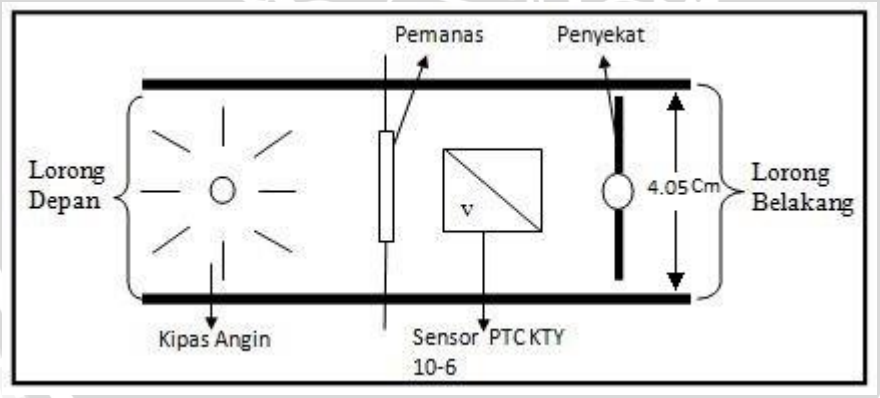
BAB II
Tinjauan Pustaka

Pada Bab II ini akan dijelaskan teori penunjang yang digunakan dalam penulisan ini, dengan tujuan untuk memudahkan dalam memahami cara kerja rangkaian maupun dasar-dasar perencanaan alat ini. Teori penunjang yang akan dijelaskan dalam Bab II ini adalah:

- *Plant* 73412
- STM32F4 *Discovery*
- Sinyal *Pulse Width Modulation* (PWM)
- *Driver* L298
- Sensor Suhu KTY 10-6
- Kontroler

2.1 *Plant* 73412

Plant 73412 merupakan prototipe *plant* suhu. Dalam perubahan suhu di *plant* 73412 terdapat beberapa elemen yang mempengaruhi terdiri dari pemanas, kipas angin, dan penyekat sebagaimana ditunjukkan Gambar 2.1.



