

MODUL DASAR ROBOTIKA
<BLAKASUTHA>
KONTEST ROBOT ABU INDONESIA



SOEDIRMAN ROBOTIC TEAM

edisi 2024

**DISCLAIMER => JANGAN DISEBAR KE ORANG LAIN SELAIN ANGGOTA
BLAKASUTHA, KALAU ADA YANG MAKSA MINTA, MINTA LANGSUNG KE
YANG BIKIN. MODUL INI BERISI INTI MATERI SAJA, UNTUK LEBIH
LENGKAP EXPLORE SUMBER INFORMASI SECARA MANDIRI**

DAFTAR ISI

BAB I PENGENALAN ROBOTIKA	7
A. Apa itu Robot	7
BAB II MIKROKONTROLER, KOMPUTER dan HARDWARE PROGRAMMING	9
A. Mikrokontroler dalam Robotika.....	9
a. Apa itu Mikrokontroler?.....	9
b. Peran Mikrokontroler dalam Robotika:.....	9
c. Contoh Mikrokontroler dalam Robotika:	9
B. Komputer dalam Robotika	9
a. Komputer sebagai Unit Pemrosesan Pusat (CPU):	9
b. Fungsi Komputer dalam Sistem Robotik:	9
c. Contoh Komputer dalam Robotika:	10
C. Hardware Programming dalam Robotika.....	10
a. Pemrograman Hardware:.....	10
b. Bahasa Pemrograman yang Digunakan:	10
c. Tool dan IDE untuk Pemrograman:	10
d. Proses Hardware Programming:.....	10
e. Contoh Proyek Robotika dengan Hardware Programming:.....	11
D. Komponen Penyusun Mikrokontroler dan Single Board Computer	11
E. Jenis-Jenis I/O pada Mikrokontroler.....	12
a. USB Connector	12
b. Power Connector	12
c. Pin Digital:	12
d. Pin Analog:.....	12
e. Pin PWM (Pulse Width Modulation):.....	13
f. Interrupt Routine pada Mikrokontroler	13
g. Komunikasi Antar Perangkat	13
F. Contoh Mikrokontroler dan Komputer	16
a. Arduino.....	16
b. Raspberry Pi	18
G. Dasar pemrograman menggunakan Arduino IDE	19

BAB III DASAR ELEKTRONIKA.....	24
A. Elektronika pada Robotika	24
B. Power Supply pada Robotika	25
C. Komponen Elektronika	29
D. Desain dan Simulasi Rangkaian Elektronik	31
BAB IV SENSOR DAN AKTUATOR	37
A. Sensor	37
a. Sensor Posisi dan Orientasi	37
b. Sensor Lingkungan.....	39
c. Sensor Kecepatan dan Percepatan.....	40
d. Sensor Sentuhan (Tactile Sensor)	40
e. Sensor Lain yang Umum Digunakan	41
e. Programming Sensor : Ultrasonic	41
f. Programming Sensor : Proximity IR (Infrared)	43
B. Aktuator.....	44
a. Motor DC	45
b. Power Window	46
c. Pneumatic	46
d. LCD (Liquid Crystal Display).....	48
e. Servo.....	51
BAB V KOMPONEN GERAK ROBOT BERODA DAN ODOMETRI	53
A. Konsep Kemudi dan Roda Pergerakan.....	53
a. Konsep Pergerakan pada Robot Beroda.....	53
b. Jenis-jenis Kemudi pada Robot Beroda	53
c. Jenis-jenis Roda Kemudi pada Robot	55
B. Odometri.....	57
C. Sensor Odometri Dasar	58
a. Rotary Encoder.....	58
b. <i>Inertial Measurement Unit</i> (IMU).....	59
BAB VI PEMODELAN PERGERAKAN ROBOT	62
A. Pengenalan Kinematika.....	62
a. Persamaan Gerak Lurus Beraturan (GLB) dan Gerak Lurus Berubah Beraturan (GLBB).....	63
b. Gerak Rotasi.....	64
d. Kinematika Robot Beroda.....	65

B.	Persamaan Kinematika Menggunakan Konsep Fisika Sederhana	69
BAB VII MEKANIKA BENDA		73
A.	Dasar Mekanika.....	73
B.	Gaya, Momen, dan Torsi.....	73
a.	Gaya (Force).....	73
b.	Momen (Moment)	74
c.	Torsi (Torque)	74
C.	Hukum-hukum Newton dalam Dinamika	75
a.	Hukum Pertama Newton (Hukum Inersia)	75
b.	Hukum Kedua Newton ($F = ma$)	76
c.	Hukum Ketiga Newton (Aksi = Reaksi).....	76
d.	Momen Inersia (Moment of Inertia).....	76
e.	Energi Kinetik dan Energi Potensial	76
f.	Lagrangiang.....	77
g.	Analisis Statik dan Dinamik.....	78
D.	Dinamika Forward dan Inverse	78
BAB VIII ELEMEN MEKANIKA ROBOT		79
A.	Struktur dan Rangka Robot (Robot Frame)	79
B.	Material Kerangka Robot	79
C.	Material Bahan 3D Printing	80
D.	Kekuatan Bahan	81
E.	Mekanisme Transmisi Daya dalam Robotika	82
a.	Roda Gigi (Gear System)	82
b.	Sabuk dan Katrol (Belt and Pulley).....	87
c.	Rantai dan Sproket (Chain and Sprocket)	87
d.	Sistem Lead Screw dan Ball Screw.....	88
e.	Sistem Kopling (Coupling)	88
e.	Faktor-faktor Penting dalam Pemilihan Mekanisme Transmisi Daya	89
F.	Desain End-Effector.....	89
a.	Jenis-Jenis End-Effector	89
b.	Desain Mekanik End-Effector	90
c.	Material yang Digunakan.....	90
d.	Konsiderasi Desain	90
G.	Maintenance (Pemeliharaan) Sistem Mekanik.....	90
a.	Jenis-Jenis Pemeliharaan	90

b. Prosedur Maintenance Sistem Mekanik	91
c. Tips Pemeliharaan.....	91
H. Troubleshooting Sistem Mekanik	91
a. Proses Dasar Troubleshooting	91
b. Masalah Umum dalam Sistem Mekanik.....	92
c. Alat yang Digunakan untuk Troubleshooting.....	92
BAB IX FILTER DATA DAN SISTEM KENDALI.....	93
A. Filter Data.....	93
a. Jenis-jenis Filter Data	93
b. Cara Kerja Filter Data.....	94
c. Aplikasi Filter Data dalam Robotika dan Mekatronika	95
B. Model Sistem Kendali	95
e. Sistem Kendali Terbuka (Open-Loop).....	95
f. Sistem Kendali Tertutup (Closed-Loop).....	96
g. Periode Sistem Kontrol	97
C. On-Off Control	97
D. Proportional Control (P).....	98
E. Integral Control (I).....	98
F. Derivative Control (D).....	98
G. PID Control (Proportional-Integral-Derivative).....	99
H. Bang-Bang Control (Hysteresis Control)	100
I. Adaptive Control.....	100
J. Fuzzy Logic Control.....	100
BAB X ROBOT OPERATING SYSTEM (ROS)	101
A. Pengenalan Robot Operating System (ROS)	101
B. Konsep Dasar ROS.....	101
C. ROS Filesystem (Struktur File ROS)	102
D. ROS Tools	103
E. ROS 1 vs ROS 2	103
F. Contoh Aplikasi ROS	105

BAB I

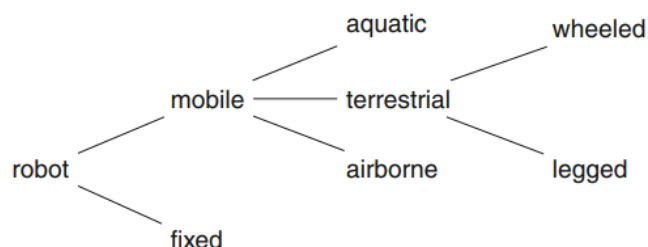
Pengenalan Robotika

A. Apa itu Robot

Meskipun semua orang tampaknya tahu apa itu robot, namun sulit untuk memberikan definisi yang tepat. Kamus Bahasa Inggris Oxford memberikan definisi sebagai berikut: “Sebuah mesin yang mampu melakukan serangkaian tindakan kompleks secara otomatis, terutama yang diprogram oleh komputer.” Definisi ini mencakup beberapa elemen yang menarik:

- “Melaksanakan tindakan secara otomatis.” Ini adalah elemen kunci dalam robotika, tetapi juga dalam banyak mesin lain yang lebih sederhana yang disebut automata. Perbedaan antara robot dan robot sederhana seperti mesin pencuci piring adalah dalam definisi apa itu “kompleks serangkaian tindakan”. Apakah mencuci pakaian terdiri dari serangkaian tindakan yang kompleks atau tidak? Apakah menerbangkan pesawat dengan autopilot merupakan tindakan yang kompleks? Apakah memasak roti termasuk tindakan yang kompleks? Untuk semua tugas ini, ada mesin yang berada di batas antara automata dan robot.
- “Dapat diprogram oleh komputer” adalah elemen kunci lain dari robot, karena beberapa automata diprogram secara mekanis dan tidak terlalu fleksibel. Di sisi lain, komputer ditemukan di mana-mana, sehingga sulit untuk menggunakan kriteria ini untuk membedakan robot dari mesin lain. Elemen penting dari robot yang tidak disebutkan secara eksplisit dalam definisi adalah penggunaan sensor. Kebanyakan robot tidak memiliki sensor dan tidak dapat menyesuaikan tindakan mereka dengan lingkungan mereka. Sensor adalah hal yang memungkinkan robot untuk melakukan tugas-tugas yang sangat kompleks.

B. Jenis-Jenis Robotika



Berdasarkan Fungsi:

- Robot Industri: Digunakan di lini produksi untuk tugas-tugas seperti pengelasan, pengecatan, pengemasan, dan perakitan. Contoh: Robot lengan ABB dan KUKA.
- Robot Mobile: Didesain untuk bergerak di berbagai lingkungan. Ini termasuk robot darat (seperti robot pembersih rumah), robot udara (drone), dan robot bawah air. Contoh: Roomba, DJI Phantom.
- Robot Sosial: Dirancang untuk berinteraksi dengan manusia. Mereka digunakan untuk pendidikan, perawatan lansia, atau sebagai pendamping. Contoh: Pepper, Nao.

- Robot Medis: Digunakan dalam prosedur bedah, rehabilitasi, dan sebagai alat bantu di rumah sakit. Contoh: Da Vinci Surgical System.
- Robot Militer: Digunakan dalam misi pengintaian, penjinakan bom, atau pertempuran. Contoh: Drone Predator, PackBot.
- Robot Eksplorasi: Didesain untuk menjelajahi lingkungan yang sulit atau berbahaya, seperti luar angkasa atau dasar laut. Contoh: Mars Rover, ROV.

Berdasarkan Bentuk Fisik dan Cara Bergerak:

- Robot Lengan (Articulated Robots): Memiliki sendi-sendi yang memungkinkan pergerakan multi-sumbu. Umumnya digunakan dalam industri.
- Robot Humanoid: Menyerupai bentuk dan gerakan manusia. Digunakan untuk interaksi manusia-robot atau penelitian AI. Contoh: ASIMO, Atlas.
- Robot Berkaki: Menggunakan kaki untuk bergerak, biasanya digunakan dalam eksplorasi medan yang tidak rata. Contoh: Boston Dynamics' Spot, BigDog.
- Robot Beroda: Menggunakan roda untuk mobilitas, sering digunakan untuk tugas-tugas yang memerlukan stabilitas dan kecepatan. Contoh: AGV (Automated Guided Vehicle).
- Robot Ular: Memiliki bentuk panjang dan fleksibel, digunakan dalam ruang sempit atau sulit dijangkau. Contoh: Robot ular Carnegie Mellon.
- Robot Swarm: Terdiri dari banyak robot kecil yang bekerja bersama sebagai satu kesatuan. Digunakan dalam operasi penyelamatan atau eksplorasi.

Berdasarkan Metode Kontrol:

- Robot Otomatis: Beroperasi berdasarkan program yang telah ditentukan sebelumnya tanpa interaksi manusia.
- Robot Semi-Otomatis: Memerlukan beberapa bentuk pengawasan manusia selama operasi.
- Robot Otonom: Mampu membuat keputusan sendiri berdasarkan lingkungan dan kondisi saat itu. Dilengkapi dengan sensor dan AI untuk memahami dan beradaptasi.

Pada modul, akan dibahas mengenai dasar-dasar fundamental dalam pembuatan robot terkhusus untuk robot berjenis mobile robot dengan tipe terrestrial beroda.

BAB II

MIKROKONTROLER, KOMPUTER dan HARDWARE PROGRAMMING

A. Mikrokontroler dalam Robotika

a. Apa itu Mikrokontroler?

Mikrokontroler adalah sebuah chip komputer kecil yang berfungsi sebagai otak dari perangkat elektronik. Ia mengintegrasikan prosesor (CPU), memori (RAM, ROM), dan I/O (Input/Output) dalam satu chip. Mikrokontroler digunakan untuk mengontrol perangkat dengan logika dan algoritma yang diimplementasikan dalam kode program.

b. Peran Mikrokontroler dalam Robotika:

- Kendali Aktuator dan Sensor: Mikrokontroler digunakan untuk mengendalikan aktuator seperti motor, servo, dan solenoida, serta membaca data dari sensor seperti sensor jarak, cahaya, suhu, dan lainnya.
- Prosesing Data: Mikrokontroler memproses data yang diterima dari sensor untuk membuat keputusan yang diperlukan dalam menjalankan tugas robot.
- Komunikasi: Mikrokontroler sering digunakan untuk komunikasi antara berbagai komponen robotik atau dengan komputer, menggunakan protokol seperti UART, SPI, I2C, dan lainnya.
- Autonomi: Mikrokontroler memungkinkan robot untuk bekerja secara otonom dengan menjalankan algoritma yang telah diprogram tanpa campur tangan manusia.

c. Contoh Mikrokontroler dalam Robotika:

- Arduino: Platform mikrokontroler yang populer untuk pemula dan profesional. Mudah digunakan dan memiliki banyak library yang mendukung berbagai aplikasi robotika.
- Raspberry Pi Pico: Mikrokontroler berbasis ARM yang lebih kuat dengan kemampuan pemrograman yang fleksibel.
- STM32: Mikrokontroler yang banyak digunakan dalam aplikasi industri dengan performa tinggi dan fitur lengkap.

B. Komputer dalam Robotika

a. Komputer sebagai Unit Pemrosesan Pusat (CPU):

Dalam robotika, komputer sering digunakan sebagai unit pemrosesan pusat untuk tugas-tugas yang membutuhkan komputasi lebih berat dibandingkan dengan mikrokontroler. Komputer bisa mengontrol mikrokontroler atau bekerja bersamanya dalam sistem robotik yang kompleks.

b. Fungsi Komputer dalam Sistem Robotik:

- Pemrosesan Data dan Sensor Fusion: Komputer digunakan untuk memproses data dari berbagai sensor dalam waktu nyata, menggabungkan informasi dari berbagai sumber untuk membuat keputusan yang lebih tepat.
- Visi Komputer: Banyak sistem robotik yang menggunakan komputer untuk menjalankan algoritma visi komputer, seperti pengenalan objek, navigasi berbasis gambar, dan pemetaan.

- Pemrograman dan Pengembangan: Komputer digunakan untuk menulis, menguji, dan mengunggah kode ke mikrokontroler atau sistem robotik lainnya.
- Simulasi dan Pemodelan: Komputer memungkinkan simulasi robotik untuk menguji algoritma dan desain sebelum diimplementasikan di dunia nyata.

c. Contoh Komputer dalam Robotika:

- Raspberry Pi: Komputer mini yang sering digunakan dalam proyek robotika untuk tugas-tugas yang memerlukan lebih banyak daya komputasi, seperti pengolahan gambar dan AI, ada juga jenis lain seperti Jetson series, Beagle Bone dll.
- Laptop atau PC: Digunakan dalam tahap pengembangan dan pemrograman, serta untuk mengontrol robot secara jarak jauh atau melalui jaringan.

C. Hardware Programming dalam Robotika

a. Pemrograman Hardware:

Apa itu Hardware Programming? Ini merujuk pada proses menulis dan mengunggah kode ke perangkat keras seperti mikrokontroler dan komputer. Dalam robotika, ini melibatkan pengaturan pin, konfigurasi modul komunikasi, pengendalian aktuator, dan pembacaan sensor.

b. Bahasa Pemrograman yang Digunakan:

- C/C++: Bahasa yang paling umum digunakan untuk mikrokontroler karena efisiensi dan kontrol langsung terhadap perangkat keras.
- Python: Sering digunakan dengan Raspberry Pi dan untuk pengembangan algoritma yang lebih kompleks seperti machine learning.
- Assembly: Digunakan dalam situasi di mana efisiensi sangat penting, meskipun jarang digunakan dalam pengembangan robotika tingkat tinggi.

c. Tool dan IDE untuk Pemrograman:

- Arduino IDE: Platform pemrograman open-source yang populer untuk mikrokontroler Arduino. Sangat cocok untuk pemula dan menawarkan banyak contoh proyek robotika.
- PlatformIO: IDE yang lebih canggih untuk mikrokontroler yang mendukung berbagai board dan framework.
- Visual Studio Code: Sering digunakan bersama dengan berbagai ekstensi untuk pengembangan robotika dan hardware programming.
- Keil uVision: Digunakan untuk memprogram mikrokontroler STM32 (bisa juga menggunakan STM32 Cube IDE) dan ARM Cortex-M dengan fitur debugging yang kuat.

d. Proses Hardware Programming:

- Penulisan Kode: Dimulai dengan menulis kode dalam bahasa pemrograman yang sesuai. Kode ini mengatur bagaimana mikrokontroler mengendalikan robot.
- Kompilasi dan Unggah: Kode kemudian dikompilasi menjadi bentuk yang dapat dimengerti oleh mikrokontroler dan diunggah ke dalamnya menggunakan programmer atau kabel USB.

- **Testing dan Debugging:** Setelah diunggah, kode diuji untuk memastikan semua fungsi robot berjalan sesuai yang diharapkan. Debugging dilakukan untuk memperbaiki kesalahan atau menyesuaikan algoritma.

e. Contoh Proyek Robotika dengan Hardware Programming:

- **Line Following Robot:** Menggunakan Arduino untuk membaca sensor garis dan mengendalikan motor DC agar robot mengikuti jalur yang ditentukan.
- **Obstacle Avoidance Robot:** Menggunakan mikrokontroler untuk membaca sensor jarak dan menghindari rintangan secara otomatis.
- **Remote-Controlled Car:** Menggunakan Raspberry Pi untuk menerima input dari remote control dan mengendalikan mobil robotik.

Dengan memahami dasar-dasar mikrokontroler, komputer, dan hardware programming, kita dapat memulai perjalanan dalam pengembangan sistem robotik yang fungsional dan inovatif.

D. Komponen Penyusun Mikrokontroler dan Single Board Computer

a. Komponen Penyusun Mikrokontroler

Mikrokontroler adalah sistem terintegrasi yang menggabungkan berbagai komponen dalam satu chip untuk mengendalikan perangkat elektronik. Berikut adalah komponen utama dari mikrokontroler:

- **CPU (Central Processing Unit) :** Otak dari mikrokontroler yang menjalankan instruksi dari program yang disimpan dalam memori. CPU mengatur operasi perangkat dan melakukan perhitungan aritmatika dan logika.

b. Memori:

- **ROM (Read-Only Memory):** Memori ini digunakan untuk menyimpan program yang akan dijalankan oleh mikrokontroler. Pada beberapa mikrokontroler, ROM bisa berupa Flash memory yang bisa diprogram ulang.
- **RAM (Random Access Memory) :** Memori ini digunakan untuk menyimpan data sementara selama operasi program, seperti variabel dan data sensor.
- **EEPROM (Electrically Erasable Programmable Read-Only Memory) :** Memori ini digunakan untuk menyimpan data yang harus dipertahankan setelah daya dimatikan, seperti konfigurasi atau pengaturan, namun perlu diperhatikan bahwa penggunaan EEPROM pada mikrokontroler mempunyai Batasan jumlah read dan write nya.

- c. I/O Ports (Input/Output Ports):** Pin-pin yang digunakan untuk menerima sinyal dari sensor (input) atau mengirim sinyal ke aktuator (output). Pin ini bisa berupa pin digital, analog, atau PWM.

- d. Timer/Counters:** Modul yang digunakan untuk mengukur waktu, menghitung peristiwa, atau menghasilkan sinyal PWM. Timer sangat penting dalam aplikasi yang memerlukan waktu presisi, seperti kontrol motor atau komunikasi serial.

- e. ADC (Analog-to-Digital Converter):** Mengubah sinyal analog dari sensor menjadi sinyal digital yang dapat diproses oleh mikrokontroler.

- f. **DAC (Digital-to-Analog Converter):** Mengubah sinyal digital menjadi sinyal analog, meskipun fitur ini tidak selalu ada di semua mikrokontroler.
- g. **Komunikasi Serial:** Modul yang memungkinkan mikrokontroler berkomunikasi dengan perangkat lain melalui protokol seperti UART, I2C, dan SPI.

E. Jenis-Jenis I/O pada Mikrokontroler

a. USB Connector

USB Connector diperlukan untuk 3 hal, yaitu:

- Memberikan supply power ke board
- Meng-upload code ke Mikrokontroler, alternatif lain menggunakan Serial Wire Debug (SWD) melalui 4 pin yang biasanya terletak terpisah dari I/O lain di mikrokontroler (ada VCC, GND, SWDIO dan SWCLK)
- Mengirim dan menerima data dari dan/atau ke komputer secara serial melalui protokol UART

b. Power Connector

Power Connector digunakan untuk catu daya arduino dan supply dari arduino dengan Pin sebagai berikut:

- VIN: Pin ini digunakan untuk menerima input tegangan ke arduino board saat menggunakan sumber tegangan dari luar.
- 5V: Mengeluarkan tegangan konstan 5V.
- 3.3V: Mengeluarkan tegangan konstan 3.3V.
- GND: Pin Ground.
- IOREF: Pin ini mengeluarkan voltage reference yang digunakan oleh microcontroller saat bekerja.

c. Pin Digital:

- Pengertian: Pin digital bekerja dengan dua kondisi, yaitu HIGH (1) dan LOW (0), HIGH berarti memberikan/menerima 5 volt atau 3.3 volt pada pin tersebut, dan ketika off atau LOW maka memberikan/menerima tegangan 0 volt pada pin tersebut. Pin ini digunakan untuk membaca atau mengirim sinyal biner ke atau dari perangkat lain.
- Contoh Aplikasi: Menyalakan/mematikan LED, membaca status tombol (on/off).

d. Pin Analog:

- Pengertian: Pin analog digunakan untuk membaca sinyal analog, yaitu sinyal yang memiliki nilai kontinu. Mikrokontroler dengan ADC mengkonversi sinyal analog ini menjadi nilai digital untuk diproses.
- Contoh Aplikasi: Membaca nilai sensor suhu, sensor jarak, atau potensiometer.

e. Pin PWM (Pulse Width Modulation):

- Pengertian: PWM adalah teknik modulasi (menumpangkan sebuah sinyal yang berisi informasi ke sinyal lain (sinyal pembawa) yang mempunyai komponen yang lebih tinggi dari sinyal informasi, bisa dari aspek amplitude, frekuensi dan lain sebagainya) dengan melakukan perubahan lebar pulsa digital yang dikeluarkan oleh pin. Sebagai Gambaran, sebuah sinyal kotak yang terdiri dari kondisi HIGH dan LOW yang terus berulang-ulang dalam frekuensi tertentu, saat diberikan teknik modulasi lebar pulsa pada sinyal tersebut. Maka yang berubah dari sinyal adalah rasio dari periode HIGH sinyal dan LOW sinyal dalam 1 buah sinyal kotak (1 fase HIGH dan 1 fase LOW). Akibatnya apa ? maka tegangan rata-rata yang dihasilkan oleh sinyal pastilah berubah ubah bergantung pada rasio fase HIGH/LOW nya atau yang disebut dengan duty cycle (0-100 %).
- Contoh Aplikasi: Mengontrol kecepatan motor DC, kecerahan LED.

f. Interrupt Routine pada Mikrokontroler

Pengertian:

- Interrupt: Interrupt adalah sinyal yang menghentikan sementara eksekusi program utama untuk menangani suatu peristiwa tertentu. Setelah peristiwa ditangani, program utama dilanjutkan dari titik terakhir yang dieksekusi.
- Interrupt Service Routine (ISR): Adalah fungsi yang dipanggil ketika interrupt terjadi. ISR mengeksekusi tugas tertentu sebelum mengembalikan kontrol ke program utama.

Cara Kerja:

- Trigger: Interrupt dapat dipicu oleh berbagai peristiwa, seperti perubahan status pin digital (misalnya, tombol ditekan), overflow timer, atau sinyal dari perangkat komunikasi.
- Prioritas: Jika ada beberapa interrupt yang terjadi secara bersamaan, mikrokontroler menggunakan sistem prioritas untuk menentukan interrupt mana yang harus ditangani terlebih dahulu.

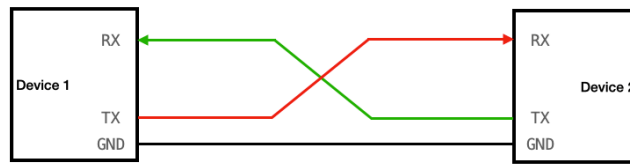
Contoh Aplikasi:

- Penghitung Pulsa: Menggunakan interrupt untuk menghitung jumlah pulsa dari sensor enkoder.
- Deteksi Tombol: Menggunakan interrupt untuk mendeteksi ketika tombol ditekan, sehingga mikrokontroler tidak perlu terus-menerus memeriksa status tombol.

g. Komunikasi Antar Perangkat

Dalam melakukan tugas yang diberikan, sering kali sebuah piranti elektronik tidak dapat untuk bekerja sendiri dan membutuhkan perangkat lain untuk membantunya. Oleh karena itu, sebuah piranti harus mempunyai protokol komunikasi yang bisa menjembatani antara beberapa piranti agar bisa saling bertukar informasi. Beberapa diantaranya yaitu :

- **UART (Universal Asynchronous Receiver/Transmitter):**

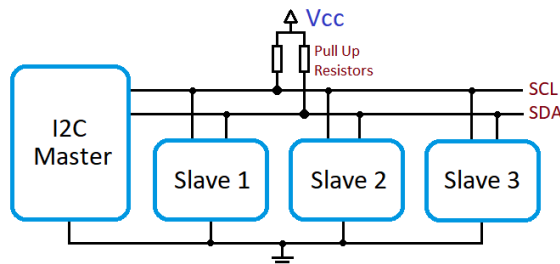


- Pengertian: UART adalah protokol komunikasi serial yang bekerja secara asinkron, artinya tidak memerlukan sinyal clock bersama. Data dikirim bit per bit melalui satu jalur data. (bit => binary digit, yakni satuan terkecil dalam sistem penyimpanan komputasi).

- Cara Kerja: Data dikirim dalam bentuk frame yang terdiri dari bit start, data, bit paritas (opsional, bit yang digunakan untuk mengecek apakah data terkirim dengan benar atau tidak), dan bit stop. Kecepatan transfer data diatur dengan baud rate yang harus sama di kedua perangkat. Untuk melakukan komunikasi, diperlukan 2 buah jalur data/kabel (TX dan RX), TX adalah jalur kabel untuk mengirimkan informasi sedangkan RX untuk menerima informasi.

- Contoh Aplikasi: Komunikasi antara mikrokontroler dan modul GPS, atau antara dua mikrokontroler.

- **I2C (Inter-Integrated Circuit):**

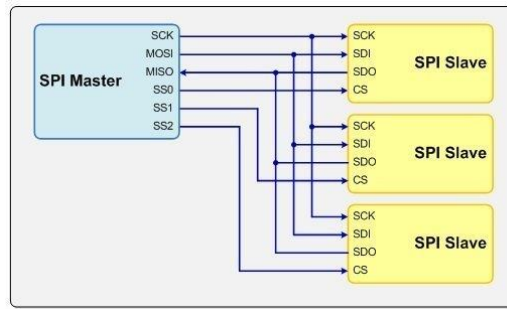


- Pengertian: I2C adalah protokol komunikasi serial yang bekerja secara sinkron menggunakan dua jalur, yaitu SDA (Serial Data Line) dan SCL (Serial Clock Line).

- Cara Kerja: I2C menggunakan satu master dan banyak slave. Master mengontrol clock (SCL) dan memulai komunikasi, sedangkan data dikirim melalui SDA. Setiap perangkat di jaringan I2C memiliki alamat unik.

- Contoh Aplikasi: Komunikasi antara mikrokontroler dan sensor atau modul lain seperti RTC (Real-Time Clock), EEPROM, atau sensor jarak.

- **SPI (Serial Peripheral Interface):**



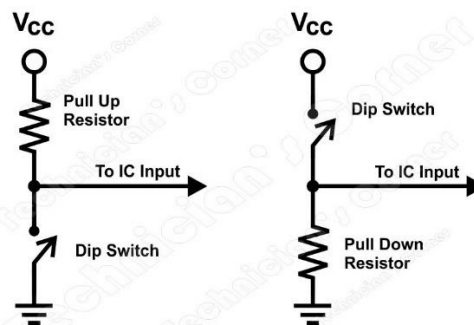
- Pengertian: SPI adalah protokol komunikasi serial yang bekerja secara sinkron dan menggunakan empat jalur utama: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), dan SS (Slave Select).

- Cara Kerja: SPI bekerja dengan satu master dan satu atau lebih slave. Master menghasilkan sinyal clock dan memilih slave tertentu dengan menurunkan sinyal SS (Slave Select). Data dikirim secara simultan antara master dan slave melalui jalur MOSI dan MISO.

- Contoh Aplikasi: Komunikasi dengan modul display (LCD, OLED), kartu SD, atau sensor yang memerlukan kecepatan transfer data tinggi.

