BAB II

LANDASAN TEORI

2.1 Definisi Robot

Kata robot berasal dari kosakata "*Robota*" yang berarti "kerja cepat", yang berasal dari Cekoslowakia. Istilah robot pertama kali dikenalkan dalam sebuah drama antara tahun 1920-1921 oleh seorang pengarang sandiwara bernama Karel Capec. Karyanya dikenal berjudul "*Rossum's Universal Robots*", yang artinya robot dunia milik Rossum [1].

Pada tahun 1942, Isaac Asimov juga memperkenalkan istilah robot pada karyanya yang berbentu novel fiksi sains "*runaround*" . Pengertian mengenai apa itu robot terus mengalami perkembangan. Menurut *Robotics Industry Association* (1985) robot adalah sebuah manipulator multifungsi yang dapat diprogram ulang yang di desain untuk memindahkan material, bagian, peralatan untuk berbagai macam tugas. Pengertian robot sendiri mengalami perluasan, hal ini nampak dari definisi yang dibuat oleh Roy Arkin (1998). Menurutnya, robot adalah sebuah mesin yang mampu mengekstrak informasi dari lingkungannya dan menggunakan pengetahuan mengenai lingkungannya tersebut untuk melakukan tindakan yang sadar dan bertujuan secara aman. Selain itu pengertian lain dari robot adalah sistem atau alat yang terdiri dari sistem mekanik, sistem elektronik dan sistem kendali yang dapat berperilaku atau meniru perilaku manusia dengan tujuan menggantikan dan mempermudah kerja/aktifitas manusia [6].

Dari pengertian tersebut dapat ditarik tiga hal utama yang harus dimiliki oleh sebuah mesin untuk mendapatkan predikat sebagai robot, diantaranya adalah sebagai berikut:

a. Kemampuan untuk mengekstrak/mendapatkan informasi dari sekelilingnya.
b. Kemampuan untuk mengolah informasi tersebut.
c. Kemampuan untuk melakukan sesuatu secara fisik seperti bergerak atau memanipulasi objek.

Untuk dapat dikatakan sebagai robot sebuah sistem tidak perlu untuk meniru semua tingkah laku manusia. Namun suatu sistem tersebut dapat mengadopsi satu atau dua dari sistem yang ada pada diri manusia saja sudah dapat dikatakan sebagai robot. Untuk melakukan ketiga hal tersebut di atas, robot senantiasa dilengkapi dengan tiga komponen utama: sensor, aktuator dan pengendali atau kontroler. Sampai saat ini terdapat beberapa jenis robot yang dibedakan berdasarkan bentuknya antara lain *humanoid robot*, robot industri dan *mobile robot* (*Autonomous Mobile Robot*).

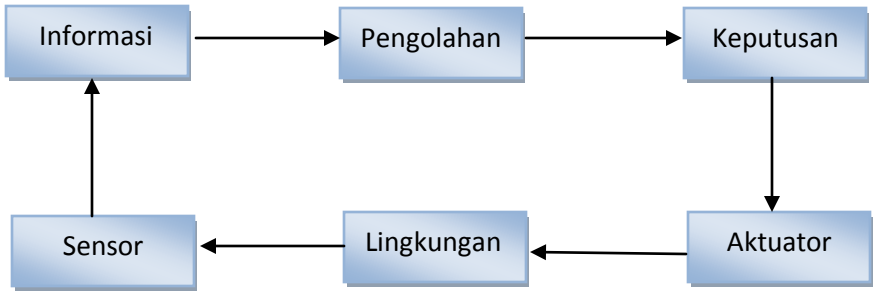
2.1.1 *Autonomous Mobile Robot* (AMR)

Mobile dapat diartikan bergerak, sehingga robot ini dapat memindahkan dirinya dari satu tempat ke tempat lain. Jadi, *Autonomous Mobile Robot* adalah robot bergerak yang mampu melakukan tugas-tugas suatu lingkungan tanpa panduan dari manusia secara berkelanjutan. Secara praktek, sebuah *Autonomous Mobile Robot* memiliki definisi sebagai gabungan alat elektronik dan mekanik seperti motor, gearbox, ban, dan susunan barang elektronik yang setiap bagiannya membentuk

sebagian kehidupan dan keseluruhannya akan menghasilkan suatu benda menyerupai manusia yang mampu melakukan tugas-tugas seperti bergerak, pemberi peringatan terhadap bahaya, dapat memberikan informasi dari sensor, dan sebagainya.

Kemampuan yang harus dimiliki setiap *Autonomous Mobile Robot* adalah otonom, bergerak, dan cerdas. Otonom artinya mampu melakukan tugasnya tanpa bantuan dari manusia secara terus menerus. Bergerak artinya robot dapat berpindah dari satu tempat ke tempat lain dengan aktuator seperti motor DC, motor *servo*, dan motor *stepper*. Untuk melakukan pergerakan otonom tersebut, maka dibutuhkan suatu kecerdasan. Kecerdasan tersebut harus mampu mengenali lingkungan yang dilewatinya, beradaptasi, mempelajarinya, kemudian menelaahnya sesuai dengan algoritma yang dipakai dan menggunakannya untuk mengambil keputusan pada setiap langkah berikutnya untuk mencapai tujuan. Kemampuan-kemampuan tersebut dapat diperoleh dari berbagai komponen seperti sensor, aktuator, dan perangkat untuk mengambil keputusan. Secara umum, struktur dari sebuah *Autonomous Mobile Robot* dapat dilihat pada gambar 2.1.

Kemampuan mengenali lingkungan yang dilakukan oleh sensor-sensor akan memberikan suatu informasi yang akan digunakan oleh pengambilan keputusan yang dapat berupa kontroler atau prosesor. Setelah pengolahan informasi, keputusan akan dihasilkan yang kemudian dilaksanakan oleh aktuator. Pergerakan aktuator memberikan perubahan *Autonomous Mobile Robot* terhadap lingkungan. Sensor-sensor akan bekerja kembali untuk beradaptasi terhadap langkah selanjutnya.



Gambar 2.1 Skema Sistem Kerja Robot

Dalam melakukan tugasnya, sebuah *Autonomous Mobile Robot* memerlukan beberapa kemampuan dasar seperti berikut:

a. Otonom

Autonomous Mobile Robot harus otonom atau tidak memerlukan kendali dari manusia secara langsung. Pengguna hanya perlu melakukan pengaturan (*setting*) dan menetapkan algoritma sistem pada saat pengaturan awal. Setelah itu *Autonomous Mobile Robot* diharapkan mampu menentukan arah pergerakannya sendiri dalam memenuhi misi dengan mengatasi keadaan di lingkungannya.

b. Bergerak

Autonomous Mobile Robot harus mampu bergerak atau berpindah dari suatu tempat ke tempat lain, sehingga harus memiliki komponen-komponen penggerak seperti motor, kontroler aktuator, dan desain roda penggerak robot.

