BAB II
KAJIAN PUSTAKA DAN LANDASAN TEORI

Berikut ini adalah beberapa landasan teori dari berbagai referensi sebagai penunjang agar tercapai hasil yang maksimal dari proses pengerjaan penelitian ini.

2.1 Alat Pelipat Pakaian (*TERAPSI)*

Alat pelipat pakaian adalah sebuah alat bantu yang digunakan untuk melipat pakaian, dimana melipat pakaian merupakan kegiatan yang rutin dilakukan di rumah tangga maupun di kegiatan usaha pakaian seperti binatu (*laundry*). Alat pelipat pakaian sederhana telah dikembangkan oleh tangan kreatif lima orang anak bangsa dengan alat yang diberi nama *Terapsi* [2]. Alat bantu pelipat ini sejak awalnya sengaja diciptakan untuk membantu aktifitas menyetrika dan melipat baju. Munculnya ide mengembangkan alat bantu pelipat pakaian ini oleh kelompok mahasiswa di Yoyakarta yaitu sejak awal januari tahun 2012. Menggunakan sehelai karton yang berukuran 60 x 80 cm yang dipotong simetris untuk mendapatkan lipatan tiga bagian.

Gambar 2.1 *Terapsi* Hasil Desain Mahasiswa di Yogyakarta [2]

Penggunaan alat pelipat *Terapsi* yang masih secara manual yaitu menggunakan tangan manusia untuk melakukan proses pelipatan. Dengan 3 langkah pelipatan yang disesuaikan dengan ukuran pelipatan maka pakaian telah selesai dilipat. Proses pelipatan yang masih menggunakan tangan akan

membutuhkan tenaga manusia sehingga pengerjaan bergantung pada batas kemampuan tenaga manusia tersebut. Sehingga untuk pengembangan dapat dikombinasikan dengan teknologi otomasi agar proses pelipatan dapat mengurangi penggunaan tenaga manusia.

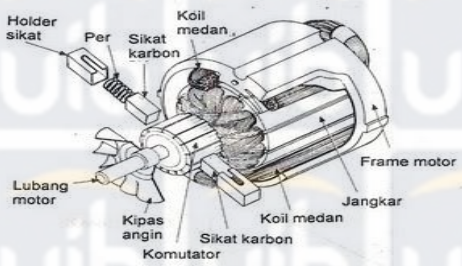
Bentuk dan desain alat *Terapsi* ini masih sangat sederhana yaitu berbentuk lembaran tebal kertas karton yang dilapisi kertas furing, sehingga masih memerlukan meja untuk melakukan proses pelipatan. Bahan karton juga tidak awet dan mudah robek karena terbuat dari kertas sehingga *lifetime* dari alat *Terapsi* cukup pendek 6 – 12 bulan [2].

2.2 Motor DC

Motor DC merupakan jenis motor yang menggunakan tegangan searah (DC) sebagai sumber tenaganya. Dengan memberikan beda tegangan pada kedua terminal, motor akan berputar pada satu arah dan bila polaritas dari tegangan tersebut dibalik maka arah putaran motor akan terbalik pula. Polaritas dari tegangan yang diberikan pada dua terminal menentukan arah putaran motor sedangkan besar dari beda tegangan pada kedua terminal menentukan kecepatan motor. Motor DC memiliki 2 bagian dasar :

1. Bagian yang tetap / stasioner yang disebut stator. Stator ini menghasilkan medan magnet, baik yang dibangkitkan dari sebuah *coil* (elektro magnet) ataupun magnet permanen.
2. Bagian yang berputar disebut rotor. Rotor ini berupa sebuah koil dimana arus listrik mengalir. Rotor ini terdiri dari *Armature* dan *Commutator*.
    a. *Armature*, adalah sebuah kumparan yang dililitkan pada lempengan-lempengan besi untuk membangkitkan medan elektromagnetik melewati bagian tengah kumparan.
    b. *Commutator*, berfungsi untuk merubah-rubah arah arus pada saat motor DC ini bekerja. *Commutator* ini berbentuk cincin yang terbagi menjadi 2 bagian tembaga terpisah.

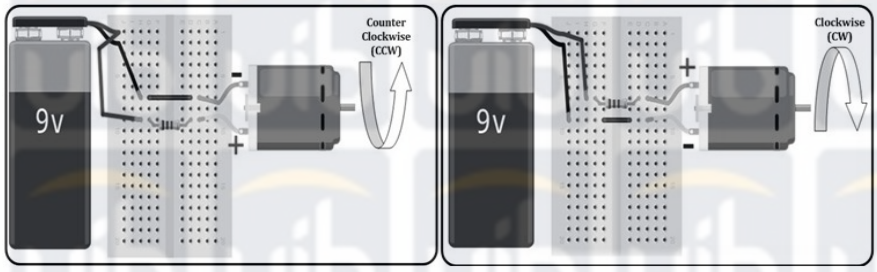
Gaya elektromagnetik pada motor DC timbul saat ada arus yang mengalir pada penghantar yang berada dalam medan magnet. Medan magnet itu sendiri ditimbulkan oleh magnet permanen. Garis-garis gaya magnet mengalir diantara dua kutub magnet dari kutub utara ke kutub selatan. Menurut hukum gaya Lourentz, arus yang mengalir pada penghantar yang terletak dalam medan magnet akan menimbulkan gaya. Gaya F, timbul tergantung pada arah arus I, dan arah medan magnet B.