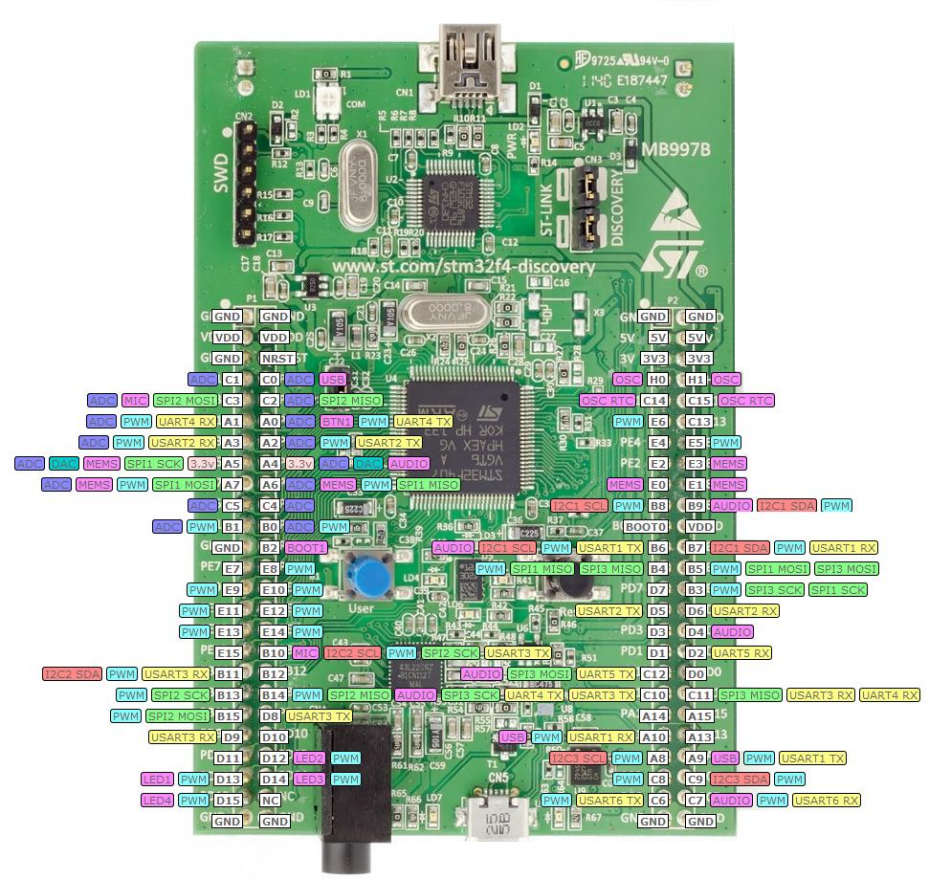
Gambar 2.1 *Plant* 73412

Pemanas berupa lampu halogen dengan tegangan *input* maksimal 12V. Di sebelah pemanas terdapat sensor suhu *Posistive Temperature Coeficient* (PTC) dengan jenis KTY 10-6 yang bertujuan untuk mengukur udara panas yang mengalir pada lorong. Kipas angin yang terletak pada lorong muka bertujuan untuk menyedot udara dari luar, kecepatan kipas angin dapat diatur dalam skala 1-10. Pada lorong ujung akhir dipasang sebuah penyekat yang dapat diatur posisi kemiringannya. Dengan mengatur sudut kemiringan

pintu pengatur tersebut aliran udara panas yang keluar dapat diperbesar maupun diperkecil.

2.2 STM32F4 *Discovery*

STM32F4 *Discovery* adalah modul mikrokontroler produksi dari ST Microelectronics. Core dari Mikrokontroler ini adalah ARM Cortex™-M4 yang merupakan core mikrokontroler 32-bit. Mikrokontroler ini terdiri dari pin *Input Output* (I/O) berjumlah 82 pin sebagaimana ditunjukkan Gambar 2.2.



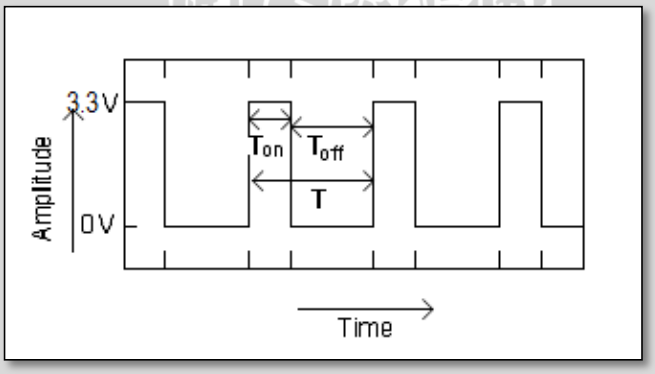
Gambar 2.2 Pin *function* STM32F4 *Discovery*

Salah satu kelebihan STM32F4 *Discovery* adalah kemampuan *floating point* secara *hardware*. Beberapa operasi matematika *floating point* 32bit dapat dilakukan oleh microkontroler ini hanya dengan 1 siklus mesin (1 cycle). Komputasi *floating point* biasa dipakai untuk pemrosesan sinyal digital yang memerlukan keakuratan yang tinggi atau bisa juga dipakai untuk pemrosesan audio atau *image* digital secara lebih mudah. Pada STM32F4 *Discovery* tertanam prosesor ARM Cortex™-M4 seri STM32F407VGT dengan *clock* maksimal 168MHz. Berikut spesifikasi STM32f4 *Discovery*:

- Core : Cortex-M4F
- Microcontroller : STM32F407VGT6
- Memory :1MB Flash
- RAM :192KB SRAM
- Package : LQFP100
- I/O pins : 82
- Timers(16-bit) : 12
- Advanced Control Timers : 2
- General Purpose Timers : 10
- Basic Timers : 2
- PWM Channels : 6
- ADC(12-bit) : 3 (16 channels)
- I2C(TWI) : 3
- USART : 4
- SPI : 3 full duplex
- DMA : 2 (8 channels each)
- USB : 1 (2.0 full speed)
- CAN : 2 (2.0 active)
- uP Supply Voltage : 1.8-3.6V

2.3 Sinyal *Pulse Width Modulation* (PWM)

Sinyal PWM adalah metode yang dapat digunakan untuk mengontrol lebar pulsa keluaran pin mikrokontroler. *Duty cycle* merupakan perbandingan lebar pulsa saat T_{on} (lebar pulsa saat logika tinggi) dengan T (periode pulsa). Sinyal PWM secara umum dapat dilihat dalam Gambar 2.3.