c. Cerdas

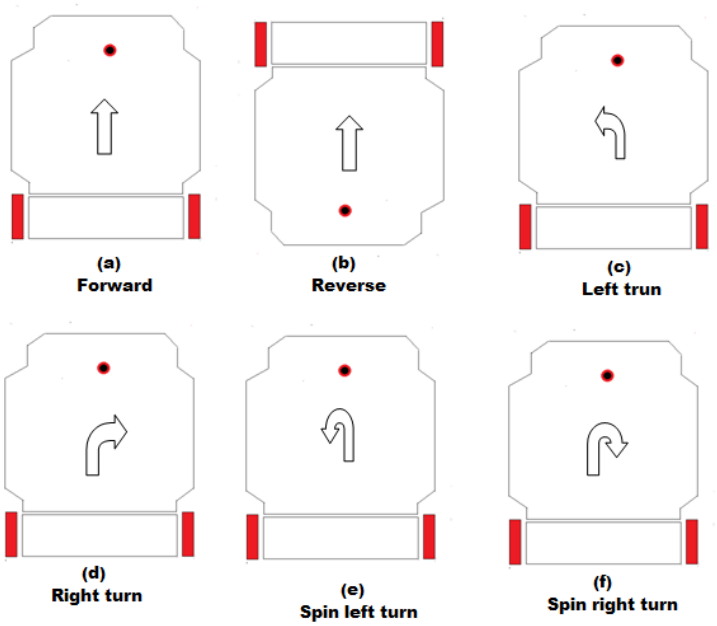
Autonomous Mobile Robot harus mampu mengenali lingkungan yang dilewatinya, beradaptasi, mempelajarinya, kemudian menelaahnya sesuai dengan algoritma yang dipakai dan menggunakannya untuk mengambil keputusan pada setiap langkah berikutnya untuk mencapai tujuan yang diberikan.

2.1.2 Prinsip Pergerakan *Mobile Robot* dan Sistem Navigasi Robot

Pergerakan merupakan cara berpindah dari satu titik ke titik lain. Terdapat banyak bentuk pergerakan robot, yang paling umum adalah roda, rel, dan kaki. Salah satu contoh sistem kemudi untuk robot beroda adalah *Differential Steering.*

1. *Differential Steering Robot*

*Differential Steering Robo*t adalah robot yang dapat bermanuver dengan memanipulasi perbedaan kecepatan kedua roda di sisi kanan dan sisi kiri. Di bawah ini merupakan ilustrasinya yang ditunjukkan pada gambar 2.2.



Gambar 2.2 Ilustrasi Sistem Pergerakan *Differential Steering*

Berdasarkan gambar 2.2, pergerakan *differential steering* dapat dijelaskan sebagai berikut:

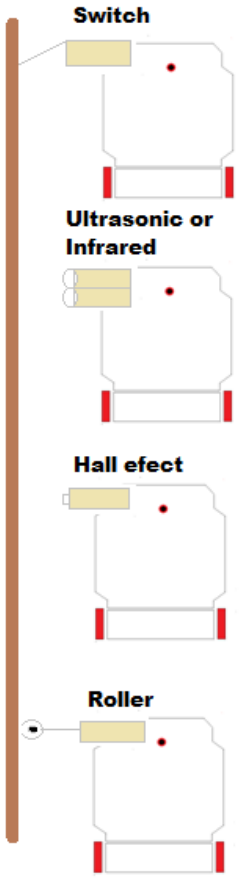
a. Posisi a (*forward*). Jika kedua motor berputar dengan arah dan kecepatan yang sama maka robot akan lurus maju ke depan.

b. Posisi b (*reverse*). Jika kedua motor berputar dengan arah dan kecepatan yang sama maka robot akan lurus mundur ke belakang.
c. Posisi c (*left turn*). Jika di sisi kanan berputar dan motor di sisi kiri tidak berputar, maka robot akan bergerak/berbelok ke kiri.
d. Posisi d (*right turn*). Jika di sisi kiri berputar dan di sisi motor di sisi kanan tidak berputar, maka robot akan bergerak/berbelok ke kanan.
e. Posisi e (*spin left turn*). Motor kiri dan kanan berputar dengan arah yang berlawanan. Pada keadaan ini robot akan berbelok/berputar ke kiri/berlawan arah jarum jam dengan pusat putaran pada titik hitam (pivot point). Untuk melakukan maneuver ini, kedua motor harus berputar dengan kecepatan yang sama sehingga pivot point tetap berada di tengah.
f. Posisi f (*spin right turn*). Motor kiri dan kanan berputar dengan arah yang berlawanan. Pada keadaan ini robot akan berbelok/berputar ke kanan/searah jarum jam dengan pusat putaran pada titik hitam (*pivot point*). Untuk melakukan maneuver ini, kedua motor harus berputar dengan kecepatan yang sama sehingga *pivot point* tetap berada di tengah.

2. *Wall Following*

Wall Following adalah suatu algoritma untuk menyediakan orientasi navigasi kepada robot dengan menyusuri dinding. Salah satu keuntungannya adalah tidak perlu adanya garis penuntun atau suatu tanda khusus sebagai arahan bagi robot. Cara kerjanya adalah dengan mengatur jarak dinding dengan robot agar tetap konstan. Bila terjadi perubahan, maka robot akan bergerak untuk kemudian menyesuaikan jarak

lagi [1]. Proses ini akan dilakukan secara berulang-ulang. Di bawah ini merupakan metode-metode *wall following* yang ditunjukkan pada gambar 2.3.



Gambar 2.3 Metode-metode *Wall Following*

Dari gambar di atas, dapat dijelaskan metode-metode *wall following* tersebut:

a. *Contact.* Robot menggunakan saklar mekanik yang merasakan sentuhan dengan dinding. Ini adalah metode yang paling mudah namun saklar akan cenderung mengalami kerusakan mekanis setelah beberapa waktu karena terus bersentuhan dengan objek (dinding).

b. *Noncontact, active sensor.* Robot menggunakan sensor aktif yang beroperasi dalam jarak dekat seperti infra merah atau ultrasonik untuk mengukur jarak antara dinding dengan robot.

c. *Noncontact, passive sensor.*Robot memakai sensor pasif seperti saklar *Hall effect* untuk mengukur jarak antara robot dengan dinding. Pada kasus ini, dinding harus berbahan logam atau dipasangkan kabel elektrik agar sensor dapat menangkap medan magnetik saat robot mendekati dinding.

d. *Soft-contact.*Robot menggunakan bahan mekanik untuk mendeteksi sentuhan dengan dinding, namun sentuhan ini diperhalus dengan memasang material lunak atau lentur contohnya roda dari busa atau karet. Kelebihan dari metode ini adalah berkurangnya kerusakan mekanis.

2.2 Pengendali

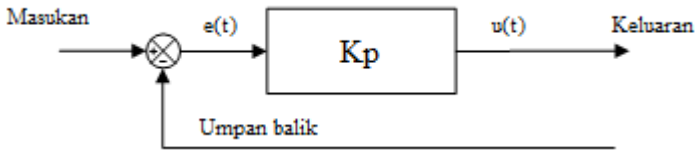
Pengendali merupakan salah satu komponen sistem yang berfungsi mengolah sinyal umpan balik dari sinyal referensi menjadi sinyal kendali sedemikian rupa sehingga performansi dari sistem yang dikendalikannya sesuai dengan spesifikasi performansi yang diinginkan. Keberadaan kendali dalam sebuah sistem kendali mempunyai kontribusi yang besar terhadap perilaku sistem. Pada prinsipnya hal itu disebabkan oleh tidak dapat diubahnya komponen penyusun sistem tersebut. Artinya, karakteristik plant harus diterima sebagaimana adanya, sehingga perubahan perilaku sistem hanya dapat dilakukan melalui penambahan suatu sub sistem, yaitu pengendali. Salah satu tugas komponen pengendali adalah mereduksi sinyal kesalahan, yaitu perbedaan antara sinyal setting dan sinyal aktual agar sama atau

mendekati *setpoint*. Semakin cepat reaksi sistem mengikuti sinyal aktual dan semakin kecil kesalahan yang terjadi, semakin baik kinerja sistem kendali yang diterapkan.