Gambar 2.2 Konstruksi Motor DC [5]

Belitan stator merupakan elektromagnet, dengan penguat magnet terpisah. Belitan jangkar ditopang oleh poros dengan ujung-ujungnya terhubung ke komutator dan sikat arang. Arus listrik DC pada penguat magnet mengalir dan menghasilkan medan magnet yang memotong belitan jangkar.

Pengaturan arah putaran motor dilakukan dengan mengubah arah polaritas yang mengalir melalui motor. Secara sederhana seperti yang terlihat pada Gambar 2.3 hal ini dapat dilakukan dengan mengubah polaritas tegangan motor.



Gambar 2.3 Pengaturan Arah Putaran Motor DC

Sumber : Diolah dari Data Primer (12 Januari 2015)

Kecepatan motor DC dapat diatur dengan beberapa cara, yaitu dengan mengatur fluks medan, dengan mengatur tahanan jangkar, dan dengan mengatur

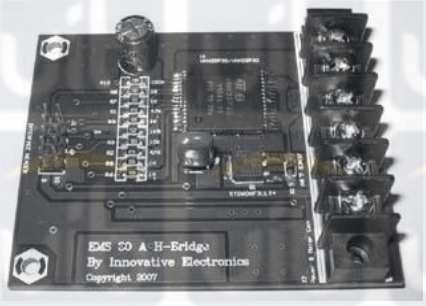
tegangan sumber. Cara yang ketiga ini merupakan pengaturan yang sering digunakan karena penggunaannya yang relatif mudah (Zuhal, 2004). Pengaturan tegangan sumber biasanya menggunakan metode PWM (*Pulse Width Modulation*).

Arah putaran motor DC dapat diubah dengan mengubah polaritas aliran arus yang terhubung ke sikat-sikatnya. Sedangkan kecepatan putar motor tergantung dari berapa besar arus yang mengalir (Paulus Andi Nalwan, Delta Elektronik, 2013).

2.3 *EMS H-Bridge Driver* Motor DC

Aktuator dalam robotika adalah suatu komponen yang sangat penting, salah satu aktuator yang sering digunakan adalah motor DC. Dalam penggunaan motor DC sangat diperlukan pengontrolan arah dan pengontrolan kecepatan putaran motor DC. Salah satu solusi untuk pengontrolan arah putar motor DC adalah menggunakan *driver* motor DC *H-Bridge*. Pengontrolan yang dilakukan *driver H-Bridge* adalah pengontrolan untuk mengatur polaritas yang diterima oleh motor DC sehingga arah putar motor dapat berubah.

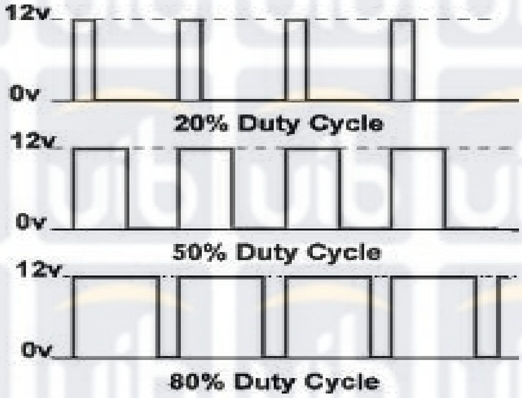
Modul *driver* motor DC yang digunakan adalah *EMS 30A H-Bridge.* Secara garis besar, fungsi *driver* motor ini adalah untuk mengendalikan arah dan kecepatan putaran motor DC sesuai instruksi kendali dari mikrokontroler Arduino Mega 2560. Berikut pada Gambar 2.4 adalah modul *driver* motor DC *EMS 30A H-Bridge* menggunakan *IC VHN2SP30*.



Gambar 2.4 *Driver EMS 30A H-Bridge* Menggunakan IC VHN2SP30 [7]

Driver motor DC *EMS 30A H-Bridge* ini dapat mengendalikan arah putaran motor DC dalam dua arah dan dapat dikontrol dengan metode PWM (*Pulse Width Modulation*) [7].

Secara umum prinsip dasar PWM yaitu membuat kondisi ON atau OFF dengan frekuensi dan waktu tertentu. Sehingga rasio ON terhadap waktu total adalah jumlah lamanya waktu ON dan waktu OFF. Hal ini dapat dinyatakan dalam bentuk persen (%) seperti pada Gambar 2.5 dibawah ini.



Gambar 2.5 Sinyal PWM (*Pulse Width Modulation)* [7]

Sehingga dapat disimpulkan bahwa semakin lama waktu ON yang kita berikan, maka kecepatan putaran motor akan mendekati kecepatan maksimum. Adapun persamaan perhitungan besaran kecepatan motor adalah.

