tetap berada pada sudut 100 derajat.



Gambar 5.15 Objek grafis interaktif: Teks/Objek (Toggle), Posisi Sudut Motor (Dial) dengan Digital Display, Nada Buzzer (Slider) dan Pola Matriks LED (Slider)

4. Karena menggunakan objek Dial, maka kontrol sudut poros motor yang paling mudah digunakan adalah kontrol sudut absolut. Posisi sudut poros Motor Stepper harus sama dengan posisi jarum pada Dial. Dengan kontrol sudut absolut ini, tidak lagi diperlukan data arah. Sehingga data yang dikirimkan hanya 3, yaitu: posisi sudut, nada buzzer dan pola matriks LED.
5. Program di Arduino harus diubah agar hanya menerima 3 data. Di samping itu, agar kontrol sudut absolut dapat diterapkan, posisi sudut yang sebelumnya harus disimpan. Berikut ini program Arduino untuk menerima 3 data dan menyimpan data posisi sudut sebelumnya:

Program_5_5.ino

```
#include <Stepper.h>
#include "LedControl.h"
#define do1  1047
#define re  1175
#define mi  1319
#define fa  1397
#define sol  1568
#define la  1760
#define si  1976
#define do2  2093
int nada[] = {do1, re, mi, fa, sol, la, si, do2};
int a1 = 0;
int b1 = 0;
int c1 = 0;
int e1 = 0;
const int stepsPerRevolution = 360;
byte pola[8][8] = {
{B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},//1 UTR
{B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},//2 TLU
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},//3 TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},//4 TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},//5 SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},//6 BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},//7 BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} //8 BLU
};
Stepper myStepper(stepsPerRevolution, 2, 3, 4, 5);
LedControl lc = LedControl(7, 9, 8, 1);
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  myStepper.setSpeed(2);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
```

```
}
void loop() {
  int a = analogRead(A0);
  int b = digitalRead(A1);
  int c = analogRead(A2);
  if (a != a1 || b != b1 || c != c1) {
    if (a == 512) Serial.print("UTR");
    if (a == 410) Serial.print("TLU");
    if (a == 682) Serial.print("TMR");
    if (a == 598) Serial.print("TGR");
    if (a == 844) Serial.print("SLT");
    if (a == 753) Serial.print("BDY");
    if (a == 892) Serial.print("BRT");
    if (a == 477) Serial.print("BLU");
    Serial.print(',');
    Serial.print(b);
    Serial.print(',');
    Serial.println(c);
    a1 = a;
    b1 = b;
    c1 = c;
  }
}
void serialEvent() {
  while (Serial.available()) {
    int e = Serial.parseInt() + 1; //posisi Motor
    int f = Serial.parseInt(); //nada Buzzer
    int g = Serial.parseInt(); //pola Matriks
    if (Serial.read() == char(13)) {
      for (int i = 0; i < 8; i++) {
        lc.setRow(0, i, pola[g - 1][i]);
      }
      tone(6, nada[f - 1], (e1 – e) * 100);
      myStepper.step(e1 - e);//menuju ke posisi yang baru
      e1 = e;// menyimpan data posisi sebelumnya
      noTone(6);
}}}
```

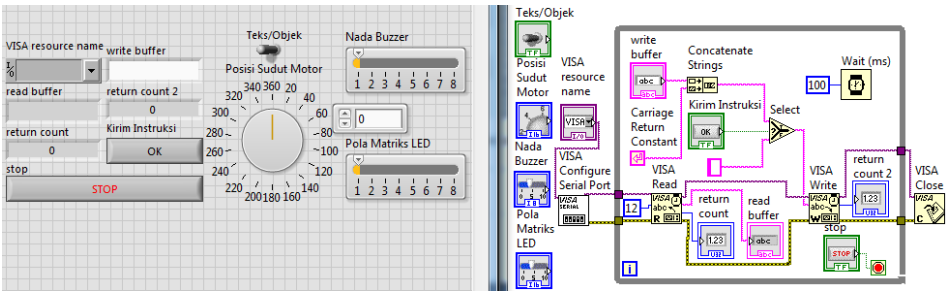
6. Pada Program_5_5.ino di atas, pada baris ke 61, tampak 2 buah garis miring di awal baris, yang menunjukkan bahwa instruksi setelah tanda tersebut diabaikan. Instruksi untuk menerima data arah dihilangkan, dengan demikian hanya 3 data yang akan diterima. Berikutnya pada baris ke 73, sebuah variabel e1 digunakan untuk menyimpan data posisi sebelumnya. Data posisi sebelumnya (e) dibandingkan dengan data posisi yang dituju (e). Selisih keduanya digunakan untuk membuat putaran

Motor Stepper. Hasil selisih negatif akan membuat Motor Stepper diputar CW sejauh nilai selisih, sedangkan hasil selisih positif akan membuat Motor Stepper diputar CCW.

7. Di samping modifikasi di atas, ada modifikasi lain, yang terlihat pada baris ke 41. Dengan instruksi pada baris ke 41 tersebut, maka Arduino hanya mengirimkan data apabila ada perubahan nilai salah satu dari ketiga Sensor. Apabila tidak ada perubahan nilai, maka Arduino tidak mengirimkan data. Hal ini selain menghemat konsumsi daya, juga membuat komunikasi Arduino dengan LabVIEW menjadi efisien, karena nantinya di LabVIEW juga akan dibuat hanya akan mengirimkan data apabila ada perubahan dari nilai ketiga objeknya.

Catatan: Komponen COMPIM di Proteus dapat digunakan untuk mengamati pengiriman data dari Arduino ke LabVIEW dan penerimaan data dari LabVIEW ke Arduino. Apabila bulatan pada kaki TX berwarna merah, berarti ada data yang dikirim, sedangkan bila bulatan yang berwarna merah pada kaki RX, berarti ada data yang diterima. Dengan seringnya keduanya berwarna merah, berarti semakin sering data dikirim dan diterima.

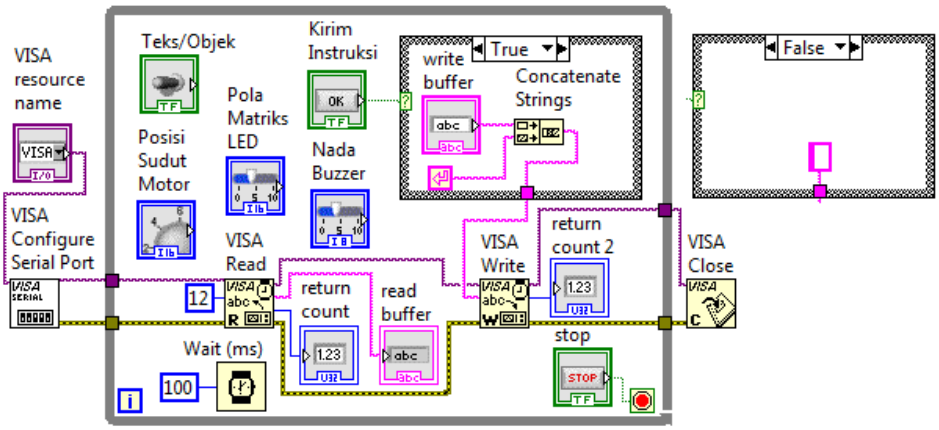
8. Setelah program Arduino dimodifikasi hanya menerima 3 data, berikut ini Program_5_2.vi LabVIEW untuk mengirimkan data 3 objek.



Gambar 5.16 Jendela Front Panel dan Block Diagram untuk Gambar 5.15

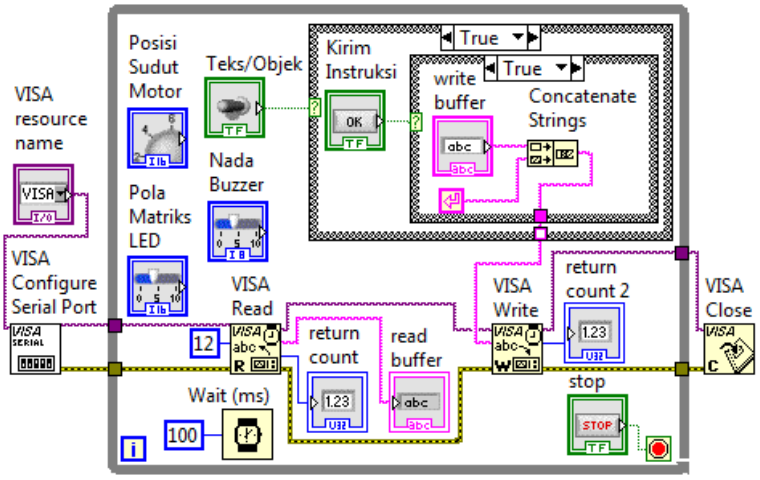
9. Masukkan keempat objek ke dalam While Loop. Agar program terlihat lebih sederhana, ganti icon Select dengan Case Structure. Tempatkan

write buffer, Carriage Return dan Concatenate Strings di kotak Case True, sedangkan Empty String Constant di kotak Case False.



Gambar 5.17 Mengganti Icon Select dengan Case Structure

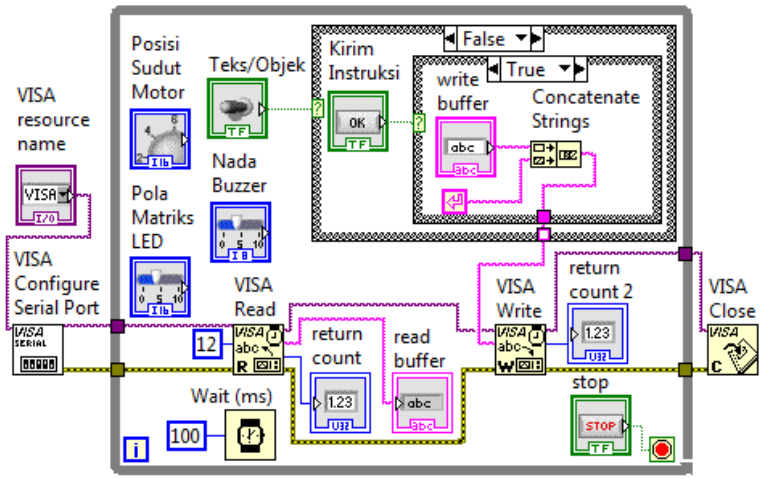
10. Berikutnya, buat Case Structure baru untuk Toggle Switch Teks/Objek, dengan menempatkan tombol Kirim Instruksi beserta Case Structurenya di dalamnya, seperti gambar berikut ini.



Gambar 5.18 Mengganti Icon Select dengan Case Structure

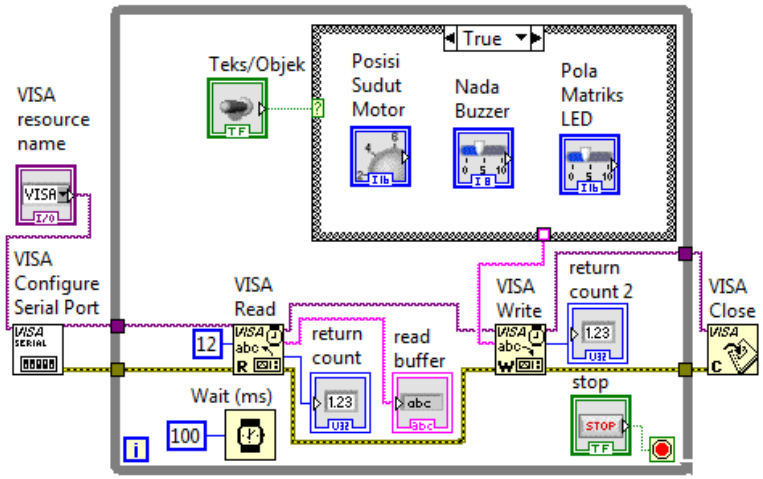
11. Karena tombol Kirim Instruksi beserta Case Structurenya hanya dapat dijalankan bila Toggle Switch Teks/Objek bernilai False, maka ubah kotak

Case True menjadi False, dengan cara meng-klik kanan pada Case Structure Teks/Objek, dan pilih Make This Case False.



Gambar 5.19 Make This Case False untuk membuat Case data teks menjadi False

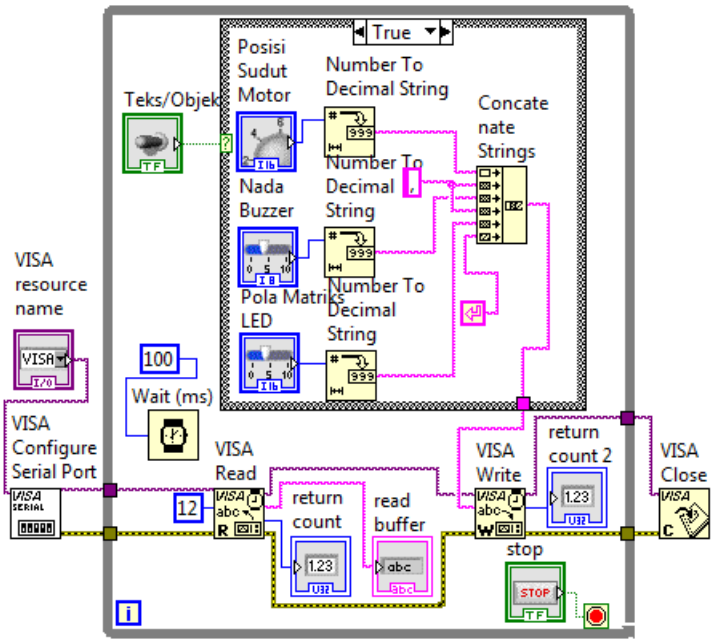
12. Kemudian pindahkan ketiga objek (Posisi Sudut Motor, Nada Buzzer dan Pola Matriks LED) ke dalam kotak Case True Teks/Objek tersebut.



Gambar 5.20 Pindahkan ketiga objek ke dalam kotak Case True Teks/Objek

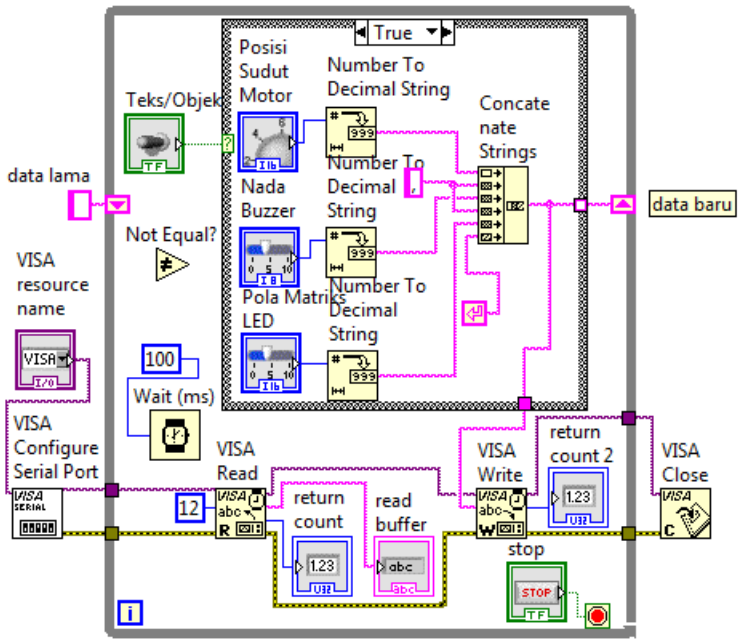
13. Karena input write buffer VISA Write hanya menerima tipe data String, maka tambahkan icon Number to Decimal String pada ketiga objek.

Kemudian satukan ketiga data objek tersebut dengan Concatenate Strings, dengan tanda koma sebagai pemisah data dan Carriage Return sebagai karakter akhir.



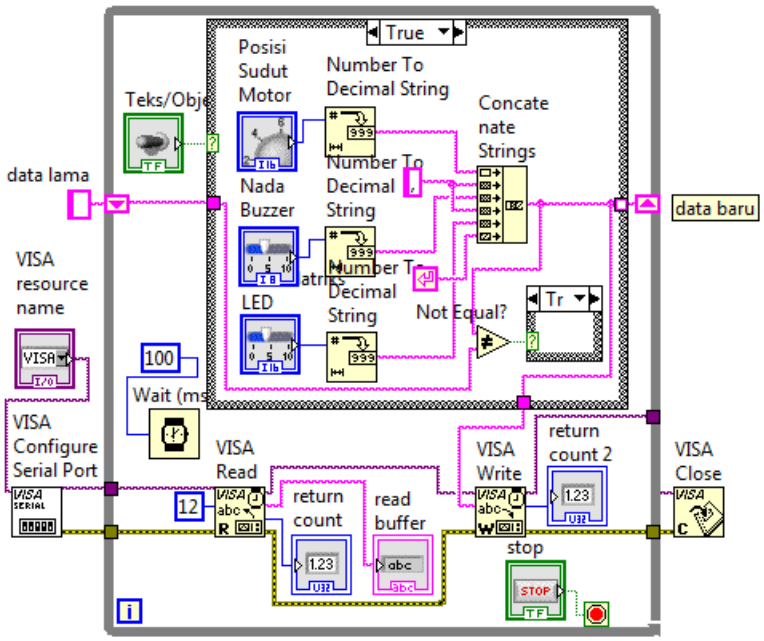
Gambar 5.21 Tambahkan Number To Decimal String dan Concatenate Strings

14. Seperti telah disebutkan di awal, saat pilihan Toggle Switch pada Objek, pengiriman data hanya dilakukan ketika salah satu objek diubah nilainya. Untuk mengetahui nilai objek diubah adalah dengan membandingkan data yang sekarang dengan data yang sebelumnya. Apabila tidak sama, berarti ada salah satu objek yang diubah nilainya. Untuk membandingkan data, gunakan icon Not Equal. Untuk mengetahui data sebelumnya, gunakan Shift Register, yang akan menyimpan data. Hubungkan terminal kanan Shift Register dengan output dari Concatenate Strings kotak Case True, dan beri nilai awal terminal kiri Shift Register, yaitu dengan cara meng-klik kanan terminal kiri, dan pilih Create Constant.

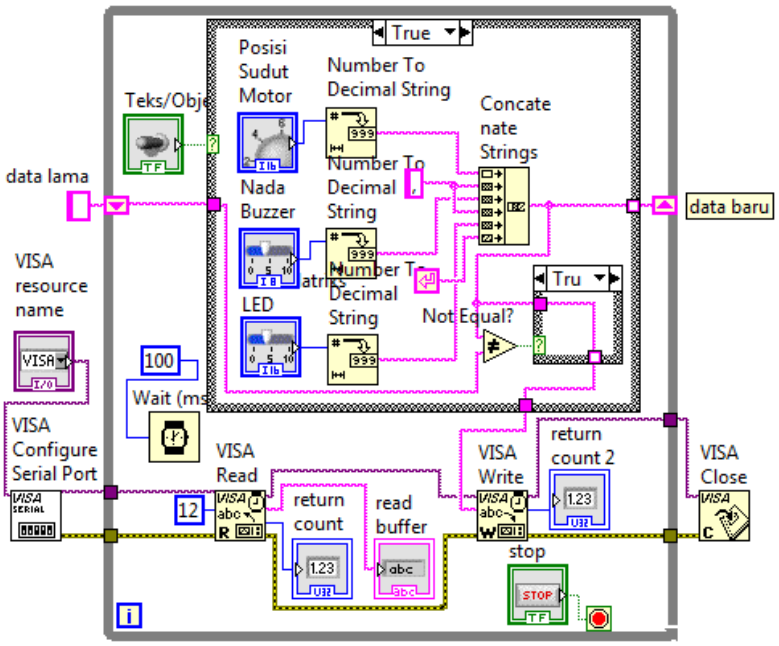


Gambar 5.22 Menambahkan Icon Not Equal dan Shift Register

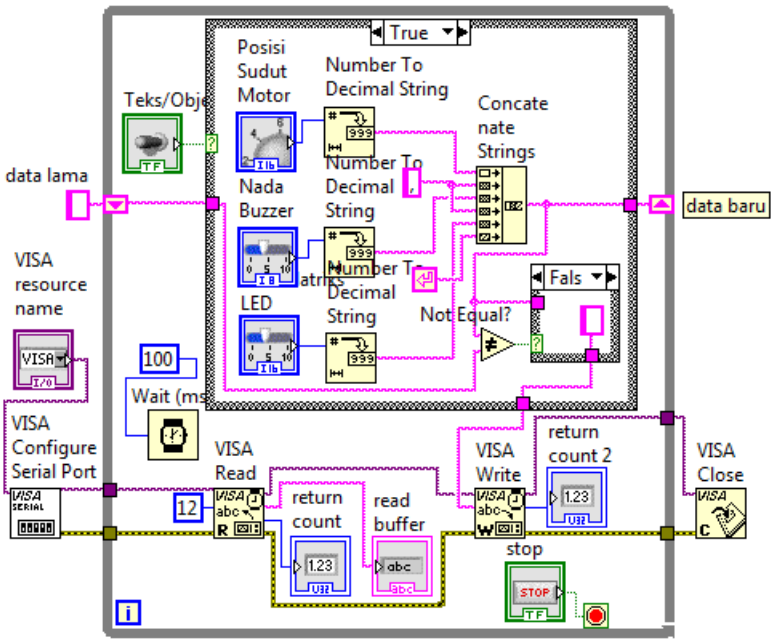
15. Berikutnya, tempatkan icon Not Equal di dalam kotak Case True. Hubungkan input kaki pertama dengan terminal kiri Shift Register (data lama), dan input kaki kedua dengan output Concatenate String (data baru). Ketika data baru tidak sama dengan data lama, maka kirimkan data yang baru tersebut. Untuk itu tambahkan sebuah Case Structure.
16. Putus output Concatenate Strings ke terminal kanan Shift Register. Buat garis data dari output Concatenate strings tersebut melalui Case Structure yang baru, yaitu di kotak Case True, yang diteruskan ke terminal kanan Shift Register dan ke write buffer VISA Read. Kemudian di Case False, buat agar garis data output Concatenate Strings tidak diteruskan ke terminal kanan Shift Register dan write buffer VISA Read. Sebagai gantinya, ambil garis data dari terminal kiri Shift Register untuk diteruskan ke terminal kanan Shift Register. Sedangkan untuk garis data write buffer VISA Read, karena tidak mendapat output Concatenate Strings, sebagai gantinya, gunakan Empty String Constant, seperti ditunjukkan pada Gambar 5.25.



Gambar 5.23 Icon Not Equal membandingkan data lama dengan data baru

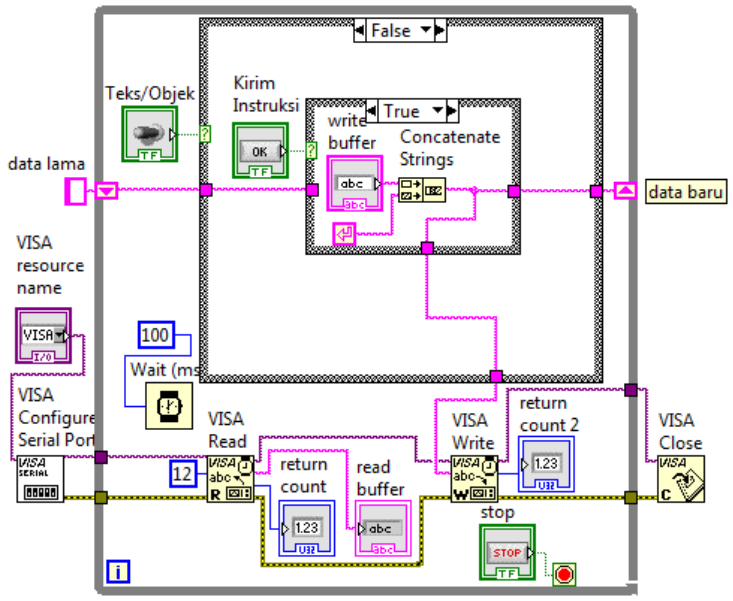


Gambar 5.24 Bila data lama tidak sama dengan data baru maka kirim data

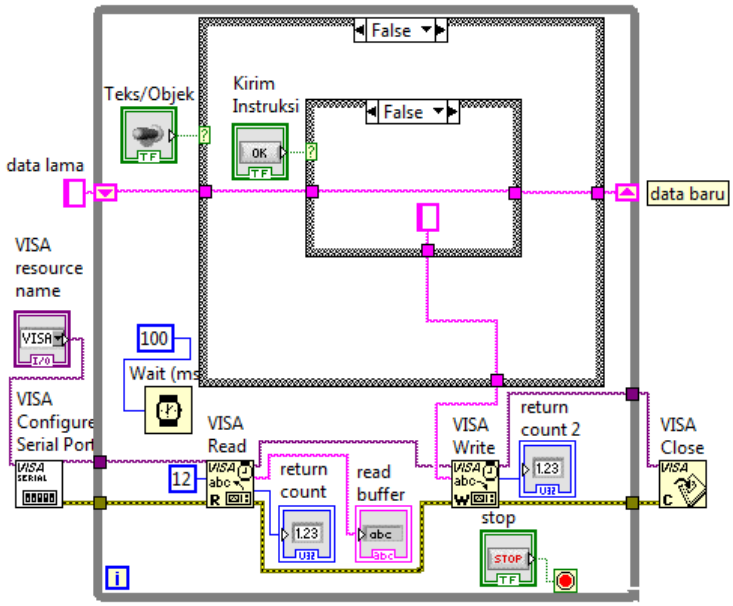


Gambar 5.25 Ketika data lama sama dengan data baru, data tidak dikirim

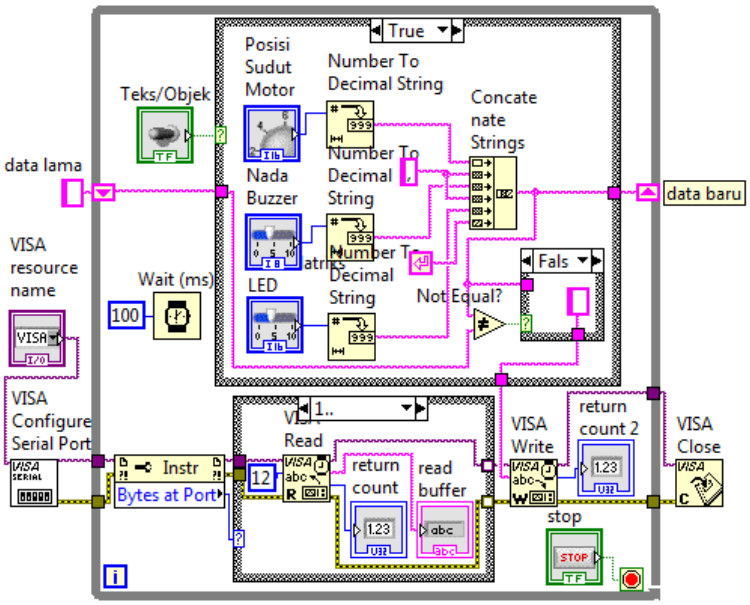
17. Dari Gambar 5.24, ketika data baru tidak sama dengan data lama, maka kirimkan data baru tersebut. Dari Gambar 5.25, ketika data baru sama dengan data lama, maka tidak ada data yang dikirimkan (mengirim Empty String Constant sama halnya dengan tidak ada data yang dikirimkan).
18. Karena setiap kali Case Structure mengeluarkan data, semua Case harus mengeluarkan nilai, maka untuk kotak Case False pada Teks/Objek, buat agar garis data dari terminal kiri Shift Register diteruskan ke terminal kanan Shift Register melalui kotak Case False Kirim Instruksi, seperti ditunjukkan pada Gambar 5.27.
19. Sama seperti pada program LabVIEW yang hanya mengirimkan data ketika ada perubahan nilai objeknya, program di Arduino juga hanya mengirimkan data ketika ada perubahan pada nilai Sensornya (lihat Program_5_5.ino baris 41). Karena itu di LabVIEW juga harus dibuat agar VISA Read hanya membaca apabila ada data yang diterima. Untuk itu tambahkan VISA Bytes at Port, seperti ditunjukkan pada Gambar 5.28.



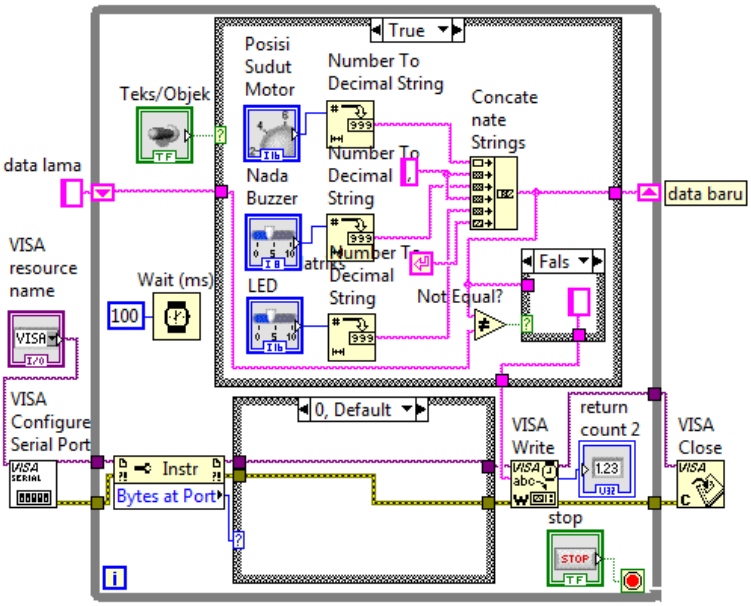
Gambar 5.26 Pada kotak Case False Teks/Objek, data output Concatenate Strings dikirimkan ke VISA Write dan diteruskan ke terminal kanan Shift Register



Gambar 5.27 Pada kotak Case False Teks/Objek, data terminal kiri Shift Register diteruskan ke terminal kanan Shift Register melalui Case False Kirim Instruksi



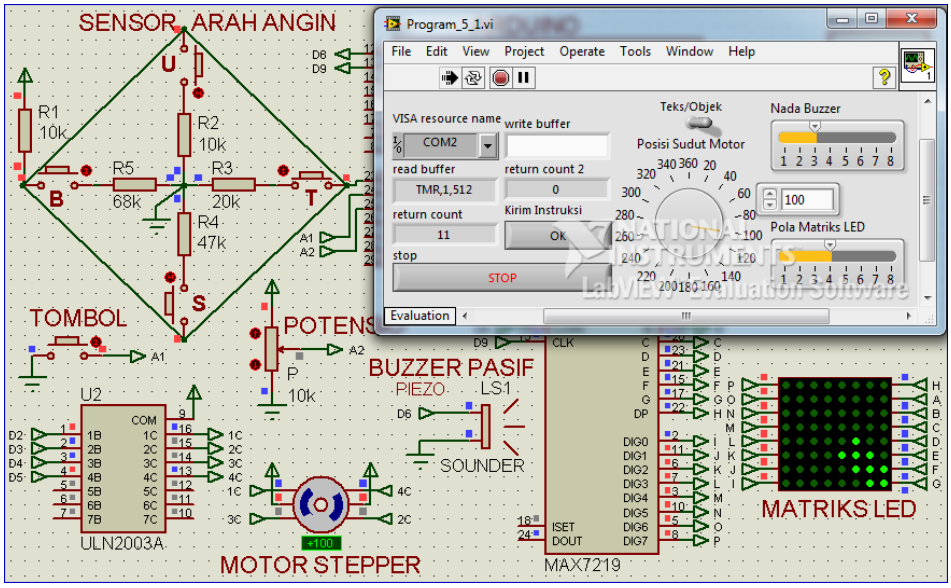
Gambar 5.28 Menambahkan VISA Bytes at Port agar VISA Read hanya membaca bila ada data yang diterima (perhatikan label Case angka 1 diberi titik 2 buah)



Gambar 5.29 Bila tidak ada data yang diterima di Port Serial, di kotak Case Bytes at Port = 0 (Default), garis data diteruskan tanpa ada VISA Read

20. Di Gambar 5.28 tampak kotak Case Bytes at Port dengan label 1.. dan di Gambar 5.29 tampak kotak Case Bytes at Port dengan label 0, Default. Untuk kotak Case dengan label 1.., pembaca harus menambahkan sendiri titik sebanyak 2 buah. Dengan tambahan titik 2 buah di belakang angka 1 tersebut dapat diartikan bahwa Case tersebut untuk Bytes at Port bernilai 1 atau lebih dari 1. Pada kondisi tersebut, VISA Read akan membaca data dengan jumlah byte maksimum sebanyak 12, yang memiliki karakter akhiran New Line (New Line = karakter ASCII 10, lihat pada VISA Configure, di kaki termination char, yang secara default diisi karakter New Line). Kemudian apabila tidak ada data yang diterima, berarti Bytes at Port bernilai 0, maka garis data diteruskan tanpa ada VISA Read.
21. Dengan menggunakan rangkaian Gambar 5.1 di Proteus, dan Program_5_5.ino pada komponen Arduino (ATmega328), jalankan simulasi Proteus dan Program_5_2.vi LabVIEW Gambar 5.29 di atas.
22. Gambar 5.30 berikut ini menunjukkan isi read buffer (TMR, 1, 512) untuk data yang dikirimkan dari Arduino ke LabVIEW, dan tampilan Proteus untuk data objek yang dikirimkan dari LabVIEW ke Arduino, dengan nilai Dial Posisi Sudut = 100, Slider Nada = 3 dan Slider Matriks LED = 4. Perhatikan tampilan Proteus, khususnya pada sudut poros yang dibentuk Motor Stepper, apabila sebelumnya untuk data 100, sudut yang dihasilkan adalah 99, maka dalam program kali ini, sudut yang dicapai sama seperti data objeknya, yaitu 100. Hal ini karena adanya penambahan pada program Arduino, yaitu di baris ke 62 dari Program_5_5.ino, tampak ada penambahan angka 1 pada data posisi motor yang diterima.
23. Dari hasil simulasi, tampak ada satu permasalahan, yaitu ketika Dial Posisi Sudut di Front Panel LabVIEW diputar dari posisi sudut 340^0 searah jarum jam ke posisi 50^0, tampak bahwa poros Motor Stepper berputar bukan searah jarum jam, tetapi malah berputar berlawanan jarum jam, padahal sudut putar yang searah jarum jam hanya 70^0, sedangkan bila berlawanan jarum jam, sudut putarnya menjadi 4 kali lipat, yaitu 290^0. Begitu pula, ketika Dial Posisi Sudut di LabVIEW diputar berlawanan jarum jam dari posisi sudut 40^0 ke posisi 350^0, maka tampak poros Motor Stepper

berputar bukan berlawanan jarum jam, tetapi malah berputar searah jarum jam, padahal sudut putar yang berlawanan jarum jam hanya 50^0, sedangkan bila searah jarum jam, sudut putarnya menjadi 6 kali lipat, yaitu 310^0. Untuk mengatasi hal ini, perlu modifikasi program, yang membuat posisi sudut yang dituju dengan posisi sudut yang sekarang, selisih nilainya tidak boleh melebihi 180^0. Jika ternyata selisih nilainya lebih dari 180^0, maka nilai posisi yang paling kecil, harus ditambah 360^0. Untuk lebih jelasnya, lihat Program_5_6.ino, yaitu di baris 68 dan 69.



Gambar 5.30 Dial Posisi Sudut Motor = 100, Nada Buzzer = 3, Pola Matriks LED = 4

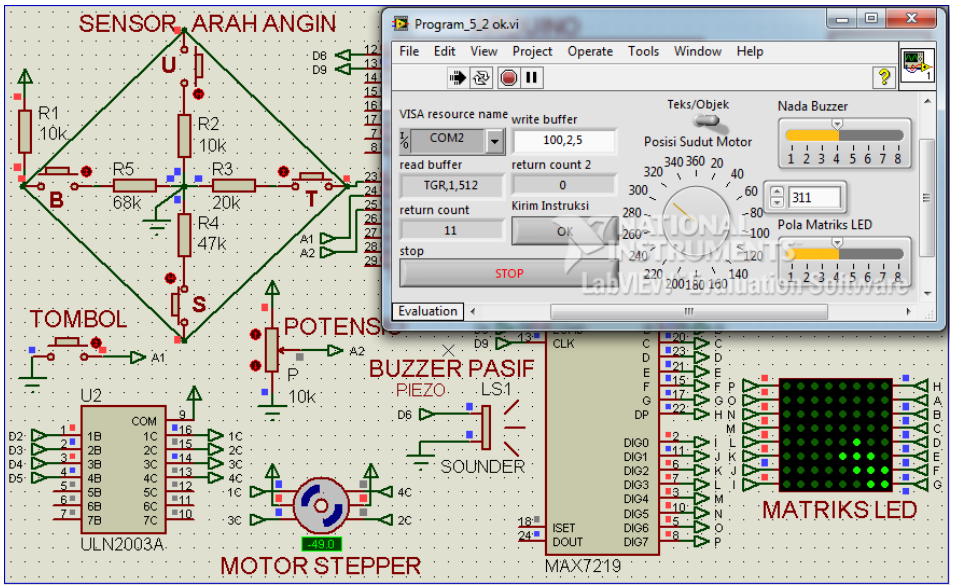
Program_5_6.ino

```
1.  #include <Stepper.h>
2.  #include "LedControl.h"
3.  #define do1  1047
4.  #define re  1175
5.  #define mi  1319
6.  #define fa  1397
7.  #define sol  1568
8.  #define la  1760
9.  #define si  1976
10. #define do2  2093
11. int nada[] = {do1, re, mi, fa, sol, la, si, do2};
12. int a1 = 0;
```

```
int b1 = 0;
int c1 = 0;
int e1 = 0;
const int stepsPerRevolution = 360;
byte pola[8][8] = {
{B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},//1 UTR
{B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},//2 TLU
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},//3 TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},//4 TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},//5 SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},//6 BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},//7 BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} //8 BLU
};
Stepper myStepper(stepsPerRevolution, 2, 3, 4, 5);
LedControl lc = LedControl(7, 9, 8, 1);
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  myStepper.setSpeed(2);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
}
void loop() {
  int a = analogRead(A0);
  int b = digitalRead(A1);
  int c = analogRead(A2);
  if (a != a1 || b != b1 || c != c1) {
    if (a == 512) Serial.print("UTR");
    if (a == 410) Serial.print("TLU");
    if (a == 682) Serial.print("TMR");
    if (a == 598) Serial.print("TGR");
    if (a == 844) Serial.print("SLT");
    if (a == 753) Serial.print("BDY");
    if (a == 892) Serial.print("BRT");
    if (a == 477) Serial.print("BLU");
    Serial.print(',');
    Serial.print(b);
    Serial.print(',');
    Serial.println(c);
    a1 = a;
    b1 = b;
    c1 = c;
  }
}
void serialEvent() {
  while (Serial.available()) {
    int e = Serial.parseInt() + 1; //posisi Motor
    int f = Serial.parseInt(); //nada Buzzer
    int g = Serial.parseInt(); //pola Matriks
```

```
64.     if (Serial.read() == char(13)) {
65.       for (int i = 0; i < 8; i++) {
66.         lc.setRow(0, i, pola[g - 1][i]);
67.       }
68.       if (e1 - e > 180) e = e + 360;
69.       if (e - e1 > 180) e1 = e1 + 360;
70.       tone(6, nada[f - 1], (e1 – e) * 100);
71.       myStepper.step(e1 - e);
72.       e1 = e;
73.       noTone(6);
74. }}}
```

24. Berikut ini hasil simulasi Proteus untuk Program_5_6.ino dan Program_5_2.vi. Tampak Proteus menampilkan nilai -49^0 pada Motor Stepper, saat Dial Posisi Sudut Motor diubah dari posisi sudut 40^0 ke 311^0. Proteus menampilkan nilai -49^0, bukan 311^0, walaupun sebenarnya sama, untuk menunjukkan bahwa poros Motor Stepper berputar berlawanan jarum jam, dari 40^0 melewati 360^0 ke 311^0.



