BAB II

TINJAUAN PUSTAKA

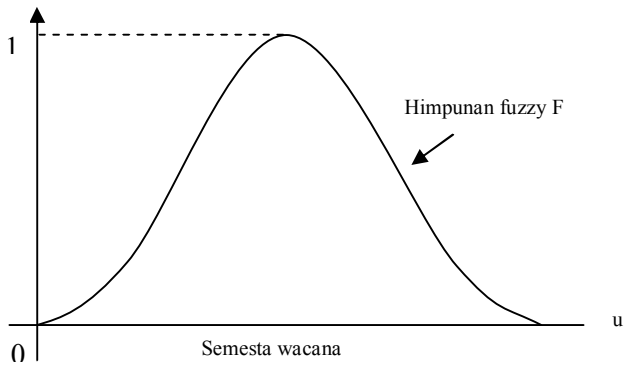
Pada bab ini diuraikan beberapa landasan teori yang dipergunakan untuk menyelesaikan masalah dan pengerjaan pada penelitian ini yang mencakup, himpunan *fuzzy*, sistem inferensi *fuzzy*, FLC (*fuzzy logic controller*), *Fuzzy C-Means*, ANFIS (*Adaptive Neuro Fuzzy Inference System*), mikrokontroler ATMega16, *Visual Basic* 2010, *weighfeeder conveyor*, *load cell*, dan pengkondisi sinyal. Landasan teori yang digunakan diambil dari jurnal, dan buku yang didapat melalui perpustakaan atau internet.

2.1 Teori Himpunan *Fuzzy*

Himpunan *fuzzy* atau himpunan kabur (*fuzzy set*) adalah generalisasi konsep himpunan biasa. Untuk semesta wacana (*universe of discourse*) U, himpunan *fuzzy* ditentukan oleh fungsi keanggotaan yang memetakan anggota U ke rentang keanggotaan dalam interval {0,1}[3].

2.1.1 Notasi *Fuzzy*

Misalkan U merupakan kumpulan objek yang dinotasikan dengan (u), U disebut semesta dan u menyatakan elemen generik dari U. Terlihat pada gambar 2.1 terdapat suatu himpunan *fuzzy* F di dalam semesta U dikarakteristikkan dengan fungsi keanggotaan μ_F yang bernilai dalam interval {0,1} atau:$\mu_F : U \rightarrow \{0,1\}$ [3].



Gambar 2.1 Himpunan *fuzzy*

Himpunan pendukung (*support set*) dari himpunan *fuzzy* F adalah himpunan tegas (*crisp set*) dari semua titik u di dalam U sedemikian sehingga μ_F (u) > 0.

Elemen u di dalam U yang mana μ_F (u) = 0,5 disebut titik persilangan (*crossover point*).

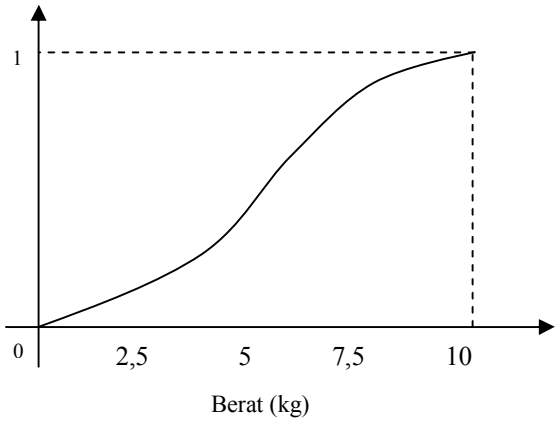
Suatu himpunan *fuzzy* yang pendukungnya merupakan satu titik di dalam U disebut *fuzzy singleton*.

Himpunan terpotong α (α–*cut-set*) dari himpunan *fuzzy* F disebut Fα, adalah himpunan tegas dari semua titik u di dalam U sedemikian sehingga μ_F (u) ≥α [3].

2.1.2 Domain Himpunan *Fuzzy*

Domain himpunan *fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan. Domain merupakan himpunan bilangan riil yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif atau negatif.

Biasanya, domain memiliki batas atas dan bawah. Namun, pada konsep *fuzzy* bisa jadi domain ini bersifat *open ended*. Gambar 2.2 berikut merupakan contoh himpunan *fuzzy* berat yang memiliki domain antara 0 kg sampai 10 kg.



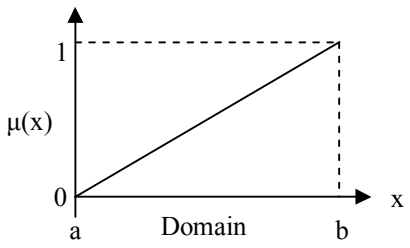
Gambar 2.2 Himpunan *fuzzy* berat dalam kg

2.1.3 Fungsi Keanggotaan

Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai keanggotaannya. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Beberapa fungsi yang ada antara lain:

1. Fungsi Linier

   Pada representasi linier, permukaan digambarkan sebagai suatu garis lurus. Ada dua keadaan himpunan *fuzzy* linier yang memiliki parameter a,b yaitu, yang pertama kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan μ(x) nol [0] bergerak kekanan menuju nilai domain yang memiliki derajat keanggotaan μ(x) tinggi, seperti terlihat pada gambar 2.3.

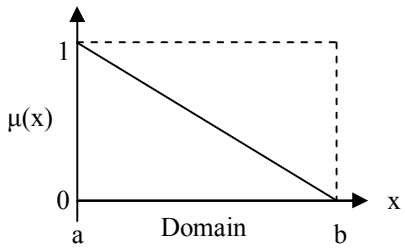


Gambar 2.3 Representasi kurva linier naik

Fungsi keanggotaan linier naik didefinisikan sebagai berikut:

$$\mu(x) = \begin{cases} 0, & x < a \\ (x-a)/(b-a), & a \le x \le b \\ 1, & x \ge b \end{cases} \quad (2.1)$$

Yang kedua merupakan kebalikan yang pertama, garis lurus dimulai dari nilai domain dengan derajat keanggotaan μ(x) tertinggi sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan μ(x) lebih rendah, seperti terlihat pada gambar 2.4.



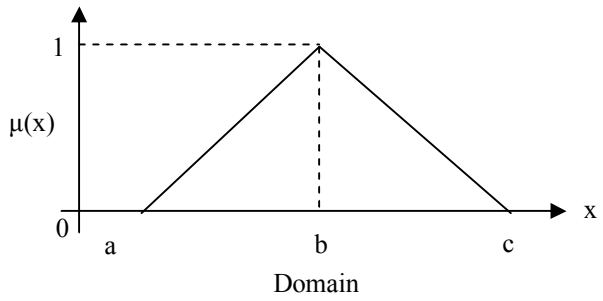
Gambar 2.4 Representasi kurva linier turun

Fungsi keanggotaan linier turun didefinisikan sebagai berikut:

$$\mu(x) = \begin{cases} 1, & x \le a \\ (b-x)/(b-a), & a \le x \le b \\ 0, & x \ge b \end{cases} \quad (2.2)$$

2. Fungsi Segitiga (*Triangular*)

Kurva segitiga pada dasarnya merupakan gabungan antara dua garis (linier) yang memiliki parameter a,b,c seperti terlihat pada gambar 2.5.



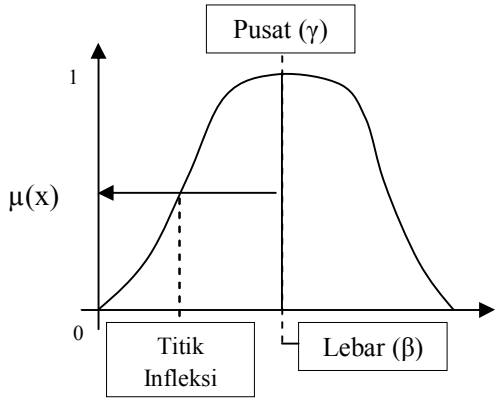
Gambar 2.5 Representasi kurva segitiga

Fungsi keanggotaan segitiga didefinisikan sebagai berikut:

$$\mu(x) = \begin{cases} 0, & x \leq a \; atau \; x \geq c \\ (x-a/(b-a), & a \leq x \leq b \\ (b-x)/(c-b), & b \leq x \leq c \end{cases} \qquad (2.3)$$

3. Fungsi Phi

Kurva phi berbentuk lonceng/bel mempunyai dua parameter yaitu β dan γ seperti terlihat pada gambar 2.6. Derajat keanggotaan bernilai satu pada pusat domain parameter γ, dan parameter β adalah lebar kurva pada titik persilangan.