Pada sub bab selanjutnya akan dibahas tentang kendali Proporsional, kendali Integral, kendali Derivatif, dan penggabungan ketiga kendali menjadi kendali PID.

2.2.1 Kendali Proporsional

Kendali Proporsional menghasilkan sinyal keluaran yang sebanding atau proporsional dengan besarnya sinyal kesalahan (selisih antara besaran yang diinginkan dengan harga aktualnya). Secara lebih sederhana dapat dikatakan, bahwa keluaran kendali proporsional merupakan perkalian antara konstanta proporsional dengan masukannya [7]. Gambar 2.4 menunjukkan diagram blok kendali proporsional.



Gambar 2.4 Diagram Blok Kendall Proporsional

Hubungan antara *output* kendali proporsional u(t) dengan sinyal *error* e(t) dibuat oleh persamaan:

$$u(t) = K_P \cdot e(t) \quad \text{.....(2.1)}$$

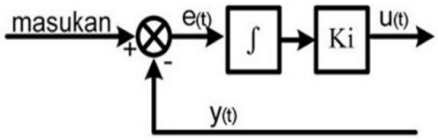
Ciri-ciri kendali proporsional dapat dijelaskan sebagai berikut.

1. Jika nilai K_P kecil maka kendali proporsional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sistem yang lambat.

2. Jika nilai K_P dinaikkan, respon sistem menunjukkan semakin cepat mencapai keadaan mantapnya.
3. Jika nilai K_P diperbesar mencapai harga yang berlebihan, akan mengakibatkan sistem bekerja tidak stabil, atau sistem akan berosilasi.

2.2.2 Kendali Integral

Kendali Integral berfungsi menghasilkan respon sistem yang memiliki *error steady state* nol. Kalau sebuah *plant* tidak memiliki unsur integrator (1/*s*), kendali proporsional tidak akan mampu menjamin keluaran sistem dengan kesalahan keadaan mantapnya nol. Karena itu dengan adanya kendali integral, respon sistem dapat diperbaiki, yaitu mempunyai kesalahan keadaan mantapnya nol. Kendali integral memiliki karakteristik seperti halnya sebuah integral. Keluaran pengendali sangat dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal kesalahan. Keluaran pengendali ini merupakan jumlahan yang terus menerus dari perubahan masukannya. Kalau sinyal kesalahan tidak mengalami perubahan, keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan [7]. Gambar 2.5 menunjukkan hubungan antara *output* kendali integral u(t) dengan sinyal *error* e(t).



Gambar 2.5 Diagram Blok Kendali Integral

Diagram blok di atas dapat diwujudkan dalam bentuk persamaan yang menunjukkan hubungan antara output kendali integral u(t) dengan sinyal *error* e(t) sebagai berikut:

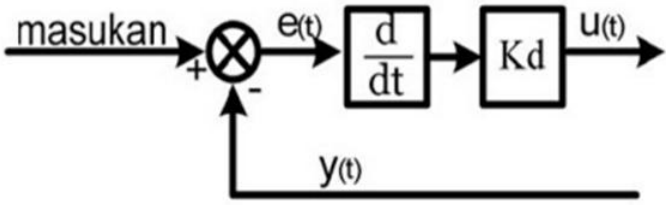
$$u(t) = K_i \int_0^t e(t)dt \quad \text{.....(2.2)}$$

Ciri-ciri kendali Integral dijelaskan sebagai berikut.

1. Keluaran pengendali membutuhkan selang waktu tertentu sehingga kendali integral cenderung memperlambat respon.
2. Ketika sinyal kesalahan berharga nol, keluaran pengendali akan bertahan pada nilai sebelumnya.
3. Jika sinyal kesalahan tidak berharga nol, keluaran akan menunjukan sinyal kesalahan dan nilai K_i.
4. Kendali integral K_i yang berharga besar akan mempercepat hilangnya *offset*. Tetapi semakin besar nilai konstanta K_i akan mengakibatkan peningkatan osilasi dari sinyal keluaran pengendali.

2.2.3 Kendali Derivatif

Keluaran kendali derivatif memiliki sifat seperti halnya suatu operasi derivatif. Perubahan yang mendadak pada masukan pengendali,akan mengakibatkan perubahan yang sangat besar dan cepat juga pada keluaran [7]. Diagram blok dari kendali derivatif ditunjukkan oleh gambar 2.6.



Gambar 2.6 Diagram Blok Kendali Derivatif

Diagram blok di atas diwujudkan dalam bentuk persamaan yang menunjukkan hubungan antara *output* kendali integral u(t) dengan sinyal *error* e(t) sebagai berikut:

$$u(t) = K_d \frac{de(t)}{dt} \quad \text{.....(2.3)}$$

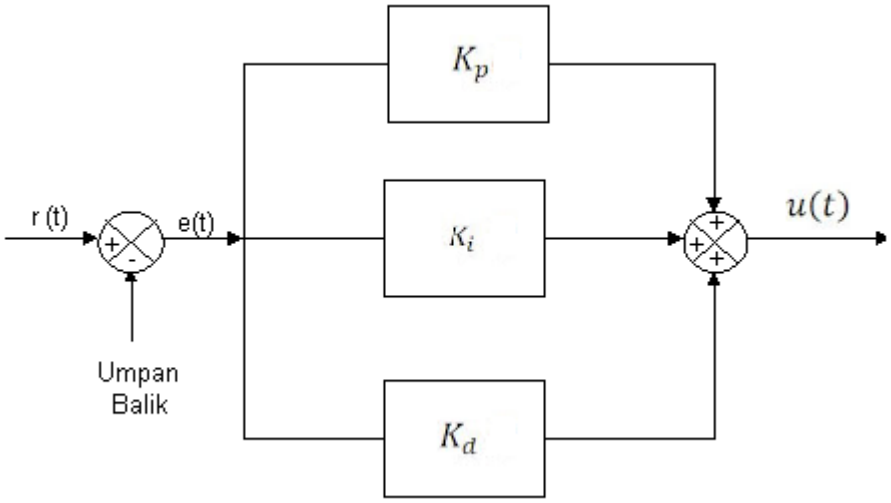
Ciri-ciri kendali derivatif sebagai berikut.

1. Pengendali ini tidak dapat menghasilkan keluaran bila tidak ada perubahan pada masukannya (berupa sinyal kesalahan).
2. Jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan pengendali tergantung pada nilai Kd dan laju perubahan sinyal kesalahan.
3. Kendali derivatif mempunyai karakteristik untuk mendahului, sehingga pengendali ini dapat menghasilkan koreksi yang signifikan sebelum pembangkit kesalahan menjadi sangat besar. Jadi kendali derivatif dapat mengantisipasi pembangkit kesalahan, memberikan aksi yang bersifat korektif, dan cenderung meningkatkan stabilitas sistem.