Gambar 2.3 Sinyal PWM

Pada sinyal PWM, frekuensi sinyal konstan sedangkan *duty cycle* bervariasi dari 0%-100%. Persamaan untuk perhitungan *duty cycle* ditunjukkan pada Persamaan 2.1 dengan T_{on} adalah periode logika tinggi, dan T adalah periode keseluruhan.

$$Duty\ Cycle = \frac{T_{on}}{T} x\ 100\% \quad (2.1)$$

dimana:

T_{on} = lebar pulsa saat logika tinggi

T = periode pulsa

Sedangkan frekuensinya dapat ditentukan dengan Persamaan 2.2 berikut:

$$fOCn = \frac{fclk\ \ I/O}{N.1000} \quad (2.2)$$

dimana:
$fOCn$ = frekuensi PWM
$fclk\ \ I/O$ = frekuensi I/O
N = *prescaler factor*

2.4 *Analog Digital Converter* (ADC)

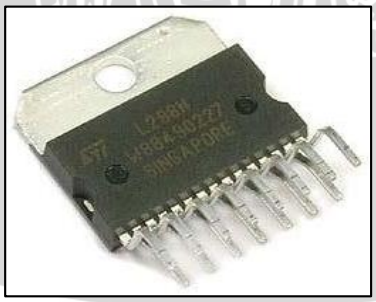
ADC merupakan suatu piranti yang digunakan untuk mengubah tegangan analog menjadi suatu data digital. Resolusi konverter *analog digital* mengacu pada jumlah bit dalam keluaran nilai digital. Persamaan untuk menghitung nilai bit ADC ditunjukkan pada Persamaan 2.3.

$$ADC = \frac{Vin\ .\ 4096}{Vref} \quad (2.3)$$

dimana:
ADC = nilai bit ADC
Vin = tegangan *analog* sebenarnya
$Vref$ = tegangan refrensi ADC

2.5 *Driver* L298

Driver L298 menggunakan menggunakan IC L298 berbasis *H-Bridge*. Pada skripsi *driver* L298 digunakan untuk memperbesar daya karena Pin I/O mikrokontroler tidak mampu menangani beban aktuator *plant* 73412. Kemampuan maksimal *driver* L298 menangani beban hingga 4A pada tegangan 6V – 46V. *Driver* L298 ditunjukkan Gambar 2.4.



Gambar 2.4 IC *Driver* L298

Aktuator pada *plant* 73412 tidak dapat dikontrol secara langsung oleh mikrokontroler, karena kebutuhan daya listrik yang cukup besar sedangkan daya keluaran pada mikrokontroler sangat kecil. Pada *driver* L298 pengendalian tegangan keluaran *driver* menggunakan metode PWM.

2.6 Sensor Suhu KTY10-6

Sensor Suhu KTY10-6 adalah jenis sensor yang mengkonversi perubahan suhu yang naik akan mengubah resistansi yang dimiliki sensor. Sensor KTY10-6 ditunjukkan Gambar 2.5.

Gambar 2.5 Sensor KTY 10-6 (Datasheet KTY 10-6)

KTY 10-6 dirancang untuk pengukuran gas dan cairan dalam kisaran suhu -50°C hingga +150°C, dengan toleransi resistansi 3%.

2.7 Kontroler

Kontroler dalam sebuah sistem kontrol mempunyai kontribusi yang besar terhadap perilaku sistem. Hal itu disebabkan oleh tidak dapat dirubahnya komponen penyusun sistem tersebut. Artinya, karakteristik *plant* harus diterima sebagaimana adanya, perubahan perilaku sistem hanya dapat dilakukan melalui penambahan kontroler.

Salah satu fungsi kontroler adalah mengurangi sinyal *error*, sinyal *error* adalah perbedaan nilai *setpoint* dengan nilai *output plant*. *Setpoint* adalah nilai referensi atau nilai yang diinginkan, sedangkan *output plant* adalah nilai aktual yang terukur pada *output plant*. Semakin kecil nilai sinyal *error* maka kinerja sistem kontrol dinilai semakin baik.