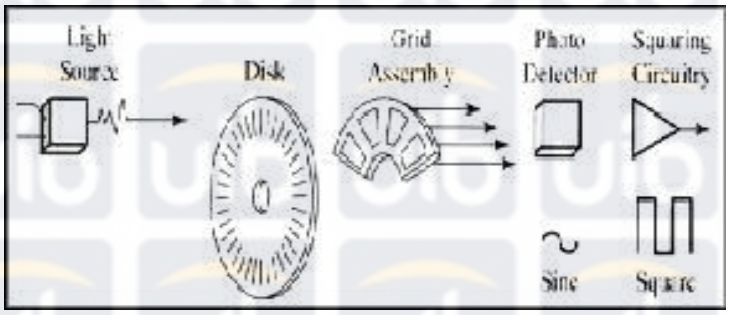
Kecepatan Motor (%) = $\frac{Waktu\ ON}{Waktu\ Total}\ x\ 100$……………………………..…......…(1)

2.4 Sensor *Rotary Encoder*

Rotary encoder adalah *device elektromechanic* yang dapat memonitor gerakan dan posisi. *Rotary encoder* umumnya menggunakan sensor optik untuk menghasilkan serial pulsa yang dapat diartikan menjadi gerakan, posisi, dan arah. Sehingga posisi sudut suatu poros benda berputar dapat diolah menjadi informasi berupa kode digital oleh *rotary encoder* untuk diteruskan oleh rangkaian kendali. *Rotary encoder* umumnya digunakan pada pengendalian robot, *motor driver*, dsb.

Rotary encoder tersusun dari suatu piringan tipis yang memiliki lubang-lubang pada bagian lingkaran piringan. *LED* ditempatkan pada salah satu sisi piringan sehingga cahaya akan menuju ke piringan. Di sisi yang lain suatu *photo-transistor* diletakkan sehingga *photo-transistor* ini dapat mendeteksi cahaya dari *LED* yang berseberangan. Piringan tipis tadi dikopel dengan poros motor, atau

divais berputar lainnya yang ingin kita ketahui posisinya, sehingga ketika motor berputar piringan juga akan ikut berputar. Apabila posisi piringan mengakibatkan cahaya dari *LED* dapat mencapai *photo-transistor* melalui lubang-lubang yang ada, maka *photo-transistor* akan mengalami saturasi dan akan menghasilkan suatu pulsa gelombang persegi [6]. Gambar 2.6 menunjukkan skematik sederhana dari *rotary encoder*. Semakin banyak deretan pulsa yang dihasilkan pada satu putaran menentukan akurasi *rotary encoder* tersebut, akibatnya semakin banyak jumlah lubang yang dapat dibuat pada piringan menentukan akurasi *rotary encoder* tersebut.



Gambar 2.6 Blok Penyusun *Rotary Encoder* [6]

Rangkaian penghasil pulsa pada Gambar 2.6 yang digunakan umumnya memiliki output yang berubah dari 5V menjadi 0.5V ketika cahaya diblok oleh piringan dan ketika diteruskan ke *photo-transistor*. Karena divais ini umumnya bekerja dekat dengan motor DC maka banyak *noise* yang timbul sehingga biasanya output akan dimasukkan ke *low-pass filter* dahulu. Apabila *low-pass filter* digunakan, frekuensi *cut-off* yang dipakai umumnya ditentukan oleh jumlah slot yang ada pada piringan dan seberapa cepat piringan tersebut berputar, dinyatakan dengan:

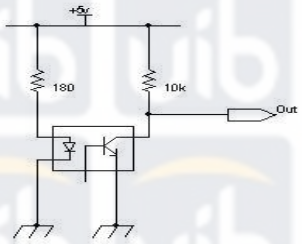
$$fc = \frac{S_w \cdot n}{60} \quad \text{..... (2)}$$

dimana :

f_c = frekuensi cut-off filter,

s_w = kecepatan piringan dan

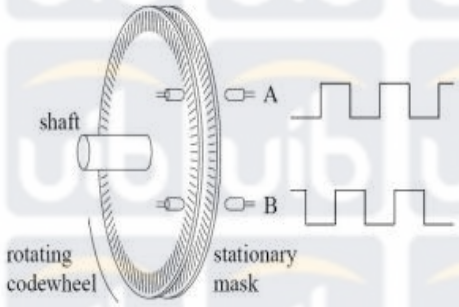
n = jumlah slot pada piringan.



Gambar 2.7 Rangkaian Tipikal Penghasil Pulsa *Rotary Encoder* [6]

2.4.1 *Incremental Rotary Encoder*

Incremental rotary encoder terdiri dari dua *track* atau *single track* dan dua sensor yang disebut *channel* A dan B pada Gambar 2.8. Ketika poros berputar, deretan pulsa akan muncul di masing-masing *channel* pada frekuensi yang proporsional dengan kecepatan putar sedangkan hubungan fasa antara *channel* A dan B menghasilkan arah putaran. Dengan menghitung jumlah pulsa yang terjadi terhadap resolusi piringan maka putaran dapat diukur. Untuk mengetahui arah putaran, dengan mengetahui *channel* mana yang *leading* terhadap *channel* satunya dapat kita tentukan arah putaran yang terjadi karena kedua *channel* tersebut akan selalu berbeda fasa seperempat putaran *(quadrature signal)*. Seringkali terdapat output *channel* ketiga, disebut *INDEX*, yang menghasilkan satu pulsa per putaran berguna untuk menghitung jumlah putaran yang terjadi.



Gambar 2.8 Susunan Piringan *Incremental Encoder* [6]

Pada *incremental encoder*, beberapa cara dapat digunakan untuk menentukan kecepatan yang diamati dari sinyal pulsa yang dihasilkan dengan menggunakan *frequencymeter* dan periodimeter [6]. Diantaranya adalah:

$$\omega_1 = \frac{\alpha_f}{T} \quad \text{..... (3)}$$

dimana:

ω_1 = kecepatan putar,

α_f = sudut putaran satu periode pulsa encoder, dan

T = selang waktu yang tetap.