Gambar 5.31 Dial Posisi Sudut Motor di LabVIEW diputar dari 40^0 ke 311^0, Motor Stepper di Proteus berputar berlawanan jarum jam dari posisi +40^0 ke -49^0

5.3 Penggabungan Program

Sub Bab Penggabungan Program ini akan menggabungkan program-program sebelumnya. Apabila di Bab 3 pembaca dapat menampilkan data Sensor dalam bentuk teks dan gambar, kemudian di Bab 4 pembaca dapat menyajikan data Sensor dalam bentuk grafik, tabel dan penyimpanan datanya, dan kemudian dilanjutkan di Bab 5 ini pembaca dapat melakukan pengendalian aktuator baik secara teks maupun objek, maka sebagai penutup di bagian akhir dari Bab 5 ini pembaca akan menggabungkan semua program tersebut. Dengan penggabungan program ini, maka interface Sensor dan Aktuator menggunakan Arduino dan LabVIEW telah dapat direalisasikan. Berikut ini hal-hal yang akan ditambahkan pada program Arduino dan LabVIEW yang telah dibuat di Sub Bab 5.2:

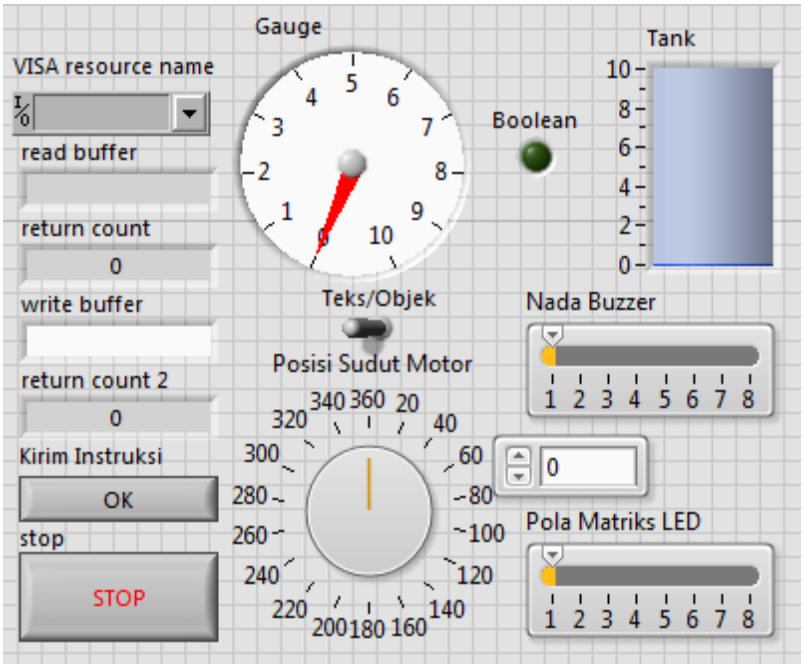
1. Penambahan 3 Indicator untuk 3 Sensor.
   Setelah di Sub Bab 5.2 dibuat pengendalian 3 buah Aktuator (Motor Stepper, Buzzer, Matriks LED) dengan 2 pilihan; pilihan pertama menggunakan teks, dan pilihan kedua melalui 3 objek Control (berupa Dial, Slider1 dan Slider2), maka di Sub Bab 5.3.1 ini akan ditambahkan 3 objek Indicator (berupa Gauge, LED dan Tank), untuk menampilkan data 3 buah Sensor (yaitu Sensor Arah Angin, Tombol dan Potensio).
2. Penambahan Grafik dan Tabel.
   Di Sub Bab 5.3.2, data ketiga buah Sensor disajikan dalam bentuk grafik 2D Compass, Waveform Graph dan Table beserta catatan waktunya.
3. Penyimpanan ke dalam File dan Pemanggilannya
   Di Sub Bab 5.3.3 data ketiga buah Sensor disimpan beserta catatan waktunya ke dalam sebuah file dan dapat dibuka dari tampilan program.
4. Pengaturan Pengendalian Aktuator
   Di Sub Bab 5.3.4, ditambahkan 3 pilihan pengaturan pengendalian Aktuator. Pilihan pertama adalah melalui komputer, yaitu secara teks atau objek. Pilihan kedua adalah otomatis dari Sensor Arah Angin. Pilihan ketiga adalah secara manual menggunakan Potensio.

Berikut ini Sub Bab 5.3.1 hingga 5.3.4 yang berisi keempat penambahan di atas.

5.3.1 Penambahan 3 Indicator untuk 3 Sensor

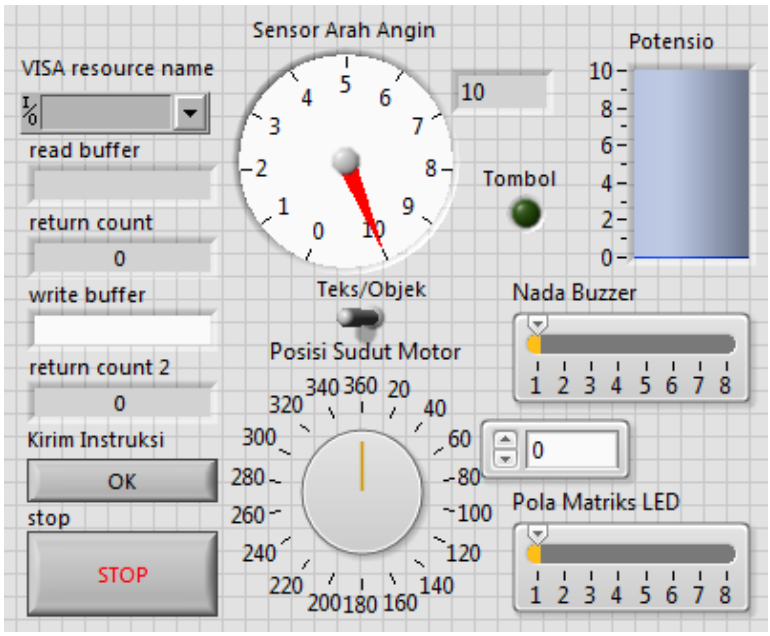
Berikut langkah-langkahnya:

1. Buka kembali Program_5_2.vi. Di jendela Front Panel, tambahkan sebuah Gauge (Modern, Numeric), Round LED (Modern, Boolean) dan Tank (Modern, Numeric) yang diambil dari Palet Controls. Atur ulang penempatan objek-objek sehingga seperti Gambar 5.32 berikut ini.

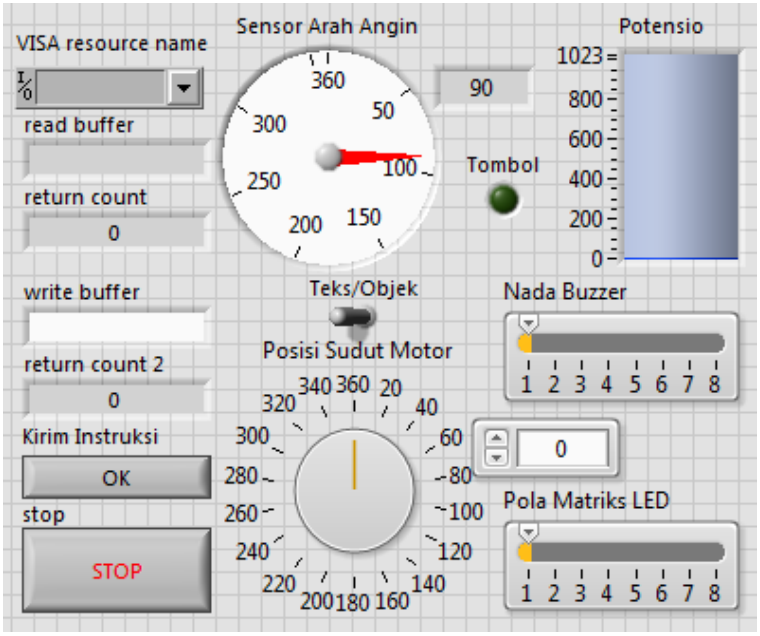


Gambar 5.32 Menambahkan objek Gauge, Round LED Boolean dan Tank

2. Munculkan Digital Display pada Gauge dengan mengklik kanan, pilih Visible Items, pilih Digital Display. Ganti nama label Gauge dengan Sensor Arah Angin, Boolean dengan Tombol, Tank dengan Potensio, seperti ditunjukkan pada Gambar 5.33.
3. Ubah tipe data Sensor Arah Angin dan Potensio dari Double ke Integer 16 bit dengan meng-klik kanan, pilih Representation, pilih I16. Ubah skala nilai Sensor Arah Angin dari 0-10 menjadi 0-360 dengan nilai mengelilingi satu lingkaran penuh, dan Potensio dari 0-10 menjadi 0-1023 seperti ditunjukkan pada Gambar 5.34.

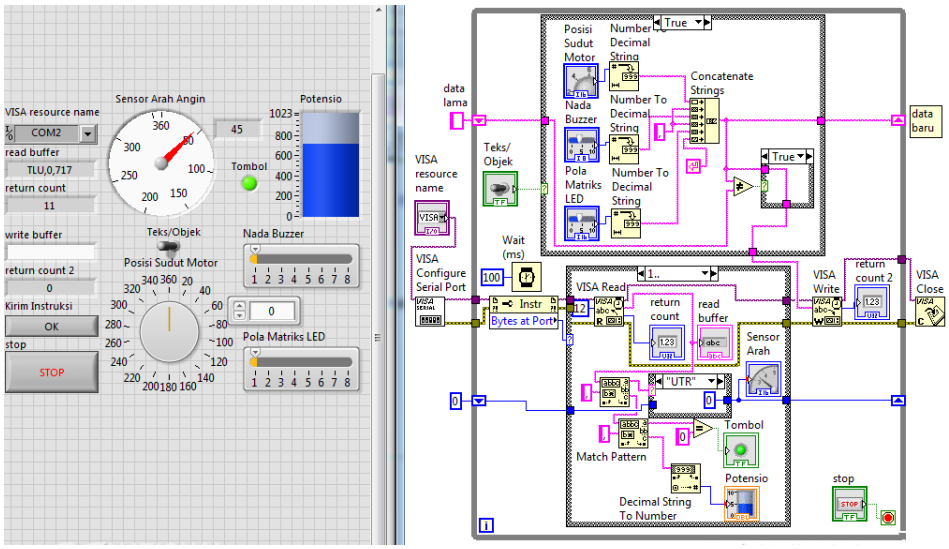


Gambar 5.33 Mengganti nama dengan Sensor Arah Angin, Tombol dan Potensio

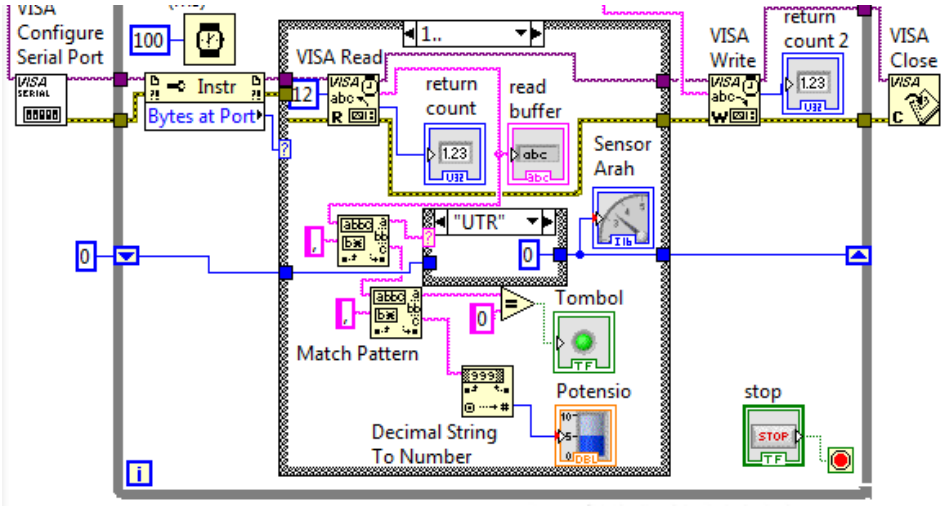


Gambar 5.34 Mengatur tipe data dan skala Sensor Arah Angin dan Potensio

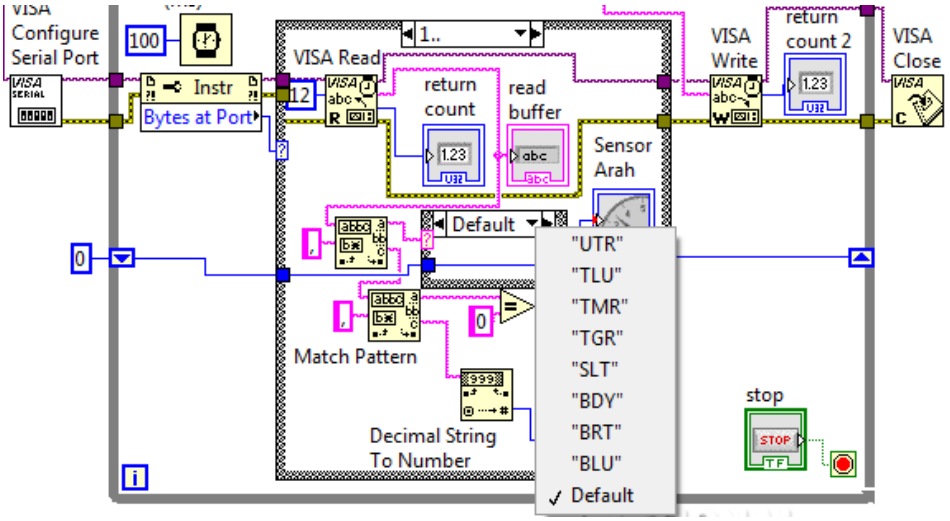
4. Di jendela Block Diagram, masukkan icon Sensor Arah Angin, Tombol dan Potensio ke dalam Case Structure Bytes At Port. Tambahkan 2 buah Match Pattern untuk memisahkan data yang diterima oleh VISA Read, berdasarkan tanda koma. Data sebelum koma pertama adalah data arah angin, data setelah koma pertama adalah data tombol, dan data setelah koma kedua adalah data potensio. Untuk data arah angin, karena ada 8 variasi, maka seperti di Gambar 3.38, tambahkan Case Structure, dengan label berupa UTR, TLU, TMR, TGR, SLT, BDY, BRT dan BLU. Tambahkan Case Default, dengan label kosong. Isi pada semua Case tersebut berturut-turut dengan angka 0, 45, 90, 135, 180, 225, 270, 315 dan pada Case Default, teruskan data Shift Register di terminal kiri hingga ke terminal kanan, seperti ditunjukkan pada Gambar 5.35. Untuk data tombol, tambahkan icon Equal, untuk membandingkan apakah data tersebut berisi String "0" atau "1". Jika "0", maka nyalakan LED, jika "1" padamkan LED. Untuk data potensio, tambahkan icon Decimal String to Number, untuk mengubah data String menjadi Integer. Untuk label Case 0, Default pada Bytes at Port, teruskan data ke VISA Write (lihat Gambar 5.29).



Gambar 5.35 Di Block Diagram, icon Sensor Arah Angin, Tombol dan Potensio dimasukkan ke Case Structure Bytes at Port di Case label 1.



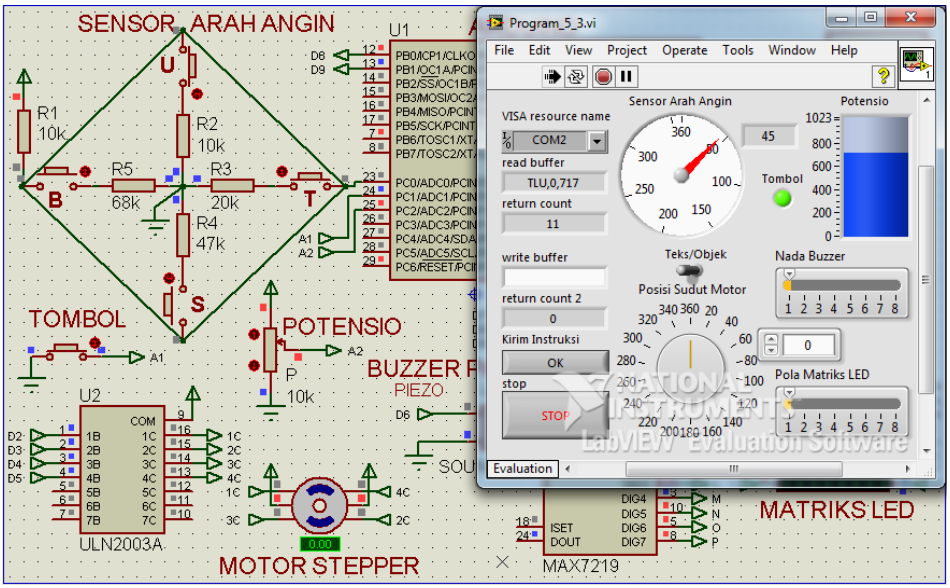
Gambar 5.36 Memperbesar gambar Case Structure Bytes at Port label 1..



Gambar 5.37 Isi Case Structure data Arah Angin, dengan Default diisi dengan data yang diambil dari Shift Register kiri untuk diteruskan ke Shift Register kanan

Catatan: Mengapa perlu menambahkan Shift Register untuk data Arah Angin? Agar jarum pada Gauge tidak lari ke 0, atau ke label UTR ketika data yang diterima dari Arduino adalah data kosong. Data kosong ini bisa muncul ketika terjadi perpindahan posisi penekanan dari keempat tombol Arah Angin di Proteus.

5. Jalankan simulasi Proteus dan program LabVIEW di atas. Gambar 5.38 menunjukan hasil simulasi Proteus dan LabVIEW. Tampak bahwa Gauge Sensor Arah Angin menunjukkan 45 derajat ketika tombol U dan T di Sensor Arah Angin ditekan. LED Tombol menyala ketika Tombol di luar Sensor Arah Angin ditekan, Objek Tank Potensio terisi hingga 717 ketika Potensio digeser naik. Saran: agar nilai Tank Potensio bisa terlihat jelas, klik kanan objek Tank, pilih Visible Items, pilih Digital Display.

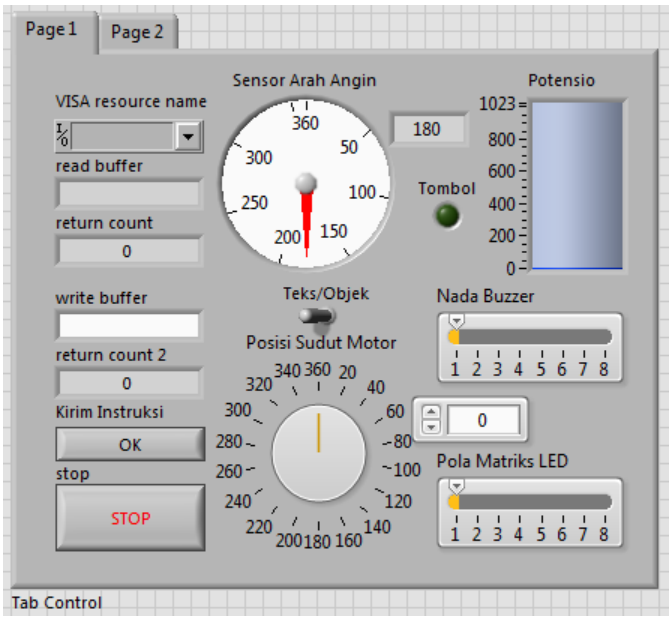


Gambar 5.38 LabVIEW menampilkan data ketiga Sensor dalam teks (TLU, 0, 717) dan objek (Gauge Arah Angin = 45, LED Tombol ON, Tank Potensio = 717)

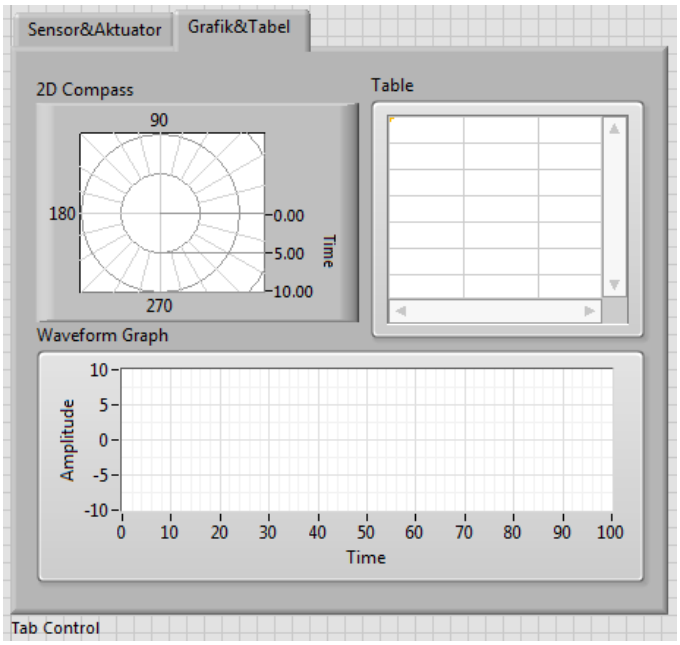
5.3.2 Penambahan Grafik dan Tabel

Berikut langkah-langkahnya:

1. Tambahkan Tab Control (Modern, Containers) dan tempatkan semua objek di Page 1 seperti ditunjukkan pada Gambar 5.39.
2. Ubah nama Tab Control Page 1 menjadi Sensor&Aktuator dan Page 2 menjadi Grafik&Tabel. Kemudian tempatkan objek 2D Compass (Modern, Graph), Tabel (Silver, List Table & Tree) dan Waveform Graph (Silver, Graph) di Grafik&Tabel seperti ditunjukkan pada Gambar 5.40.



Gambar 5.39 Menempatkan semua objek di Page 1 Tab Control



Gambar 5.40 Mengubah nama Page1 menjadi Sensor&Aktuator, Page2 menjadi Grafik&Tabel, dan menempatkan 2D Compass, Table dan Waveform Graph

3. Diinginkan 2D Compass dapat menampilkan data Sensor Arah Angin, Waveform Graph dapat menampilkan data Tombol dan Potensio, dan Table dapat menampilkan ketiga data tersebut dilengkapi catatan waktu. Agar data yang ditampilkan cukup memadai, maka diinginkan ketiganya dapat menampilkan data perdetik dengan rentang waktu 5 menit.
4. Agar Arduino bisa mengirimkan data perdetik ke LabVIEW secara terus-menerus, tanpa membuat Arduino menjadi terlambat dalam menerima data, maka pembaca perlu menggunakan instruksi tanpa delay (lihat Example BlinkWithoutDelay di software Arduino IDE). Berikut ini program Arduino untuk mengirimkan data perdetik ke LabVIEW tanpa delay.

Program_5_7.ino

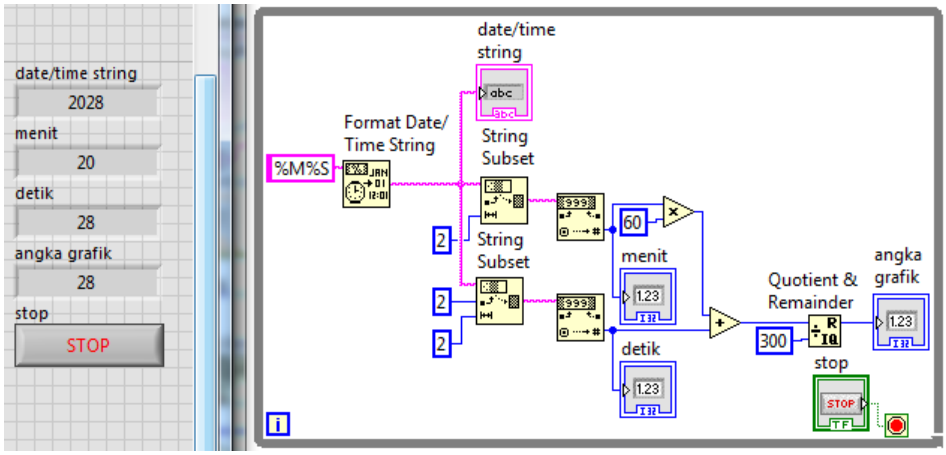
```
#include <Stepper.h>
#include "LedControl.h"
#define do1  1047
#define re  1175
#define mi  1319
#define fa  1397
#define sol  1568
#define la  1760
#define si  1976
#define do2  2093
int nada[] = {do1, re, mi, fa, sol, la, si, do2};
int a1 = 0;
int b1 = 0;
int c1 = 0;
int e1 = 0;
const int stepsPerRevolution = 360;
byte pola[8][8] = {
{B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},//1 UTR
{B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},//2 TLU
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},//3 TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},//4 TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},//5 SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},//6 BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},//7 BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} //8 BLU
};
Stepper myStepper(stepsPerRevolution, 2, 3, 4, 5);
LedControl lc = LedControl(7, 9, 8, 1);
unsigned long sebelum = 0;
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  myStepper.setSpeed(2);
```

```
34.   lc.shutdown(0, false);
35.   lc.setIntensity(0, 8);
36.   lc.clearDisplay(0);
37. }
38. void loop() {
39.   int a = analogRead(A0);
40.   int b = digitalRead(A1);
41.   int c = analogRead(A2);
42.   unsigned long sekarang = millis();
43.   if (sekarang - sebelum >= 1000) {
44.     sebelum = sekarang;
45.     if (a == 512) Serial.print("UTR");
46.     if (a == 410) Serial.print("TLU");
47.     if (a == 682) Serial.print("TMR");
48.     if (a == 598) Serial.print("TGR");
49.     if (a == 844) Serial.print("SLT");
50.     if (a == 753) Serial.print("BDY");
51.     if (a == 892) Serial.print("BRT");
52.     if (a == 477) Serial.print("BLU");
53.     Serial.print(',');
54.     Serial.print(b);
55.     Serial.print(',');
56.     Serial.println(c);
57.     a1 = a;
58.     b1 = b;
59.     c1 = c;
60.   }
61. }
62. void serialEvent() {
63.   while (Serial.available()) {
64.     int e = Serial.parseInt() + 1; //posisi Motor
65.     int f = Serial.parseInt(); //nada Buzzer
66.     int g = Serial.parseInt(); //pola Matriks
67.     if (Serial.read() == char(13)) {
68.       for (int i = 0; i < 8; i++) {
69.         lc.setRow(0, i, pola[g - 1][i]);}
70.       if (e1 - e > 180) e = e + 360;
71.       if (e - e1 > 180) e1 = e1 + 360;
72.       tone(6, nada[f - 1], (e1 - e) * 100);
73.       myStepper.step(e1 - e);
74.       e1 = e;
75.       noTone(6);
76. }}}
```

5. Sebagai ganti delay, tampak pada baris ke 42, penambahan fungsi millis(). Fungsi millis() adalah fungsi yang dapat menghasilkan nilai waktu saat itu dengan satuan milidetik. Agar dapat mengirimkan data setiap detik tanpa delay dilakukan dengan membandingkan 2 buah variabel, di mana

variabel pertama diisi nilai millis() dan variabel kedua diisi dengan nilai variabel pertama. Apabila selisih kedua variabel tersebut sudah sama dengan atau melampaui 1000 milidetik, maka data dikirimkan.

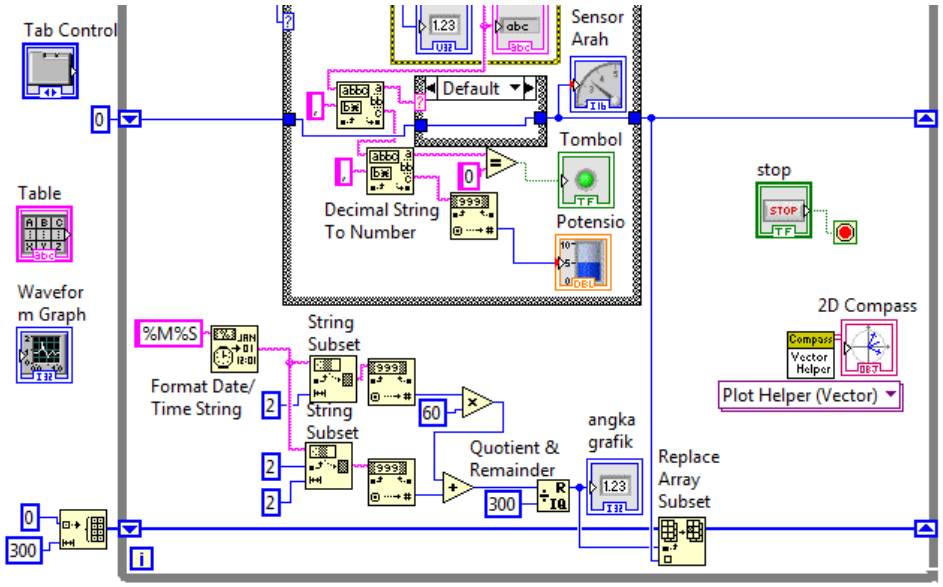
6. Berikutnya, agar grafik 2D Compass memiliki rentang waktu 5 menit; yang berarti bahwa setiap 5 menit atau 300 detik, grafik akan dimulai lagi dari awal, atau dari ujung kiri, maka perlu menambahkan fungsi Quotient&Remainder. Input x (angka yang dibagi) adalah nilai waktu saat ini dan input y (angka pembagi) adalah 300 (dijadikan detik). Untuk lebih jelasnya, lihat program Gambar 5.41 berikut ini.



Gambar 5.41 Membuat rentang waktu 5 menit atau 300 detik, dengan Quotient& Remainder, setiap kali angka grafik mencapai 300, angka grafik akan kembali ke 0

7. Dari program di Gambar 5.41 di atas, ketika nilai menit mencapai 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, atau 55 (setiap kelipatan 5), angka grafik akan dimulai lagi dari 0. Dengan memasukkan angka grafik tersebut sebagai titik koordinat x, dan nilai sensor sebagai titik koordinat y, maka dapat dibuat grafik perdetik dengan rentang waktu 5 menit.
8. Tambahkan sebuah Shift Register untuk menampung data grafik. Beri data awal di terminal kiri Shift Register dengan icon Initialize Array, dengan input element diisi 0 dan dimension size diisi 300.
9. Tambahkan sebuah icon Replace Array Subset, yang akan mengganti isi data di Shift Register dengan data yang baru. Hubungkan input array icon

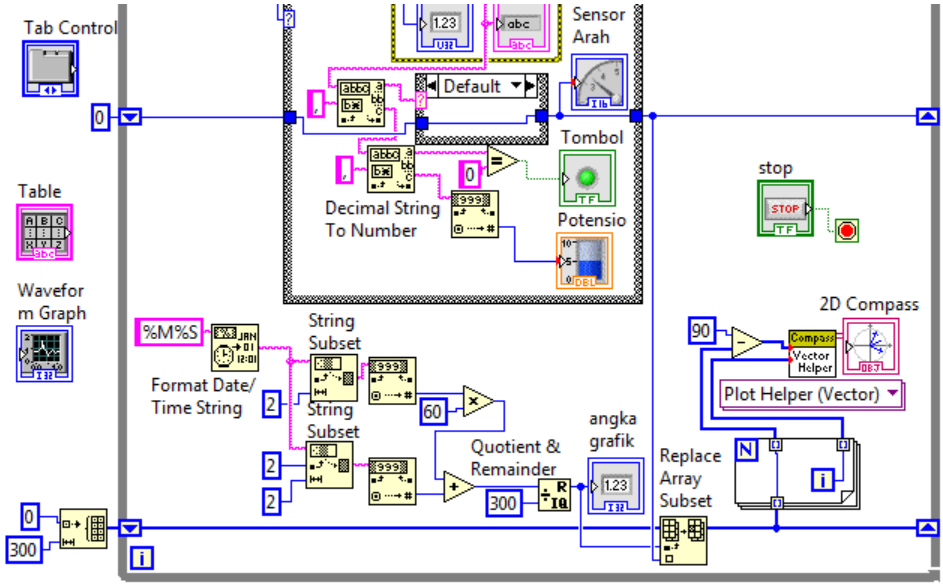
tersebut dengan terminal kiri Shift Register dan output array icon dengan terminal kanan Shift Register. Kemudian hubungkan input indeks dengan angka grafik dan input new element dengan garis data Shift Register Sensor Arah Angin seperti ditunjukkan pada Gambar 5.42.



Gambar 5.42 Menambahkan Replace Array Subset untuk mengganti data grafik (waktu dan nilai Sensor) di Shift Register dengan data yang baru

10. Posisi angka 0^0 di 2D Compass berbeda dengan di Sensor Arah Angin. Tidak hanya posisi angka 0^0, arah putaran juga berbeda. Titik 0^0 Sensor Arah Angin ada di posisi 90^0 2D Compass. Sedangkan posisi 0^0 2D Compass ada di posisi 90^0 Sensor Arah Angin. Di samping itu, bila Sensor Arah Angin diputar searah jarum jam, maka nilai sudutnya akan membesar dan akan mengecil apabila diputar berlawanan jarum jam. Sebaliknya di 2D Compass, sudutnya akan mengecil apabila diputar searah jarum jam, dan akan membesar apabila diputar berlawanan jarum jam. Untuk mengatasi perbedaan ini, tambahkan angka 90, dan kurangkan angka tersebut dengan garis data grafik. Masukkan hasil pengurangan 90 dengan garis data grafik sebagai input theta 2D Compass.

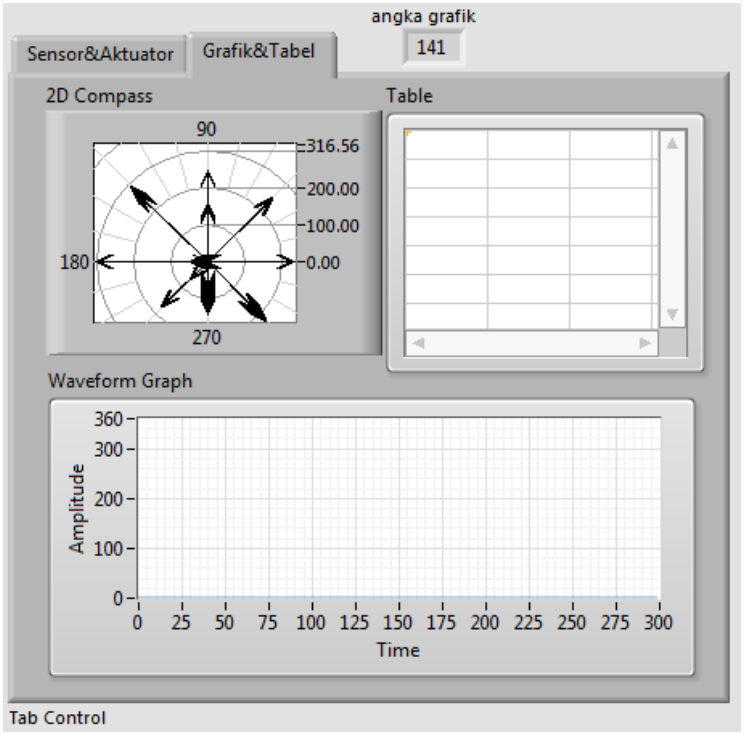
11. Untuk input radius 2D Compass, karena isinya harus berupa deretan angka dari 0 sampai 299, maka gunakan For Loop dengan input N tidak perlu diisi, tetapi diambilkan dari index garis data theta, yang dibuat dengan cara melewatkan garis data ke kotak For Loop. Kemudian ambil nilai iterasi (i) yang dihasilkan For Loop sebagai nilai input radius 2D Compass, seperti ditunjukkan pada Gambar 5.43



Gambar 5.43 Mengurangkan garis data grafik dari 90 untuk input theta dan menambahkan For Loop dengan nilai i For Loop untuk input radius

12. Jalankan simulasi Proteus dan program LabVIEW di atas. Lakukan pengubahan data Sensor Arah Angin dengan menekan tombol U, T, S, B atau kombinasinya di Proteus, dan perhatikan grafik di 2D Compass. Gambar 5.44 berikut ini menunjukkan data Sensor Arah Angin perdetik yang ditampilkan di Grafik 2D Compass dengan rentang waktu 5 menit.
13. Berikutnya adalah menampilkan data Tombol dan Potensio pada Waveform Graph. Karena Potensio memiliki jangkauan nilai dari 0 – 1023, maka agar data Tombol dapat terlihat jelas pada grafik, nilai Tombol dibuat menjadi 750 ketika ditekan, dan 250 ketika tidak ditekan. Untuk itu

tambahkan icon Select dengan input s terhubung ke Tombol, input t ke 750, dan input f ke 250 seperti ditunjukkan Gambar 5.45.

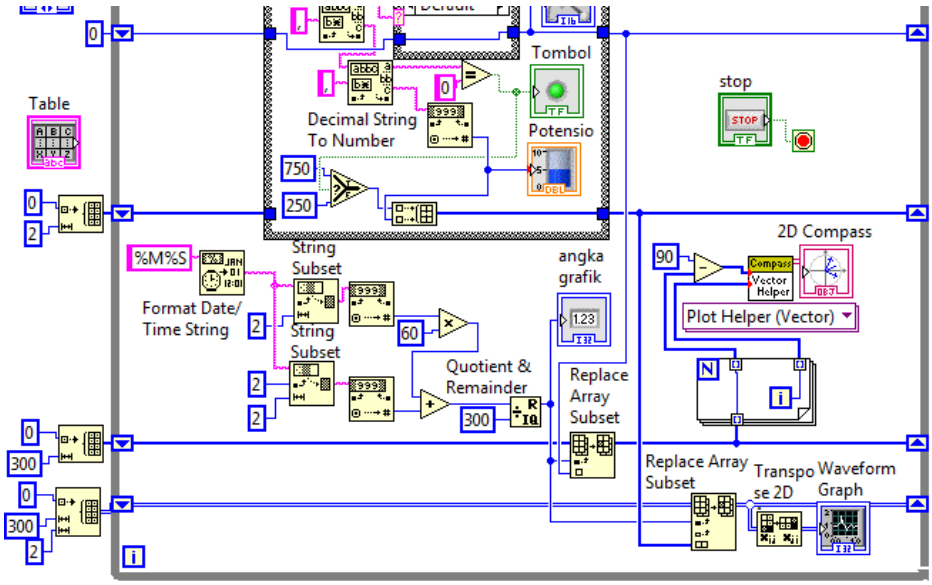


Gambar 5.44 Grafik 2D Compass menampilkan data Sensor Arah Angin perdetik dengan rentang waktu 5 menit (perhatikan angka grafik sudah di 141 detik)

14. Tambahkan Build Array untuk menggabungkan data Tombol dan Potensio. Tarik ke bawah icon Build Array tersebut hingga memiliki 2 input. Hubungkan input 1 dengan Tombol dan input 2 dengan Potensio.
15. Tambahkan Shift Register untuk menyimpan data gabungan Tombol dan Potensio, dengan nilai awal Shift Register diisi Initialize Array, dengan input element diisi 0 dan dimension size diisi 2.
16. Agar Waveform Graph bisa menampilkan 2 data dalam satu grafik (multiplot), maka input Waveform Graph harus berupa transpose dari array 2D (harus di-transprose karena secara default, array 2D menyusun data berdasarkan kolom, yaitu data1 di kolom 1 dan data2 di kolom 2.

Sedangkan Waveform Graph menampilkan data berdasarkan baris, yaitu data di baris 1 sebagai grafik 1, data di baris 2 sebagai grafik 2).

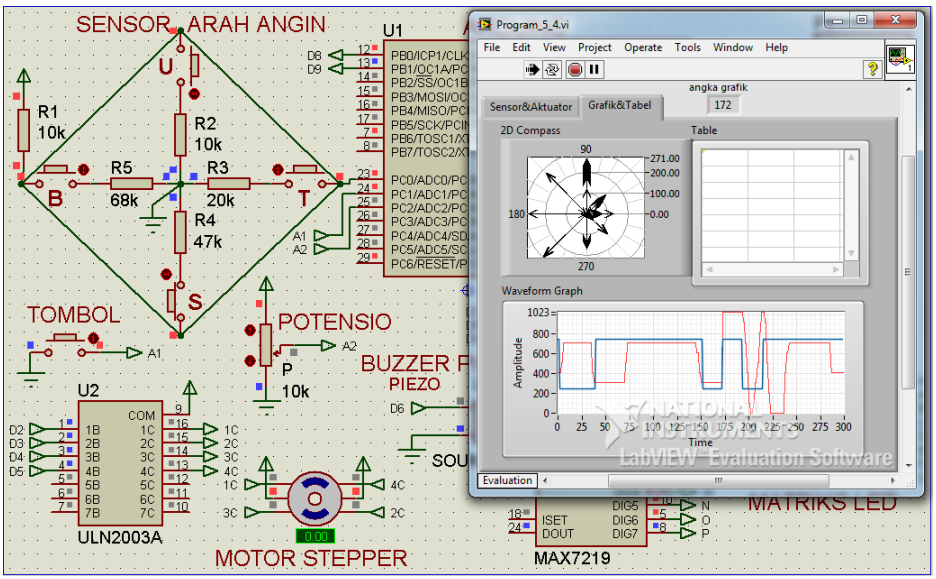
17. Untuk menghasilkan array 2D, tambahkan Shif Register dengan nilai awal diisi Initialize Array dengan 2 input dimension size (tarik ke bawah icon Initialize Array untuk memunculkan input dimension size tambahan). Isi element dengan 0, dimension size 1 diisi 300 dan dimension size 2 diisi 2.
18. Tambahkan icon Replace Array Subset. Isi input index (row) dengan garis data angka grafik, input index (col) dikosongkan (karena semua kolom dalam 1 baris akan diisi dengan data yang baru), dan input new element diisi dengan garis data Shift Register Tombol dan Potensio.
19. Kemudian hubungkan garis data Shift Register array 2D tersebut dengan icon Transpose 2D Array (Programming, Array). Hubungkan output Transpose 2D Array ke Waveform Graph seperti Gambar 5.45 berikut.