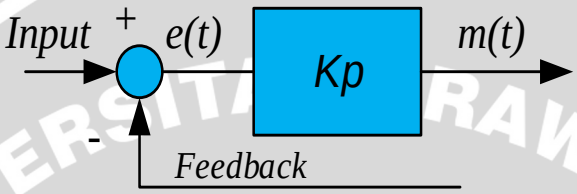
Prinsip kerja kontroler adalah membandingkan nilai *output plant* dengan nilai *setpoint*, menentukan nilai kesalahan dan akhirnya menghasilkan sinyal kontrol untuk meminimalkan *error*(Ogata K, 1995).

2.7.1 Kontroler *Proportional Integral Derivative* (PID)

Kontroler PID terbentuk dari tiga buah kontroler, yaitu kontroler *Proportional*, *Integral* dan *Derivative*. Ketiga kontroler bekerja secara pararel menjadi kontroler PID. *Output* kontroler PID merupakan penjumlahan dari *output* masing-masing kontroler.

2.7.1.1 Kontroler *Proportional*

Kontroler *proportional* memiliki *output* yang besarnya sebanding dengan besarnya sinyal *error*. *Output* kontroler merupakan perkalian antara penguatan proporsional dengan sinyal *error*. Gambar 2.6 menunjukkan diagram blok kontroler *proportional* dan Persamaan 2.4 menunjukkan hubungan antara *output* kontroler dengan sinyal *error*.



Gambar 2.6 Diagram Blok Kontroler *Proportional* (Ogata K.,1997)

$$m(t) = Kp\, e(t) \qquad (2.4)$$

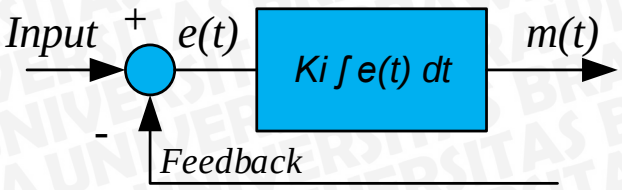
dimana:

Kp = adalah penguatan proporsional
$e(t)$ = sinyal *error*
$m(t)$ = *output* kontroler

Penambahan Kp akan mempercepat kecepatan respon *transient* dan mengurangi kesalahan keadaan mantap.

2.7.1.2 Kontroler *Integral*

Kontroler *integral* memiliki karakteristik seperti sebuah operasi *integral*, *output* kontroler dipengaruhi oleh perubahan yang sebanding dengan perubahan nilai sinyal *error*. *Output* kontroler merupakan penjumlahan terus menerus dari perubahan sinyal *error*. Gambar 2.7 menunjukkan diagram blok kontroler *proportional* dan Persamaan 2.5 dan Persamaan 2.6 menunjukkan hubungan antara *output* kontroler dengan sinyal *error*.



Gambar 2.7 Diagram Blok Kontroler Integral (Ogata K., 1997)

$$\frac{dm(t)}{dt} = Ki\, e(t) \quad (2.5)$$

$$m(t) = Ki \int e(t)\, dt \quad (2.6)$$

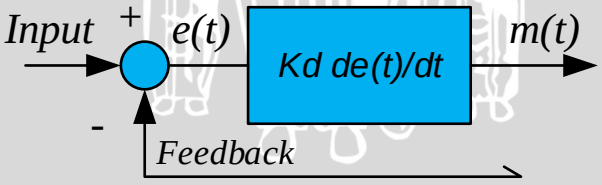
dimana:

Ki = adalah penguatan integral
$e(t)$ = sinyal *error*
$m(t)$ = *output* kontroler

Aksi kontrol *integral* digunakan untuk menghilangkan sinyal *error* dalam keadaan mantap.

2.7.1.3 Kontroler *Derivative*

Kontroler *derivative* memiliki sifat seperti suatu operasi *derivative*. Perubahan yang mendadak pada masukkan kontroler mengakibatkan perubahan yang sangat besar dan cepat. Kontroler ini tidak akan menghasilkan *output* saat sinyal *error* konstan sehingga tidak akan mempengaruhi keadaan mantap. Gambar 2.8 menunjukkan diagram blok kontroler *derivative* dan Persamaan 2.7 menunjukkan hubungan antara *output* kontroler dengan sinyal *error*.