- **Pull-up dan Pull-down Resistor:**

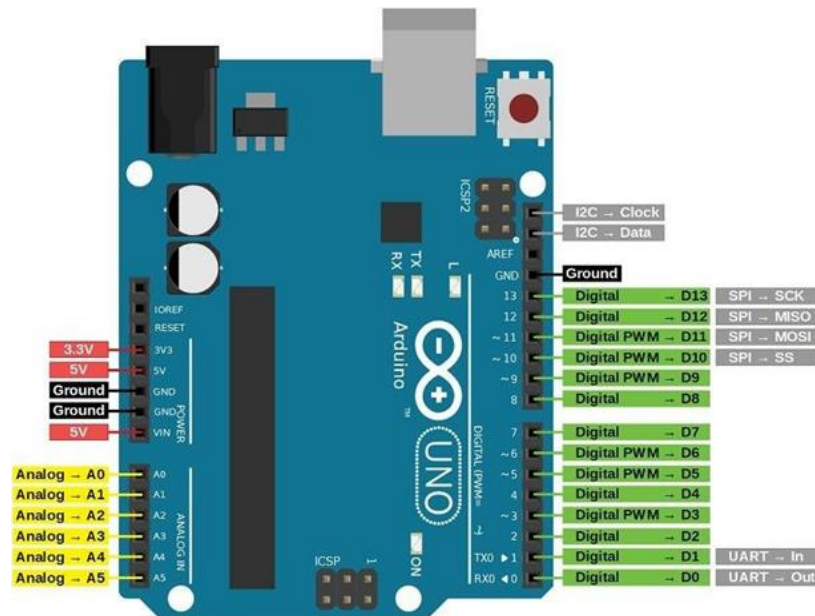
Pada jalur komunikasi digital seperti I2C dan SPI, pull-up atau pull-down resistor digunakan untuk menjaga tegangan pada keadaan tertentu ketika jalur tidak aktif. Contohnya, pada I2C, resistor pull-up sering digunakan pada jalur SDA dan SCL untuk menjaga tegangan tetap tinggi ketika tidak ada transmisi data.



Dari ketiga protokol yang telah dijelaskan jika dikomparasi berdasarkan kecepatan transfer datanya, maka SPI adalah yang tertinggi, dilanjut dengan I2C lalu yang paling lambat yakni UART. Dengan memahami komponen, jenis I/O, interrupt, dan komunikasi antar mikrokontroler, kita dapat merancang dan mengimplementasikan sistem kontrol robotika yang efisien dan responsif sesuai kebutuhan aplikasi.

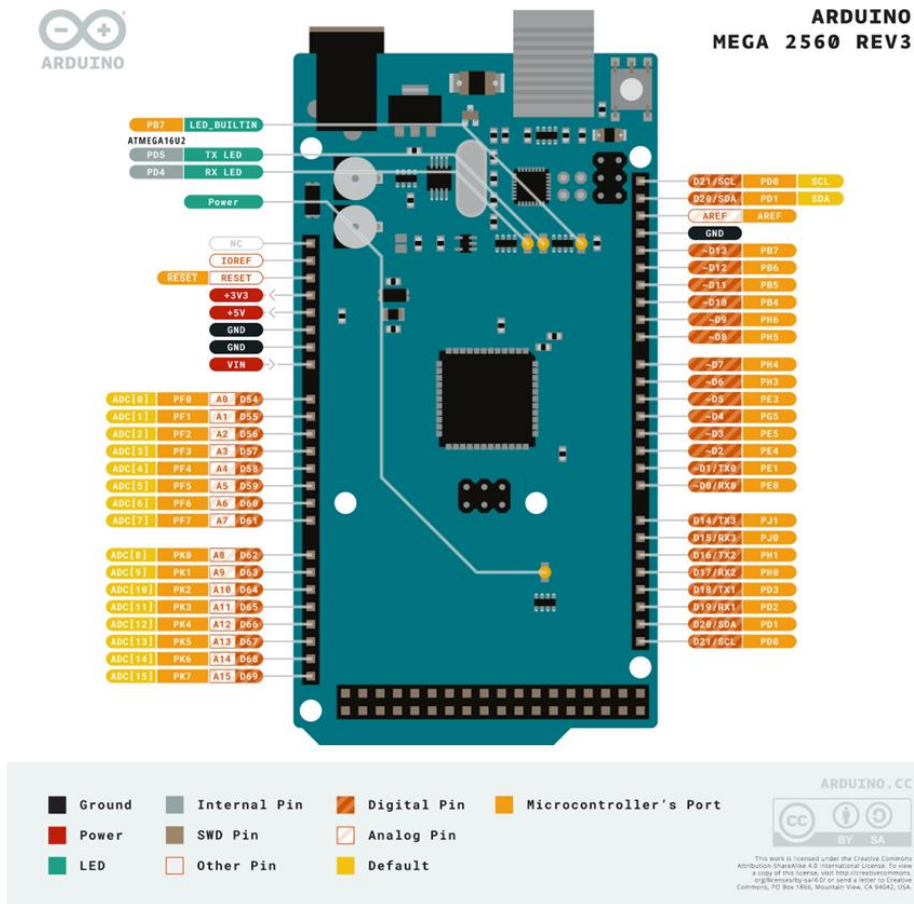
F. Contoh Mikrokontroler dan Komputer

a. Arduino



Pada Arduino Uno terdapat 31 header pin, dimana pin tersebut bisa diatur untuk melakukan fungsi atau tugas yang berbeda. Arduino Uno board berbasis pada mikrokontroler ATmega328. Di dalam Arduino Uno terdapat:

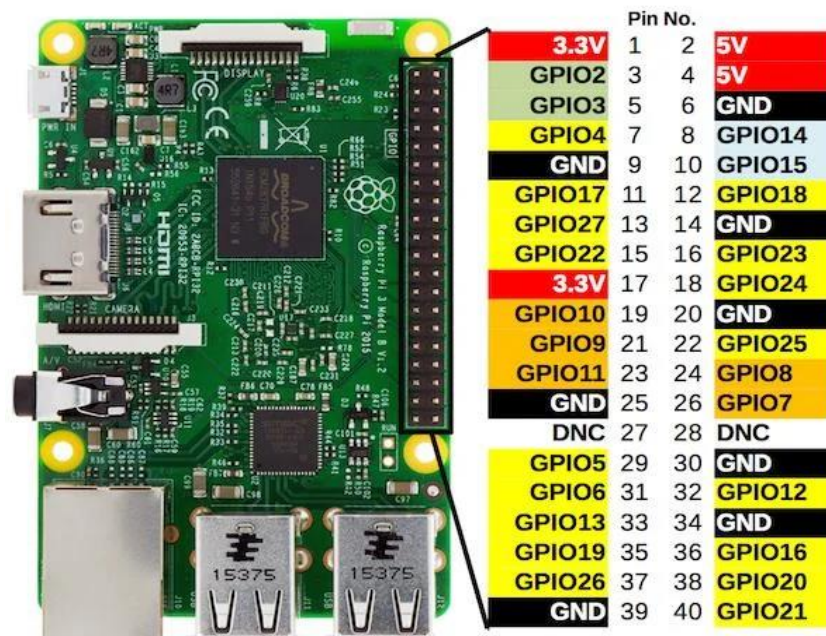
- 16MHz clock,
- 14 digital input/output pins (6 diantaranya merupakan pin PWM),
- 6 analog inputs,
- ICSP header,
- USB connection,
- Power jack
- Reset button, and
- on board LEDs



Arduino Mega board berbasis pada mikrokontroler ATmega2560. Di dalam Arduino Mega terdapat:

- 16MHz clock,
- 54 digital input/output pins (15 diantaranya merupakan pin PWM),
- 16 analog inputs,
- ICSP header,
- USB connection,
- Power jack
- Reset button, and
- on board LEDs

b. Raspberry Pi



Raspberry Pi adalah sebuah SBC (Single Board Computer) seukuran telapak tangan yang dikembangkan oleh Yayasan Raspberry Pi di Inggris (UK) dengan maksud untuk memicu pengajaran ilmu komputer dasar di sekolah-sekolah. Raspberry Pi juga dilengkapi dengan pin-pin seperti GPIO (General Purpose Input-Output), ground, 3.3V, dan 5V.

Dalam menjalankan Raspberry Pi secara baik diperlukan beberapa komponen atau media tambahan, diantaranya:

a. SD Card

SD Card atau kartu memori digunakan sebagai media penyimpanan data pada Raspberry Pi. SD Card juga dibutuhkan untuk men-loading sistem operasi dan kapasitas minimal SD Card adalah 4 GigaByte.

b. Sistem Operasi / Operating System

Semua perangkat pasti memerlukan sistem operasi sebagai jembatan antara hardware dan brainware serta menjadi interface hardware dan aplikasi. Sebagian besar sistem operasi Raspberry Pi adalah berbasis linux bukan windows. Contoh: debian, kali linux, fedora, dan lain lain. Pada umumnya raspberry pi menggunakan sistem operasi yang dibuat khusus yaitu Raspberry Pi OS.

c. Mouse dan Keyboard USB

Mouse dan keyboard USB adalah suatu komponen standar komputer yang harus ada karena sebagai media input. Dianjurkan menyediakan USB hub tambahan karena USB pada Raspberry Pi hanya 4 slot.

d. Monitor

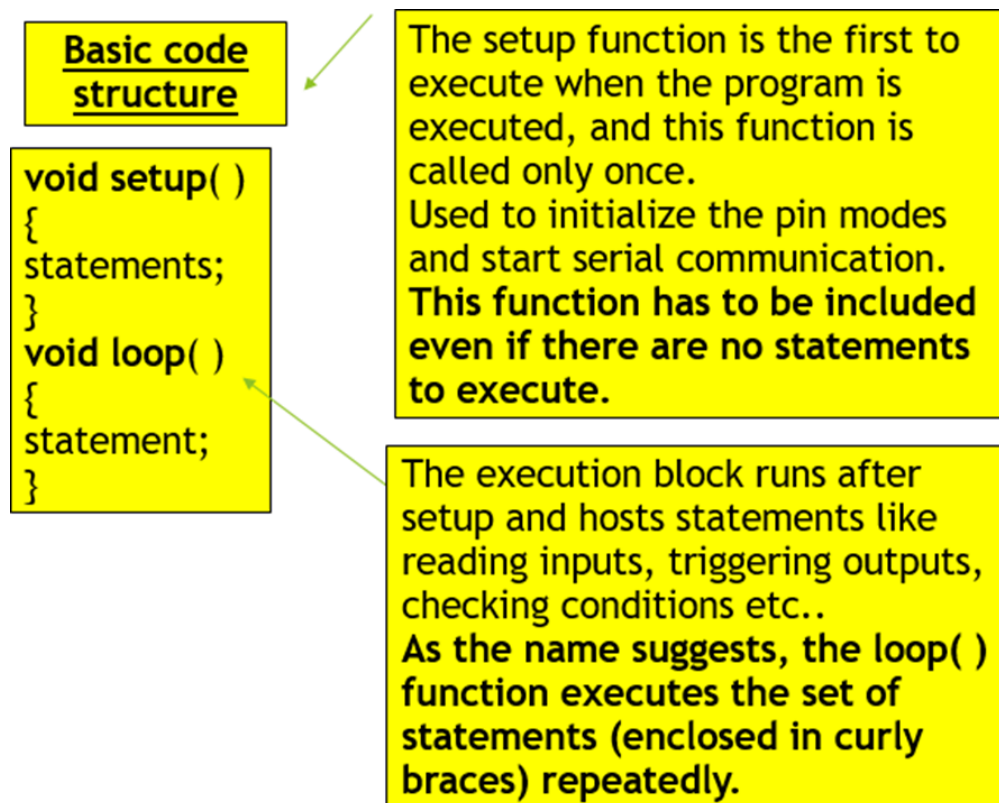
Monitor digunakan sebagai media output Raspberry Pi. Ada dua port tersedia, yaitu port HDMI dan Mikro HDMI.

e. Power supply

Raspberry Pi menggunakan kabel power supply mini USB. Kabel ini dapat dengan mudah diperoleh tanpa harus beli lagi karena kabel charger handphone seperti Samsung atau Blackberry bisa dipakai pada Raspberry Pi. Power Supply ini harus memiliki arus minimal 700 mA dan tegangan 5V. Pada bidang robotika biasanya power supply raspberry pi dihubungkan ke baterai melalui modul khusus sehingga mudah untuk dipakai.

G. Dasar pemrograman menggunakan Arduino IDE

Program yang digunakan dalam Arduino IDE adalah hardware programming language yang disebut processing, dan bahasanya mirip seperti bahasa C/C++. Berikut adalah struktur Pemrograman pada Arduino IDE:



Hal yang penting dalam programming adalah memahami bagaimana codenya terstruktur. Program akan berjalan dari atas ke bawah, namun program juga bisa “lompat” ke bagian program yang lain apabila diperintah.

Saat pertama kita membuka Arduino IDE kita akan melihat 2 fungsi khusus dan kosong secara default, yaitu:

void setup()

void loop()

Void setup adalah hal pertama yang dijalankan oleh program, semua program yang ada didalam void setup function akan dijalankan hanya sekali, dan ketika mencapai baris akhir, program akan lanjut berjalan ke fungsi selanjutnya yaitu void loop function. Void loop function akan dijalankan terus menerus hingga perangkat dimatikan.

Sebelum melakukan pemrograman kita harus mengetahui indentasi dan fungsinya. Umumnya dalam pemrograman, setiap level atau nest dalam pemrograman di indentasikan ke kanan dibanding level sebelumnya. Contohnya void setup dan void loop berada pada indentasi 0.

Hal selanjutnya adalah semicolon atau titik koma. Semicolon adalah terminating character yang digunakan oleh compiler untuk membedakan setiap baris dalam program.

Variabel

Variabel digunakan untuk menyimpan informasi di dalam program, variable memiliki: Nama, Data type, serta membutuhkan memory. Variabel bisa digunakan untuk menyimpan value, misalnya dari sebuah sensor atau input. Untuk membuat variabel bisa menggunakan format sebagai berikut

```
Data_type Variable_name = value;
```

```
Int namaVariable = 32;
```

Kita juga bisa membuat variabel tanpa harus memberikan value, secara default value nya akan bernilai 0.

Hal penting lain adalah ada 2 jenis variabel, yaitu Global variable dan Local variable. Global variabel didefinisikan di luar fungsi, dan nilainya dapat diakses dari semua fungsi. Local variabel didefinisikan di dalam sebuah fungsi, dan hanya dapat diakses dari fungsi atau level tersebut. Dari segi sistem, global variable membutuhkan lebih banyak memory, sedangkan local variable dibuat saat dideklarasikan dan dihapus saat sebuah fungsi telah complete.

Tipe yang umum digunakan dalam Arduino adalah:

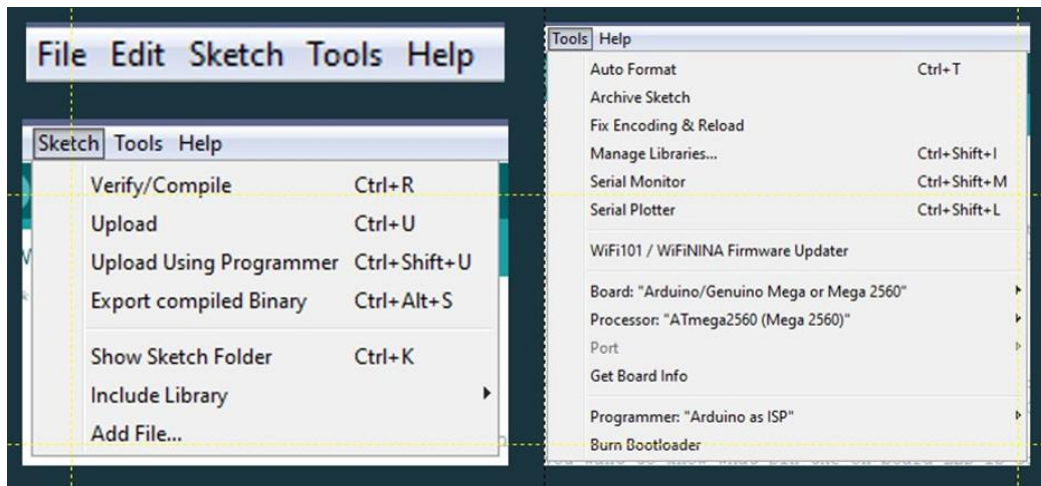
- Int : Tipe data Integer digunakan untuk menyimpan angka
- Float: Tipe data yang digunakan untuk menyimpan angka dengan titik desimal
- Long: Tipe data long digunakan untuk menyimpan angka dalam nilai yang besar dan tipe data yang lain bisa dibuka pada web berikut

https://www.tutorialspoint.com/arduino/arduino_data_types.htm

Arduino juga memiliki beberapa konstanta yang sudah didefinisikan nilainya secara default yaitu:

- HIGH yang berarti menyala, pin tersebut akan diberikan tegangan 5/3.3 volts
- LOW yang berarti mati, pin tersebut akan diberikan tegangan 0 volts
- INPUT pin sebagai input
- OUTPUT pin sebagai output

Pengenalan Arduino IDE



Arduino IDE adalah software yang digunakan untuk menulis program, lalu di test / compile, serta mengupload program ke arduino board.

Pada software IDE arduino di bagian atas terdapat tab menu yaitu: File, Edit, Sketch, Tools, Help. Untuk File dan Edit kurang lebih sama seperti teks editor notepad, file untuk tentang lokasi, save as, dll. Edit untuk melakukan edit teks pada IDE.

Selanjutnya menu yang sering digunakan pada saat memprogram arduino adalah: Sketch dan Tools. Pada bagian sketch ada menu Verify / Compile, untuk melakukan compile pada program. Upload untuk mengirim program yang telah di compile ke arduino. Menu selanjutnya include library untuk menginstall library tambahan yang digunakan pada sensor tertentu. Add file biasanya untuk OOP.

Pada arduino ide di menu tools juga terdapat fitur serial monitor, sehingga kita bisa membaca data dari arduino yang telah diprogram melalui port COM , Port COM itu sendiri adalah sebuah Interface I/O yang digunakan untuk koneksi serial devices ke komputer.

Langkah Upload Program ke Arduino

1. Membuka arduino IDE yang telah di download dan di Install
2. Install driver apabila dibutuhkan (opsional)
3. Tulis program yang ingin digunakan
4. Klik tombol verify atau compile untuk mengecek program apakah ada error atau tidak

Langkah pemasangan:

1. Hubungkan kabel USB ke arduino board dan PC.
2. Setup serial port. Maksudnya adalah memilih port usb yang terhubung ke arduino.
3. Setup arduino board. Maksudnya adalah memilih jenis board yang saat ini sedang digunakan.
4. Klik tombol upload untuk mengirim program ke arduino.

Konsep Input dan Output pada arduino

Pada dasarnya input digunakan untuk mendapatkan suatu arus listrik untuk melakukan sesuatu, contoh paling mudah dari input ini adalah Push Button. Sedangkan Output, digunakan untuk mengolah arus dari mikrokontroler menjadi suatu energi tertentu, contoh paling mudah yaitu LED. Sinyal input pada arduino terbagi dalam dua yakni digital dan analog, sedangkan output dari arduino hanya berupa sinyal digital. Digital berarti sinyal yang dikirimkan/diterima bernilai 1 atau 0, on atau off, HIGH atau LOW, ada atau tidak ada sinyal. Berbeda dengan sinyal analog yang nilainya bersifat kontinyu, yakni nilai antara 0 dan 1 dipertimbangkan. Pin digital berarti pin dapat menerima/mengirim sinyal digital.

Untuk mengeset mode pin, kita gunakan fungsi *pinMode()*. Fungsi ini biasanya dipanggil di dalam fungsi *setup()*. fungsi ini memerlukan dua parameter, *pinMode([nomorPin], [mode])*. Parameter pertama diisi oleh nomor pin, dan parameter kedua diisi oleh konstanta INPUT atau OUTPUT, sesuai dengan mode yang ingin kita gunakan. Sebagai contoh, lihat pada kode berikut. Untuk menerima input digital yang masuk ke pin, kita gunakan fungsi *digitalRead(nomorPin)*. Kemudian Pada OUTPUT digital untuk mengirimkan sinyal, kita gunakan fungsi *digitalWrite(nomorPin, nilaiDigital)*.

Pin analog memiliki fitur untuk dapat mengubah sinyal analog yang masuk menjadi nilai digital yang mudah diukur. Pin analog dapat mengenali sinyal pada rentang nilai voltase tersebut. Hal ini sangat berguna ketika kita hendak mengukur sesuatu dari sensor dan menggunakan nilai masukan tersebut untuk keperluan lain. Pin analog ini terhubung dengan converter pada mikrokontroler yang dikenal dengan istilah analog-to-digital converter (disingkat ADC atau A/D). Converter ini mengubah nilai analog berbentuk sinyal voltase ke dalam bentuk digital/angka supaya nilai analog ini dapat digunakan dengan lebih mudah dan aplikatif.

Untuk menerima input analog yang masuk ke pin, kita gunakan fungsi *AnalogRead(nomorPin)*. Untuk lebih detail terkait source code bagaimana aplikasi I/O pada Arduino termasuk penggunaan pin PWM, Interrupt, Timer bisa kalian akses pada official website Arduino. Search pada google keyword terkait apa yang kalian ingin tahu dengan menyertakan kata Arduino didalamnya.

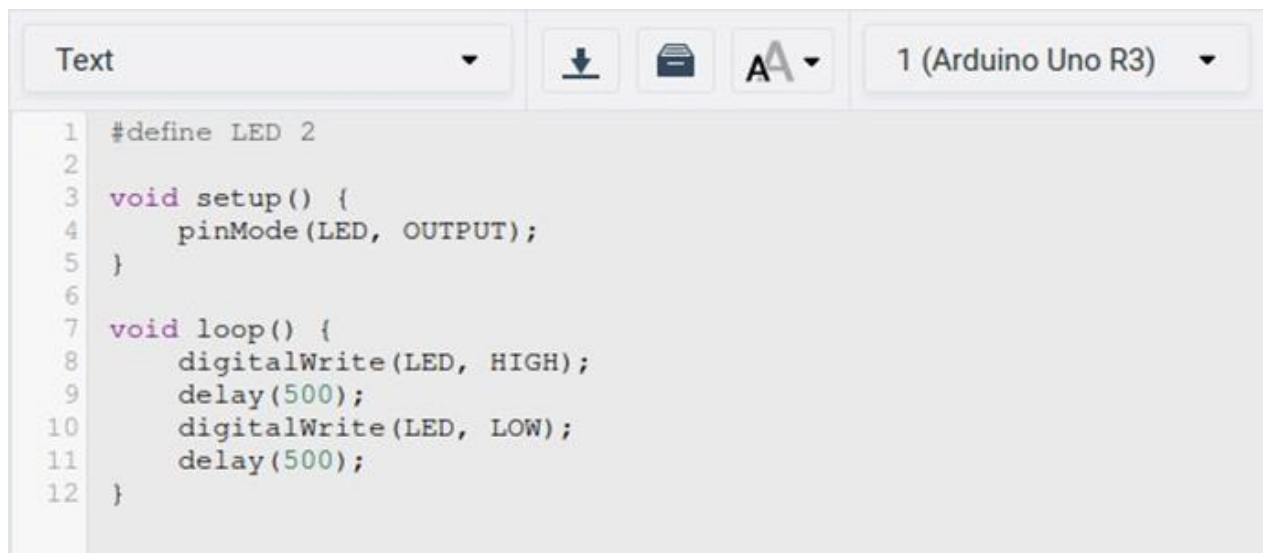
Projek arduino dengan tinkercad

Tinkercad merupakan sebuah program simulasi komponen elektronika berbasis web. Dengan memanfaatkan tinkercad, buatlah sebuah percobaan simulasi percobaan lampu LED berkedip!

Langkah-langkah

1. Buka website tinkercad.com.
2. Login pada tinkercad (Registrasi apabila belum memiliki akun).
3. Klik tombol “new” dan pilih “circuit”.

4. Pilih komponen Arduino Uno R3 dan lampu LED (warna bebas), dan letakkan di workspace.
5. Hubungkan katoda LED ke pin GND pada Arduino dan anoda LED ke pin D2 pada Arduino.
6. Klik tab “Code” dan ubah Edit Mode dari Blocks ke Text.
7. Ketikkan kode berikut dan klik “Start Simulation”!



The screenshot shows the Arduino IDE interface. At the top, there is a toolbar with icons for download, upload, and font size adjustment. To the right of the toolbar, a dropdown menu shows '1 (Arduino Uno R3)'. Below the toolbar, the code editor displays the following C++ code:

```
1  #define LED 2
2
3  void setup() {
4      pinMode(LED, OUTPUT);
5  }
6
7  void loop() {
8      digitalWrite(LED, HIGH);
9      delay(500);
10     digitalWrite(LED, LOW);
11     delay(500);
12 }
```

BAB III

DASAR ELEKTRONIKA

A. Elektronika pada Robotika

Elektronika dalam robotika memainkan peran penting dalam menghubungkan berbagai komponen seperti sensor, aktuator, dan mikrokontroler agar robot dapat berfungsi dengan baik. Pada dasarnya, sistem elektronika menyediakan infrastruktur listrik yang diperlukan untuk mengatur daya, mengolah sinyal, dan mengendalikan komponen fisik dalam robot. Sebelum melangkah lebih jauh, mari kita pahami apa itu elektronika dalam konteks robotika dan perannya dalam sebuah sistem robot.

Definisi Elektronika dalam Robotika

Elektronika adalah cabang ilmu teknik yang mempelajari aliran dan kontrol elektron melalui perangkat semikonduktor seperti transistor, dioda, dan IC (Integrated Circuit). Dalam robotika, elektronika bertanggung jawab untuk:

- **Menyuplai daya:** Elektronika memastikan komponen robot mendapatkan daya yang dibutuhkan untuk beroperasi.
- **Mengolah sinyal:** Sensor-sensor pada robot menghasilkan sinyal yang harus diolah dan diterjemahkan menjadi data yang berguna oleh sistem kontrol.
- **Mengendalikan aktuator:** Elektronika bertindak sebagai penghubung antara mikrokontroler dan aktuator seperti motor atau servo, yang memungkinkan robot melakukan tugas-tugas fisik.

Peran Elektronika dalam Sistem Robotika

Sistem robotika memiliki tiga elemen utama: **sensor**, **kontrol**, dan **aktuator**. Elektronika adalah "jembatan" yang menghubungkan ketiga elemen ini, memastikan informasi dapat mengalir dengan benar dari satu elemen ke elemen lainnya.

- **Sensor:** Alat yang digunakan untuk mendeteksi perubahan lingkungan (seperti suhu, cahaya, jarak, dll.) dan mengubahnya menjadi sinyal listrik yang dapat dibaca oleh mikrokontroler.
- **Kontrol (Pengolahan Sinyal):** Elektronika bertanggung jawab untuk mengolah sinyal dari sensor agar dapat digunakan oleh sistem kontrol robot, biasanya melalui penggunaan sirkuit analog atau digital.
- **Aktuator:** Elektronika diperlukan untuk mengendalikan aktuator seperti motor, servo, atau perangkat lain yang bergerak, berdasarkan perintah dari mikrokontroler.

Rangkaian Elektronika pada Robotika

Dalam robotika, terdapat berbagai jenis rangkaian yang digunakan untuk menjalankan fungsi-fungsi tertentu:

- **Rangkaian Pengaturan Daya:** Mengatur tegangan dan arus yang diberikan ke komponen robot agar sesuai dengan kebutuhan daya masing-masing komponen.
- **Rangkaian Sensor:** Mengambil sinyal dari lingkungan melalui sensor (misalnya, sensor jarak, cahaya, atau suhu) dan mengubahnya menjadi sinyal listrik yang diproses oleh mikrokontroler.

- **Rangkaian Driver Motor:** Mengontrol motor yang digunakan untuk menggerakkan robot. Biasanya menggunakan transistor, H-Bridge, atau IC driver motor untuk mengendalikan kecepatan dan arah putaran motor.
- **Rangkaian Pengolahan Sinyal:** Bertanggung jawab untuk menyaring dan memperkuat sinyal dari sensor, sehingga sinyal tersebut dapat digunakan oleh mikrokontroler atau komputer robot.

Fungsi Utama Elektronika dalam Robotika

Dalam sebuah robot, elektronika berfungsi untuk:

1. **Distribusi Daya:** Komponen robot memiliki kebutuhan daya yang berbeda-beda. Elektronika mengatur bagaimana daya disalurkan dari sumber daya (misalnya, baterai) ke komponen lain seperti sensor, motor, dan mikrokontroler.
2. **Pengolahan Sinyal:** Sinyal analog dari sensor perlu diubah menjadi sinyal digital yang bisa diproses oleh mikrokontroler. Elektronika mengolah sinyal ini, misalnya dengan menggunakan ADC (Analog-to-Digital Converter) dan filter sinyal.
3. **Kontrol Motor:** Elektronika memastikan bahwa motor dapat berfungsi sesuai perintah dari mikrokontroler. Ini mencakup pengaturan kecepatan motor (menggunakan PWM) dan arah putaran motor (menggunakan H-Bridge).
4. **Proteksi Sistem:** Elektronika bertanggung jawab untuk melindungi sistem dari gangguan seperti lonjakan tegangan atau arus berlebih yang bisa merusak komponen.

Tantangan dalam Elektronika Robotika

- **Manajemen Daya:** Menyediakan daya yang cukup tanpa mengorbankan efisiensi sangat penting, terutama pada robot mobile yang memiliki batasan baterai.
- **Pengolahan Sinyal yang Tepat:** Sensor sering memberikan sinyal yang lemah atau penuh dengan noise. Sistem elektronika harus mampu memperkuat sinyal tanpa menambah noise, serta memfilter sinyal yang tidak relevan.
- **Efisiensi Komponen Aktuator:** Mengendalikan motor atau aktuator dengan efisiensi tinggi tanpa menyebabkan panas berlebih atau gangguan lainnya menjadi tantangan tersendiri dalam sistem robotika.

B. Power Supply pada Robotika

Power supply adalah salah satu elemen kritis dalam sistem robotika. Ia bertanggung jawab untuk menyediakan daya yang diperlukan untuk mengoperasikan berbagai komponen elektronik, seperti sensor, aktuator, mikrokontroler, dan perangkat komunikasi. Tanpa pengaturan daya yang baik, robot mungkin tidak akan bekerja secara efisien, atau bahkan bisa mengalami kerusakan.

1. Fungsi Power Supply

Power supply dalam robotika berfungsi untuk:

- Menyediakan **tegangan dan arus** yang stabil dan sesuai kebutuhan komponen.
- **Melindungi** komponen dari tegangan berlebih atau lonjakan arus.
- **Mengkonversi** sumber daya dari baterai, sumber daya AC, atau panel surya menjadi bentuk tegangan yang bisa digunakan oleh komponen.