Gambar 5.45 Menampilkan data Tombol dan Potensio di Waveform Graph

Catatan: Apabila gambar program tidak terlihat jelas, pembaca dapat membuka Program_5_4_b.vi yang ada di CD pendukung buku ini.

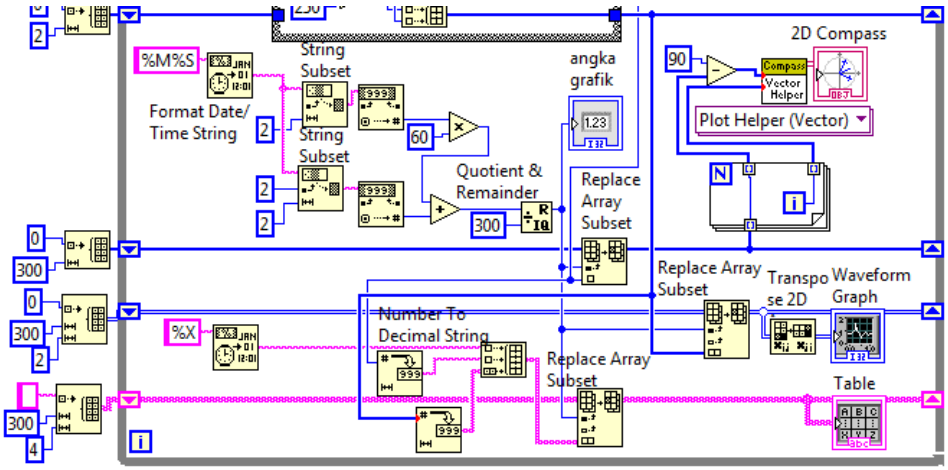
20. Jalankan simulasi Proteus, dan program LabVIEW Gambar 5.45. Tekan tombol dan geser Potensio, seharusnya Waveform Graph menampilkan data Tombol dan Potensio tersebut, seperti Gambar 5.46 berikut.
21. Berikutnya adalah menampilkan ketiga data Sensor tersebut pada Table yang disertai dengan catatan waktu. Untuk itu tambahkan sebuah Shift Register untuk menyimpan data Table. Isi data awal Shift Register tersebut dengan Initialize Array, dengan element Empty String Constant, dan dimension size 1 diisi 300 dan dimension size 2 diisi 4 (karena ada 4 jenis data, yaitu catatan waktu, Sensor Arah, Tombol dan Potensio).



Gambar 5.46 Hasil simulasi grafik Tombol dan Potensio di Waveform Graph

22. Tambahkan sebuah icon Replace Array Subset untuk mengganti data Table dengan data yang baru. Beri input index (row) dengan garis data angka grafik, input index (col) dikosongkan, dan input new element dengan gabungan data catatan waktu, Sensor Arah, Tombol dan Potensio.
23. Gunakan icon Build Array untuk menggabungkan keempat data tersebut.
24. Untuk catatan waktu, gunakan format %X yang akan menampilkan waktu dalam jam, menit dan detik diikuti keterangan (AM/PM).

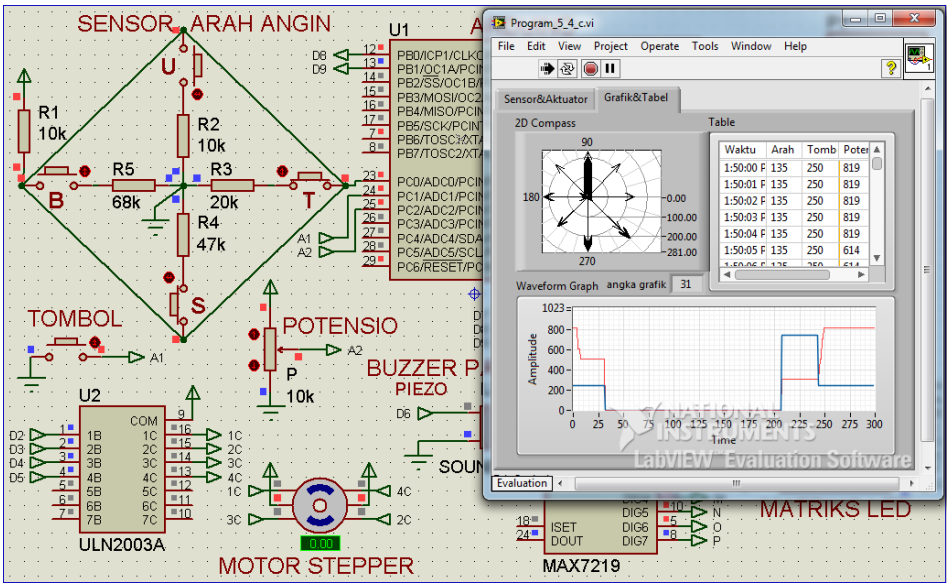
25. Karena Table hanya menerima input dengan tipe data String, maka ubah data Sensor Arah, Tombol dan Potensio menjadi String dengan icon Number to Decimal String.
26. Terakhir, hubungkan data Shift Register ke input icon Table. Namun sebelum menghubungkan, ubah dulu icon Table dari Control menjadi Indicator, dengan klik kanan icon Table, dan pilih Change to Indicator.



Gambar 5.47 Menampilkan ketiga data sensor dengan catatan waktu pada Table

Catatan: Apabila gambar program tidak terlihat jelas, pembaca dapat membuka Program_5_4_c.vi yang ada di CD pendukung buku ini.

27. Agar Table memiliki judul di setiap kolomnya, klik kanan objek Table di Front Panel, dan pilih Visible Items, klik untuk mencentang Column Headers. Tuliskan pada Column Headers tersebut kata “Waktu” di kolom 1, “Arah” di kolom 2, “Tombol” di kolom 3 dan “Potensio” di kolom 4.
28. Jalankan simulasi Proteus dan program LabVIEW Gambar 5.47 di atas. Tekan tombol U, T, B atau S di Sensor Arah Angin, dan juga tombol di luar Sensor Arah Angin, serta geser Potensio. Gambar 5.48 berikut ini menunjukkan hasil program yang menampilkan catatan waktu, Sensor Arah, Tombol dan Potensio pada Table.

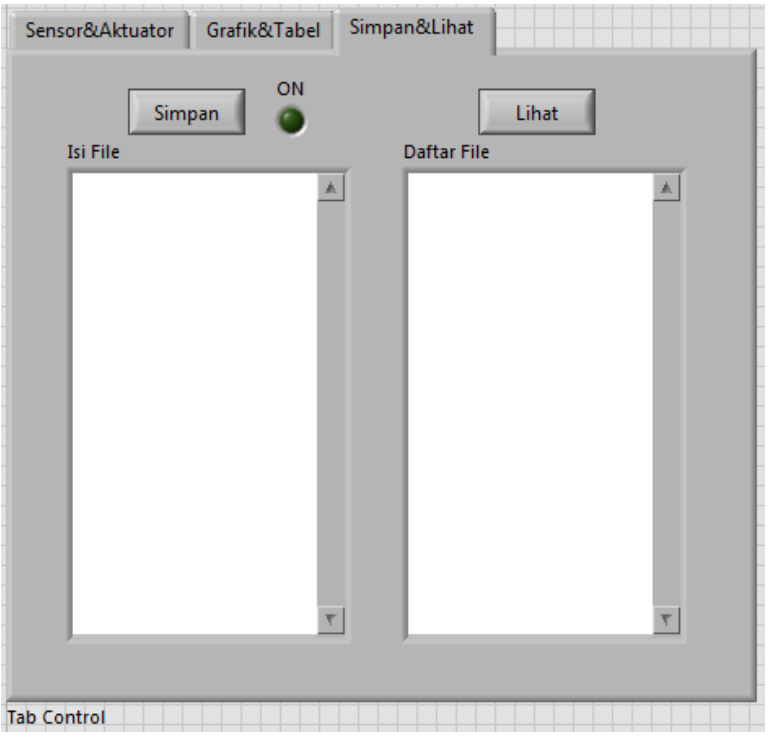


Gambar 5.48 Hasil simulasi ketiga data sensor (Sensor Arah, Tombol, Potensio) dan catatan waktunya ditampilkan pada 2D Compass, Waveform Graph dan Table

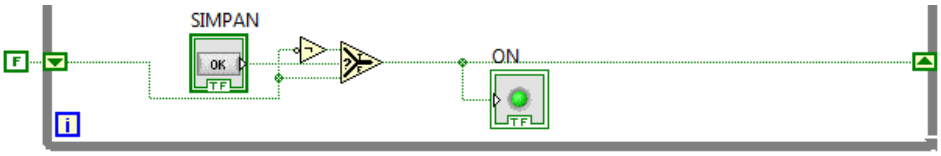
5.3.3 Penyimpanan Data ke File dan Pemanggilannya

Berikut langkah-langkahnya:

1. Berikutnya, tambahkan sebuah Page lagi pada Tab Control, dengan meng-klik kanan Tab Grafik&Table, dan pilih Add Page After. Beri nama Page yang baru dengan Simpan&Lihat.
2. Isi Page Simpan&Lihat ini adalah sama seperti Sub Bab 4.3, yaitu berisi Listbox data yang disimpan dan daftar nama file, serta tombol untuk menyimpan dan melihat daftar file. Untuk itu, tambahkan 2 OK Button, 1 Round LED dan 2 Listbox. Beri nama tombol OK Button 1 dengan Simpan, OK Button 2 dengan Lihat, Round LED dengan ON, Listbox 1 dengan Isi File, Listbox 2 dengan Daftar File, seperti ditunjukkan pada Gambar 5.49.
3. Buat program untuk menghidup/matikan LED dengan tombol Simpan Ketika tombol ditekan sekali, LED ON hidup, ketika tombol ditekan lagi, LED ON padam, demikian seterusnya, seperti ditunjukkan Gambar 5.50.

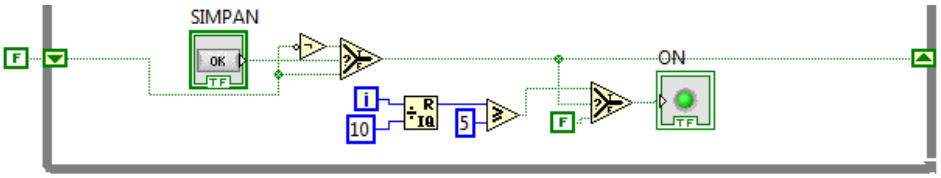


Gambar 5.49 Tombol Simpan, Tombol Lihat, LED ON, Listbox Isi File, Daftar File



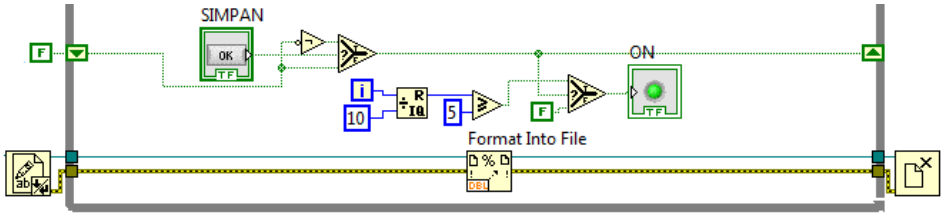
Gambar 5.50 Program tombol Simpan untuk menghidup/matikan LED ON

4. Buat program agar LED ON dapat berkedip ketika dihidupkan, seperti ditunjukan Gambar 5.51 berikut.



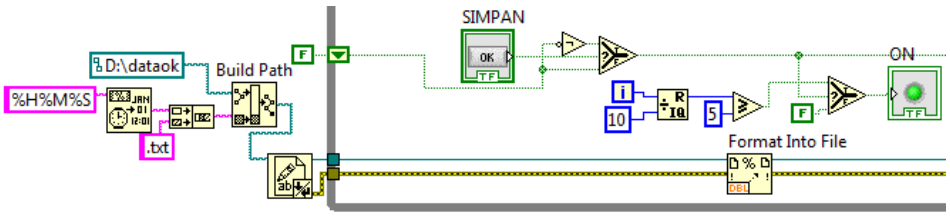
Gambar 5.51 Program membuat LED ON berkedip ketika dihidupkan

5. Berikutnya adalah program untuk penyimpanan data. Tambahkan icon Write to Text File, Format Into File dan Close File. Hubungkan ketiganya, dengan icon Format Into File ditempatkan di tengah dan di dalam While Loop, seperti Gambar 5.52 berikut.



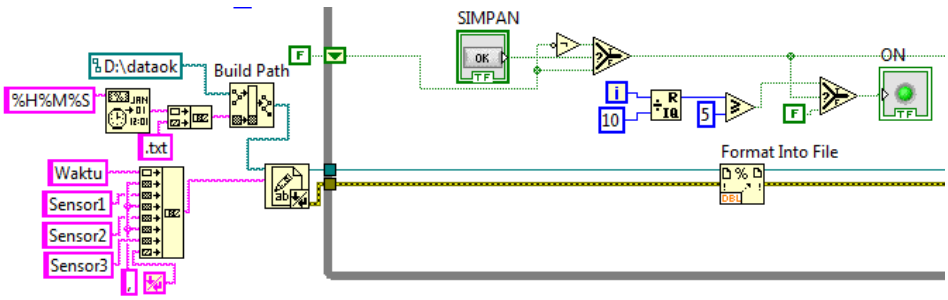
Gambar 5.52 Menempatkan icon Write to Text File, Format Into File dan Close File

6. Pastikan di Drive D terdapat folder dengan nama dataok. Apabila belum ada, buat folder dengan nama dataok. Agar file hasil penyimpanan dapat ditempatkan pada folder tersebut, gunakan icon Build Path, klik kanan pada input base path, pilih Create Constant, dan ketikkan D:\dataok. Untuk input name path Build Path, isi dengan nilai waktu jam menit detik yang dihasilkan oleh icon Format Date/Time String, dan tambahkan .txt di akhir nama file, seperti ditunjukkan Gambar 5.53 berikut.



Gambar 5.53 Hubungkan input file Write to Text File dengan path D:\dataok diikuti nama file yang diambil dari nilai waktu jam, menit dan detik

7. Agar isi file memiliki judul atau keterangan data di awal baris, tambahkan kata Waktu, Sensor1, Sensor2, Sensor3 diikuti Enter di input text icon Write to Text File, seperti ditunjukkan Gambar 5.54.
8. Pada icon Format Into File, tarik ke bawah icon tersebut sehingga muncul 8 kaki input data seperti ditunjukkan Gambar 5.55.

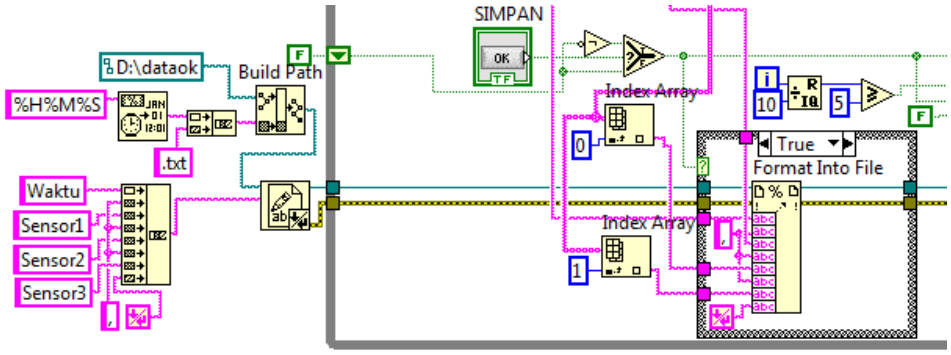


Gambar 5.54 Menambahkan kata Waktu, Sensor1, Sensor2, Sensor3 di awal isi file

9. Hubungkan ke-8 kaki input data Format Into File mengikuti tabel berikut:

No.	Kaki	Garis Data
1.	Input1	Format Date/Time String dengan format %X
2.	Input2	Tanda koma
3.	Input3	Sensor Arah Angin yang sudah diubah menjadi String
4.	Input4	Tanda koma
5.	Input5	Index Array dengan index 0, untuk data String Tombol
6.	Input6	Tanda koma
7.	Input7	Index Array dengan index 1, untuk data String Potensio
8.	Input8	Tanda Enter

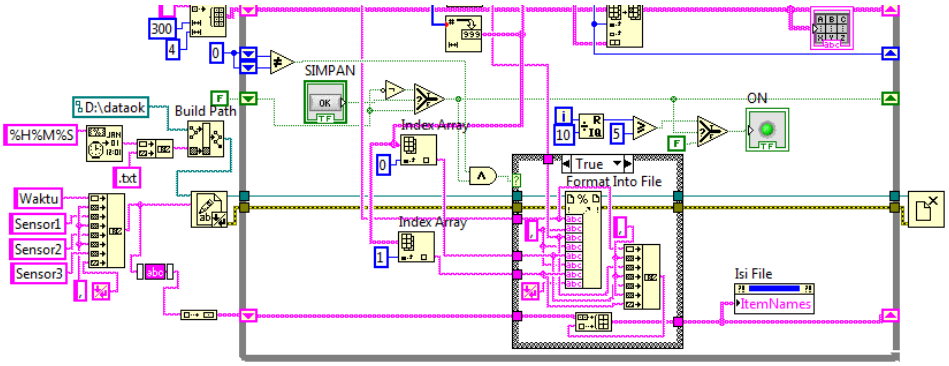
Keterangan: Index Array untuk Input5 dan Input7, mengambil input data dari Shift Register untuk Array Tombol dan Potensio yang kemudian diubah menjadi String. Apabila gambar program tidak jelas, pembaca dapat membuka Program_5_4_d.vi yang ada di CD pendukung buku ini.



Gambar 5.55 Menghubungkan data waktu, sensor1 (Arah), sensor2 (Tombol), sensor3 (Potensio) ke input Format Into File dengan disisipi koma diakhiri Enter

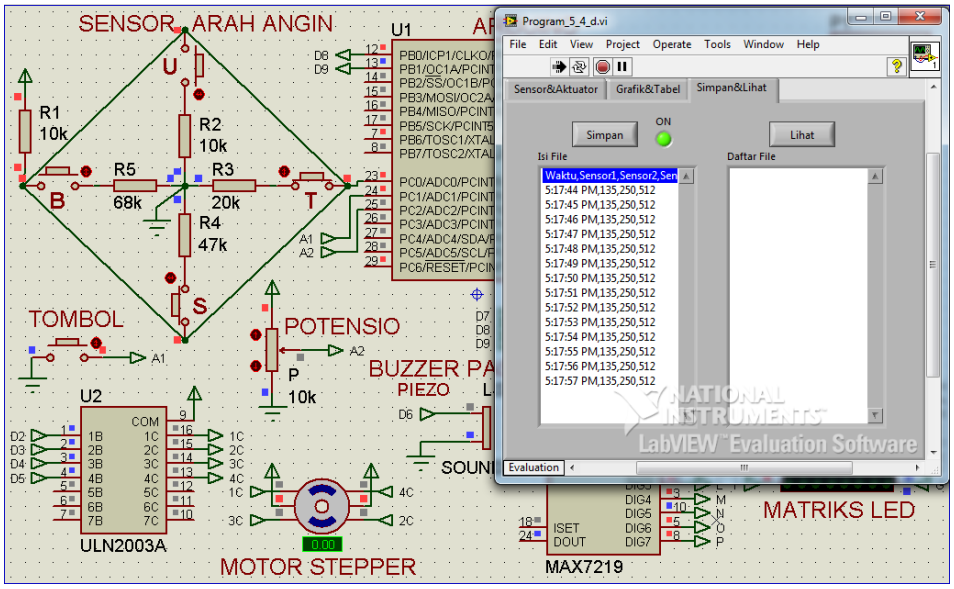
10. Berikutnya, untuk mengetahui isi data yang sedang disimpan, maka perlu menampilkan data tersebut di Listbox Isi File. Agar data dapat tampil di Listbox, dibutuhkan Shift Register untuk menyimpan data dan data harus berbentuk Array. Untuk tambahkan Shift Register, dengan nilai awal Shift Register diisi dengan output Concatenate Strings dari Waktu, Sensor1, Sensor2, Sensor3 diikuti Enter, yang kemudian disisipi dengan Trim Whitespace (untuk menghilangkan tanda Enter) dan Build Array (untuk mengubah data String menjadi Array String).
11. Saat penyimpanan dijalankan, data Array String di Shift Register tersebut digabungkan dengan data Array String yang baru dengan Build Array. Hasil penggabungan Array tersebut kemudian ditampilkan di Listbox dengan cara menghubungkan garis data Shift Register ke ItemNames Isi File.
12. Untuk memunculkan ItemNames Isi File, klik kanan icon Listbox Isi File, pilih Create, pilih Property Node, pilih ItemNames, seperti Gambar 5.56.
13. Dari Gambar 5.56 tersebut, perhatikan, bahwa syarat Case Structure untuk penyimpanan data dan penggabungan Array akan bernilai True apabila ada 2 syarat yang dipenuhi (lihat input fungsi AND), yaitu apabila ada perubahan pada angka grafik dan apabila proses SIMPAN sedang bekerja (LED ON sedang berkedip). Mengapa syarat perubahan angka grafik diperlukan? Karena bila tidak dimasukkan, maka penyimpanan data bisa lebih dari satu kali dalam satu detik. Dengan mensyaratkan harus ada

perubahan angka grafik akan memaksa penyimpanan hanya dilakukan sekali saja dalam 1 detik (angka grafik hanya berubah bila sudah 1 detik).



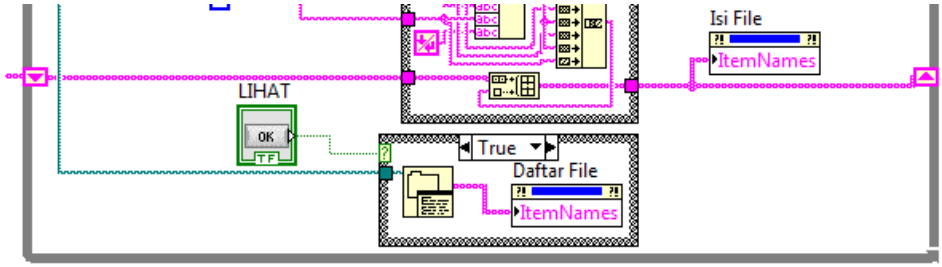
Gambar 5.56 Menggabungkan data Shift Register dengan Array String yang baru dan menampilkan data yang disimpan di Shift Register pada Listbox Isi File

14. Jalankan simulasi Proteus dan program LabVIEW di atas. Berikut ini hasil simulasi dan program LabVIEW ketika penyimpanan dijalankan (ditandai dengan LED ON berkedip). Tampak bahwa data antara satu baris dengan baris berikutnya memiliki selisih waktu 1 detik.



Gambar 5.57 Hasil simulasi program menampilkan data di Listbox Isi File

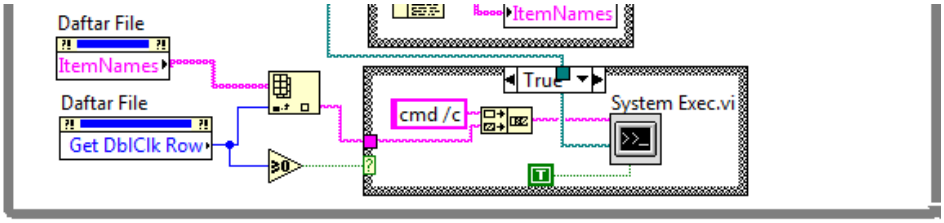
15. Berikutnya adalah menampilkan nama-nama file hasil penyimpanan di Listbox Daftar File dan membuka isi file tersebut apabila namanya diklik 2 kali. Untuk itu tambahkan icon List Folder, yang diambil dari Palet Functions, Programming, File I/O, Advanced File Functions.
16. Hubungkan input List Folder dengan path D:\dataok, dan output List Folder dengan ItemNames Listbox Daftar File. Untuk memunculkan ItemNames Listbox Daftar File, klik kanan pada icon Listbox Daftar File, pilih Create, pilih Property Node, pilih Item Names.
17. List Folder hanya akan menampilkan dan mengupdate daftar nama file yang ada di folder D:\dataok apabila tombol LIHAT ditekan. Untuk itu tambahkan Case Structure, tempatkan List Folder dan ItemNames Listbox Daftar File di Case True, dan hubungkan terminal selector Case Structure tersebut ke tombol LIHAT, seperti ditunjukkan Gambar 5.58 berikut ini.



Gambar 5.58 List Folder dan ItemNames DaftarFile ditempatkan di Case True

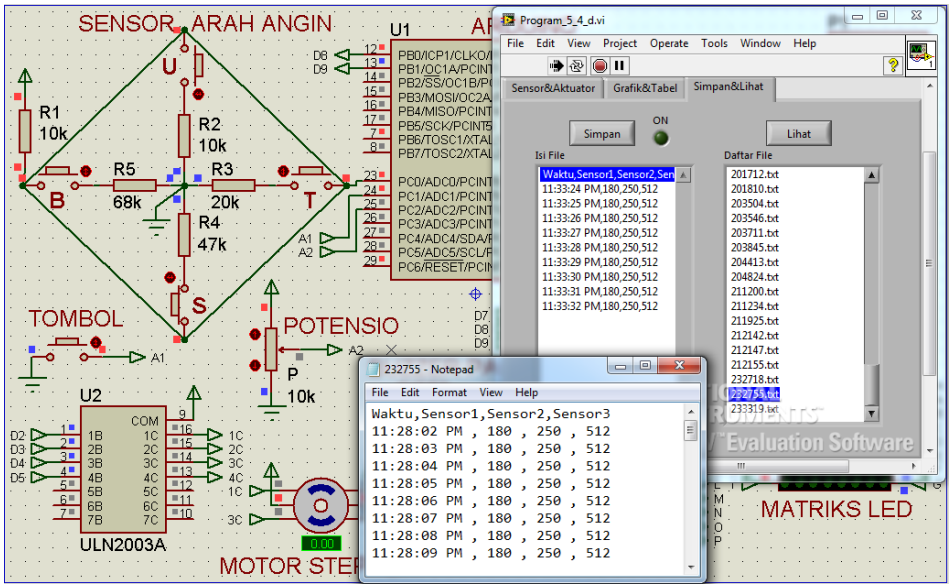
18. Berikutnya adalah membuka file ketika namanya diklik 2 kali. Untuk itu tambahkan icon System Exec, yang diambil dari Palet Functions, Connectivity, Libraries & Executables.
19. Beri input True untuk kaki input run minimized pada icon Syste Exec.
20. Beri input path D:\OK untuk kaki input working directory.
21. Beri input command line dengan gabungan kata cmd /c dan nama file.
22. Untuk mengetahui nama file, gunakan icon Index Array dengan kaki input Array dihubungkan dengan ItemNames Daftar File, dan kaki index dihubungkan dengan icon GetDblClkRow Daftar File. Untuk memunculkan icon GetDblClkRow, klik kanan icon Listbox Daftar File, pilih Create, pilih Invoke Node, pilih Get Double-Clicked Row.

23. Terakhir, tempatkan System Exec di dalam Case Structure True, dengan syarat nilai GetDblClkRow lebih besar sama dengan 0 (nilai GetDblClkRow akan bernilai sesuai nomor index atau nomor urut nama file pada Listbox Daftar File, yang akan muncul apabila diklik 2 kali. Jika tidak diklik 2 kali, nilai icon GetDblClkRow ini kurang dari 0).



Gambar 5.59 System Exec akan bekerja membuka file ketika GetDblClkRow bernilai lebih besar sama dengan 0, atau ketika nama file tersebut diklik 2 kali

Catatan: Apabila gambar program tidak terlihat jelas, pembaca dapat membuka Program_5_4_d.vi yang ada di CD pendukung buku ini.



Gambar 5.60 Sebuah file terbuka ketika namanya di Listbox Daftar File diklik 2 kali

24. Gambar 5.60 di atas menunjukkan hasil simulasi Proteus dan program LabVIEW Program_5_4_d.vi. Tekan tombol LIHAT, seharusnya daftar nama file muncul di Listbox Daftar File. Kemudian klik 2 kali pada sebuah nama file, seharusnya isi dari file tersebut terbuka, seperti Gambar 5.60.

5.3.4 Pengaturan Pengendalian Aktuator

Berikut langkah-langkahnya:

1. Tambahkan Page Pengaturan di samping Page Simpan&Lihat.
2. Ada 3 pilihan pengaturan yang akan dibuat, yang ditandai dengan angka 1, 2 dan 3, dengan keterangan seperti ditunjukkan dalam tabel berikut:

Tabel 5.1 Pilihan Pengaturan

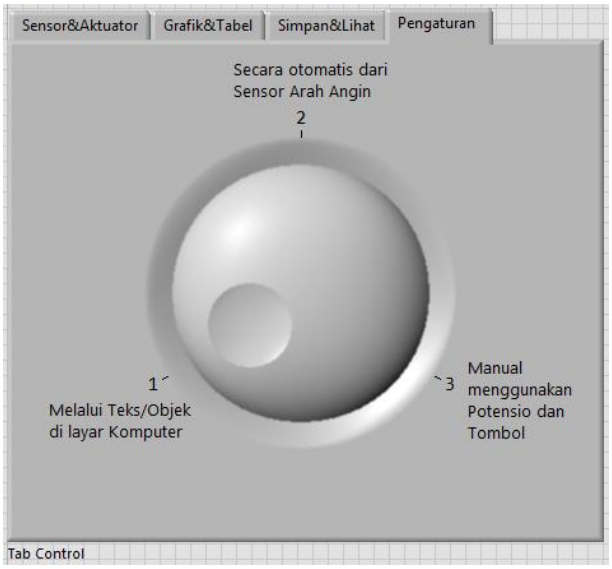
No.	Pilihan	Keterangan Pengaturan
1	Komputer dengan Teks atau Objek.	Ketiga aktuator dapat dikendalikan dari komputer, baik melalui Teks maupun 3 Objek Control (Dial, Slider1 dan Slider2).
2	Otomatis dari Sensor Arah Angin.	Ketiga aktuator dikendalikan secara otomatis untuk mengikuti posisi Sensor Arah Angin.
3	Manual dari Potensio dan Tombol.	Ketiga aktuator dikendalikan secara manual menggunakan Potensio. Penyimpanan data juga dapat dikendalikan dengan Tombol (Sensor2).

Keterangan: Berikut rincian dari Sensor, Indicator, Control dan Aktuator.

3 Sensor	Sensor Arah Angin, Tombol, Potensio
3 Indicator	Gauge Arah Angin, LED Tombol, Tank Potensio
3 Control	Dial Poros Motor, Slider1 Nada, Slider2 Pola Matriks
3 Aktuator	Motor Stepper, Buzzer, Matriks LED

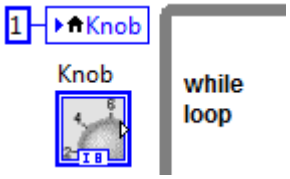
3. Untuk menyediakan ketiga pilihan pengaturan di atas, tambahkan sebuah Knob dengan nilai 1, 2, 3, seperti ditunjukkan pada Gambar 5.61.
4. Beri keterangan dengan menuliskan teks di setiap pilihan. Untuk bisa menuliskan teks, gunakan Palet Tool Text, yang dimunculkan dengan cara

menekan tombol Shift dan klik kanan bersamaan. Pada Palet Tool yang muncul, klik pada tombol berhuruf A. Setelah pointer berubah bentuknya, ketikkan teks yang diingini. Setelah selesai, jangan lupa untuk mengembalikan pointer ke bentuk otomatisnya, dengan cara meng-klik indikator LED kotak pada Palet Tool sehingga LED tersebut menyala.



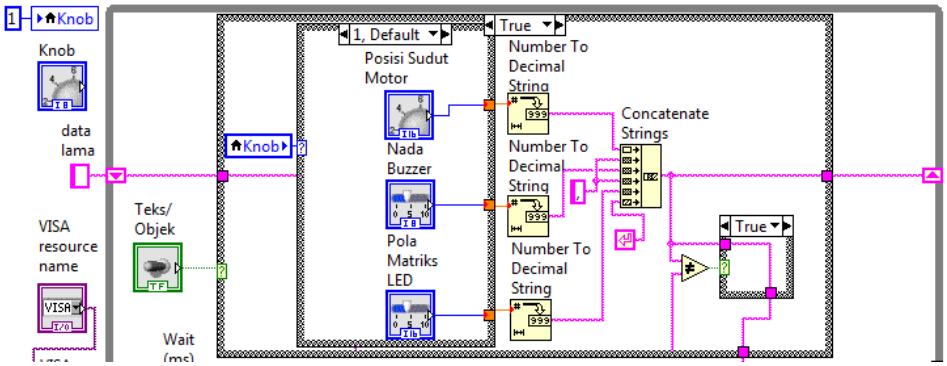
Gambar 5.61 Page Pengaturan dengan Knob 3 pilihan pengaturan: 1, 2 atau 3

5. Buka Block Diagram. Cari icon Knob. Klik kanan icon Knob, pilih Create, pilih Local Variabel, maka akan muncul kotak icon bertuliskan Knob. Nilai local variable Knob selalu mengikuti nilai Knob.
6. Agar setiap kali program dijalankan pilihan Knob selalu ke angka 1, maka beri nilai 1 pada input local variabel Knob, dan tempatkan local variabel Knob tersebut di luar While Loop.



Gambar 5.62 Klik kanan Knob, pilih Create Local Variable, tempatkan di luar kotak While Loop, dan beri input 1 agar secara default memilih pengaturan no. 1

7. Buat lagi local variable Knob yang lain.Kemudian ubah dari kondisi untuk ditulis menjadi untuk dibaca, dengan klik kanan, pilih Change to Read.
8. Tempatkan local variable Knob untuk dibaca itu di dalam Case Structure Toggle Teks/Objek, di Case True.
9. Agar pilihan Knob bisa mengatur pengendalian aktuator, buat icon Posisi Sudut Motor, Nada Buzzer, Pola Matriks LED dimasukkan ke dalam Case Structure, dengan local variabel Knob sebagai pemilihnya, yaitu dihubungkan dengan terminal selector (?), seperti Gambar 5.63 berikut.

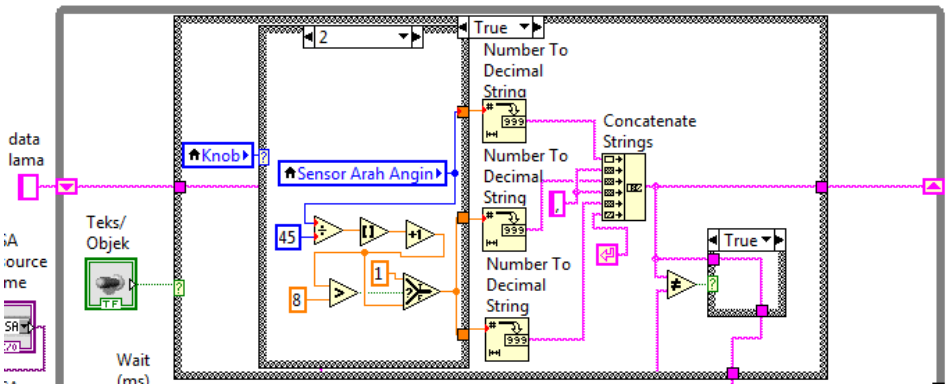


Gambar 5.63 Local variabel Knob terhubung ke terminal selector Case Structure yang berisi Posisi Sudut Motor, Nada Buzzer dan Pola Matriks LED

10. Buat Case yang berisi Posisi Sudut Motor, Nada Buzzer dan Pola Matriks LED menjadi Default (klik kanan label dan pilih Make This The Default Case), dan buat nilainya menjadi 1.
11. Klik pada tanda panah label selector untuk berpindah ke Case yang lain. Ubah label Case tersebut menjadi 2.
12. Pada pilihan no. 2 ini, semua Aktuator akan mengikuti nilai Sensor Arah Angin. Untuk itu klik kanan icon Gauge Sensor Arah Angin, pilih Create, pilih Local Variable. Tempatkan local variabel Sensor Arah Angin di Case 2.
13. Agar dapat dibaca, klik kanan icon local variable Sensor Arah Angin, dan pilih Change to Read.
14. Karena nilai Sensor Arah Angin memiliki jangkauan nilai antara 0 – 360, sedangkan poros Motor Stepper memiliki jangkauan nilai yang sama,

maka output Sensor Arah Angin langsung bisa dihubungkan ke terminal kotak kecil untuk Motor Stepper.

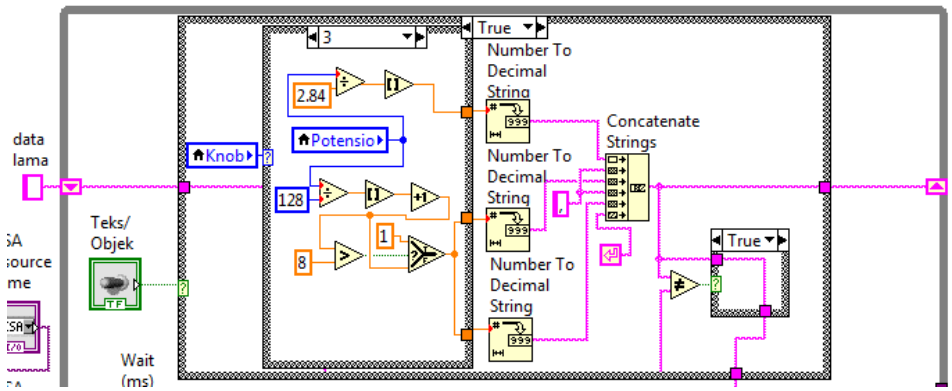
15. Berbeda dengan Motor Stepper, Buzzer dan Matriks LED hanya memiliki jangkauan nilai 1 – 8. Untuk itu, agar nilai Sensor Arah Angin dapat diberikan ke Buzzer dan Matriks LED, nilai tersebut harus dikonversi dengan cara membaginya dengan 45, kemudian hasilnya dibulatkan dengan icon Round to Nearest. Hasil pembulatan kemudian ditambah 1 untuk menghilangkan angka 0. Terakhir, agar nilai 1-8 tersebut bisa berlanjut berputar, maka ketika nilainya lebih besar dari 8, nilai tersebut diubah ke nilai 1, seperti ditunjukkan pada Gambar 5.64 berikut.



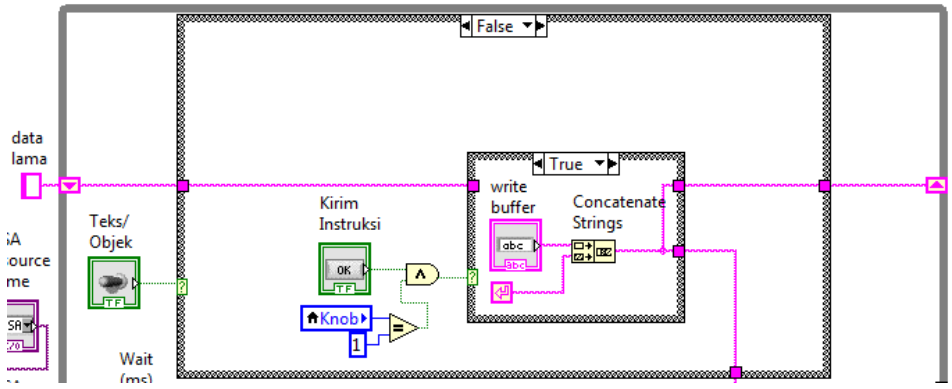
Gambar 5.64 Di Case 2, yaitu bila knob diputar ke pilihan 2, ketiga aktuator dikontrol otomatis mengikuti nilai Sensor Arah Angin

16. Klik kanan label Case 2, kemudian pilih Add Case After, seharusnya muncul Case 3. Di Case 3 ini, ketiga aktuator akan dikendalikan menggunakan nilai Potensio. Berhubung nilai Potensio dari 0 – 1023, sedangkan Motor Stepper dari 0 – 360, maka konversi nilai Sensor Arah Angin dengan membaginya dengan 2,84 dan kemudian hasilnya dibulatkan.
17. Konversi nilai Potensio untuk Buzzer dan Matriks LED dilakukan dengan membagi nilai Potensio dengan 128, kemudian dibulatkan, hasil pembulatan ditambah 1 untuk menghilangkan 0, dan ketika nilainya lebih besar dari 8, ubah nilai tersebut menjadi 1, seperti Gambar 5.65.

18. Kemudian untuk Case False pada Case Structure Toggle Teks/Objek, tambahkan syarat Kirim Instruksi, yaitu selain tombol Kirim Instruksi harus ditekan, nilai Knob harus 1, seperti ditunjukkan Gambar 5.66.