2.2.4 Kendali PID

Setiap kekurangan dan kelebihan dari masing-masing kendali Proporsional, Integral dan Dervatif dapat saling menutupi dengan menggabungkan ketiganya

menjadi kendali proporsional plus integral plus derivatif (kendali PID). Elemen-elemen kendali P, I dan D masing-masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset* dan menghasilkan perubahan awal besar. Diagram Blok kendali PID ditunjukkan pada gambar 2.7.



Gambar 2.7 Diagram blok kendali PID

Dari diagram blok di atas dapat diperoleh persamaan:

$$u(t) = \mathrm{K}_P e(t) + \mathrm{K}_i \int_0^t e\,(t)dt + \mathrm{K}_d \frac{de\,(t)}{dt} \quad \text{.....(2.4)}$$

2.3 Mikrokontroler ATmega32

Mikrokontroler ATmega32 merupakan salah satu mikrokontroler buatan Atmel yang memiliki banyak kegunaan. ATmega32 memiliki empat port yang dapat digunakan untuk banyak masukan atau keluaran, memiliki memori *flash* 32K *Bytes*, ADC dan PWM internal, timer dan fasilitas lainnya.

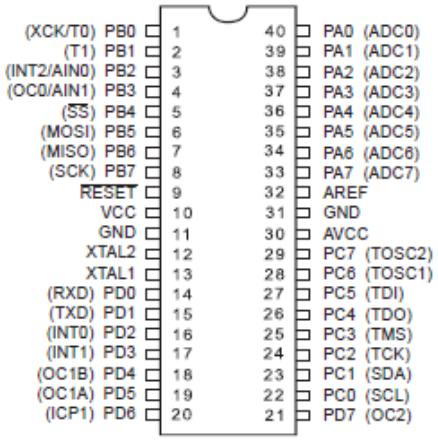


Gambar 2.8 Bentuk fisik ATmega32

Adapun beberapa fitur/keistimewaan dari mikrokontroler ATmega32 diantaranya:

- Memiliki 32 port I/O
- 32K Bytes Flash PEROM
- 1024 Bytes EEPROM
- 2K Bytes Internal SRAM
- 3 buah Timer/Counter
- 4 channel PWM
- 8 channel ADC dengan resolusi 10 bit
- Osilator internal
- USART
- Catu daya
    - 2.7 - 5.5V untuk ATmega32L
    - 4.5 – 5.5V untuk ATmega32
- Kecepatan
    - Maksimal 8 MHz untuk ATmega32L
    - Maksimal 16 MHz untuk ATmega32

Konfigurasi pin pada ATmega32 dengan kemasan 40 pin DIP (*dual in-line package*) ditunjukan pada gambar 2.9. Kemasan pin tersebut terdiri dari 4 port yaitu: port A, port B, port C, port D yang masing-masing terdiri dari 8 buah pin. Selain itu juga terdapat RESET, VCC, GND 2 buah, AVCC, XTAL1, XTAL2 dan AREF.



Gambar 2.9 Pin-pin ATmega32

Dari gambar 2.10 dapat dijelaskan pin-pin dari ATmega32.

1. **VCC** : Supply tegangan digital
2. **GND** : Ground
3. **PORT A** : Port A sebagai input analog ke A/D konverter. Port A juga sebagai 8-bit *bi-directional* port I/O, jika A/D konverter tidak digunakan. Pin-pin port dapat menyediakan resistor-resistor internal *pull-up*. Ketika port A digunakan sebagai input dan *pull* eksternal yang rendah akan menjadi sumber arus jika resistor-resistor *pull-up* diaktifkan. Berikut merupakan tabel fungsi-fungsi khusus pin-pin port A.

Tabel 2.1 Fungsi-fungsi Khusus Port A

Port	Fungsi
PA0	ADC0 (ADC *input channel* 0)
PA1	ADC1 (ADC *input channel* 1)
PA2	ADC2 (ADC *input channel* 2)
PA3	ADC3 (ADC *input channel* 3)
PA4	ADC4 (ADC *input channel* 4)
PA5	ADC5 (ADC *input channel* 5)
PA6	ADC6 (ADC *input channel* 6)
PA7	ADC7 *(*ADC *input channel* 7)

4. **PORT B** : Port B adalah port I/O 8-bit yang bersifat *bi-directional* dan setiap pin mengandung internal *pull-up* resistor. *Buffer* output port B mempunyai karakteristik *drive* yang simetris dengan kemampuan keduanya *sink* dan *source* yang tinggi. Sebagai input, port B yang mempunyai *pull* eksternal yang rendah akan menjadi sumber arus jika resistor-resistor *pull-up* diaktifkan. Berikut merupakan tabel fungsi-fungsi khusus pin-pin port B.

Tabel 2.2 Fungsi-fungsi Khusus Port B

Port	Fungsi
PB0	T0 (*Timer/Counter External Counter Input*) XCK (*USART External Clock Input/Output*)
PB1	T1 (*Timer/Counter1 External Counter Input*)
PB2	AIN0 (*Analog Comparator Positive Input*) INT2 (*External Interrupt 2 Input*)
PB3	AIN1 (*Analog Comparator Negative Input*) OCO (*Timer/Counter0 Output Compare Match Output*)
PB4	SS (*SPI Slave Select Input*)
PB5	MOSI (*SPI Bus Master Output/Slave Input*)
PB6	MISO (*SPI Bus Master Input/Slave Output*)
PB7	SCK (*SPI Bus Serial Clock*)

5. **PORT C** : Port C adalah port I/O 8-bit yang bersifat *bi-directional* dan setiap pin mengandung internal *pull-up* resistor. *Buffer* output port C mempunyai karakteristik *drive* yang simetris dengan kemampuan keduanya *sink* dan *source*

yang tinggi. Sebagai input, port C yang mempunyai *pull* eksternal yang rendah akan menjadi sumber arus jika resistor-resistor *pull-up* diaktifkan. Berikut merupakan tabel fungsi-fungsi khusus pin-pin port C.

Tabel 2.3 Fungsi-fungsi Khusus Port C

Port	Fungsi
PC0	SCL (*Two-wire Serial Bus Clock Line)*
PC1	SDA (*Two-wire Serial Bus Data Input/Output Line)*
PC2	TCK (JTAG *Test Clock)*
PC3	TMS (JTAG *Test Mode Select)*
PC4	TD0 (JTAG *Test Data Out*)
PC5	TD1 (JTAG *Test Data In*)
PC6	TOSC1 (*Timer Oscillator* Pin 1)
PC7	TOSC2 (*Timer Oscillator* Pin 2)

6. **PORT D** : Port D adalah port I/O 8-bit yang bersifat *bi-directional* dan setiap pin mengandung internal *pull-up* resistor. *Buffer* output port B mempunyai karakteristik *drive* yang simetris dengan kemampuan keduanya *sink* dan *source* yang tinggi. Sebagai input, port D yang mempunyai *pull* eksternal yang rendah akan menjadi sumber arus jika resistor-resistor *pull-up* diaktifkan. Berikut merupakan tabel fungsi-fungsi khusus pin-pin port D.