Cara yang sederhana untuk menentukan kecepatan dapat dengan *frequencymeter*, yakni menghitung jumlah pulsa dari *encoder* (n) pada selang waktu yang tetap, T yang merupakan periode loop kecepatan. Apabila α adalah sudut antara pulsa *encoder*, maka sudut putaran pada suatu periode adalah:

$$\alpha_f = n.\alpha \quad \text{..... (4)}$$

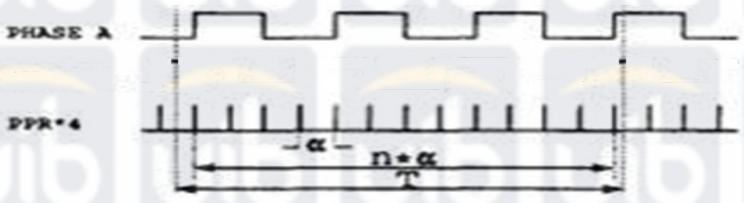
dimana:

α_f = sudut putaran satu periode pulsa *encoder*,

n = jumlah pulsa dari *encoder*, dan

α = sudut antara pulsa *encoder*.

Kelemahan yang muncul pada cara ini adalah pada setiap periode sudut α_f yang didapat merupakan kelipatan integer dari α. Ini akan dapat menghasilkan *quantification error* pada kecepatan yang ingin diukur.



Gambar 2.9 Sinyal Ukur Kecepatan *Encoder* dengan *Frequencymeter* [6]

Cara yang lain adalah dengan menggunakan periodimeter. Dengan cara ini kita akan mengukur kecepatan tidak lagi dengan menghitung

jumlah pulsa *encoder* tetapi dengan menghitung *clock* frekuensi tinggi (HF *clock*) untuk sebuah pulsa dari *encoder* yaitu mengukur periode pulsa dari *encoder* pada Gambar 2.10.

Apabila α_p adalah sudut dari pulsa *encoder*, t adalah periode dari HF *clock*, dan n adalah jumlah pulsa HF yang terhitung pada *counter* [10]. Maka waktu untuk sebuah pulsa *encoder*, T_p, adalah:

$$T_p = n.t \quad \text{.....(5)}$$

dimana:

T_p = waktu yang diperlukan untuk sebuah pulsa *encoder*,

n = jumlah pulsa HF yang terhitung pada *counter*, dan

t = periode HF *clock*.

Sehingga kecepatan yang akan kita ukur dapat kita peroleh dengan:

$$\omega_1 = \frac{\alpha_p}{T_p} \quad \text{.....(6)}$$

dimana:

ω_1 = kecepatan putar,

α_p = sudut dari pulsa *encoder*, dan

T_p = waktu yang diperlukan untuk sebuah pulsa *encoder*.

Seperti pada *frequencymeter*, disini juga muncul *quantification error* karena waktu T_p akan selalu merupakan perkalian *integer* dengan t.

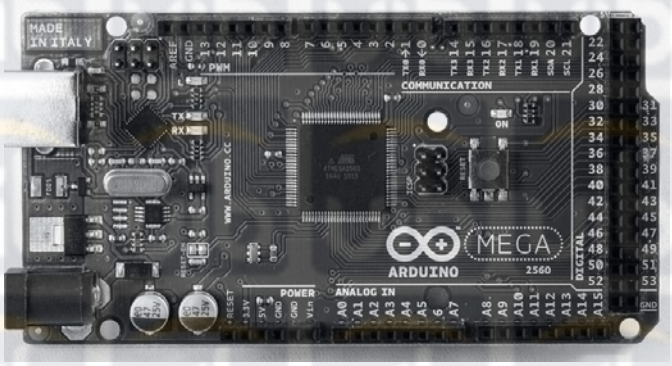
Gambar 2.10 Pengukuran Kecepatan Menggunakan *Periodimeter* [6]

2.5 Arduino MEGA 2560

Arduino MEGA 2560 adalah sebuah *board* mikrokontroller yang berbasis ATmega2560. Arduino memiliki 54 pin *input/output* yang mana 15 pin dapat digunakan sebagai *output* PWM, 16 pin analog *input*, 4 pin UART, *crystal osilator* 16 MHz, koneksi *USB*, *jack power*, kepala ICSP, dan tombol reset.

Arduino mampu *support* ke mikrokontroller, dapat dikoneksikan dengan komputer menggunakan kabel *USB* atau listrik AC yang ke adaptor AC-DC atau baterai untuk menjalankannya. Arduino disebut juga *single board microcontroller* (mikrokontroler dalam satu papan rangkaian) yang bersifat *open source* dan sangat populer saat ini. Merupakan turunan dari *platform wiring* dan dirancang agar pembuatan proyek mikrokontroler menjadi lebih mudah dilakukan oleh semua kalangan. Sistem Arduino adalah berupa *hardware* menggunakan *chip* Atmel AVR, *software* yang berupa bahasa pemrograman standar C, serta *bootloader* yang dipasang pada *chip* utama [8].