Gambar 5.65 Di Case 3, yaitu ketika Knob diputar ke pilihan 3, ketiga aktuator harus mengikuti nilai Potensio, yang mana nilai Potensio tersebut harus dikonversi menyesuaikan nilai jangkauan masing-masing aktuator

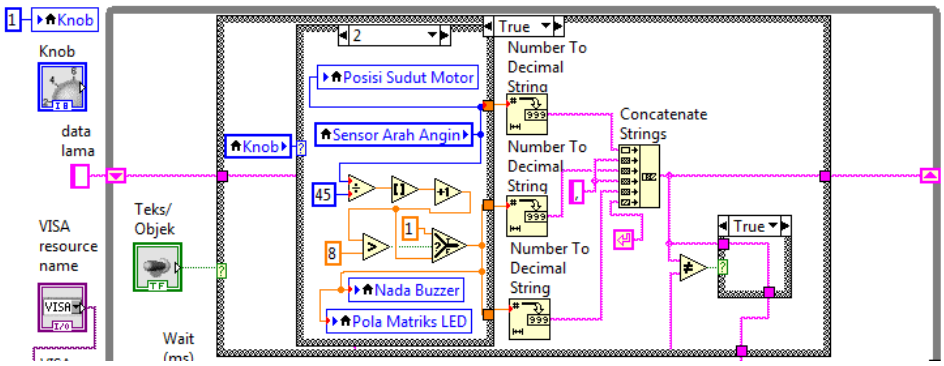


Gambar 5.66 Di Case False Case Structure Toggle Teks/Objek, syarat Kirim Instruksi melalui teks tidak hanya tombol Kirim Instruski harus ditekan, tetapi ditambah dengan syarat nilai Knob harus 1 (lihat input fungsi AND yang terhubung ke terminal selector)

19. Ketika program dijalankan dan pilihan Knob di 2, dengan program di atas, ketiga aktuator akan dapat mengikuti nilai Sensor Arah Angin. Hanya saja, kondisi aktuator tersebut belum diikuti oleh objek Controlnya masing-masing. Seharusnya ketika posisi Motor Stepper berubah, jarum di Dial Posisi Sudut Motor juga ikut berubah, begitu pula Slider1 Nada Buzzer dan

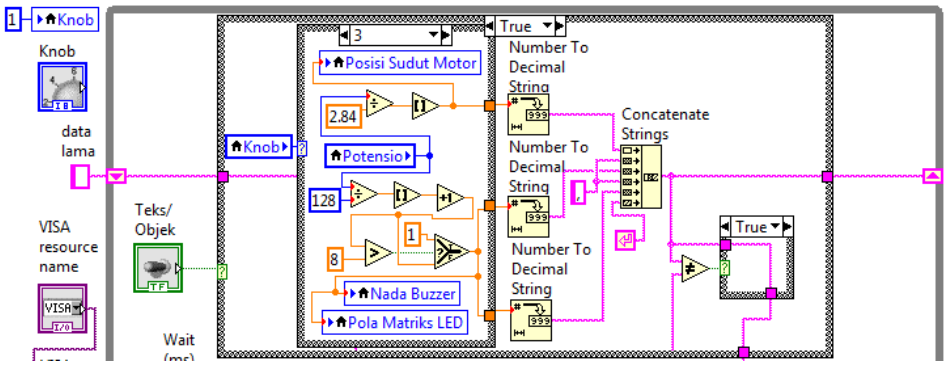
Slider2 Pola Matriks LED. Untuk membuat perubahan juga terjadi di objek Control, tambahkan Local Variable masing-masing icon Control tersebut.

20. Hubungkan local variable Dial Posisi Sudut Motor dengan local variable Sensor Arah Angin. Kemudian hubungkan local variable Slider1 Nada Buzzer dan local variable Slider2 Pola Matriks LED dengan hasil konversi nilai Sensor Arah Angin ke Buzzer dan Matriks LED, seperti Gambar 5.67.

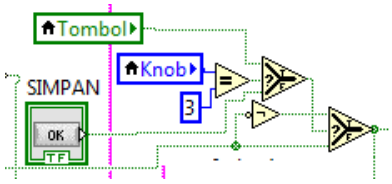


Gambar 5.67 Di Case 2, agar tidak hanya ketiga aktuatornya saja yang mengikuti Sensor Arah Angin, tetapi juga masing-masing objek Controlnya, tambahkan local variable objek Controlnya, dan hubungkan ke Sensor Arah Angin dan konversinya

21. Di Case 3, agar tidak hanya ketiga aktuatornya saja yang mengikuti nilai Potensio, tetapi objek Control dari masing-masing aktuator tersebut juga mengikuti nilai Potensio, tambahkan Local Variable Dial Posisi Sudut Motor, Local Variable Slider1 Nada Buzzer dan Local Variable Slider2 Pola Matriks LED. Tempatkan di Case 3, dan hubungkan dengan konversi dari nilai Local Variable Potensio, seperti ditunjukkan Gambar 5.68.
22. Terakhir, karena di Case 3, selain Potensio (Sensor3), ada Tombol (Sensor2) yang juga digunakan untuk menjalankan dan menghentikan penyimpanan data secara manual, maka tambahkan Local Variabel LED Tombol, dan tempatkan di dekat tombol Simpan. Tambahkan icon Select, dan hubungkan kaki input t dengan Local Variable LED Tombol, dan kaki input s dengan syarat apakah nilai Knob sama dengan 3, dan kaki input f dengan garis data tombol Simpan, seperti ditunjukkan Gambar 5.69.

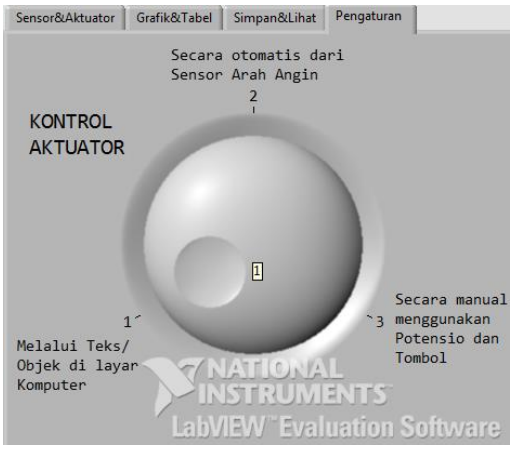


Gambar 5.68 Di Case 3, agar tidak hanya aktuatornya saja yang mengikuti nilai Potensio, tetapi juga objek Controlnya, hubungkan local variable masing-masing objek Control dengan garis data yang dikirim ke masing-masing aktuator



Gambar 5.69 Agar Tombol (Sensor2) dapat digunakan untuk menjalankan atau menghentikan penyimpanan data, tambahkan Select, hubungkan input s dengan Knob = 3, tombol Simpan dengan input f dan local variable LED Tombol dengan input t

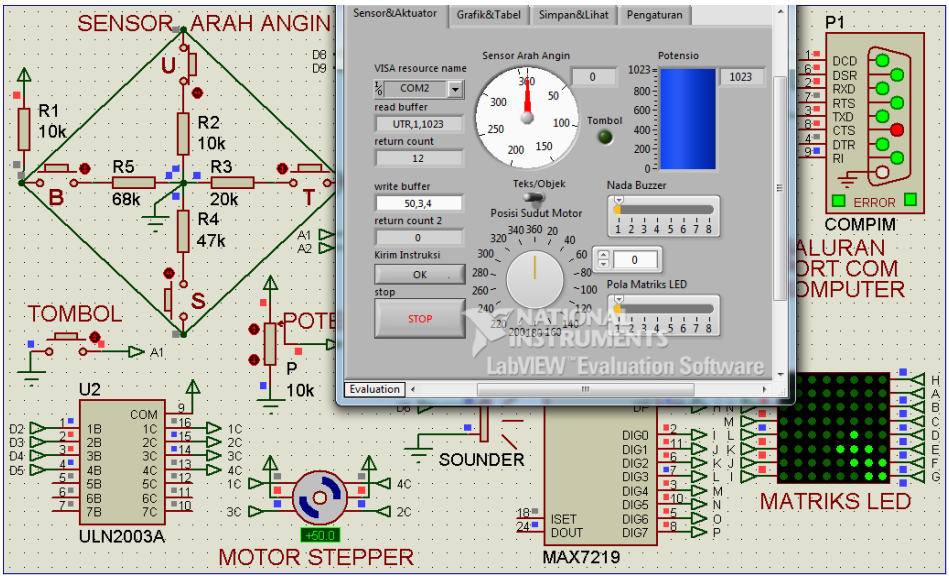
23. Jalankan simulasi Proteus dan program LabVIEW. Secara default, atau saat awal, Knob di Page Pengaturan akan menunjuk pada pilihan 1.



Gambar 5.70 Secara default Knob pada pilihan 1

Catatan: Apabila gambar program untuk Page Pengaturan tidak terlihat jelas, pembaca dapat membuka Program_5_4_e.vi yang ada di CD pendukung buku ini.

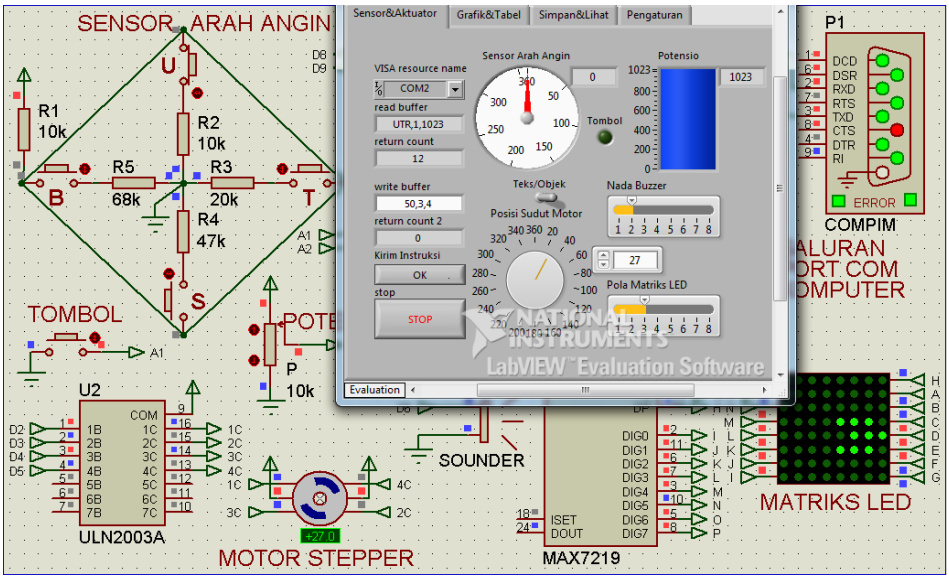
24. Dengan Knob pada pilihan 1, di Page Sensor&Aktuator, geser tuas Toggle Teks/Objek ke kiri untuk memilih teks. Ketikkan 50,3,4 diikuti penekanan tombol Kirim Instruksi. Maka akan terlihat Motor Stepper di Proteus berputar hingga 50 derajat, diiringi bunyi nada ketiga Buzzer dan Matriks LED pada pola keempat seperti ditunjukkan Gambar 5.71.



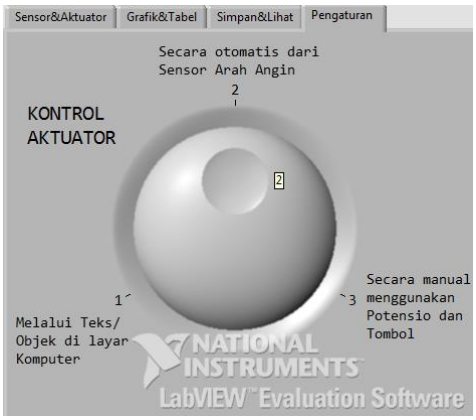
Gambar 5.71 Saat Knob=1, ketiga aktuator dapat dikontrol melalui teks (50,3,4)

25. Setelah pengendalian aktuator melalui teks, sementara Knob masih di 1, berikutnya pengendalian aktuator melalui objek. Geser tuas Toggle Teks/Objek ke kanan. Atur Dial Posisi Sudut Motor, Slider Nada Buzzer dan Slider Pola Matriks LED. Gambar 5.72 menunjukkan poros Motor Stepper berputar sesuai posisi Dial, yaitu di sudut 27, begitu pula Buzzer dan Matriks LED berubah sesuai pengaturan objek Controlnya.
26. Berikutnya, ubah Knob ke pilihan 2 seperti Gambar 5.73.
27. Tekan tombol U, T, S atau B Sensor Arah Angin. Perhatikan ketiga aktuator akan mengikuti nilai Sensor Arah Angin. Gambar 5.74 menunjukkan posisi

Motor Stepper di sudut 225, dengan Buzzer di nada ke-6 dan Matriks LED di pola ke-6, ketika tombol S dan B Sensor Arah Angin ditekan.



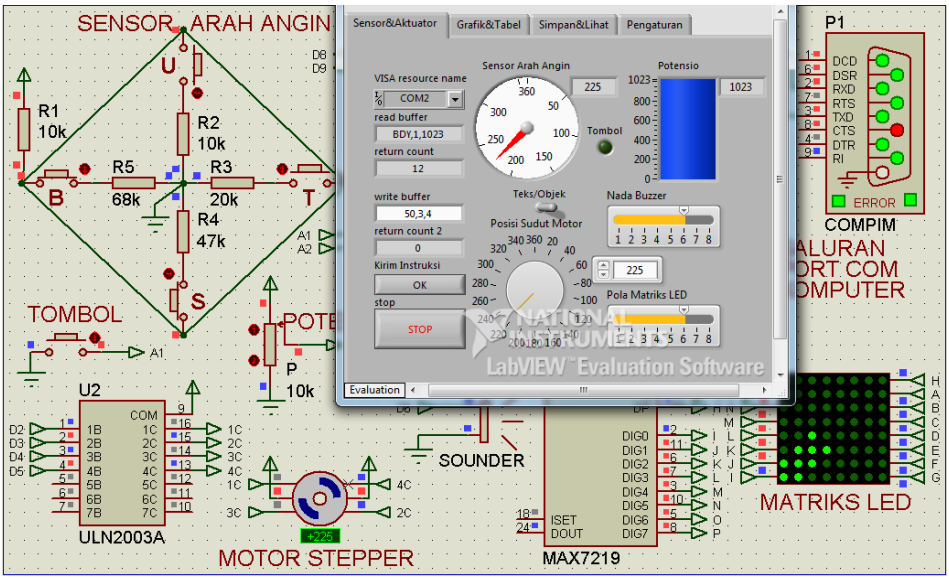
Gambar 5.72 Saat Knob=1, ketiga aktuator dapat dikontrol melalui objek Control



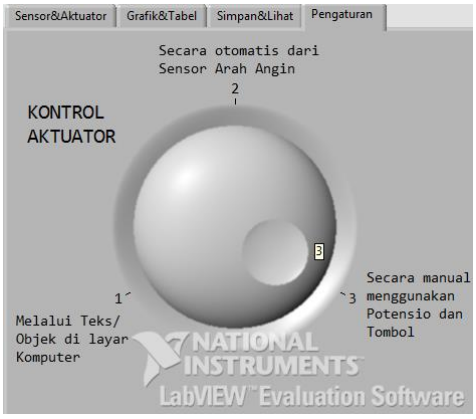
Gambar 5.73 Knob diputar ke pilihan 2

28. Berikutnya, ubah Knob ke pilihan 3 seperti Gambar 5.75.
29. Geser Potensio di Proteus ke atas dan ke bawah. Perhatikan ketiga aktuator akan mengikuti nilai Potensio. Gambar 5.76 menunjukkan posisi

Motor Stepper di sudut 325, dengan bunyi Buzzer di nada ke-8 dan Matriks LED di pola ke-8, ketika nilai Potensio sebesar 922.



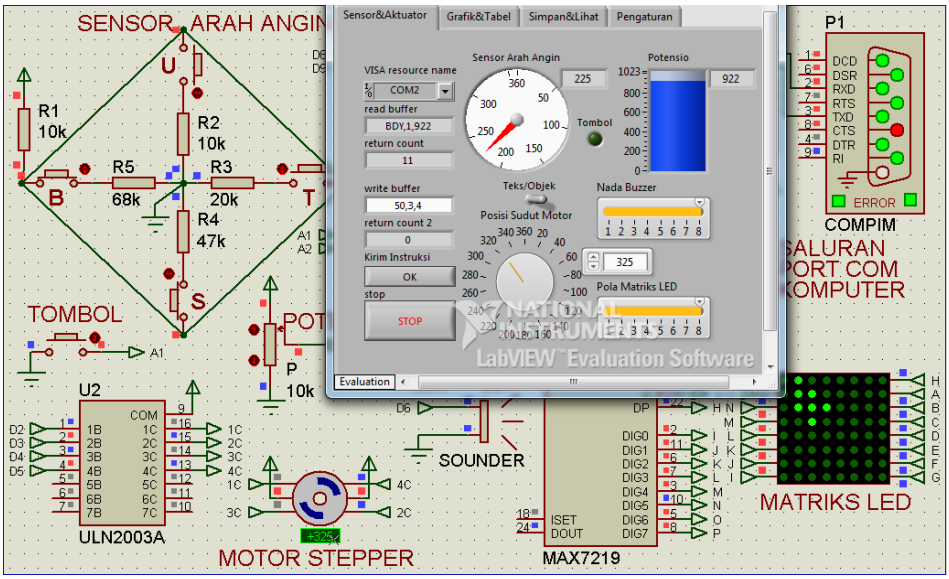
Gambar 5.74 Saat Knob=2, ketiga aktuator mengikuti Sensor Arah Angin



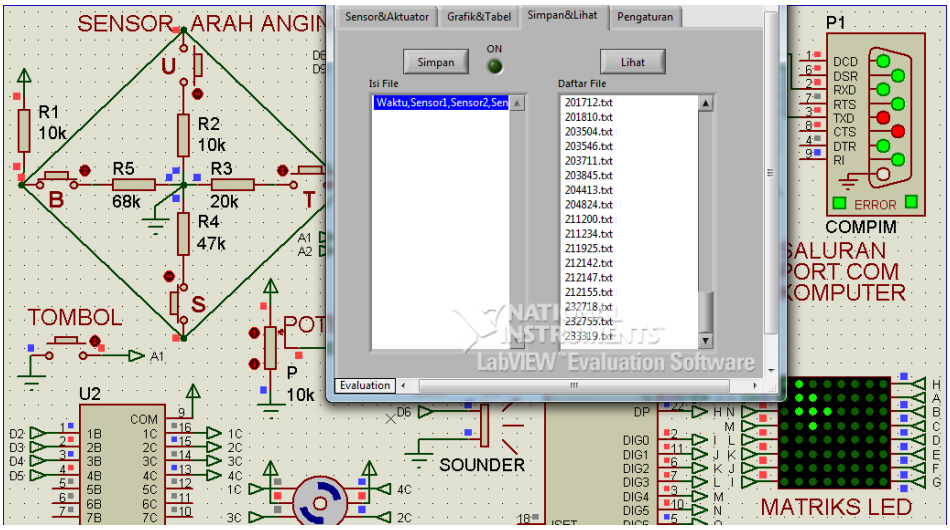
Gambar 5.75 Knob diputar ke pilihan 3

30. Saat Knob = 3, tombol penyimpanan data bukanlah tombol Simpan. tetapi berada di luar, yaitu Tombol (Sensor2) di Proteus. Tampak pada Gambar 5.77 sebelum Tombol (Sensor2) ditekan, Listbox Isi File hanya berisi judul

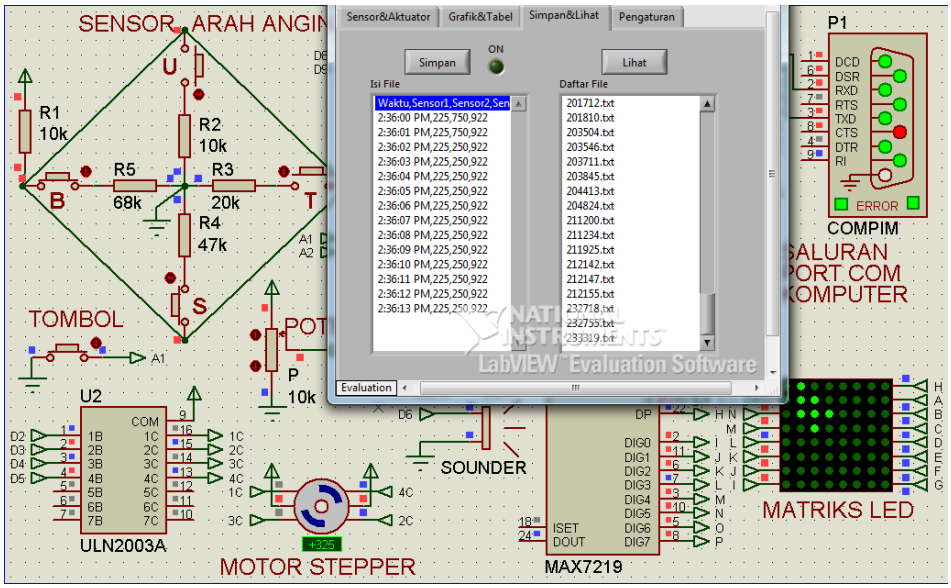
data. Ketika Tombol (Sensor2) ditekan, Listbox isi File terisi dengan data Waktu, Sensor1, Sensor2, Sensor3, seperti ditunjukkan Gambar 5.78.



Gambar 5.76 Saat Knob=3, ketiga aktuator mengikuti nilai Potensio. Ketika nilai Potensio=922, Motor Stepper di 325, Nada Buzzer di 8 dan Pola Matriks LED di 8



Gambar 5.77 Saat Knob=3, penyimpanan data dikontrol dari Tombol (Sensor 2) di Proteus, sedangkan tombol Simpan di LabVIEW tidak difungsikan. Penekanan sekali akan menjalankan penyimpanan, penekanan berikutnya menghentikan



Gambar 5.78 Tampak ketika Tombol (Sensor2) di Proteus ditekan, penyimpanan data djalankan, penekanan berikutnya membuat penympanan data dihentikan

5.4 Soal Latihan

1. Aplikasi Interface Sensor yang dibuat di Bab 5 ini masih belum lengkap, yaitu belum ada pembuatan Grafik dari File, seperti di Sub Bab 4.4. Untuk itu tambahkan Page baru dengan nama File&Grafik, dan buat Listbox Daftar File yang ada di Page Simpan&Lihat ditampilkan kembali di Page baru tersebut. Diinginkan setiap kali salah satu file dari Listbox Daftar File di-klik 2 kali, Grafik (2D Compass dan Waveform Graph) di sampingnya akan menampilkan grafik dari isi File yang dipilih tersebut. Karena setiap File berisi data 3 Sensor, maka data Sensor Arah Angin akan tampil di 2D Compass, sedangkan data Tombol dan Potensio di Waveform Graph.
2. Diinginkan ketika Knob Pengaturan diputar ke pilihan 2, penyimpanan data langsung dijalankan tanpa perlu menekan Tombol Simpan. Penyimpanan data akan berhenti ketika pilihan di Knob dipindahkan ke pilihan yang lain. Penyimpanan data yang langsung seperti ini hanya berlaku untuk pilihan 2

saja. Untuk pilihan 1 harus menekan Tombol Simpan, dan pilihan 3 harus menekan Tombol Sensor2.

3. Tambahkan pada Knob di Page Pengaturan, sebuah pilihan keempat, yang dinamai Alarm. Ketika Knob diputar ke pilihan 4 ini, diinginkan hal-hal seperti berikut ini:
   - Sensor Arah Angin akan mengendalikan Motor Stepper, Tombol akan mengendalikan Buzzer, dan Potensio akan mengendalikan Matriks LED, dengan rincian seperti dalam tabel berikut:

Sensor Arah Angin	Motor Stepper
Utara	0
Timur Laut	45
Timur	90
Tenggara	135
Selatan	180
Barat Daya	225
Barat	270
Barat Laut	315

Tombol	Buzzer
Ditekan	Bunyi
Dilepas	Diam

Nilai Potensio	Matriks LED
0-127	Pola ke-1
128-255	Pola ke-2
256-383	Pola ke-3
384-511	Pola ke-4
512-639	Pola ke-5
640-767	Pola ke-6
768-895	Pola ke-7
896-1023	Pola ke-8

   - Selain karena Tombol (Sensor2) ditekan, Buzzer juga akan berbunyi apabila posisi Sensor Arah Angin sama dengan Nilai Potensio. Jadi dalam aplikasi ini, Potensio dianggap sebagai setpoint untuk alarm. Sebagai contoh, diinginkan bahwa setiap kali Arah Angin ke Selatan, maka alarm berbunyi. Untuk itu, atur nilai ADC Potensio antara 512 – 639, hingga bisa menampilkan Matriks LED Pola ke-5. Ketika Sensor Arah Angin mendeteksi Arah Angin ke Selatan, disimulasikan dengan menekan tombol S di Proteus, maka seharusnya Buzzer akan berbunyi. Buzzer berhenti berbunyi apabila Sensor Arah Angin sudah tidak mendeteksi Selatan, yang disimulasikan di Proteus dengan menekan tombol lain, missal tombol B di Sensor Arah Angin.

BAB 6
HARDWARE SENSOR DAN AKTUATOR

Target Materi:

- Membuat Sensor Arah Angin secara riil.
- Menyusun rangkaian untuk membuat Interface Sensor dan Aktuator secara riil.
- Memprogram hardware Arduino untuk membuat Interface Sensor dan Aktuator secara riil.
- Melakukan troubleshooting dan perbaikan dalam pembuatan Interface Sensor dan Aktuator secara riil.

Persoalan:

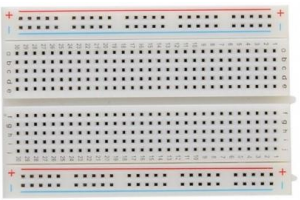
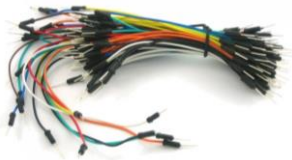
Buatlah hardware Sensor Arah Angin, dan implementasikan Interface Sensor dan Aktuator secara riil dengan menambahkan beberapa Sensor dan Aktuator.

Uraian Materi:

6.1 Komponen dan Peralatan yang Diperlukan

Berikut ini komponen dan peralatan yang diperlukan:

No.	Komponen/Peralatan	Jumlah	Gambar
1.	Arduino Nano	1	

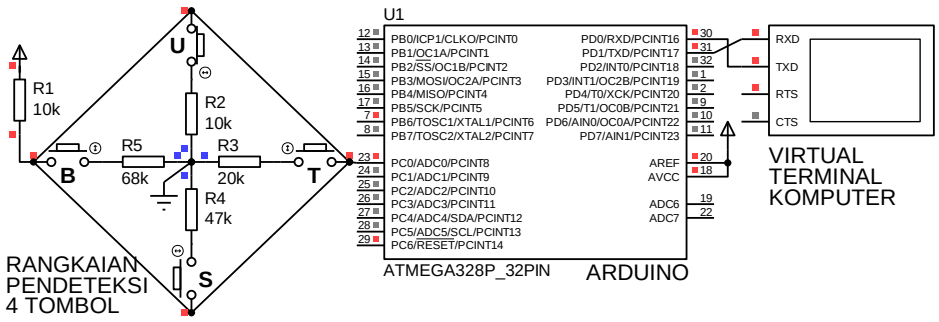
No.	Komponen/Peralatan	Jumlah	Gambar
2.	Kabel mini USB Arduino Nano	1	
3.	Breadboard	1	
4.	Kabel Jumper	±20	
5.	Reed Switch	4	
6.	Resistor 1/4 W	5	 (10k, 10k, 20k, 47k, 68k)
7.	*Bearing* (ukuran diameter luar antara 22 – 28 mm agar bisa menutup pipa PVC ukuran 3/4")	1	
8.	Magnet	1	
9.	Mur Baut (ukuran M2 dengan panjang 5 cm 1 buah, dan panjang 2 cm 2 buah)	3	

No.	Komponen/Peralatan	Jumlah	Gambar
10.	Pipa PVC (diameter 3/4"dengan panjang 5 - 10 cm)	1	
11.	PCB Lubang ukuran jarak kaki IC	1	
12.	Header Male (40 pin)	2	
13.	Lem Bakar dan pemanasnya	1	
14.	Solder dan tenol	1	
15.	Potensio 10k ohm	1	

No.	Komponen/Peralatan	Jumlah	Gambar
16.	Tombol Tactile Mini Switch	1	
17	Motor Stepper dan Drivernya	1	
18	Buzzer (Piezoelektrik)	1	
19	Matriks LED dan IC MAX7219	1	

6.2 Pembuatan Sensor Arah Angin

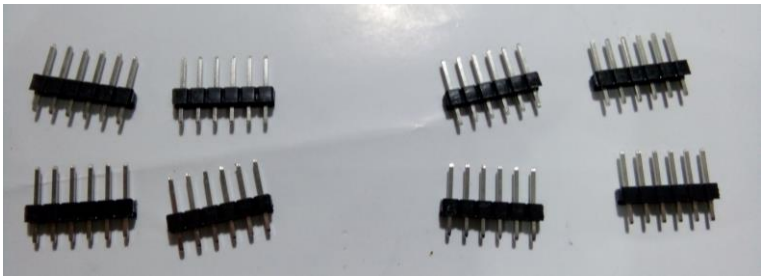
Perhatikan kembali Gambar 3.1 tentang Rangkaian Pendeteksi 4 Tombol.



Gambar 6.1 Menampilkan kembali Gambar 3.1 Rangkaian Pendeteksi 4 Tombol

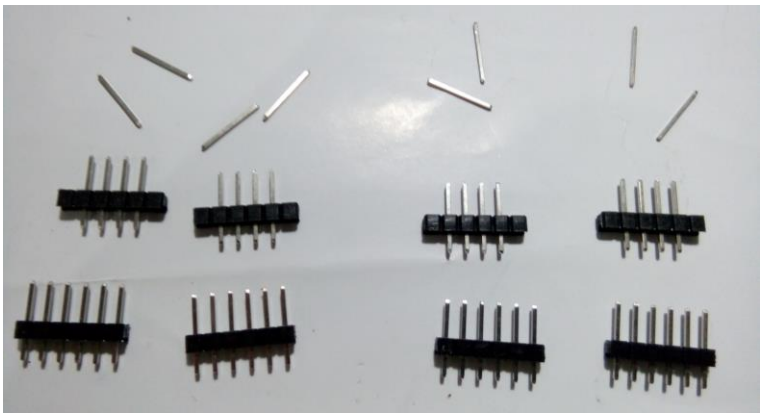
Sensor Arah Angin yang dibuat di Bab ini adalah sama seperti rangkaian pendeteksi 4 tombol. Hanya saja, sebagai ganti 4 tombol digunakan Reed Switch. Reed Switch adalah sebuah saklar yang diaktifkan oleh magnet. Ketika magnet didekatkan ke Reed Switch, maka kedua kontak Reed Switch akan terhubung, sedangkan ketika magnet dijauhkan, maka kedua kontak Reed Switch akan terputus. Dengan menempatkan 4 buah Reed Switch sebagai pengganti saklar pada rangkaian Gambar 3.1, dan menempatkan magnet di tengah-tengahnya, yang menempel pada poros yang dapat berputar bila tertiup Angin, maka Sensor Arah Angin dapat dihasilkan. Berikut langkah pembuatan Sensor Arah Angin:

1. Buat potongan header male sebanyak 8 buah, dengan masing-masing potongan memiliki 6 kaki seperti gambar berikut:



Gambar 6.2 Delapan potongan Header Male dengan 6 buah kaki

2. Pada empat potongan pertama, ambil kaki di kedua ujungnya:

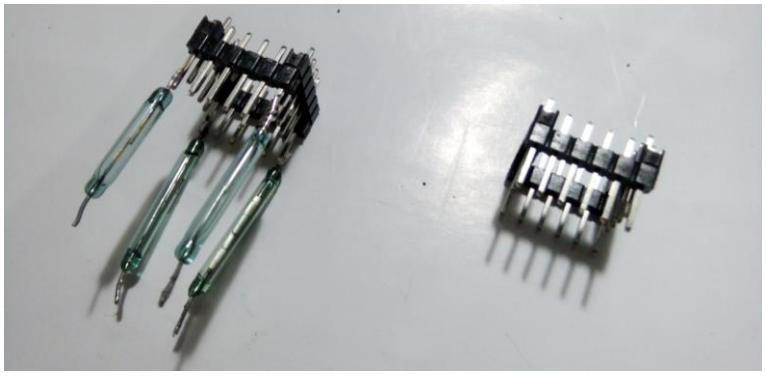


Gambar 6.3 Kedua ujung kaki diambil pada empat potongan pertama

3. Buat 2 buah kotak dengan menyatukan keempat potongan:

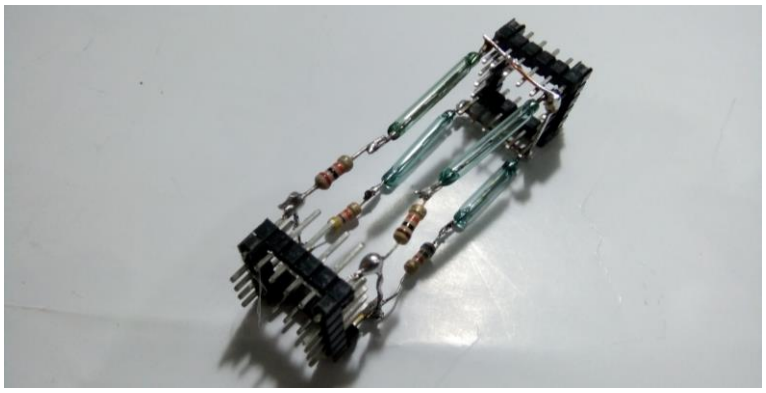
Gambar 6.4 Menyatukan keempat potongan untuk membuat kotak

4. Solder empat buah Reed Switch pada keempat ujung di kotak pertama.



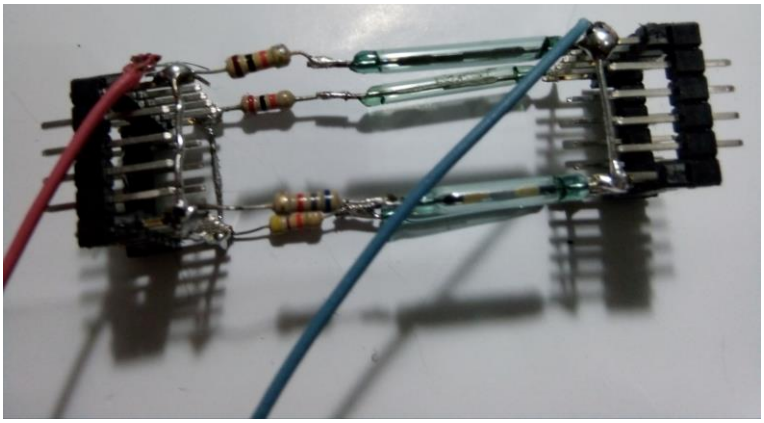
Gambar 6.5 Empat Reed Switch disolder di keempat ujung kotak pertama

5. Pada setiap ujung Reed Switch, solder 4 buah resistor dengan nilai hambatan berturut-turut 10k ohm, 20k ohm, 47k ohm dan 68k ohm.



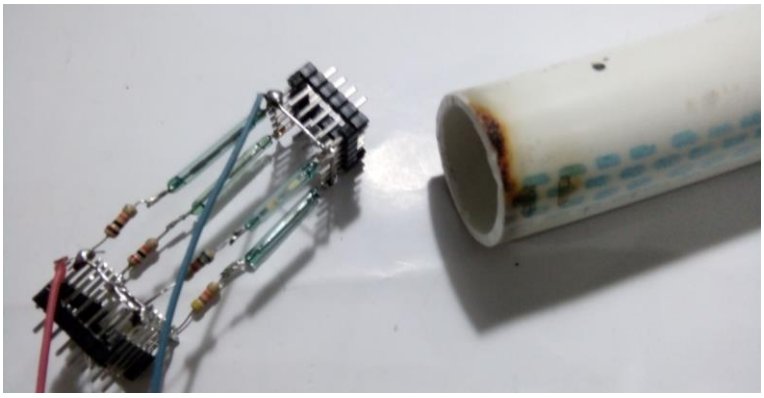
Gambar 6.6 Menambahkan resistor: 10k, 20k, 47k dan 68k pada Reed Switch

6. Tutup ujung keempat resistor dengan menyoldernya pada keempat kaki ujung kotak header male yang kedua. Kemudian buat keempat kaki di kotak pertama terhubung dengan satu kabel, dan keempat kaki di kotak kedua terhubung dengan kabel yang kedua.



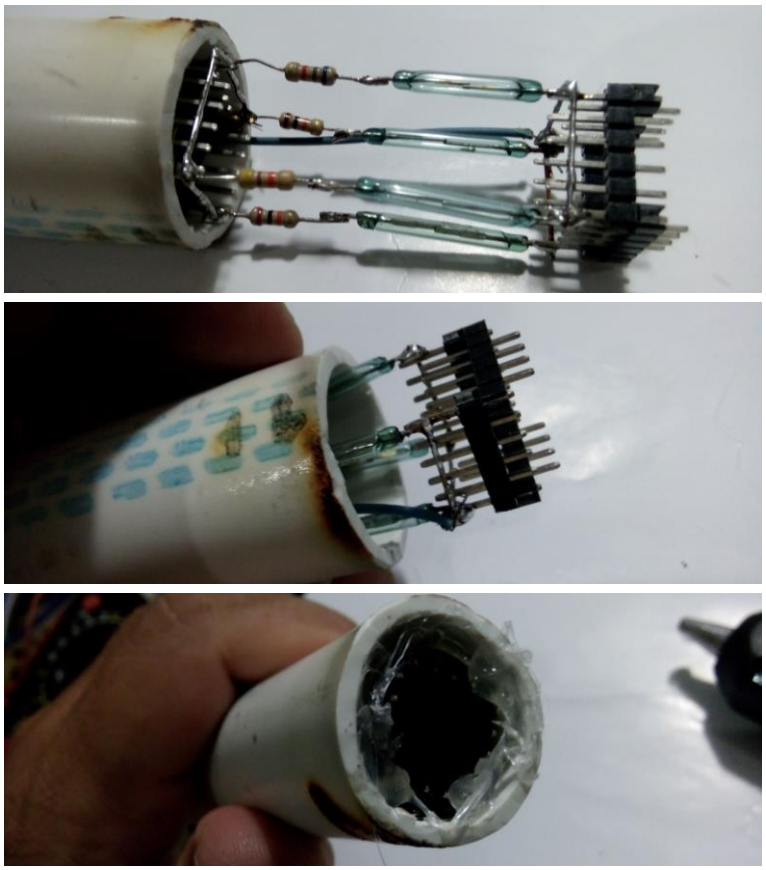
Gambar 6.7 Sesuai gambar skematik 6.1, satukan keempat kaki Reed Switch (yang disolder dengan kotak header male) dengan kabel, dan keempat kaki resistor dengan kabel yang lain

7. Berikutnya, masukkan rangkaian sensor Reed Switch di atas ke dalam sebuah pipa PVC 3/4 in seperti gambar berikut:



Gambar 6.8 Memasukkan rangkaian sensor Reed Switch ke dalam pipa PVC 3/4 in

8. Masukkan rangkaian hingga jauh ke dalam pipa. Beri lem bakar pada kotak header male agar rangkaian menempel di pipa dan tidak bergeser turun, seperti ditunjukkan pada Gambar 6.9.