Gambar 2.8 Diagram Blok kontroler *Derivative* (Ogata K., 1997)

$$m(t) = Kd\, \frac{de(t)}{dt} \quad (2.7)$$

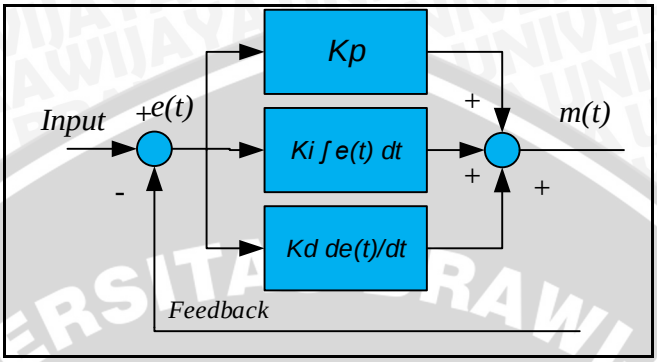
dimana:

Kd = adalah penguatan *derivative*
$e(t)$ = sinyal *error*
$m(t)$ = *output* kontroler

Kontroler ini digunakan untuk memperbaiki atau mempercepat respon *transient*.

2.7.1.4 Kontroler PID

Gabungan aksi kontrol *proportional*, *integral*, dan *derivative* yang terlihat dalam Gambar 2.9 mempunyai keunggulan dapat saling menutupi kekurangan dan kelebihan dari masing-masing kontroler. Persamaan kontroler PID ini dapat dinyatakan sebagai berikut (Persamaan 2.8):



Gambar 2.9 Diagram Blok Kontroler PID (Ogata K., 1997)

$$m(t) = Kp\,.e(t) + Ki.\int e(t)dt + Kd\frac{de(t)}{dt} \qquad (2.8)$$

dimana:

Kp = adalah penguatan *proportional*
Ki = adalah penguatan *integral*
Kd = adalah penguatan *derivative*
$e(t)$ = sinyal *error*
$m(t)$ = *output* kontroler

2.7.2 *Fuzzy Logic*

Fuzzy secara harfiah berarti samar, sedangkan kebalikannya dalam hal ini adalah C*risp* yang secara harfiah berarti tegas. Dalam kehidupan sehari-hari nilai samar lebih akrab daripada nilai tegas. Suhu tertentu biasa dinyatakan sebagai panas, agak panas, atau sangat dingin daripada dinyatakan dalam nilai terukur tertentu.

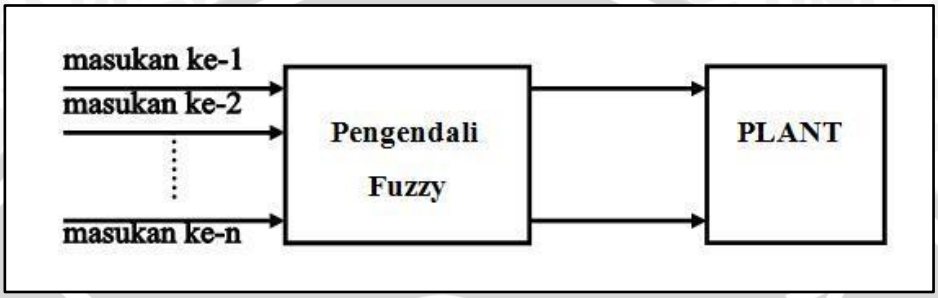
Tahun 1965 L.A. Zadeh memodifikasi teori himpunan yang disebut himpunan kabur (*Fuzzy Set*). Himpunan *fuzzy* didasarkan pada gagasan untuk memperluas jangkauan fungsi karakteristik sehingga fungsi tersebut akan mencakup bilangan real pada interval [0, 1]. Nilai keanggotaannya menunjukkan bahwa suatu nilai dalam semesta pembicaraan tidak hanya berada pada 0 atau 1, namun juga nilai yang terletak diantaranya. Dengan kata lain nilai kebenaran suatu hal tidak hanya bernilai benar atau salah. Nilai 0 menunjukkan salah, nilai 1 menunjukkan benar dan masih ada nilai-nilai yang terletak diantaranya.