Gambar 2.6 Representasi kurva phi

Fungsi keanggotaan phi didefinisikan sebagai berikut:

$$\mu(x,\beta,\gamma) = \begin{cases} S(x;\gamma-\beta,\gamma-\beta/2,\gamma), & x \leq \gamma \\ 1 - S(x;\gamma,\gamma+\beta/2,\gamma+\beta), & x \geq \gamma \end{cases} \quad (2.4)$$

2.1.4 Operator *Fuzzy*

Operator *fuzzy* memiliki dua model yaitu; operator dasar yang dikemukakan oleh Zadeh, dan operator alternatif yang dikembangkan dengan menggunakan konsep transformasi aritmatika ataupun fungsi yang lebih kompleks[4].

Pada operator-operator dasar Zadeh terdapat beberapa operasi yang didefinisikan secara khusus untuk mengkombinasi dan memodifikasi himpunan *fuzzy*. Nilai keanggotaan sebagai hasil dari operasi dua himpunan sering dikenal dengan nama *fire strength* atau α-predikat. Ada tiga operator dasar yang diciptakan oleh Zadeh, yaitu:

1. Operator AND

   Operator ini berhubungan dengan operasi interseksi pada himpunan. α-predikat diperoleh dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan-himpunan yang bersangkutan. Berikut definisi fungsi operator AND:

   $$\mu_{A\cap B} = \min(\mu_A(x), \mu_A(y)) \quad (2.5)$$

2. Operator OR

   Operator ini berhubungan dengan operasi union pada himpunan. α- predikat diperoleh dengan mengambil nilai keanggotaan terbesar antar elemen pada himpunan-himpunan yang bersangkutan. Berikut definisi fungsi operator OR:

   $$\mu_{A\cup B} = \max(\mu_A(x), \mu_A(y)) \quad (2.6)$$

3. Operator NOT

   Operator ini berhubungan dengan operasi komplemen pada himpunan. α-predikat diperoleh dengan mengurangkan nilai keanggotaan elemen pada himpunan yang bersangkutan dari 1. Berikut definisi fungsi AND:

   $$\mu_{A'} = 1 - \mu_A(x) \quad (2.7)$$

2.1.5 Fungsi Implikasi

Tiap-tiap aturan (proposisi) pada basis pengetahuan *fuzzy* akan berhubungan dengan suatu relasi *fuzzy*. Bentuk umum dari aturan yang digunakan dalam fungsi implikasi adalah:IF x is A THEN y is B, dengan x dan y adalah skalar, dan A dan B adalah himpunan *fuzzy*. Proposisi yang mengikuti IF disebut

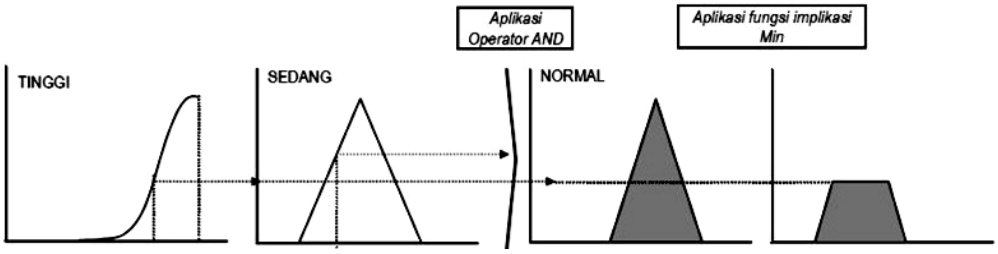
sebagai anteseden, sedangkan proposisi yang mengikuti THEN disebut sebagai konsekuen. Proposisi ini dapat diperluas dengan menggunakan operator *fuzzy*, seperti [5]:

IF (x_1 is A_1) • (x_2 is A_2) • (x_3 is A_3) • …. • (x_n is A_n) THEN y is B (2.8)

Dengan tanda • adalah operator *fuzzy* (seperti AND atau OR).

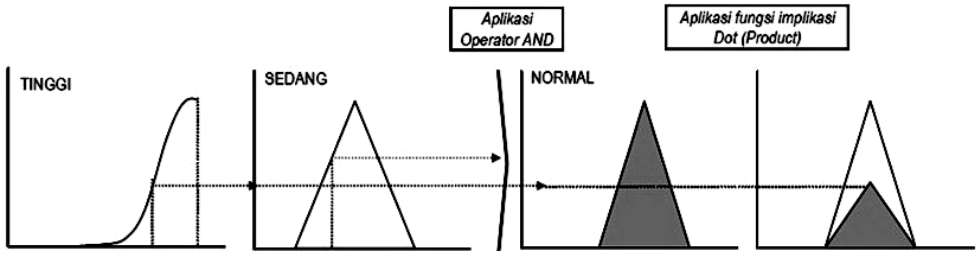
Secara umum, ada dua fungsi implikasi yang dapat digunakan, yaitu:

1. Min (*minimum*), fungsi ini akan memotong *output* himpunan *fuzzy*. Gambar 2.7 menunjukkan salah satu contoh penggunaan fungsi min.



Gambar 2.7 Fungsi implikasi Min[5]

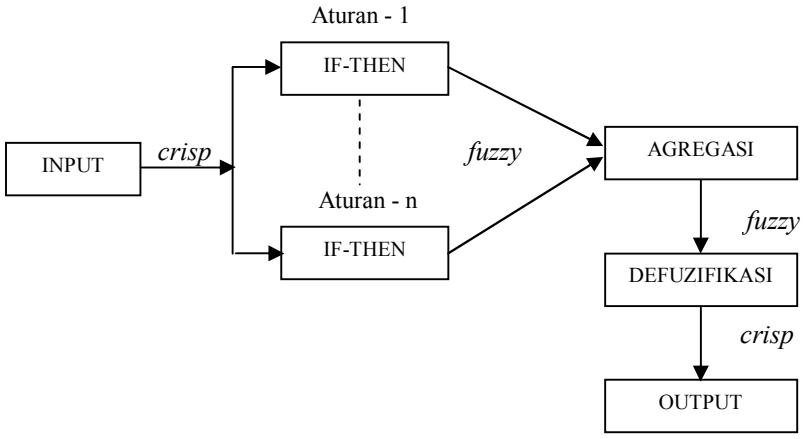
2. Dot (*product*), fungsi ini akan menskala *output* himpunan *fuzzy*. Gambar 2.8 menunjukkan salah satu contoh penggunaan fungsi dot.



Gambar 2.8 Fungsi implikasi DOT[5]

2.2 Sistem Inferensi *Fuzzy*

Sistem inferensi *fuzzy* (*Fuzzy Inference System*, FIS) merupakan suatu kerangka komputasi yang didasarkan pada teori himpunan *fuzzy*, aturan *fuzzy* berbentuk IF-THEN, dan penalaran *fuzzy*. Secara garis besar, diagram blok proses inferensi *fuzzy* terlihat pada gambar 2.9 [4].



Gambar 2.9 Diagram blok sistem inferensi *fuzzy*[4]

Pada gambar 2.9 dapat dijelaskan bahwa, sistem inferensi *fuzzy* menerima *inputcrisp* yang kemudian dikirim ke basis pengetahuan yang berisi n aturan *fuzzy* dalam bentuk IF-THEN. *Fire strength* atau α-predikat akan dicari pada setiap aturan. Apabila jumlah aturan lebih dari satu, maka akan dilakukan agregasi dari semua aturan. Selanjutnya, pada hasil agregasi akan dilakukan defuzifikasi untuk mendapatkan nilai *crisp* sebagai *output* sistem[4].