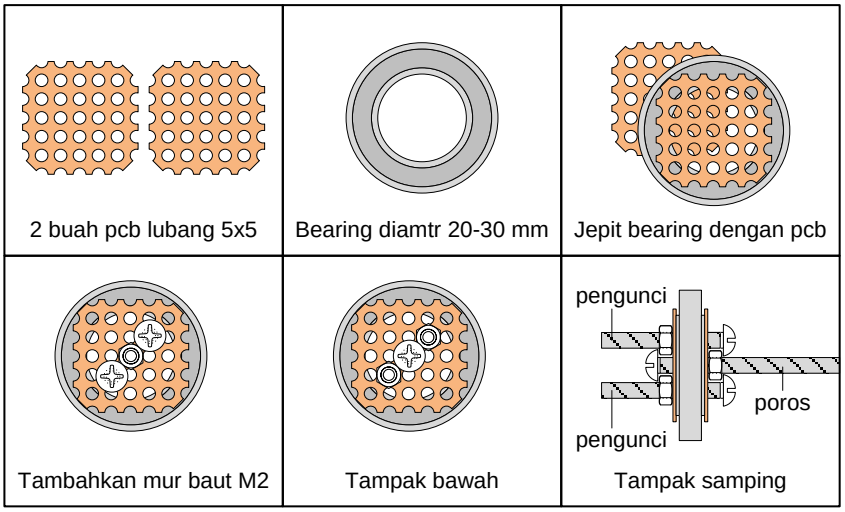


Gambar 6.9 Memasukkan rangkaian ke dalam pipa PVC 3/4 in dan memberi lem bakar

9. Tambahkan *Bearing* untuk menempatkan poros bermagnet di ujung pipa.

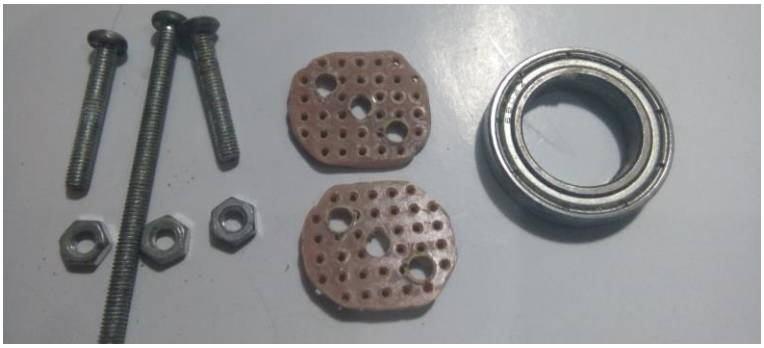
Gambar 6.10 Menambahkan Bearing pada ujung pipa untuk penempatan poros bermagnet

10. Buat sebuah Poros dari Mur Baut M2 dengan panjang 5cm, yang dipasang di tengah-tengah *Bearing*, untuk menempelkan magnet yang akan diputar di tengah rangkaian sensor Reed Switch. Agar Poros bisa tegak lurus di tengah *Bearing*, tambahkan 2 potongan PCB lubang IC dengan lubang 5x5, yang dipasang mengapit sisi dalam *Bearing*, kemudian kencangkan dengan pengunci menggunakan 2 buah Mur Baut M2 dengan panjang 2 cm, seperti ditunjukkan pada Gambar 6.11 berikut.



Gambar 6.11 Ilustrasi pembuatan poros bermagnet yang dipasang pada Bearing

11. Gambar 6.12 berikut ini menunjukkan komponen yang diperlukan untuk pembuatan poros bermagnet seperti pada ilustrasi Gambar 6.11.



Gambar 6.12 Komponen: 2 Mur Baut M2 pendek, 1 M2 panjang, 2 potong PCB dan Bearing

12. Apit Bearing dengan 2 potong PCB yang sebelumnya telah dilubangi dengan 3 lubang untuk memasukkan Mur Baut M2.

Gambar 6.13 Apit Bearing dengan 2 potongan PCB yang telah dilubangi

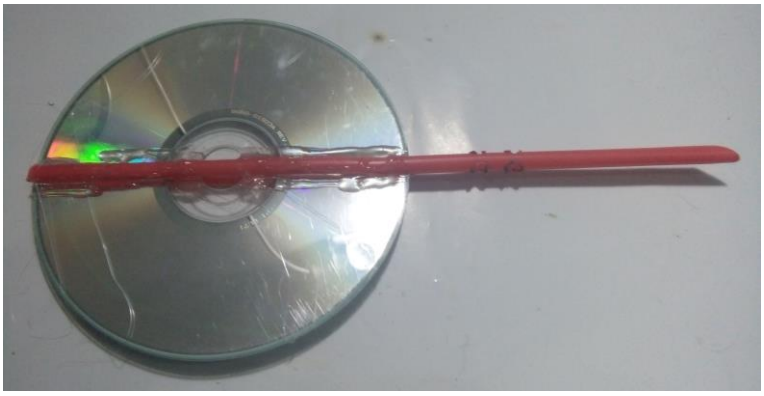
13. Tempatkan mur baut M2 yang panjang di lubang tengah PCB, dan 2 mur baut M2 yang pendek di pinggir. Agar bisa saling mengunci, maka mur baut M2 yang pendek ditempatkan terbalik, seperti Gambar 6.14.

Gambar 6.14 PCB di Bearing dikunci dengan M2 pendek di pinggir, dan M2 panjang di tengah

14. Setelah Poros dapat dipasang pada *Bearing* dengan bantuan PCB lubang, tempelkan sebuah Magnet pada Poros tersebut. Agar Magnet tidak bergeser posisinya, tempel dengan lem bakar, seperti Gambar 6.15.

Gambar 6.15 Menempelkan magnet pada poros, menambahkan lem agar tidak bergeser

15. Untuk penangkap angin, gunakan CD bekas, dan tempel sebuah batang plastik (gagang balon) pada CD seperti Gambar 6.16.

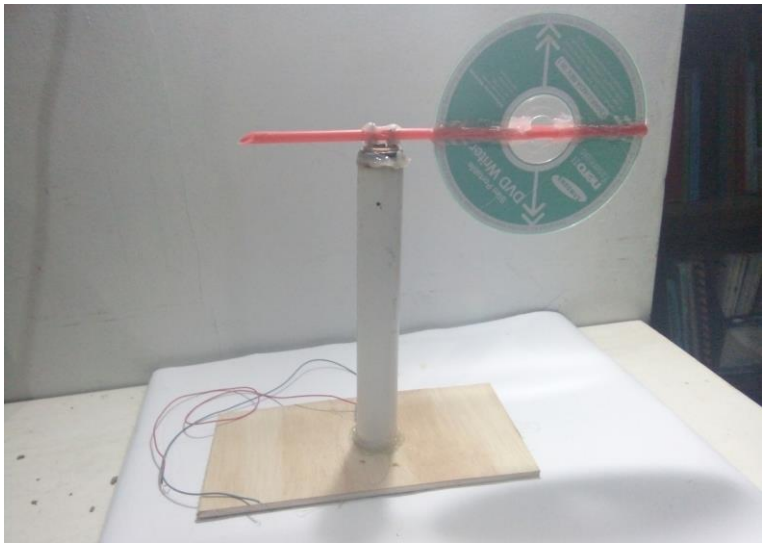


Gambar 6.16 Menempelkan batang plastik pada CD bekas untuk penangkap angin

16. Buat agar batang plastik menempel di poros, sehingga ketika CD tertiup angin, poros bermagnet ikut berputar, seperti Gambar 6.17.

Gambar 6.17 Menempelkan batang plastik pada poros bermagnet

17. Terakhir, masukkan poros bermagnet pada ujung pipa yang telah diberi rangkaian sensor Reed Switch, hingga ujung pipa tersebut tertutup Bearing. Tambahkan lem bakar agar Bearing tidak bergeser. Kemudian tegakkan pipa PVC tersebut pada sepotong papan. Tambahkan lem bakar. Berikut hasil sensor arah angin yang telah selesai dibuat.



Gambar 6.18 Hasil sensor arah angin yang telah selesai dibuat

Catatan: Mengapa penulis memilih membuat Sensor Arah Angin. Mengapa tidak menggunakan Sensor yang lain. Berikut beberapa alasan penulis:

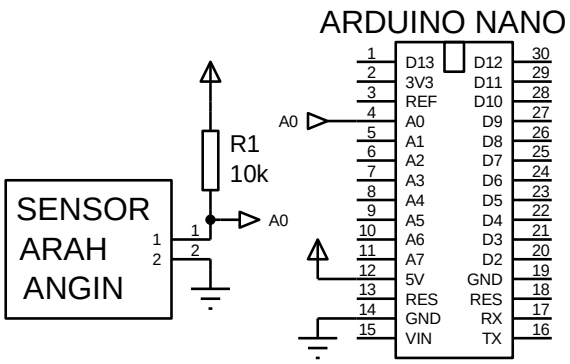
- Alasan pertama adalah karena prinsip kerja dari Sensor Arah Angin yang mudah dipahami, sehingga dapat digunakan untuk materi pembelajaran, khususnya pengenalan elektronika. Prinsip kerja rangkaian pembagi tegangan yang terdapat pada Sensor Arah Angin ini cukup menarik untuk disampaikan sebagai pengatar elektronika, seperti dituliskan di Bab 1-3.
- Alasan kedua adalah karena kebutuhan penelitian yang dikembangkan oleh penulis bersama rekan di Univ. Sanata Dharma, yaitu membuat alat penyuling air dengan energi matahari. Sensor Arah Angin diperlukan di sini untuk mendapatkan sumber angin, yang akan membuat proses pengembunan uap air pada alat menjadi optimal.

- Alasan ketiga adalah pertimbangan biaya, di mana harga Sensor Arah Angin di pasaran masih cukup mahal (lebih dari 500 ribu rupiah, karena import dan ongkos kirim). Sehingga dengan membuat sendiri Sensor Arah Angin, akan banyak menghemat biaya (kurang dari 100 ribu rupiah).

6.3 Pembacaan Sensor Arah Angin, Tombol dan Potensio

Setelah hardware Sensor Arah Angin seperti Gambar 6.18 selesai dibuat, berikutnya adalah membaca Sensor tersebut. Berikut langkah-langkahnya:

1. Rangkai Sensor Arah Angin, Arduino Nano dan Resistor 10k ohm seperti gambar skematik Gambar 6.19 berikut:



Gambar 6.19 Rangkaian Sensor arah angin, Arduino Nano, Resistor 10k ohm

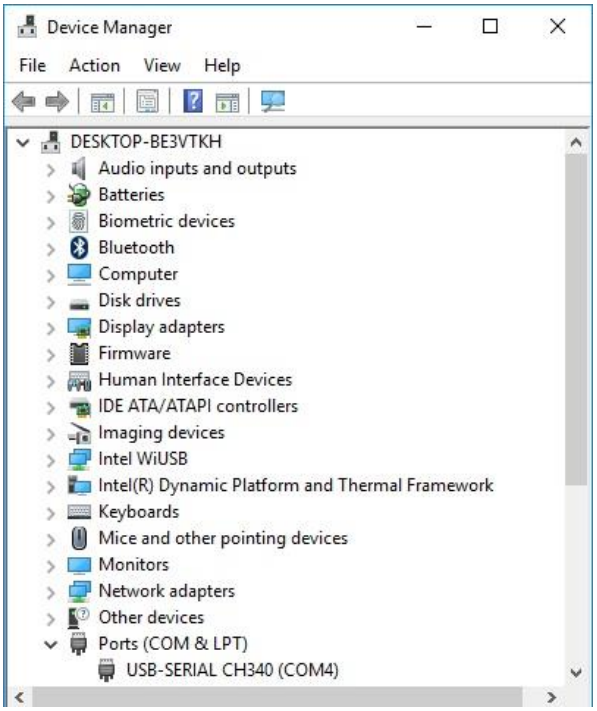
2. Hubungkan kabel USB mini Arduino Nano ke port USB komputer.
3. Buka software IDE Arduino. Ketikkan Program_6_1.ino berikut ini.

Program_6_1.ino

```
1.  void setup() {
2.    Serial.begin(9600);
3.  }
4.  void loop() {
5.    int a = analogRead(A0);
6.    Serial.println(a);
7.    delay(1000);
8.  }
```

4. Setelah selesai, pada menu Tools, pilih Board, pilih Arduino Nano.

5. Pada menu Tools, pilih Port, pilih saluran Port COM yang sedang digunakan. Untuk mengetahui saluran port yang sedang digunakan, buka Device Manager dan buka folder Ports (COM & LPT). Seharusnya di dalam folder Ports (COM & LPT) muncul nama COM yang sedang digunakan. Apabila belum muncul, pembaca perlu menginstal driver USB Arduino Nano yang bernama CH340. Pembaca bisa mendapatkan driver tersebut di CD pendukung buku ini.



Gambar 6.20 Device Manager untuk mengetahui Port COM yang digunakan

6. Setelah driver CH340 berhasil diinstal, maka pilih COM (yang muncul di Device Manager seperti Gambar 6.20) pada pilihan Port di Menu Tools.
7. Setelah itu upload Program_6_1.ino ke Arduino Nano dengan menekan tombol Upload (bergambar tanda panah ke kanan) di Toolbar.
8. Setelah pesan “Done Uploading” muncul, buka jendela Serial Monitor dengan menekan tombol bergambar kaca pembesar, yang terletak di pojok kanan Toolbar.

9. Setelah jendela Serial Monitor terbuka, amati data yang muncul di Serial Monitor untuk posisi batang penunjuk berturut-turut menunjuk ke arah Utara, Timur Laut, Timur, Tenggara, Selatan, Barat Daya, Barat dan Barat Laut. Catat data yang muncul tersebut ke dalam tabel berikut:

No	Arah batang sensor (sudut)	Nilai ADC di Serial Monitor
1.	Utara (0^0)	...
2.	Timur Laut (45^0)	...
3.	Timur (90^0)	...
4.	Tenggara (135^0)	...
5.	Selatan (180^0)	...
6.	Barat Daya (225^0)	...
7.	Barat (270^0)	...
8.	Barat Laut (315^0)	...

10. Berikutnya, setelah diperoleh nilai ADC untuk kedelapan arah mata angin, maka langkah selanjutnya adalah bagaimana membuat Arduino dapat menerjemahkan nilai ADC tersebut menjadi data arah, yang nantinya ditampilkan di Serial Monitor dalam bentuk teks, yaitu berturut-turut menjadi “UTR”, “TLU”, “TMR”, “TGR”, “SLT”, “BDY”, “BRT” dan “BLU”. Namun untuk bisa menerjemahkan nilai ADC menjadi data teks arah tersebut, 2 hal berikut ini perlu menjadi perhatian, yaitu:
    - Nilai ADC yang dibaca Arduno tidak konstan, nilai tersebut selalu berubah dengan kisaran variasi simpangan antara 1 sampai 50.
    - Nilai ADC Sensor yang dihasilkan oleh Sensor tidak sebanding dengan besar posisi sudutnya, dan tidak berurutan sesuai posisinya.

    Untuk mengatasi nilai ADC yang tidak konstan tersebut, dapat dilakukan dengan membuat jangkauan nilai selebar mungkin, sehingga setiap variasi nilai bisa masuk dalam jangkauan. Untuk mendapatkan nilai batas minimum dan maksimal jangkauan, nilai ADC yang berdekatan harus

diketahui. Untuk itu, sekaligus untuk mengatasi hal yang kedua, nilai ADC kedelapan arah mata angin tersebut perlu diurutkan.

11. Tata ulang tabel di langkah no. 9 di atas, dengan cara memasukkan isi tabel di langkah no. 9 di atas ke tabel berikut, namun dengan nilai ADC yang diurutkan, dari nilai ADC kecil ke besar.

Nama ADC	Nilai ADC	Arah batang sensor (sudut)
ADC1	...	...
ADC2	...	...
ADC3	...	...
ADC4	...	...
ADC5	...	...
ADC6	...	...
ADC7	...	...
ADC8	...	...

12. Berikutnya, agar jangkauan nilai di setiap ADC bisa selebar mungkin, maka perlu menentukan batas minimal dan maksimalnya. Untuk itu isi tabel berikut dengan menghitung batas minimal dan maksimal di setiap ADC.

Nama ADC	Nilai ADC	Arah	Batas Minimal	Batas Maksimal
ADC1	...	...	0	(ADC1+ADC2)/2= ...
ADC2	...	...	(ADC1+ADC2)/2= ...	(ADC2+ADC3)/2= ...
ADC3	...	...	(ADC2+ADC3)/2= ...	(ADC3+ADC4)/2= ...
ADC4	...	...	(ADC3+ADC4)/2= ...	(ADC4+ADC5)/2= ...
ADC5	...	...	(ADC4+ADC5)/2= ...	(ADC5+ADC6)/2= ...
ADC6	...	...	(ADC5+ADC6)/2= ...	(ADC6+ADC7)/2= ...
ADC7	...	...	(ADC6+ADC7)/2= ...	(ADC7+ADC8)/2= ...
ADC8	...	...	(ADC7+ADC8)/2= ...	1023

13. Berikutnya, setelah batas minimal dan maksimal di setiap ADC diketahui, maka langkah berikutnya adalah memasukkan nilai-nilai tersebut pada program Arduino berikut ini.

Program_6_2.ino

```
1.  void setup() {
2.    Serial.begin(9600);
3.  }
4.  void loop() {
5.    int a = analogRead(A0);
6.    if (a < bMax1) Serial.println("arah1");
7.    if (a >= bMin2 && a < bMax2) Serial.println("arah2");
8.    if (a >= bMin3 && a < bMax3) Serial.println("arah3");
9.    if (a >= bMin4 && a < bMax4) Serial.println("arah4");
10.   if (a >= bMin5 && a < bMax5) Serial.println("arah5");
11.   if (a >= bMin6 && a < bMax6) Serial.println("arah6");
12.   if (a >= bMin7 && a < bMax7) Serial.println("arah7");
13.   if (a >= bMin8 && a < 1023) Serial.println("arah8");
14.   delay(1000);
15. }
```

Keterangan: bMin adalah nilai batas minimal, **bMax** adalah nilai batas maksimal, angka yang mengikutinya adalah nomor baris sesuai tabel di langkah no. 13.

14. Sebagai contoh, dari Sensor Arah Angin yang telah penulis buat, diperoleh data tabel di langkah no.9 di atas sebagai berikut:

No	Arah batang sensor (sudut)	Nilai ADC di Serial Monitor
1.	Utara (0^0)	508
2.	Timur Laut (45^0)	407
3.	Timur (90^0)	683
4.	Tenggara (135^0)	596
5.	Selatan (180^0)	842
6.	Barat Daya (225^0)	751
7.	Barat (270^0)	893
8.	Barat Laut (315^0)	474

15. Setelah nilai ADC diurutkan, tabelnya menjadi:

Nama ADC	Nilai ADC	Arah batang sensor (sudut)
ADC1	407	TLU (45)
ADC2	474	BLU (315)
ADC3	508	UTR (0)
ADC4	596	TGR (135)
ADC5	683	TMR (90)
ADC6	751	BDY (225)
ADC7	842	SLT (180)
ADC8	893	BRT (270)

16. Berikutnya, batas minimal dan batas maksimal dapat dihitung:

Nama ADC	Nilai ADC	Arah	Batas Minimal	Batas Maksimal
ADC1	407	TLU	0	(ADC1+ADC2)/2= 440.5
ADC2	474	BLU	(ADC1+ADC2)/2= 440.5	(ADC2+ADC3)/2= 491
ADC3	508	UTR	(ADC2+ADC3)/2= 491	(ADC3+ADC4)/2= 552
ADC4	596	TGR	(ADC3+ADC4)/2= 552	(ADC4+ADC5)/2= 639.5
ADC5	683	TMR	(ADC4+ADC5)/2= 639.5	(ADC5+ADC6)/2= 717
ADC6	751	BDY	(ADC5+ADC6)/2= 717	(ADC6+ADC7)/2= 796.5
ADC7	842	SLT	(ADC6+ADC7)/2= 796.5	(ADC7+ADC8)/2= 867.5
ADC8	893	BRT	(ADC7+ADC8)/2= 867.5	1023

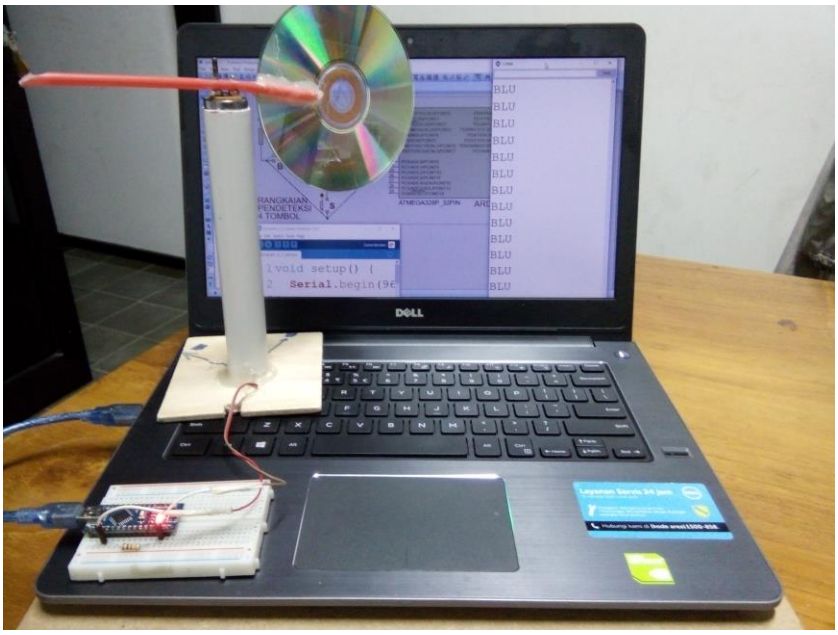
17. Dengan nilai batas minimal dan maksimal serta data arah yang diperoleh dari tabel di langkah no. 16 di atas, maka Program_6_2.ino menjadi seperti berikut:

Program_6_2_contoh.ino

```
1.  void setup() {
2.    Serial.begin(9600);
3.  }
4.  void loop() {
```

```
5.    int a = analogRead(A0);
6.    if (a < 440.5) Serial.println("TLU");
7.    if (a >= 440.5 && a < 491) Serial.println("BLU");
8.    if (a >= 491 && a < 552) Serial.println("UTR");
9.    if (a >= 552 && a < 639.5) Serial.println("TGR");
10.   if (a >= 639.5 && a < 717) Serial.println("TMR");
11.   if (a >= 717 && a < 796.5) Serial.println("BDY");
12.   if (a >= 796.5 && a < 867.5) Serial.println("SLT");
13.   if (a >= 867.5) Serial.println("BRT");
14.   delay(100);
15. }
```

18. Sesuaikan program di langkah no. 17 di atas dengan data yang pembaca peroleh dari Sensor yang pembaca buat. Kemudian upload program tersebut, dan buka Serial Monitor. Gambar 6.21 berikut menunjukkan pembacaan data Sensor dan tampilan datanya di Serial Monitor.

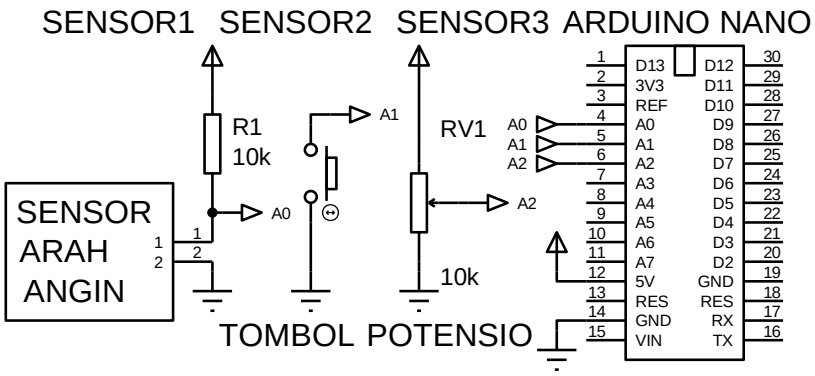


Gambar 6.21 Serial Monitor menampilkan data Sensor Arah Angin

19. Tambahkan sebuah tombol dan sebuah Potensio, dan hubungkan pada kaki A1 dan A2 Arduino Nano, seperti gambar skematik Gambar 6.22.
20. Berikut program untuk menampilkan data Sensor, Tombol dan Potensio.

Program_6_3.ino

```
1.  void setup() {
2.    Serial.begin(9600);
3.    pinMode(A1, INPUT_PULLUP);
4.  }
5.  void loop() {
6.    int a = analogRead(A0);
7.    if (a < 440.5) Serial.println("TLU");
8.    if (a >= 440.5 && a < 491) Serial.println("BLU");
9.    if (a >= 491 && a < 552) Serial.println("UTR");
10.   if (a >= 552 && a < 639.5) Serial.println("TGR");
11.   if (a >= 639.5 && a < 717) Serial.println("TMR");
12.   if (a >= 717 && a < 796.5) Serial.println("BDY");
13.   if (a >= 796.5 && a < 867.5) Serial.println("SLT");
14.   if (a >= 867.5) Serial.println("BRT");
15.   int b = !digitalRead(A1);
16.   Serial.print(',');
17.   Serial.print(b);
18.   int c = analogRead(A2);
19.   Serial.print(',');
20.   Serial.println(c);
21.   delay(100);
22. }
```

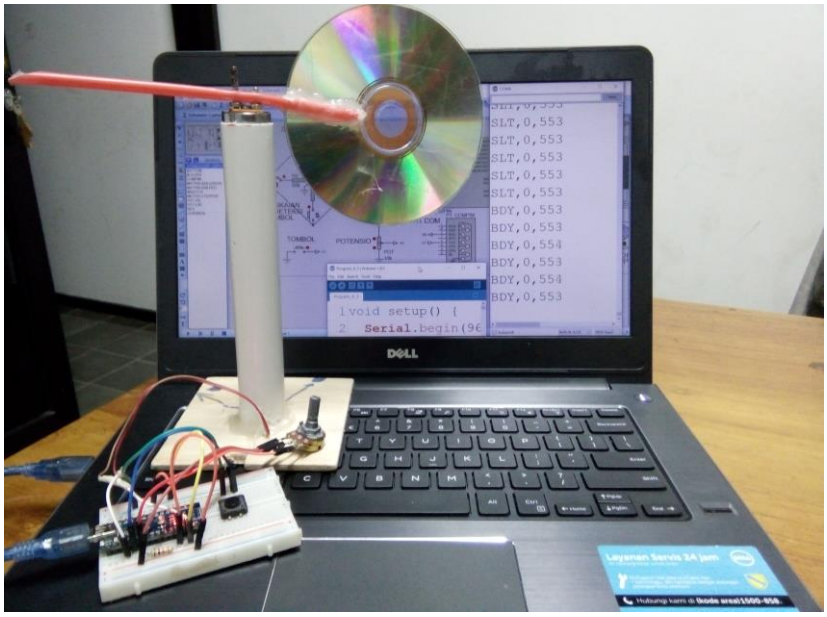


Gambar 6.22 Rangkaian Sensor Arah Angin, Tombol, Potensio dan Arduino Nano

21. Perhatikan pada baris ke 15 Program_6_3.ino di atas, yaitu untuk pembacaan Tombol di kaki A1, digunakan instruksi digitalRead(A1) dengan tanda seru di depan. Tanda seru disini berfungsi sebagai NOT, untuk membalik nilai. Apabila tanpa tanda seru, maka ketika tombol

ditekan, nilai b = 0, dan ketika dilepas nilai b = 1. Dengan penambahan tanda seru, nilai b = 1 ketika tombol ditekan, dan b = 0 ketika dilepas.

22. Upload Program_6_3.ino di atas (jangan lupa menyesuaikan nilai batas minimal dan maksimal serta data arah dalam program dengan Sensor yang pembaca buat). Gambar 6.23 berikut ini menunjukkan tampilan data Sensor Arah Angin, Tombol dan Potensio di Serial Monitor.



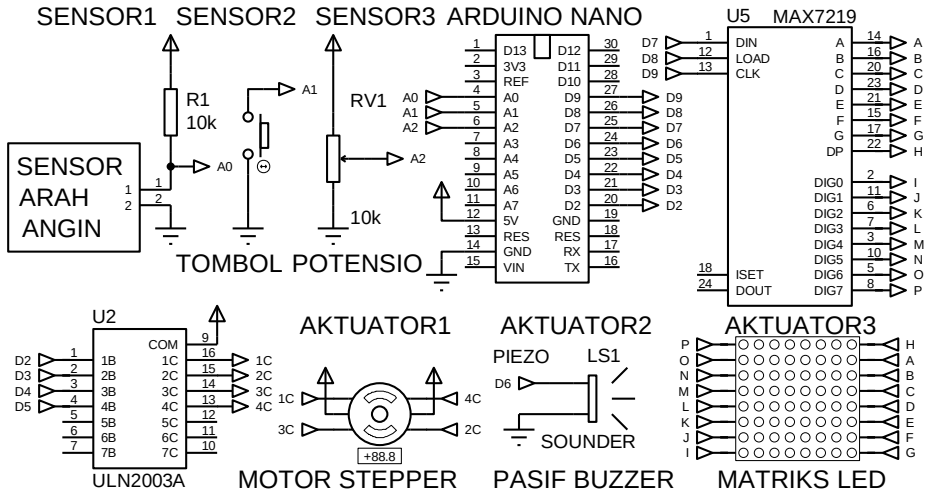
Gambar 6.23 Serial Monitor menampilkan data Sensor, Tombol dan Potensio

6.4 Pengendalian Aktuator Motor Stepper, Buzzer dan Matriks LED

1. Tambahkan sebuah Motor Stepper 28BYJ-48 dengan drivernya (IC ULN2003), sebuah Buzzer tipe Pasif dan sebuah Matriks LED dengan drivernya (MAX7219), sesuai dengan gambar skematik Gambar 6.24.

Catatan: Mengulang apa yang sudah diuraikan di Sub Bab 5.1, di langkah ke 9, tentang Buzzer tipe Pasif. Buzzer jenis ini tidak memiliki rangkaian pengondisi sinyal, sehingga memerlukan input sinyal berbentuk pulsa agar bisa berbunyi.

Namun keuntungannya, Buzzer tipe Pasif ini menghasilkan bunyi yang nadanya dapat diatur sesuai dengan frekuensi inputnya, yaitu semakin tinggi frekuensinya, semakin tinggi nadanya. Buzzer tipe Pasif ini juga sering dikenal sebagai piezoelectric. Pembaca dapat menemukan piezoelectric ini di dalam mainan anak-anak yang mengeluarkan bunyi bernada, biasanya berbentuk bundar pipih seukuran uang logam, namun lebih tipis.



Gambar 6.24 Rangkaian Arduino Nano dengan 3 Sensor dan 3 Aktuator

2. Setelah rangkaian disusun sesuai Gambar 6.24 di atas, berikut program untuk membaca data 3 Sensor dan mengendalikan 3 Aktuator.

Program_6_4.ino

```
#include <Stepper.h>
#include "LedControl.h"
#define do1  1047
#define re  1175
#define mi  1319
#define fa  1397
#define sol  1568
#define la  1760
#define si  1976
#define do2  2093
int nada[] = {do1, re, mi, fa, sol, la, si, do2};
int a1 = 0;
int b1 = 0;
```

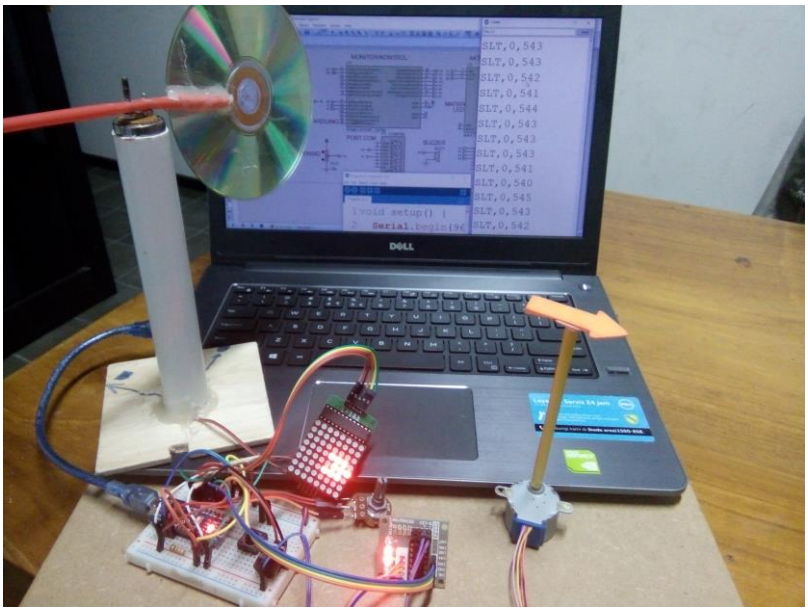
```
int c1 = 0;
int e1 = 0;
const int stepsPerRevolution = 32;
byte pola[8][8] = {
{B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},// UTR
{B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},// TLU
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},// TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},// TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},// SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},// BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},// BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} // BLU
};
Stepper myStepper(stepsPerRevolution, 2, 4, 3, 5);
LedControl lc = LedControl(7, 9, 8, 1);
unsigned long sebelum = 0;
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  myStepper.setSpeed(500);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
}
void loop() {
  int a = analogRead(A0);
  int b = !digitalRead(A1);
  int c = analogRead(A2);
  unsigned long sekarang = millis();
  if (sekarang - sebelum >= 1000) {
    sebelum = sekarang;
    if (a < 440.5) Serial.println("TLU");
    if (a >= 440.5 && a < 491) Serial.println("BLU");
    if (a >= 491 && a < 552) Serial.println("UTR");
    if (a >= 552 && a < 639.5) Serial.println("TGR");
    if (a >= 639.5 && a < 717) Serial.println("TMR");
    if (a >= 717 && a < 796.5) Serial.println("BDY");
    if (a >= 796.5 && a < 867.5) Serial.println("SLT");
    if (a >= 867.5) Serial.println("BRT");
    Serial.print(',');
    Serial.print(b);
    Serial.print(',');
    Serial.println(c);
    a1 = a;
    b1 = b;
    c1 = c;
    noTone(6);
  }
}
void serialEvent() {
  while (Serial.available()) {
```

```
65.     int e = Serial.parseInt() + 1; //posisi Motor
66.     int f = Serial.parseInt(); //nada Buzzer
67.     int g = Serial.parseInt(); //pola Matriks
68.     if (Serial.read() == char(13)) {
69.       for (int i = 0; i < 8; i++) {
70.         lc.setRow(0, i, pola[g - 1][7-i]);
71.       }
72.       if (e1 - e > 180) e = e + 360;
73.       if (e - e1 > 180) e1 = e1 + 360;
74.       tone(6, nada[f - 1], (e1 - e) * 100);
75.       myStepper.step((e – e1) * 5.69);
76.       e1 = e;
77. }}}
```

3. Program_6_4.ino di atas diambil dari Program_5_7.ino, dengan beberapa modifikasi, yaitu di baris 16, baris 27, baris 33, baris 45-52, baris 69 dan baris 74, berikut keterangannya:
   - Baris 16 berkaitan dengan Motor Stepper yang digunakan, yaitu tipe 28BYJ-48, di mana tipe motor ini memiliki 32 step untuk mencapai 1 putaran penuh.
   - Baris 27 berkaitan dengan urutan kumparan Motor Stepper, di mana urutan kumparan kedua dan ketiga terbalik, sehingga perlu untuk mengubah urutannya di kode program.
   - Baris 33 berkaitan dengan kecepatan Motor Stepper.
   - Baris 45-52 berkaitan dengan nilai batas minimal dan maksimal Sensor Arah Angin, yang nilainya diambil dari Program_6_3.ino (pembaca perlu menyesuaikan nilainya dengan data yang diperoleh dari Sensor yang telah pembaca buat).
   - Baris 60 berkaitan dengan mematikan bunyi Buzzer. Sebelumnya instruksi ini ditempatkan di bawah instruksi **tone()**, tepatnya setelah instruksi untuk menggerakkan Stepper, **myStepper.step()**. Secara simulasi, bunyi Buzzer akan terdengar, karena Proteus menunda instruksi berikutnya saat simulasi motor Stepper berjalan. Namun dalam prakteknya, Buzzer tidak pernah bisa berbunyi. Untuk membuat Buzzer berbunyi, setelah instruksi **tone()**, harus ada jeda waktu yang cukup sebelum dibuat diam dengan **noTone()**. Untuk itu, tempatkan **noTone()** di dalam blok timer perdetik..

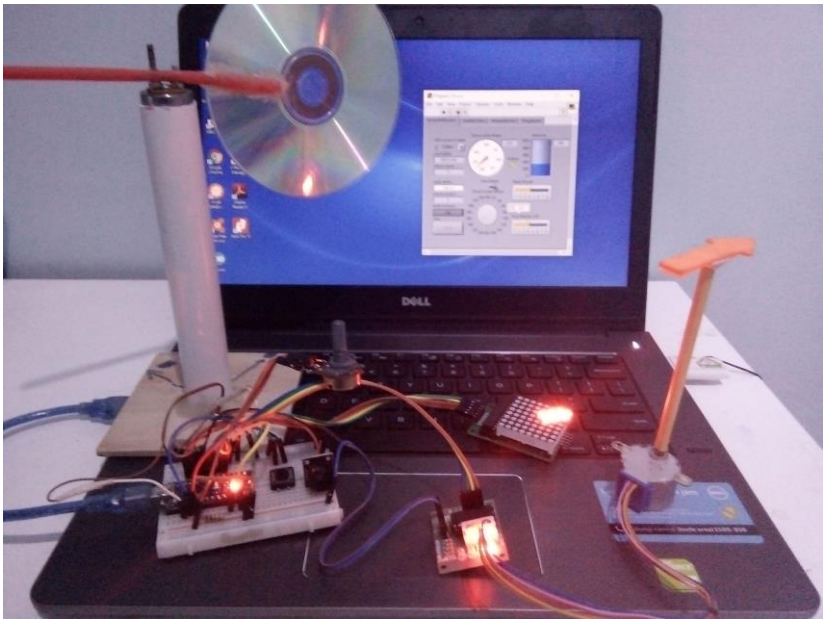
- Baris 69 berkaitan dengan urutan pola Matriks LED. Antara simulasi dengan riil ternyata ada perbedaan pengawatan, sehingga urutannya berbeda, tidak dimulai dari pola pertama, tetapi dari pola terakhir. Untuk mengatasi hal itu, ditambahkan angka 7 yang dikurangi i.
- Baris 74 berkaitan dengan Motor Stepper tipe 28BYJ-48, di mana Motor Stepper tersebut memiliki rasio roda gigi antara poros eksternal motor dengan poros internal motor sebesar 1:64. Jadi ketika poros internal berputar 64 kali putaran penuh, poros eksternal motor baru berputar satu putaran penuh. Sementara untuk bisa memutar 1 putaran penuh di poros internal, dibutuhkan 32 step, sehingga total step untuk memutar poros eksternal satu putaran penuh (atau 360 derajat) adalah 32x64 = 2048 step. Jadi setiap 1 derajat, dibutuhkan 2048/360 = 5,69 step.
- Tambahan di baris 74, selain mengalikan dengan 5.69, nilai e dan e1 juga dibalik karena secara riil, cw dan ccw putaran motor terbalik.

4. Upload Program_6_4.ino di atas. Buka Serial Monitor untuk menampilkan data ketiga Sensor dan mengendalikan ketiga Aktuator.



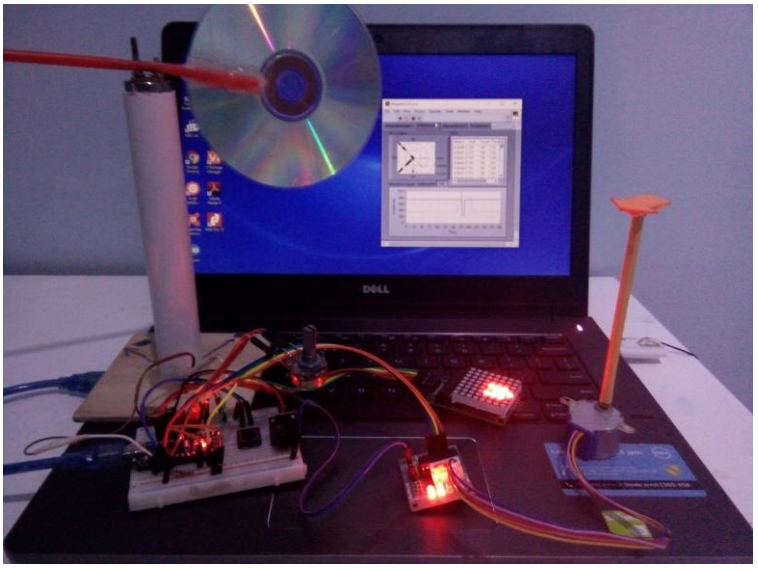
Gambar 6.25 Rangkaian Arduino Nano dengan 3 Sensor dan 3 Aktuator

5. Agar Motor Stepper berputar 90 derajat, Buzzer membunyikan nada ketiga, dan Matriks LED menampilkan pola keempat, ketikkan pada kolom Send 90,3,4 diikuti penekanan tombol Send (namun sebelum menekan tombol Send, pastikan karakter akhiran di Serial Monitor adalah Carriage Return, atau Both NL & CR). Gambar 6.25 di atas menunjukkan hasil Program_6_4.ino.
6. Agar tampilan program di komputer lebih menarik dan interaktif, ganti Serial Monitor dengan tampilan Program LabVIEW, yaitu Program_5_4_e.vi. Tekan tombol Run untuk menjalankan program LabVIEW tersebut. Namun sebelum menekan tombol Run, pastikan VISA Resource Name berisi saluran port yang sedang digunakan. Jangan lupa menutup jendela Serial Monitor, karena program LabVIEW menggunakan saluran port yang sama. Gambar 6.26 berikut menunjukkan tampilan program LabVIEW dan rangkaian 3 Sensor dan 3 Aktuator.