Sejak tahun 1985 pengendalian berbasis *fuzzy logic* mengalami perkembangan pesat, terutama dalam hubungannya dengan penyelesaian masalah kendali yang bersifat

tak linier, sulit dimodelkan, berubah karakteristiknya terhadap waktu (*time varying*) dan kompleks.

2.7.2.1 Struktur Dasar *Fuzzy Logic*

Dalam sistem pengendalian dengan *fuzzy logic* dilibatkan suatu blok pengendali yang menerima satu atau lebih masukan dan mengeluarkan satu atau lebih keluaran ke *plant* atau blok lain sebagaimana ditunjukkan dalam Gambar 2.10.



Gambar 2.10 Kontroler*Fuzzy* (J.Ross, 1991)

Komponen utama penyusun *fuzzy logic* adalah unit fuzzifikasi, *fuzzy inference*, dan defuzzifikasi.

2.7.2.2 Fungsi Keanggotaan

Fungsi keanggotaan menotasikan nilai kebenaran anggota-anggota himpunan *fuzzy*. Interval nilai yang digunakan untuk menentukan fungsi keanggotaan, yaitu nol dan satu. Tiap fungsi keanggotaan memetakan elemen himpunan *crisp* ke semesta himpunan *fuzzy*.

Suatu himpunan *fuzzy* A dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan, μ_A yang harganya berada dalam interval [0, 1] (Kuswadi, 2000:27). Secara matematika hal ini dinyatakan dengan:

$$\mu_A : U \rightarrow [0,1] \qquad (2.9)$$

2.7.2.3 Fuzzifikasi

Proses fuzzifikasi merupakan proses untuk mengubah variabel non *fuzzy* (variabel numerik) menjadi variabel *fuzzy* (variabel linguistik). Nilai masukan-masukan yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh *fuzzy logic* harus diubah terlebih dahulu ke dalam 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* pula. Proses ini disebut fuzzifikasi (Yan, 1994). Proses fuzzifikasi diekspresikan sebagai berikut:

$$x = fuzzyfier(xo) \quad (2.10)$$

dengan:

x_0 = nilai *crisp* variabel masukan
x = himpunan *fuzzy* variabel yang terdefinisi
fuzzifier = operator fuzzifikasi yang memetakan himpunan *crisp* ke himpunan *fuzzy*.

Pedoman memilih fungsi keanggotaan untuk proses fuzzifikasi, menurut Jun Yan, menggunakan:

1. Himpunan *fuzzy* dengan distribusi simetris.
2. Gunakan himpunan *fuzzy* dengan jumlah ganjil, berkaitan erat dengan jumlah kaidah (*rules*).
3. Mengatur himpunan *fuzzy* agar saling menumpuk.

2.7.2.4 *Fuzzy Rule*

Fuzzy Rule adalah bagian yang menggambarkan dinamika suatu sistem terhadap masukan yang dikarakteristikan oleh sekumpulan variabel-variabel linguistik dan berbasis pengetahuan seorang operator ahli. Pernyataan tersebut umumnya dinyatakan oleh suatu pernyataan bersyarat.

Dalam pengendali berbasis *fuzzy*, aturan pengendalian *fuzzy* berbentuk aturan " IF – THEN". Untuk sebuah sistem *Multi Input Single Output* (MISO) basis aturan pengendalian *fuzzy* berbentuk seperti berikut ini,

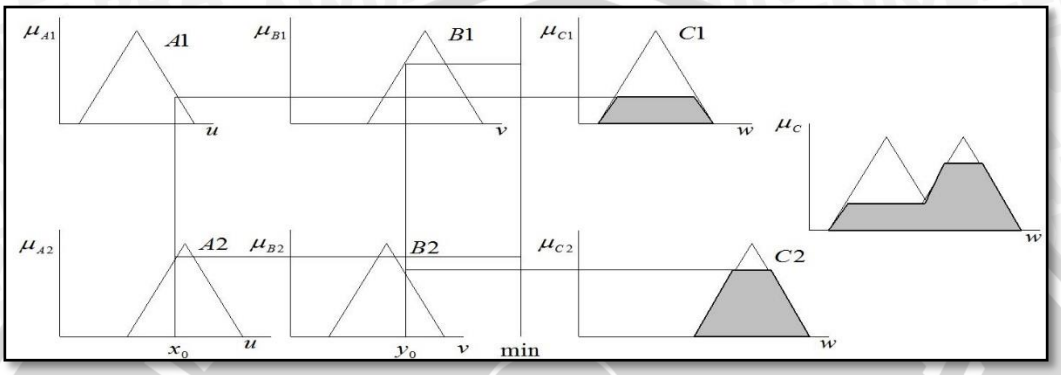
Rule 1 IF X is A_1 AND Y is B_1 THEN Z is C_1
Rule 2 IF X is A_2 AND Y is B_2 THEN Z is C_2
. .
. .
. .
Rule n IF X is A_n AND Y is B_n THEN Z is C_n