Hardware arduino diprogram menggunakan bahasa pemrograman C/C++, hanya saja sudah disederhanakan dan dimodifikasi. Arduino mengikuti pola pemrograman *wiring* (*syntax* dan *library*). Sementara untuk editor pemrograman (IDE – *Intergrated Development Enviroment*) dikembangkan dari *processing.*



Gambar 2.11. Arduino MEGA 2560 [8]

Adapun data teknis board Arduino MEGA 2560 adalah sebagai berikut :

- Mikrokontroler : ATmega2560
- Tegangan Operasi : 5V

- Tegangan *Input* (*recommended*) : 7 - 12 V
- Tegangan *Input* (limit) : 6 - 20 V
- Pin digital I/O : 54 pin (15 pin PWM)
- Pin Analog *input* : 16 pin
- Arus DC per pin I/O : 40 mA
- Arus DC untuk pin 3.3 V : 50 mA
- Flash Memory : 256 KB dengan 8 KB digunakan untuk *bootloader*
- SRAM : 8 KB
- EEPROM : 4 KB
- Kecepatan Pewaktuan : 16 Mhz

2.6 *Fuzzy Logic*

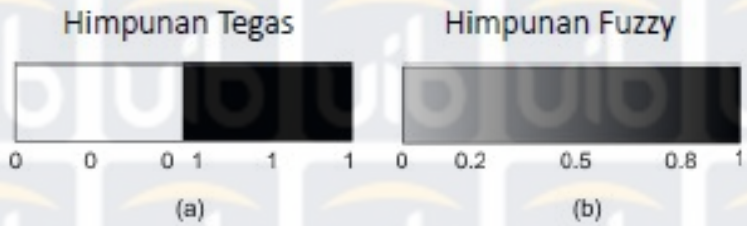
2.6.1 *Fuzzy Logic*

Logika *Fuzzy* pertama kali diperkenalkan oleh Zadeh dari Universitas California di Barkeley (1965). *Fuzzy* secara bahasa diartikan sebagai kabur atau samar-samar, suatu nilai dapat bernilai benar maupun salah secara bersamaan. Dalam *fuzzy* dikenal derajat keanggotaan yang memiliki rentang nilai 0 (nol) hingga 1 (satu). Berbeda dengan himpunan tegas yang memiliki nilai 1 atau 0.

Logika *Fuzzy* merupakan sesuatu logika yang memiliki nilai kekaburan atau kesamaran (*fuzzyness*) antara benar atau salah. Dalam teori logika *fuzzy* suatu nilai bisa bernilai benar atau salah secara bersama. Namun berapa besar keberadaan dan kesalahan suatu tergantung pada bobot keanggotaan yang dimilikinya. Logika *fuzzy* digunakan untuk menterjemahkan suatu besaran yang diekspresikan menggunakan bahasa (*linguistic*), misalkan besaran kecepatan laju kendaraan yang diekspresikan dengan pelan, agak cepat, cepat, dan sangat cepat.

Dan logika *fuzzy* menunjukan sejauh mana suatu nilai itu benar dan sejauh mana suatu nilai itu salah. Tidak seperti logika klasik *crisp* / tegas, suatu nilai hanya mempunyai 2 kemungkinan yaitu merupakan suatu

anggota himpunan atau tidak. Derajat keanggotaan 0 (nol) artinya nilai bukan merupakan anggota himpunan dan 1 (satu) berarti nilai tersebut adalah anggota himpunan.



Gambar 2.12 Perbedaan Himpunan Tegas dan Himpunan *Fuzzy* [9]

Secara umum, logika *fuzzy* adalah sebuah metodologi "berhitung" dengan variabel kata-kata (*linguistic*), sebagai pengganti berhitung dengan bilangan. Kata-kata yang di gunakan dalam logika *fuzzy* memang tidak sepresisi bilangan, namun kata-kata jauh lebih dekat dengan bahasa manusia sehari-hari [9].

2.6.2 Himpunan *Fuzzy (Fuzzy Set)*

Dalam teori logika *fuzzy* dikenal himpunan *fuzzy* (*fuzzy set*) yang merupakan pengelompokkan sesuatu berdasarkan bahasa (*linguistic variable*), yang dinyatakan dalam fungsi keanggotaan [10]. Di dalam semesta pembicaraan (*universe of discourge*) U, fungsi keanggotaan dari suatu himpunan *fuzzy* tersebut bernilai antara 0,0 sampai dengan 1,0.

Salah satu bentuk contoh dari himpunan variabel bahasa adalah himpunan jarak yang dapat dinyatakan dengan : 1) Dekat; 2) Agak Jauh; 3) Jauh.

2.6.3 Fungsi Keanggotaan (*Membership Function*)

Fungsi keanggotaan dari suatu himpunan *fuzzy* dinyatakan dengan derajat keanggotaan suatu nilai terhadap nilai tegasnya yang berkisar antara 0,0 sampai dengan 1,0. Jika himpunan *fuzzy* (A), fungsi

keanggotaan (μA), semesta (x), maka fungsi keanggotaan dalam suatu himpunan *fuzzy* dapat dinyatakan dengan :

$$A = \{x, \mu A(x) | x \in X\} \quad \text{..... (7)}$$

Fungsi keanggotaan dari *fuzzy* dapat ditentukan dengan tiga fungsi, yaitu Segitiga (*Triangle*), Trapesium (*Trapezoidal*), dan fungsi Gauss (*Gaussian*). Berikut bentuk persamaan dari fungsi-fungsi tersebut:

1. Segitiga *(Triangle).*

Fungsi Keanggotaan:

$$\mu(x) = \begin{cases} 0 & ; x \leq a \; atau \; x \geq c \\ \frac{(x-a)}{(b-a)} & ; a \leq x \leq b \\ \frac{(c-x)}{(c-b)} & ; b \leq x \leq c \end{cases} \quad \text{..... (8)}$$

Persamaan diatas jika digambarkan dalam grafik dapat dilihat seperti gambar dibawah ini:

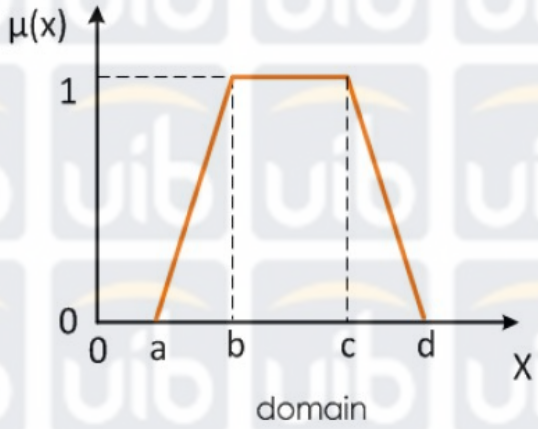
Gambar 2.13 Segitiga *(Triangle Membership)* [10]

2. Trapesium *(Trapezoidal).*

Fungsi Keanggotaan:

$$\mu(x) = \begin{cases} 0 & ; x \leq a \; atau \; x \geq d \\ \frac{(x-a)}{(b-a)} & ; a \leq x \leq b \\ 1 & ; b \leq x \leq c \\ \frac{(d-x)}{(d-c)} & ; c \leq x \leq d \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

Persamaan diatas jika digambarkan dalam grafik dapat dilihat seperti gambar dibawah ini:



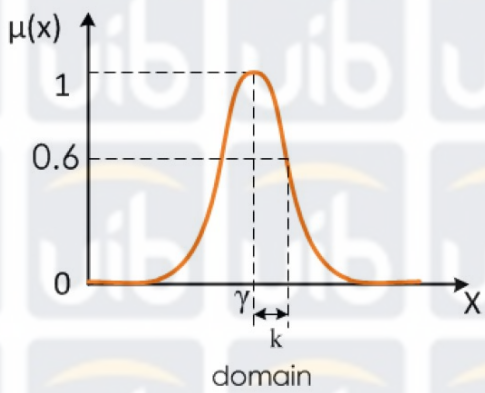
Gambar 2.14 Trapesium *(Trapezoidal Membership)* [10]

3. Gauss *(Gaussian).*

Fungsi Keanggotaan:

$$\mu(x;\gamma,k) = exp\left[-\frac{1}{2}\left|\frac{x-\gamma}{k}\right|^2\right] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (10)$$

Persamaan diatas jika digambarkan dalam grafik dapat dilihat seperti gambar dibawah ini:



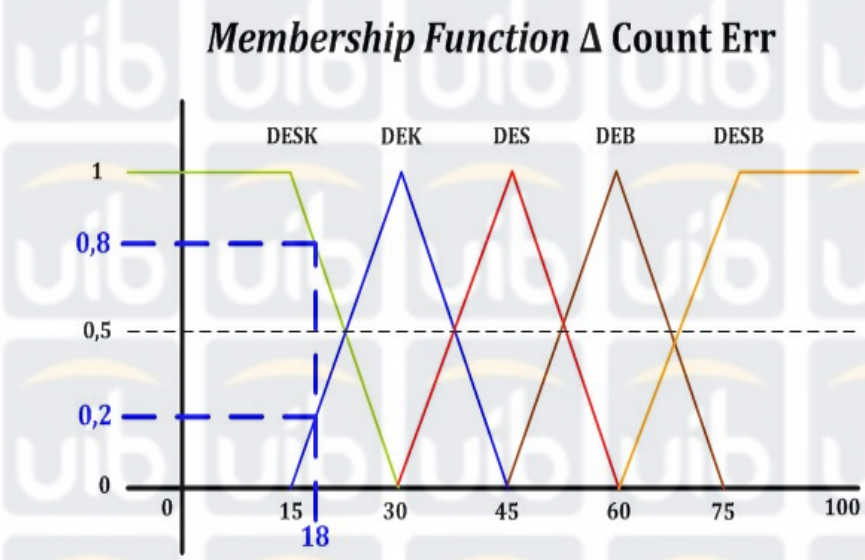
Gambar 2.15 Gauss *(Gaussian Membership)* [10]

2.6.4 Fuzzifikasi *(Fuzzification)*

Proses fuzzifikasi merupakan proses untuk mengubah variabel non-*fuzzy* (variabel-numerik) menjadi variabel *fuzzy* (variabel linguistik). Nilai masukan yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh pengendali *fuzzy* harus diubah terlebih dahulu kedalam variabel *fuzzy*. Melalui fungsi keanggotaan yang telah disusun maka dari nilai-nilai masukan tersebut menjadi informasi *fuzzy* yang berguna nantinya untuk proses pengolahan secara *fuzzy*, dengan kata lain fuzzifikasi adalah merupakan pemetaan titik-titik numeric atau suatu proses pengubahan nilai tegas / real yang ada ke dalam fungsi keanggotaan.