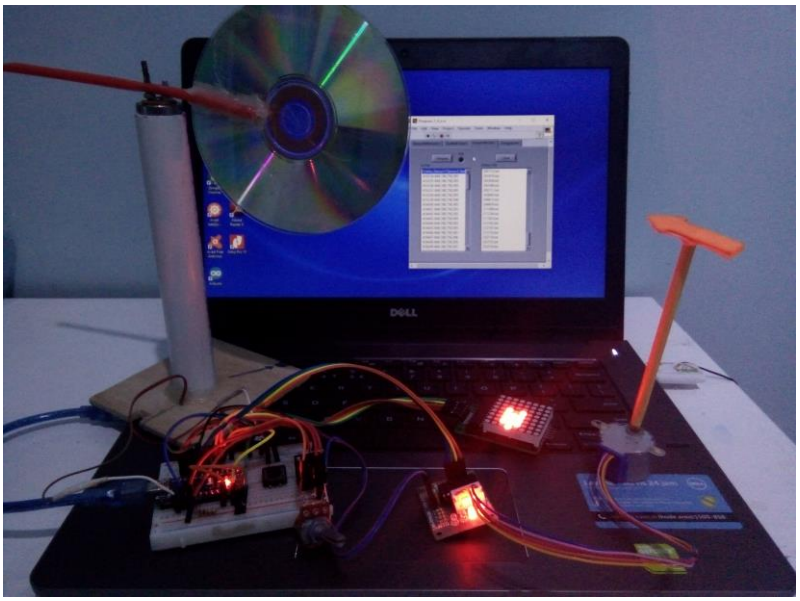
Gambar 6.26 Tampilan program LabVIEW yang menginterface (menghubungkan) 3 Sensor dan 3 Aktuator dalam bentuk objek Gauge, LED, Tank, Dial, dan Slider

7. Putar batang penunjuk di Sensor Arah Angin, dan perhatikan tampilan Gauge di Page Sensor&Aktuator program LabVIEW. Seharusnya jarum di Gauge juga mengikuti arah batang penunjuk. Tekan tombol dan putar potensio. Seharusnya LED menyala (atau padam) dan isi Tank berubah.
8. Geser Toggle Switch Teks/Objek ke kiri. Ketikkan 215,4,5 di kolom write buffer, kemudian tekan tombol Kirim Instruksi. Seharusnya Motor Stepper menggerakkan poros ke posisi sudut 215^0, Buzzer membunyikan nada keempat (fa) dan Matriks LED menampilkan pola kelima.
9. Geser Toggle Switch Teks/Objek ke kanan. Ubah posisi jarum Dial ke 125 (atau dengan cara menuliskan angkanya di Digital Display), geser Slider Nada Buzzer ke nilai 3 dan Slider Pola Matriks LED ke nilai 4. Seharusnya Motor Stepper menggerakkan poros dari posisi sudut 215^0 ke posisi sudut 125^0 (bergerak 90^0 berlawanan arah jarum jam), Buzzer membunyikan nada ketiga (mi) dan Matriks LED menampilkan pola keempat.
10. Ubah Tab ke Page Grafik&Tabel dan amati grafik yang muncul di 2D Compass dan Waveform Graph, setiap kali batang penunjuk Sensor Arah Angin diputar, tombol ditekan dan knob Potensio diputar. Perhatikan juga data yang ditampilkan di Table, seperti ditunjukkan pada Gambar 6.27.



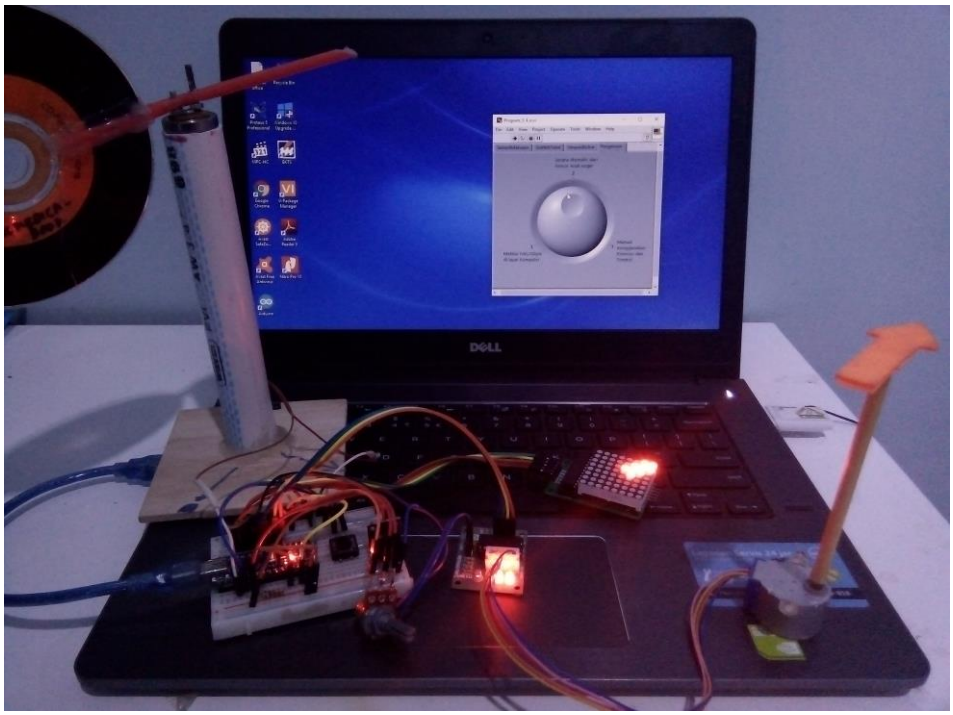
Gambar 6.27 LabVIEW menampilkan data 3 Sensor pada Grafik dan Tabel

11. Ubah Tab ke Page Simpan&Lihat, dan tekan tombol Simpan hingga LED berkedip. Listbox Isi File akan menampilkan data Waktu dan data 3 Sensor dalam satu baris setiap detik, seperti ditunjukkan pada Gambar 6.28. Tekan kembali tombol Simpan untuk menghentikan proses penyimpanan. Untuk melihat file-file yang dihasilkan, tekan tombol Lihat, maka akan muncul daftar nama file di Listbox Daftar File. Klik 2 kali pada salah satu nama file, maka isi file tersebut akan terbuka.



Gambar 6.28 LabVIEW menyimpan data Sensor ke dalam file

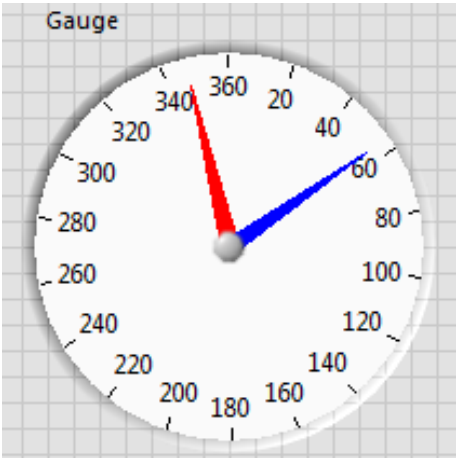
12. Ubah Tab ke Page Pengaturan seperti ditunjukkan pada Gambar 6.29. Secara default Knob Pengaturan diset ke nilai 1, yang artinya ketiga Aktuator (Motor Stepper, Buzzer dan Matriks LED) dapat dikendalikan di Page Sensor&Aktuator baik secara teks/objek.
13. Ubah Knob Pengaturan ke nilai 2, maka ketiga Aktuator tersebut dapat dikendalikan secara otomatis oleh Sensor Arah Angin.
14. Ubah Knob Pengaturan ke nilai 3, maka ketiga Aktuator tersebut dapat dikendalikan secara manual melalui Potensio.



Gambar 6.29 Pengaturan hubungan antara Sensor dan Aktuator

6.5 Soal Latihan

1. Buat sebuah aplikasi Alarm Arah Angin dengan ketentuan sebagai berikut:
   a. Gunakan rangkaian Gambar 6.24 untuk hardware Alarm Arah Angin.
   b. Gunakan objek Gauge dengan skala 0-360 di LabVIEW untuk menampilkan data Sensor Arah Angin. Buat tipe data Gauge menjadi Integer 16 bit (I16).
   c. Tambahkan jarum di objek Gauge, dan buat jarum yang kedua ini menampilkan nilai Potensio. Karena nilai ADC Potensio dari 0-1023, buat penskalaan sehingga nilainya menjadi 0-360. Buat tipe data menjadi I16.
   d. Buat jarum Sensor Arah Angin berwarna merah dan jarum Potensio berwarna biru (gunakan kuas warna di Palet Tools)



Gambar 6.30 Gauge dengan 2 jarum, gunakan Add Needle.

e. Buat agar Motor Stepper menggerakkan porosnya sesuai dengan posisi Sensor Arah Angin, dan Matriks LED menampilkan pola panah dari 1 sampai dengan 8, sesuai nilai Potensio, mengikuti tabel berikut:

Sensor Arah Angin	Motor Stepper
Utara	0
Timur Laut	45
Timur	90
Tenggara	135
Selatan	180
Barat Daya	225
Barat	270
Barat Laut	315

Nilai Potensio	Matriks LED
0-44	Pola ke-1
45-89	Pola ke-2
90-134	Pola ke-3
135-179	Pola ke-4
180-224	Pola ke-5
225-269	Pola ke-6
270-314	Pola ke-7
315-359	Pola ke-8

f. Tambahkan sebuah objek Round LED, yang akan menyala apabila jarum Sensor Arah Angin dan jarum Potensio berada pada daerah yang sama. Sebagai contoh, ketika Sensor Arah Angin di posisi Barat, dan nilai Potensio berada di antara nilai 270-314, maka objek Round LED menyala. Ketika jarum Sensor Arah Angin dan jarum Potensio sudah tidak berada pada daerah yang sama, objek Round LED akan padam.

g. Buat agar ketika objek Round LED menyala, Buzzer berbunyi, dan ketika objek Round LED padam, Buzzer tetap berbunyi. Buzzer hanya berhenti berbunyi apabila Tombol (Sensor 2) ditekan.
h. Buat ketika Buzzer berbunyi, nada dari Buzzer semakin meninggi setiap menit, dari nada do rendah ke re, dari re ke mi, dan seterusnya, dan ketika sudah sampai nada do tinggi (berarti sudah lebih 8 menit), kembali berulang ke nada do rendah lagi dan seterusnya, hingga berhenti, yaitu ketika Tombol (Sensor 2) ditekan.

2. Buat grafik data perdetik dengan rentang waktu 60 menit (=3600 detik) untuk posisi Sensor Arah Angin, nilai Potensio, kondisi Round LED, kondisi Buzzer dan kondisi Tombol.
3. Tambahkan Table yang menampilkan data catatan waktu di kolom pertama dan keterangan di kolom kedua. Table hanya memunculkan data baru apabila:
    a. Objek Round LED dari padam menjadi menyala.
    b. Objek Round LED dari menyala menjadi padam.
    c. Tombol (Sensor 2) ditekan.

    Buat kolom pertama menampilkan catatan waktu dengan urutan tahun-bulan-tgl, jam-menit-detik. Buat kolom kedua menampilkan keterangan berupa data posisi Sensor Arah Angin, nilai Potensio dan kondisi Tombol (Sensor 2). Buat agar data terbaru selalu berada di atas.
4. Tambahkan penyimpanan data grafik (langkah no. 2) secara otomatis ke dalam file txt setiap jam, dengan nama file mengikuti format waktu, yaitu tahun-bulan-tgl-jam-menit-detik.

BAB 7
KOMUNIKASI DATA SECARA NIRKABEL

Capaian Pembelajaran:

- Pembaca mampu menghubungkan Arduino dengan komputer melalui komunikasi data secara nirkabel menggunakan Bluetooth HC-05.
- Pembaca mampu menghubungkan 2 buah atau lebih Arduino melalui komunikasi data secara nirkabel menggunakan NRF24L01.

Persoalan:

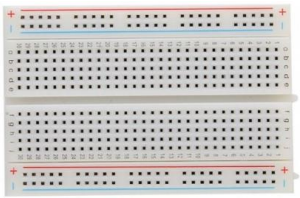
Buat agar Sensor dan Aktuator ditempatkan secara terpisah pada jarak yang cukup jauh dari komputer, dan keduanya tetap dapat dimonitor dan dikontrol dari komputer secara nirkabel.

Uraian Materi:

7.1 Komponen yang Diperlukan

Berikut daftar komponen tambahan yang diperlukan:

No.	Bahan/Komponen	Jumlah	Gambar
1.	USB Bluetooth Adapter untuk komputer yang tidak memiliki fasilitas Bluetooth	1	

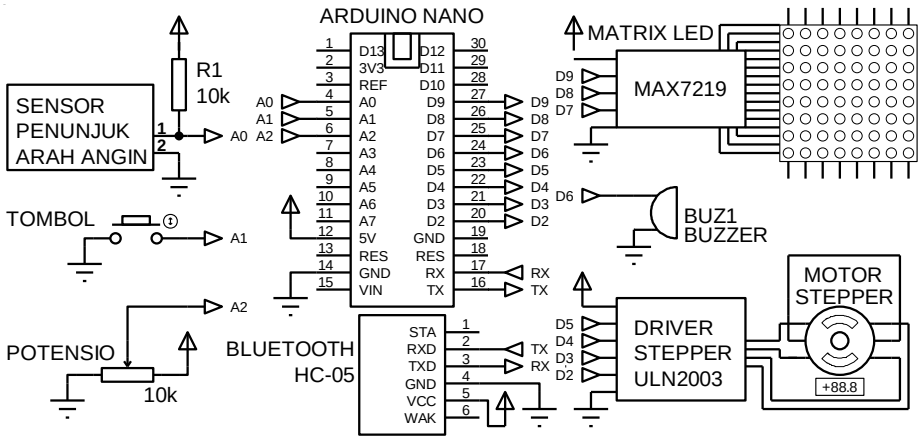
No.	Bahan/Komponen	Jumlah	Gambar
2	Bluetooth HC-05	1	
3.	NRF24L01	3	
4.	Breadboard	2	
5.	Kabel Jumper	±20	
6.	Regulator 3,3V (AMS1117)	3	
7.	Kapasitor 100uF	3	
8.	Arduino Nano dan Kabel	2	

Catatan: Perhatikan bahwa di Bab ini diperlukan tambahan Arduino, Breadboard dan kabel jumper, karena akan dilakukan pemisahan rangkaian Sensor dan Aktuator, yang keduanya membutuhkan sebuah Arduino.

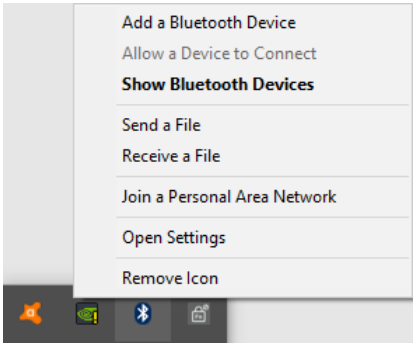
7.2 Komunikasi Nirkabel dengan HC-05

Sebuah hal yang menarik ketika ada alat yang dapat dimonitor posisi dan kondisinya di komputer, dan dapat dikendalikan gerakannya dari komputer, tanpa ada kabel yang menghubungkan alat tersebut dengan komputer. Bagaikan "sesuatu" yang magis dan supranatural. Itulah sebabnya di Bab terakhir ini, penulis menambahkan komunikasi nirkabel. Sebenarnya bukan hanya agar terlihat lebih menarik saja, tetapi karena alasan kebutuhan di lapangan. Seperti diketahui, penempatan Sensor Arah Angin tentunya tidak di dalam ruangan, karena harus dapat membaca Arah Angin di luar ruangan. Sementara itu, komputer harus ditempatkan di dalam ruangan. Sekalipun keduanya ditempatkan di lokasi yang berbeda, keduanya harus dihubungkan. Menghubungkan keduanya bisa dengan kabel, namun cara ini kurang praktis dan cukup merepotkan. Karena itu, dipilihlah komunikasi tanpa kabel dengan Bluetooth HC-05. Berikut ini langkah-langkahnya:

1. Buat rangkaian seperti Gambar 6.24, dan upload Program_6_4.ino.
2. Pastikan bahwa komputer memiliki fasilitas Bluetooth. Apabila belum ada, gunakan USB Adapter Bluetooth.
3. Tambahkan Bluetooth HC-05 pada rangkaian. Sambungkan kaki TXD HC-05 dengan kaki RX Arduino, dan kaki RXD HC-05 dengan kaki TX Arduino, seperti ditunjukkan pada gambar skematik Gambar 7.1.
4. Agar bisa berkomunikasi, buat agar Bluetooth HC-05 berpasangan (*pair*) dengan Bluetooth di komputer. Untuk itu klik kanan pada icon Bluetooth, pilih Show Bluetooth Devices, seperti ditunjukkan pada Gambar 7.2. (penulis menggunakan versi Windows 10, apabila pembaca menggunakan versi Windows yang lebih lama, pembaca dapat mengunduh artikel di internet dengan judul "Cheap-2-Way-Bluetooth-Connection-Between-Arduino-a.pdf" untuk membuat *pairing* HC-05 dengan komputer).

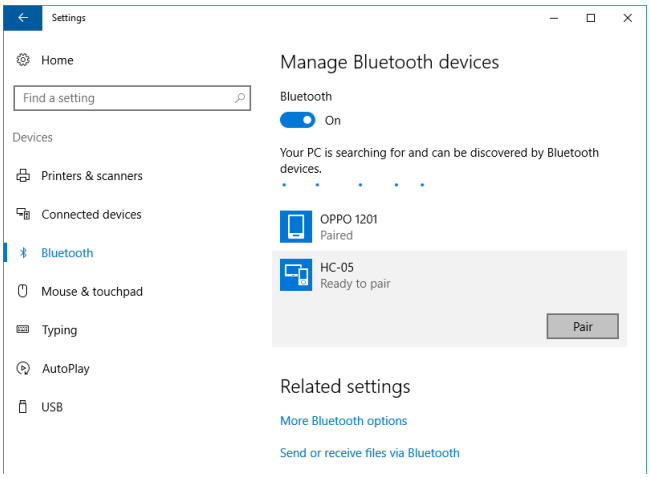


Gambar 7.1 Penyambungan Bluetooth HC-05 dengan kaki RX dan TX Arduino

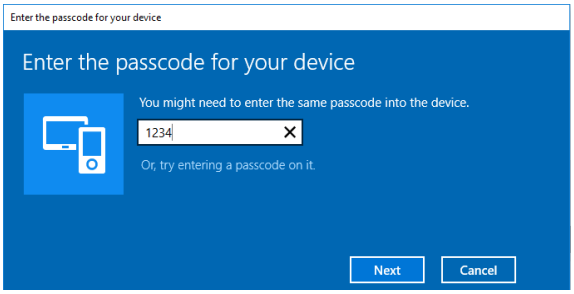


Gambar 7.2 Klik icon Bluetooth, pilih Show Bluetooth Devices

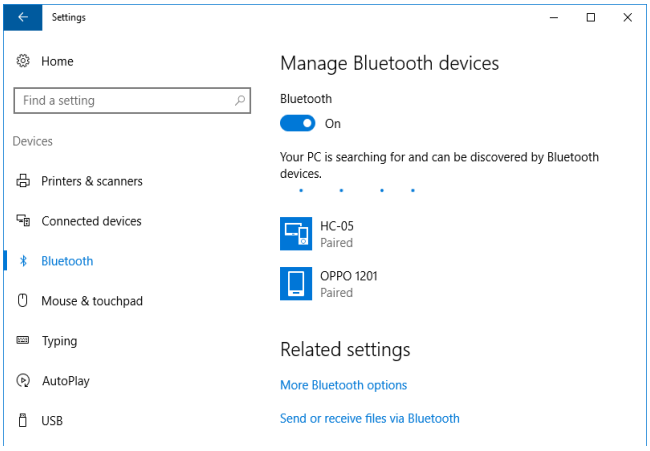
5. Pada jendela yang muncul, pilih nama HC-05, dan tekan tombol Pair, seperti ditunjukkan pada Gambar 7.3.
6. Masukkan kode Pair 1234 pada jendela permintaan passcode, seperti ditunjukkan pada Gambar 7.4.
7. Jika *pairing* berhasil, maka status pada HC-05 adalah *paired*, seperti ditunjukkan pada Gambar 7.5.
8. Untuk mengetahui saluran port yang digunakan oleh Bluetooth HC-05, pilih HC-05, klik pada More Bluetooth options. Pada jendela yang muncul, buka Tab Com Ports. Cari nama COM dengan Direction Outgoing, dan gunakan COM tersebut sebagai pilihan saluran port pada komputer.



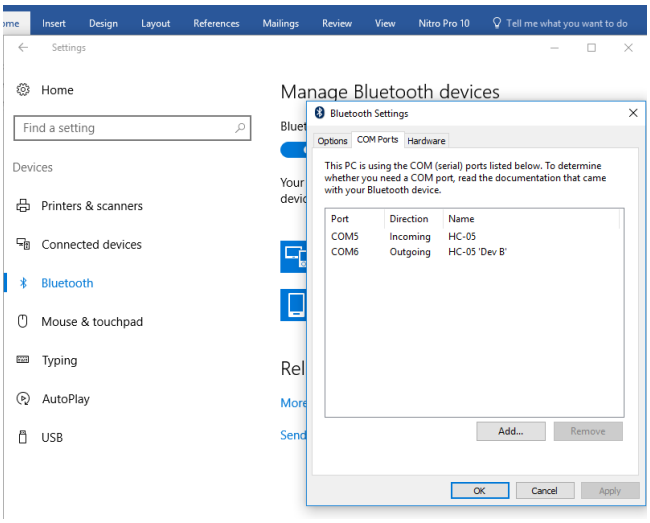
Gambar 7.3 Cari nama HC-05, dan tekan tombol Pair



Gambar 7.4 Masukkan passcode = 1234

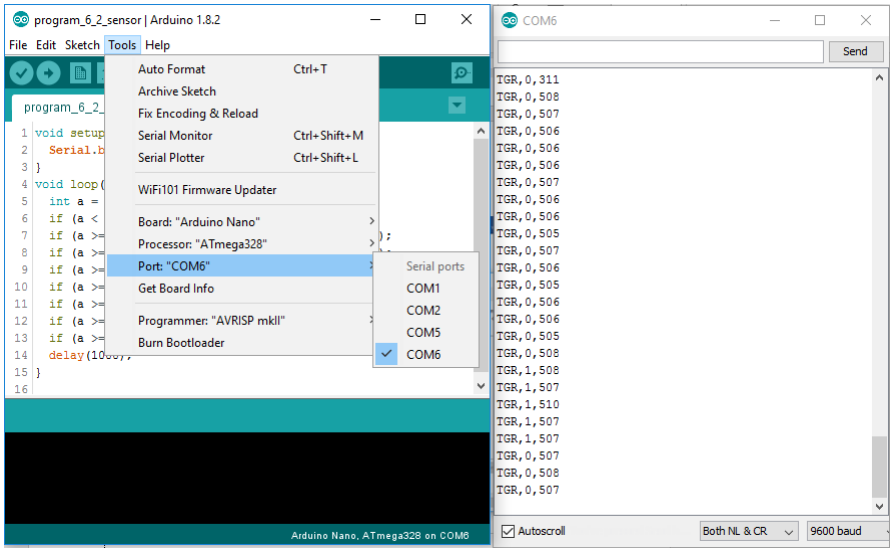


Gambar 7.5 Status Bluetooth HC-05 telah paired dengan komputer



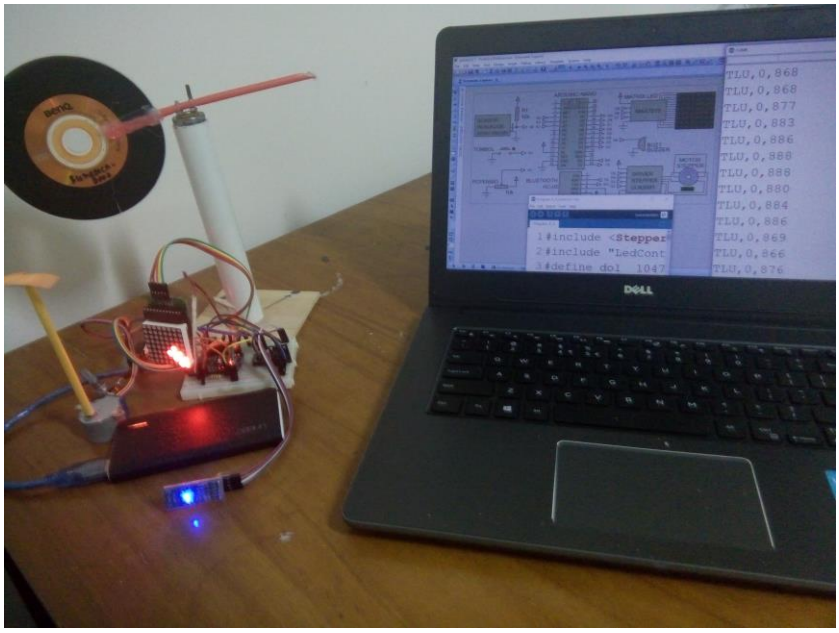
Gambar 7.6 Untuk mengetahui saluran port yang digunakan, buka Bluetooth Settings (More Bluetooth options), pilih Tab COM Ports, dan cari nama COM dengan Direction Outgoing

9. Dari Gambar 7.6 di atas, tampak nama COM dengan Direction Outgoing adalah COM6. Gunakan nama COM ini sebagai saluran port di computer untuk komunikasi dengan Arduino yang terhubung dengan HC-05, seperti ditunjukkan pada Gambar 7.7.



Gambar 7.7 Menggunakan COM6 (Outgoing) untuk komunikasi nirkabel dengan HC-05

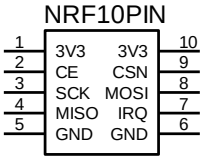
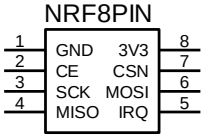
10. Gambar 7.8 berikut menunjukkan bagaimana rangkaian Sensor dan Aktuator terhubung dengan komputer secara nirkabel melalui Bluetooth. Data Sensor dikirimkan oleh Arduino ke komputer, dan ditampilkan di Serial Monitor. Sebaliknya dari komputer melalui Serial Monitor, data Aktuator dikirimkan ke Arduino.



Gambar 7.8 Komunikasi nirkabel melalui Bluetooth, yaitu komputer menerima data Sensor, menampilkan datanya di Serial Monitor dan mengirimkan data Aktuator ke Arduino

7.3 Komunikasi Nirkabel dengan NRF24L01

Kelemahan Bluetooth HC-05 adalah jarak jangkauan komunikasi yang tidak bisa lebih dari 10 m (kecuali untuk versi Bluetooth yang baru). Padahal kebutuhan praktis di lapangan memerlukan komunikasi dengan jarak jangkauan yang lebih jauh. Untuk mengatasi hal ini, sebuah modul komunikasi yang sangat populer (karena harganya yang murah namun cukup handal) yaitu modul NRF24l01, dapat menjadi solusi. Ada beberapa jenis NRF24L01, 2 di antaranya yang penulis gunakan adalah sebagai berikut:

No.	Nama	Gambar	Skema	Keterangan
1	NRF24L01 warna hijau dengan 10 kaki			Jarak jangkauan hingga 100m
2.	NRF24L01+PA+LNA dengan 8 kaki dan batang antenna .			Jarak jangkauan hingga 1000m

Berikut hal-hal yang perlu diperhatikan dalam menggunakan modul NRF24L01, yang penulis ambil dari ***arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo***:

1. Tegangan kerja NRF24L01 adalah 3,3V, bukan 5V. Jangan mensuplai tegangan tersebut dari 3,3V Arduino, karena arus yang dikeluarkan sangat kecil, tidak cukup untuk memberi suplai daya ke NRF24L01. Solusi yang terbaik adalah menggunakan tegangan 5V Arduino untuk diubah ke 3,3V dengan penambahan regulator 3,3V.
2. Di samping suplai daya harus memadai, tegangan yang diberikan juga harus bersih dari noise dan rata. Untuk menghlangkan noise digunakan kapasitor decoupling 0,1uF. Penulis menyarankan penggunaan modul AMS1117 di Gambar 7.9 berikut ini, yang merupakan regulator 3,3V yang sudah dilengkapi dengan kapasitor decoupling.

Gambar 7.9 Modul AMS1117 regulator 3,3V dengan kapasitor decoupling

3. Agar tegangan yang diterima NRF24L01 lebih rata, pembaca dapat menambahkan kapasitor dengan nilai minimal 10uF pada kaki 3,3V dan GND NRF24L01. Semakin besar nilai kapasitansinya semakin baik.
4. Setelah suplai daya harus memadai dan bersih dari noise, hal lain yang perlu diperhatikan adalah jangkauan komunikasi. Untuk mendapatkan jangkauan yang lebih jauh, pembaca dapat mengatur laju pengiriman data ke nilai paling rendah, yaitu 250kHz. Gunakan instruksi berikut ini:

```
radio.setDataRate(RF24_250KBPS);
```

5. Selain laju pengiriman diatur ke nilai paling rendah, level PA (Power Amplifier) dapat diatur ke nilai maksimum agar mendapatkan jangkauan yang jauh. Gunakan instruksi berikut ini:

```
radio.setPALevel(RF24_PA_MAX);
```

   Namun demikian, perlu diperhatikan bahwa instruksi di atas akan membuat konsumsi daya ke rangkaian sangat besar. Apabila pembaca tidak yakin bahwa suplai daya yang dimiliki sudah cukup memadai, saran penulis sebaiknya mengatur level PA ini ke LOW. Apabila komunikasi sudah berhasil, baru tingkatkan level PA ini ke yang lebih tinggi. Urutan level PA ini adalah dari MIN, LOW, HIGH, hingga MAX.
6. Apabila sampai langkah no. 5 di atas, ternyata komunikasi antara NRF24l01 masih belum berhasil, maka ada kemungkinan dikarenakan interferensi gelombang akibat banyaknya pemancar dengan frekuensi yang sama dengan NRF24L01 di lingkungan sekitar. Frekuensi yang digunakan NRF24L01 adalah 2,4GHz, yang juga dipakai untuk Bluetooth, Wifi router, dll. Untuk mengatasi hal ini, pembaca dapat membuat frekuensi NRF24L01 menjadi sedikit lebih tinggi dari 2,4GHz, yaitu dengan menambahkan instruksi berikut:

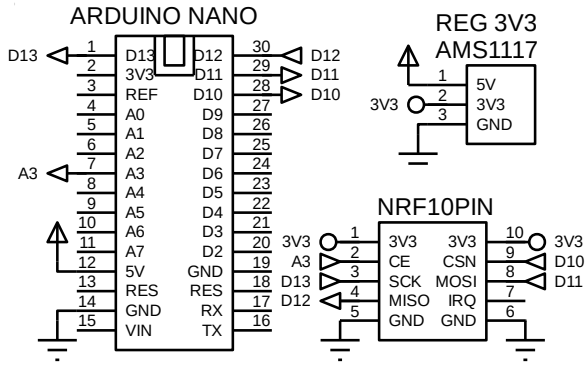
```
radio.setChannel(117);
```

NRF24L01 menyediakan Channel dari 1 sampai dengan 124 yang akan menambah nilai frekuensi sebesar 1 MHz untuk setiap kenaikan. Dengan mengatur Channel ke 117, berarti penambahan frekuensi sebesar 117 MHz, sehingga total frekuensi NRF24L01 menjadi 2,5 GHz + 117 MHz = 2,517 GHz. Dengan membuat frekuensi sebesar itu, cukup untuk menghilangkan interferensi dari sinyal WiFi dan Bluetooth secara umum.

7.3.1 Komunikasi 1 Arah NRF24L01

Agar tidak terlalu berpanjang lebar, maka akan lebih baik bila pembaca langsung menggunakan NRF24L01 dengan hal yang sederhana dulu, yaitu membuat komunikasi 1 arah antara 2 buah NRF24L01. Jenis NRF24L01 yang penulis gunakan sebagai contoh di sini adalah NRF24L01 warna hijau dengan 10 kaki. Bila pembaca menggunakan jenis yang 8 kaki, silahkan disesuaikan. Berikut langkah-langkahnya:

1. Buat 2 rangkaian yang sama seperti gambar skematik berikut ini.



Gambar 7.10 Rangkaian pemancar dan penerima dengan NRF24L01, Arduino dan AMS1117

2. Tentukan salah satu rangkaian sebagai rangkaian pemancar dan rangkaian yang lain sebagai rangkaian penerima. Upload kode program berikut ini untuk Arduino di rangkaian pemancar.

Program_7_1.ino

```
1.  #include <SPI.h>
2.  #include <RF24.h>
3.  RF24 radio(A3, 10);
4.  byte pipa[][6] = {"pipa1"};
5.  unsigned long sebelum = 0;
```

```
 6.  struct dataStruct {
 7.    int data1;
 8.    float data2;
 9.    bool data3;
10.  } dataKu;
11.  void setup() {
12.    Serial.begin(9600);
13.    radio.begin();
14.    radio.setChannel(117);
15.    radio.setDataRate(RF24_250KBPS);
16.    radio.setPALevel(RF24_PA_LOW);
17.    radio.openWritingPipe(pipa[0]);
18.    radio.stopListening();
19.  }
20.  void loop() {
21.    dataKu.data1 = analogRead(A0);
22.    dataKu.data2 = analogRead(A1)/10.0;
23.    dataKu.data3 = analogRead(A2)/1023;
24.    unsigned long sekarang = millis();
25.    if (sekarang - sebelum >= 2000) {
26.      radio.write(&dataKu, sizeof(dataKu));
27.      sebelum = sekarang;
28.      Serial.print(dataKu.data1);
29.      Serial.print(',');
30.      Serial.print(dataKu.data2);
31.      Serial.print(',');
32.      Serial.println(dataKu.data3);
33.    }
34.  }
```

3. Upload kode program berikut ini untuk Arduino di rangkaian penerima.

Program_7_2.ino

```
 1.  #include "SPI.h"
 2.  #include "RF24.h"
 3.  RF24 radio(A3, 10);
 4.  byte pipa[][6] = {"pipa1"};
 5.  struct dataStruct {
 6.    int data1;
 7.    float data2;
 8.    bool data3;
 9.  } dataKu;
10.  void setup() {
11.    Serial.begin(9600);
12.    radio.begin();
13.    radio.setChannel(117);
14.    radio.setDataRate(RF24_250KBPS);
15.    radio.setPALevel(RF24_PA_LOW);
16.    radio.openReadingPipe(1, pipa[0]);
```

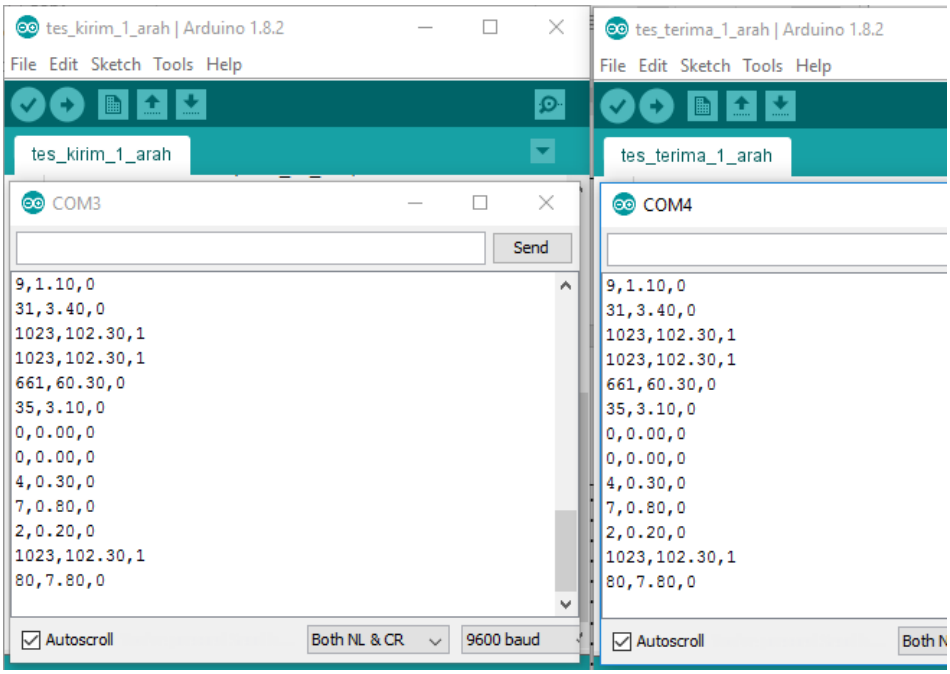
```
17.   radio.startListening();
18. }
19. void loop()  {
20.   if (radio.available()) {
21.     radio.read( &dataKu, sizeof(dataKu) );
22.     Serial.print(dataKu.data1);
23.     Serial.print(',');
24.     Serial.print(dataKu.data2);
25.     Serial.print(',');
26.     Serial.println(dataKu.data3);
27.   }
28. }
```

Catatan: Program_7_1.ino membuat rangkaian pemancar mengirimkan 3 buah data, yaitu data sinyal analog di kaki A0, A1 dan A2 Arduino di rangkaian pemancar ke rangkaian penerima setiap 2 detik dan menampilkan ketiga data tersebut di Serial Monitor. Program_7_2.ino membuat rangkaian penerima menerima 3 buah data tersebut dan menampilkannya di Serial Monitor.

Perhatikan kode program NRF24L01 di blok void setup() pada kedua program tersebut. Tampak bahwa programnya hampir sama, hanya berbeda pada 2 baris program terakhir. Untuk program rangkaian pemancar, instruksi yang digunakan adalah radio.openWritingPipe(alamat) dan radio.stopListening(), dengan alamat diisi "pipa1". Sedangkan untuk program rangkaian penerima, instruksi yang digunakan adalah radio.openReadingPipe(n,alamat) dan radio.startListening(), dengan alamat yang sama, yaitu "pipa1" (pembaca dapat memberi nama lain, dalam contoh ini menggunakan nama pipa1 agar mudah diingat). Nilai n di sini diisi 1 karena data yang diterima hanya dari satu pengirim. Apabila ada lebih dari satu pengirim yang akan diterima pada saat yang bersamaan, maka pembaca dapat mengisi nilai n dengan jumlah pengirim. Maksimum jumlah pengirim yang dapat diterima pada saat bersamaan adalah 6 buah.

Jumlah byte data yang dapat dikirimkan dalam sekali pengiriman maksimum adalah 32 byte. Ketika data yang dikirimkan lebih dari 32 byte, maka pembaca perlu untuk memecahnya menjadi beberapa bagian. Apabila data yang dikirimkan memiliki tipe data yang berbeda-beda, maka pembaca dapat menggunakan instruksi struct, yang mengumpulkan data-data yang berbeda tipenya tersebut dalam satu nama, seperti terlihat pada contoh program di atas.

4. Setelah upload program selesai, sementara kabel USB di kedua rangkaian masih terhubung dengan komputer, buka Serial Monitor. Diinginkan, agar software Arduino IDE dapat menampilkan data dari rangkaian pemancar di Serial Monitor, dan juga data dari rangkaian penerima di Serial Monitor kedua, seperti ditunjukkan pada Gambar 7.11. Untuk membuat hal itu, pembaca perlu membuka software Arduino IDE lagi yang baru. Atur Port pada Arduino IDE yang sebelumnya telah terbuka ke COM yang digunakan rangkaian pemancar (dalam contoh di sini menggunakan COM3), sedangkan pada Arduino IDE yang baru saja dibuka, atur Port ke COM yang digunakan rangkaian penerima (dalam contoh di sini menggunakan COM4). Setelah itu buka Serial Monitor di kedua Arduino IDE tersebut. Seharusnya data yang ditampilkan di Serial Monitor rangkaian pemancar, juga muncul di Serial Monitor rangkaian penerima.
5. Sesekali, sentuhkan jari tangan di kaki A0, A1 dan A2 Arduino di rangkaian pemancar untuk mengubah sinyal analog di ketiga kaki tersebut.



Gambar 7.11 Data rangkaian pemancar ditampilkan di Serial Monitor di COM3 dan data rangkaian penerima ditampillkan di Serial Monitor di COM4

7.3.2 Komunikasi 2 Arah NRF24L01

Pada prakteknya, komunikasi 1 arah tidak cukup memenuhi kebutuhan. Diinginkan ada komunikasi 2 arah antara rangkaian satu dengan rangkaian lainnya. Untuk bisa menerapkan komunikasi 2 arah, ikuti langkah-langkah berikut ini:

1. Gunakan kedua rangkaian yang sama seperti pada komunikasi 1 arah di atas, yaitu mengikuti gambar skematik Gambar 7.10. Karena kedua rangkaian tersebut sama-sama bisa mengirim dan menerima, maka untuk membedakan keduanya, beri nama yang satu Master dan yang kedua Slave. Rangkaian Master adalah rangkaian yang memulai komunikasi, sedangkan rangkaian Slave adalah rangkaian yang merespon balik.
2. Upload kode program berikut ini untuk Arduino di rangkaian Master.