Model inferensi *fuzzy* yang umum digunakan, antara lain:

1. Model *fuzzy* Mamdani.
2. Model *fuzzy* Tsukamoto.
3. Model *fuzzy* Sugeno.

2.2.1 Penalaran *Fuzzy* Metode Sugeno

Sistem inferensi *fuzzy* metode Sugeno memiliki karakteristik yaitu konsekuen tidak merupakan himpunan *fuzzy*, namun merupakan suatu konstanta atau persamaan linier. Terdapat dua model yang secara umum dimiliki oleh metode Sugeno, yaitu[4]:

1. Model Sugeno orde-0

   IF (X_1 is A_1) • (X_2 is A_2) • (X_3 is A_3) • …. • (X_n is A_n) THEN Z is K (2.9)

   Dengan A_i adalah himpunan *fuzzy* ke-i sebagai anteseden (*input*), dan K adalah suatu konstanta (tegas) sebagai konsekuen (*output*). Tanda • adalah operator *fuzzy* (seperti AND atau OR).

2. Model Sugeno orde-1

   IF (X_1 is A_1) •…. • (X_n is A_n) THEN $Z = P_1*X_1+\ldots+P_n*X_n+Q$ (2.10)

   Dengan A_i adalah himpunan *fuzzy* ke-i sebagai anteseden (*input*), dan P_i adalah suatu konstanta (tegas) ke-i dan Q juga merupakan konstanta dalam konsekuen. Tanda • adalah operator *fuzzy* (seperti AND atau OR).

Untuk mendapatkan *output* pada penalaran *fuzzy* metode Sugeno hampir sama dengan penalaran Mamdani, yang memerlukan empat tahapan, yaitu[5]:

1. Pembentukan himpunan *fuzzy*

   Pada metode Sugeno variabel *input* merupakan himpunan *fuzzy*, sedangkan *output* berupa konstanta atau persamaan linier.

2. Aplikasi fungsi implikasi (aturan)

   Fungsi implikasi yang digunakan adalah min (*minimum*).

3. Komposisi aturan

Ada tiga metode yang diperoleh dari kumpulan dan korelasi aturan, antara lain:

a. Metode Max (*maximum*)

Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara mengambil nilai maksimum aturan, kemudian menggunakannya untuk memodifikasi daerah *fuzzy*, dan mengaplikasikannya ke *output* dengan operator OR (union). Jika semua proposisi telah dievaluasi, maka *output* akan berisi suatu himpunan *fuzzy* yang merefleksikan kontribusi dari tiap-tiap proposisi. Secara umum dapat dituliskan:

$\mu_{sf}[x_i] = \max(\mu_{sf}[x_i], \mu_{kf}[x_i]$(2.11)

b. Metode Additive (Sum)

Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara melakukan *bounded-sum* terhadap semua *output* daerah *fuzzy*. Secara umum dituliskan:

$$\mu_{sf}[x_i] = \min(1, \mu_{sf}[x_i] + \mu_{kf}[x_i]) \quad (2.12)$$

c. Metode Probabilistik OR (*probor*)

Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara melakukan *product* terhadap semua *output* daerah *fuzzy*. Secara umum dituliskan:

$\mu_{sf}[x_i] = \min(\mu_{sf}[x_i] + \mu_{kf}[x_i]) - (\mu_{sf}[x_i] * \mu_{kf}[x_i])$(2.13)

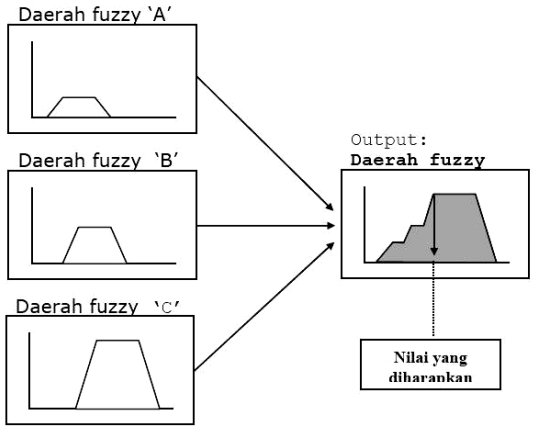
dengan:

$\mu_{sf}[x_i]$: nilai keanggotaan solusi *fuzzy* sampai aturan ke-i.

$\mu_{kf}[x_i]$: nilai keanggotaan konsekuen *fuzzy* aturan ke-i.

4. Penegasan (defuzifikasi)

   Input dari proses defuzifikasi adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan *output* yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut. Sehingga jika diberikan suatu himpunan *fuzzy* dalam *range* tertentu, maka harus dapat diambil suatu nilai *crisp* tertentu sebagai *output* seperti terlihat pada gambar 2.10.

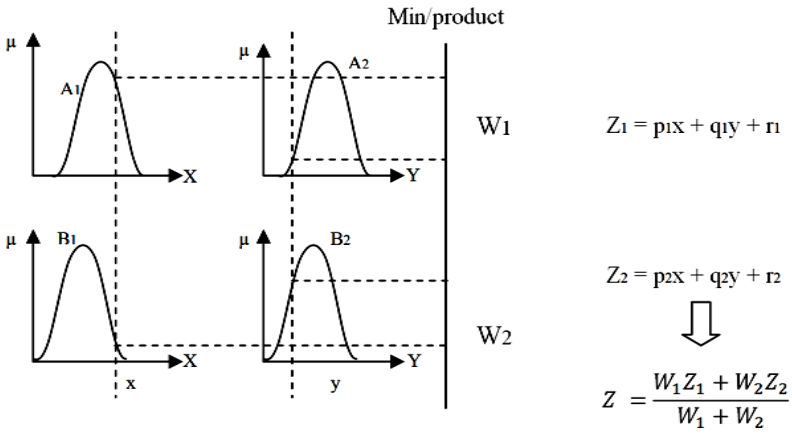


Gambar 2.10 Proses defuzifikasi[5]

Proses agregasi dan defuzifikasi untuk mendapatkan nilai tegas sebagai *output* M aturan *fuzzy* juga dilakukan dengan menggunakan rata-rata terbobot (*weighted average*), yang didefinisikan sebagai berikut:

$$Z = \frac{\sum_{k=1}^{M} W_k Z_k}{\sum_{k=1}^{M} W_k} \quad (2.14)$$

Pada gambar 2.11 berikut menunjukkan mekanisme penalaran inferensi model Sugeno orde satu.

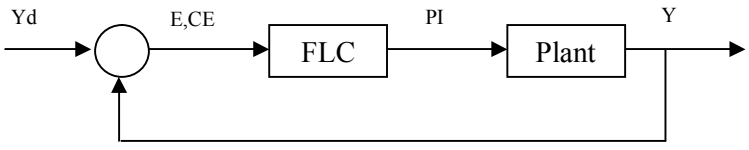


Gambar 2.11 Mekanisme penalaran *fuzzy* model Sugeno[3]

2.3 *Fuzzy Logic Controller* (FLC)

Tujuan utama dalam sistem pengendali adalah mendapatkan keluaran (*output*) sebagai respon dari masukan (*input*). Pengendali logika *fuzzy* (*Fuzzy Logic Controller*) merupakan suatu sistem kendali yang berdasar pada basis pengetahuan manusia didalam melakukan kendali terhadap suatu proses. Konsep matematika yang mendasari logika *fuzzy* sangat sederhana dan mudah dimengerti[3].

FLC merupakan sistem inferensi *fuzzy* yang struktur dasarnya seperti penjelasan pada sub-bab 2.2 sebelumnya (sistem inferensi *fuzzy*). FLC dapat digabungkan dengan sistem *close loop* seperti terlihat pada gambar 2.12 [3].