2. Sumber Daya

a. Baterai LiPo (Lithium Polymer)

- **Keunggulan:**
 - Daya per satuan berat yang tinggi (densitas energi tinggi).
 - Mampu menyediakan arus besar dalam waktu singkat, cocok untuk motor berdaya besar.
 - Bentuk fleksibel dan bisa dibuat sangat tipis.
- **Kelemahan:**
 - Cenderung mudah rusak jika tidak digunakan atau diisi daya dengan benar.
 - Memerlukan perawatan khusus, seperti keseimbangan pengisian tiap sel (balance charging) agar tidak terjadi overcharge atau overdischarge.
 - Dapat berbahaya jika rusak atau salah perlakuan, termasuk risiko kebakaran.
- **Spesifikasi:**
 - Tegangan per sel: 3.7V (nominal), 4.2V (penuh), 3.0V (habis).
 - Kapasitas: Diukur dalam milliampere-hour (mAh) atau ampere-hour (Ah), yang menentukan berapa lama baterai dapat menyediakan arus tertentu.

b. Baterai Li-Ion (Lithium Ion)

- **Keunggulan:**
 - Lebih stabil dan lebih aman dibandingkan LiPo.
 - Digunakan dalam banyak perangkat elektronik seperti laptop dan smartphone.
- **Kelemahan:**
 - Tidak dapat menyediakan arus sebesar LiPo dalam waktu singkat.
 - Memiliki siklus hidup (life cycle) yang lebih pendek daripada LiPo dalam beberapa aplikasi.
- **Spesifikasi:**
 - Tegangan per sel: 3.6V (nominal), 4.2V (penuh), 2.5V (habis).
 - Biasanya dikemas dalam sel silinder (seperti sel 18650).

c. Baterai NiMH (Nickel-Metal Hydride)

- **Keunggulan:**
 - Lebih ramah lingkungan dibandingkan baterai NiCd (Nickel-Cadmium).
 - Stabil dan aman untuk digunakan dalam banyak aplikasi.
- **Kelemahan:**
 - Kapasitas energi lebih rendah dibandingkan baterai LiPo atau Li-Ion.
 - Cenderung mengalami self-discharge yang cukup cepat, di mana kapasitas berkurang saat tidak digunakan.
- **Spesifikasi:**
 - Tegangan per sel: 1.2V (nominal).
 - Sering digunakan dalam robot sederhana dan proyek DIY.

d. Baterai Alkaline

- **Keunggulan:**
 - Mudah ditemukan dan murah.

- Digunakan dalam aplikasi dengan kebutuhan daya rendah dan durasi pendek.
- **Kelemahan:**
 - Tidak dapat diisi ulang.
 - Kapasitas yang lebih rendah dibandingkan baterai rechargeable lainnya.

3. Regulasi Tegangan

Sistem robot biasanya memerlukan tegangan yang berbeda-beda tergantung pada komponen yang digunakan, seperti 3.3V, 5V, atau 12V. Oleh karena itu, diperlukan regulator untuk mengatur dan menstabilkan tegangan.

a. Regulator Tegangan Linear

Regulator linear bekerja dengan cara membuang energi yang tidak dibutuhkan dalam bentuk panas. Mereka memiliki desain yang sederhana, namun tidak efisien untuk aplikasi dengan arus besar.

- **Keunggulan:**
 - Murah dan sederhana.
 - Tidak menghasilkan noise switching yang dapat mengganggu komponen sensitif.
- **Kelemahan:**
 - Sangat tidak efisien, terutama ketika perbedaan antara tegangan input dan output besar.
 - Menghasilkan panas yang signifikan.
- **Contoh:**
 - **LM7805:** Regulator tegangan linear yang menurunkan tegangan menjadi 5V. Kelemahannya adalah efisiensinya rendah karena energi yang tidak digunakan dibuang dalam bentuk panas.

b. Regulator Tegangan Switching (SMPS)

Regulator switching menggunakan teknik pensaklaran cepat untuk menurunkan atau menaikkan tegangan dengan efisiensi yang tinggi. Mereka jauh lebih efisien dibandingkan regulator linear, namun lebih kompleks.

- **Keunggulan:**
 - Efisiensi tinggi, terutama pada arus besar.
 - Dapat digunakan untuk menaikkan atau menurunkan tegangan (step-up, step-down).
- **Kelemahan:**
 - Dapat menghasilkan noise frekuensi tinggi, yang perlu diatasi dalam aplikasi sensitif.
- **Contoh:**
 - **Buck Converter:** Menurunkan tegangan dari tegangan input ke tegangan output yang lebih rendah.
 - **Boost Converter:** Menaikkan tegangan dari input ke output yang lebih tinggi.
 - Misalnya, **LM2596** untuk buck dan **XL6009** untuk boost.

4. Kapasitas dan Kebutuhan Daya

Untuk memilih baterai atau sumber daya lainnya, penting untuk menghitung kebutuhan daya total dari robot. Ini melibatkan:

- **Kapasitas Baterai:** Diukur dalam milliampere-hour (mAh) atau watt-hour (Wh). Kapasitas ini menunjukkan seberapa lama baterai dapat menyediakan daya sebelum habis.
- **Kebutuhan Daya Komponen:** Setiap komponen robot (sensor, mikrokontroler, motor) memiliki konsumsi daya yang berbeda. Dengan menjumlahkan semua kebutuhan daya, kita bisa menentukan baterai dengan kapasitas yang tepat.

5. Kapasitor untuk Stabilisasi Tegangan

Kapasitor sering digunakan dalam sistem daya robotika untuk mengurangi fluktuasi tegangan dan noise, serta untuk memastikan daya yang stabil.

- **Penggunaan:**
 - **Kapasitor Decoupling:** Dipasang dekat dengan komponen kritis (seperti mikrokontroler) untuk meredam noise.
 - **Kapasitor Penyangga (Bulk Capacitor):** Digunakan pada catu daya untuk menyimpan energi dan memberikan pasokan daya sementara jika ada lonjakan arus tiba-tiba.
- **Jenis Kapasitor:**
 - **Kapasitor Elektrolit:** Memiliki kapasitas besar, biasanya digunakan sebagai penyimpan energi jangka pendek.
 - **Kapasitor Keramik:** Digunakan untuk aplikasi frekuensi tinggi karena memiliki karakteristik stabil.

6. Sistem Manajemen Daya Baterai (BMS)

BMS adalah sistem yang dirancang untuk mengelola, memantau, dan melindungi baterai dari kondisi yang tidak aman, seperti overcharge, overdischarge, dan kelebihan arus.

- **Fungsi Utama:**
 - **Pengisian Seimbang (Balance Charging):** Menjaga semua sel baterai memiliki tegangan yang sama selama pengisian agar umur baterai lebih lama.
 - **Perlindungan:** Memastikan baterai tidak diisi daya atau dikosongkan hingga melebihi batas aman.
 - **Monitoring:** Mengukur tegangan, arus, dan suhu baterai.
- **Contoh Aplikasi BMS:**
 - Pada baterai LiPo, BMS sangat penting untuk mencegah kerusakan dan risiko keselamatan.

7. Distribusi Daya di Robot

Distribusi daya yang efisien adalah kunci agar robot dapat berfungsi dengan baik tanpa mengalami penurunan performa atau kerusakan komponen. Berikut ini adalah langkah-langkah umum dalam merancang distribusi daya:

a. Menentukan Kebutuhan Daya

- Hitung total konsumsi daya dari semua komponen robot (sensor, motor, mikrokontroler, dsb.).
- Tentukan tegangan yang diperlukan oleh setiap komponen.
- Hitung arus yang dibutuhkan oleh masing-masing bagian dari sistem.

b. Perhitungan Sistem Daya

Gunakan hukum Ohm dan rumus daya untuk menghitung kapasitas yang dibutuhkan:

$$P = V \times IP = V \times IP = V \times I$$

Dimana:

- **P** = Daya (Watt),
- **V** = Tegangan (Volt),
- **I** = Arus (Ampere).

Misalnya, jika total konsumsi daya sebuah motor adalah 10W dan sistem beroperasi pada 5V, maka arus yang dibutuhkan adalah: $I = \frac{P}{V} = \frac{10}{5} = 2A$

c. Pilihan Regulator Tegangan

- Gunakan regulator tegangan untuk mengonversi tegangan catu daya utama ke tegangan yang dibutuhkan oleh berbagai komponen.
- Pilih regulator yang efisien untuk meminimalkan panas dan memaksimalkan daya tahan baterai.

8. Perlindungan Sirkuit

Robotik sering beroperasi dalam kondisi yang tidak ideal, sehingga perlindungan terhadap lonjakan arus dan kerusakan adalah hal yang sangat penting.

a. Sekering (Fuse)

Sekering adalah perangkat perlindungan yang memutuskan arus ketika terjadi kelebihan arus yang dapat merusak rangkaian.

- **Slow-Blow Fuse:** Sekering yang memutuskan arus hanya setelah kelebihan arus berlanjut dalam waktu tertentu, cocok untuk perangkat yang memerlukan arus tinggi sesaat (seperti motor saat startup).

b. Dioda Flyback

Dioda ini digunakan untuk melindungi sirkuit dari lonjakan tegangan yang dihasilkan oleh motor atau relay saat dimatikan. Ini mencegah arus balik yang dapat merusak komponen elektronik lain.

c. Proteksi Overcurrent dan Overvoltage

Regulator tegangan modern dan IC BMS sering kali dilengkapi dengan proteksi overcurrent dan overvoltage untuk memastikan bahwa rangkaian tidak melebihi batas daya yang diizinkan.

C. Komponen Elektronika

Komponen elektronika aktif adalah komponen yang dapat mengendalikan aliran arus listrik dan memerlukan daya eksternal untuk berfungsi. Mereka sering digunakan untuk memperkuat sinyal, mengontrol tegangan, atau mengalirkan arus sesuai kebutuhan.

1. Transistor

Transistor adalah salah satu komponen aktif paling penting dalam elektronika robotika. Fungsinya meliputi penguat sinyal, switching, dan pengendalian arus.

- **BJT (Bipolar Junction Transistor):** Terdiri dari dua jenis, NPN dan PNP. BJT bekerja dengan mengendalikan arus basis untuk mengatur arus kolektor-emitor. BJT digunakan dalam aplikasi switching sederhana atau sebagai penguat.

- Contoh: **2N2222** (NPN), **2N2907** (PNP).
- **MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor):** MOSFET digunakan pada aplikasi yang memerlukan efisiensi tinggi dan kecepatan switching yang cepat. MOSFET memiliki dua jenis utama: N-channel dan P-channel. MOSFET dikendalikan oleh tegangan pada gerbang (gate).
 - Contoh: **IRF540N** (N-channel), **IRF9540** (P-channel).
 - MOSFET sering digunakan dalam driver motor, konverter daya, dan aplikasi switching berdaya tinggi.

2. Dioda

Dioda adalah komponen yang hanya memungkinkan arus listrik mengalir dalam satu arah. Ini sangat berguna dalam berbagai aplikasi seperti rectifier (penyearah) dan proteksi arus balik.

- **Dioda Penyearah:** Dioda yang digunakan untuk mengubah arus bolak-balik (AC) menjadi arus searah (DC). Contoh: **1N4007**.
- **Zener Diode:** Digunakan untuk regulasi tegangan. Dioda ini memungkinkan arus mengalir balik ketika tegangan mencapai nilai tertentu, menjaga kestabilan tegangan pada level tertentu. Contoh: **1N4733** (5.1V).
- **Schottky Diode:** Dioda dengan tegangan maju rendah, sering digunakan dalam aplikasi switching cepat dan proteksi flyback pada motor.

3. IC (Integrated Circuit)

IC adalah chip yang menggabungkan beberapa komponen elektronika dalam satu paket. IC digunakan untuk berbagai fungsi, termasuk pengendalian motor, pengolahan sinyal, dan komunikasi antar komponen.

- **IC Driver Motor:** IC seperti **L298N** atau **DRV8833** digunakan untuk mengendalikan motor DC dan stepper. IC ini dapat mengatur arah putaran dan kecepatan motor.
- **IC Op-Amp (Operational Amplifier):** Digunakan dalam penguatan sinyal analog dari sensor. Contoh: **LM741**, **TL081**.
- **IC Timer 555:** Digunakan untuk menghasilkan sinyal PWM (Pulse Width Modulation) yang digunakan untuk mengendalikan kecepatan motor.

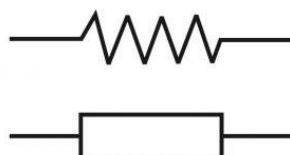
4. Thyristor dan Triac

- **Thyristor:** Semikonduktor yang digunakan dalam kontrol daya tinggi. Thyristor bekerja sebagai saklar yang hanya bisa dimatikan ketika arus di bawah ambang tertentu.
- **Triac:** Komponen yang digunakan dalam aplikasi kontrol daya AC, misalnya untuk mengatur intensitas cahaya atau kecepatan motor AC.

Komponen elektronika pasif ini ketika dialiri listrik, tidak akan menghasilkan tenaga, seperti perubahan tegangan, pembalikan fasa, penguatan, atau lainnya

1. Resistor

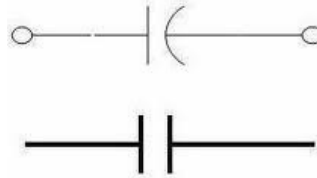
Resistor adalah komponen yang digunakan untuk membatasi jumlah arus yang mengalir dalam rangkaian. Fungsi utamanya adalah mengontrol tegangan dan arus di berbagai bagian rangkaian.



Nilai resistor ditentukan oleh kode warna pada badan resistor atau dengan pengukuran langsung menggunakan multimeter.

2. Kapasitor

Kapasitor menyimpan energi dalam bentuk medan listrik dan mampu melepaskan energi tersebut dalam waktu yang singkat. Dalam robotika, kapasitor sering digunakan untuk menyaring noise pada catu daya atau untuk menyimpan daya sementara pada rangkaian tertentu.

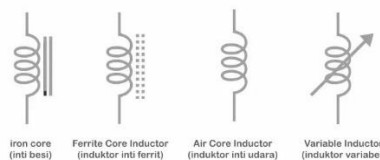


Kapasitor memiliki dua jenis utama:

- **Kapasitor Elektrolit:** Polaritasnya harus diperhatikan.
- **Kapasitor Keramik:** Tidak memiliki polaritas.

Induktor

Induktor menyimpan energi dalam bentuk medan magnet saat arus mengalir melaluinya. Dalam robotika, induktor sering digunakan dalam rangkaian filter dan regulator daya.



D. Desain dan Simulasi Rangkaian Elektronik

Rangkaian elektronik merupakan tulang punggung sistem robotik, menghubungkan sensor, aktuator, dan mikrokontroler yang menjalankan logika pengendali. Untuk merancang rangkaian yang tepat, insinyur robotika harus menguasai proses desain, simulasi, dan implementasi rangkaian. Simulasi digunakan untuk menguji konsep sebelum memproduksi perangkat fisik, yang membantu menghemat waktu dan biaya.

1. Software Simulasi Elektronik

Dalam proses desain rangkaian elektronik, simulasi adalah tahap penting yang memungkinkan desainer menguji perilaku rangkaian tanpa perlu langsung membuatnya secara fisik. Beberapa software simulasi populer yang dapat digunakan antara lain:

a. LTSpice

- **Fungsi Utama:** Simulasi sirkuit elektronik, terutama yang menggunakan komponen analog.
- **Keunggulan:**
 - Gratis dan banyak digunakan dalam industri.
 - Dapat digunakan untuk simulasi berbagai rangkaian elektronik, termasuk rangkaian power supply, amplifier, dan rangkaian analog lainnya.

- **Kelemahan:**

- Antarmuka yang sedikit ketinggalan zaman dan butuh waktu untuk mempelajari fitur-fiturnya.

b. Proteus

- **Fungsi Utama:** Simulasi rangkaian elektronik serta mikrokontroler (seperti Arduino atau PIC).

- **Keunggulan:**

- Memungkinkan simulasi yang lebih komprehensif, termasuk komponen analog, digital, dan mikrokontroler.
- Anda dapat memprogram mikrokontroler dalam simulasi dan melihat interaksi antara perangkat lunak dan perangkat keras.

- **Kelemahan:**

- Berbayar, sehingga mungkin tidak cocok untuk penggunaan pemula yang memerlukan opsi gratis.

c. Tinkercad Circuits

- **Fungsi Utama:** Simulasi rangkaian elektronik sederhana dan pemrograman mikrokontroler seperti Arduino.

- **Keunggulan:**

- Berbasis web dan gratis.
- Mudah digunakan untuk pemula.
- Memiliki visualisasi interaktif, sehingga sangat baik untuk pembelajaran dasar.

- **Kelemahan:**

- Hanya dapat digunakan untuk rangkaian yang relatif sederhana.

d. KiCad

- **Fungsi Utama:** Desain skema dan layout PCB (Printed Circuit Board).

- **Keunggulan:**

- Open-source dan gratis.
- Dapat digunakan untuk membuat skema dan desain PCB untuk rangkaian kompleks.
- Banyak digunakan dalam komunitas elektronik DIY dan profesional.

- **Kelemahan:**

- Fitur simulasi terbatas, sehingga lebih fokus pada desain PCB.

2. Membuat Skema Rangkaian

Sebelum membuat rangkaian fisik atau simulasi, desain dimulai dengan **skema rangkaian**. Skema adalah representasi visual dari rangkaian elektronik, yang menunjukkan bagaimana komponen dihubungkan secara listrik.

Langkah-langkah Membuat Skema:

1. Menentukan Kebutuhan Komponen:

- Tentukan komponen yang diperlukan, seperti resistor, kapasitor, transistor, IC (Integrated Circuit), sensor, dan motor.

2. Merancang Skema:

- Gunakan software desain skema seperti KiCad atau Proteus untuk menggambar diagram rangkaian.
- Pastikan semua komponen dihubungkan dengan benar sesuai dengan spesifikasi komponen.

3. Menambahkan Nilai Komponen:

- Tambahkan nilai resistor, kapasitor, atau tegangan dari sumber daya berdasarkan perhitungan awal.

4. Verifikasi Koneksi:

- Pastikan tidak ada kesalahan sambungan seperti jalur yang terputus atau komponen yang terhubung secara tidak benar.

3. Desain PCB (Printed Circuit Board)

Setelah skema selesai, langkah berikutnya adalah merancang PCB untuk mengimplementasikan rangkaian secara fisik. PCB memungkinkan komponen elektronik terhubung dengan jalur tembaga yang dicetak di atas substrat isolator.

Langkah-langkah Membuat PCB:

1. Pilih Software Desain PCB:

- Gunakan software seperti KiCad atau Eagle untuk mendesain PCB.

2. Membuat Layout PCB:

- Posisikan komponen pada layout PCB dengan mempertimbangkan efisiensi ruang dan kebutuhan termal.
- Buat jalur tembaga yang menghubungkan pin-pin komponen sesuai dengan skema yang telah dirancang.

3. Routing Jalur:

- **Autoroute:** Fitur otomatis yang membantu mengatur jalur koneksi pada PCB.
- **Manual Routing:** Pengaturan jalur secara manual untuk kontrol lebih presisi.

4. Desain Layer:

- Jika rangkaian kompleks, PCB dapat memiliki banyak lapisan (multi-layer) untuk memaksimalkan ruang dan efisiensi.

5. Generate Gerber File:

- Setelah layout selesai, file Gerber dihasilkan untuk digunakan oleh produsen PCB dalam proses pembuatan.

4. Simulasi Rangkaian

Sebelum membuat PCB atau merakit komponen fisik, simulasi dilakukan untuk menguji rangkaian.

Tahapan Simulasi:

1. Memasukkan Skema ke Dalam Simulator:

- Masukkan skema yang sudah dirancang ke dalam software simulasi seperti LTSpice atau Proteus.

2. Menjalankan Simulasi:

- Jalankan simulasi dan periksa apakah rangkaian bekerja sesuai dengan ekspektasi. Amati tegangan, arus, dan sinyal pada berbagai titik.

3. Menganalisis Hasil:

- Gunakan hasil simulasi untuk mendeteksi potensi masalah, seperti tegangan yang terlalu tinggi, arus yang berlebihan, atau sinyal yang tidak stabil.

4. Iterasi Desain:

- Jika hasil simulasi tidak sesuai, perbaiki skema dan jalankan simulasi ulang sampai hasilnya memuaskan.

E. Praktik dan Troubleshooting Elektronik Robotik

Setelah desain dan simulasi selesai, saatnya untuk merealisasikan rangkaian secara fisik dan melakukan uji coba. Dalam proses ini, troubleshooting adalah keterampilan penting untuk mengidentifikasi dan memperbaiki masalah yang muncul.

1. Teknik Soldering dan Desoldering

Soldering adalah proses menghubungkan komponen elektronik dengan menggunakan solder (timah) dan soldering iron. Kualitas soldering yang baik sangat penting agar koneksi komponen aman dan fungsional.

Langkah-langkah Soldering yang Baik:

1. Persiapan Alat:

- Pastikan soldering iron sudah dipanaskan dengan suhu yang tepat (biasanya sekitar 350°C untuk timah timbal).

- Gunakan solder berkualitas yang mengandung fluks untuk membantu membersihkan dan melindungi titik sambungan.

2. Soldering:

- Panaskan pin komponen dan pad PCB secara bersamaan dengan soldering iron.
- Oleskan solder pada sambungan yang dipanaskan hingga solder mencair dan menyelimuti sambungan dengan rapi.
- Jangan terlalu lama memanaskan komponen agar tidak merusaknya.

3. Periksa Koneksi:

- Pastikan hasil soldering rapi dan tidak ada solder yang menghubungkan pin-pin komponen yang tidak diinginkan (solder bridge).

Langkah-langkah Desoldering:

- **Desoldering Pump:** Gunakan alat ini untuk menyedot solder cair dari sambungan.
- **Desoldering Wick (Kawat Serabut):** Alat ini digunakan untuk menyerap solder dari sambungan saat dipanaskan.

2. Penggunaan Alat Ukur

Untuk mendiagnosis masalah pada rangkaian, alat ukur seperti multimeter dan osiloskop sangat penting.

a. Multimeter

Multimeter digunakan untuk mengukur beberapa parameter, seperti tegangan (V), arus (A), dan resistansi (Ω). Ini adalah alat dasar untuk troubleshooting rangkaian elektronik.

- **Cara Menggunakan:**
 - **Mengukur Tegangan:** Set multimeter ke mode voltmeter dan letakkan probe pada titik pengukuran.
 - **Mengukur Arus:** Letakkan multimeter dalam mode ammeter dan sambungkan secara seri dengan rangkaian.
 - **Mengukur Resistansi:** Matikan catu daya, dan ukur resistansi komponen atau jalur untuk mencari sirkuit pendek.

b. Osiloskop

Osiloskop digunakan untuk melihat bentuk gelombang sinyal listrik secara real-time. Ini berguna untuk memantau sinyal digital atau analog yang berubah secara cepat.

- **Cara Menggunakan:**
 - Hubungkan probe osiloskop ke titik yang ingin diukur, dan set pengaturan waktu serta tegangan.
 - Amati gelombang sinyal untuk memeriksa kestabilan, frekuensi, dan amplitudo sinyal.

3. Mengatasi Masalah Umum pada Rangkaian Elektronik

Berikut adalah beberapa masalah umum yang mungkin muncul saat membangun rangkaian elektronik, beserta cara mengatasinya.

a. Sirkuit Pendek

Sirkuit pendek terjadi ketika dua jalur listrik yang seharusnya terpisah terhubung secara tidak sengaja, biasanya akibat solder bridge atau kesalahan pada PCB.

- **Solusi:** Gunakan multimeter untuk mengukur resistansi antara dua titik yang dicurigai terhubung secara tidak sengaja. Gunakan soldering iron untuk memperbaiki solder yang berlebihan.

b. Tegangan Tidak Sesuai

Jika tegangan pada komponen berbeda dari yang diharapkan, bisa jadi karena masalah regulator tegangan, soldering yang buruk, atau baterai yang lemah.

- **Solusi:** Periksa tegangan pada input dan output regulator menggunakan multimeter. Jika regulator rusak, gantilah dengan yang baru.

c. Komponen Tidak Berfungsi

Jika komponen seperti motor, sensor, atau mikrokontroler tidak berfungsi, ini bisa disebabkan oleh kesalahan koneksi atau komponen yang rusak.

- **Solusi:** Periksa jalur koneksi menggunakan multimeter untuk memastikan bahwa komponen mendapatkan tegangan dan arus yang cukup. Gantilah komponen yang rusak jika ditemukan.

BAB IV

SENSOR DAN AKTUATOR

A. Sensor

Robot tidak dapat bergerak dalam jarak tertentu ke arah tertentu hanya dengan mengatur daya relatif motor dari dua roda dan periode waktu motor berjalan. Misalkan kita ingin robot bergerak lurus ke depan. Jika kita mengatur kekuatan kedua motor ke tingkat yang sama, bahkan perbedaan kecil dalam karakteristik motor dan roda akan menyebabkan robot berbelok sedikit ke satu sisi. Ketidakrataan dalam permukaan tempat robot bergerak juga akan menyebabkan roda berputar pada kecepatan yang berbeda kecepatan. Peningkatan atau penurunan gesekan antara roda dan permukaan dapat mempengaruhi jarak yang ditempuh dalam periode waktu tertentu. Oleh karena itu, jika kita ingin robot bergerak menuju dinding sejauh 1 m dan berhenti 20 cm di depannya, robot harus merasakannya keberadaan dinding dan berhenti ketika mendeteksi bahwa dinding berjarak 20 cm. Sensor adalah komponen yang mengukur beberapa aspek lingkungan. Komputer di dalam robot menggunakan pengukuran ini untuk mengontrol tindakan robot.

Sensor diklasifikasikan sebagai proprioseptif (Bertanggung jawab untuk mendeteksi kondisi kendaraan, seperti status sistem (mesin, transmisi, pengereman, dan kemudi), dan posisi kendaraan di dunia dengan menggunakan lokalisasi GPS, rotasi roda, kecepatan kendaraan, dan percepatan) atau eksteroseptif (Bertanggung jawab untuk merasakan lingkungan sekitar dengan menggunakan sensor seperti kamera, lidar, radar, dan sensor ultrasonik), dan sensor eksteroseptif selanjutnya diklasifikasikan sebagai aktif atau pasif.

Sensor proprioseptif mengukur sesuatu yang bersifat internal pada robot itu sendiri. Contoh yang paling dikenal adalah speedometer mobil yang mengukur kecepatan mobil dengan menghitung putaran roda. Sebuah sensor eksteroseptif mengukur sesuatu yang eksternal ke robot seperti jarak ke suatu objek. Sensor aktif mempengaruhi lingkungan biasanya dengan memancarkan energi: Pencari jarak sonar di kapal selam memancarkan gelombang suara dan menggunakan suara yang dipantulkan untuk menentukan jangkauan. Sensor pasif tidak mempengaruhi lingkungan: kamera hanya merekam cahaya yang dipantulkan dari suatu objek. Robot selalu menggunakan beberapa sensor eksteroseptif sensor untuk mengoreksi kesalahan yang mungkin timbul dari sensor proprioseptif atau untuk mengambil memperhitungkan perubahan lingkungan.

a. Sensor Posisi dan Orientasi

Sensor-sensor ini digunakan untuk menentukan lokasi dan orientasi robot, baik di ruang dua dimensi (2D) maupun tiga dimensi (3D).

- Encoders

Encoders adalah sensor yang digunakan untuk mengukur kecepatan rotasi dan posisi dari roda atau motor.

- **Optical Encoders:** Menggunakan sinar cahaya untuk mendeteksi rotasi. Sebuah piringan berputar dengan pola yang dilewatkan di depan detektor optik. Pola ini digunakan untuk menghitung jumlah putaran roda.
 - **Kelebihan:** Akurat, mudah digunakan.
 - **Kekurangan:** Rentan terhadap debu dan kotoran.
- **Magnetic Encoders:** Menggunakan perubahan medan magnet untuk mendeteksi rotasi.
 - **Kelebihan:** Tahan terhadap debu, lebih andal di lingkungan keras.
 - **Kekurangan:** Biasanya lebih mahal.
- **Linear Encoders:** Digunakan untuk mengukur gerakan linear atau posisi lurus.

- Gyroscope (Gyro)

Gyroscope mengukur kecepatan sudut (rotasi) robot. Ini penting untuk menstabilkan dan menjaga orientasi robot, khususnya dalam aplikasi navigasi dan keseimbangan.

- **Kelebihan:** Deteksi akurat terhadap rotasi.
- **Kekurangan:** Rentan terhadap drift (kesalahan kecil yang terakumulasi dari waktu ke waktu).

- Accelerometer

Accelerometer mengukur percepatan linier robot, baik dalam sumbu X, Y, atau Z. Digunakan untuk menghitung perubahan kecepatan dan juga sebagai bagian dari sistem penstabilan robot.

- **Kelebihan:** Dapat mendeteksi gerakan mendadak atau getaran.
- **Kekurangan:** Pengukuran bisa terganggu oleh kebisingan mekanis.

- Inertial Measurement Unit (IMU)

IMU adalah gabungan dari gyroscope dan accelerometer. IMU dapat memberikan data komprehensif tentang gerakan rotasi dan linear robot, memungkinkan pengukuran posisi dan orientasi secara tiga dimensi.

- **Kelebihan:** Dapat mengukur gerakan dengan presisi tinggi dalam 3D.
- **Kekurangan:** Harganya lebih mahal dan lebih kompleks dalam penggunaannya.

- GPS (Global Positioning System)

GPS digunakan untuk mengetahui posisi absolut robot di luar ruangan. Sensor ini sangat umum digunakan pada robot yang memerlukan penentuan posisi secara global, seperti robot outdoor dan kendaraan otonom.

- **Kelebihan:** Akurat dalam mendeteksi posisi secara global di ruang terbuka.
- **Kekurangan:** Tidak efektif di dalam ruangan atau lingkungan yang terhalang.

b. Sensor Lingkungan

Robot beroda perlu memahami lingkungannya agar dapat menghindari rintangan, mengenali jalan, dan mengadaptasi perilakunya sesuai kondisi di sekitar.

- Ultrasonic Sensor

Sensor ultrasonik menggunakan gelombang suara frekuensi tinggi untuk mendeteksi jarak dari objek. Sensor ini bekerja dengan mengirimkan gelombang suara dan mengukur waktu yang dibutuhkan untuk pantulan kembali (echo).

- **Kelebihan:** Dapat mendeteksi objek dalam jarak menengah (hingga 5 meter), murah, dan andal.
- **Kekurangan:** Kurang akurat dalam lingkungan dengan permukaan yang sangat lembut atau terlalu keras (seperti kaca), karena dapat menyerap atau memantulkan suara secara tidak konsisten.

- Infrared (IR) Sensor

Sensor infrared mendeteksi objek atau permukaan berdasarkan pantulan cahaya inframerah. Sensor ini banyak digunakan untuk deteksi rintangan jarak dekat.

- **Kelebihan:** Murah dan sederhana dalam penggunaannya.
- **Kekurangan:** Jarak deteksi pendek dan sangat dipengaruhi oleh cahaya sekitar atau warna objek.

- LIDAR (Light Detection and Ranging)

LIDAR menggunakan cahaya laser untuk mengukur jarak. Ini bekerja dengan mengukur waktu yang diperlukan bagi sinar laser untuk memantul dari objek kembali ke sensor. LIDAR digunakan dalam aplikasi navigasi robotik yang memerlukan pemetaan lingkungan secara akurat.