Proses fuzzifikasi merupakan proses untuk mengubah variabel non-*fuzzy* (variabel-numerik) menjadi variabel *fuzzy* (variabel linguistik). Nilai masukan yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh pengendali *fuzzy* harus dirubah terlebih dahulu kedalam variabel *fuzzy*.

Melalui fungsi keanggotaan yang telah disusun maka dari nilai-nilai masukan tersebut menjadi informasi *fuzzy* yang berguna nantinya untuk proses pengolahan secara *fuzzy*, dengan kata lain fuzzifikasi adalah merupakan pemetaan titik-titik numerik atau suatu proses pengubahan nilai tegas / riil yang ada ke dalam fungsi keanggotaan.



Gambar 2.16 Proses *Fuzzification*

Sumber : Diolah dari Data Primer (09 November 2014)

Pada gambar 2.16 adalah merupakan proses fuzzifikasi dengan menggunakan keanggotaan fungsi Segitiga *(Triangle)* dan Trapesium *(Trapezoidal).* Contoh perhitungan fuzzifikasi dapat ditunjukkan sebagai berikut:

$$\mu_{DESK}(18) = \frac{x-c}{b-c} = \frac{18-30}{15-30} = \frac{-12}{-15} = \mathbf{0,8} \quad \text{..... (11)}$$

$$\mu_{DEK}(18) = \frac{b-x}{b-c} = \frac{15-18}{15-30} = \frac{-3}{-15} = \mathbf{0,2} \quad \text{..... (12)}$$

2.6.5 Aturan Dasar Kontrol Logika Fuzzy (*Rule Based*)

Aturan dasar (*rule based*) pada kontrol *fuzzy* merupakan suatu bentuk aturan relasi / implikasi "Jika – Maka" atau "*if – then*" sebagai contoh adalah "Jika" S = 1 "Maka" M = 0.

Contoh dari aturan "Jika – Maka" ini pada pengendalian kecepatan motor DC dengan berdasarkan kondisi sensor jarak adalah sebagai berikut : **"JIKA" deteksi jauh "MAKA" PWM motor DC naik.**

2.6.6 Defuzzifikasi (*Defuzzification*)

Proses defuzzifikasi adalah pemetaan himpunan fuzzy yang diperoleh dari komposisi aturan-aturan fuzzy, sedangkan keluaran yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* dalam *range* tertentu, maka harus dapat diambil suatu nilai tegas (*crisps)* tertentu sebagai keluaran. Model strategi defuzzifikasi adalah:

1. Metode Mamdani

Metode ini juga disebut dengan metode COG *(Center Of Gravity)* dengan persamaan sebagai berikut:

$$COG[C(z)] = \frac{\int_{-\infty}^{\infty} zC(z)\,dz}{\int_{-\infty}^{\infty} C(z)\,dz} \quad \text{.....(13)}$$

2. Metode Sugeno

Metode ini juga disebut WA *(Weighted Average)* atau COA *(Center Of Area)* dengan persamaan sebagai berikut:

$$WA = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m}\propto(i,j) * \mu c(i,j)}{\sum_{i=0}^{n}\sum_{j=0}^{m}\propto(i,j)} \quad \text{..... (14)}$$

2.7 *Fuzzy Proportional Derivative (FPD) Controller*

Sistem kendali merupakan komponen penting yang berfungsi untuk membandingkan sinyal keluaran dengan sinyal acuan (Aström & Hägglund, 1988 dan Hartanto & Praseto, 2003). Kontroler PID merupakan gabungan tiga kontroler terpisah yaitu *Proportional*, *Integral*, dan *Derivative.* Masing – masing aksi kontrol ini memberikan kontribusi kontrol pada *error* yang terjadi *(Proportional)*, jumlah *error (Integral)*, dan perubahan *error (Derivative).* Kombinasi tiga kontroler ini dapat mempercepat *steady state*, mengurangi *over-shoot*, dan mengurangi *setting time* osilasi dari sistem yang dikontrol. Penggunaan *self tuning* PID pada sistem kontrol memberikan perbaikan secara signifikan stabilitas yang dinamis. Parameter penguat PID ditala secara adaptif *self tuning* [11].

Penggunaan *Fuzzy* PD sebagai metode sistem kontrol untuk menggantikan konstanta-konstanta *Proportional* dan *Derivative* dari kontroler PD konvensional dengan metode *Fuzzy Logic*, sehingga masukan untuk Fuzzy Logic adalah *error* dan *change of error* yang mereprensentasikan *Proportional* dan *Derivative* dari sinyal masukan.