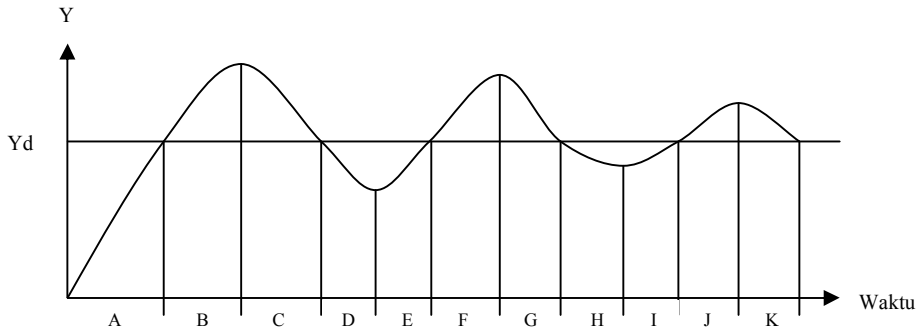


Gambar 2.12 Struktur sistem kendali yang menerapkan FLC[3]

Keluaran plant yang diinginkan oleh nilai acuan (Yd) dibandingkan dengan keluaran aktual (Y), sehingga terdapat galat (*Error*, E). *Error* (E) dan perubahan galat (*Change of Error*, CE) merupakan variabel masukan bagi FLC. Keluaran FLC merupakan masukan *plant* atau masukan proses (*Process Input*, PI) [3].

Basis aturan disusun berdasarkan pengetahuan kualitatif (pendekatan heuristik) sistem *close loop* yang bertujuan untuk mengurangi lonjakan (*overshoot*), waktu bangkit (*rise time*), dan osilasi[3].

Gambar 2.13 merupakan tanggapan undak (*step respone*) sistem yang dimisalkan himpunan *fuzzy* untuk setiap variabel *input*/*output*dinyatakan dengan istilah [N,Z,P] yang berarti *Negative*, *Zero*, dan *Positive*[3].



	A	B	C	D	E	F	G	H	I	J	K
E	+	-	-	+	+	-	-	+	+	-	-
CE	-	-	+	+	-	-	+	+	-	-	+

Gambar 2.13 Tanggapan undak sistem[3]

Berdasarkan tanggapan undak sistem diatas, maka aturan kendali *fuzzy* dapat diformulasikan sebagai berikut:

1. Tanggapan untuk mempersingkat waktu bangkit.

   IF E is Positive AND CE is Negative THEN output is Positive.

2. Tanggapan untuk mengurangi lonjakan.

   IF E is Negative AND CE is Negative THEN output is Negative.

3. Tanggapan untuk mengurangi osilasi.

   IF E is ZeroAND CE is Positive THEN output is Positive[3].

2.4 *Fuzzy C-Means*

Fuzzy C-Means (FCM) adalah suatu teknik peng-*cluster*-an data yang mana keberadaan tiap-tiap titik data dalam suatu *cluster* ditentukan oleh nilai keanggotaan. Teknik ini pertama kali diperkenalkan oleh Jim Bezdek pada tahun 1981. Konsep dasar FCM, pertama kali adalah menentukan pusat *cluster* yang akan menandai lokasi rata-rata untuk tiap-tiap *cluster*. Pada kondisi awal, pusat *cluster* ini masih belum akurat. Dengan cara memperbaiki pusat *cluster* dan nilai keanggotaan tiap-tiap titik data secara berulang, maka akan dapat dilihat bahwa pusat *cluster* akan bergerak menuju lokasi yang tepat. Perulangan ini didasarkan pada minimisasi fungsi obyektif yang menggambarkan jarak dari titik data yang diberikan ke pusat *cluster* yang terbobot oleh derajat keanggotaan titik data tersebut[4].

Keluaran dari FCM bukan merupakan *fuzzy inference system*, namun merupakan deretan pusat *cluster* dan beberapa derajat keanggotaan untuk tiap-tiap

titik data. Informasi ini dapat digunakan untuk membangun suatu *fuzzy inference system*[4].

Algoritma FCM dapat diberikan sebagai berikut:

1. Tentukan :
   a. Matriks X berukuran n x m, dengan n = jumlah data yang akan di-*cluster*; dan m = jumlah variabel.
   b. Jumlah *cluster*yang akan dibentuk = C(≥ 2).
   c. Pangkat (pembobot) = w (> 1).
   d. Maksimum iterasi.
   e. Kriteria penghentian = ξ (nilai positif yang sangat kecil)
   f. Iterasi awal, t = 1, dan Δ = 1.
2. Bentuk matriks partisi awal, U°, sebagai berikut:

$$U = \begin{bmatrix} \mu_{11}(x_1) & \mu_{12}(x_2) \ldots & \mu_{1n}(x_n) \\ \mu_{21}(x_1) & \mu_{22}(x_2) \ldots & \mu_{2n}(x_n) \\ \vdots & & \vdots \\ \mu_{C1}(x_1) \mu_{C1}(x_2) & \ldots & \mu_{Cn}(x_n) \end{bmatrix} \quad (2.15)$$

   (matrik partisi awal dipilih secara acak)
3. Hitung pusat *cluster*, V, untuk setiap *cluster*:

$$V_{ij} = \frac{\sum_{k=1}^{n} (\mu_{ik})^w \cdot x_{kj}}{\sum_{k=1}^{n} (\mu_{ik})^w} \quad (2.16)$$

4. Perbaiki derajat keanggotaan setiap data pada setiap *cluster* (perbaiki matriks partisi), sebagai berikut:

$$\mu_{ik} = \left[\sum_{j=1}^{C} \left(\frac{d_{ik}}{d_{jk}} \right)^{2/(w-1)} \right]^{-1} \quad (2.17)$$

Dengan:

$$d_{ik} = d(X_k - V_i) = \left[\sum_{j=1}^{m}(X_{kj} - V_{ij})\right]^{1/2} \quad (2.18)$$

5. Tentukan kriteria berhenti, yaitu perubahan matriks partisi pada iterasi sekarang dengan iterasi sebelumnya, sebagai berikut:

$$\Delta = \|U^t - U^{t-1}\| \quad (2.19)$$

Apabila $\Delta \leq \xi$ maka iterasi dihentikan, namun apabila $\Delta > \xi$ maka naikkan iterasi (t=t+1) dan kembali ke langkah-3. Pencarian Δ dapat dilakukan dengan mengambil elemen terbesar dari nilai mutlak selisih antara $\mu_{ik}(t)$ dengan $\mu_{ik}(t-1)$ [4].

2.5 *Adaptive Neuro Fuzzy Inference System*

ANFIS (*Adaptive Neuro Fuzzy Inference System* atau *Adaptive Network-based Fuzzy Inference System*) adalah arsitektur yang secara fungsional sama dengan *fuzzy rule base* model Sugeno. Arsitektur ANFIS juga sama dengan jaringan syaraf dengan fungsi radial dengan sedikit batasan tertentu. Bisa dikatakan bahwa ANFIS adalah suatu metode yang mana dalam melakukan penyetelan aturan digunakan algoritma pembelajaran terhadap sekumpulan data. Pada ANFIS juga memungkinkan aturan-aturan untuk beradaptasi[4].