Program_7_3.ino

```
#include "SPI.h"
#include "RF24.h"
RF24 radio(A3, 10);
byte pipa[][6] = {"pipa1", "pipa2"};
struct dataStruct {
  int data1;
  float data2;
  bool data3;
} dataKirim;
struct dataStruct1 {
  int data1;
  float data2;
  bool data3;
} dataTerima;
unsigned long sebelum = 0;
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openWritingPipe(pipa[0]);
  radio.openReadingPipe(1, pipa[1]);
  radio.stopListening();
}
void loop()  {
  dataKirim.data1 = analogRead(A0);
  dataKirim.data2 = analogRead(A1)/10.0;
  dataKirim.data3 = analogRead(A2)/1023;
  unsigned long sekarang = millis();
```

```
  if (sekarang - sebelum >= 5000) {
    sebelum = sekarang;
    radio.stopListening();
    radio.write(&dataKirim, sizeof(dataKirim));
    radio.startListening();
  }
  if (radio.available()) {
    radio.read( &dataTerima, sizeof(dataTerima) );
    Serial.print(dataTerima.data1);
    Serial.print(',');
    Serial.print(dataTerima.data2);
    Serial.print(',');
    Serial.println(dataTerima.data3);
  }
}
```

3. Upload kode program berikut ini untuk Arduino di rangkaian Slave.

Program_7_4.ino

```
#include "SPI.h"
#include "RF24.h"
RF24 radio(A3, 10);
byte pipa[][6] = {"pipa1", "pipa2"};
struct dataStruct {
  int data1;
  float data2;
  bool data3;
} dataKirim;
struct dataStruct1 {
  int data1;
  float data2;
  bool data3;
} dataTerima;
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openReadingPipe(1, pipa[0]);
  radio.openWritingPipe(pipa[1]);
  radio.startListening();
}
void loop() {
  dataKirim.data1 = analogRead(A0);
  dataKirim.data2 = analogRead(A1)/10.0;
  dataKirim.data3 = analogRead(A2)/1023;
  if (radio.available()) {
    radio.read(&dataTerima, sizeof(dataTerima));
```

```
31.      Serial.print(dataTerima.data1);
32.      Serial.print(',');
33.      Serial.print(dataTerima.data2);
34.      Serial.print(',');
35.      Serial.println(dataTerima.data3);
36.      radio.stopListening();
37.      radio.write(&dataKirim, sizeof(dataKirim));
38.      radio.startListening();
39.    }
40.  }
```

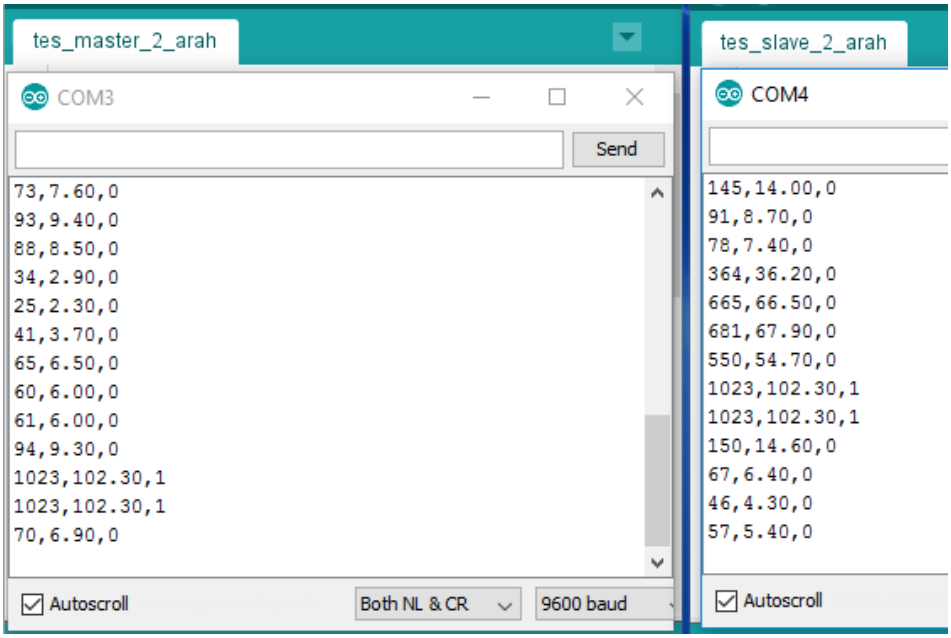
Catatan: Program_7_3.ino membuat rangkaian Master mengirimkan data sinyal analog di kaki A0, A1 dan A2 Arduino di rangkaian Master ke rangkaian Slave setiap 5 detik. Program_7_4.ino membuat rangkaian Slave merespon balik kiriman data dari rangkaian Master tersebut dengan mengirimkan data sinyal analog di kaki A0, A1 dan A2 Arduino di rangkaian Slave ke rangkaian Master.

Perhatikan kode program NRF24L01 di blok void setup() pada kedua program tersebut. Tampak bahwa kode program di blok void setup() tersebut hampir sama dengan kode program di blok void setup() untuk komunikasi 1 arah, hanya ada satu penambahan instruksi radio.openReadingPipe(n,alamat) di program untuk Master, dan penambahan satu instruksi radio.openWritingPipe(alamat) di program untuk Slave. Alamat di kedua instruksi yang baru tersebut menggunakan alamat "pipa2". Jadi ada 2 alamat yang digunakan, "pipa1" dan "pipa2". Alamat "pipa1" digunakan untuk mengirimkan data dari Master ke Slave, sedangkan alamat "pipa2" digunakan untuk mengirimkan data dari Slave ke Master.

Hal yang perlu diperhatikan berikutnya adalah penggunaan instruksi millis sebagai ganti instruksi delay. Sebaiknya jangan menggunakan delay. Jika pembaca menggunakan delay, maka besar kemungkinan data yang datang tidak dapat langsung diterima karena sedang menunggu delay selesai. Dengan instruksi millis, sementara waktu pengiriman data belum tercapai, data yang datang sewaktu-waktu akan langsung dapat diterima dan dibaca, tanpa perlu menunggu.

4. Setelah upload program selesai, buka Serial Monitor untuk menampilkan data yang diterima rangkaian Master dan Serial Monitor yang lain untuk menampilkan data yang diterima rangkaian Slave.

5. Berikut ini tampilan data di kedua Serial Monitor. Serial Monitor yang kiri, yang terhubung dengan Port COM3, menampilkan data yang diterima rangkaian Master. Serial Monitor yang kanan, yang terhubung dengan Port COM4, menampilkan data yang diterima rangkaian Slave.



Gambar 7.12 Serial Monitor di COM3 menampilkan data yang diterima Arduino di rangkaian Master dan Serial Monitor di COM4 menampilkan data yang diterima Arduino Slave.

7.3.3 Komunikasi Multi-Nodes NRF24L01

Dalam perkembangannya, seringkali dibutuhkan jumlah Slave yang lebih dari satu. Komunikasi antara Master dengan Slave yang jumlahnya lebih dari satu sering disebut sebagai komunikasi Multi-Nodes. Untuk bisa menerapkan komunikasi Multi-Nodes ini, ikuti langkah-langkah berikut ini:

1. Buat lagi satu rangkaian yang sama, mengikuti gambar skematik Gambar 7.10, sehingga total ada 3 buah rangkaian.
2. Tentukan mana dari ketiga rangkaian tersebut yang menjadi Master, Slave1 dan Slave2. Upload kode program berikut ini untuk Arduino di rangkaian Master:

Program_7_5.ino

```
#include "SPI.h"
#include "RF24.h"
RF24 radio(A3, 10);
byte pipa[][6] = {"pipa1", "pipa2", "pipa3", "pipa4"};
struct dataStruct {
  int counter;
  int data1;
  float data2;
  bool data3;
} dataKirim;
struct dataStruct1 {
  int counter;
  int data1;
  float data2;
  bool data3;
} dataTerima1;
struct dataStruct2 {
  int counter;
  int data1;
  float data2;
  bool data3;
} dataTerima2;
int giliran = 0;
unsigned long sebelum = 0;
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.stopListening();
}
void loop()  {
  dataKirim.data1 = analogRead(A0);
  dataKirim.data2 = analogRead(A1) / 10.0;
  dataKirim.data3 = analogRead(A2) / 1023;
  //  dataKirim.data1 = 11;
  //  dataKirim.data2 = 11.11;
  //  dataKirim.data3 = 1;
  unsigned long sekarang = millis();
  if (sekarang - sebelum >= 3000) {
    dataKirim.counter++;
    radio.stopListening();
    if (giliran == 0) {
      radio.openWritingPipe(pipa[0]);
      radio.write(&dataKirim, sizeof(dataKirim));
      radio.openReadingPipe(1, pipa[1]);
    }
    else {
```

```
50.       radio.openWritingPipe(pipa[2]);
51.       radio.write(&dataKirim, sizeof(dataKirim));
52.       radio.openReadingPipe(1, pipa[3]);
53.     }
54.     radio.startListening();
55.     sebelum = sekarang;
56.   }
57.   if (radio.available()) {
58.     Serial.print(giliran + 1);
59.     Serial.print(',');
60.     if (giliran == 0) {
61.       radio.read( &dataTerima1, sizeof(dataTerima1));
62.       Serial.print(dataTerima1.counter);
63.       Serial.print(',');
64.       Serial.print(dataTerima1.data1);
65.       Serial.print(',');
66.       Serial.print(dataTerima1.data2);
67.       Serial.print(',');
69.       Serial.println(dataTerima1.data3);
69.       giliran = 1;
70.     }
71.     else {
72.       radio.read( &dataTerima2, sizeof(dataTerima2));
73.       Serial.print(dataTerima2.counter);
74.       Serial.print(',');
75.       Serial.print(dataTerima2.data1);
76.       Serial.print(',');
77.       Serial.print(dataTerima2.data2);
78.       Serial.print(',');
79.       Serial.println(dataTerima2.data3);
80.       giliran = 0;
81.     }
82.   }
83. }
```

3. Upload kode program berikut ini untuk Arduino di rangkaian Slave1:

Program_7_6.ino

```
1.  #include "SPI.h"
2.  #include "RF24.h"
3.  RF24 radio(A3, 10);
4.  byte pipa[][6] = {"pipa1", "pipa2", "pipa3", "pipa4"};
5.  struct dataStruct {
6.    int counter;
7.    int data1;
8.    float data2;
9.    bool data3;
10. } dataKirim;
11. struct dataStruct1 {
```

```
12.   int counter;
13.   int data1;
14.   float data2;
15.   bool data3;
16. } dataTerima;
17. void setup() {
18.   Serial.begin(9600);
19.   radio.begin();
20.   radio.setChannel(117);
21.   radio.setDataRate(RF24_250KBPS);
22.   radio.setPALevel(RF24_PA_LOW);
23.   radio.openWritingPipe(pipa[1]);
24.   radio.openReadingPipe(1, pipa[0]);
25.   radio.startListening();
26. }
27. void loop() {
28.   dataKirim.data1 = analogRead(A0);
29.   dataKirim.data2 = analogRead(A1) / 10.0;
30.   dataKirim.data3 = analogRead(A2) / 1023;
31.   //  dataKirim.data1 = 22;
32.   //  dataKirim.data2 = 22.22;
33.   //  dataKirim.data3 = 0;
34.   if (radio.available()) {
35.     radio.read(&dataTerima, sizeof(dataTerima));
36.     Serial.print(dataTerima.counter);
37.     Serial.print(',');
38.     Serial.print(dataTerima.data1);
39.     Serial.print(',');
40.     Serial.print(dataTerima.data2);
41.     Serial.print(',');
42.     Serial.println(dataTerima.data3);
43.     radio.stopListening();
44.     dataKirim.counter++;
45.     radio.write(&dataKirim, sizeof(dataKirim));
46.     radio.startListening();
47.   }
48. }
```

4. Upload kode program berikut ini untuk Arduino di rangkaian Slave2:

Program_7_7.ino

```
1.  #include "SPI.h"
2.  #include "RF24.h"
3.  RF24 radio(A3, 10);
4.  byte pipa[][6] = {"pipa1", "pipa2", "pipa3", "pipa4"};
5.  struct dataStruct {
6.    int counter;
7.    int data1;
8.    float data2;
9.    bool data3;
```

```
} dataKirim;
struct dataStruct1 {
  int counter;
  int data1;
  float data2;
  bool data3;
} dataTerima;
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openWritingPipe(pipa[3]);
  radio.openReadingPipe(1, pipa[2]);
  radio.startListening();
}
void loop() {
  dataKirim.data1 = analogRead(A0);
  dataKirim.data2 = analogRead(A1) / 10.0;
  dataKirim.data3 = analogRead(A2) / 1023;
  //  dataKirim.data1 = 33;
  //  dataKirim.data2 = 33.33;
  //  dataKirim.data3 = 1;
  if (radio.available()) {
    radio.read(&dataTerima, sizeof(dataTerima));
    Serial.print(dataTerima.counter);
    Serial.print(',');
    Serial.print(dataTerima.data1);
    Serial.print(',');
    Serial.print(dataTerima.data2);
    Serial.print(',');
    Serial.println(dataTerima.data3);
    radio.stopListening();
    dataKirim.counter++;
    radio.write(&dataKirim, sizeof(dataKirim));
    radio.startListening();
  }
}
```

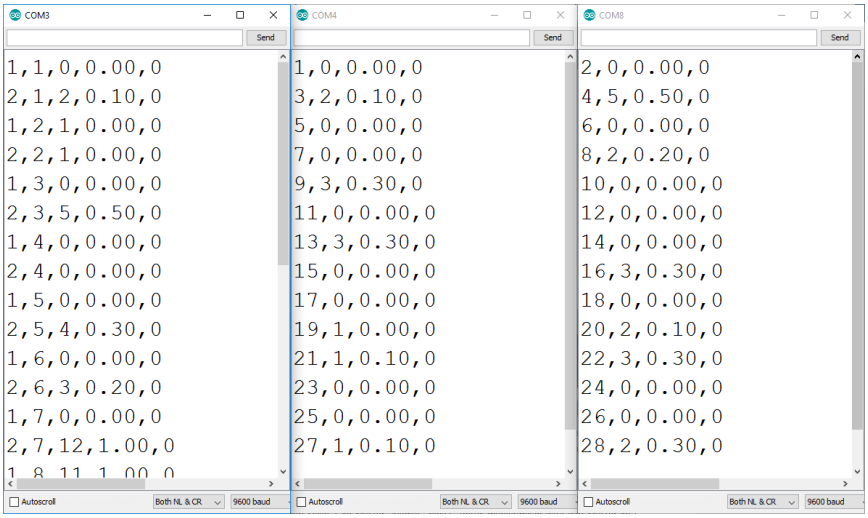
Catatan: Program_7_5.ino membuat rangkaian Master mengirimkan data sinyal analog di kaki A0, A1 dan A2 ke rangkaian Slave 1 dan kemudian dilanjutkan ke rangkaian Slave 2 secara bergantian setiap 3 detik sekali. Program_7_6.ino membuat rangkaian Slave 1 merespon balik kiriman data dari rangkaian Master tersebut dengan mengirimkan data sinyal analog di kaki A0, A1 dan A2 ke rangkaian Master. Begitu pula Program_7_7.ino, membuat rangkaian Slave 2

merespon balik kiriman data dari rangkaian Master dengan mengirimkan data sinyal analog di kaki A0, A1 dan A2 ke rangkaian Master. Pengiriman data dari Master ke Slave 1 dan Slave 2 dibuat bergantian, agar data yang diterima Master dari Slave 1 dan Slave 2 tidak datang secara bersamaan. Sekalipun sebenarnya NRF24L01 mampu untuk menerima sekaligus data yang dikirimkan dari 6 buah NRF24L01 secara bersamaan, namun penulis memilih membuat pengirimannya bergantian untuk menghindari resiko data hilang.

Sebenarnya data yang dikirimkan tidak hanya data sinyal analog pada kaki A0, A1 dan A2 saja, tetapi juga data counter, yang nilainya akan bertambah 1 setiap kali pengiriman data dilakukan. Data counter ini sengaja dimasukkan untuk mengetahui apakah ada data yang hilang dengan melihat urutan nilainya.

Untuk komunikasi 2 arah antara 1 Master dengan 2 Slave tersebut, diperlukan 4 buah alamat, yaitu “pipa1”, “pipa2”, “pipa3” dan “pipa4”. Alamat “pipa1” untuk mengirimkan data dari Master ke Slave 1. Alamat “pipa2” untuk mengirimkan data dari Slave 1 ke Master. Alamat “pipa3” untuk mengirimkan data dari Master ke Slave 2. Alamat “pipa4” untuk mengirimkan data dari Slave 2 ke Master.

5. Setelah upload selesai, tampilkan Serial Monitor untuk Master, Slave 1 dan Slave 2, seperti ditunjukkan pada Gambar 7.13 berikut ini.



Gambar 7.13 Komunikasi Multi-Nodes dengan 1 Master dan 2 Slave

Catatan: Perhatikan tampilan data di Serial Monitor untuk Master, Slave 1 dan Slave 2 pada Gambar 7.13 di atas. Tampak ada 5 data di Master, 4 data di Slave 1 dan 4 data di Slave 2. Berikut ini perinciannnya:

Data di Master:

- Nomor Slave
- Counter dari Slave
- Data sinyal analog pertama dari Slave (tipe data integer)
- Data sinyal analog kedua dari Slave (tipe data float)
- Data sinyal analog ketiga dari Slave (tipe data Boolean)

Data di Slave 1 dan Slave 2:

- Counter dari Master
- Data sinyal analog pertama dari Master (tipe data integer)
- Data sinyal analog kedua dari Master (tipe data float)
- Data sinyal analog ketiga dari Master (tipe data Boolean)

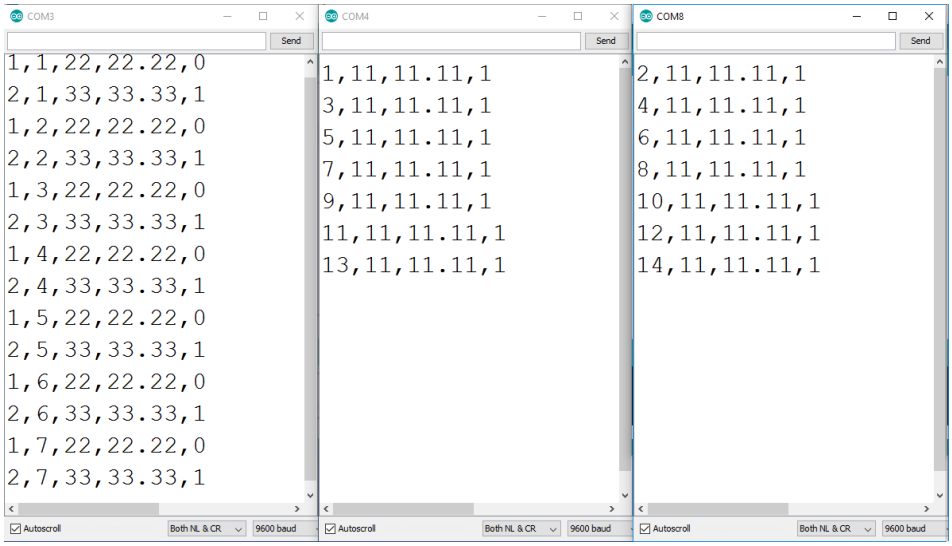
Contoh untuk data di Master di baris pertama tertulis: 2, 7, 10, 1.30, 1. Ini berarti data yang diterima dikirimkan oleh Slave 2, dengan urutan pengiriman ke 7, dengan nilai analog dari Slave 2 berturut-turut 10, 1.30 dan 1.

Contoh untuk data di Slave 1 di baris pertama, tertulis: 2, 1, 0.20, 0. Ini berarti data yang diterima dikirimkan oleh Master pada urutan pengiriman ke 2, dengan nilai analog dari Master berturut-turut 1, 0.20 dan 0.

Contoh untuk data di Slave 2 di baris pertama, tertulis: 3, 17, 1.70, 0. Ini berarti data yang diterima dikirimkan oleh Master pada urutan pengiriman ke 3, dengan nilai analog dari Master berturut-turut 17, 1.70 dan 0.

6. Karena data ketiga sinyal analog bersifat acak (kaki A0, A1 dan A2 mengambang, tidak terhubung dengan sumber sinyal), maka sebenarnya sulit untuk mengetahui apakah data analog tersebut dari Master, ataukah dari Slave 1 atau dari Slave 2. Agar lebih pasti, maka pembaca dapat mengganti data ketiga sinyal analog tersebut dengan angka konstan. Untuk Program_7_5.ino, pembaca dapat menghilangkan tanda komentar (//) pada baris 37-39. Untuk Program_7_6.ino dan Program_7_7.ino, pembaca dapat menghilangkan tanda komentar pada baris 31-33.

7. Setelah tanda komentar dihilangkan, upload kembali program pada rangkaian Master, Slave 1 dan Slave 2. Kemudian tampilkan Serial Monitor untuk Master, Slave 1 dan Slave 2, seperti Gambar 7.14 berikut ini.



Gambar 7.14 Serial Monitor di COM3, COM4 dan COM8 berturut-turut menampilkan data yang diterima rangkaian Master, Slave 1 dan Slave 2.

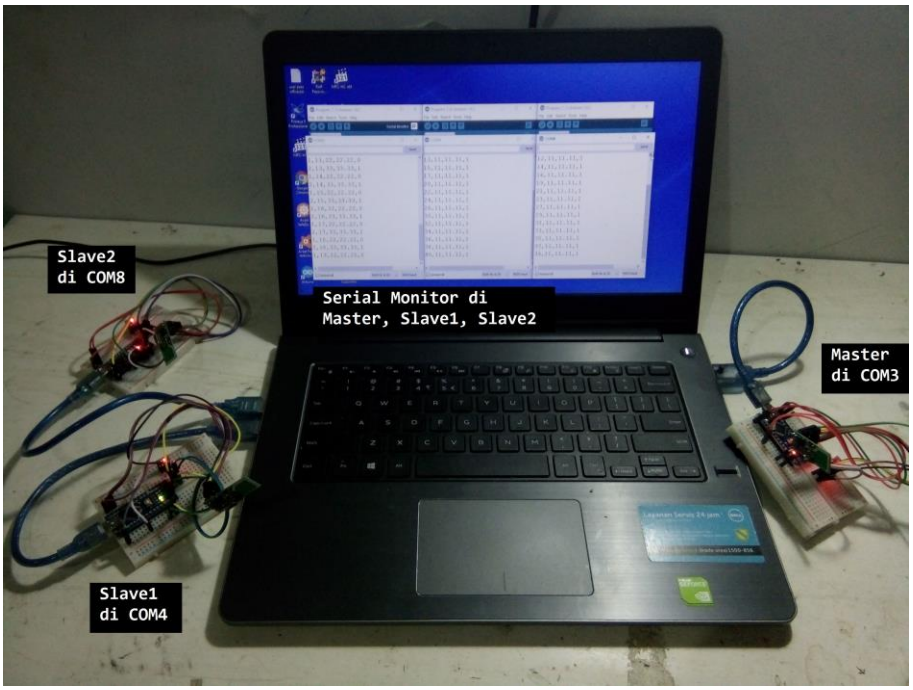
Catatan: Perhatikan data Serial Monitor COM 3 (Master) di baris terakhir, tertulis: 2, 7, 33, 33.33, 1. Ini berarti data yang diterima dikirimkan oleh Slave 2, pada pengiriman ke-7, dengan nilai konstan dari Slave 2 adalah 33, 33.33 dan 1.

Berikutnya data Serial Monitor COM4 (Slave 1) di baris terakhir tertulis: 13, 11, 11.11, 1. Ini berarti data yang diterima dikirimkan oleh Master, pada pengiriman ke-13, dengan nilai konstan dari Master adalah 11, 11.11 dan 1.

Berikutnya data Serial Monitor COM8 (Slave 2) di baris terakhir tertulis: 14, 11, 11.11, 1. Ini berarti data yang diterima dikirimkan oleh Master, pada pengiriman ke-14, dengan nilai konstan dari Master adalah 11, 11.11 dan 1.

Tampak dari Gambar 7.14 di atas, dengan mengganti data 3 buah nilai analog menjadi 3 buah nilai konstan, memudahkan untuk mengatahui data tersebut berasal. Nilai konstan 11, 11.11, 1 berasal dari Master, nilai konstan 22, 22.22, 0 berasal dari Slave 1, dan nilai konstan 33, 33.33, 1 berasal dari Slave 2.

Dengan demikian diketahui pengiriman data dari Master ke Slave 1 dan Slave 2 berjalan dengan baik, tanpa ada data yang hilang, terbukti dengan nomor pengiriman yang berurutan. Begitu pula respon balik dari Slave 1 dan Slave 2 ke Master, juga berjalan baik, terbukti dengan nomor pengiriman yang berurutan.



Gambar 7.15 Komunikasi Multi-Nodes menggunakan 3 rangkaian sesuai Gambar 7.10

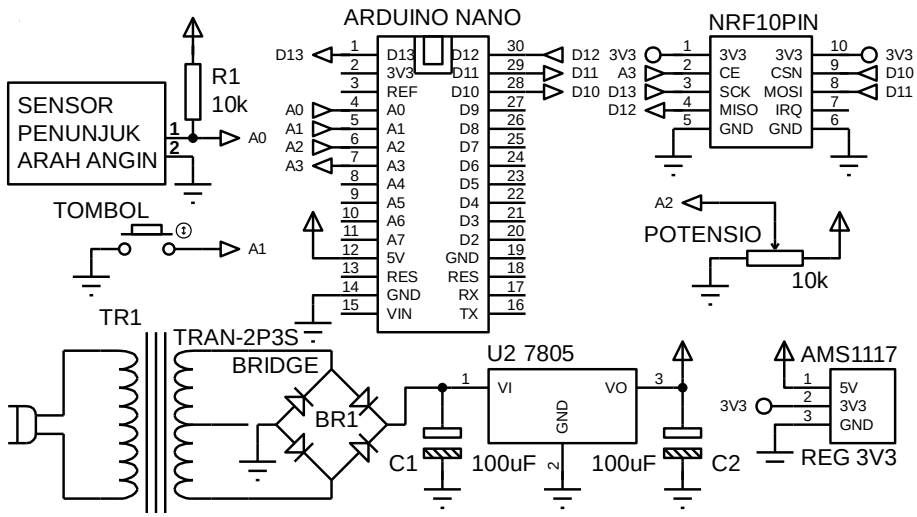
7.4 Implementasi Komunikasi Nirkabel

Setelah pembaca dapat memprogram Arduino untuk komunikasi menggunakan NRF24L01, baik secara 1 arah, 2 arah hingga Multi-nodes, maka langkah berikutnya adalah mengimplementasikan program komunikasi tersebut untuk keperluan pemantauan dan pengendalian secara nirkabel pada rangkaian 3 sensor dan 3 aktuator yang telah dibuat sebelumnya di Bab 6. Secara bertahap, berikut ini implementasi pemantauan dan pengendalian nirkabel untuk rangkaian 3 sensor 3 aktuator, mulai dari komunikasi 1 arah, 2 arah hingga Multi-Nodes.

7.4.1 Implementasi Komunikasi 1 Arah

Dari contoh komunikasi 1 arah yang diuraikan di Sub Bab 7.3.1, maka berikut penerapannya untuk pengiriman data dari satu rangkaian dengan 3 sensor, ke rangkaian yang lain dengan 3 aktuator. Berikut langkah-langkahnya:

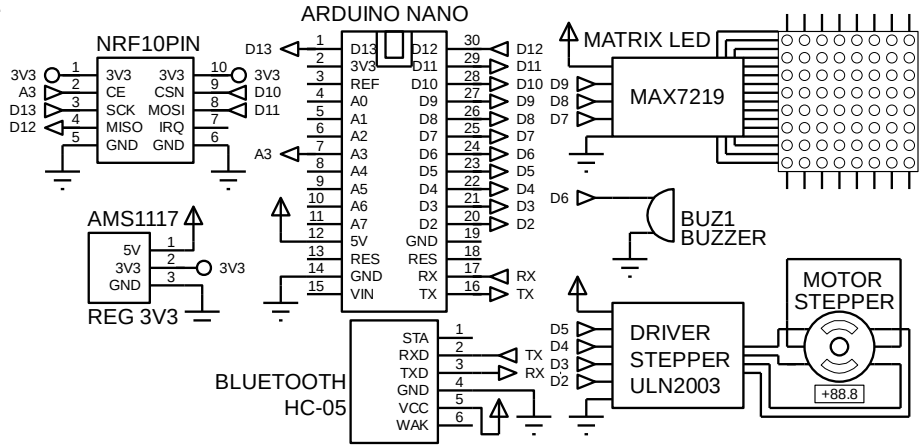
1. Pisahkan rangkaian 3 Sensor dan 3 Aktuator pada Gambar 7.1 menjadi 2 rangkaian. Rangkaian pertama disebut rangkaian pemancar dan rangkaian kedua disebut rangkaian penerima. Rangkaian pemancar terdiri dari Arduino Nano, 3 Sensor (Sensor Angin, Tombol dan Potensio), IC Regulator 3,3V AMS1117, NRF24L01 dan Adaptor 5V (Trafo, Diode Bridge, Kapasitor 100 uF dan IC Regulator 5V 7805) seperti ditunjukkan pada gambar skematik Gambar 7.16 berikut ini.



Gambar 7.16 Gambar skematik rangkaian pemancar dengan 3 Sensor dan suplai dayanya

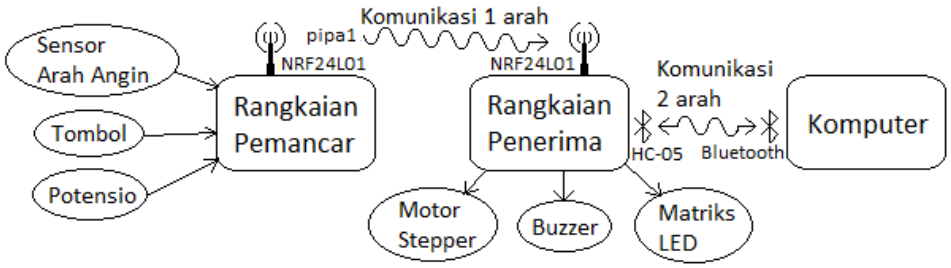
Catatan: Apabila pada contoh di Sub Bab 7.3.1, masalah suplai daya tidak dibahas, karena baik rangkaian pemancar maupun rangkaian penerima mendapat suplai dari Komputer. Namun di sini, masalah suplai daya perlu diperhatikan, karena baik rangkaian pemancar maupun rangkaian penerima tidak terhubung langsung dengan Komputer. Untuk itu, perhatikan bahwa rangkaian pemancar pada Gambar 7.16 menyediakan suplai tegangan 3,3V untuk NRF 24L01 yang disusun dari Adaptor 5V dan IC Regulator 3,3V AMS1117.

2. Rangkaian penerima terdiri dari Arduino Nano, 3 Aktuator (Motor Stepper dengan ULN2003, Buzzer, Matriks LED dengan MAX7219), Regulator 3,3V AMS1117, NRF24L01 dan Bluetooth HC-05.



Gambar 7.17 Gambar skematik rangkaian penerima dengan 3 Aktuator

Catatan: Berbeda dengan rangkaian pemancar, rangkaian penerima di sini tidak menggunakan Adaptor 5V. Sebagai gantinya, pada rangkaian penerima di sini menggunakan Power Bank HP yang juga dapat memberikan suplai tegangan 5V. Gunakan kabel mini USB untuk mengalirkan daya dari Power Bank HP ke Arduino Nano. Untuk suplai daya ke IC Regulator 3,3V AMS1117, hubungkan kaki VCC-nya ke kaki 5V Arduino Nano, dan kaki GND ke kaki GND Arduino.



Gambar 7.18 Komunikasi nirkabel 1 arah (NRF24L01) antara pemancar dengan penerima, dan komunikasi nirkabel 2 arah (Bluetooth) antara penerima dengan komputer

3. Dari contoh di Sub Bab 7.3.1 diketahui bahwa komunikasi antara 2 buah NRF24L01 dapat diibaratkan seperti saluran pipa air 1 arah. Untuk komunikasi 1 arah, hanya diperlukan 1 pipa seperti ditunjukkan pada Gambar 7.18 di atas. Untuk komunikasi 2 arah, diperlukan 2 buah pipa, seperti terlihat pada Gambar 7.22.
4. Dari diagram Gambar 7.18, berikut urutan kerjanya:
   - Rangkaian Pemancar akan mengirimkan data Sensor Arah Angin, Tombol dan Potensio setiap 2 detik ke Rangkaian Penerima melalui komunikasi nirkabel NRF24L01.
   - Rangkaian Penerima akan meneruskan data ketiga Sensor tersebut ke Komputer melalui komunikasi nirkabel Bluetooth.
   - Komputer akan menampilkan data ketiga Sensor dan mengirimkan data untuk ketiga Aktuator ke Rangkaian Penerima melalui komunikasi nirkabel Bluetooth.
5. Berikut program untuk Rangkaian Pemancar agar dapat mengirimkan data Sensor Arah Angin, Tombol dan Potensio setiap detik ke Rangkaian Penerima melalui komunikasi nirkabel NRF24L01.

Program_7_8.ino

```
#include <SPI.h>
#include <RF24.h>
RF24 radio(A3, 10);
byte pipa[][6] = {"pipa1"};
unsigned long sebelum = 0;
struct dataStruct {
  int sudut;
  int tombol;
  int potensio;
} dataKu;
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openWritingPipe(pipa[0]);
  radio.stopListening();
}
void loop() {
```

```
  int a = analogRead(A0);
  if (a < 459.5) dataKu.sudut = 1; //1 = UTR
  if (a >= 459.5 && a < 494) dataKu.sudut = 3; //3 = TMR
  if (a >= 494 && a < 564.5) dataKu.sudut = 2; //2 = TLU
  if (a >= 564.5 && a < 636.5) dataKu.sudut = 7; //7 = BRT
  if (a >= 636.5 && a < 692) dataKu.sudut = 5; //5 = SLT
  if (a >= 692 && a < 747) dataKu.sudut = 6; //6 = BDY
  if (a >= 747 && a < 806.5) dataKu.sudut = 8; // 8 = BLU
  if (a >= 806.5) dataKu.sudut = 4; // 4 = TGR
  dataKu.tombol = !digitalRead(A1);
  dataKu.potensio = analogRead(A2);
  unsigned long sekarang = millis();
  if (sekarang - sebelum >= 2000) {
    radio.write(&dataKu, sizeof(dataKu));
    sebelum = sekarang;
    Serial.print(dataKu.sudut);
    Serial.print(',');
    Serial.print(dataKu.tombol);
    Serial.print(',');
    Serial.println(dataKu.potensio);
  }
}
```

6. Berikut program untuk Rangkaian Penerima agar dapat menerima data dari Rangkaian Pemancar melalui komunikasi nirkabel NRF24L01, dan meneruskan datanya ke Komputer melalui komunikasi Bluetooth.

Program_7_9.ino

```
#include "SPI.h"
#include "RF24.h"
RF24 radio(A3, 10);
byte pipa[][6] = {"pipa1"};
struct dataStruct {
  int sudut;
  int tombol;
  int potensio;
} dataKu;
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openReadingPipe(1, pipa[0]);
  radio.startListening();
}
void loop()  {
  if (radio.available()) {
```

```
21.     radio.read( &dataKu, sizeof(dataKu) );
22.     switch (dataKu.sudut) {
23.       case 1: Serial.print("UTR"); break;
24.       case 2: Serial.print("TLU"); break;
25.       case 3: Serial.print("TMR"); break;
26.       case 4: Serial.print("TGR"); break;
27.       case 5: Serial.print("SLT"); break;
28.       case 6: Serial.print("BDY"); break;
29.       case 7: Serial.print("BRT"); break;
30.       case 8: Serial.print("BLU"); break;
31.     }
32.     Serial.print(',');
33.     Serial.print(dataKu.tombol);
34.     Serial.print(',');
35.     Serial.println(dataKu.potensio);
36.   }
37. }
```

7. Berikut tambahan program untuk Rangkaian Penerima agar dapat menerima data Aktuator dari Komputer melalui komunikasi Bluetooth dan menerjemahkannya ke Motor Stepper, Buzzer dan Matriks LED.

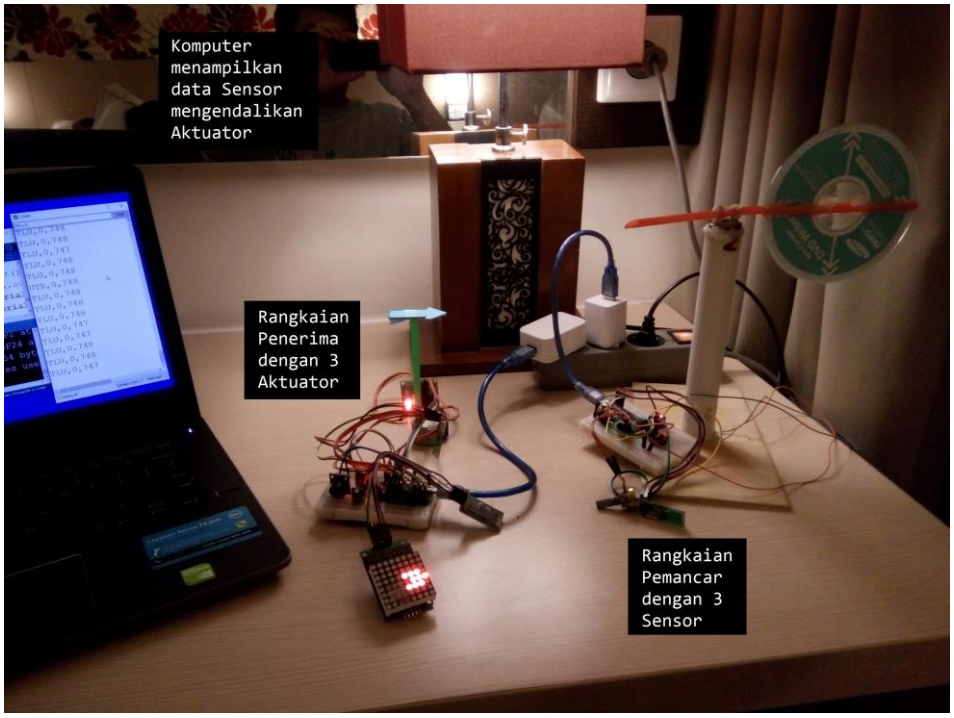
Program_7_10.ino

```
1.  #include "Stepper.h"
2.  #include "LedControl.h"
3.  #include "SPI.h"
4.  #include "RF24.h"
5.  #define do1  1047
6.  #define re  1175
7.  #define mi  1319
8.  #define fa  1397
9.  #define sol  1568
10. #define la  1760
11. #define si  1976
12. #define do2  2093
13. RF24 radio(A3, 10);
14. const int stepsPerRevolution = 32;
15. Stepper myStepper(stepsPerRevolution, 2, 4, 3, 5);
16. LedControl lc = LedControl(7, 9, 8, 1);
17. byte pipa[][6] = {"pipa1"};
18. struct dataStruct {
19.   int sudut;
20.   int tombol;
21.   int potensio;
22. } dataKu;
23. int nada[] = {do1, re, mi, fa, sol, la, si, do2};
24. int e1 = 0;
25. byte pola[8][8] = {
```

```
{B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},// UTR
{B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},// TLU
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},// TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},// TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},// SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},// BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},// BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} // BLU
};
unsigned long sebelum = 0;
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  myStepper.setSpeed(500);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openReadingPipe(1, pipa[0]);
  radio.startListening();
}
void loop()  {
  if (radio.available()) {
    radio.read( &dataKu, sizeof(dataKu));
    switch (dataKu.sudut) {
      case 1: Serial.print("UTR"); break;
      case 2: Serial.print("TLU"); break;
      case 3: Serial.print("TMR"); break;
      case 4: Serial.print("TGR"); break;
      case 5: Serial.print("SLT"); break;
      case 6: Serial.print("BDY"); break;
      case 7: Serial.print("BRT"); break;
      case 8: Serial.print("BLU"); break;
    }
    Serial.print(',');
    Serial.print(dataKu.tombol);
    Serial.print(',');
    Serial.println(dataKu.potensio);
    noTone(6);
  }
}
void serialEvent() {
  while (Serial.available()) {
    int e = Serial.parseInt() + 1; //posisi Motor
    int f = Serial.parseInt(); //nada Buzzer
    int g = Serial.parseInt(); //pola Matriks
    if (Serial.read() == char(13)) {
      for (int i = 0; i < 8; i++) {
```

```
77.           lc.setRow(0, i, pola[g - 1][7 - i]);
78.         }
79.         if (e1 - e > 180) e = e + 360;
80.         if (e - e1 > 180) e1 = e1 + 360;
81.         tone(6, nada[f - 1], (e1 - e) * 100);
82.         myStepper.step((e - e1) * 5.69);
83.         e1 = e;
84.       }
85.     }
86. }
```

8. Gambar 7.19 berikut menunjukkan hasil program komunikasi 1 arah NRF24L01. Tampak rangkaian pemancar dan rangkaian penerima saling terpisah, dan keduanya juga terpisah dengan Komputer. Sekalipun ketiganya terpisah secara fisik, namun Komputer dapat menampilkan data ketiga Sensor dan mengendalikan ketiga Aktuator.