Tabel 2.4 Fungsi-fungsi Khusus Port D

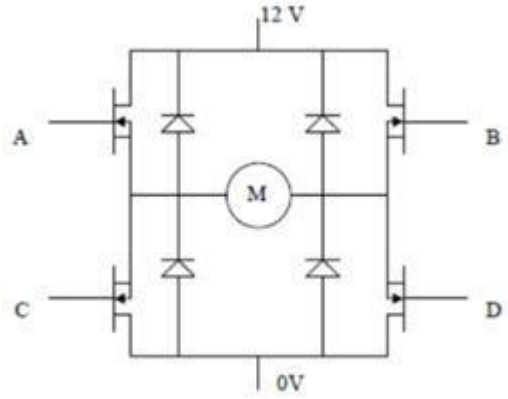
Port	Fungsi
PD0	RXD (USART *Input Pin)*
PD1	TXD (USART *Output Pin)*
PD2	INT0 (*External Interrupt 0 Input)*
PD3	INT1(*External Interrupt 1 Input)*
PD4	TD0 (JTAG *Test Data Out*)
PD5	OCIB (*Timer/Counter1 Output Compare B Match*)
PD6	ICP1 (*Timer/Counter1 Input Capture Pin*)
PD7	OC2 (*Timer/Counter2 Output Compare Match Output*)

7. **RESET** : Sebuah *low level* pulsa yang lebih lama dari pada lebar pulsa minimum pada pin ini akan menghasilkan reset meskipun *clock* tidak berjalan.
8. **XTAL 1** : Input inverting penguat *Oscilator* dan input internal clock operasi rangkaian.
9. **XTAL 2** : Output dari *inverting* penguat *Oscilator*.
10. **AVCC** : Pin *supply* tegangan untuk port A dan A/D konverter. Sebaliknya eksternalnya dihubungkan ke VCC meskipun ADC tidak digunakan.
11. **AREF** : pin referensi analog untuk A/D konverter.

2.4 *Driver* Motor DC

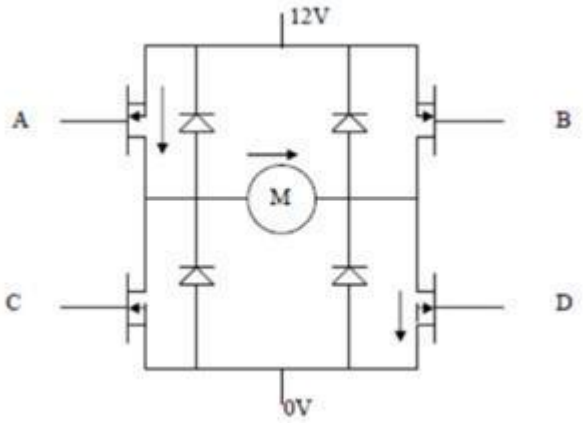
Dalam menggerakkan motor, pada dasarnya mikrokontroler dapat langsung dihubungkan dengan motor DC, dengan catatan harus memenuhi keluaran arus dari mikrontroler. Jika arus motor besar, maka tidak dapat langsung dihubungkan dengan mikrokontroler karena keluaran arus dari pin-pin mikrokontroler terlalu kecil. Jadi dibutuhkan *driver* yang sesuai dengan spesifikasi dari motor yang akan digunakan. Dalam prakteknya, dapat dijumpai berbagai macam driver motor seperti L293D, L298, dan *Dual H-Bridge* Mosfet.

H-bridge adalah sekumpulan elemen elektronik yang disatukan dalam satu rangkaian yang berfungsi untuk menggerakkan motor. Rangkaian ini diberi nama *H-bridge* karena bentuk rangkaiannya yang menyerupai huruf H seperti pada gambar 2.10.



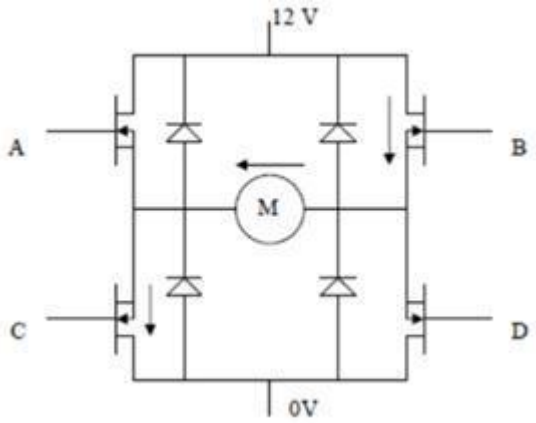
Gambar 2.10 Konfigurasi *H-Bridge*

Rangkaian ini terdiri dari dua buah MOSFET kanal P dan dua buah MOSFET kanal N. Prinsip kerja rangkaian ini adalah dengan mengatur *On-Off* ke empat MOSFET tersebut. Titik tengah dari rangkaian merupakan motor DC yang akan dikendalikan. Bagian atas rangkaian akan dihubungkan dengan sumber daya kutub positif, sedangkan bagian bawah rangkaian akan dihubungkan dengan sumber daya kutub negatif. Pada saat MOSFET A dan MOSFET D *on* sedangkan MOSFET B dan MOSFET C *off*, maka sisi kiri dari gambar motor akan terhubung dengan kutub positif dari catu daya, sedangkan sisi sebelah kanan motor akan terhubung dengan kutub negatif dari catu daya sehingga motor akan bergerak searh jarum jam [1]. Dijelaskan pada gambar 2.11.



Gambar 2.11 Konfigurasi MOSFET A&D *ON*, B&C *OFF*

Sebaliknya, jika MOSFET B dan MOSFET C *on* sedangkan MOSFET A dan MOSFET D *off*, maka sisi kanan motor akan terhubung dengan kutub positif dari catu daya sedangkan sisi kiri motor akan terhubung dengan kutub negatif dari catu daya. Maka motor akan bergerak berlawanan arah jarum jam. Dijelaskan pada gambar 2.12.



Gambar 2.12 Konfigurasi MOSFET A&D *OFF*, B&C *ON*

Konfigurasi lainnya adalah apabila MOSFET A dan MOSFET B sedangkan MOSFET C dan MOSFET D *off*. Konfigurasi ini akan menyebabkan sisi kiri dan kanan motor terhubung pada kutub yang sama yaitu kutub positif sehingga tidak ada

perbedaan tegangan diantara dua buah polaritas motor, sehingga motor tidak ada respon. Konfigurasi ini disebut sebagai konfigurasi *break.* Begitu juga jika MOSFET C dan MOSFET D saklar *on,* sedangkan MOSFET A dan MOSFET C *off,* kedua polaritas motor akan terhubung pada kutub negatif dari catu daya. Maka tidak ada perbedaan tegangan pada kedua polaritas motor, dan motor tidak akan memberikan respon. Konfigurasi yang harus dihindari adalah pada saat MOSFET A dan MOSFET C *on* secara bersamaan atau MOSFET B dan MOSFET D *on* secara bersamaan. Pada konfigurasi ini akan terjadi hubungan arus singkat antara kutub positif catu daya dengan kutub negatif catu daya.

2.5 Motor DC

Motor DC adalah transducer yang mengubah energi listrik menjadi energi mekanik berupa gerak rotasi. Bentuk fisik motor DC ditunjukkan oleh gambar 2.13.