- **Kelebihan:** Sangat akurat dalam pemetaan 3D dan deteksi rintangan.
- **Kekurangan:** Mahal dan rentan terhadap gangguan lingkungan (misalnya, kabut atau hujan).

- Camera / Vision Sensors

Kamera digunakan untuk menangkap gambar atau video yang dapat diolah secara visual oleh algoritma penglihatan mesin (computer vision). Dengan kamera, robot bisa mengenali objek, mendeteksi gerakan, atau melakukan pelacakan visual.

- **Kelebihan:** Dapat memberikan informasi visual yang kaya.
- **Kekurangan:** Memerlukan pemrosesan gambar yang kompleks dan sering kali sensitif terhadap pencahayaan lingkungan.

- Time of Flight (ToF) Sensor

Sensor ToF menggunakan cahaya inframerah untuk mengukur jarak dari objek berdasarkan waktu tempuh cahaya. Sensor ini sangat akurat dalam mengukur jarak dalam jarak pendek hingga menengah.

- **Kelebihan:** Akurasi tinggi dan respon cepat.
- **Kekurangan:** Terbatas pada jarak pendek hingga menengah.

c. Sensor Kecepatan dan Percepatan

Selain mengukur posisi dan orientasi, robot beroda juga perlu mengukur kecepatan dan percepatan untuk mengontrol gerakannya dengan presisi.

- Speed Sensors

Speed sensor pada robot beroda digunakan untuk mengukur kecepatan rotasi roda. Tipe umum dari sensor kecepatan adalah **tachometer** yang mengukur kecepatan putaran poros.

- **Kelebihan:** Memberikan data kecepatan secara real-time.
- **Kekurangan:** Biasanya membutuhkan kalibrasi ulang secara berkala.

- Wheel Encoders

Wheel encoder, seperti yang telah disebutkan di atas, juga digunakan untuk mengukur kecepatan roda dan menghitung jarak yang ditempuh robot.

- **Kelebihan:** Mudah digunakan untuk menghitung jarak tempuh dan kecepatan rotasi.
- **Kekurangan:** Terkadang memerlukan pelindung dari kotoran atau debu untuk mencegah kerusakan.

d. Sensor Sentuhan (Tactile Sensor)

Sensor sentuhan memberikan umpan balik ketika robot beroda menyentuh atau menabrak objek di sekitarnya.

- Bumper Switch

Bumper switch adalah sensor yang sangat sederhana yang mendeteksi tabrakan fisik dengan objek. Jika bumper tertekan, maka switch akan aktif dan memberikan sinyal kepada sistem kontrol untuk berhenti atau berbelok.

- **Kelebihan:** Sederhana, murah, dan efektif.
- **Kekurangan:** Hanya bekerja setelah terjadi kontak fisik dengan objek.

- Force-Sensitive Resistor (FSR)

FSR adalah sensor yang mendeteksi tekanan pada permukaan. Semakin besar tekanan yang diberikan, semakin besar perubahan resistansi.

- **Kelebihan:** Dapat mendeteksi intensitas sentuhan, cocok untuk aplikasi di mana kontrol gaya diperlukan.
- **Kekurangan:** Sensitivitasnya bisa berubah seiring waktu dan pemakaian.

- Capacitive Touch Sensor

Sensor ini bekerja dengan mendeteksi perubahan medan listrik ketika ada sentuhan atau pendekatan dari objek konduktif, seperti tangan manusia.

- **Kelebihan:** Dapat digunakan untuk mendeteksi sentuhan tanpa kontak fisik langsung.
- **Kekurangan:** Rentan terhadap gangguan dari lingkungan elektromagnetik.

e. Sensor Lain yang Umum Digunakan

- Temperature Sensors

Mengukur suhu lingkungan atau komponen robot, seperti motor. Sensor ini penting untuk memastikan bahwa robot tidak bekerja dalam kondisi yang dapat merusak komponen akibat panas berlebih.

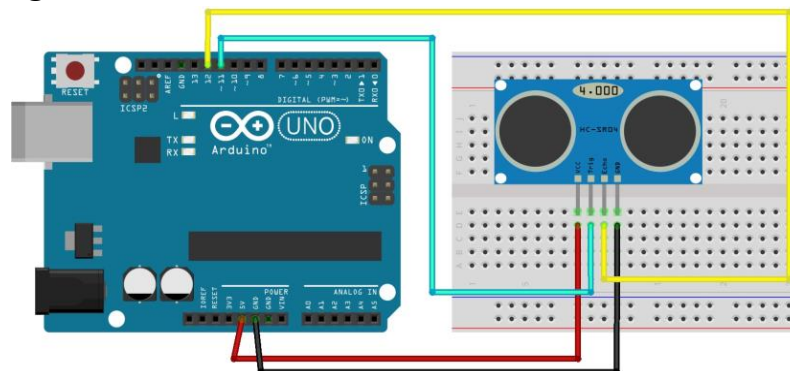
- **Kelebihan:** Mencegah overheating dan kerusakan.
- **Kekurangan:** Sensor suhu biasanya memberikan pembacaan lambat.

- Proximity Sensors

Digunakan untuk mendeteksi keberadaan objek di dekat robot tanpa kontak fisik. Ada beberapa jenis proximity sensor, termasuk **inductive** (untuk objek logam) dan **capacitive** (untuk objek konduktif).

- **Kelebihan:** Tangguh dan dapat bekerja dalam berbagai kondisi.
- **Kekurangan:** Jarak deteksinya terbatas.

e. Programming Sensor : Ultrasonic

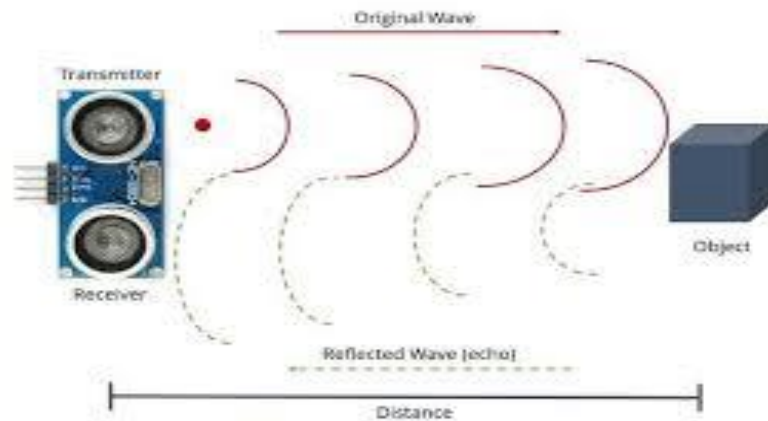


Sensor ultrasonik adalah sensor jarak yang menggunakan gelombang suara ultrasonik (suara frekuensi di atas 20.000 hertz) untuk memprediksikan suatu jarak. Kecepatan suara di udara adalah sekitar 340 m/s atau 34.000 cm/s. Jika sebuah benda berada pada jarak tertentu dari robot, maka ketika sebuah gelombang suara ultrasonik ditembakkan ke arah object, maka suara akan akan menjalar ke objek dan dipantulkan kembali persamaan :

$$s = 0.5vt$$

Keuntungan dari sensor ultrasound adalah bahwa mereka tidak sensitif terhadap perubahan warna atau pantulan cahaya objek, atau intensitas cahaya di lingkungan. Namun, mereka sensitif terhadap tekstur: kain akan menyerap sebagian suara, sementara kayu atau logam akan memantulkan hampir semua suara. Itulah mengapa tirai, karpet dan ubin langit-langit digunakan untuk membuat ruangan lebih nyaman. Sensor ultrasound relatif murah dan dapat bekerja di lingkungan luar ruangan. Sensor ini digunakan di mobil untuk mendeteksi jarak pendek, seperti untuk membantu parkir.

Kerugian utamanya adalah pengukuran jarak relatif lambat, karena kecepatannya suara jauh lebih kecil daripada kecepatan cahaya. Kerugian lainnya adalah bahwa mereka tidak bisa difokuskan untuk mengukur jarak ke objek tertentu.. Pin trigger memberikan sinyal untuk mengeluarkan gelombang suara, setelah gelombang tersebut mengenai suatu benda gelombang tersebut akan dipantulkan dan ditangkap oleh pin echo.



Untuk menggunakan sensor ultrasonik ikuti langkah-langkah berikut:

1. Sambungkan wiring sesuai dengan gambar di atas.
2. Ketikkan kode berikut:

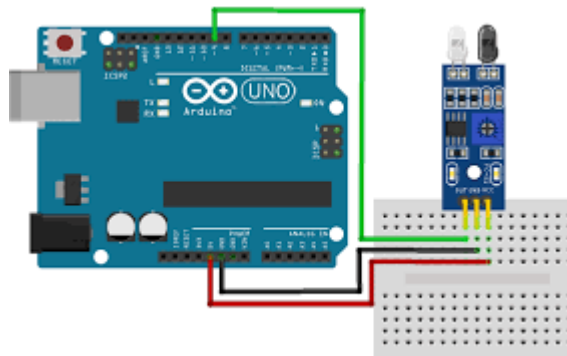
```

1  const int trigPin = 12;
2  const int echoPin = 11;
3
4  void setup() {
5      pinMode(trigPin, OUTPUT);
6      pinMode(echoPin, INPUT);
7      Serial.begin(9600);
8  }
9
10 void loop() {
11     digitalWrite(trigPin, LOW);
12     delayMicroseconds(2);
13     digitalWrite(trigPin, HIGH);
14     delayMicroseconds(10);
15     digitalWrite(trigPin, LOW);
16
17     long duration = pulseIn(echoPin, HIGH);
18     int distance = duration * 0.034 / 2;
19     Serial.print("Jarak: ");
20     Serial.print(distance);
21     Serial.println(" cm");
22     delay(100);
23 }

```

- a. Line 1 dan 2 menetapkan pin trigger dan pin echo sesuai dengan wiring di atas
 - b. Line 5 dan 6 menetapkan mode pin untuk pin trigger dan echo
 - c. Line 7 digunakan untuk memulai serial monitor
 - d. Ketika trigPin dalam keadaan high, sensor akan mengeluarkan gelombang suara, namun pertama kita perlu mematikan gelombang tersebut untuk mengurangi noise, itulah yang kita lakukan pada line 11 dan 12.
 - e. Line 13 dan 14 membuat sensor mengeluarkan gelombang suara.
 - f. Line 17 membaca input dari echoPin
 - g. Line 18 menghitung jarak dari pembacaan sensor, nilai 0.034 didapatkan dari kecepatan suara (cm/mikro sekon)
 - h. Line 19 sampai 21 menampilkan hasil pembacaan
3. Upload kode.

f. Programming Sensor : Proximity IR (Infrared)



Cahaya inframerah adalah cahaya yang panjang gelombangnya lebih panjang dari cahaya merah, yang merupakan cahaya dengan dengan panjang gelombang terpanjang yang dapat dilihat oleh mata kita. Mata dapat melihat cahaya dengan panjang gelombang dari sekitar 390 hingga 700 nm (nanometer adalah sepersejuta milimeter). Inframerah Cahaya inframerah memiliki panjang gelombang antara 700 dan 1000 nm. Cahaya ini tidak terlihat oleh mata manusia dan oleh karena itu digunakan dalam kendali jarak jauh perangkat TV dan peralatan elektronik lainnya.

Sensor proximity adalah perangkat sederhana yang menggunakan cahaya untuk mendeteksi keberadaan objek dengan mengukur intensitas cahaya yang dipantulkan. Intensitas cahaya berkurang dengan kuadrat jarak dari sumber dan hubungan ini dapat digunakan untuk mengukur perkiraan jarak ke objek. Pengukuran jarak adalah tidak terlalu akurat karena intensitas yang dipantulkan juga tergantung pada reflektifitas objek. Objek hitam memantulkan lebih sedikit cahaya daripada objek putih yang ditempatkan pada jarak yang sama, sehingga sensor jarak tidak dapat membedakan antara objek hitam yang dekat dan objek putih yang ditempatkan agak jauh. Inilah alasan mengapa sensor ini disebut sensor proximity, bukan sensor jarak. Sebagian besar robot pendidikan menggunakan sensor jarak karena harganya jauh lebih murah daripada sensor jarak

Sensor Inframerah adalah komponen elektronik yang dapat mendeteksi cahaya inframerah. Ketika ada objek yang menghalangi sinar inframerah yang dipancarkan,

sinar tersebut akan dipantulkan kembali ke penerima. Penerima kemudian akan mengubah sinyal cahaya menjadi sinyal listrik yang dapat dibaca oleh arduino. Berikut cara menggunakan sensor tersebut.

1. Sambungkan sesuai dengan wiring di atas.
2. Ketikkan kode berikut:

```
1  const int sensorPin = 9;
2
3  void setup() {
4      Serial.begin(9600);
5      pinMode(sensorPin, INPUT);
6  }
7
8  void loop() {
9      int sensorValue = digitalRead(sensorPin);
10
11     if (sensorValue == LOW) {
12         Serial.println("Objek terdeteksi");
13     } else {
14         Serial.println("Tidak ada objek");
15     }
16
17     delay(100);
18 }
```

- a. Line 1 mendeklarasi sensor pin
- b. Line 5 menetapkan sensorPin sebagai pin input.
- c. Line 4 memulai serial monitor.
- d. Line 9 membaca input dari sensorPin.
- e. Line 11 sampai 15 digunakan untuk menampilkan hasil pembacaan sensor.

3. Upload kode ke Arduino

B. Aktuator

Aktuator adalah perangkat yang mengubah sinyal listrik menjadi gerakan fisik. Aktuator memungkinkan robot untuk melakukan tindakan fisik di lingkungan, seperti bergerak, mengangkat objek, atau memutar bagian tertentu.

Fungsi Aktuator dalam Robotika:

- **Gerakan dan Navigasi:** Mengontrol roda, kaki, atau mekanisme lain untuk pergerakan robot.
- **Manipulasi Objek:** Menggerakkan lengan robot atau gripper untuk mengambil, memegang, atau memindahkan objek.
- **Kontrol Lingkungan:** Mengontrol katup, pintu, atau perangkat lain yang mengubah kondisi lingkungan.

- **Interaksi dengan Pengguna:** Menjalankan mekanisme yang merespons input dari pengguna, seperti tombol atau saklar.

a. Motor DC

Motor DC adalah perangkat elektris yang mengkonversi energi listrik menjadi energi mekanik. Dalam industri, motor DC telah digunakan untuk menggerakkan berbagai macam perangkat khususnya perangkat industri yang memerlukan torsi tinggi dan stabil.

Pengendalian Motor DC dengan Motor Driver

Seperti yang kita ketahui bahwa arduino merupakan modul yang tidak bisa mengirimkan sinyal analog sebagai output. Artinya akan menjadi terbatas jika pengendalian motor dilakukan dalam ranah sinyal analog. Oleh karena itu diperlukan perangkat berupa motor driver sebagai pengendali gerakan motor. Dengan menggunakan motor driver dan sinyal PWM, motor DC tersebut dapat dikendalikan dengan aturan algoritme tertentu. Sinyal PWM yang diberikan adalah berkisar antara 1 – 255 yang mewakili kecepatan motor DC dengan HIGH menyatakan PWM maksimal serta kecepatan maksimal dari motor DC dan LOW menyatakan PWM minimum serta kecepatan minimum dari motor DC. Sinyal PWM akan dikirimkan pada pin RPWM dan LPWM sebagai output dari mikrokontroller dan input dari motor DC. Berikut merupakan kombinasi yang dapat diberikan kepada motor driver sebagai pengendali motor.

	Kondisi PWM (1 - 255)			
RPWM	HIGH	LOW	LOW	HIGH
LPWM	LOW	HIGH	LOW	HIGH
Gerakan	Berputar berlawanan arah jarum jam	Berputar searah jarum jam	Berhenti	Motor Driver terbakar

Note : Silahkan pelajari datasheet dari motor dc yang digunakan serta melakukan koordinasi dengan tim elektronik terkait wiring agar terhindar dari kesalahan!

Pada robot akan digunakan motor DC sebagai aktuator roda omni dan motor driver sebagai perangkat kendali dari motor agar dapat dilakukan pemrograman sesuai arah gerak roda yang diinginkan. Motor DC yang digunakan adalah jenis PG36, ada juga beberapa jenis lain yang bisa digunakan. Kemudian motor DC tersebut akan dikendalikan dengan motor driver BTS7960.



b. Power Window

Pada dasarnya motor power window merupakan motor dc yang sama dengan motor pg36 hanya saja kemampuan kecepatannya serta torsi yang diberikan berbeda. Sebenarnya peruntukan dari Power Window adalah untuk kendaraan (mobil), sehingga tidak terlalu disarankan untuk menggunakan motor jenis ini karena arus maksimum yang dibutuhkan lumayan besar.



Note : Silahkan pelajari datasheet dari motor power window yang digunakan serta melakukan koordinasi dengan tim elektronik terkait wiring agar terhindar dari kesalahan!

c. Pneumatic

Secara sederhana, pneumatic merupakan tekanan udara yang dinaikkan oleh kompresor sehingga mampu menggerakkan berbagai alat industri. Tekanan udara ini akan menggerakkan suatu cylinder kerja di mana cylinder inilah yang nantinya akan mengubah tenaga atau tekanan udara menjadi tenaga mekanik (gerakan maju mundur).

1. Konsep Pneumatic Valve, Pneumatic Cylinder, dan Relay

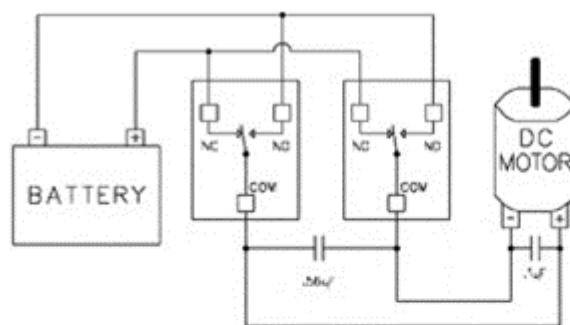
Pneumatic valve merupakan sebuah aktuator dengan pneumatic cylinder. Pada robot akan digunakan satu buah pneumatik dengan panjang silinder adalah cm.

Pengendalian pneumatik akan dilakukan melalui dua buah relay. Relay adalah suatu saklar yang digerakkan secara elektromagnetik. Bila sumber tegangan diberikan pada kumparan solenoid, maka akan terbangkit suatu medan elektromagnetik yang mengakibatkan tertariknya armatur ke inti kumparan. Armature tersebut menggerakkan kontak relay apakah menutup atau membuka sesuai dengan perancangannya. Pegas akan mengembalikan armatur ke posisi semula jika arus listrik yang mengalir ke kumparan tidak ada.

Relay tersebut akan dikirimkan sinyal berupa sinyal digital dari mikrokontroler sehingga berfungsi sebagai saklar yang akan memberikan arus pada pneumatik valve sehingga akan membuka katup sehingga melewati tekanan udara sehingga pneumatik cylinder akan memanjang. pneumatik ini akan digunakan sebagai sistem penjepit pada pencap bola pada robot. Karena pneumatik valve yang digunakan memiliki dua buah katup maka dibutuhkan dua channel dari relay agar berlogika saklar tegangan yakni akan membuka dan menutup salah satu dari kedua katup atau menutup kedua katup bersamaan. Mekanisme algoritme dari pengendalian pneumatik dapat dilihat melalui tabel berikut.

	Kondisi Input Relay			
Relay 1	HIGH	LOW	LOW	HIGH
Relay 2	LOW	HIGH	LOW	HIGH
Kondisi pneumatik Cylinder	Silinder memanjang	Silinder memendek	Reset	Relay terbakar

Tabel Mekanisme Algoritma Relay



Relay 1 Off Relay 2 Off = Motor Brake to +
 Relay 1 On Relay 2 Off = Motor Forward
 Relay 1 Off Relay 2 On = Motor Backward
 Relay 1 On Relay 2 On = Motor Brake to -
 .56uF Induction Capacitor Should be Located Near Relays
 .1uF Filter Capacitor Should be Located Near Motor
 Additional Capacitors May be Desirable for Some Motors

Gambar Mekanisme Algoritma Relay



Gambar Pneumatic Valve (kiri) dan Pneumatic Cylinder (kanan)

2. Pengendalian Sistem Pneumatic dengan Relay

1. Ketiklah program berikut:

```

sketch_sep17a$
1 int Relay = 30;
2
3 void setup() {
4   pinMode(Relay, OUTPUT);
5 }
6
7 void loop() {
8   Serial.println("Nyala");
9   digitalWrite (Relay, HIGH);
10  delay(1500);
11  digitalWrite (Relay, LOW);
12  Serial.println("Mati");
13  delay(1500);
14 }

```

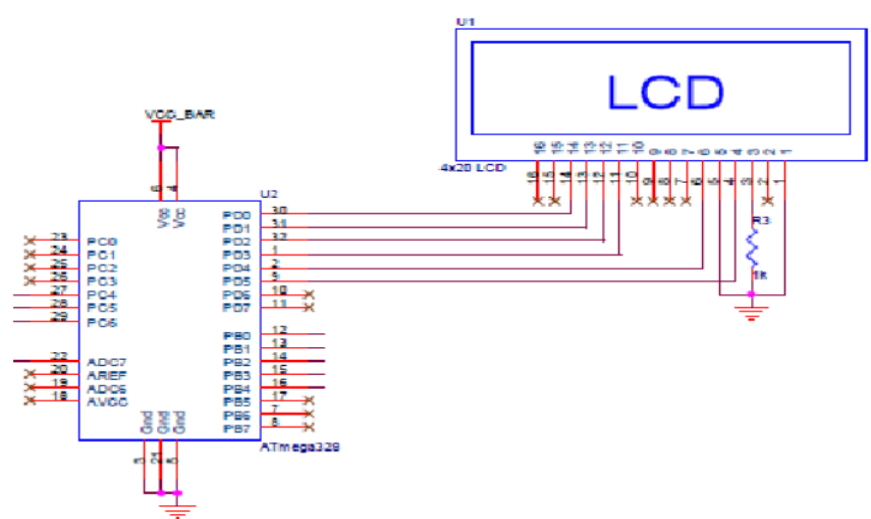
Gambar Test Relay

2. Setelah relay berhasil dijalankan, berkolaborasi dengan divisi electronic untuk dapat menghubungkan wiring pneumatic dengan relay.
3. Setelah wiring selesai, uji coba keberhasilan interfacing antara relay dengan pneumatic berdasarkan program di atas.

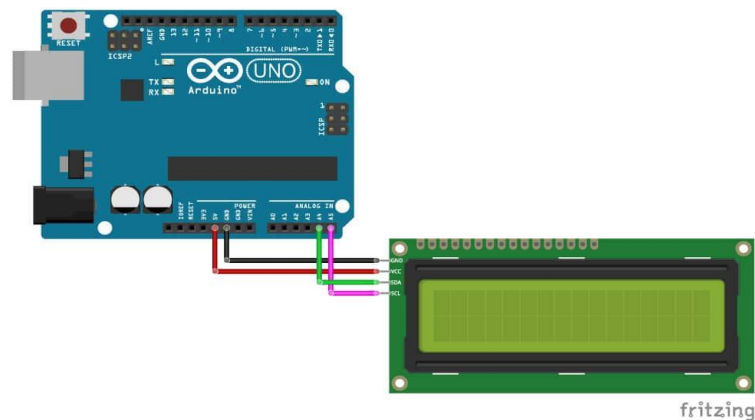
d. LCD (Liquid Crystal Display)



LCD adalah layar kecil yang digunakan untuk menampilkan suatu informasi, LCD yang kita gunakan di blakasutha adalah LCD yang berukuran 16x2 pixel, namun tidak menutup kemungkinan untuk menggunakan LCD jenis lain seperti oled dan sebagainya. Umumnya LCD menggunakan modul I2C (Inter-Integrated Circuit) untuk mempermudah proses perangkaian dan menghemat pin pada arduino.

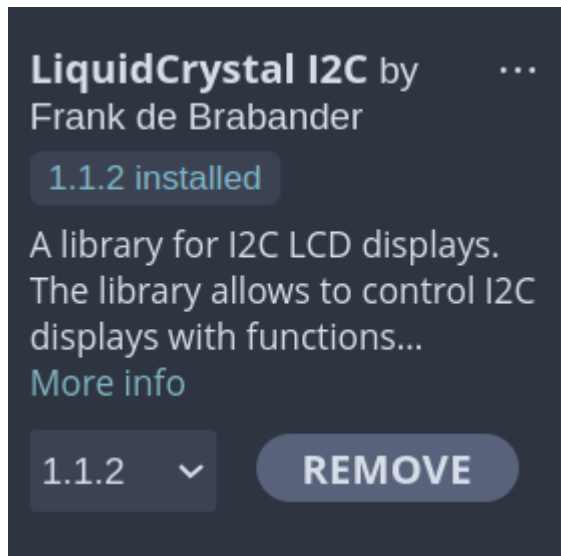


Di atas adalah gambar LCD tanpa modul I2C, bandingkan dengan gambar di bawah yang menggunakan modul I2C.



Maka dari itu kita akan menggunakan modul I2C.

1. Download library berikut.



https://github.com/johnrickman/LiquidCrystal_I2C

2. Buat projek baru dan ketikkan kode berikut:

```
1  #include <Wire.h>
2  #include <LiquidCrystal_I2C.h>
3
4  LiquidCrystal_I2C lcd(0x27,16,2);
5
6  void setup()
7  {
8      lcd.init();
9      lcd.backlight();
10     lcd.setCursor(0,0);
11     lcd.print("halo dek");
12     lcd.setCursor(0,1);
13     lcd.print("lagi sibuk ga?");
14 }
15
16
17 void loop()
18 {
19 }
```

- a. Line 1 dan 2 digunakan untuk mengimport library yang diperlukan.
- b. Line 4 adalah membuat object lcd, parameternya adalah 0x27 yang merupakan alamat dari I2C (jika belum dirubah maka akan sama), selanjutnya adalah 16 dan 2 yang merupakan ukuran dari layar LCD.
- c. Line 8 digunakan untuk menginisialisasi LCD.
- d. Line 9 menyalakan backlight.
- e. Line 10 memindahkan kursor ke koordinat (0, 0) atau pojok kiri atas
- f. Line 11 menuliskan pesan yang diinginkan.
- g. Line 12 dan 13 melakukan hal yang sama.

3. Sambungkan arduino dengan mengikuti schematic di atas.
4. Upload kode ke arduino.

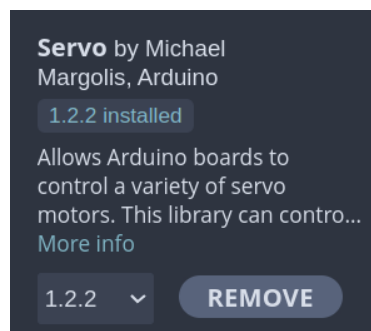
e. Servo

Servo adalah komponen penggerak (aktuator) yang dapat menentukan posisi sudut dari poros output. Pada servo terdapat sebuah motor dc mini sebagai penggerak, gear dan juga sistem elektronik untuk kontrol sistem secara tertutup dan potensiometer. Pada intinya bagaimana cara kerja dari servo adalah, ia akan menerima sinyal input berupa sinyal PWM dari kontroler (ada 3 pin, masing-masing Vcc, Gnd dan Signal). Dari nilai PWM tertentu yang diberikan, maka motor akan berputar, bersamaan dengan itu putaran motor membuat sistem gear pada servo ikut berputar begitu juga dengan potensiometer yang ada pada servo. Dari potensiometer, akan dihasilkan output berupa tegangan listrik yang digunakan sebagai masukan sistem elektronik kontrolnya untuk dilakukan komparasi nilai tegangan terhadap nilai tegangan dari PWM yang diberikan, jika nilainya belum sama maka motor akan terus berputar dan jika tidak maka motor akan berhenti. Namun tidak perlu khawatir terkait bagaimana cara membuat algoritme pengendalian servo karena pada Arduino IDE sudah terdapat library untuk servo motor, sehingga kita hanya cukup memasukkan nilai derajat putaran servo untuk membuat servo dapat menyesuaikan posisinya.



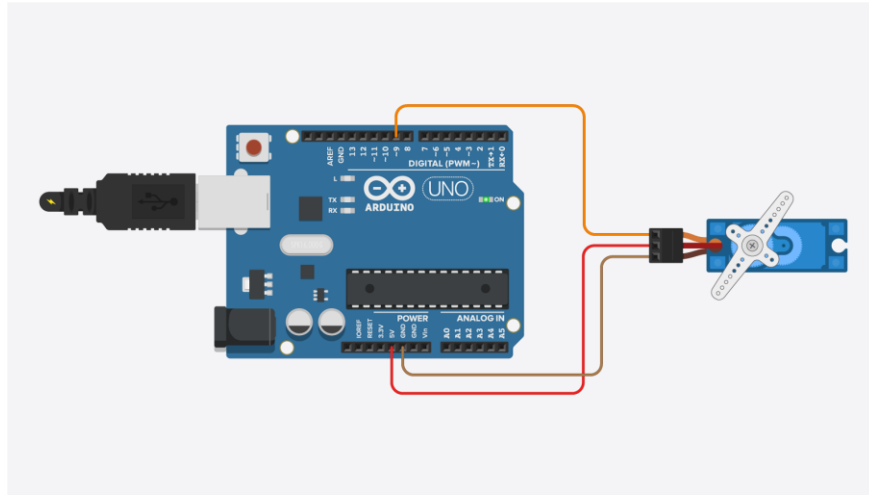
Berdasarkan sudut putarannya, servo dibagi menjadi 2 yaitu servo 180° dan servo 360°. Berikut langkah untuk mengoperasikan servo:

1. Download library berikut.



<https://github.com/arduino-libraries/Servo>

2. Sambungkan servo dengan schematic berikut.



3. Ketikkan kode berikut

```
1  #include <Servo.h>
2
3  Servo myservo;
4
5  int pos = 0;
6
7  void setup() {
8      myservo.attach(9);
9  }
10
11 void loop() {
12     for (pos = 0; pos <= 180; pos += 1) {
13         myservo.write(pos);
14         delay(15);
15     }
16     for (pos = 180; pos >= 0; pos -= 1) {
17         myservo.write(pos);
18         delay(15);
19     }
20 }
```

- Line 1 mengimport library yang diperlukan
 - Line 3 membuat objek myservo
 - Line 5 mendeklarasikan posisi servo
 - Line 8 menyambungkan kabel sinyal pada servo ke objek myservo
 - Line 12 sampai 15 memutar servo dari sudut 0 sampai 180
 - Line 16 sampai 20 memutar servo dari sudut 180 sampai 0
4. Upload kode ke arduino.