Dengan X, Y, Z merupakan variabel linguistik, dimana X dan Y merupakan variabel masukan, dan Z merupakan variabel keluaran sistem. A_n, B_n, dan C_n merupakan nilai linguistik dari X, Y, dan Z (Yan, 1993).

2.7.2.5 Metode Inferensi *MAX-MIN*

Metode inferensi merupakan proses untuk mendapatkan keluaran dari suatu kondisi masukan dengan mengikuti aturan-aturan yang telah ditetapkan. Keputusan yang didapatkan pada proses ini masih dalam bentuk *fuzzy* yaitu derajat keanggotaan keluaran.

Pada metode *Max–Min* aturan operasi minimum Mamdani digunakan untuk implikasi *fuzzy*. Lebih jelas metode ini dideskripsikan dalam Gambar 2.11.



Gambar 2.11 Inferensi *Fuzzy* dengan metode *MAX-MIN* (Yan, 1994)

Sebagai contoh , terdapat dua basis kaidah atur *fuzzy*, yaitu:
R_1 : Jika x adalah A_1 dan y adalah B_1 maka z adalah C_1
R_2 : Jika x adalah A_2 dan y adalah B_2 maka z adalah C_2

$$\mu_{C'} = \bigcup_{1}^{n} \alpha_i \wedge \mu ci \qquad (2.11)$$

dengan $\alpha_i = \mu_{Ai}(x_0) \wedge \mu_{Bi}(y_0)$

Pada metode penalaran MAX-MIN fungsi keanggotaan konsekuen dinyatakan dengan

$$\mu c_{1'}(W) = \mu_{c'1} \vee \mu_{c'2} = [\alpha_1 \wedge \mu_{c1}(w)] \vee [\alpha_2 \wedge \mu_{c2}(w)] \qquad (2.12)$$

dimana

$$\alpha_1 = \mu_{A1}(x_0) \wedge \mu_{B1}(y_0) \qquad (2.13)$$

$$\alpha_2 = \mu_{A2}(x_0) \wedge \mu_{B2}(y_0) \qquad (2.14)$$

2.7.2.6 Metode Defuzzifikasi *Weighted Average*

Defuzzifikasi adalah proses untuk mendapatkan nilai numerik dari data *fuzzy* yang dihasilkan dari proses inferensi (Yan, 1994). Proses defuzzifikasi dinyatakan sebagai berikut :

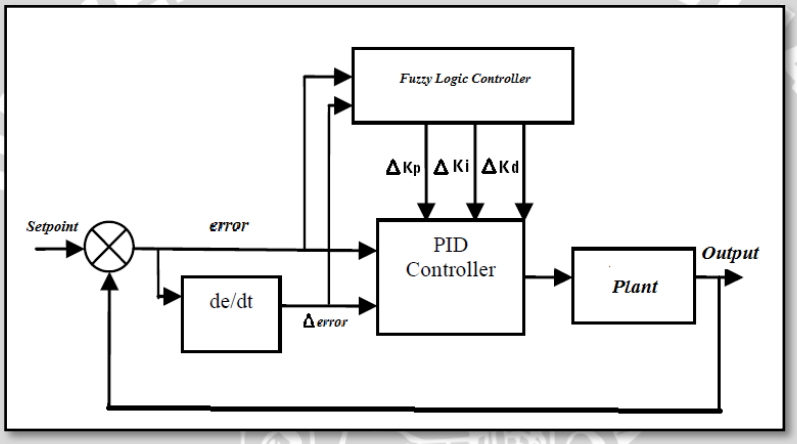
$$y_0 = defuzzifier\ (y) \qquad (2.15)$$

dengan:

y	= aksi kontrol *fuzzy*
y_0	= aksi kontrol *crisp*
defuzzifier	= operator defuzzifikasi