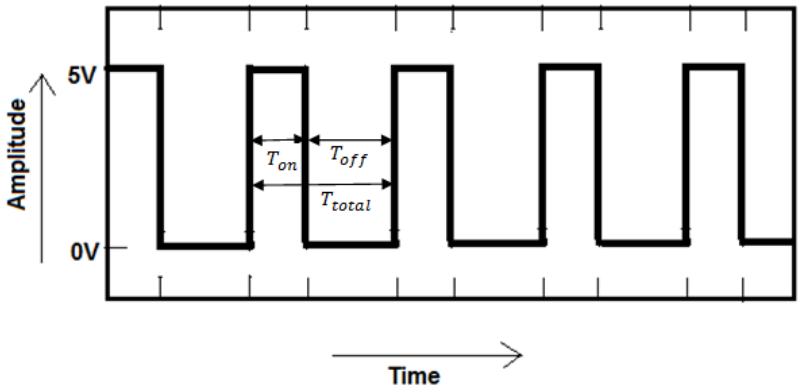
Gambar 2.13 Bentuk fisik motor DC

Kecepatan minimum sampai maksimumnya dari motor DC dapat diatur berdasarkan *duty cycle* yang diberikan, jika *duty cycle* pada sinyal PWM besar, maka akan menghasilkan rata-rata tegangan yang yang besar pula sehingga motor DC akan

berputar cepat, begitu juga sebaliknya jika *duty cycle* yang diberikan kecil, akan menghasilkan rata-rata tegangan yang kecil yang mengakibatkan motor DC berputar lambat.

2.6 *Pulse Width Modulation* (PWM)

PWM adalah sebuah cara memanipulasi lebar sinyal yang dinyatakan dengan pulsa dalam periode yang berulang antara *high* atau *low*, dimana durasi sinyal *high* atau *low* dapat diatur. Ilustrasi sinyal PWM ditunjukkan oleh gambar 2.14.



Gambar 2.14 Sinyal PWM

Duty cycle merupakan perbandingan periode lamanya suatu sistem bernilai logika *high* dan *low*. Variasi *duty cycle* ini memberikan harga tegangan rata-rata yang berbeda. Sinyal PWM dengan *duty cycle* yang besar memiliki nilai rata-rata tegangan yang lebih besar dibandingkan dengan *duty cycle* kecil. Nilai tegangan yang diberikan sebanding dengan nilai *duty cycle* yang diberikan [3]. Adapun rumus dari *duty cycle*, yaitu:

$$Duty\ Cycle = \frac{t_{high}}{(t_{high}+t_{low})} \text{ x } 100\% \quad \text{.....(2.5)}$$

Sedangkan untuk menghitung tegangan rata-rata, *output* dapat dihitung dengan rumus:

$$V_{out} = Duty\ Cycle \times V_{total} \quad (2.6)$$

Dari rumus di atas, V_{total} merupakan tegangan yang diberikan untuk mengaktifkan motor.

2.7 Modul Bluetooth HC-05

Modul Bluetooth HC-05 adalah modul yang digunakan untuk mengirimkan data serial secara *wireless* (tanpa kabel). Modul Bluetooth HC dibedakan menjadi dua yaitu modul tipe ganjil dan genap. Modul tipe ganjil memiliki keunggulan yaitu dapat dijadikan sebagai master maupun slave pada mode operasinya [8]. Berbeda dengan tipe genap yang hanya mampu memiliki salah satu mode operasi, karena sudah diset oleh pabrik.



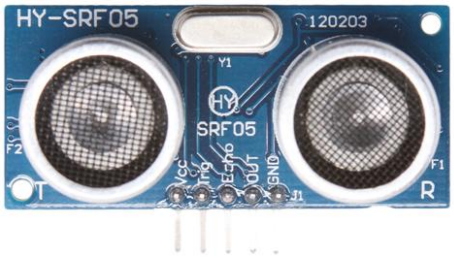
Gambar 2.15 Modul Bluetooth HC-05

Modul Bluetooth HC-05 digunakan sebagai peranti untuk menjembatani antara mikrokontroler dengan laptop supaya dapat berkomunikasi. Modul ini dihubungkan

dengan robot dan ketika power robot ON, modul akan secara otomatis mengirimkan data pada laptop.

2.8 Sensor Jarak Ultrasonik HY-SRF05

Sensor HY-SRF05 merupakan salah satu jenis sensor ultrasonik yang banyak digunakan khususnya pada bidang robotika. Sensor ini terdiri dari rangkaian pemancar (*transmitter*) dan rangkaian penerima (*receiver*), dan dapat mendeteksi jarak objek dengan cara memancarkan gelombang ultrasonik dengan frekuensi 40 kHz dan kemudian mendeteksi pantulannya. Sensor HY-SRF05 memiliki jarak pengukuran terendah 2cm dan maksimal pengukuran jarak sampai 450cm. selain itu sensor ini juga memiliki tingkat akurasi yang tinggi yaitu 0,3cm. Tampilan sensor HY-SRF05 ditunjukkan oleh gambar 2.16.



Gambar 2.16 Sensor Ultrasonik HY-SRF05

Adapun spesifikasi dari sensor HY-SRF05 adalah sebagai berikut:

- Jarak pengukuran : 2cm – 450cm
- Catu daya : 5Vdc
- Arus statis : 2mA
- Akurasi pengukuran : 0,3cm

- Frekuensi kerja : 40kHz
- Temperatur : 20-60°C
- Koneksi : VCC, Trigger, Echo, GND

Pada dasarnya sensor ultrasonik terdiri dari 3 bagian utama, yaitu chip pembangkit, speaker ultrasonik dan mikropon ultrasonik. Chip pembangkit berfungsi untuk membangkitkan sinyal 40kHz. Speaker ultrasonik berfungsi untuk mengubah sinyal 40kHz menjadi suara. Mikropon ultrasonik berfungsi untuk mendeteksi pantulannya [10].

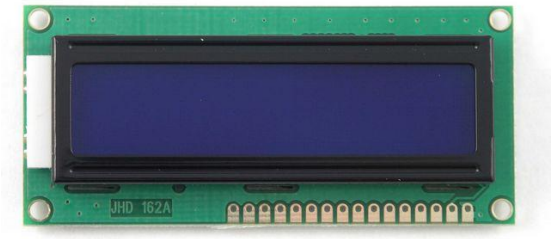
Prinsip kerja dari sensor ultrasonik yaitu gelombang ultrasonik akan dibangkitkan dan dipancarkan melalui *transmitter* secara menyebar. Gelombang ultrasonik yang dipancarkan tersebut akan merambat sebagai sinyal/gelombang bunyi dengan kecepatan bunyi yang berkisar 340 m/s. Sinyal tersebut kemudian akan dipantulkan dan akan diterima kembali oleh bagian *receiver* ultrasonik. Setelah sinyal tersebut diterima kembali, kemudian sinyal tersebut akan diproses untuk menghitung jaraknya. Jarak tersebut dihitung berdasarkan rumus :

$$S = 340.t/2 \quad (2.6)$$

Dari persamaan di atas, S adalah jarak antara sensor ultrasonik dengan bidang pantul, dan t adalah selisih waktu antara pemancaran gelombang ultrasonik sampai diterima kembali oleh bagian penerima ultrasonik.