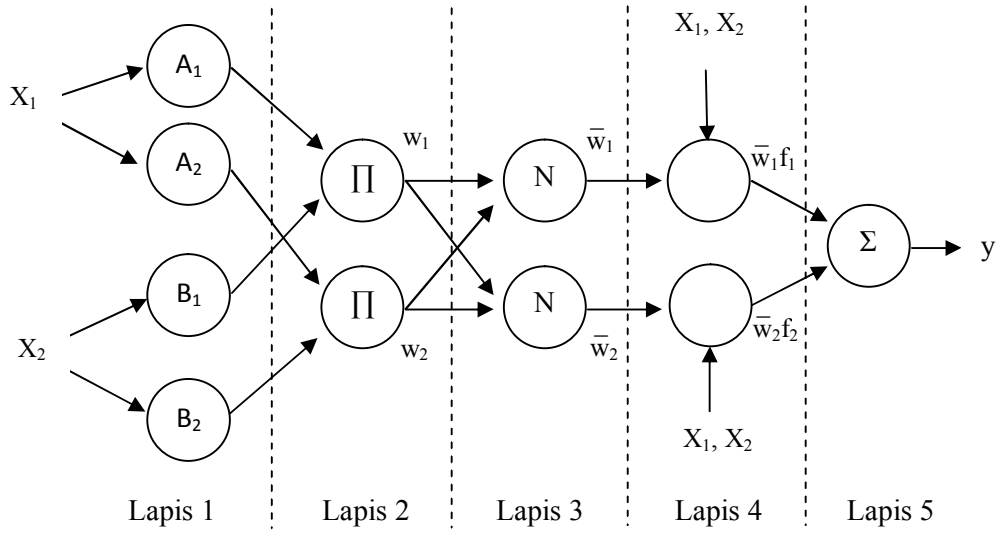
Jika dimisalkan ANFIS terdapat dua aturan pada basis aturan model Sugeno yang memiliki dua *input* x_1, x_2 dan satu *output* y, sebagai berikut:

If x_1 is A_1 and x_2 is B_1 Then $y_1 = c_{11}x_1 + c_{12}x_2 + c_{10}$

If x_1 is A_2 and x_2 is B_2 Then $y_2 = c_{21}x_1 + c_{22}x_2 + c_{20}$

Jika α-predikat untuk aturan kedua aturan adalah w_1 dan w_2, maka dapat dihitung rata-rata terbobot:

$$y = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2} = \bar{w}_1 y_1 + \bar{w}_2 y_2 (2.20)$$



Gambar. 2.14 Arsitektur jaringan ANFIS[4]

Gambar 2.14 merupakan arsitektur ANFIS Sugeno yang terdiri atas lima lapis, dan setiap lapis terdapat *node*. Fungsi dari setiap lapis adalah sebagai berikut:

1. Lapis 1

   Tiap-tiap neuron i pada lapisan pertama adaptif terhadap parameter suatu fungsi aktivasi. *Output* dari tiap neuron berupa derajat keanggotaan yang diberikan oleh fungsi keanggotaan *input*, yaitu: $\alpha_{A1}(x_1)$, $\alpha_{B1}(x_2)$, $\alpha_{A2}(x_1)$, atau $\alpha_{B2}(x_2)$,. Sebagai contoh misalkan fungsi keanggotaan diberikan sebagai:

$$\mu(x) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}} \quad (2.21)$$

Dimana {a,b,c} adalah parameter-parameter, biasanya b=1. Jika nilai parameter-parameter ini berubah, maka bentuk kurva yang terjadi-pun akan ikut berubah. Parameter-parameter itu biasanya dikenal dengan nama *premise parameters*.

2. Lapis 2

   Tiap-tiap neuron pada lapisan kedua berupa neuron tetap yang *output*-nya adalah hasil dari masukan. Biasanya digunakan operator AND. Tiap-tiap *node* merepresentasikan α predikat dari aturan ke-i.

3. Lapis 3

   Tiap-tiap neuron pada lapisan ketiga berupa *node* tetap yang merupakan hasil perhitungan rasio dari α predikat (w), dari aturan ke-i terhadap jumlah dari keseluruhan α predikat.

   $$\overline{w_i} = \frac{w_i}{w_1 + w_2} \text{ , dengan } i = 1,2 \quad (2.22)$$

   Keluaran lapis ini disebut kuat penyulutan ternormalisasi atau *normalised firing strength*.

4. Lapis 4

   Tiap-tiap neuron pada lapisan keempat merupakan *node* adaptif terhadap suatu *output*.

   $$\overline{w}_i y_i = \overline{w}_i \left(c_{i1} x_1 + c_{i2} x_2 + c_{i0}\right) \quad (2.23)$$

Dengan $\bar{w}_i$adalah *normalised firing strength* dari lapis 3 dan $\{c_{i1},c_{i2},c_{i0}\}$ adalah parameter-parameter pada neuron tersebut. Parameter-parameter pada lapis ini disebut parameter konsekuensi (*consequent parameters*).

5. Lapis 5

   Tiap-tiap neuron pada lapisan kelima adalah *node* tetap yang merupakan jumlah dari semua masukan[4].

2.5.1 Algoritma Pelatihan *Hybrid*

ANFIS dilatih dengan algoritma pelatihan *hybrid*. Algoritma pelatihan *hybrid* terdiri atas dua langkah, yaitu langkah maju dan langkah balik[3].

1. Pada langkah maju, parameter premis tetap, sedangkan parameter konsekuensi diidentifikasi dengan metode estimasi kuadrat *error* (*Least Square Estimator*, LSE).
2. Pada langkah balik sinyal galat (*error*) antara keluaran yang diinginkan dan keluaran aktual dirambatkan mundur sedangkan parameter premis diperbarui dengan metode penurunan gradien (*Gradient Descent*, GD)[3].

Pada saat *premise parameters* ditemukan, *output* yang terjadi akan merupakan kombinasi linier dari *consequent parameters*, yaitu[4]:

$$y = \frac{w_1}{w_1+w_2} y_1 + \frac{w_2}{w_1+w_2} y_2 \qquad (2.24)$$

$$= \bar{w}_1(c_{11}x_1 + c_{12}x_2 + c_{10}) + \bar{w}_2(c_{21}x_1 + c_{22}x_2 + c_{20}); \qquad (2.25)$$

$$= (\bar{w}_1x_1)c_{11} + (\bar{w}_1x_2)c_{12} + \bar{w}_1c_{10} + (\bar{w}_2x_1)c_{21} + (\bar{w}_2x_2)c_{22} + \bar{w}_2c_{20}; \qquad (2.26)$$

adalah linier terhadap parameter c_{ij} (i=1,2 dan j=0,1,2).

Algoritma *hybrid* akan mengatur parameter-parameter c_{ij} secara maju (*forward*) dan akan mengatur parameter-parameter $\{a_i, b_i, c_i\}$ secara mundur (*backward*).

Pada langkah maju, *input* jaringan akan merambat maju sampai pada lapisan keempat, dimana parameter-parameter c_{ij} akan diidentifikasi dengan menggunakan metode *least-square*. Sedangkan pada langkah mundur, *error* sinyal akan merambat mundur dan parameter-parameter $\{a_i, b_i, c_i\}$ akan diperbaiki dengan menggunakan metode *gradient-descent*[4].

2.5.2 LSE Rekursif

Pada pembelajaran *off-line*, misalkan terapat satu *output* pada jaringan adaptif, yaitu[4]:

$$O = F(i,S) \tag{2.27}$$

dengan i adalah vektor dari variabel *input*, S adalah himpunan parameter-parameter, dan F adalah fungsi yang diimplementasikan oleh jaringan adaptif. Jika terdapat fungsi H sedemikian hingga fungsi komposit H ° F adalah linier untuk elem-elemen S, maka elemen-elemen ini dapat diidentifikasi dengan metode *least square*. Andaikan parameter S dapat dibagi menjadi 2, yaitu:

$$S = S_1 \oplus S_2 \tag{2.28}$$

dengan $\oplus$ adalah *direct sum*, sedemikian hingga H ° F linier untuk elemen-elemen S_2, kemudian dengan mengaplikasikan H ke dalam persamaan (2.27), diperoleh:

$$H(°) = H ° F(Bi, S) \tag{2.29}$$

yang linier terhadap elemen-elemen S_2. Apabila diberikan elemen-elemen S_1, maka dapat ditempatkan P data pelatihan ke dalam persamaan (2.29), dan mendapatkan sistem persamaan linier, sebagai berikut:

$$A\theta = y \tag{2.30}$$

dengan θ adalah vektor yang tidak diketahui dan elemen-elemennya merupakan parameter-parameter dari S_2. Persamaan (2.30) ini kemudian dapat diselesaikan dengan menggunakan metode LSE.