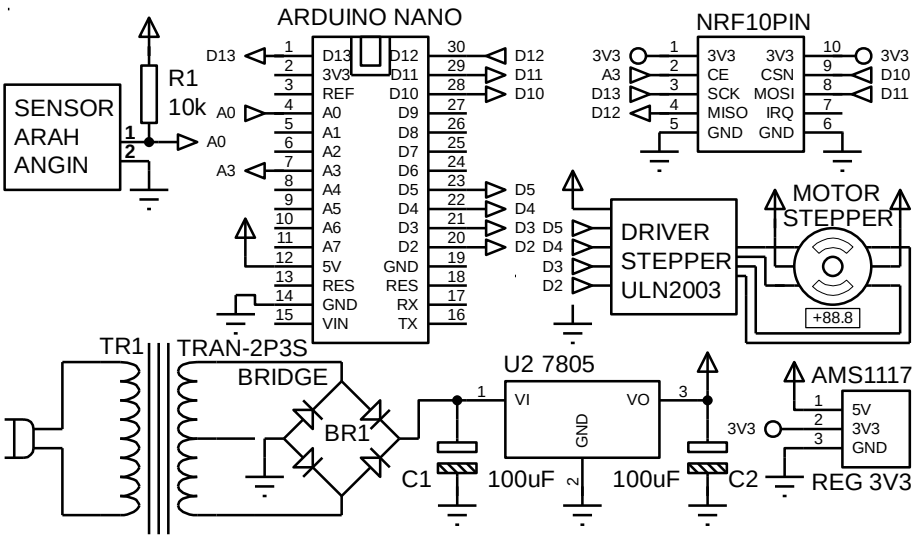


Gambar 7.19 Implementasi Komunikasi 1 Arah NRF24L01 pada rangkaian pemancar dengan 3 sensor dan rangkaian penerima dengan 3 aktuator yang dipantau dan dikendalikan komputer

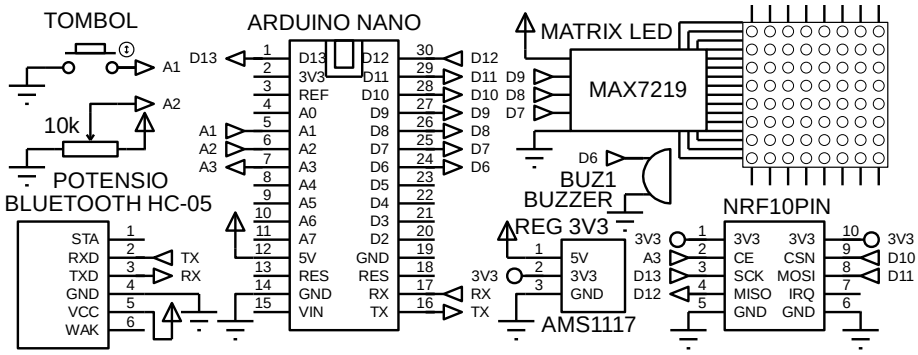
7.4.2 Implementasi Komunikasi 2 Arah

Dalam kasus pengukuran efisiensi alat penyuling air yang sedang penulis kerjakan, Motor Stepper harus ditempatkan di luar ruangan karena diperlukan untuk memutar alat penyuling hingga tepat menghadap sinar matahari. Selain itu, untuk mengetahui arah angin, tentunya Sensor Arah Angin juga harus ditempatkan di luar ruangan. Sedangkan Tombol, Potensio, Buzzer dan Matriks LED harus ditempatkan di dalam ruangan, yang digunakan untuk keperluan pengaturan manual dan indikator oleh operator. Komunikasi antara rangkaian di luar dengan rangkaian di dalam harus 2 arah, karena rangkaian di luar perlu mengirimkan data Sensor Arah Angin ke rangkaian yang di dalam, dan sebaliknya rangkaian yang di dalam perlu mengirimkan data Motor Stepper ke rangkaian di luar. Untuk itu perlu mengimplementasikan komunikasi 2 arah antara rangkaian luar dengan rangkaian dalam. Untuk memudahkan penamaan, maka rangkaian luar disebut rangkaian Slave, dan rangkaian dalam disebut rangkaian Master. Berikut langkah-langkah implementasinya:

1. Gambar 7.20 dan 7.21 berikut ini berturut-turut menunjukkan gambar skematik rangkaian Slave (di luar) dan rangkaian Master (di dalam). Susun kedua rangkaian tersebut.

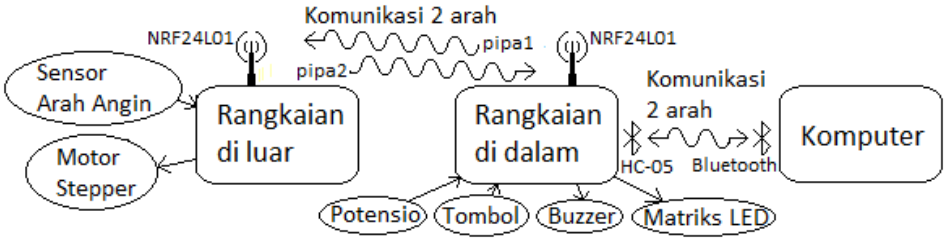


Gambar 7.20 Gambar skematik rangkaian Slave (Sensor Arah Angin dan Motor Stepper)



Gambar 7.21 Gambar skematik rangkaian Master (Tombol, Potensio, Buzzer, Matriks LED)

2. Rangkaian Master yang berada di dalam ruangan akan memulai komunikasi dengan mengirimkan data ke rangkaian Slave. Rangkaian Slave yang berada di luar ruangan akan merespon balik dengan mengirimkan data ke rangkaian Master. Hubungan antara rangkaian Master dengan Slave dan Master dengan Komputer dapat dilihat pada gambar berikut ini.



Gambar 7.22 Komunikasi Nirkabel 2 Arah menggunakan NRF24L01 antara Master dengan Slave dan komunikasi 2 arah antara Master dengan Komputer menggunakan Bluetooth.

3. Berdasarkan diagram hubungan antara rangkaian Master, Slave dan Komputer pada Gambar 7.22 di atas, maka berikut ini urutan kerjanya:
   - Master akan mengirimkan data posisi sudut Motor Stepper ke Slave setiap 5 detik. Mula-mula data posisi sudut Motor Stepper adalah 0. Data posisi sudut Motor Stepper ini diatur di Komputer. Ketika Komputer mengirimkan data posisi sudut Motor Stepper, maka data di Master akan berubah mengikuti data dari Komputer tersebut. Data tersebut akan bertahan hingga Komputer kembali mengirimkan data

posisi sudut Motor Stepper yang baru. Tidak hanya data posisi sudut Motor Stepper, bersama dengan itu Komputer juga mengirimkan data Buzzer dan Matriks LED. Namun kedua data ini tidak dikirimkan ke Slave, karena kedua komponen tersebut terhubung dengan Master.

- Ketika Slave menerima data posisi sudut Motor Stepper setiap 5 detik dari Master, Slave akan langsung merespon dengan mengirimkan data Sensor Arah Angin ke Master.
- Setelah Master menerima data Sensor Arah Angin, data tersebut diteruskan ke Komputer bersama dengan data Tombol dan Potensio yang terhubung dengan Master. Komputer akan menerima dan menampilkan data Sensor Arah Angin, Tombol dan Potensio tersebut.

4. Berikut program untuk membuat Master menerima data posisi sudut Motor Stepper, Buzzer dan Matriks LED dari Komputer, dan mengirimkan data posisi sudut Motor Stepper ke Slave setiap 5 detik, dan mengatur Buzzer dan Matriks LED berdasarkan data yang diterima tersebut.

Program_7_11.ino

```
1.  #include "LedControl.h"
2.  #include "SPI.h"
3.  #include "RF24.h"
4.  #define do1  1047
5.  #define re  1175
6.  #define mi  1319
7.  #define fa  1397
8.  #define sol  1568
9.  #define la  1760
10. #define si  1976
11. #define do2  2093
12. RF24 radio(A3, 10);
13. LedControl lc = LedControl(7, 9, 8, 1);
14. byte pipa[][6] = {"pipa1", "pipa2"};
15. struct dataStruct {
16.   int stepper;
17. } dataKirim;
18. struct dataStruct1 {
19.   int sudut;
20. } dataTerima;
21. unsigned long sebelum = 0;
22. int nada[] = {do1, re, mi, fa, sol, la, si, do2};
23. byte pola[8][8] = {
24. {B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},// UTR
25. {B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},// TLU
```

```
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},// TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},// TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},// SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},// BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},// BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} // BLU
};
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openWritingPipe(pipa[0]);
  radio.stopListening();
}
void loop()  {
  int tombol = !digitalRead(A1);
  int potensio = analogRead(A2);
  unsigned long sekarang = millis();
  if (sekarang - sebelum >= 5000) {
    sebelum = sekarang;
    radio.write(&dataKirim, sizeof(dataKirim));
    noTone(6);
  }
}
void serialEvent() {
  while (Serial.available()) {
    int e = Serial.parseInt() + 1; //posisi Motor
    int f = Serial.parseInt(); //nada Buzzer
    int g = Serial.parseInt(); //pola Matriks
    if (Serial.read() == char(13)) {
      dataKirim.stepper = e;
      for (int i = 0; i < 8; i++) {
        lc.setRow(0, i, pola[g - 1][7 - i]);
      }
      tone(6, nada[f - 1], 1000);
    }
  }
}
```

5. Berikut program untuk Slave merespon data posisi sudut Motor Stepper yang diterima dari Master. Begitu data posisi sudut Motor Stepper diterima, Slave akan mengirimkan data Sensor Arah Angin ke Master.

Program_7_12.ino

```
#include "Stepper.h"
#include "SPI.h"
#include "RF24.h"
RF24 radio(A3, 10);
const int stepsPerRevolution = 32;
Stepper myStepper(stepsPerRevolution, 2, 4, 3, 5);
byte alamat[][6] = {"pipa1", "pipa2"};
struct dataStruct {
  int stepper;
} dataTerima;
struct dataStruct1 {
  int sudut;
} dataKirim;
int e = 0;
int e1 = 0;
void setup() {
  Serial.begin(9600);
  myStepper.setSpeed(500);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openWritingPipe(alamat[1]);
  radio.openReadingPipe(1, alamat[0]);
  radio.startListening();
}
void loop() {
  int a = analogRead(A0);
  if (a < 459.5) dataKirim.sudut = 1; //1 = UTR
  if (a >= 459.5 && a < 494) dataKirim.sudut = 3; //3 = TMR
  if (a >= 494 && a < 564.5) dataKirim.sudut = 2; //2 = TLU
  if (a >= 564.5 && a < 636.5) dataKirim.sudut = 7; //7 = BRT
  if (a >= 636.5 && a < 692) dataKirim.sudut = 5; //5 = SLT
  if (a >= 692 && a < 747) dataKirim.sudut = 6; //6 = BDY
  if (a >= 747 && a < 806.5) dataKirim.sudut = 8; // 8 = BLU
  if (a >= 806.5) dataKirim.sudut = 4; // 4 = TGR
  if (radio.available()) {
    radio.read(&dataTerima, sizeof(dataTerima));
    Serial.println(dataTerima.stepper);
    e = dataTerima.stepper;
    if (e1 - e > 180) e = e + 360;
    if (e - e1 > 180) e1 = e1 + 360;
    myStepper.step((e - e1) * 5.69);
    e1 = e;
    radio.stopListening();
    radio.write(&dataKirim, sizeof(dataKirim));
    radio.startListening();
  }
}
```

6. Berikut program tambahan untuk Master menerima data Sensor Arah Angin dari Slave, dan meneruskan data tersebut ke Komputer, bersama dengan data Tombol dan Potensio yang terhubung dengan Master.

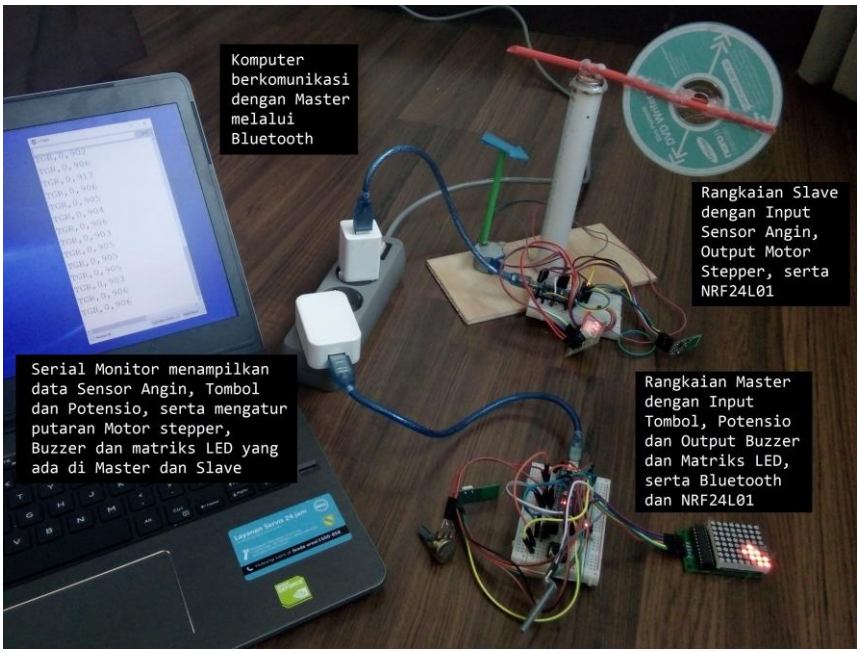
Program_7_13.ino

```
#include "LedControl.h"
#include "SPI.h"
#include "RF24.h"
#define do1  1047
#define re  1175
#define mi  1319
#define fa  1397
#define sol  1568
#define la  1760
#define si  1976
#define do2  2093
RF24 radio(A3, 10);
LedControl lc = LedControl(7, 9, 8, 1);
byte pipa[][6] = {"pipa1", "pipa2"};
struct dataStruct {
  int stepper;
} dataKirim;
struct dataStruct1 {
  int sudut;
} dataTerima;
unsigned long sebelum = 0;
int nada[] = {do1, re, mi, fa, sol, la, si, do2};
byte pola[8][8] = {
{B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},// UTR
{B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},// TLU
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},// TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},// TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},// SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},// BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},// BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} // BLU
};
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openWritingPipe(pipa[0]);
  radio.openReadingPipe(1, pipa[1]);
```

```
45.   radio.stopListening();
46. }
47. void loop()  {
48.   int tombol = !digitalRead(A1);
49.   int potensio = analogRead(A2);
50.   unsigned long sekarang = millis();
51.   if (sekarang - sebelum >= 5000) {
52.     sebelum = sekarang;
53.     radio.stopListening();
54.     radio.write(&dataKirim, sizeof(dataKirim));
55.     radio.startListening();
56.   }
57.   if (radio.available()) {
58.     radio.read( &dataTerima, sizeof(dataTerima) );
59.     switch (dataTerima.sudut) {
60.       case 1: Serial.print("UTR"); break;
61.       case 2: Serial.print("TLU"); break;
62.       case 3: Serial.print("TMR"); break;
63.       case 4: Serial.print("TGR"); break;
64.       case 5: Serial.print("SLT"); break;
65.       case 6: Serial.print("BDY"); break;
66.       case 7: Serial.print("BRT"); break;
67.       case 8: Serial.print("BLU"); break;
68.     }
69.     Serial.print(',');
70.     Serial.print(tombol);
71.     Serial.print(',');
72.     Serial.println(potensio);
73.     noTone(6);
74.   }
75. }
76. void serialEvent() {
77.   while (Serial.available()) {
78.     int e = Serial.parseInt() + 1; //posisi Motor
79.     int f = Serial.parseInt(); //nada Buzzer
80.     int g = Serial.parseInt(); //pola Matriks
81.     if (Serial.read() == char(13)) {
82.       dataKirim.stepper = e;
83.       for (int i = 0; i < 8; i++) {
84.         lc.setRow(0, i, pola[g - 1][7 - i]);
85.       }
86.       tone(6, nada[f - 1], 1000);
87.     }
88.   }
89. }
```

7. Gambar 7.23 berikut menunjukkan hasil implementasi program komunikasi 2 arah. Tampak rangkaian Master dan rangkaian Slave

terpisah, dan juga terpisah dengan Komputer. Sekalipun ketiganya terpisah secara fisik, namun Komputer dapat berkomunikasi dengan Master dan Master dapat berkomunikasi dengan Slave.

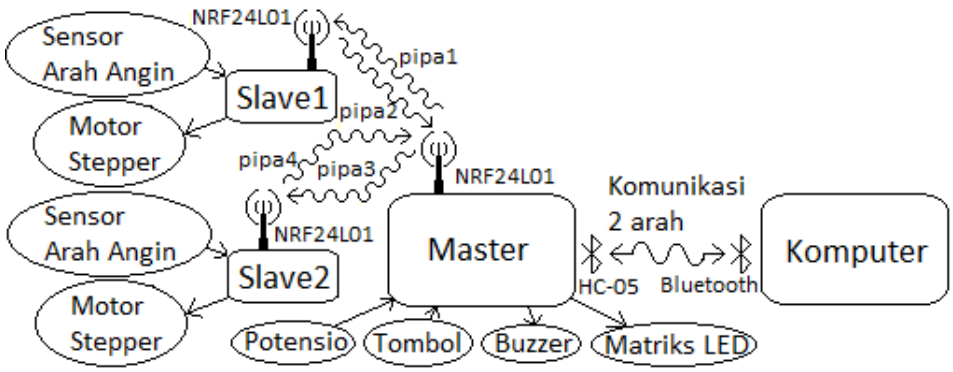


Gambar 7.23 Implementasi program komunikasi 2 arah antara Master, Slave dan Komputer

7.4.3 Implementasi Komunikasi Multi-Nodes

Pada perkembangannya, komunikasi antara dua rangkaian saja tidak cukup. Dibutuhkan lebih dari 2 rangkaian untuk memenuhi kebutuhan di lapangan. Sebagai contoh kasus, diinginkan ada Alat Destilasi yang kedua, yang ditempatkan pada jarak yang cukup jauh dari Alat Destilasi yang pertama. Diinginkan kedua Alat Destilasi tersebut dapat dipantau dan dikendalikan dari Komputer. Untuk memudahkan penamaan, Alat Destilasi yang pertama disebut Slave1, dan Alat Destilasi yang kedua disebut Slave2. Untuk menghubungkan Master, yaitu rangkaian di dalam ruangan dengan kedua Slave yang berada di luar ruangan tersebut, dibutuhkan komunikasi NRF24L01 Multi-Nodes. Berikut langkah-langkah implementasinya:

1. Buat rangkaian Slave1 dan Slave2 mengikuti rangkaian Gambar 7.20 dan rangkaian Master mengikuti rangkaian Gambar 7.21.
2. Hubungan antara Slave1, Slave2 dan Master seperti ditunjukkan pada diagram Gambar 7.24 berikut ini.



Gambar 7.24 Diagram komunikasi 2 arah (NRF24L01) antara Slave1 dan Slave2 dengan Master dan komunikasi 2 arah (Bluetooth) antara Master dengan Komputer

3. Dari diagram Gambar 7.24 di atas, berikut urutan kerjanya:
   - Master akan mengirimkan data posisi sudut Motor Stepper1 ke Slave 1 setiap 2 detik pertama. Kemudian 2 detik berikutnya, Master akan mengirimkan data posisi sudut Motor Stepper2 ke Slave2. Pengiriman ke Slave 1 dan kemudian ke Slave 2 tersebut dilakukan terus-menerus bergantian. Baik data posisi sudut Motor Stepper1 maupun data posisi sudut Motor Stepper2, hanya dapat diatur dari Komputer. Apabila Komputer mengirimkan data posisi sudut yang baru, maka data posisi sudut yang ada di Master akan diganti dengan data posisi sudut yang baru. Data ini akan bertahan hingga pengiriman data berikutnya dari Komputer. Tidak hanya data posisi sudut Motor Stepper1 dan Motor Stepper2 saja, Komputer juga mengirimkan data Buzzer dan Matriks LED ke Master.
   - Baik Slave1 maupun Slave2, akan langsung merespon dengan mengirimkan data Sensor Arah Angin ke Master, setiap kali menerima data posisi sudut Motor Stepper dari Master.

- Setelah Master menerima data Sensor Arah Angin, baik dari Slave1 maupun Slave2, data tersebut diteruskan ke Komputer bersama dengan data Tombol dan Potensio yang terhubung dengan Master. Komputer akan menerima dan menampilkan data Sensor Arah Angin dari Slave1 dan Slave2, serta data Tombol dan Potensio.

4. Berikut program untuk Master menerima data dari Komputer, mengirimkan data posisi sudut Motor Stepper ke Slave1 dan kemudian ke Slave2 secara bergantian setiap 2 detik, dan mengatur Buzzer dan Matriks LED berdasarkan data yang diterima tersebut.

Program_7_14.ino

```
#include "LedControl.h"
#include "SPI.h"
#include "RF24.h"
#define do1  1047
#define re  1175
#define mi  1319
#define fa  1397
#define sol  1568
#define la  1760
#define si  1976
#define do2  2093
RF24 radio(A3, 10);
LedControl lc = LedControl(7, 9, 8, 1);
byte pipa[][6] = {"pipa1", "pipa2", "pipa3", "pipa4"};
struct dataStruct {
  int stepper;
} dataKirim1;
struct dataStruct1 {
  int stepper;
} dataKirim2;
unsigned long sebelum = 0;
int giliran = 0;
int nada[] = {do1, re, mi, fa, sol, la, si, do2};
byte pola[8][8] = {
{B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},// UTR
{B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},// TLU
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},// TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},// TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},// SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},// BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},// BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} // BLU
};
void setup() {
  Serial.begin(9600);
```

```
  pinMode(A1, INPUT_PULLUP);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.stopListening();
}
void loop()  {
  unsigned long sekarang = millis();
  if (sekarang - sebelum >= 2000) {
    if (giliran == 0) {
      radio.openWritingPipe(pipa[0]);
      radio.write(&dataKirim1, sizeof(dataKirim1));
      giliran = 1;
    }
    else {
      radio.openWritingPipe(pipa[2]);
      radio.write(&dataKirim2, sizeof(dataKirim2));
      giliran = 0;
    }
    sebelum = sekarang;
    noTone(6);
  }
}
void serialEvent() {
  while (Serial.available()) {
    int d = Serial.parseInt() + 1; //posisi Motor
    int e = Serial.parseInt() + 1; //posisi Motor
    int f = Serial.parseInt(); //nada Buzzer
    int g = Serial.parseInt(); //pola Matriks
    if (Serial.read() == char(13)) {
      dataKirim1.stepper = d;
      dataKirim2.stepper = e;
      for (int i = 0; i < 8; i++) {
        lc.setRow(0, i, pola[g - 1][7 - i]);
      }
      tone(6, nada[f - 1], 1000);
    }
  }
}
```

5. Berikut program untuk membuat Slave1 merespon data yang diterima dari Master. Begitu data posisi sudut Motor Stepper1 diterima, Slave1 akan mengirimkan data Sensor Arah Angin Slave1 ke Master.

Program_7_15.ino

```
#include "Stepper.h"
#include "SPI.h"
#include "RF24.h"
RF24 radio(A3, 10);
const int stepsPerRevolution = 32;
Stepper myStepper(stepsPerRevolution, 2, 4, 3, 5);
byte pipa[][6] = {"pipa1", "pipa2", "pipa3", "pipa4"};
struct dataStruct {
  int stepper;
} dataTerima;
struct dataStruct1 {
  int sudut;
} dataKirim;
int e = 0;
int e1 = 0;
void setup() {
  Serial.begin(9600);
  myStepper.setSpeed(500);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.openWritingPipe(pipa[1]);
  radio.openReadingPipe(1, pipa[0]);
  radio.startListening();
}
void loop() {
  int a = analogRead(A0);
  if (a < 459.5) dataKirim.sudut = 1; //1 = UTR
  if (a >= 459.5 && a < 494) dataKirim.sudut = 3; //3 = TMR
  if (a >= 494 && a < 564.5) dataKirim.sudut = 2; //2 = TLU
  if (a >= 564.5 && a < 636.5) dataKirim.sudut = 7; //7 = BRT
  if (a >= 636.5 && a < 692) dataKirim.sudut = 5; //5 = SLT
  if (a >= 692 && a < 747) dataKirim.sudut = 6; //6 = BDY
  if (a >= 747 && a < 806.5) dataKirim.sudut = 8; // 8 = BLU
  if (a >= 806.5) dataKirim.sudut = 4; // 4 = TGR
  if (radio.available()) {
    radio.read(&dataTerima, sizeof(dataTerima));
    Serial.println(dataTerima.stepper);
    e = dataTerima.stepper;
    if (e1 - e > 180) e = e + 360;
    if (e - e1 > 180) e1 = e1 + 360;
    myStepper.step((e - e1) * 5.69);
    e1 = e;
    radio.stopListening();
    radio.write(&dataKirim, sizeof(dataKirim));
    radio.startListening();
  }
}
```

6. Berikut program untuk membuat Slave2 merespon data yang diterima dari Master. Program ini sama seperti Program_7_15.ino, hanya berbeda pada alamat pipa yang digunakan, seperti terlihat pada baris 23 – 24.

Program_7_16.ino

```
 1. #include "Stepper.h"
 2. #include "SPI.h"
 3. #include "RF24.h"
 4. RF24 radio(A3, 10);
 5. const int stepsPerRevolution = 32;
 6. Stepper myStepper(stepsPerRevolution, 2, 4, 3, 5);
 7. byte pipa[][6] = {"pipa1", "pipa2", "pipa3", "pipa4"};
 8. struct dataStruct {
 9.   int stepper;
10. } dataTerima;
11. struct dataStruct1 {
12.   int sudut;
13. } dataKirim;
14. int e = 0;
15. int e1 = 0;
16. void setup() {
17.   Serial.begin(9600);
18.   myStepper.setSpeed(500);
19.   radio.begin();
20.   radio.setChannel(117);
21.   radio.setDataRate(RF24_250KBPS);
22.   radio.setPALevel(RF24_PA_LOW);
23.   radio.openWritingPipe(pipa[3]);
24.   radio.openReadingPipe(1, pipa[2]);
25.   radio.startListening();
26. }
27. void loop() {
28.   int a = analogRead(A0);
29.   if (a < 459.5) dataKirim.sudut = 1; //1 = UTR
30.   if (a >= 459.5 && a < 494) dataKirim.sudut = 3; //3 = TMR
31.   if (a >= 494 && a < 564.5) dataKirim.sudut = 2; //2 = TLU
32.   if (a >= 564.5 && a < 636.5) dataKirim.sudut = 7; //7 = BRT
33.   if (a >= 636.5 && a < 692) dataKirim.sudut = 5; //5 = SLT
34.   if (a >= 692 && a < 747) dataKirim.sudut = 6; //6 = BDY
35.   if (a >= 747 && a < 806.5) dataKirim.sudut = 8; // 8 = BLU
36.   if (a >= 806.5) dataKirim.sudut = 4; // 4 = TGR
37.   if (radio.available()) {
38.     radio.read(&dataTerima, sizeof(dataTerima));
39.     Serial.println(dataTerima.stepper);
40.     e = dataTerima.stepper;
41.     if (e1 - e > 180) e = e + 360;
42.     if (e - e1 > 180) e1 = e1 + 360;
43.     myStepper.step((e - e1) * 5.69);
```

```
44.     e1 = e;
45.     radio.stopListening();
46.     radio.write(&dataKirim, sizeof(dataKirim));
47.     radio.startListening();
48.   }
49. }
```

7. Berikut program tambahan untuk Master menerima data Sensor Arah Angin dari Slave1 dan Slave2, dan meneruskan kedua data tersebut ke Komputer, bersama dengan data Tombol dan Potensio.

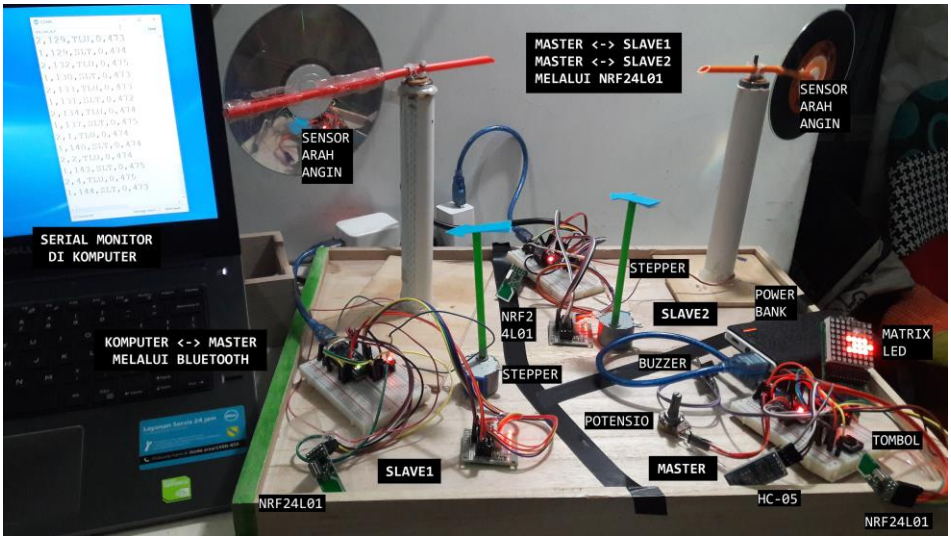
Program_7_17.ino

```
1.  #include "LedControl.h"
2.  #include "SPI.h"
3.  #include "RF24.h"
4.  #define do1  1047
5.  #define re  1175
6.  #define mi  1319
7.  #define fa  1397
8.  #define sol  1568
9.  #define la  1760
10. #define si  1976
11. #define do2  2093
12. RF24 radio(A3, 10);
13. LedControl lc = LedControl(7, 9, 8, 1);
14. byte pipa[][6] = {"pipa1", "pipa2", "pipa3", "pipa4"};
15. struct dataStruct {
16.   int counter;
17.   int stepper;
18. } dataKirim1;
19. struct dataStruct1 {
20.   int counter;
21.   int sudut;
22. } dataTerima1;
23. struct dataStruct2 {
24.   int counter;
25.   int stepper;
26. } dataKirim2;
27. struct dataStruct3 {
28.   int counter;
29.   int sudut;
30. } dataTerima2;
31. unsigned long sebelum = 0;
32. int giliran = 0;
33. int nada[] = {do1, re, mi, fa, sol, la, si, do2};
34. byte pola[8][8] = {
35. {B00000000, B00000000, B00000000, B01110000, B11100000, B01110000, B00000000, B00000000},// UTR
36. {B00000000, B00000000, B00000000, B00000000, B00100000, B01110000, B11100000, B11000000},// TLU
```

```
{B00000000, B00000000, B00000000, B00000000, B00010100, B00011100, B00011100, B00001000},// TMR
{B00000000, B00000000, B00000000, B00000000, B00000100, B00001110, B00000111, B00000011},// TGR
{B00000000, B00000000, B00001110, B00000111, B00001110, B00000000, B00000000, B00000000},// SLT
{B00000011, B00000111, B00001110, B00000100, B00000000, B00000000, B00000000, B00000000},// BDY
{B00010000, B00111000, B00111000, B00111000, B00101000, B00000000, B00000000, B00000000},// BRT
{B11000000, B11100000, B01110000, B00100000, B00000000, B00000000, B00000000, B00000000} // BLU
};
void setup() {
  Serial.begin(9600);
  pinMode(A1, INPUT_PULLUP);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
  radio.begin();
  radio.setChannel(117);
  radio.setDataRate(RF24_250KBPS);
  radio.setPALevel(RF24_PA_LOW);
  radio.stopListening();
}
void loop()  {
  int tombol = !digitalRead(A1);
  int potensio = analogRead(A2);
  unsigned long sekarang = millis();
  if (sekarang - sebelum >= 2000) {
    dataKirim1.counter++;
    dataKirim2.counter++;
    radio.stopListening();
    if (giliran == 0) {
      radio.openWritingPipe(pipa[0]);
      radio.write(&dataKirim1, sizeof(dataKirim1));
      radio.openReadingPipe(1, pipa[1]);
    }
    else {
      radio.openWritingPipe(pipa[2]);
      radio.write(&dataKirim2, sizeof(dataKirim2));
      radio.openReadingPipe(1, pipa[3]);
    }
    radio.startListening();
    sebelum = sekarang;
  }
  if (radio.available()) {
    Serial.print(giliran + 1);
    Serial.print(',');
    if (giliran == 0) {
      radio.read( &dataTerima1, sizeof(dataTerima1));
      Serial.print(dataTerima1.counter);
      Serial.print(',');
      switch (dataTerima1.sudut) {
        case 1: Serial.print("UTR"); break;
        case 2: Serial.print("TLU"); break;
        case 3: Serial.print("TMR"); break;
```

```
88.           case 4: Serial.print("TGR"); break;
89.           case 5: Serial.print("SLT"); break;
90.           case 6: Serial.print("BDY"); break;
91.           case 7: Serial.print("BRT"); break;
92.           case 8: Serial.print("BLU"); break;
93.         }
94.         giliran = 1;
95.       }
96.       else {
97.         radio.read( &dataTerima2, sizeof(dataTerima2));
98.         Serial.print(dataTerima2.counter);
99.         Serial.print(',');
100         switch (dataTerima2.sudut) {
101           case 1: Serial.print("UTR"); break;
102           case 2: Serial.print("TLU"); break;
103           case 3: Serial.print("TMR"); break;
104           case 4: Serial.print("TGR"); break;
105           case 5: Serial.print("SLT"); break;
106           case 6: Serial.print("BDY"); break;
107           case 7: Serial.print("BRT"); break;
108           case 8: Serial.print("BLU"); break;
109         }
110         giliran = 0;
111       }
112       Serial.print(',');
113       Serial.print(tombol);
114       Serial.print(',');
115       Serial.println(potensio);
116       noTone(6);
117     }
118   }
119   void serialEvent() {
120     while (Serial.available()) {
121       int d = Serial.parseInt() + 1; //posisi Motor
122       int e = Serial.parseInt() + 1; //posisi Motor
123       int f = Serial.parseInt(); //nada Buzzer
124       int g = Serial.parseInt(); //pola Matriks
125       if (Serial.read() == char(13)) {
126         dataKirim1.stepper = d;
127         dataKirim2.stepper = e;
128         tone(6, nada[f - 1], 1000);
129         for (int i = 0; i < 8; i++) {
130           lc.setRow(0, i, pola[g - 1][7 - i]);
131         }
132       }
133     }
134   }
```

8. Gambar 7.25 berikut menunjukkan hasil implementasi program komunikasi Multi-Nodes. Tampak rangkaian Master, Slave1 dan Slave2 terpisah, dan juga terpisah dengan Komputer. Sekalipun keempatnya terpisah secara fisik, namun Komputer dapat berkomunikasi dengan Master dan Master dapat berkomunikasi dengan Slave1 dan Slave2. Serial Monitor di Komputer dapat mengirimkan data untuk posisi Motor stepper di Slave1 dan Slave2 melalui Master, serta dapat mengatur Nada Buzzer dan Pola Matriks LED di rangkaian Master. Serial Monitor juga menampilkan data Sensor Arah Angin di Slave1, Sensor Arah Angin di Slave2, kondisi Tombol dan Potensio di rangkaian Master.



Gambar 7.25 Implementasi program komunikasi Multi-Nodes dengan NRF24L01 antara Master dengan Slave1 & Slave2, yang dipantau dan dikendalikan Komputer melalui Bluetooth

7.5 Soal Latihan

1. Ganti tampilan Serial Monitor untuk aplikasi Komunikasi 1 Arah di Sub Bab 7.4.1 dengan tampilan LabVIEW. Buat tampilan LabVIEW tersebut dapat menampilkan data Sensor Arah Angin, Tombol dan Potensio perdetik yang dikirimkan dari Rangkaian Pemancar dalam bentuk objek grafis (Gauge, LED,

Tank, dll), Grafik Waveform Chart, Waveform Graph, 2D Compass Plot dan Table dengan rentang waktu 10 menit, dan dapat menyimpan data tersebut ke dalam sebuah file. Juga tambahkan pada tampilan LabVIEW tersebut data posisi sudut Motor Stepper, Buzzer dan Matriks LED, yang ditampilkan dalam bentuk objek grafis, seperti Dial, Slider, Knob, dll.

2. Ulangi Soal No. 1 di atas, untuk aplikasi Komunikasi 2 Arah di Sub Bab 7.4.2. Tambahkan pengaturan hubungan untuk ketiga Aktuator tersebut secara Grafis, Manual dan Otomatis. Jika dipilih Grafis, maka ketiga Aktuator secara bersama dapat dikendalikan melalui objek grafis yang telah dibuat. Jika dipilih Manual, maka ketiga Aktuator dapat dikendalikan secara bersama melalui hardware Potensio. Jika dipilih Otomatis, maka ketiga Aktuator dapat dikendalikan secara otomatis dari data Sensor Arah Angin.
3. Ulangi Soal No. 2 di atas, untuk aplikasi Komunikasi Multi-Nodes di Sub Bab 7.4.3 dengan tambahan data untuk Sensor Arah Angin dan Motor Stepper di Slave2. Di samping itu, tambahkan pengaturan hubungan untuk keempat Aktuator (Motor Stepper Slave1, Motor Stepper Slave2, Buzzer, Matriks LED) secara Grafis, Manual, Otomatis dan Alarm. Jika dipilih Grafis, maka keempat Aktuator dapat dikendalikan secara bersama melalui objek grafis yang telah dibuat. Jika dipilih Manual, maka keempat Aktuator dapat dikendalikan secara bersama melalui Potensio. Jika dipilih Otomatis, maka Motor Stepper di Slave1, Buzzer, Matriks LED dikendalikan secara otomatis dari data Sensor Arah Angin di Slave1, dan hanya Motor Stepper di Slave2 dikendalikan secara otomatis dari data Sensor Arah Angin di Slave2. Jika dipilih Alarm, Motor Stepper di Slave1 dan Motor Stepper di Slave2 akan mengikuti nilai Potensio. Ketika data Sensor Arah Angin di Slave1 atau Sensor Arah Angin di Slave2 berada tepat pada posisi daerah nilai potensio, maka Buzzer akan berbunyi. Buzzer hanya berhenti berbunyi apabila Tombol ditekan dan kedua Sensor Arah Angin sudah tidak berada pada daerah nilai Potensio. Tambahkan catatan waktu pada Table saat Buzzer berbunyi dan saat Tombol ditekan.

DAFTAR PUSTAKA

[1]. Dian Artanto, Andreas Prasetyadi, Doddy Purwadianto, Rusdi Sambada. 2016. Design of a GPS-based solar tracker system for a vertical solar still. ICSGTEIS 2016. Universitas Udayana. Bali.

[2]. Dian Artanto, Andreas Prasetyadi, Doddy Purwadianto. 2015. Pengembangan Datalogger Pemantau Efisiensi Destilasi Air Energi Surya. SNTT 2015. Polinema. Malang.

[3]. Dian Artanto. 2013. The Development of a Simple Tool to Reduce the Sitting Time using Seeduino Stalker and LabVIEW. ICICI BME2013. ITB. Bandung.

[4]. Dian Artanto. 2012. Interaksi Arduino dan LabVIEW. PT Elex Media Komputindo.

[5]. Paul P.L. Regtien. 2012. Sensor for Mechatronics. Elsevier Inc.

[6]. Ronald W Larsen. 2011, LabVIEW for Engineers. Pearson Inc.

[7]. Michael Margolis. 2011. Arduino Cookbook. O'Reilly Media Inc.

[8]. Jon S. Wilson. 2005. Sensor Technology Handbook. Elsevier Inc.

[9]. Sumber bahan dari internet di www.ni.com.

[10]. Sumber bahan dari internet di arduino-info.wikispaces.com.