BAB V

KOMPONEN GERAK ROBOT BERODA DAN ODOMETRI

A. Konsep Kemudi dan Roda Pergerakan

Robot beroda adalah salah satu jenis robot yang paling umum digunakan dalam berbagai aplikasi karena kemudahan dalam desain dan kontrolnya. Namun, pergerakan pada robot beroda memerlukan pemahaman yang kuat mengenai mekanika dasar serta kinematika serta dinamika dari roda. Pergerakan robot beroda dipengaruhi oleh jenis roda dan konfigurasi kemudi yang digunakan. Berikut adalah penjelasan lengkap mengenai konsep pergerakan pada robot beroda, jenis-jenis kemudi, dan jenis-jenis roda kemudi.

a. Konsep Pergerakan pada Robot Beroda

Robot beroda bergerak dengan mengubah kecepatan dan arah rotasi roda-roda mereka. Untuk memodelkan gerakan ini, kinematika robot digunakan untuk menentukan hubungan antara kecepatan motor dengan pergerakan robot secara keseluruhan. Kinematika sering kali dibagi menjadi dua jenis:

- Kinematika Maju (Forward Kinematics): Digunakan untuk menghitung posisi dan orientasi robot berdasarkan kecepatan roda yang diketahui.
- Kinematika Mundur (Inverse Kinematics): Digunakan untuk menentukan kecepatan roda yang dibutuhkan agar robot mencapai posisi atau orientasi tertentu.

Dalam pergerakan robot beroda, dua faktor utama mempengaruhi arah dan lintasan:

- Kecepatan Translasi: Kecepatan gerakan maju, mundur, atau ke samping.
- Kecepatan Rotasi: Kecepatan perubahan orientasi robot (putaran di tempat).

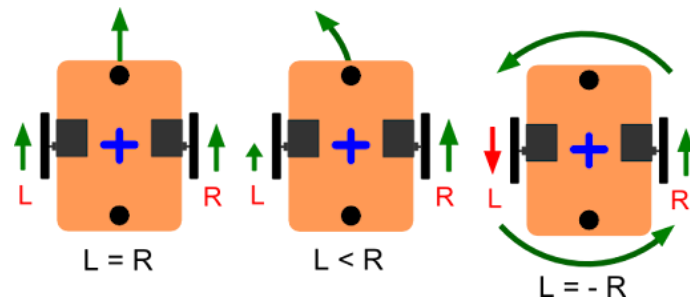
Pada robot beroda, kedua kecepatan ini dapat dikombinasikan untuk menghasilkan berbagai macam pola pergerakan.

b. Jenis-jenis Kemudi pada Robot Beroda

Jenis kemudi pada robot beroda sangat menentukan fleksibilitas dan kemampuan manuver robot. Berikut adalah beberapa jenis kemudi yang sering digunakan dalam robotika: (Holonomic => Kemampuan untuk bergerak ke berbagai arah tanpa merubah arah orientasi)

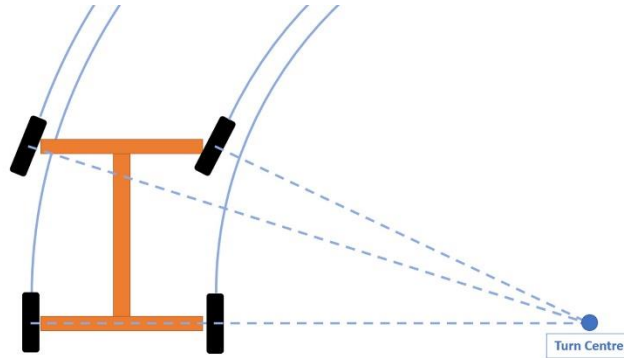
a. Differential Drive => Non Holonomic

- Deskripsi: Pada kemudi jenis ini, robot memiliki dua roda penggerak yang dipasang pada sisi kiri dan kanan. Setiap roda dapat bergerak dengan kecepatan yang berbeda untuk mengarahkan robot.
- Keunggulan: Mudah dikendalikan dan sederhana dalam desain.
- Kekurangan: Memiliki keterbatasan dalam pergerakan lateral (gerak ke samping), serta membutuhkan banyak ruang untuk berbelok.
- Contoh: Kebanyakan robot mobile sederhana dan robot pembersih otomatis menggunakan sistem kemudi ini.



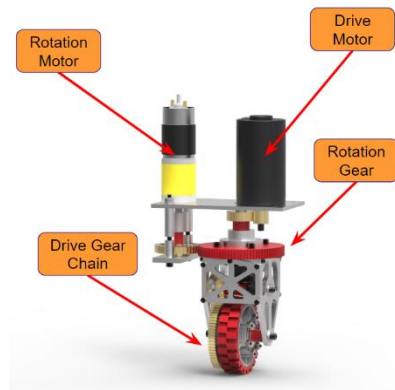
b. Steering Drive (Kemudi Mobil) / Ackerman Drive => Non Holonomic

- Deskripsi: Seperti mobil konvensional, sistem ini memiliki satu set roda untuk penggerak (*driving wheels*) dan satu set roda yang dapat berbelok (*steering wheels*).
- Keunggulan: Baik untuk gerakan yang presisi dalam jalur lurus dan tikungan.
- Kekurangan: Tidak bisa bergerak lateral, dan lebih kompleks dalam kontrol dibanding differential drive.
- Contoh: Kendaraan autonomous dan beberapa robot transportasi.



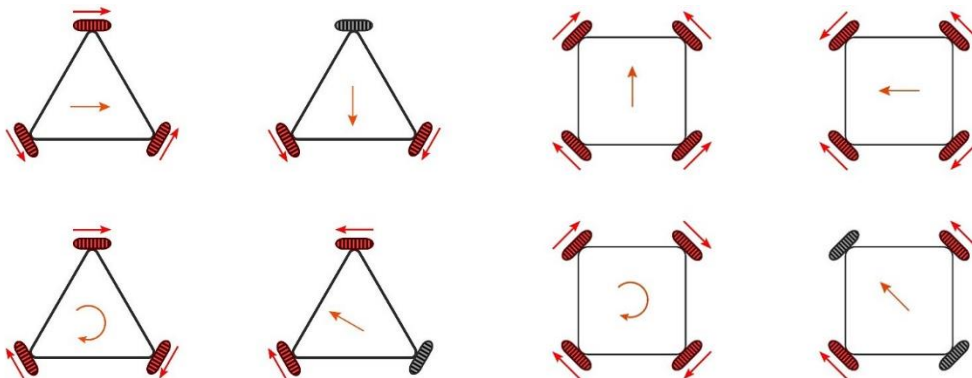
c. Swerve Drive => Holonomic

- Deskripsi: Setiap roda pada robot dapat berputar secara independen (baik untuk rotasi maupun translasi), memberikan kemampuan manuver yang sangat fleksibel. Terdapat 2 jenis Swerve Drive, yakni tipe Coaxial (motor digunakan masing-masing untuk mengontrol sudut roda dan kecepatan roda secara independent) serta tipe Diferensial (penyesuaian arah roda dan kecepatan dikontrol oleh 2 buah motor sekaligus yang bekerja secara bersama-sama).
- Keunggulan: Mampu bergerak ke segala arah tanpa perlu berbelok, manuver sangat baik di area sempit.
- Kekurangan: Kompleks secara mekanis dan kontrol.
- Contoh: Robot-robot yang membutuhkan manuver cepat dan presisi tinggi seperti robot kompetisi.



d. Omnidirectional Drive => Holonomic

- Deskripsi: Memiliki roda-roda omniwheel atau mecanum yang dapat bergerak ke segala arah. Setiap roda dapat berkontribusi pada pergerakan translasi maupun rotasi.
- Keunggulan: Sangat fleksibel dan memungkinkan pergerakan lateral serta berbelok dengan mudah.
- Kekurangan: Desain roda lebih rumit dan membutuhkan kontrol yang lebih kompleks.
- Contoh: Banyak digunakan dalam robot industri dan kompetisi robotika.



c. Jenis-jenis Roda Kemudi pada Robot

Jenis roda yang digunakan sangat memengaruhi kemampuan manuver robot. Berikut adalah beberapa jenis roda yang umum digunakan pada robot beroda:

a. Roda Konvensional (Fixed Wheel)

- Deskripsi: Roda biasa yang hanya dapat bergerak maju atau mundur.
- Keunggulan: Stabil dan sederhana dalam desain.
- Kekurangan: Tidak memungkinkan pergerakan lateral (gerakan ke samping).
- Contoh: Kendaraan sehari-hari dan robot yang menggunakan steering drive.



b. Caster Wheel

- Deskripsi: Roda bebas yang dipasang pada sumbu putar, sering digunakan untuk stabilitas tambahan pada robot.
- Keunggulan: Menambah stabilitas robot, khususnya pada desain robot tiga atau empat roda.
- Kekurangan: Membatasi manuver robot, khususnya pada gerakan lateral.
- Contoh: Banyak digunakan pada robot pembersih otomatis.



c. Roda Omni (Omniwheel)

- Deskripsi: Roda yang memiliki rol-rol kecil yang dipasang di sepanjang lingkaran roda. Rol ini memungkinkan pergerakan lateral tanpa harus memutar roda utama.
- Keunggulan: Fleksibilitas tinggi, memungkinkan pergerakan di segala arah.
- Kekurangan: Kompleks dalam hal kontrol, serta efisiensi energi tidak sebaik roda konvensional.
- Contoh: Digunakan dalam sistem omnidirectional drive dan holonomic drive.



d. Roda Mecanum

- Deskripsi: Mirip dengan roda omni, tetapi rol dipasang dengan sudut tertentu terhadap sumbu roda, memungkinkan gerakan diagonal dan lateral.
- Keunggulan: Fleksibilitas tinggi dan memungkinkan pergerakan kompleks di ruang yang sempit.
- Kekurangan: Sama seperti roda omni, kontrol lebih rumit, dan efisiensi energi lebih rendah.
- Contoh: Banyak digunakan pada robot industri atau robot dengan mobilitas tinggi.



Jenis Kemudi	Jenis Roda yang Umum Digunakan	Manuver Lateral	Kompleksitas Kontrol	Aplikasi Utama
Differential Drive	Roda Konvensional	Tidak	Rendah	Robot sederhana, pembersih
Steering Drive	Roda Konvensional	Tidak	Sedang	Kendaraan autonomous
Swerve Drive	Roda Konvensional	Ya	Tinggi	Kompetisi robot, manuver cepat
Omnidirectional	Omnivheel atau Mecanum	Ya	Tinggi	Robot industri, robot kompetisi

B. Odometri

Odometri adalah metode untuk memperkirakan posisi dan orientasi (pose) robot dalam suatu lingkungan berdasarkan informasi dari sensor gerak, seperti rotary encoder atau sensor IMU (Inertial Measurement Unit). Ini adalah teknik penting dalam navigasi robot karena memberikan perkiraan posisi secara real-time yang memungkinkan robot untuk bergerak secara otonom di lingkungan yang tidak diketahui atau terstruktur.

Odometri menggunakan data dari sensor gerak untuk menghitung perubahan posisi dan orientasi robot dari waktu ke waktu. Pada robot beroda, informasi ini biasanya diambil dari putaran roda yang diukur menggunakan **rotary encoder**. Pada robot yang lebih canggih, odometri juga dapat ditingkatkan menggunakan **IMU** yang memberikan informasi tentang percepatan dan rotasi.

Prinsip dasar odometri adalah menghitung jarak yang ditempuh dan perubahan orientasi robot berdasarkan kecepatan sudut roda dan waktu. Proses ini dilakukan secara terus menerus untuk memperbarui posisi robot di ruang koordinat global.

a. Posisi dan Orientasi

- **Posisi** mengacu pada koordinat robot dalam ruang dua dimensi atau tiga dimensi, biasanya dinyatakan sebagai (x, y) atau (x, y, z) .
- **Orientasi** merujuk pada arah kepala robot dalam sistem koordinat, biasanya dinyatakan sebagai sudut θ dalam ruang dua dimensi (yaw dalam ruang tiga dimensi).

b. Pose Robot

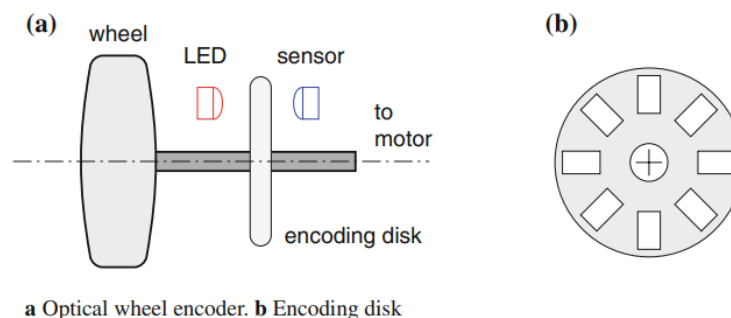
Pose menggabungkan posisi dan orientasi dalam satu sistem koordinat. Pose dalam ruang dua dimensi dinyatakan sebagai (x, y, θ) .

C. Sensor Odometri Dasar

a. Rotary Encoder

Rotary encoder adalah sensor yang dipasang pada sumbu roda untuk mengukur rotasi roda. Encoder ini bekerja dengan menghitung jumlah putaran (atau sebagian putaran) roda dan mengubahnya menjadi sinyal digital yang dapat diproses oleh mikrokontroler. Keliling dari sebuah roda adalah $2\pi r$, di mana r adalah jari-jari roda dalam cm, jadi jika n rotasi dihitung, kita tahu bahwa robot telah bergerak sejauh $2\pi nr$ cm. Encoder dapat dibuat yang mengukur pecahan revolusi. Jika sebuah sinyal dihasilkan 8 kali per revolusi, jarak yang ditempuh adalah $2\pi nr/8$ cm, dimana n adalah jumlah sinyal yang dihitung oleh komputer.

Ada banyak cara yang berbeda untuk mengimplementasikan encoder. Desain yang populer adalah dengan menggunakan sumber cahaya seperti dioda pemancar cahaya (LED), sensor cahaya dan sebuah piringan penyandi yang dipasang pada sumbu roda (a). Piringan tersebut adalah dilubangi dengan lubang (b) sehingga setiap kali lubang berlawanan dengan sumber Cahaya sumber cahaya, sensor menghasilkan sinyal.



• Jenis-jenis:

- **Incremental Encoder:** Menghitung perubahan sudut relatif terhadap posisi awal. Biasanya digunakan untuk mengukur kecepatan dan jarak tempuh.

- **Absolute Encoder:** Mengukur sudut rotasi secara absolut, memberikan informasi posisi yang lebih akurat dibandingkan incremental encoder, berbeda dari encoder incremental, tipe ini memberikan data nilai tertentu berdasarkan posisi orientasinya, jadi data hasil pembacaan bernilai spesifik untuk satu nilai orientasi tertentu.
- **Aplikasi:** Digunakan pada robot beroda untuk menghitung jarak tempuh dan kecepatan rotasi roda.

b. Inertial Measurement Unit (IMU)

Inertial Measurement Unit (IMU) adalah perangkat elektronik yang mengukur dan melaporkan **percepatan linear**, **kecepatan angular**, dan **orientasi** suatu objek. IMU sering digunakan dalam berbagai aplikasi robotika, drone, kendaraan otonom, dan perangkat mobile untuk membantu navigasi dan stabilisasi.

- Komponen Utama IMU

IMU biasanya terdiri dari tiga sensor utama:

- **Akselerometer**
 - **Fungsi:** Mengukur percepatan linear dalam tiga sumbu (X, Y, Z).
 - **Aplikasi:** Deteksi gerakan, pengukuran kecepatan, dan penghitungan posisi relatif melalui integrasi percepatan.
- **Gyroscope (Giroskop)**
 - **Fungsi:** Mengukur kecepatan angular atau rotasi dalam tiga sumbu (roll (x), pitch (y), yaw (z)).
 - **Aplikasi:** Menentukan perubahan orientasi, stabilisasi platform, dan kontrol gerakan.
- **Magnetometer** (opsional pada beberapa IMU)
 - **Fungsi:** Mengukur medan magnet bumi untuk menentukan arah kompas.
 - **Aplikasi:** Koreksi drift gyroscope, penentuan heading absolut.

- Cara Kerja IMU

IMU bekerja dengan menggabungkan data dari akselerometer dan gyroscope (serta magnetometer jika tersedia) untuk menentukan **pose** (posisi dan orientasi) objek. Proses ini sering melibatkan **sensor fusion** (penggabungan data dari berbagai sensor untuk meningkatkan akurasi dan mengurangi kesalahan).

- Sensor Fusion dan Algoritma

Untuk menghasilkan informasi orientasi yang akurat, data dari IMU sering diproses menggunakan algoritma sensor fusion seperti:

- **Filter Kalman**
 - Menggabungkan data dari berbagai sensor dengan mempertimbangkan ketidakpastian dan noise.
- **Complementary Filter**
 - Menggabungkan data gyroscope dan akselerometer dengan cara yang sederhana dan efisien dengan cara menghitung nilai akhir data dari dua atau lebih data sensor dengan mempertimbangkan bobot kepercayaan pembacaan dari masing-masing sensor, seperti menghitung rerata tetapi setiap elemen nya dikalikan dengan sebuah konstanta (bobot).
- **Mahony dan Madgwick Filters**
 - Algoritma yang dioptimalkan untuk aplikasi real-time dengan computational efficiency tinggi.

- Aplikasi IMU dalam Robotika

IMU memainkan peran krusial dalam berbagai aspek robotika, antara lain:

- **Navigasi dan Pelacakan**
 - Menghitung posisi dan orientasi robot secara real-time, terutama pada lingkungan yang tidak memiliki akses GPS.
- **Stabilisasi dan Kontrol Gerakan**
 - Menjaga keseimbangan robot, terutama pada robot humanoid atau drone.
- **Pengendalian Lengan Robot**
 - Mengatur pergerakan dan posisi lengan dengan presisi tinggi.
- **Interaksi Lingkungan**
 - Mengukur gaya dan percepatan yang dialami robot saat berinteraksi dengan objek atau medan eksternal.

- Keunggulan dan Keterbatasan IMU

Keunggulan:

- **Portabilitas:** Ukuran kecil dan ringan, mudah diintegrasikan ke dalam berbagai sistem.
- **Respons Cepat:** Memberikan data real-time untuk kontrol dan navigasi.
- **Tidak Bergantung pada Sinyal Eksternal:** Berfungsi baik di dalam ruangan atau area tanpa sinyal GPS.

Keterbatasan:

- **Drift dan Akumulasi Error:** Gyroscope dapat mengalami drift seiring waktu, yang mempengaruhi akurasi orientasi.
- **Noise dan Kalibrasi:** Sensor dapat terpengaruh oleh noise elektronik dan memerlukan kalibrasi rutin untuk mempertahankan akurasi.

- Pemilihan IMU yang Tepat

Beberapa faktor yang perlu dipertimbangkan saat memilih IMU untuk aplikasi robotika meliputi:

- **Resolusi dan Akurasi:** Tingkat presisi pengukuran yang dibutuhkan.

- **Range Pengukuran:** Rentang percepatan dan kecepatan angular yang dapat diukur.
- **Frekuensi Sampling:** Kecepatan pengambilan data yang diperlukan untuk aplikasi real-time.
- **Konsumsi Daya:** Penting untuk aplikasi portabel dan robot yang mengandalkan baterai.
- **Integrasi dan Kompatibilitas:** Kemudahan integrasi dengan mikrokontroler atau sistem kontrol yang digunakan.

- Contoh IMU Populer

Beberapa modul IMU yang sering digunakan dalam robotika dan proyek mekatronika meliputi:

- **MPU-6050:** Menggabungkan akselerometer dan gyroscope, populer untuk aplikasi dasar dan edukasi.
- **IMU MPU-9250:** Menambahkan magnetometer untuk sensor fusion yang lebih akurat.
- **BNO055:** IMU yang dilengkapi dengan sensor fusion terintegrasi, memberikan data orientasi langsung.

- Implementasi IMU dalam Sistem Kontrol Robot

Integrasi IMU dalam sistem kontrol robot melibatkan beberapa langkah:

1. **Pengambilan Data:** Membaca data sensor dari akselerometer, gyroscope, dan magnetometer.
2. **Sensor Fusion:** Menggunakan algoritma untuk menggabungkan data sensor dan menghitung orientasi.
3. **Feedback Control:** Menggunakan informasi pose untuk mengatur aktuator dan mengendalikan gerakan robot.
4. **Kalibrasi dan Pemeliharaan:** Melakukan kalibrasi rutin untuk memastikan akurasi data sensor.

BAB VI

PEMODELAN PERGERAKAN ROBOT

A. Pengenalan Kinematika

Kinematika dalam fisika berkaitan dengan analisa pergerakan benda dengan mengabaikan sebab terjadinya benda tersebut bergerak, dalam hal ini berfokus pada aspek jarak, kecepatan, percepatan dan waktu saja tanpa melibatkan gaya dan massa. Kinematika tidak memperhatikan mengapa suatu benda bergerak, melainkan hanya bagaimana benda tersebut bergerak. Ini membuat kinematika berbeda dengan dinamika, yang juga mempelajari gaya yang menyebabkan pergerakan.

Dalam kinematika, terdapat beberapa parameter dasar yang digunakan untuk mendeskripsikan gerakan suatu benda, yaitu:

- Posisi (x, y, z): Posisi adalah lokasi suatu benda pada suatu waktu tertentu dalam sistem koordinat tertentu. Posisi dapat dinyatakan dalam koordinat kartesian (x, y, z) atau dalam bentuk vektor posisi.
- Perpindahan (Δx): Perpindahan adalah perubahan posisi suatu benda dalam suatu periode waktu. Perpindahan adalah besaran vektor, yang berarti memiliki magnitudo dan arah.
- Jarak Tempuh: Jarak tempuh adalah panjang lintasan yang dilalui oleh benda selama bergerak dari satu titik ke titik lainnya. Berbeda dengan perpindahan, jarak tempuh adalah besaran skalar dan tidak memperhatikan arah.

Kecepatan (v): Kecepatan adalah laju perubahan posisi terhadap waktu. Kecepatan juga merupakan besaran vektor, yang artinya memiliki arah dan magnitudo. Kecepatan rata-rata dapat dihitung dengan persamaan:

$$[v = \frac{\Delta x}{\Delta t}]$$

Di mana (Δx) adalah perpindahan dan (Δt) adalah selang waktu. Kecepatan sesaat adalah kecepatan pada suatu titik waktu tertentu dan dapat diperoleh dengan turunan pertama dari posisi terhadap waktu.

- Laju: Laju adalah besaran skalar yang menunjukkan seberapa cepat suatu benda bergerak tanpa memperhatikan arah. Laju dihitung sebagai jarak yang ditempuh dibagi dengan waktu yang diperlukan.
- Percepatan (a): Percepatan adalah laju perubahan kecepatan terhadap waktu. Seperti kecepatan, percepatan juga merupakan besaran vektor. Percepatan rata-rata dapat dihitung dengan persamaan:

$$[a = \frac{\Delta v}{\Delta t}]$$

Di mana (Δv) adalah perubahan kecepatan dan (Δt) adalah selang waktu. Percepatan sesaat adalah percepatan pada suatu titik waktu tertentu dan dapat diperoleh

dengan turunan pertama dari kecepatan terhadap waktu, atau turunan kedua dari posisi terhadap waktu.

a. Persamaan Gerak Lurus Beraturan (GLB) dan Gerak Lurus Berubah Beraturan (GLBB)

Gerak Lurus Beraturan (GLB):

Gerak lurus beraturan terjadi ketika sebuah benda bergerak dengan kecepatan konstan, artinya tidak ada percepatan ($a = 0$).

Persamaan posisi sebagai fungsi waktu untuk GLB:

$$[x(t) = x_0 + v \cdot t]$$

Di mana:

- ($x(t)$) adalah posisi benda pada waktu t
- (x_0) adalah posisi awal
- (v) adalah kecepatan konstan
- (t) adalah waktu

Gerak Lurus Berubah Beraturan (GLBB):

Gerak lurus berubah beraturan terjadi ketika sebuah benda bergerak dengan percepatan konstan. Persamaan posisi sebagai fungsi waktu untuk GLBB :

$$[x(t) = x_0 + v_0 \cdot t + \frac{1}{2} a \cdot t^2]$$

Di mana:

- ($x(t)$) adalah posisi benda pada waktu t
- (x_0) adalah posisi awal
- (v_0) adalah kecepatan awal
- (a) adalah percepatan konstan
- (t) adalah waktu

Persamaan kecepatan sebagai fungsi waktu untuk GLBB:

$$[v(t) = v_0 + a \cdot t]$$

Di mana:

- ($v(t)$) adalah kecepatan benda pada waktu t
- (v_0) adalah kecepatan awal
- (a) adalah percepatan konstan

- (t) adalah waktu

b. Gerak Rotasi

Gerak rotasi adalah gerakan di mana suatu benda berputar mengelilingi suatu sumbu. Dalam gerak rotasi, analog dari posisi, kecepatan, dan percepatan adalah sebagai berikut:

- Sudut (θ): Sudut rotasi, biasanya diukur dalam radian, menggambarkan seberapa jauh suatu benda telah berputar dari posisi awalnya.
- Kecepatan Sudut (ω): Kecepatan sudut adalah laju perubahan sudut terhadap waktu, dan diukur dalam radian per detik (rad/s). Kecepatan sudut rata-rata dapat dihitung dengan persamaan:

$$[\omega = \frac{\Delta\theta}{\Delta t}]$$

Di mana ($\Delta\theta$) adalah perubahan sudut dan (Δt) adalah selang waktu.

- Percepatan Sudut (α): Percepatan sudut adalah laju perubahan kecepatan sudut terhadap waktu, dan diukur dalam radian per detik kuadrat (rad/s²). Percepatan sudut rata-rata dapat dihitung dengan persamaan:

$$[\alpha = \frac{\Delta\omega}{\Delta t}]$$

Di mana ($\Delta\omega$) adalah perubahan kecepatan sudut dan (Δt) adalah selang waktu.

Persamaan gerak rotasi, yang merupakan analog dari gerak lurus, adalah:

- Persamaan sudut sebagai fungsi waktu untuk gerak dengan percepatan sudut konstan:

$$[\theta(t) = \theta_0 + \omega_0 \cdot t + \frac{1}{2} \alpha \cdot t^2]$$

- Persamaan kecepatan sudut sebagai fungsi waktu:

$$[\omega(t) = \omega_0 + \alpha \cdot t]$$

c. Gerak Dua Dimensi dan Tiga Dimensi

Dalam gerak dua dimensi (2D) dan tiga dimensi (3D), posisi, kecepatan, dan percepatan dinyatakan sebagai vektor. Misalnya, dalam koordinat kartesian:

- Posisi:

$$[r(t) = (x(t)y(t)) \text{ atau } r(t) = \begin{pmatrix} x(t) \\ y(t)z(t) \end{pmatrix}]$$

- Kecepatan:

$$[v(t) = \frac{dr(t)}{dt} = (\dot{x}(t)\dot{y}(t)) \text{ atau } v(t) = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t)\dot{z}(t) \end{pmatrix}]$$

- Percepatan:

$$[a(t) = \frac{dv(t)}{dt} = (\ddot{x}(t)\ddot{y}(t)) \text{ atau } a(t) = \begin{pmatrix} \ddot{x}(t) \\ \ddot{y}(t)\ddot{z}(t) \end{pmatrix}]$$

Dalam gerak 2D dan 3D, kinematika menjadi lebih kompleks karena harus mempertimbangkan lebih dari satu komponen gerak secara simultan.

d. Kinematika Robot Beroda

Kinematika robot beroda adalah bagian dari kinematika yang berfokus pada studi gerak robot yang menggunakan roda sebagai alat untuk bergerak. Pemahaman tentang kinematika ini sangat penting dalam desain, pengendalian, dan navigasi robot beroda, baik itu robot otonom maupun robot yang dikendalikan secara manual. Kinematika robot beroda melibatkan analisis hubungan antara kecepatan roda dengan kecepatan gerak robot secara keseluruhan, termasuk posisi, orientasi, kecepatan linear, dan kecepatan angular (rotasi).

Jenis-jenis Robot Beroda

Terdapat beberapa jenis konfigurasi robot beroda yang umum digunakan, masing-masing memiliki karakteristik kinematika yang berbeda:

1. Robot Diferensial (Differential Drive)
2. Robot Traksi (Tricycle Drive)
3. Robot Holonomik (Omniwheel, Mecanum Wheel, dan Swerve Drive)

Masing-masing konfigurasi ini memiliki persamaan kinematika sendiri yang menghubungkan kecepatan roda dengan gerakan keseluruhan robot.

a. Kinematika Robot Diferensial

Robot diferensial adalah salah satu konfigurasi robot beroda yang paling umum. Robot ini biasanya memiliki dua roda penggerak yang dipasang pada sumbu yang sama dan satu atau lebih roda tambahan untuk menjaga keseimbangan (caster wheel).

Prinsip Kerja

- Robot diferensial bergerak dengan mengontrol kecepatan dari dua roda penggeraknya (kanan dan kiri).
- Jika kedua roda bergerak dengan kecepatan yang sama, robot akan bergerak dalam garis lurus.
- Jika kedua roda bergerak dengan kecepatan yang berbeda, robot akan berputar atau bergerak dalam lintasan melengkung.

Persamaan Kinematika

Untuk robot diferensial, kecepatan linier (v) dan kecepatan angular (ω) dari robot dapat dinyatakan sebagai:

$$[v = \frac{v_r + v_l}{2}]$$

$$[\omega = \frac{v_r - v_l}{L}]$$

Di mana:

- (v_r) adalah kecepatan roda kanan.
- (v_l) adalah kecepatan roda kiri.
- (L) adalah jarak antara roda kanan dan roda kiri (track width).

Kinematika Maju (Forward Kinematics)

Kinematika maju digunakan untuk menghitung kecepatan linier dan angular robot berdasarkan kecepatan roda:

$$[v = \frac{r}{2} (\omega_r + \omega_l)]$$

$$[\omega = \frac{r}{L} (\omega_r - \omega_l)]$$

Di mana:

- (r) adalah jari-jari roda.
- (ω_r) dan (ω_l) adalah kecepatan angular roda kanan dan kiri.

Kinematika Mundur (Inverse Kinematics)

Kinematika mundur digunakan untuk menghitung kecepatan roda yang diperlukan untuk mencapai kecepatan linier dan angular tertentu:

$$[\omega_r = \frac{2v + \omega L}{2r}]$$

$$[\omega_l = \frac{2v - \omega L}{2r}]$$

b. Kinematika Robot Traksi

Robot traksi memiliki tiga roda: satu roda depan yang dapat digerakkan dan diatur arah geraknya, dan dua roda belakang yang bebas. Model ini sering digunakan pada kendaraan, seperti sepeda motor dan mobil.

Prinsip Kerja

- Roda depan digunakan untuk mengatur arah gerakan, sementara roda belakang digerakkan untuk memberikan kecepatan.
- Pengendalian arah dilakukan dengan mengubah sudut kemudi roda depan.