Apabila terdapat m elemen pada vektor *output* y (y berukuran m x 1), dan n parameter θ (θ berukuran n x 1), dengan baris ke-i pada matrik [A⋮ y] dinotasikan sebagai [a_i^T⋮ y]. Apabila m = n, maka dapat ditentukan nilai q dari persamaan (2.30) sebagai berikut:

$$\theta = A^{-1} y \tag{2.31}$$

Namun, apabila m > n, maka persamaan (2.30) harus dimodifikasi dengan menambahkan vektor *error* (e), sehingga:

$$A\theta + e = y \tag{2.32}$$

Untuk mendapatkan solusi eksak dari persamaan (2.30), maka harus dicari $a\theta = \hat{\theta}$ yang meminimumkan jumlah kuadrat *error* sebagai berikut:

$$E(\theta) = \sum_{i=1}^{m} (y_i - {a_i}^T)^2 = e^T e = (y - A\theta)^T (y - A\theta) \tag{2.33}$$

dengan e = y – Aθ adalah vektor *error* yang terjadi sebagai akibat pemilihan θ. Jumlah kuadrat *error* pada persamaan (2.33) akan menjadi minimum bila $\theta = \hat{\theta}$, yang sering disebut dengan nama *Least*-S*quareEstimator* (LSE), yang ditulis sebagai berikut:

$$A^T\hat{\theta} = A^T y \quad (2.34)$$

jika A^TA adalah nonsingular, dan $\hat{\theta}$ bersifat unik, maka dapat diberikan:

$$\hat{\theta} = (A^T A)^{-1} A^T y \quad (2.35)$$

atau dengan membuang (^), dan dengan mengasumsikan jumlah baris dari pasangan A dan y adalah k, maka diperoleh:

$$\theta_k = (A^T A)^{-1} A^T y \quad (2.36)$$

Salah satu metode LSE adalah LSE rekursif. Pada LSE rekursif dapat ditambahkan suatu pasangan data [a^T⋮ y], sehingga dimiliki sebanyak (m+1) pasangan data. Dari sini dapat dihitung kembali LSE θ_{k+1} dengan bantuan θ_k. Bentuk semacam ini dikenal dengan nama LSE rekursif.

Karena jumlah parameter ada sebanyak n, maka dapat diselesaikan matriks n x n dengan menggunakan metode invers, sebagai berikut:

$$P_n = (A_n{}^T A_n)^{-1} \quad (2.37)$$

$$\theta_n = P_n A_n{}^T y_n \quad (3.38)$$

selanjutnya, iterasi dimulai dari data ke-(n+1), dengan nilai P_{k+1} dan θ_{k+1} dapat dihitung sebagai berikut:

$$P_{k+1} = P_k - \frac{P_k a_{k+1} a_{k+1}^T P_k}{1 + a_{k+1}^T P_k a_{k+1}} \quad (2.39)$$

$$\theta_{k+1} = \theta_k + P_{k+1} a_{k+1} (y_{k+1} - a_{k+1}^T \theta_k) \quad (3.40)$$

Nilai P_0 dan θ_0 dihitung berdasarkan persamaan (2.37) dan (2.38).

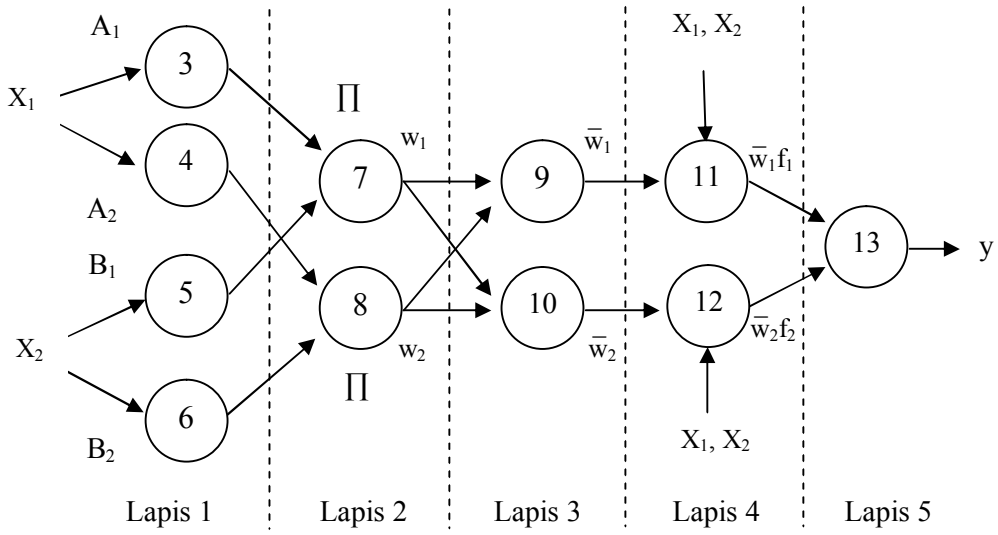
Sehingga, jika dilihat kembali persamaan (2.26), maka dapat disimpulkan bahwa disana ada 6 parameter (n=6) untuk n pasangan data pelatihan[4].

2.5.3 Model Propagasi *Error*

Pada jaringan adaptif dapat dilatih untuk mendapatkan nilai parameter a dan c, pada persamaan (2.21). Dengan mengambil nilai b = 1, persamaan (2.21) menjadi[4]:

$$\mu(x) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^2} \quad (2.41)$$

Untuk melakukan perbaikan terhadap parameter a dan c tersebut, digunakan model propagasi *error* dengan konsep *gradient descent*.



Gambar 2.15 Arsitektur jaringan adaptif dengan 5 lapisan[4]

Misalkan dimiliki jaringan adaptif seperti terlihat pada gambar 2.15 yang terdiri atas 5 lapisan, dan memiliki sebanyak N(L) neuron pada lapisan ke-L, maka jumlah kuadrat *error* (SSE) pada lapisan ke-L data ke-p, $1 \leq p \leq N$, adalah[4]:

$$E_p = \sum_{k=1}^{N(L)} (d_k - X_{L,k})^2 \text{ (2.42)}$$

1. *Error* pada lapisan ke-5

   Apabila dimiliki jaringan adaptif seperti pada gambar 2.15, yang hanya memiliki 1 neuron pada lapisan *output* (neuron 13), maka propagasi *error* yang menuju pada lapisan ke-5 dapat dirumuskan sebagai berikut:

   $$\varepsilon_{13} = \frac{\partial E_p}{\partial x_{13}} = -2(d_{13} - x_{13}) = -2\left(y_p - y_p^*\right) \text{(2.43)}$$

   Dengan y_p adalah target *output* data pelatihan ke-p, dan adalah *output* jaringan data pelatihan ke-p.