2.7.3 *Fuzzy Self-tuning PID*

Fuzzy self-tuning PID terdiri dari kontroler PID yang parameternya berubah secara *realtime* dan *fuzzy logic* yang akan menghasilkan perubahan parameternya. Kontroler PID konvensional kurang baik diterapkan pada *sistem* yang mengalami gangguan yang tidak dapat diprediksi. Parameter PID pada kontroler *fuzzy self-tuning PID* diatur melalui logika *fuzzy* menghasilkan parameter yang lebih baik karena terus diperbaharui (*realtime*), *rule* dari *fuzzy logic* mampu memetakan secara non linier *input* ke *output*. Kontroler *fuzzy self-tuning PID* mendapatkan *input* berupa *error* dan Δ*error* (turunan dari *error / error rate*). Sedangkan *output*nya berupa Δ*Kp*, Δ*Ki*, dan Δ*Kd*. Untuk lebih jelasnya struktur kontroler *fuzzy self-tuning PID* ditunjukkan Gambar 2.12.



Gambar 2.12 Diagram Blok *Fuzzy Self-Tuning PID* (Wei, 2010)

2.7.6.1 *Rule Fuzzy Self-Tuning PID*

Dalam menyusun *rule* didasarkan pada pertimbangan karakteristik kontroler PID dan *output* sistem. Perubahan parameter pada kontroler PID menghasilkan karakteristik keluaran sistem (*rise time*, *overshoot*, *settling time*, *error steady state*) yang berbeda-beda. Berikut pengaruh masing parameter-parameter ditunjukkan Tabel 2.1:

Tabel 2.1 Parameter PID dan Pengaruhnya dalam Sistem Kontrol (Hongbo,2009)

Parameter Kontroler PID	*Rise Time*	*Over Shoot*	*Settling Time*	*Error Steady State*
Kp	Berkurang	Bertambah	Berubah sedikit	Berkurang
Ki	Berkurang	Bertambah	Bertambah	Menghilangkan
Kd	Berubah sedikit	Berkurang	Berkurang	Tak Berubah

Dengan mengetahui aksi setiap parameter kontroler PID, maka dapat dirumuskan *rule* untuk setiap parameter ΔKp, ΔKi, dan ΔKd berdasarkan *error* dan $\Delta error$. Proses *tuning* parameter akan memenuhi kriteria (Wei,2010):

1. Ketika nilai *error* besar, untuk mempercepat respon sistem maka membutuhkan *Kp* yang besar, *Kd* yang kecil, dan disaat bersamaan untuk menghindari *overshoot* diperlukan *Ki* yang kecil
2. Ketika nilai *error* dan $\Delta error$ sedang, untuk menghindari *overshoot Ki* bernilai kecil, dan nilai *Kp* dan *Kd* sedang untuk tetap menjaga kecepatan respon sistem.
3. Ketika nilai *error* kecil , untuk menjaga performa *steady state* tetap baik nilai *Kp* dan *Ki* seharusnya besar, dan disaat yang bersamaan untuk menjaga sistem supaya tidak berosilasi maka nilai *Kd* harus dapat menyesuaikan.
4. Nilai *Kd* dipengaruhi oleh $\Delta error$, ketika $\Delta error$ kecil maka *Kd* besar, sebaliknya ketika $\Delta error$ besar *Kd* kecil.

Dan untuk mengakhiri proses *self-tuning PID*, parameter PID akan lakukan proses *deblurrying* seperti yang ditunjukkan Persamaan 2.16, Persamaan 2.17 dan Persamaan 2.18(Wei, 2010) :

$$Kp = Kp^* + \Delta Kp \quad (2.16)$$

$$Ki = Ki^* + \Delta Ki \quad (2.17)$$

$$Kd = Kd^* + \Delta Kd \quad (2.18)$$

dengan:

Kp	= parameter *Kp* baru
Kp^*	= parameter *Kp* lama
ΔKp	= perubahan parameter *Kp*
Ki	= parameter *Ki* baru
Ki^*	= parameter *Ki* lama
ΔKi	= perubahan parameter *Ki*
Kd	= parameter *Kd* baru
Kd^*	= parameter *Kd* lama
ΔKd	= perubahan parameter *Kd*