Persamaan Kinematika

Kecepatan linier (v) dan kecepatan angular (ω) dari robot traksi dapat dinyatakan sebagai:

$$[v = \sqrt{v_x^2 + v_y^2}]$$

$$[\omega = \frac{v_x \sin \theta}{d}]$$

Di mana:

- (v_x) dan (v_y) adalah komponen kecepatan pada sumbu x dan y.
- (θ) adalah sudut kemudi.
- (d) adalah jarak antara roda belakang dan roda depan.

Kinematika Maju

Untuk kecepatan maju (v_x) dan kecepatan samping (v_y), kita bisa menggunakan persamaan:

$$[v_x = v \cos \theta]$$

$$[v_y = v \sin \theta]$$

Kinematika Mundur

Untuk mencapai kecepatan tertentu pada roda depan dan roda belakang, sudut kemudi (θ) dan kecepatan maju (v) harus diatur sesuai dengan:

$$[\theta = \arctan\left(\frac{v_y}{v_x}\right)]$$

$$[v = \frac{v_x}{\cos \theta}]$$

c. Kinematika Robot Holonomik

Robot holonomik memiliki kemampuan untuk bergerak ke segala arah tanpa perlu mengubah orientasi terlebih dahulu. Ini dicapai dengan menggunakan roda khusus seperti omniwheel, mecanum wheel, atau sistem swerve drive.

➤ Robot Omniwheel

Robot omniwheel menggunakan roda omni yang memungkinkan gerakan lateral (samping) tanpa memutar roda.

Prinsip Kerja

- Setiap roda omni memiliki rol kecil yang berputar di sekitar sumbu utama roda, memungkinkan gerakan bebas ke segala arah.

- Robot omniwheel biasanya menggunakan tiga atau empat roda yang diatur dalam pola tertentu (misalnya, X-drive atau Y-drive).

Persamaan Kinematika

Untuk robot dengan empat roda omniwheel yang diatur dalam konfigurasi X-drive, kecepatan linier (v_x), (v_y), dan kecepatan angular (ω) dapat dihitung dengan:

$$[v_x = \frac{r}{4}(\omega_1 - \omega_2 - \omega_3 + \omega_4)]$$

$$[v_y = \frac{r}{4}(\omega_1 + \omega_2 + \omega_3 + \omega_4)]$$

$$[\omega = \frac{r}{4L}(\omega_1 + \omega_2 - \omega_3 - \omega_4)]$$

Di mana:

- ($\omega_1, \omega_2, \omega_3, \omega_4$) adalah kecepatan angular roda-roda omni.
- (L) adalah jarak antara pusat robot dan roda.

➤ **Robot Mecanum**

Robot mecanum menggunakan roda mecanum, yang terdiri dari roller-roller kecil yang dipasang dengan sudut tertentu terhadap sumbu roda. Ini memungkinkan gerakan omni-directional seperti pada robot omniwheel.

Prinsip Kerja

- Roda mecanum memiliki roller yang dipasang secara diagonal pada permukaan roda. Dengan mengontrol kecepatan dan arah rotasi setiap roda, robot dapat bergerak ke segala arah.

Persamaan Kinematika

Persamaan kinematika untuk robot mecanum mirip dengan robot omniwheel, tetapi dengan penyesuaian pada sudut roller:

$$[v_x = \frac{r}{4}(\omega_1 + \omega_2 + \omega_3 + \omega_4)]$$

$$[v_y = \frac{r}{4}(\omega_1 - \omega_2 + \omega_3 - \omega_4)]$$

$$[\omega = \frac{r}{4L}(\omega_1 - \omega_2 - \omega_3 + \omega_4)]$$

➤ Robot Swerve Drive

Robot swerve drive menggunakan modul swerve yang memungkinkan setiap roda berputar secara independen untuk mengubah arah. Ini memberikan robot kemampuan holonomik dengan kontrol yang sangat fleksibel.

Prinsip Kerja

- Setiap modul swerve memiliki dua aktuator: satu untuk mengendalikan kecepatan putar roda dan satu lagi untuk mengendalikan arah roda.
- Dengan mengatur kedua aktuator, robot dapat bergerak ke segala arah dengan kecepatan dan orientasi yang diinginkan.

Persamaan Kinematika

Persamaan kinematika untuk robot swerve drive lebih kompleks karena melibatkan koordinasi antara sudut dan kecepatan setiap roda. Setiap roda memiliki sudut kemudi (θ_i) dan kecepatan (v_i).

$$[v_x = \sum_{i=1}^4 v_i \cos(\theta_i)]$$

$$[v_y = \sum_{i=1}^4 v_i \sin(\theta_i)]$$

$$[\omega = \frac{1}{L} \sum_{i=1}^4 v_i (\cos(\theta_i - \alpha_i) \sin(\theta_i) - \sin(\theta_i - \alpha_i) \cos(\theta_i))]$$

Di mana:

- (α_i) adalah sudut antara roda dan pusat robot.

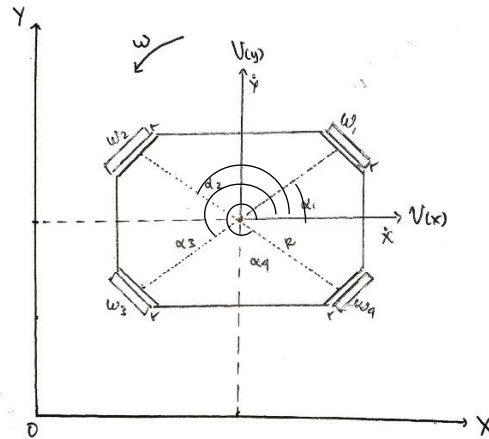
Aplikasi Kinematika Robot Beroda

- Navigasi Otonom : Kinematika digunakan untuk menentukan jalur dan kecepatan yang optimal bagi robot agar dapat bergerak dari titik awal ke titik tujuan dengan aman.
- Kontrol Robot: Dalam kontrol robot, kinematika digunakan untuk menentukan sinyal kontrol yang harus dikirim ke motor penggerak agar robot bergerak sesuai dengan yang diinginkan.
- Simulasi dan Pemodelan: Kinematika juga digunakan dalam simulasi dan pemodelan robot untuk memprediksi perilaku robot sebelum diterapkan dalam dunia nyata.

B. Persamaan Kinematika Menggunakan Konsep Fisika Sederhana

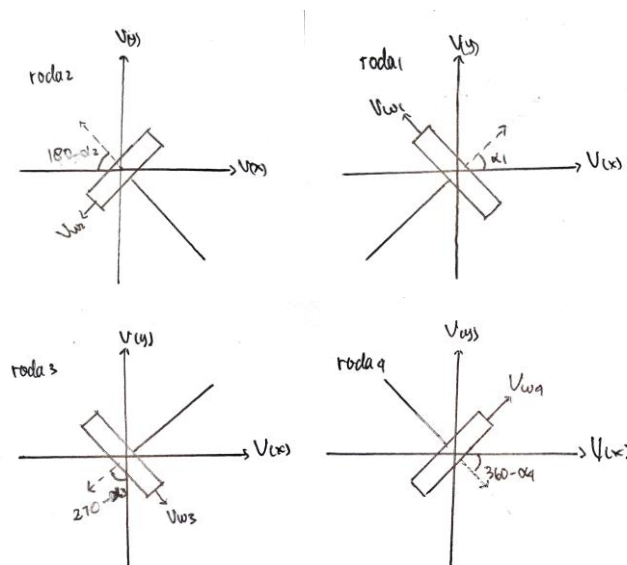
Gambar dibawah mengilustrasikan tampilan diagram bingkai pemosisian roda robot *omnidirectionall* X-Drive. Dengan menggunakan rangka robot sebagai kerangka acuan inersial sistem, maka diperoleh nilai sudut antara roda dan titik referensi robot

(titik tengah) yang diukur dari sumbu x positif dan dilambangkan dengan $\alpha_1, \alpha_2, \alpha_3$, dan α_4 dalam diagram ini. Kecepatan sudut dari setiap roda dilambangkan dengan $\omega = [\omega_1 \ \omega_2 \ \omega_3 \ \omega_4]^T$ dan kecepatan linear setiap roda adalah $v_w = [v_1 \ v_2 \ v_3 \ v_4]^T$. Nilai kecepatan roda ini akan menjadi positif ketika roda berputar berlawanan arah jarum jam (CCW), dan akan menjadi negatif ketika searah jarum jam (CW). Kecepatan linear dan kecepatan sudut robot terhadap kerangka acuan dilambangkan dengan menggunakan $v_r = [v_x \ v_y \ \omega]^T$. R adalah jarak dari roda ke pusat rangka robot dan r adalah jari-jari roda omni.



Penempatan roda dan vektor gerak robot X-Drive

Dengan melakukan peninjauan terhadap setiap rodanya, maka bisa didapatkan diagram vektor kecepatan untuk setiap rodanya seperti yang ditunjukkan oleh gambar dibawah. Nilai dari $\alpha_1, \alpha_2, \alpha_3$, dan α_4 , diperoleh yakni dengan melakukan perhitungan



Vektor kecepatan setiap roda

trigonometri terhadap posisi peletakan roda terhadap titik tengah robot menggunakan parameter panjang dan lebar. Dari gambar tersebut maka dapat diperoleh nilai masing-masing sudut yaitu :

Perhitungan Kinematika Balik

Kinematika balik adalah penggunaan persamaan kinematika untuk menentukan perencanaan gerakan. Dalam hal ini, persamaan kinematika mendefinisikan beberapa bagian dari gerakan robot yaitu gerakan linear dan melingkar dengan kecepatan v_x , v_y , dan ω dan diubah menjadi kecepatan untuk setiap roda nya $v_w = [v_1 \ v_2 \ v_3 \ v_4]^T$. Nilai kecepatan linear setiap roda v_w dapat ditentukan berdasarkan perubahan pergerakan linier robot terhadap koordinat acuan yang dilambangkan dengan simbol v_r . Dengan mengacu pada kedua gambar diatas, maka diperoleh persamaan kinematika balik untuk setiap rodanya sebagai berikut:

a. Roda 1

$$v_{w1} = -v_x \sin \alpha_1 + v_y \cos \alpha_1$$

b. Roda 2

$$v_{w2} = -v_x \sin (180 - \alpha_2) - v_y \cos(180 - \alpha_2)$$

$$v_{w2} = -v_x \sin \alpha_2 + v_y \cos \alpha_2$$

c. Roda 3

$$v_{w3} = v_x \cos (270 - \alpha_2) + v_y \sin(270 - \alpha_2)$$

$$v_{w3} = -v_x \sin \alpha_3 + v_y \cos \alpha_3$$

d. Roda 4

$$v_{w4} = v_x \sin (360 - \alpha_2) + v_y \cos(360 - \alpha_2)$$

$$v_{w4} = -v_x \sin \alpha_4 + v_y \cos \alpha_4$$

Dari persamaan diatas untuk kecepatan setiap roda yang diperoleh, maka didapatkan matriks kinematika balik dengan nilai berikut ini :

$$v_w = \begin{bmatrix} v_{w1} \\ v_{w2} \\ v_{w3} \\ v_{w4} \end{bmatrix} = T(\alpha) v_g \quad (\text{Note : } v = \omega * R)$$

Dengan matriks $T(\alpha)$ adalah persamaan matriks rotasi untuk setiap roda pada robot yang bisa ditulis dengan persamaan berikut :

$$T(\alpha) = \begin{bmatrix} -\sin(\alpha_1) & \cos(\alpha_1) & R \\ -\sin(\alpha_2) & \cos(\alpha_1) & R \\ -\sin(\alpha_3) & \cos(\alpha_1) & R \\ -\sin(\alpha_4) & \cos(\alpha_1) & R \end{bmatrix}$$

Perhitungan Kinematika Maju

Pergerakan linear robot v_g ditentukan berdasarkan nilai kecepatan linear setiap roda v_w . Mengacu pada persamaan nilai v_w , nilai v_g dapat diperoleh dengan membalik persamaan kinematika balik menjadi $v_g = v_w \cdot T(\theta)^{-1}$. Namun, karena matriks $T(\alpha)$ pada persamaan tersebut merupakan matriks 4×3 atau matriks non-persegi, maka $T(\alpha)^{-1}$ merupakan *pseudo-invers* atau inverse semu. Kesalahan perhitungan akan

terjadi jika rumus ini digunakan dan *inverse matriks* dipaksa dilakukan. Oleh karena itu, sebuah persamaan baru diusulkan seperti yang terlihat pada persamaan berikut :

$$v_g = v_{w1} + v_{w2} + v_{w3} + v_{w4}$$

a. Roda 1

$$v_{x1} = -v_{w1} \sin(\alpha_1)$$

$$v_{y1} = v_{w1} \cos(\alpha_1)$$

b. Roda 2

$$v_{x2} = -v_{w1} \sin(180 - \alpha_2)$$

$$v_{y2} = -v_{w1} \cos(180 - \alpha_2)$$

c. Roda 3

$$v_{x3} = v_{w3} \cos(270 - \alpha_3)$$

$$v_{y3} = -v_{w3} \sin(270 - \alpha_3)$$

d. Roda 4

$$v_{x4} = v_{w4} \sin(360 - \alpha_4)$$

$$v_{y4} = v_{w4} \cos(360 - \alpha_4)$$

Dari persamaan diatas, maka dapat diperoleh persamaan kinematika maju dengan memasukkannya ke dalam persamaan untuk nilai V_x , V_y dan ω untuk memperoleh kecepatan linear robot terhadap sumbu x dan y serta kecepatan sudut robot yakni :

$$v_x = (-v_{w1} \sin(\alpha_1) - v_{w2} \sin(\alpha_2) - v_{w3} \sin(\alpha_3) - v_{w4} \sin(\alpha_4))/2$$

$$v_y = (v_{w1} \cos(\alpha_1) + v_{w2} \cos(\alpha_2) + v_{w3} \cos(\alpha_3) + v_{w4} \cos(\alpha_4))/2$$

$$\omega = (\omega_1 + \omega_2 + \omega_3 + \omega_4)/4 \Rightarrow \omega = v/R$$

Yang dapat dituliskan dalam matriks menjadi :

$$v_g = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} v_{w1} \\ v_{w2} \\ v_{w3} \\ v_{w4} \end{bmatrix} D(\alpha)$$

Dengan matriks $D(\alpha)$ adalah matriks rotasi yang bisa ditulis sebagai :

$$D(\alpha) = \frac{1}{2} \begin{bmatrix} -\sin(\alpha_1) & \cos(\alpha_1) & 2R^{-1} \\ -\sin(\alpha_2) & \cos(\alpha_1) & 2R^{-1} \\ -\sin(\alpha_3) & \cos(\alpha_1) & 2R^{-1} \\ -\sin(\alpha_4) & \cos(\alpha_1) & 2R^{-1} \end{bmatrix}$$

BAB VII

MEKANIKA BENDA

A. Dasar Mekanika

Mekanika dalam robotika berkaitan dengan perancangan, analisis, dan implementasi komponen fisik dari robot, yang meliputi struktur, sambungan, aktuator, dan transmisi daya. Tujuan utama mekanika robotika adalah untuk menciptakan sistem yang efisien, tangguh, dan berfungsi secara optimal dalam menjalankan tugasnya.

Dalam konteks ini, beberapa aspek utama yang perlu dipertimbangkan adalah:

- **Desain Struktur:** Membuat robot yang kuat namun ringan.
- **Gerakan Robot:** Memastikan pergerakan robot sesuai dengan kebutuhan aplikasi, apakah itu gerakan translasi, rotasi, atau kombinasi keduanya.
- **Aktuator dan Transmisi Daya:** Bagaimana tenaga dari sumber daya (motor atau aktuator) dapat diterjemahkan menjadi gerakan yang diinginkan.

Dalam desain mekanik robot, terdapat beberapa parameter penting yang harus diperhatikan:

- **Berat:** Harus dijaga agar tidak terlalu berat karena akan mempengaruhi daya aktuator dan konsumsi energi.
- **Kekuatan:** Komponen harus cukup kuat untuk menahan gaya eksternal atau beban yang ditanggung robot.
- **Stabilitas:** Robot harus memiliki pusat massa yang seimbang agar tidak mudah jatuh (terutama untuk robot humanoid atau berkaki).
- **Efisiensi Energi:** Desain mekanik harus meminimalkan energi yang dibutuhkan untuk bergerak, khususnya pada robot mobile.

B. Gaya, Momen, dan Torsi

Mekanika merupakan cabang ilmu fisika yang mempelajari gerak serta gaya yang bekerja pada suatu benda. Dalam konteks robotika, pemahaman tentang gaya, momen, dan torsi sangat penting untuk mendesain sistem robot yang dapat bergerak, stabil, dan tahan terhadap beban kerja, pada bab ini, focus pembahasan adalah pada bagian Dinamika Benda (Analisa pergerakan benda dengan memperhitungkan sebab terjadinya pergerakan tersebut (Gaya)). Berikut adalah penjelasan dasar dari konsep-konsep tersebut:

a. Gaya (Force)

Gaya adalah dorongan atau tarikan yang dapat menyebabkan benda bergerak, berubah bentuk, atau berubah arah. Gaya diukur dalam satuan Newton (N), dan menurut Hukum Newton, gaya adalah hasil dari massa benda dan percepatan yang dialaminya.

$$[F = m \times a]$$

Di mana:

- (F) adalah gaya (N),
- (m) adalah massa benda (kg),
- (a) adalah percepatan (m/s²).

Jenis-jenis Gaya:

- Gaya Gravitasi: Gaya yang menarik benda ke bawah menuju pusat Bumi.
- Gaya Normal: Gaya yang diberikan oleh permukaan terhadap suatu objek yang ada di atasnya.
- Gaya Gesek: Gaya yang menahan gerakan benda ketika bersentuhan dengan permukaan lain.
- Gaya Tahanan Udara: Gaya yang bekerja berlawanan dengan arah gerak benda di udara.

Aplikasi Gaya pada Robotika:

Dalam robotika, gaya digunakan untuk:

- Mendorong atau menarik beban.
- Menggerakkan aktuator seperti motor atau silinder hidrolik.
- Mengukur tekanan atau gaya yang diterima oleh sensor gaya.

b. Momen (Moment)

Momen adalah kecenderungan suatu gaya untuk menyebabkan rotasi atau putaran pada suatu titik atau sumbu. Dalam kata lain, momen adalah efek rotasi yang dihasilkan oleh suatu gaya pada jarak tertentu dari titik poros.

Rumus momen (kadang disebut juga momen gaya atau torsi):

$$[M = F \times d]$$

Di mana:

- (M) adalah momen (Nm),
- (F) adalah gaya (N),
- (d) adalah jarak tegak lurus dari titik poros ke arah gaya (m).

Aplikasi Momen pada Robotika:

- Lengan Robot: Saat merancang lengan robot, momen digunakan untuk menghitung gaya yang diperlukan untuk menggerakkan segmen lengan pada sumbu tertentu.
- Stabilitas: Momen juga berperan dalam menjaga keseimbangan robot, misalnya dalam sistem kaki robot atau robot humanoid, di mana gaya gravitasi harus diimbangi oleh momen gaya pada sendi robot untuk mencegah robot jatuh.

c. Torsi (Torque)

Torsi adalah jenis khusus dari momen yang menyebabkan benda berputar pada porosnya. Torsi sangat penting dalam sistem robotik, terutama yang menggunakan motor dan aktuator untuk menggerakkan roda, lengan, atau komponen lainnya.

Rumus torsi mirip dengan momen:

$$[\tau = F \times r]$$

Di mana:

- (τ) adalah torsi (Nm),
- (F) adalah gaya yang bekerja pada benda (N),
- (r) adalah lengan momen atau jarak dari poros rotasi ke titik di mana gaya bekerja (m).

Hubungan Torsi dan Rotasi:

Torsi menyebabkan percepatan sudut pada benda yang berputar. Menurut hukum kedua Newton dalam rotasi:

$$[\tau = I \times \alpha]$$

Di mana:

- (I) adalah momen inersia benda ($\text{kg} \cdot \text{m}^2$),
- (α) adalah percepatan sudut (rad/s^2).

Aplikasi Torsi pada Robotika:

- Motor Penggerak: Setiap motor menghasilkan torsi untuk menggerakkan roda atau lengan robot. Jumlah torsi yang dibutuhkan tergantung pada berat beban yang digerakkan dan jarak dari poros ke beban.
- Aktuator Lengan Robot: Aktuator yang dipasang di sendi lengan robot harus menghasilkan torsi yang cukup untuk mengangkat atau memutar beban di ujung lengan.
- Sistem Kemudi: Pada robot beroda, torsi yang dihasilkan oleh motor roda menentukan seberapa cepat roda dapat berputar dan menggerakkan robot.

C. Hukum-hukum Newton dalam Dinamika

Dinamika robot adalah studi tentang bagaimana gaya dan momen yang bekerja pada robot mempengaruhi gerakan serta interaksi mekanis antara komponen-komponennya. Dinamika robot sangat penting untuk memastikan robot bergerak secara stabil, efisien, dan sesuai dengan tujuan perancangan. Dinamika robot mencakup dua aspek penting:

1. **Forward Dynamics:** Digunakan untuk menghitung bagaimana gaya dan torsi tertentu mempengaruhi gerakan robot. Ini digunakan dalam simulasi untuk memprediksi perilaku robot.
2. **Inverse Dynamics:** Digunakan untuk menghitung gaya atau torsi yang diperlukan pada setiap sendi atau aktuator untuk menghasilkan gerakan yang diinginkan. Ini banyak digunakan dalam kontrol robot, terutama untuk menghasilkan gerakan yang presisi.

a. Hukum Pertama Newton (Hukum Inersia)

Benda akan tetap dalam keadaan diam atau bergerak lurus beraturan kecuali jika ada gaya luar yang bekerja padanya. Ini berarti jika robot sedang bergerak dalam garis lurus,

ia akan terus bergerak dengan kecepatan yang sama kecuali ada gaya yang mengubah keadaan itu (misalnya gaya gesek atau gaya lain yang diterapkan).

$$[\sum F = 0 \quad (\text{jika benda dalam keadaan diam atau bergerak lurus beraturan})]$$

b. Hukum Kedua Newton ($F = ma$)

Percepatan suatu benda berbanding lurus dengan gaya yang bekerja padanya dan berbanding terbalik dengan massanya. Ini adalah hukum yang paling umum digunakan dalam analisis dinamika robot.

$$[\sum F = m \times a]$$

Dalam robotika, ini digunakan untuk menghitung gaya yang dibutuhkan untuk menghasilkan percepatan tertentu pada robot.

c. Hukum Ketiga Newton (Aksi = Reaksi)

Untuk setiap aksi, ada reaksi yang sama besar tetapi berlawanan arah. Misalnya, ketika roda robot mendorong tanah dengan gaya tertentu, tanah juga memberikan gaya reaksi yang sama besar kepada roda yang memungkinkan robot bergerak.

d. Momen Inersia (Moment of Inertia)

Momen inersia adalah ukuran dari "kelembaman" rotasi suatu benda, atau seberapa sulit benda untuk diputar. Momen inersia tergantung pada distribusi massa benda relatif terhadap poros rotasi. Rumus momen inersia untuk benda dengan bentuk sederhana adalah:

$$[I = \sum m_i \times r_i^2]$$

Di mana:

- (I) adalah momen inersia ($\text{kg} \cdot \text{m}^2$),
- (m_i) adalah massa elemen benda,
- (r_i) adalah jarak dari elemen massa ke poros rotasi.

Aplikasi Momen Inersia pada Robotika:

- Desain Lengan Robot: Pada robot dengan lengan bersegmen, momen inersia digunakan untuk menentukan seberapa besar torsi yang dibutuhkan untuk memutar lengan dengan beban pada ujungnya.
- Penggerak Roda: Pada robot beroda, momen inersia roda mempengaruhi seberapa cepat roda bisa diakselerasi atau dihentikan.

e. Energi Kinetik dan Energi Potensial

Dalam sistem mekanika robotik, dua jenis energi utama yang penting adalah energi kinetik dan energi potensial.

- **Energi Kinetik (Energi Gerak)**

Energi kinetik adalah energi yang dimiliki oleh benda yang sedang bergerak. Energi ini tergantung pada massa dan kecepatan benda.

$$[E_k = \frac{1}{2}mv^2]$$

Di mana:

- (E_k) adalah energi kinetik (Joule),
- (m) adalah massa benda (kg),
- (v) adalah kecepatan benda (m/s).

- Energi Potensial (Energi Tersimpan)

Energi potensial adalah energi yang dimiliki oleh suatu benda karena posisinya dalam suatu medan gaya, seperti medan gravitasi. Misalnya, benda yang berada di ketinggian tertentu memiliki energi potensial gravitasi.

$$[E_p = m \times g \times h]$$

Di mana:

- (E_p) adalah energi potensial (Joule),
- (m) adalah massa benda (kg),
- (g) adalah percepatan gravitasi (m/s^2),
- (h) adalah ketinggian benda di atas titik referensi (m).

Aplikasi dalam Robotika:

Energi kinetik dan energi potensial sangat penting dalam merancang robot yang bergerak, terutama untuk menghitung energi yang diperlukan untuk bergerak dari satu titik ke titik lain atau untuk mengangkat beban.

f. Lagrangian

Lagrangian adalah fungsi yang menggambarkan keadaan sistem dinamis dalam bentuk koordinat posisi dan turunan waktunya dan sama dengan perbedaan antara energi potensial dan energi kinetik dibandingkan hamiltonian. Metode Lagrangian lebih umum digunakan dalam dinamika robot karena lebih mudah diterapkan untuk sistem multi-derajat kebebasan (degrees of freedom - DOF). Metode ini didasarkan pada energi kinetik (T) dan energi potensial (V) robot. **Lagrangian (L)**: Didefinisikan sebagai perbedaan antara energi kinetik dan energi potensial:

$$L=T-V$$

Persamaan Euler-Lagrange: Persamaan ini digunakan untuk mendapatkan persamaan gerak robot:

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = \tau_i$$

Di mana:

- q_i adalah generalized coordinate (biasanya posisi sendi).
- τ_i adalah gaya atau torsi yang bekerja pada sendi tersebut.

Metode ini memungkinkan kita menghitung gaya dan torsi yang diperlukan untuk setiap gerakan robotik.

g. Analisis Statik dan Dinamik

- Analisis Statik: Digunakan untuk menghitung gaya dan momen yang bekerja pada robot atau komponennya dalam keadaan diam. Ini penting untuk memastikan bahwa komponen robot kuat dan tidak akan gagal di bawah beban statis.
- Analisis Dinamik: Digunakan untuk menghitung gaya, torsi, dan energi yang diperlukan saat robot bergerak atau berinteraksi dengan lingkungannya.

Contoh Aplikasi: Lengan Robot => Dalam analisis statik, Anda menghitung gaya dan momen pada lengan robot saat berada dalam posisi diam memegang beban. Dalam analisis dinamik, Anda memperhitungkan torsi dan gaya yang diperlukan untuk menggerakkan lengan robot sambil membawa beban.

D. Dinamika Forward dan Inverse

Dinamika Forward (Forward Dynamics)

Dalam dinamika forward, kita diberikan gaya atau torsi yang bekerja pada robot dan tujuan kita adalah menghitung gerakannya (kecepatan dan percepatan). Forward dynamics digunakan dalam simulasi fisik untuk memprediksi perilaku robot dalam lingkungan tertentu.

Langkah-langkah Dasar Forward Dynamics:

1. Tentukan gaya atau torsi yang bekerja pada setiap sendi atau aktuator.
2. Gunakan persamaan gerak (misalnya persamaan Lagrange atau hukum Newton) untuk menghitung percepatan.
3. Integrasikan percepatan untuk mendapatkan kecepatan dan posisi.

Dinamika Inverse (Inverse Dynamics)

Dinamika inverse lebih sering digunakan dalam kontrol robot, terutama untuk menghasilkan gerakan yang presisi. Dalam dinamika inverse, kita diberikan gerakan (trajektori) yang diinginkan dan tujuan kita adalah menghitung gaya atau torsi yang diperlukan untuk mencapai gerakan tersebut.

Langkah-langkah Dasar Inverse Dynamics:

1. Tentukan gerakan yang diinginkan (posisi, kecepatan, dan percepatan).
2. Gunakan metode Lagrange atau persamaan gerak lainnya untuk menghitung gaya atau torsi yang diperlukan.
3. Terapkan gaya atau torsi tersebut ke motor atau aktuator robot.

BAB VIII

ELEMEN MEKANIK ROBOT

A. Struktur dan Rangka Robot (Robot Frame)

- **Definisi dan Peran Rangka Robot**

Rangka atau **robot frame** adalah struktur mekanik utama yang mendukung dan menahan semua komponen robot. Ini meliputi motor, sensor, penggerak, baterai, dan bagian lain dari robot. Rangka robot berfungsi untuk memberikan stabilitas, integritas mekanik, dan tempat untuk menghubungkan semua bagian robot.

- **Kriteria Utama dalam Desain Rangka Robot**

- **Kekuatan:** Rangka harus mampu menahan beban robot, termasuk beban statis (seperti komponen elektronik) dan beban dinamis (seperti gaya yang dihasilkan oleh aktuator).
- **Berat:** Rangka harus dirancang se-ringan mungkin untuk memaksimalkan efisiensi energi, terutama pada robot mobile yang memiliki sumber daya terbatas.
- **Kekakuan:** Struktur rangka harus cukup kaku untuk menghindari deformasi yang signifikan saat robot bergerak atau saat ada beban eksternal.
- **Material:** Pemilihan material yang tepat (aluminium, baja, serat karbon, plastik) sangat penting untuk mendapatkan keseimbangan antara kekuatan dan berat.

B. Material Kerangka Robot

Pemilihan material merupakan salah satu aspek penting dalam desain rangka robot. Berikut adalah beberapa material yang umum digunakan:

a. Aluminium

- **Kelebihan:** Ringan, kuat, mudah dibentuk, dan memiliki ketahanan korosi yang baik.
- **Kekurangan:** Cenderung lebih mahal dibandingkan baja.
- **Aplikasi:** Banyak digunakan dalam robot industri dan mobile karena kekuatannya yang tinggi dan bobot yang ringan.

b. Baja (Steel)

- **Kelebihan:** Sangat kuat, kokoh, dan relatif murah.
- **Kekurangan:** Berat, sehingga tidak ideal untuk robot mobile yang perlu efisiensi energi tinggi.
- **Aplikasi:** Digunakan dalam robot yang memerlukan kekuatan tinggi, seperti robot pengangkut atau robot yang bekerja di lingkungan industri berat.

c. Serat Karbon (Carbon Fiber)

- **Kelebihan:** Sangat ringan dan kuat, memiliki rasio kekuatan terhadap berat yang sangat tinggi.
- **Kekurangan:** Sangat mahal dan sulit untuk dibentuk.