2.9 LCD 2x16

Liquid Crystal Display (LCD 2x16) adalah modul elektronik yang dapat menampilkan suatu karakter tertentu dan memiliki 2 baris dan 16 kolom. LCD ini dilengkapi panel LCD serta dilengkapi dengan kontroler LCD (CMOS) yang sudah terintegrasi dengan modul tersebut. Selain itu LCD jenis ini memiliki konsumsi daya yang rendah. LCD 2x16 memiliki memori yang dapat diakses yaitu *Display Data Random Access Memory* (DDRAM).



Gambar 2.17 Bentuk fisik LCD 2x16

Berikut merupakan tabel pin-pin untuk LCD 2x16:

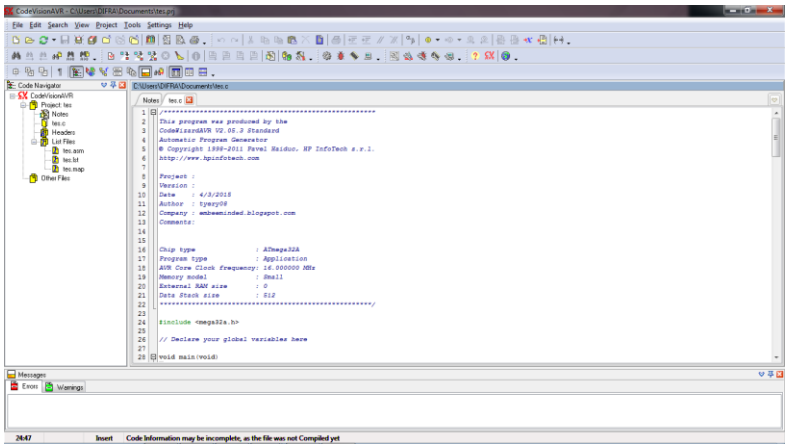
Tabel 2.5 Susunan kaki LCD 2x16

NO	Nama Pin	Deskripsi	Port
1	VCC	+ 5 V	VCC
2	GND	0 V	GND
3	VEE	Tegangan kontras	Trimpot
4	RS	*Register select*, 0 = input intruksi, 1 = input data	PD 5
5	R/W	1 = *read* : 0 = *write*	PD 6
6	E	*Enable clock*	PD 7
7	D0	*Data bus 0*	PC 0
8	D1	*Data bus 1*	PC 1
9	D2	*Data bus 2*	PC 2
10	D3	*Data bus 3*	PC 3

11	D4	*Data bus 4*	PC 4
12	D5	*Data bus 5*	PC 5
13	D6	*Data bus 6*	PC 6
14	D7	*Data bus 7*	PC 7
15	Anoda	Tegangan Positif *backlight*	VCC
16	Katoda	Tegangan Negatif *backlight*	GND

2.10 Pemrograman Bahasa C (CodeVision AVR)

Bahasa pemrograman yang digunakan dalam tugas akhir ini adalah program bahasa C dengan compiler CodeVision AVR. CodeVision AVR merupakan salah satu bahasa pemrograman yang dapat digunakan untuk memprogram mikrokontroler. Keunggulan dari CodeVision AVR yaitu memiliki fasilitas codewizard. Fasilitas ini akan memudahkan dalam inisialisasi mikrokontroler yang akan digunakan.



Gambar 2.18 Tampilan depan CodeVision AVR

CodeWizardAVR *Automatic Program Generator* merupakan fasilitas CodeVision AVR yang dapat menghasilkan kode program secara otomatis. Dengan adanya fasilitas ini, maka penulisan program dapat dilakukan dengan waktu yang

relatif cepat dan lebih efisien. Seluruh kode dapat dimplementasikan dengan fungsi sebagai berikut:

- Inisialisasi port input/output
- Inisialisasi LCD
- Inisialisasi interupsi eksternal
- Inisialisasi timer/counter dan *watchdog timer*
- Inisialisasi USART (*Universal Synchronous Asychronous Serial Receiver Transmitter*)
- Inisialisasi ADC dan komparator analog
- Inisialisasi antarmuka SPI dan TWI
- Inisialisasi *watchdog timer*
- Inisialisasi komparator analog
- Inisialisasi ADC
- Inisialisasi I2C Bus, sensor suhu, DS1621 thermometer/thermostat dan *real time* PCF8563, PCF8583, DS1302, DS1307
- Inisialisasi 1 *wire* bus dan sensor suhu DS1820/DS18S20

CodeVision AVR adalah *compiler* dengan bahasa pemrograman C. Untuk dapat membangun program, maka harus diperhatikan struktur yang digunakan dalam pemrograman C.

1. Header

   Header berisi *include* file (.hex), yaitu *library* (pustaka) yang akan digunakan dalam pemrograman.

   Contoh penulisan:

   ```
   #include <mega32.h>
   #include <delay.h>
   #include <alcd.h>
   ```

2. Deklarasi konstanta global atau variabel

   a. Deklarasi konstanta

   Aturan penulisan konstanta dijelaskan sebagai berikut:

   - Integer atau Ing integer : dapat ditulis dengan format desimal dan tidak didahului angka nol (contoh 678), biner ditulis dengan awalan 0b (contoh 0b00001111), octal ditulis dengan diawali angka nol (0645), heksadesimal ditulis dengan diawali 0x (contoh : 0x42)
   - Data berupa karakter diapit ‘’ (contoh : ‘a’)
   - Data berupa string diapit “” (contoh : “TES”)
   - Unsigned integer : ditulis dengan diakhiri U (contoh 1000U)
   - Long integer : ditulis dengan diakhiri L (46L)
   - Unsigned long integer : ditulis dengan diakhiri UL (46UL)
   - Floating point : dititulis dengan diakhiri F (contoh 2.56F)

b. Variabel

Variabel adalah tempat untuk menyimpan dan mengakses data yang mewakili memori dalam mikrokontroler. Tabel 2.6 menunjukkan jenis tipe data dan banyaknya bilangan yang dapat ditampung.

Tabel 2.6 Tipe Data

Tipe Data`	Byte	bit	Minimum	Maksimum
Char	1	8	-128	127
Signed char	1	8	-128	127
Unsigned char	1	8	0	255
Int	2	16	-32768	32767
Signed int	2	16	-32768	32767
Unsigned int	2	16	0	65535
Long	3	32	-2147483648	2147483647
Signed long	3	32	-2147483648	2147483647
Unsigned long	3	32	0	4294967295
Float	3	32	1.28E-38	3.4E38

3. Fungsi Utama

Dalam pemrograman, fungsi utama wajib ada. Penulisannya diawali dengan **void main** () kemudian Program atau perintah ditempatkan diantara tanda kurung kurawal buka '{' dan diakhiri dengan kurung kurawal tutup '}'.

4. Operator

Operator adalah karakter-karakter khusus untuk memanipulasi variabel. *Operand* adalah variabel atau konstanta yang merupakan bagian dari pernyataan.