Sinyal keluaran *y(t)* dibandingkan dengan *set point* *r(t)* akan menghasilkan *error e(t).*

$$e(t) = r(t) - y(t) \quad \text{.....(15)}$$

Dimana :

$r(t)$ = referensi *(set point),*

$y(t)$ = keluaran *(process variable).*

Sinyal *error e(t)* selanjutnya diproses dalam pengendali, hasilnya adalah sinyal keluar pengendali yaitu *u(t),* yang disebut sebagai *Manipulated Variable* (MV)[11].

$$u(t) = (parameter\ pengendali) - e(t) \quad \text{.....(16)}$$

Sistem kendali PID *Fuzzy* dapat diturunkan berdasarkan PID analog, yaitu:

Error E :

$$e(t) = r(t) - y(t) \quad \text{.....(17)}$$

Penjumlahan *error* SE :

$$SE(t) = e(t) + e(t-1) \quad \text{.....(18)}$$

Perubahan *error* CE :

$$CE(t) = e(t) + e(t-1) \quad \text{.....(19)}$$

Berdasarkan sistem kendali proporsional analog, sinyal keluaran adalah:

$$u(t) = Kp.e(t)$$……..(20)

Maka untuk sistem kendali proporsional *Fuzzy* sinyal keluaran adalah sebagai berikut:

$$U = Kp.E$$……..(21)

Dimana :

Kp = penguatan sinyal proporsional.

Sedangkan untuk sinyal keluaran PD *(Proportional Derivative) Fuzzy*, sinyal keluaran PD analog adalah sebagai berikut:

$$u(t) = Kp.E(t) + Kp.Td\frac{d\,e(t)}{dt}$$ ……(22)

$$u(t) = Kp.E(t) + Kd\frac{d\,e(t)}{dt}$$ ……(23)

Dan sinyal keluaran PD *Fuzzy* adalah sebagai berikut :

$$U = Kp.E + Kd.CE$$ ……(24)

Dimana :

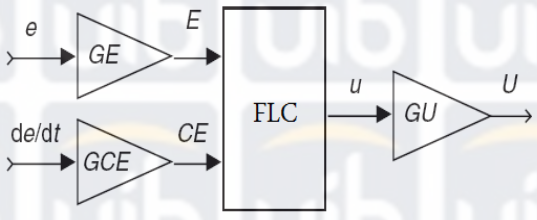
E = *error*,

CE = *change of error*,

Kp = penguatan sinyal proporsional,

Td = waktu derivatif,

Kd = konstanta derivatif.



Gambar 2.17 Blok Diagram FPD *Controller* [12]

Sinyal kontrol *U(t)* adalah fungsi non linier dari *"error"* dan *"change of error"*[12]. Maka sinyal kendali FPD adalah sebagai berikut :

$$U(t) = f\,(GE.e(t),\ GCE.e'(t)).GU \quad \text{.....(25)}$$

Dimana *f* merepresentasikan dari algoritma control. Pendekatan linier harus didapatkan dengan pemilihan yang paling baik, yaitu:

$$f\left(GE.e(t),\ GCE.e'(t)\right) \approx \quad GE.e(t) + GCE.e'(t) \text{.....(26)}$$

Kemudian,

$$U(t) = (GE.e(t) + GCE.e'(t)).GU \quad \text{.....(27)}$$

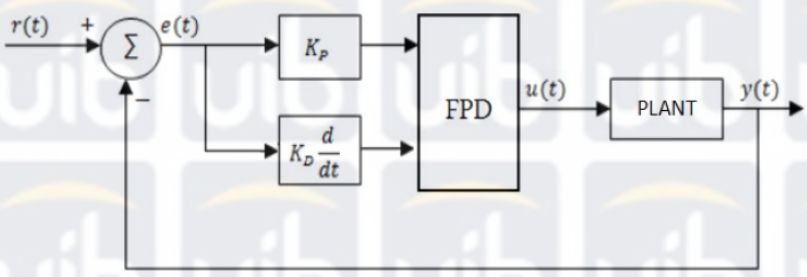
$$U(t) = GE.GU.\ \left(e(t) + \frac{GCE}{GE}\ .e'(t)\right) \text{.....(28)}$$

Ketika membandingkan persamaan ini dengan sinyal kontrol dari *crisp* PD *controller* hubungan antara penguatan sinyal PD *controller* dan FPD *controller* adalah sebagai berikut [12]:

$$GE.GU = Kp \text{.....(29)}$$

$$\frac{GCE}{GE} = Td \text{.....(30)}$$

Mengakibatkan, nilai parameter-parameter linier FPD *controller* dapat ditentukan dari penyetelan PD *controller*. Berikut pada Gambar 2.18 dapat dilihat blok diagram dari sistem kontrol dengan FPD *controller* :



Gambar 2.18 Blok Diagram dari Sistem Kontrol FPD [12]

Apabila *Fuzzy Logic* ini diaplikasikan untuk pengaturan gerak motor DC, maka metode yang digunakan adalah *Fuzzy Proportional Derivative* (FPD) dimana respon akan mengikuti sinyal yang diberikan seperti kontrol PD. Meskipun kontrol ini dibangun dengan PD tetapi tetap menggunakan model *fuzzy rule* [12].

Penggunaan metode *Fuzzy Proportional Derivative* (FPD) sebagai sistem kendali pada *plant* dimana aplikasinya untuk pengaturan pergerakan motor DC dengan satu input atau satu sensor. Pada *Fuzzy Logic* input yang digunakan minimal dua nilai input, yang mana input tersebut akan diolah pada masing-masing *membership function.*

Pada *plant* yang hanya menggunakan satu input atau satu sensor, untuk pengaturan gerak motor DC maka diperlukan satu input tambahan untuk memenuhi aturan penerapan *Fuzzy Logic*, maka parameter-parameter input untuk menghasilkan sinyal kontrol adalah sinyal *error (err)* dan sinyal *delta error (Δ err)* dimana parameter-parameter tersebut ditentukan dari penyetelan PD kontrol.