- **Aplikasi:** Banyak digunakan dalam robot berkinerja tinggi atau aplikasi luar angkasa di mana berat sangat penting.

d. Plastik

- **Kelebihan:** Sangat ringan dan murah, mudah dicetak dengan teknologi seperti 3D printing.
- **Kekurangan:** Kurang kuat dibandingkan logam, tidak cocok untuk beban berat.
- **Aplikasi:** Sering digunakan dalam robot prototipe atau robot yang beroperasi di lingkungan ringan.

e. Polimer

- **Akrilik dan ABS:** Ringan, murah, dan cukup kuat untuk aplikasi ringan.
- **Nylon:** Fleksibel dan tahan abrasi, cocok untuk bagian yang memerlukan ketahanan terhadap gesekan.

C. Material Bahan 3D Printing

3D printing adalah metode yang populer dalam pembuatan prototipe atau komponen robot karena memungkinkan pencetakan bagian yang kompleks dengan biaya rendah. Ada beberapa jenis material yang digunakan dalam 3D printing untuk robotika:

a. PLA (Polylactic Acid)

- **Kelebihan:**
 - Mudah digunakan.
 - Ramah lingkungan, karena berbasis bahan alami.
 - Cocok untuk bagian prototipe atau bagian yang tidak memerlukan kekuatan tinggi.
- **Kekurangan:**
 - Rapuh dan cenderung pecah di bawah tekanan.
 - Tidak tahan suhu tinggi (melting point sekitar 60°C).

b. ABS (Acrylonitrile Butadiene Styrene)

- **Kelebihan:**
 - Lebih kuat dan tahan panas dibanding PLA.
 - Lebih tahan terhadap benturan.
 - Bagus untuk bagian mekanis yang memerlukan ketahanan lebih baik.
- **Kekurangan:**
 - Lebih sulit dicetak karena membutuhkan alas yang dipanaskan.
 - Berpotensi mengeluarkan asap beracun selama pencetakan.

c. PETG (Polyethylene Terephthalate Glycol)

- **Kelebihan:**
 - Kombinasi kekuatan dan fleksibilitas yang baik.
 - Lebih tahan terhadap air dan kimia.
 - Cocok untuk komponen struktural dan casing yang membutuhkan kekuatan menengah.
- **Kekurangan:**
 - Lebih sulit dicetak dengan presisi dibandingkan PLA.

d. Nylon

- **Kelebihan:**
 - Sangat kuat dan tahan gesekan.
 - Fleksibel, cocok untuk bagian bergerak atau engsel.
- **Kekurangan:**
 - Rentan terhadap kelembaban dan membutuhkan penyimpanan khusus agar tidak menyerap air.

D. Kekuatan Bahan

Analisis kekuatan bahan melibatkan pemahaman terhadap sifat mekanik bahan dan bagaimana bahan tersebut akan merespons berbagai jenis beban. Beberapa parameter penting dalam analisis kekuatan bahan adalah:

a. Kekuatan Tarik (Tensile Strength)

Ini adalah kemampuan material untuk menahan gaya tarik sebelum mengalami kerusakan. Material yang memiliki kekuatan tarik tinggi cocok untuk aplikasi di mana material akan ditarik atau ditegangkan.

- **Aluminium:** Kekuatan tarik sedang, cocok untuk struktur ringan.
- **Baja:** Kekuatan tarik sangat tinggi, cocok untuk aplikasi yang memerlukan daya dukung berat.
- **PLA:** Kekuatan tarik rendah, cocok untuk bagian yang tidak memerlukan kekuatan tinggi.

b. Modulus Young (Young's Modulus)

Modulus Young mengukur kekakuan material, yaitu kemampuan material untuk mengalami deformasi elastis (bisa kembali ke bentuk semula setelah beban dihilangkan).

- **Serat Karbon:** Memiliki modulus Young tinggi, sehingga sangat kaku.
- **Nylon:** Modulus Young rendah, lebih fleksibel.

c. Kekuatan Tekan (Compressive Strength)

Ini adalah kemampuan material untuk menahan gaya tekan. Material dengan kekuatan tekan tinggi tidak mudah rusak saat ditekan.

- **Aluminium dan Baja:** Keduanya memiliki kekuatan tekan tinggi.
- **PLA dan ABS:** Kekuatan tekan lebih rendah, sehingga lebih cocok untuk prototipe.

d. Tahan Gesekan (Abrasion Resistance)

Material dengan tahan gesekan yang baik diperlukan untuk bagian-bagian yang terus-menerus bergesekan, seperti roda atau bagian penggerak.

- **Nylon:** Sangat baik dalam menahan gesekan.
- **Aluminium:** Rentan terhadap aus akibat gesekan.

e. Fatigue Strength (Kekuatan Lelah)

Fatigue strength adalah kemampuan material untuk menahan siklus berulang-ulang tanpa mengalami kegagalan.

- **Baja dan Serat Karbon:** Kekuatan lelah yang tinggi, cocok untuk bagian yang harus bertahan dari tekanan siklus yang berulang.
- **PLA:** Kekuatan lelah rendah, sehingga kurang cocok untuk bagian mekanis yang bergerak.

E. Mekanisme Transmisi Daya dalam Robotika

Mekanisme transmisi daya adalah bagian penting dari robot yang berfungsi untuk mentransfer daya dari motor atau aktuator ke bagian yang digerakkan, seperti roda, lengan, atau segmen lainnya. Desain yang baik dari mekanisme transmisi daya sangat penting untuk mencapai efisiensi, presisi, dan performa optimal dari robot.

➤ Fungsi Mekanisme Transmisi Daya

Mekanisme transmisi daya digunakan untuk:

- **Mengubah kecepatan:** Mempercepat atau memperlambat gerakan dari motor atau aktuator ke bagian yang digerakkan.
- **Mengubah torsi:** Meningkatkan atau menurunkan torsi (momen putar) sesuai kebutuhan.
- **Menyalurkan daya secara efisien:** Memastikan daya dari motor dapat ditransmisikan ke beban dengan kerugian energi minimal.
- **Mentransfer daya ke lokasi lain:** Memungkinkan motor atau sumber daya diposisikan jauh dari bagian yang digerakkan.

➤ Jenis Mekanisme Transmisi Daya

Terdapat beberapa jenis mekanisme transmisi daya yang umum digunakan dalam robotika. Setiap jenis memiliki karakteristik dan kelebihan masing-masing, tergantung pada aplikasi robot yang dirancang.

a. Roda Gigi (Gear System)

Roda gigi adalah salah satu mekanisme transmisi daya yang paling umum digunakan dalam robotika, terutama untuk mengubah kecepatan dan torsi. Sistem roda gigi memungkinkan pengurangan kecepatan dan peningkatan torsi atau sebaliknya, tergantung pada rasio gigi yang digunakan.

- **Gear Ratio (Rasio Gigi):** Rasio gigi didefinisikan sebagai perbandingan antara jumlah gigi pada roda gigi penggerak (driving gear) dan roda gigi yang digerakkan (driven gear). Jika roda gigi penggerak memiliki 10 gigi dan roda gigi yang digerakkan memiliki 20 gigi, maka rasio gigi adalah 1:2, artinya kecepatan putaran akan berkurang setengahnya, tetapi torsi akan meningkat dua kali lipat.
- **Fungsi Roda Gigi**
 - **Transmisi Daya:** Roda gigi mentransmisikan daya antara dua poros yang terhubung, sering kali dengan mengubah kecepatan dan torsi.

- **Perubahan Kecepatan:** Roda gigi dapat mempercepat atau memperlambat rotasi antara dua poros.
- **Perubahan Torsi:** Dengan merubah rasio roda gigi, torsi yang dihasilkan dapat ditingkatkan atau dikurangi.
- **Perubahan Arah:** Roda gigi dapat mengubah arah rotasi, baik itu secara langsung atau dengan bantuan roda gigi tambahan.
- **Jenis-jenis Roda Gigi**

Berikut adalah beberapa jenis roda gigi yang umum digunakan dalam berbagai aplikasi mekanik dan robotik:

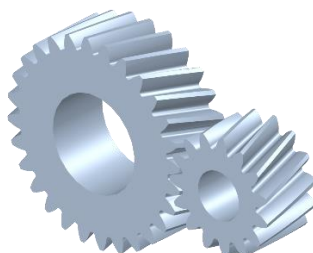
- Roda Gigi Lurus (Spur Gear)

- **Ciri-ciri:** Gigi pada roda ini sejajar dengan sumbu poros. Roda gigi lurus adalah jenis yang paling umum dan sederhana.
- **Kelebihan:** Efisien dalam mentransmisikan daya, desain sederhana, mudah diproduksi, dan cocok untuk kecepatan sedang hingga tinggi.
- **Kekurangan:** Berisik pada kecepatan tinggi karena tumbukan antar gigi.
- **Aplikasi:** Digunakan dalam mesin-mesin ringan, jam tangan, dan sistem robotik sederhana.



- Roda Gigi Heliks (Helical Gear)

- **Ciri-ciri:** Gigi-gigi pada roda gigi heliks miring terhadap sumbu rotasi. Ini menyebabkan kontak antara gigi berlangsung lebih halus dan bertahap.
- **Kelebihan:** Lebih halus dan lebih tenang dibanding roda gigi lurus, cocok untuk aplikasi dengan kecepatan tinggi.
- **Kekurangan:** Menyebabkan gaya aksial yang harus ditangani oleh bantalan.
- **Aplikasi:** Digunakan pada transmisi mobil, mesin industri, dan robot yang memerlukan transmisi daya halus.



- Roda Gigi Kerucut (Bevel Gear)

- **Ciri-ciri:** Bentuknya kerucut dan digunakan untuk mengubah arah gerakan rotasi dari satu poros ke poros lain dengan sudut 90 derajat.
- **Kelebihan:** Efisien dalam mengubah arah rotasi.
- **Kekurangan:** Lebih sulit diproduksi dan membutuhkan penyesuaian presisi.
- **Aplikasi:** Banyak digunakan pada diferensial mobil, bor sudut, dan sistem robot yang membutuhkan perubahan arah rotasi.



- Roda Gigi Cacing (Worm Gear)

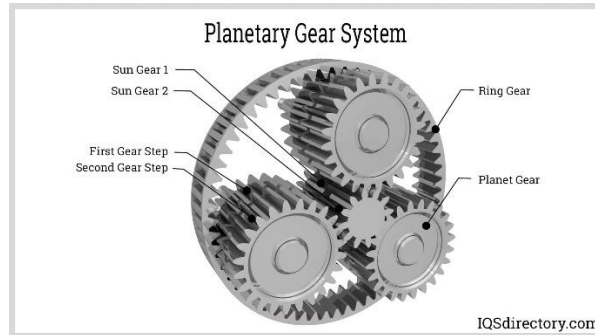
- **Ciri-ciri:** Terdiri dari sekrup (cacing) yang terhubung dengan roda gigi biasa. Sistem ini memungkinkan pengurangan kecepatan yang sangat besar dengan satu tahapan.
- **Kelebihan:** Rasio reduksi yang tinggi, gerakan yang sangat halus, dan self-locking (mencegah gerakan balik).
- **Kekurangan:** Tidak efisien dalam hal perpindahan daya karena adanya gesekan tinggi.
- **Aplikasi:** Digunakan pada sistem pengangkat, aktuator linier, dan aplikasi yang membutuhkan reduksi kecepatan tinggi.



- Roda Gigi Planetary (Planetary Gear)

- **Ciri-ciri:** Terdiri dari satu roda gigi tengah (sun gear), beberapa roda gigi kecil (planet gear), dan roda gigi cincin di bagian luar (ring gear). Sistem ini menawarkan rasio kecepatan yang tinggi dalam desain kompak.

- **Kelebihan:** Desain ringkas, distribusi beban merata, dan dapat memberikan rasio reduksi yang besar.
- **Kekurangan:** Kompleks dalam pembuatan dan perawatan.
- **Aplikasi:** Digunakan dalam transmisi otomatis, robot dengan torsi tinggi, dan aplikasi presisi tinggi.



- **Parameter-parameter dalam Roda Gigi**

- **Modul (Module, m)**

Modul adalah ukuran yang menggambarkan besar kecilnya gigi pada roda gigi. Modul ditentukan oleh diameter lingkaran pitch dibagi dengan jumlah gigi.

- Rumus:

$$[m = \frac{d}{z}]$$

Di mana:

- (m) adalah modul (mm),
- (d) adalah diameter pitch (mm),
- (z) adalah jumlah gigi.

- **Rasio Transmisi (Gear Ratio)**

Rasio transmisi adalah perbandingan antara jumlah gigi roda gigi yang terhubung. Rasio ini mempengaruhi kecepatan dan torsi yang dihasilkan.

- Rumus:

$$[\text{Gear Ratio} = \frac{z_2}{z_1}]$$

Di mana:

- (z_1) adalah jumlah gigi roda penggerak,
- (z_2) adalah jumlah gigi roda yang digerakkan.

- **Lingkaran Pitch (Pitch Circle Diameter, PCD)**

Lingkaran pitch adalah lingkaran imajiner di tengah gigi roda gigi di mana kontak antar gigi terjadi. Diameter pitch ini penting untuk menghitung kecepatan dan torsi yang dihasilkan.

- Tekanan Sudut (Pressure Angle)

Tekanan sudut adalah sudut antara garis normal pada titik kontak gigi dan garis tangensial pada lingkaran pitch. Tekanan sudut yang umum digunakan adalah 20 derajat, yang menghasilkan efisiensi dan daya tahan yang baik.

- Ketinggian Gigi (Addendum dan Dedendum)

- Addendum: Jarak dari lingkaran pitch ke puncak gigi.
- Dedendum: Jarak dari lingkaran pitch ke dasar gigi.

- Backlash

Backlash adalah celah kecil antara gigi-gigi roda gigi yang saling bersinggungan. Backlash diperlukan untuk menghindari terjadinya gesekan berlebihan atau kekakuan antara gigi yang bersentuhan, terutama ketika terjadi perubahan arah putaran.

- Perhitungan Torsi pada Roda Gigi

Torsi yang ditransmisikan oleh roda gigi tergantung pada rasio roda gigi dan ukuran gigi. Hubungan antara torsi dan rasio roda gigi diberikan oleh:

$$[T_2 = T_1 \times \frac{z_2}{z_1}]$$

Di mana:

- (T_1) adalah torsi pada roda penggerak (Nm),
- (T_2) adalah torsi pada roda yang digerakkan (Nm),
- (z_1) adalah jumlah gigi roda penggerak,
- (z_2) adalah jumlah gigi roda yang digerakkan.

Keuntungan:

- Efisiensi tinggi dalam transmisi daya.
- Presisi yang baik untuk kontrol gerakan.
- Beragam konfigurasi untuk berbagai aplikasi.

Kekurangan:

- Memerlukan pelumasan untuk mengurangi gesekan dan keausan.
- Pembuatan yang presisi bisa mahal.

b. Sabuk dan Katrol (Belt and Pulley)

Sistem **sabuk dan katrol** digunakan untuk transmisi daya antar poros yang terpisah. Sistem ini biasanya digunakan untuk aplikasi yang memerlukan transfer daya ke jarak yang lebih jauh dibandingkan roda gigi, dengan kecepatan tinggi namun torsi yang lebih rendah.

- **Sabuk (Belt):** Biasanya terbuat dari bahan karet atau poliuretan yang fleksibel namun kuat. Sabuk mentransmisikan daya dari satu katrol ke katrol lain.
- **Katrol (Pulley):** Katrol berfungsi sebagai roda yang memandu sabuk. Ukuran katrol menentukan rasio transmisi, mirip dengan rasio gigi pada roda gigi.

Jenis-jenis Sabuk dan Katrol:

1. **Flat Belts:** Sabuk datar digunakan untuk kecepatan tinggi dan torsi rendah. Sistem ini kurang efisien dibandingkan sabuk bergerigi karena bisa mengalami slip.
2. **V-Belts:** Sabuk berbentuk V yang lebih efisien daripada flat belts karena desainnya mencegah slip dengan lebih baik.
3. **Timing Belts:** Sabuk dengan gigi di permukaannya yang memastikan tidak terjadi slip dan sangat cocok untuk aplikasi yang memerlukan sinkronisasi yang tepat.

Keuntungan:

- Mampu menyalurkan daya ke jarak yang lebih jauh.
- Desain sederhana dan biaya lebih rendah dibandingkan roda gigi.
- Cocok untuk aplikasi berkecepatan tinggi.

Kekurangan:

- Cenderung slip jika sabuk tidak diatur dengan ketegangan yang tepat.
- Kurang presisi dibandingkan sistem roda gigi.

c. Rantai dan Sproket (Chain and Sprocket)

Sistem **rantai dan sproket** sangat mirip dengan sabuk dan katrol, tetapi menggunakan rantai logam yang kuat alih-alih sabuk fleksibel. Sistem ini biasanya digunakan dalam aplikasi yang membutuhkan torsi lebih besar dan transmisi daya yang lebih kuat.

- **Rantai (Chain):** Dibuat dari bahan logam untuk menahan beban yang lebih besar. Rantai terhubung dengan sproket di kedua ujungnya.
- **Sproket:** Roda bergerigi yang terhubung dengan rantai untuk mentransmisikan daya. Rasio sproket menentukan rasio transmisi seperti halnya rasio gigi pada roda gigi.

Keuntungan:

- Lebih kuat daripada sabuk dan katrol, cocok untuk aplikasi torsi tinggi.
- Tidak ada slip, sehingga lebih presisi.

Kekurangan:

- Membutuhkan pelumasan agar rantai bergerak lancar.
- Lebih bising dan lebih berat dibandingkan sistem sabuk.

d. Sistem Lead Screw dan Ball Screw

Sistem **lead screw** dan **ball screw** mengubah gerakan putar menjadi gerakan linear. Kedua sistem ini banyak digunakan dalam aplikasi robotika yang memerlukan gerakan presisi dalam sumbu linier, seperti robot CNC atau printer 3D.

- **Lead Screw:** Menggunakan ulir pada batang untuk menggerakkan mur (nut) yang dihubungkan dengan bagian yang digerakkan. Lead screw menghasilkan torsi tinggi, tetapi cenderung memiliki efisiensi yang lebih rendah dibandingkan ball screw.
- **Ball Screw:** Mirip dengan lead screw, tetapi menggunakan bola-bola baja yang bergulir di dalam ulir, mengurangi gesekan dan meningkatkan efisiensi. Ball screw biasanya digunakan dalam aplikasi yang memerlukan akurasi tinggi dan efisiensi energi, seperti mesin presisi.

Keuntungan:

- Sangat presisi untuk kontrol gerakan linier.
- Efisien dalam mentransmisikan daya.

Kekurangan:

- Lead screw cenderung memiliki gesekan yang lebih tinggi dibandingkan ball screw.
- Ball screw lebih mahal daripada lead screw.

e. Sistem Kopling (Coupling)

Kopling digunakan untuk menghubungkan dua poros yang berputar bersama-sama. Kopling berfungsi untuk mentransmisikan daya antara dua komponen yang tidak bisa dihubungkan secara langsung karena keterbatasan desain atau untuk melindungi komponen lain dari beban yang berlebihan.

Jenis-jenis Kopling:

1. **Rigid Couplings:** Menghubungkan dua poros dengan sangat erat, tanpa memberikan fleksibilitas. Digunakan ketika presisi tinggi diperlukan, tetapi berisiko jika ada pergeseran poros.
2. **Flexible Couplings:** Memberikan sedikit fleksibilitas, memungkinkan sedikit ketidaksejajaran antara poros yang terhubung. Ini dapat melindungi komponen dari kerusakan akibat ketidaksejajaran.

Keuntungan:

- Melindungi komponen dari ketegangan akibat ketidaksejajaran poros.
- Fleksibilitas dalam berbagai desain poros.

Kekurangan:

- Kopling kaku dapat menyebabkan kerusakan pada poros jika terjadi ketidaksejajaran.

e. **Faktor-faktor Penting dalam Pemilihan Mekanisme Transmisi Daya**

Saat memilih mekanisme transmisi daya untuk robot, beberapa faktor penting yang perlu dipertimbangkan adalah:

1. **Kebutuhan Torsi dan Kecepatan:** Setiap jenis transmisi daya memiliki kemampuan yang berbeda dalam hal mengubah torsi dan kecepatan. Untuk aplikasi yang memerlukan torsi tinggi, sistem roda gigi, rantai dan sproket, atau lead screw adalah pilihan yang baik. Untuk aplikasi kecepatan tinggi, sabuk dan katrol bisa lebih sesuai.
2. **Presisi:** Jika presisi menjadi prioritas, seperti dalam robotika industri, sistem roda gigi, timing belt, atau ball screw harus dipertimbangkan. Untuk aplikasi yang kurang memerlukan presisi tinggi, sabuk atau rantai dapat digunakan.
3. **Efisiensi:** Sistem transmisi daya yang lebih efisien akan mengurangi kerugian energi dan menghasilkan performa yang lebih baik secara keseluruhan. Ball screw dan roda gigi cenderung lebih efisien dibandingkan sistem sabuk dan katrol.
4. **Biaya dan Kompleksitas:** Sistem transmisi daya yang lebih presisi dan efisien, seperti ball screw atau bevel gears, biasanya lebih mahal dan membutuhkan perawatan yang lebih rumit dibandingkan sabuk atau katrol.

F. **Desain End-Effector**

End-effector adalah bagian dari robot yang berinteraksi langsung dengan objek. Pada lengan robot, end-effector seringkali dianggap sebagai "tangan" atau "alat" yang melakukan tugas spesifik, seperti menggenggam, memotong, mengelaspada, atau memanipulasi objek. Desain end-effector harus mempertimbangkan tujuan aplikasi dan interaksi dengan lingkungan.

a. **Jenis-Jenis End-Effector**

1. **Gripper:** Digunakan untuk memegang atau menggenggam objek. Terdapat beberapa tipe gripper:
 - **Parallel Gripper:** Menggunakan dua rahang yang bergerak secara paralel untuk menggenggam objek.
 - **Angled Gripper:** Rahang bergerak dengan sudut tertentu dan cocok untuk aplikasi presisi.
 - **Vacuum Gripper:** Menggunakan vakum untuk mengangkat objek, sering digunakan di industri elektronik atau pengemasan.
 - **Magnetic Gripper:** Menggunakan magnet untuk memanipulasi objek logam.
2. **Tooling End-Effector:** Digunakan untuk membawa alat seperti obeng otomatis, pengelasan, atau alat pemotong.

3. **Suction Cups:** Digunakan untuk memindahkan objek dengan permukaan halus atau datar dengan menciptakan tekanan negatif.
4. **Multifingered Hand:** End-effector dengan jari-jari yang lebih menyerupai tangan manusia, digunakan untuk manipulasi objek yang lebih kompleks dan fleksibel.

b. Desain Mekanik End-Effector

Beberapa faktor penting dalam desain mekanik end-effector:

- **Kapasitas beban (Payload):** Kemampuan end-effector untuk menahan beban tertentu tanpa kehilangan fungsinya.
- **Kekuatan genggaman:** Menentukan gaya yang dibutuhkan untuk memegang objek tanpa merusaknya.
- **Kestabilan:** End-effector harus memiliki pusat gravitasi yang sesuai agar stabil saat bergerak atau melakukan manipulasi.
- **Ketepatan posisi:** Penting dalam aplikasi presisi seperti assembling komponen elektronik.
- **Kemampuan adaptasi:** End-effector harus bisa menangani berbagai bentuk, ukuran, dan berat objek dengan cepat.

c. Material yang Digunakan

Material end-effector dipilih berdasarkan sifat objek yang akan dimanipulasi dan lingkungan kerja:

- **Aluminium:** Ringan, kokoh, dan umum digunakan untuk gripper.
- **Baja tahan karat:** Tahan terhadap korosi, cocok untuk aplikasi industri berat.
- **Nylon dan Polimer:** Cocok untuk bagian gripper yang perlu fleksibilitas atau tahan terhadap gesekan tinggi.

d. Konsiderasi Desain

1. **Ergonomi:** Desain harus mudah diintegrasikan dengan lengan robot dan nyaman untuk digunakan dalam aplikasi yang sering berubah.
2. **Kapasitas Sensorik:** Beberapa end-effector dilengkapi dengan sensor tekanan, gaya, atau kamera untuk membantu manipulasi objek dengan lebih cerdas.
3. **Kecepatan dan Respon:** Kecepatan pergerakan end-effector harus optimal agar dapat beroperasi dengan efisien.

G. Maintenance (Pemeliharaan) Sistem Mekanik

Pemeliharaan sistem mekanik robot bertujuan untuk menjaga kelancaran operasional dan memperpanjang umur sistem. Perawatan yang tepat juga membantu mengurangi waktu henti (downtime) dan mencegah kerusakan besar.

a. Jenis-Jenis Pemeliharaan

1. **Preventive Maintenance:** Melibatkan pemeriksaan rutin dan perawatan komponen mekanik secara berkala untuk mencegah masalah sebelum terjadi.

Contoh: pelumasan bantalan (bearing) secara teratur, pengecekan sekrup, dan pengencangan sabuk.

2. **Predictive Maintenance:** Menggunakan sensor dan analisis data untuk memprediksi kapan komponen mekanik akan rusak berdasarkan kinerja saat ini. Contoh: menggunakan sensor getaran untuk mendeteksi tanda-tanda keausan.
3. **Corrective Maintenance:** Dilakukan ketika kerusakan sudah terjadi. Ini termasuk penggantian komponen yang rusak dan perbaikan segera untuk mengembalikan sistem ke kondisi normal.

b. Prosedur Maintenance Sistem Mekanik

1. **Pelumasan:** Komponen bergerak seperti roda gigi, rantai, dan bantalan harus dilumasi secara berkala untuk mengurangi gesekan dan keausan. Jenis pelumas harus sesuai dengan kondisi lingkungan operasi (suhu, kelembaban).
2. **Kalibrasi Ulang:** Beberapa sistem mekanik memerlukan kalibrasi ulang untuk menjaga akurasi. Hal ini sangat penting dalam sistem robotik yang berurusan dengan pengukuran dan presisi tinggi.
3. **Penggantian Komponen:** Bagian yang aus atau mengalami kerusakan seperti sabuk penggerak, bantalan, atau roda gigi harus diganti sebelum mempengaruhi sistem secara keseluruhan.
4. **Pembersihan:** Debu dan kotoran bisa menumpuk pada komponen mekanik, terutama di lingkungan industri. Pembersihan berkala sangat penting untuk menjaga performa optimal.

c. Tips Pemeliharaan

- Buatlah jadwal pemeliharaan yang teratur berdasarkan tingkat penggunaan robot.
- Gunakan suku cadang asli atau sesuai spesifikasi pabrik untuk penggantian komponen.
- Dokumentasikan setiap perawatan yang dilakukan untuk referensi di masa mendatang.
- Lakukan inspeksi visual harian untuk memeriksa adanya kerusakan atau keausan yang dapat terlihat.

H. Troubleshooting Sistem Mekanik

Troubleshooting adalah proses identifikasi dan perbaikan masalah dalam sistem mekanik robot. Ini melibatkan analisis komponen dan subsistem untuk menemukan sumber masalah dan mengatasi kerusakan dengan efisien.

a. Proses Dasar Troubleshooting

1. **Identifikasi Masalah:** Langkah pertama adalah memahami gejala masalah yang muncul, seperti getaran yang berlebihan, suara tidak biasa, atau gerakan yang tidak normal.

2. **Analisis Penyebab:** Setelah gejala diidentifikasi, lakukan analisis penyebab masalah. Ini bisa melibatkan pemeriksaan mekanik seperti keausan roda gigi, atau komponen listrik yang tidak berfungsi.
3. **Pengujian dan Diagnosis:** Gunakan alat seperti osiloskop atau sensor getaran untuk memverifikasi kondisi mekanik dan menentukan akar masalah.
4. **Perbaikan:** Setelah penyebabnya ditemukan, perbaiki atau ganti komponen yang bermasalah.
5. **Pengujian Ulang:** Setelah perbaikan, uji kembali sistem untuk memastikan masalah telah terselesaikan.

b. Masalah Umum dalam Sistem Mekanik

1. **Overheating:** Sistem mekanik yang bekerja dalam waktu lama dapat menghasilkan panas berlebih. Overheating bisa merusak komponen seperti motor atau bantalan.
 - **Solusi:** Periksa sistem pendingin dan pastikan ventilasi bekerja dengan baik. Tambahkan pendingin atau kurangi beban kerja jika perlu.
2. **Suara Tidak Normal:** Suara mendesing, berderak, atau getaran biasanya merupakan tanda adanya masalah di komponen bergerak.
 - **Solusi:** Periksa komponen yang bergerak seperti roda gigi, rantai, dan sabuk. Pastikan semua bagian dipasang dengan benar dan tidak ada komponen yang aus atau longgar.
3. **Gerakan yang Tidak Akurat:** Jika robot tidak bergerak seperti yang diharapkan, bisa jadi ada masalah pada aktuator atau sensor posisi.
 - **Solusi:** Kalibrasi ulang sensor, periksa koneksi kabel, dan pastikan aktuator bekerja dengan benar.
4. **Kerusakan Bantalan (Bearing Failure):** Bantalan yang aus atau rusak dapat menyebabkan gesekan berlebih dan mempengaruhi kinerja mekanik.
 - **Solusi:** Ganti bantalan yang rusak dan pastikan pelumasan yang tepat.

c. Alat yang Digunakan untuk Troubleshooting

- **Sensor Getaran:** Untuk mendeteksi ketidakseimbangan atau keausan pada komponen bergerak.
- **Thermometer Inframerah:** Digunakan untuk mendeteksi panas berlebih pada motor atau bantalan.
- **Alat Pengukur Gaya:** Mengukur tekanan atau beban yang diterapkan pada end-effector atau bagian lain dari sistem robotik.

BAB IX

FILTER DATA DAN SISTEM KENDALI

A. Filter Data

Filter data adalah teknik dalam pengolahan sinyal yang digunakan untuk menghaluskan, membersihkan, atau mengurangi noise (gangguan) dari data yang diukur. Dalam robotika dan mekatronika, filter data penting untuk memperoleh sinyal yang lebih akurat dari sensor dan perangkat keras, terutama dalam kondisi yang rentan terhadap gangguan seperti getaran, interferensi, atau noise lingkungan.

a. Jenis-jenis Filter Data

Beberapa filter yang sering digunakan dalam pengolahan data sensor dan sinyal antara lain:

- Filter Low-Pass (LPF)
 - Fungsi: Mengizinkan sinyal dengan frekuensi rendah untuk lewat, sementara memblokir sinyal frekuensi tinggi (noise).
 - Aplikasi: Digunakan untuk mengurangi noise pada data sensor IMU, seperti accelerometer atau gyroscope, yang cenderung menghasilkan sinyal frekuensi tinggi.
 - Contoh: Membersihkan data sensor percepatan pada robot yang bergerak lambat agar tidak terganggu oleh getaran kecil.
 - Persamaan:

$$[y[n] = \alpha x[n] + (1 - \alpha)y[n - 1]]$$

Di mana:

- $(y[n])$ adalah output filter,
- $(x[n])$ adalah input sinyal pada waktu (n) ,
- (α) adalah faktor penghalusan (antara 0 dan 1, semakin kecil (α) , semakin lambat responnya).