Aritmatika:

- (+) adalah penjumlahan
- pengurangan (-)

- (*) adalah perkalian
- (/) adalah pembagian
- (++) adalah increment
- (--) adalah decrement

Logika:

- (==) adalah logika sama dengan
- (!=) adalah logika tidak sama dengan
- (<) adalah logika lebih kecil
- (<=) adalah logika lebih kecil sama dengan
- (>) adalah logika lebih besar
- (>=) adalah logika lebih besar sama dengan
- (!) adalah logika not
- (&&) adalah logika AND
- (!!) adalah logika OR

Manipulasi Bit:

- (~) adalah mengkomplementkan
- (&) adalah mengANDkan
- (|) adalah mengORkan
- (^) adalah mengXORkan
- (<<) adalah shift left
- (>>) adalah shift right

5. Kontrol Aliran Program

Kontrol aliran program digunakan untuk mencerdaskan sebuah program, dengan cara melakukan pengendalian aliran program.

- If (…) {…}

  Digunakan untuk mengecek satu kondisi satu jawaban.

- If (…) {…} else {…}

  Digunakan untuk mengecek satu kondisi dua blok jawaban.

- If (…) {…} else if (…) {…} else {…}

  Digunakan untuk mengecek beberapa kondisi yang berkaitan.

- While (…) {…}

  Digunakan untuk perulangan/*looping* jika kondisi yang diuji bernilai benar.

- Do {…} while (…)

  Digunakan untuk perulangan/*looping* jika kondisi yang diuji bernilai benar. Pada *while* tanpa *do*, blok dieksekusi dulu baru diuji. Sementara pada *while* dengan *do* blok diuji dulu baru dieksekusi.

- For (… ; … ; … ; …) {…}

  Digunakan untuk perulangan/*looping* jika kondisi dan syarat yang ditentukan.

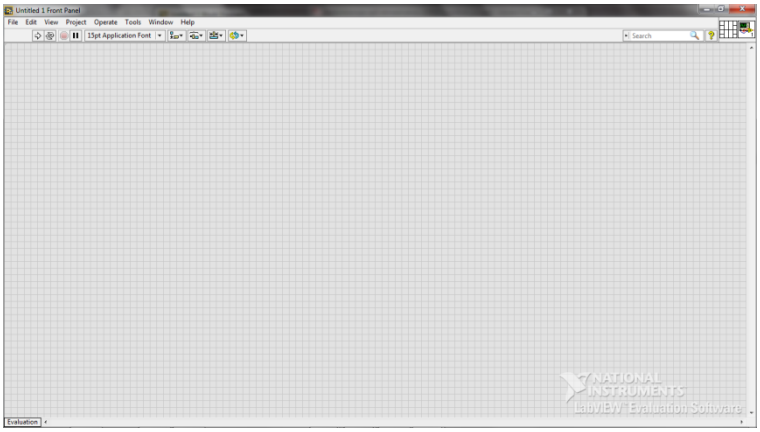
2.11 LabVIEW

LabVIEW adalah sebuah perangkat lunak (*software*) yang diproduksi oleh National Instruments dengan konsep yang berbeda dalam pemrogramannya. Pada dasarnya pemrograman pada LabVIEW berbasis *Graphical User Interface* (GUI) atau

diagram blok, berbeda dengan bahasa pemrograman yang digunakan oleh Matlab atau Visual Basic yang berbasis text. Program LabVIEW dikenal dengan sebutan VI atau *Virtual Instruments* karena penampilan dan operasinya dapat meniru sebuah instrument. LabVIEW terdiri dari 3 bagian utama.

1. *Front Panel*

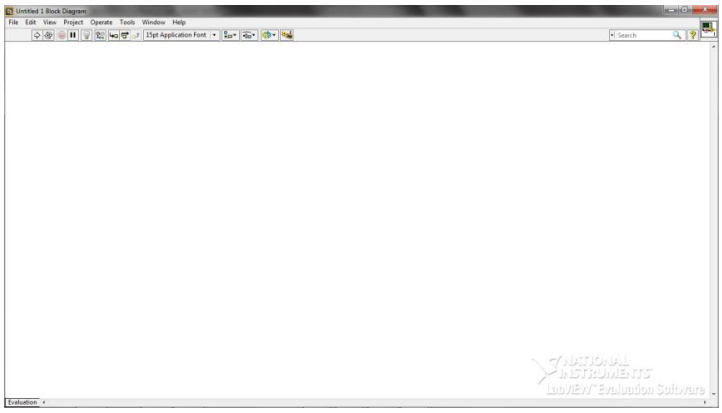
   Bagian *front panel* adalah bagian yang dapat digunakan untuk membangun sebuah VI, menjalankan program dan menghentikan program. Tampilan *front panel* ditunjukkan oleh gambar 2.19.



Gambar 2.19 Tampilan *Front Panel* LabVIEW

2. Blok Diagram dari Vi

   Bagian blok diagram adalah bagian yang berisi *source code* yang telah dibuat dan berfungsi sebagai instruksi untuk *front panel*. Tampilan dari diagram blok ditunjukkan oleh gambar 2.20.



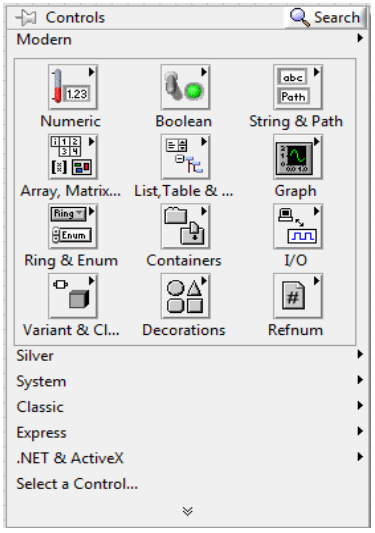
Gambar 2.20 Tampilan Diagram Blok LabVIEW

3. *Control* dan *Functions Pallete*

Bagian *Control* dan *Functions Pallete* adalah bagian yang digunakan untuk membangun sebuah Vi.

a. *Control Pallete*

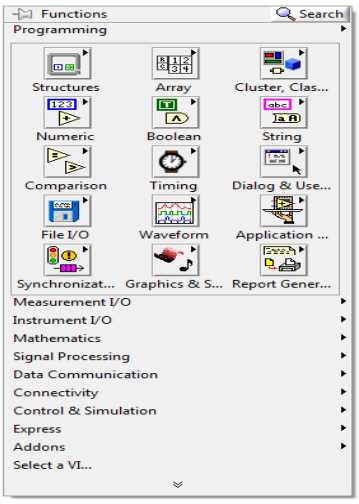
Control Pallete merupakan tempat untuk beberapa *control* dan indikator. *Control* digunakan untuk memasukan *input* ke dalam suatu VI. *Control* dapat dibuat melalui diagram tetapi hanya dapat dimodifikasi melalui *front panel*. Indikator digunakan untuk menampilkan *output*. Indikator dapat dibuat melalui diagram tetapi hanya dapat dimodifikasi melalui *front panel*. Tampilan dari *control pallete* ditunjukkan oleh gambar 2.21.



Gambar 2.21 Tampilan *Control Pallete*

b. *Functions Pallete*

Function Pallete digunakan untuk membangun suatu diagram blok. *Function Pallete* hanya tersedia pada diagram blok. Tampilan *function pallete* ditunjukkan oleh gambar 2.22.



Gambar 2.22 Tampilan *Function Pallete*