- Filter High-Pass (HPF)
 - Fungsi: Mengizinkan sinyal dengan frekuensi tinggi untuk lewat, sementara memblokir sinyal frekuensi rendah.
 - Aplikasi: Berguna untuk mendeteksi perubahan mendadak pada data sensor atau gerakan yang cepat, sementara mengabaikan perubahan lambat atau noise frekuensi rendah.
 - Contoh: Mendeteksi perubahan cepat dalam posisi robot yang bergerak secara dinamis.
 - Persamaan:

$$[y[n] = \alpha(y[n - 1] + x[n] - x[n - 1])]$$

Di mana:

- $(x[n])$ adalah input sinyal,
- $(y[n])$ adalah output sinyal,
- (α) adalah konstanta filter (sering dipilih berdasarkan frekuensi cut-off yang diinginkan).

- Filter Band-Pass (BPF)
 - Fungsi: Mengizinkan hanya sinyal dalam rentang frekuensi tertentu (band) untuk lewat, sementara memblokir frekuensi di atas dan di bawah band tersebut.

- Aplikasi: Digunakan dalam aplikasi yang memerlukan isolasi sinyal dalam rentang frekuensi tertentu, misalnya dalam komunikasi nirkabel atau pengukuran getaran.
- Persamaan: Filter band-pass sering kali merupakan kombinasi dari filter low-pass dan high-pass, dengan dua frekuensi cut-off ((f_{low}) dan (f_{high})).
- Filter Kalman
 - Fungsi: Filter adaptif yang menggabungkan data dari sensor dengan model sistem untuk memprediksi dan mengoreksi sinyal yang masuk. Cocok untuk menangani data yang bising atau memiliki ketidakpastian tinggi.
 - Aplikasi: Sering digunakan dalam robotika, khususnya pada aplikasi yang melibatkan estimasi posisi dan kecepatan, seperti pada odometri atau IMU.
 - Contoh: Memperbaiki estimasi posisi robot otonom berdasarkan data sensor IMU dan encoder.
 - Persamaan (kalman filter sederhana):

$$\hat{X}_k = K_k \cdot Z_k + (1 - K_k) \cdot \hat{X}_{k-1}$$

Dalam hal ini, variabel $X(k)$ merupakan estimasi nilai state pada waktu ke- k , variabel $Z(k)$ merupakan measured value pada waktu ke- k , dan $K(k)$ merupakan nilai dari koefisien Kalman Gain. Nilai koefisien ini lah yang perlu dihitung agar $X(k)$ dapat mengestimasi nilai state lebih baik daripada $Z(k)$, karena jika $Z(k)$ sudah baik, kita mungkin tidak perlu lagi melakukan estimasi lebih lanjut. Estimasi ini dilakukan karena perhitungan $Z(k)$ melibatkan ketidakpastian (noise), dan pada algoritma ini diasumsikan distribusi nilai dari data Z mengikuti distribusi Gaussian. Selain persamaan diatas, ada juga pengembangan lebih lanjut dari Kalman Filter yakni Extended Kalman Filter dan Unscented Kalman Filter untuk sistem yang lebih kompleks.

- Filter Moving Average
 - Fungsi: Menghaluskan data dengan cara menghitung rata-rata dari beberapa nilai data terbaru.
 - Aplikasi: Digunakan untuk mengurangi fluktuasi kecil pada data sensor secara sederhana.
 - Contoh: Mengurangi noise pada sensor jarak ultrasonik atau inframerah.

b. Cara Kerja Filter Data

Pada dasarnya, filter bekerja dengan memodifikasi atau mengabaikan bagian dari sinyal data yang tidak diinginkan (seperti noise atau gangguan frekuensi tinggi) dan mempertahankan atau memperkuat bagian yang penting. Filter dapat diimplementasikan secara fisik (menggunakan komponen elektronik seperti resistor dan kapasitor) atau secara digital (menggunakan algoritme pemrosesan sinyal).

- Filter Fisik: Filter yang dibuat dengan komponen elektronik, biasanya diterapkan pada perangkat keras yang menangani sinyal analog.

- Filter Digital: Diterapkan menggunakan algoritme pada data yang telah diubah menjadi sinyal digital. Filter ini lebih fleksibel dan dapat dikustomisasi sesuai kebutuhan.

c. Aplikasi Filter Data dalam Robotika dan Mekatronika

- Pengolahan Data Sensor: Filter digunakan untuk membersihkan data dari sensor seperti accelerometer, gyroscope, atau GPS agar lebih akurat dan bebas dari noise.
- Kontrol Robot: Dalam sistem kontrol robot, filter memastikan bahwa sinyal yang digunakan untuk mengendalikan motor atau aktuator bebas dari gangguan yang dapat menyebabkan respon robot tidak stabil.
- Navigasi dan Odometri: Filter Kalman sering digunakan untuk menggabungkan data dari berbagai sensor (seperti IMU dan encoder) guna menghasilkan estimasi posisi yang lebih akurat dalam sistem navigasi robot otonom.

B. Model Sistem Kendali

Algoritme dalam robotika luarannya adalah membuat keputusan. Robot diberi tugas, tetapi untuk melakukan tugas itu harus mengambil tindakan dan tindakan ini tergantung pada lingkungan yang terdeteksi oleh sensor. Misalnya, jika robot akan membawa objek dari rak di gudang ke jalur pengiriman, ia harus menggunakan sensor untuk menavigasi ke rak yang tepat, untuk mendeteksi dan mengambil objek, dan kemudian menavigasi kembali ke truk dan memuat objek. Hanya robot yang bekerja di lingkungan yang terdefinisi dengan sangat baik yang dapat melakukan tugas-tugas tersebut tanpa informasi dari sensor. Contohnya adalah lengan robot yang merakit perangkat di pabrik; jika bagian-bagiannya diposisikan dengan tepat di permukaan kerja, robot bisa memanipulasi bagian-bagian tanpa merasakannya. Tetapi di sebagian besar lingkungan, sensor harus digunakan. Di gudang mungkin ada rintangan dalam perjalanan ke rak, objek tidak akan diposisikan dengan tepat di rak dan truk tidak pernah diparkir dengan tepat ditempat yang sama. Robot perlu beradaptasi dengan variasi kecil ini menggunakan kontrol algoritme untuk membuat keputusan: berdasarkan data dari sensor, tindakan apa yang dilakukan yang harus dilakukan robot untuk menyelesaikan tugas? Ada yang canggih teori kontrol matematika yang canggih yang sangat penting dalam robotika.

Pada intinya, yang disebut Sistem kendali adalah sistem yang dirancang untuk mengendalikan perilaku suatu proses atau sistem fisik. Tujuan dari sistem kendali adalah untuk menjaga output sesuai dengan nilai yang diinginkan (setpoint) meskipun ada gangguan atau perubahan kondisi. Sistem kendali banyak digunakan dalam robotika, otomasi industri, dan bidang teknik lainnya untuk memastikan sistem bekerja sesuai spesifikasi yang diinginkan.

e. Sistem Kendali Terbuka (Open-Loop)



Pemanggang roti adalah mesin yang melakukan tindakan semi-otonom. Anda menempatkan irisan roti ke dalam pemanggang roti, atur pengatur waktu dan tekan tuas ke bawah untuk memulai tindakan pemanggang. Seperti yang kita semua tahu, hasilnya tidak dijamin: jika durasi pengatur waktu terlalu pendek, kita harus pemanggang roti lagi;

jika durasi pengatur waktu terlalu lama, maka aroma roti panggang gosong akan menyebar ke seluruh dapur. Hasilnya tidak pasti karena pemanggang roti adalah sistem kontrol loop terbuka. Karena sistem tidak memeriksa hasil pemanggangan sebagai tindakan untuk melihat apakah hasil yang diinginkan telah tercapai.

Jadi, definisi dari sistem loop terbuka adalah sistem yang mengontrol keluaran tanpa umpan balik. Artinya, sistem hanya menjalankan aksi sesuai perintah tanpa memperhatikan hasil aktual dari proses. Persamaannya :

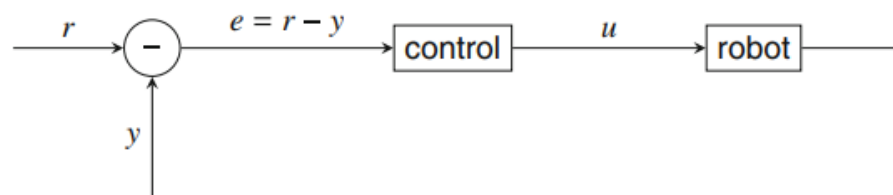
$$[y(t) = G(s)u(t)]$$

Di mana:

- $(y(t))$ adalah output,
- $(G(s))$ adalah transfer fungsi sistem, dan
- $(u(t))$ adalah input.

f. Sistem Kendali Tertutup (Closed-Loop)

Robot harus mengukur jarak ke objek dan berhenti ketika jarak ini cukup besar. Pengaturan daya motor tergantung pada pengukuran jarak, tetapi robot bergerak dengan kecepatan yang bergantung pada kekuatan motor yang mengubah jarak ke objek, yang sekali lagi mengubah pengaturan daya. Perilaku melingkar ini adalah asal mula istilah “loop tertutup”. Sehingga definisi dari sistem kendali tertutup adalah sistem yang mengontrol keluaran berdasarkan umpan balik dari hasil proses. Umpan balik digunakan untuk membandingkan hasil aktual dengan nilai yang diinginkan (setpoint).



Persamaan:

$$[y(t) = G(s)[u(t) + e(t)]]$$

Di mana:

- $(e(t))$ adalah error atau selisih antara setpoint dan output aktual,
- $(G(s))$ adalah transfer fungsi sistem,
- $(u(t))$ adalah input.

g. Periode Sistem Kontrol

Algorithm 6.1: Control algorithm outline	
integer period	// Duration of timer period
integer timer	// Timer variable
1: period \leftarrow ...	// Period in milliseconds
2: timer \leftarrow period	// Initialize the timer
3: loop	
4: when timer-expired-event occurs	
5: control algorithm	// Run the algorithm
6: timer \leftarrow period	// Reset the timer
// Operating system	
7: when hardware-clock-interrupt occurs	
8: timer \leftarrow timer - 1	// Decrement the timer
9: if timer = 0	// If the timer expires
10: raise timer-expired-event	// raise an event

Algoritme kontrol dijalankan secara berkala (tidak setiap waktu berjalan), lihat gambar diatas. Perangkat lunak robot menginisialisasi variabel pengatur waktu ke durasi periode yang diperlukan untuk menjalankan algoritme, misalnya misalnya, setiap 20 ms. Komputer tertanam memiliki jam perangkat keras yang “berdetak” pada interval tetap yang menyebabkan interupsi. Interupsi ditangani oleh sistem operasi yang mengurangi nilai variabel pengatur waktu. Ketika nilai variabel ini pergi ke nol, pengatur waktu telah kedaluwarsa, dan sebuah peristiwa muncul dalam perangkat lunak yang menyebabkan algoritme kontrol untuk dijalankan. Periode algoritma adalah parameter penting dalam desain kontrol sistem. Jika periode terlalu pendek, sumber daya komputasi yang berharga akan terbuang percuma dan komputer dapat menjadi kelebihan beban hingga perintah ke robot tiba terlambat. Jika periode terlalu lama, robot tidak akan merespons tepat waktu untuk memperbaiki kesalahan dalam gerakannya.

Sebagai contoh, pertimbangkan sebuah robot yang mendekati sebuah objek yang berjarak 10 cm dengan kecepatan 2 cm/s. Periode kontrol 1 ms akan membuang sumber daya komputasi karena robot akan bergerak hanya 0,002 cm (0,02 mm) selama setiap siklus 1 ms dari algoritme kontrol. Perubahan dalam daya motor pada jarak sekecil itu tidak akan mempengaruhi kemampuan robot memenuhi tugasnya. Pada ekstrem yang berlawanan, periode kontrol 2 detik bahkan lebih buruk: itu akan membuat robot akan bergerak 4 cm selama periode ini dan kemungkinan besar akan menabrak objek. Periode kontrol sekitar 0,25 detik (250 ms) di mana robot bergerak 0,5 cm tampaknya nilai yang masuk akal untuk memulai karena 0,5 cm adalah jarak yang berarti dalam hal mendekati suatu objek. Anda dapat bereksperimen dengan periode di sekitar nilai ini untuk menentukan periode optimum: yakni ketika periode menghasilkan perilaku yang memuaskan dengan periode yang selama mungkin untuk mengurangi biaya komputasi.

C. On-Off Control

- Pengertian: Sistem kendali yang paling sederhana, di mana sistem hanya memiliki dua status: ON dan OFF. Sistem ini bekerja berdasarkan apakah output sudah mencapai setpoint (target) atau belum.
- Cara Kerja:
 - o Jika nilai aktual lebih rendah dari setpoint, maka sistem ON (aktif).
 - o Jika nilai aktual lebih tinggi dari setpoint, maka sistem OFF (mati).
- Contoh: Thermostat di rumah yang menyalakan pemanas ketika suhu terlalu rendah dan mematikannya ketika suhu mencapai setpoint.
- Kelebihan: Sangat sederhana dan mudah diimplementasikan.

- Kekurangan: Sistem bisa mengalami "osilasi" atau perubahan status yang cepat antara ON dan OFF, yang bisa menyebabkan keausan pada perangkat keras.

D. Proportional Control (P)

- Pengertian: Algoritme ini menghasilkan sinyal kontrol yang proporsional terhadap error (selisih antara setpoint dan output aktual). Semakin besar error, semakin besar aksi kontrolnya.
- Persamaan:

$$[u(t) = K_p \cdot e(t)]$$

Di mana:

- $(u(t))$ adalah sinyal kontrol,
- (K_p) adalah konstanta penguatan proporsional,
- $(e(t))$ adalah error (selisih antara setpoint dan output).
- Cara Kerja: Jika error besar, sinyal kontrol juga besar, dan jika error kecil, sinyal kontrol kecil.
- Contoh: Pengaturan kecepatan motor berdasarkan kecepatan target. Jika motor terlalu lambat, tegangan meningkat; jika terlalu cepat, tegangan berkurang.
- Kelebihan: Mudah diimplementasikan dan respons cepat.
- Kekurangan: Tidak bisa sepenuhnya menghilangkan error, terutama jika ada gangguan atau perubahan beban.

E. Integral Control (I)

- Pengertian: Algoritme yang menghitung error secara akumulatif seiring waktu. Bagian integral menambah atau mengurangi aksi kontrol tergantung pada akumulasi error.
- Persamaan:

$$[u(t) = K_i \cdot \int e(t)dt]$$

Di mana:

- (K_i) adalah konstanta penguatan integral,
- $(e(t))$ adalah error.
- Cara Kerja: Error diakumulasikan, sehingga semakin lama error bertahan, semakin besar pengaruhnya terhadap sinyal kontrol.
- Contoh: Pengaturan suhu di mana sistem secara perlahan menambah daya sampai error hilang sepenuhnya.
- Kelebihan: Bisa menghilangkan error jangka panjang (menghilangkan offset).
- Kekurangan: Respons lambat dan dapat menyebabkan overshoot jika tidak disetel dengan baik.

F. Derivative Control (D)

- Pengertian: Algoritme ini merespon perubahan dari error, bukan nilai error itu sendiri. Bagian derivative memperkirakan arah dan kecepatan perubahan error.
- Persamaan:

$$[u(t) = K_d \cdot \frac{de(t)}{dt}]$$

Di mana:

- (K_d) adalah konstanta penguatan derivative,
- ($\frac{de(t)}{dt}$) adalah turunan dari error (tingkat perubahan error).
- Cara Kerja: Sistem memberikan aksi kontrol berdasarkan prediksi perubahan error di masa depan.
- Contoh: Pengendalian robot yang mencegah perubahan mendadak dalam kecepatan atau posisi, menjaga pergerakan tetap halus.
- Kelebihan: Meningkatkan stabilitas dan mempercepat respons sistem.
- Kekurangan: Sangat sensitif terhadap noise pada sinyal sensor.

G. PID Control (Proportional-Integral-Derivative)

- Pengertian: Kombinasi dari kontrol Proportional (P), Integral (I), dan Derivative (D). Sistem PID menggabungkan ketiga aksi tersebut untuk mengendalikan sistem dengan lebih akurat dan stabil.
- Persamaan:

$$[u(t) = K_p \cdot e(t) + K_i \cdot \int e(t)dt + K_d \cdot \frac{de(t)}{dt}]$$

Di mana:

- (K_p), (K_i), dan (K_d) adalah konstanta penguatan proporsional, integral, dan derivative.
- Penjelasan:
 - Proportional (P): Menghasilkan respons yang proporsional terhadap error. Semakin besar error, semakin besar aksi kontrol.
 - Integral (I): Menghitung akumulasi dari error seiring waktu, digunakan untuk menghilangkan offset.
 - Derivative (D): Mengantisipasi perubahan error di masa depan dengan menghitung kecepatan perubahan error.
- Contoh: Kendali kecepatan motor DC atau pengaturan posisi lengan robot - Contoh: Kontrol posisi robot otonom, kontrol suhu mesin, atau pengaturan kecepatan motor DC.
- Kelebihan: Sangat fleksibel dan digunakan di banyak aplikasi industri.
- Kekurangan: Memerlukan penyesuaian parameter (K_p), (K_i), dan (K_d) yang tepat agar sistem berfungsi optimal.

Tuning tiga parameter PID

K_p , K_i , dan K_d adalah masalah yang penting. Panduan berikut ini dapat digunakan untuk menemukan nilai yang sesuai secara eksperimental.

- Pilih pengaturan operasi tipikal untuk kecepatan yang diinginkan, matikan integral dan bagian turunan, kemudian tingkatkan K_p hingga maksimum atau hingga terjadi osilasi.
- Jika sistem berhasil, bagi K_p dengan 2.

- Tingkatkan K_d dan amati perilaku saat menaikkan / menurunkan kecepatan yang diinginkan sekitar 5%. Pilih nilai K_d yang memberikan respons teredam
- Tingkatkan K_i secara perlahan hingga osilasi dimulai. Kemudian bagi K_i dengan 2 atau 3.
- Periksa apakah kinerja pengontrol secara keseluruhan memuaskan di bawah kondisi sistem yang khas.

H. Bang-Bang Control (Hysteresis Control)

- Pengertian: Algoritme kendali ini mirip dengan on-off control, namun memiliki zona histeresis, yang berarti sistem tidak langsung beralih dari ON ke OFF atau sebaliknya ketika melewati setpoint, melainkan ada sedikit keterlambatan.
- Cara Kerja: Sistem akan tetap ON selama nilai aktual di bawah ambang batas bawah dan akan OFF ketika melebihi ambang batas atas.
- Contoh: Pengontrol suhu yang memiliki rentang toleransi untuk mencegah sistem sering berganti antara ON dan OFF.
- Kelebihan: Mencegah osilasi cepat (frekuensi switching yang tinggi).
- Kekurangan: Akurasi kontrol lebih rendah dibandingkan metode kendali lainnya.

I. Adaptive Control

- Pengertian: Algoritme kendali yang mampu menyesuaikan parameter kendalinya secara otomatis berdasarkan kondisi sistem atau lingkungan yang berubah-ubah.
- Cara Kerja: Sistem mengukur perubahan dalam perilaku sistem (misalnya beban yang berubah) dan secara otomatis menyesuaikan parameter seperti (K_p) , (K_i) , dan (K_d) .
- Contoh: Sistem kendali robot yang menyesuaikan pengaturan berdasarkan jenis permukaan atau kecepatan.
- Kelebihan: Dapat beradaptasi terhadap kondisi yang bervariasi, ideal untuk lingkungan yang dinamis.
- Kekurangan: Lebih kompleks dan memerlukan komputasi lebih banyak.

J. Fuzzy Logic Control

- Pengertian: Algoritme kendali yang bekerja berdasarkan logika fuzzy, di mana nilai-nilai kendali tidak lagi terbatas pada kondisi biner (seperti ON/OFF) melainkan memiliki nilai derajat kebenaran antara 0 dan 1.
- Cara Kerja: Fuzzy control menggunakan aturan berbasis logika linguistik untuk menentukan aksi kontrol. Misalnya, jika error kecil, maka sedikit meningkatkan sinyal kontrol.
- Contoh: Sistem kontrol suhu di AC yang lebih halus dan adaptif.
- Kelebihan: Mampu menangani ketidakpastian dan sistem nonlinear.
- Kekurangan: Memerlukan desain aturan fuzzy yang tepat.

BAB X

ROBOT OPERATING SYSTEM (ROS)

A. Pengenalan Robot Operating System (ROS)

Robot Operating System (ROS) adalah sebuah framework atau platform open-source yang dirancang untuk memudahkan pengembangan perangkat lunak (software) pada robot. Meski namanya "Operating System", ROS sebenarnya bukan sistem operasi penuh seperti Windows atau Linux, melainkan sebuah lapisan middleware yang berjalan di atas sistem operasi, biasanya Ubuntu (Linux).

ROS menawarkan berbagai alat dan pustaka (library) untuk membantu perancangan, pengembangan, dan pemrograman robot. Pada dasarnya, ROS adalah ekosistem yang menyediakan standar untuk komunikasi antar komponen robot, baik itu sensor, aktuator, ataupun algoritme kendali.

Mengapa Menggunakan ROS?

1. **Open-Source:** ROS gratis dan dikembangkan secara kolaboratif oleh komunitas di seluruh dunia.
2. **Dukungan yang Luas:** ROS mendukung berbagai jenis robot dan perangkat keras (sensor, aktuator, dll.).
3. **Komunikasi Antar Komponen:** Dengan ROS, sensor, aktuator, dan algoritme dapat saling berkomunikasi dengan mudah melalui arsitektur berbasis pesan.
4. **Fleksibel:** ROS dapat digunakan untuk robot-robot dari yang sederhana hingga yang sangat kompleks, seperti robot industri atau robot otonom.
5. **Tools Powerful:** ROS memiliki berbagai tools untuk simulasi, visualisasi, dan debugging yang sangat membantu pengembangan.

B. Konsep Dasar ROS

1. Node

- **Pengertian:** Dalam ROS, sebuah **node** adalah program eksekusi yang melakukan tugas-tugas tertentu pada robot. Misalnya, node untuk membaca data sensor, node untuk menggerakkan roda, atau node untuk melakukan pemrosesan gambar.
- **Cara Kerja:** Node berjalan secara independen, dan mereka berkomunikasi satu sama lain melalui pesan (messages). Node dapat dibuat menggunakan berbagai bahasa pemrograman seperti Python atau C++.

2. Topics

- **Pengertian: Topic** adalah saluran komunikasi di mana node mengirim atau menerima pesan. Node yang menghasilkan data (seperti dari sensor) akan "memublikasikan" (publish) pesan ke topic tertentu, dan node yang membutuhkan data tersebut akan "berlangganan" (subscribe) ke topic tersebut.
- **Contoh:** Sebuah sensor laser dapat memublikasikan data jaraknya ke topic /scan, dan node yang melakukan navigasi robot bisa berlangganan ke topic tersebut untuk mengambil data laser.

3. Messages

- **Pengertian:** Pesan (message) adalah format data yang dikirim antar node melalui topics. Pesan ini bisa berupa angka, teks, gambar, atau bahkan data yang lebih kompleks seperti peta.
- **Contoh:** Pesan yang dikirim oleh sensor jarak biasanya berupa data posisi x , y , dan z , atau jarak dari objek.

4. Services

- **Pengertian:** **Service** adalah metode komunikasi di mana satu node bisa meminta informasi dari node lain dan mendapatkan tanggapan secara langsung. Berbeda dengan topics yang bersifat satu arah, services bekerja seperti panggilan fungsi di pemrograman.
- **Contoh:** Sebuah node dapat meminta status baterai robot melalui service, dan node lain memberikan jawaban berupa informasi level baterai.

5. Action Servers

- **Pengertian:** **Action server** digunakan untuk tugas-tugas yang membutuhkan waktu lebih lama dan tidak bisa langsung selesai, seperti memindahkan robot ke posisi tertentu. Action memungkinkan komunikasi dua arah selama proses eksekusi.
- **Cara Kerja:** Client mengirimkan permintaan ke action server, dan server akan memberikan update status secara berkala sampai tugas selesai.

6. Packages

- **Pengertian:** **Package** adalah unit dasar dari perangkat lunak di ROS. Sebuah package bisa berisi kode, pustaka, node, dan file konfigurasi yang diperlukan untuk menjalankan robot.
- **Contoh:** Sebuah package bisa saja khusus untuk mengontrol motor robot atau mengimplementasikan algoritma navigasi.

7. Master (ROS Master)

- **Pengertian:** **ROS Master** bertindak sebagai pusat manajemen komunikasi antara node. Ia memastikan setiap node dapat menemukan node lain dan berkomunikasi melalui topics atau services.
- **Cara Kerja:** Saat node pertama kali dijalankan, mereka mendaftarkan diri ke ROS Master, sehingga ROS Master mengetahui node mana yang aktif dan apa fungsinya.

C. ROS Filesystem (Struktur File ROS)

ROS memiliki struktur direktori khusus yang terdiri dari berbagai file dan folder:

1. **Workspace:** Tempat di mana Anda mengatur dan mengelola kode proyek ROS Anda. Workspace biasanya berada di direktori `catkin_ws`.
2. **Source Folder (src):** Semua kode, package, dan file konfigurasi disimpan di dalam folder ini.
3. **Launch Files:** File `.launch` digunakan untuk meluncurkan beberapa node sekaligus, sehingga tidak perlu menjalankan node satu per satu secara manual.
4. **Bag Files:** File `.bag` digunakan untuk menyimpan data hasil operasi ROS seperti data sensor, sehingga bisa dianalisis kembali atau digunakan untuk simulasi.

D. ROS Tools











1. **RViz:** Sebuah alat visualisasi yang sangat penting untuk melihat data sensor seperti peta, posisi robot, jalur pergerakan, dan sebagainya. Data yang dipublikasikan oleh node bisa divisualisasikan dalam bentuk 3D di RViz.
2. **Gazebo:** Simulator fisika yang memungkinkan Anda mensimulasikan robot di lingkungan yang realistis. Dengan Gazebo, Anda bisa menguji robot dan algoritme tanpa perlu hardware fisik.
3. **rqt_graph:** Alat untuk melihat diagram hubungan antar node dalam sistem ROS, membantu memahami bagaimana data mengalir di dalam robot.

E. ROS 1 vs ROS 2

ROS awalnya dimulai dengan versi pertama, yang disebut **ROS 1**, namun karena berbagai batasan dan kebutuhan akan sistem yang lebih modern, ROS 2 diperkenalkan. Beberapa perbedaan utama antara ROS 1 dan ROS 2 adalah:

1. **Komunikasi:** ROS 1 menggunakan protokol TCP/UDP sederhana, sedangkan ROS 2 menggunakan DDS (Data Distribution Service) untuk komunikasi yang lebih handal, terutama dalam sistem real-time.
2. **Real-Time Support:** ROS 2 lebih baik dalam mendukung aplikasi yang membutuhkan waktu real-time, seperti di robot industri.
3. **Keamanan:** ROS 2 mendukung fitur keamanan yang lebih baik, sehingga cocok untuk aplikasi di lingkungan yang lebih kritis.

ROS 1

Distro	Release date	Poster	Tuturtle, turtle in tutorial	EOL date
ROS Noetic Ninjemys (Recommended)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			June 27, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017

ROS 2

Distro	Release date	Logo	EOL date
Jazzy Jalisco	May 23rd, 2024		May 2029
Iron Irtwin	May 23rd, 2023		November 2024
Humble Hawksbill	May 23rd, 2022		May 2027
Galactic Geochelone	May 23rd, 2021		December 9th, 2022

Bagaimana Memulai dengan ROS?

Untuk mulai belajar ROS, Anda perlu melakukan instalasi pada sistem operasi Ubuntu, karena ROS didukung dengan baik di Linux. Langkah-langkah umum untuk memulai (dibawah ini hanyalah ringkasan sederhana, untuk lebih lengkapnya silahkan kunjungi ros.org):

1. **Instal Ubuntu:** ROS paling sering digunakan di Ubuntu. Versi yang umum digunakan adalah Ubuntu 20.04 untuk ROS 1 (Noetic) dan ROS 2 (Foxy atau Humble).
2. **Install ROS:** Anda dapat menginstal ROS dengan mengikuti panduan resmi dari website ROS, biasanya dengan perintah berikut di terminal:

```
sudo apt install ros-noetic-desktop-full
```

untuk ROS 1, atau:

```
sudo apt install ros-foxy-desktop
```

untuk ROS 2.

3. **Buat Workspace:** Setelah ROS diinstal, Anda bisa membuat workspace di mana semua kode dan package akan disimpan. Gunakan perintah:

```
bash
Copy code
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

4. **Jalankan Node:** Untuk menjalankan node dalam ROS, Anda perlu menjalankan ROS Master (untuk ROS2, ROS master tidak perlu dijalankan secara manual pada terminal) dengan perintah:

```
roscore
```

Setelah ROS Master berjalan, Anda bisa menjalankan node-node lainnya.

F. Contoh Aplikasi ROS

1. **Robot Otonom:** ROS sering digunakan untuk robot otonom seperti kendaraan tanpa pengemudi (Autonomous Vehicles) atau robot pembersih lantai. ROS memungkinkan robot ini untuk membaca sensor seperti kamera atau LIDAR, membuat peta lingkungan, dan merencanakan jalur pergerakan.
2. **Robot Manipulator:** Dalam industri, ROS digunakan untuk mengontrol robot lengan (manipulator) yang digunakan dalam proses manufaktur. Dengan ROS, pengontrolan lengan robot menjadi lebih modular dan mudah diintegrasikan dengan sistem lain.
3. **Robot Drone:** Pada drone, ROS memungkinkan integrasi antara sensor IMU, GPS, dan kamera untuk penerbangan otomatis dan stabilisasi.

Robot Operating System (ROS) adalah platform middleware yang sangat penting dalam pengembangan perangkat lunak untuk robotika. Dengan ROS, Anda dapat menghubungkan berbagai komponen robot, mengembangkan algoritma canggih, dan memvisualisasikan hasilnya. Meski terlihat kompleks, ROS sebenarnya sangat fleksibel dan bisa digunakan untuk berbagai jenis robot, dari yang sederhana hingga yang canggih. Bagi yang baru belajar, ROS menyediakan banyak dokumentasi, tutorial, dan komunitas yang besar untuk mendukung perjalanan belajar Anda.