2. *Error* pada lapisan ke-4

   Sesuai jaringan adaptif pada gambar 2.15, propagasi *error* yang menuju pada lapisan ke-4, yaitu neuron 11 dan neuron 12 dapat dirumuskan sebagai berikut:

   $$\varepsilon_{11} = \left(\frac{\partial E_p}{\partial x_{13}}\right)\left(\frac{\partial f_{13}}{\partial x_{11}}\right) = \varepsilon_{13}\left(\frac{\partial f_{13}}{\partial x_{11}}\right) = \varepsilon_{13}(1) = \varepsilon_{13} \text{(2.44)}$$

   Karena $f_{13} = \bar{w}_1 f_1 + \bar{w}_2 f_2$, maka $\frac{\partial f_{13}}{\partial(\bar{w}_1 w_1)} = 1$

   $$\varepsilon_{12} = \left(\frac{\partial E_p}{\partial x_{13}}\right)\left(\frac{\partial f_{13}}{\partial x_{12}}\right) = \varepsilon_{13}\left(\frac{\partial f_{13}}{\partial x_{12}}\right) = \varepsilon_{13}(1) = \varepsilon_{13} \text{(2.45)}$$

3. *Error* pada lapisan ke-3

   Sesuai jaringan adaptif pada gambar 2.15, propagasi *error* yang menuju pada lapisan ke-3, yaitu neuron 9 dan neuron 10 dapat dirumuskan sebagai berikut:

$$\varepsilon_9 = \left(\frac{\partial E_p}{\partial x_{13}}\right)\left(\frac{\partial f_{13}}{\partial x_{11}}\right)\left(\frac{\partial f_{11}}{\partial x_9}\right) = \varepsilon_{11}\left(\frac{\partial f_{11}}{\partial x_9}\right) = \varepsilon_{11} f_1 (2.46)$$

Karena $f_{11} = \bar{w}_1 f_1$, maka $\frac{\partial f_{11}}{\partial(\bar{w}_1)} = f_1$.

$$\varepsilon_{10} = \left(\frac{\partial E_p}{\partial x_{13}}\right)\left(\frac{\partial f_{13}}{\partial x_{12}}\right)\left(\frac{\partial f_{12}}{\partial x_{10}}\right) = \varepsilon_{12}\left(\frac{\partial f_{12}}{\partial x_{10}}\right) = \varepsilon_{12} f_2 (2.47)$$

Karena $f_{12} = \bar{w}_2 f_2$, maka $\frac{\partial f_{12}}{\partial(\bar{w}_2)} = f_2$.

4. *Error* pada lapisan ke-2

Sesuai jaringan adaptif pada gambar 2.15, propagasi *error* yang menuju pada lapisan ke-2, yaitu neuron 7 dan neuron 8 dapat dirumuskan sebagai berikut:

$$\varepsilon_7 = \left(\frac{\partial E_p}{\partial x_{13}}\right)\left(\frac{\partial f_{13}}{\partial x_{11}}\right)\left(\frac{\partial f_{11}}{\partial x_9}\right)\left(\frac{\partial f_9}{\partial x_7}\right) + \left(\frac{\partial E_p}{\partial x_{13}}\right)\left(\frac{\partial f_{13}}{\partial x_{12}}\right)\left(\frac{\partial f_{12}}{\partial x_{10}}\right)\left(\frac{\partial f_{10}}{\partial x_7}\right) (2.48)$$

$$= \left(\frac{\partial f_9}{\partial x_7}\right) + \varepsilon_{10}\left(\frac{\partial f_{10}}{\partial x_7}\right) (2.49)$$

$$= \varepsilon_9\left(\frac{w_2}{(w_1 + w_2)^2}\right) + \varepsilon_{10}\left(\frac{w_2}{(w_1 + w_2)^2}\right) (2.50)$$

$$= \frac{w_2}{(w_1 + w_2)^2}(\varepsilon_9 + \varepsilon_{10}) (2.51)$$

Karena $f_9 = \frac{w_1}{w_1 + w_2}$, maka $\frac{\partial f_9}{\partial w_1} = \frac{w_2}{(w_1 + w_2)^2}$; dan $f_{10} = \frac{w_2}{w_1 + w_2}$,

Maka $\frac{\partial f_{10}}{\partial w_1} = \frac{w_2}{(w_1 + w_2)^2}$

$$\varepsilon_8 = \left(\frac{\partial E_p}{\partial x_{13}}\right)\left(\frac{\partial f_{13}}{\partial x_{12}}\right)\left(\frac{\partial f_{12}}{\partial x_{10}}\right)\left(\frac{\partial f_{10}}{\partial x_8}\right) + \left(\frac{\partial E_p}{\partial x_{13}}\right)\left(\frac{\partial f_{13}}{\partial x_{11}}\right)\left(\frac{\partial f_{11}}{\partial x_9}\right)\left(\frac{\partial f_9}{\partial x_8}\right) (2.52)$$

$$= \varepsilon_{10}\left(\frac{\partial f_{10}}{\partial x_8}\right) + \varepsilon_9\left(\frac{\partial f_9}{\partial x_8}\right) (2.53)$$

$$= \varepsilon_{10}\left(\frac{w_1}{(w_1+w_2)^2}\right) + \varepsilon_{10}\left(-\frac{w_1}{(w_1+w_2)^2}\right) (2.54)$$

$$= \frac{w_1}{(w_1+w_2)^2}(\varepsilon_{10} - \varepsilon_9) (2.55)$$

Karena $f_9 = \frac{w_1}{w_1+w_2}$, maka $\frac{\partial f_9}{\partial w_2} = -\frac{w_1}{(w_1+w_2)^2}$; dan $f_{10} = \frac{w_2}{w_1+w_2}$,

Maka $\frac{\partial f_{10}}{\partial w_2} = \frac{w_1}{(w_1+w_2)^2}$.

5. *Error* pada lapisan ke-1

   Sesuai jaringan adaptif pada gambar 2.15, propagasi *error* yang menuju pada lapisan ke-1, yaitu neuron 3, 4, 5 dan neuron 6 dapat dirumuskan sebagai berikut:

$$\varepsilon_3 = \varepsilon_7\left(\frac{\partial f_7}{\partial x_3}\right) = \varepsilon_7 \mu_{B1}(x_2) (2.56)$$

$$\varepsilon_4 = \varepsilon_8\left(\frac{\partial f_8}{\partial x_4}\right) = \varepsilon_8 \mu_{B2}(x_2) (2.57)$$

$$\varepsilon_5 = \varepsilon_7\left(\frac{\partial f_7}{\partial x_5}\right) = \varepsilon_7 \mu_{A1}(x_1) (2.58)$$

$$\varepsilon_6 = \varepsilon_8\left(\frac{\partial f_8}{\partial x_6}\right) = \varepsilon_8 \mu_{A1}(x_1) (2.59)$$

   Karena $f_7 = (\mu_{A1}(x_1))(\mu_{B1}(x_2))$, maka $\frac{\partial f_7}{\partial(\mu_{A1}(x_1))} = \mu_{B1}(x_2)$ dan $\frac{\partial f_7}{\partial(\mu_{B1}(x_2))} = \mu_{A1}(x_1)$; dan karena $f_8 = (\mu_{A2}(x_1))(\mu_{B2}(x_2))$, maka $\frac{\partial f_8}{\partial(\mu_{A1}(x_1))} = \mu_{B2}(x_2)$ dan $\frac{\partial f_8}{\partial(\mu_{B2}(x_2))} = \mu_{A2}(x_1)$.

Selanjutnya, *error* tersebut digunakan untuk mencari informasi *error* terhadap parameter a (a_{11} dan a_{12} untuk A_1 dan A_2; a_{21} dan a_{22} untuk B_1 dan B_2), dan c(c_{11} dan c_{12} untuk A_1 dan A_2; c_{21} dan c_{22} untuk B_1 dan B_2) sebagai berikut:

Karena, $\frac{\partial f}{\partial a_{ik}} = \frac{2(x_i - c_{ik})^2}{a_{ik}^3\left(1+\left(\frac{x_i - c_{ik}}{a_{ik}}\right)^2\right)^2}$, maka

$$\varepsilon_{a11} = \varepsilon_3\left(\frac{\partial f_3}{\partial a_{11}}\right) + \varepsilon_4\left(\frac{\partial f_4}{\partial a_{11}}\right) (2.60)$$

$$= (\varepsilon_3)\frac{2(x_1 - c_{11})^2}{a_{11}^3\left(1+\left(\frac{x_1 - c_{11}}{a_{11}}\right)^2\right)^2} + \varepsilon_4(0) (2.61)$$

$$= (\varepsilon_3)\frac{2(x_1 - c_{11})^2}{a_{11}^3\left(1+\left(\frac{x_1 - c_{11}}{a_{11}}\right)^2\right)^2} (2.62)$$

$$\varepsilon_{a12} = \varepsilon_3\left(\frac{\partial f_3}{\partial a_{12}}\right) + \varepsilon_4\left(\frac{\partial f_4}{\partial a_{12}}\right) (2.63)$$

$$= (\varepsilon_3)(0) + (\varepsilon_4)\frac{2(x_1 - c_{12})^2}{a_{12}^3\left(1+\left(\frac{x_1 - c_{12}}{a_{12}}\right)^2\right)^2} (2.64)$$

$$= (\varepsilon_4)\frac{2(x_1 - c_{12})^2}{a_{12}^3\left(1+\left(\frac{x_1 - c_{12}}{a_{12}}\right)^2\right)^2} (2.65)$$

$$\varepsilon_{a21} = \varepsilon_5\left(\frac{\partial f_5}{\partial a_{21}}\right) + \varepsilon_6\left(\frac{\partial f_6}{\partial a_{21}}\right) (2.66)$$

$$= (\varepsilon_5)\frac{2(x_2 - c_{21})^2}{a_{21}^3\left(1+\left(\frac{x_2 - c_{21}}{a_{21}}\right)^2\right)^2} + \varepsilon_6(0) (2.67)$$

$$= (\varepsilon_5)\frac{2(x_2 - c_{21})^2}{a_{21}^3\left(1+\left(\frac{x_2 - c_{21}}{a_{21}}\right)^2\right)^2} (2.68)$$

$$\varepsilon_{a22} = \varepsilon_5\left(\frac{\partial f_5}{\partial a_{22}}\right) + \varepsilon_6\left(\frac{\partial f_6}{\partial a_{22}}\right) (2.69)$$

$$= (\varepsilon_5)(0) + (\varepsilon_6)\frac{2(x_2 - c_{22})^2}{a_{22}^3\left(1 + \left(\frac{x_2 - c_{22}}{a_{22}}\right)^2\right)^2} (2.70)$$

$$= (\varepsilon_6)\frac{2(x_2 - c_{22})^2}{a_{22}^3\left(1 + \left(\frac{x_2 - c_{22}}{a_{22}}\right)^2\right)^2} (2.71)$$

Karena, $\frac{\partial f}{\partial c_{ik}} = \frac{2(x_i - c_{ik})^2}{a_{ik}^2\left(1 + \left(\frac{x_i - c_{ik}}{a_{ik}}\right)^2\right)^2}$, maka

$$\varepsilon_{c11} = \varepsilon_3\left(\frac{\partial f_3}{\partial c_{11}}\right) + \varepsilon_4\left(\frac{\partial f_4}{\partial c_{11}}\right) (2.72)$$

$$= (\varepsilon_3)\frac{2(x_1 - c_{11})^2}{a_{11}^2\left(1 + \left(\frac{x_1 - c_{11}}{a_{11}}\right)^2\right)^2} + \varepsilon_4(0) (2.73)$$

$$= (\varepsilon_3)\frac{2(x_1 - c_{11})^2}{a_{11}^2\left(1 + \left(\frac{x_1 - c_{11}}{a_{11}}\right)^2\right)^2} (2.74)$$

$$\varepsilon_{c12} = \varepsilon_3\left(\frac{\partial f_3}{\partial c_{12}}\right) + \varepsilon_4\left(\frac{\partial f_4}{\partial c_{12}}\right) (2.75)$$

$$= (\varepsilon_3)(0) + (\varepsilon_4)\frac{2(x_1 - c_{12})^2}{a_{12}^2\left(1 + \left(\frac{x_1 - c_{12}}{a_{12}}\right)^2\right)^2} (2.76)$$

$$= (\varepsilon_4)\frac{2(x_1 - c_{12})^2}{a_{12}^2\left(1 + \left(\frac{x_1 - c_{12}}{a_{12}}\right)^2\right)^2} (2.77)$$

$$\varepsilon_{c21} = \varepsilon_5\left(\frac{\partial f_5}{\partial c_{21}}\right) + \varepsilon_6\left(\frac{\partial f_6}{\partial c_{21}}\right) (2.78)$$

$$= (\varepsilon_5)\frac{2(x_2 - c_{21})^2}{a_{21}^2\left(1 + \left(\frac{x_2 - c_{21}}{a_{21}}\right)^2\right)^2} + \varepsilon_6(0) \quad (2.79)$$

$$= (\varepsilon_5)\frac{2(x_2 - c_{21})^2}{a_{21}^2\left(1 + \left(\frac{x_2 - c_{21}}{a_{21}}\right)^2\right)^2} \quad (2.80)$$

$$\varepsilon_{c22} = \varepsilon_5\left(\frac{\partial f_5}{\partial c_{22}}\right) + \varepsilon_6\left(\frac{\partial f_6}{\partial c_{22}}\right) \quad (2.81)$$

$$= (\varepsilon_5)(0) + (\varepsilon_6)\frac{2(x_2 - c_{22})^2}{a_{22}^2\left(1 + \left(\frac{x_2 - c_{22}}{a_{22}}\right)^2\right)^2} \quad (2.82)$$

$$= (\varepsilon_6)\frac{2(x_2 - c_{22})^2}{a_{22}^2\left(1 + \left(\frac{x_2 - c_{22}}{a_{22}}\right)^2\right)^2} \quad (2.83)$$

Dari sini dapat ditentukan perubahan nilai parameter a_{ij} dan c_{ij} (Δa_{ij} dan Δc_{ij}) sebagai berikut:

$\Delta a_{ij} = \eta \varepsilon_{ij} x_i$, dan (2.84)

$\Delta c_{ij} = \eta \varepsilon_{cij} x_i$ (2.85)

Dengan h adalah laju pembelajaran yang terletak pada interval [0,1]. Sehingga nilai a_{ij} dan c_{ij} yang baru adalah:

$a_{ij} = a_{ij}$ (lama) + Δa_{ij} dan (2.86)

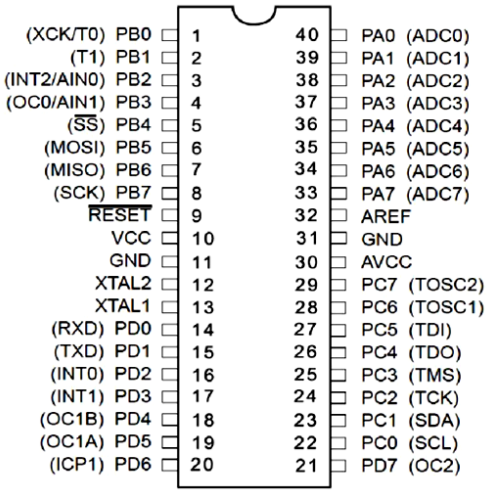
$c_{ij} = c_{ij}$ (lama) + Δc_{ij} (2.87)

2.6 Mikrokontroler ATMega 16

Mikrokontroler ATMega16 merupakan sebuah *chip* mikrokomputer yang didalamnya terdapat mikroprosessor (CPU), *memory* RAM, ROM, EEPROM, I/O Port dan bahkan ADC serta beberapa fasilitas penunjang lainnya seperti misalnya Timer, USART, dan PWM yang sudah terintegrasi dalam satu IC[6].

Mikrokontroler ATMega16 ini memiliki arsitektur Harvard, yaitu memisahkan memori untuk kode program dan memori untuk data, sehingga dapat memaksimalkan unjuk kerja. Instruksi-instruksi dalam memori program dieksekusi dalam satu alur tunggal, dimana pada saat satu instruksi dikerjakan instruksi berikutnya sudah diambil (*pre-fetched*) dari memori program. Konsep inilah yang memungkinkan instruksi-instruksi dapat dieksekusi setiap satu *clock cycle*. CPU terdiri dari 32x8-bit *general purpose register* yang dapat diakses dengan cepat dalam satu *clock cycle*, yang mengakibatkan operasi *Arithmetic Logic Unit* (ALU) dapat dilakukan dalam satu *cycle*. Pada operasi ALU, dua *operand* berasal dari *register*, kemudian operasi dieksekusi dan hasilnya disimpan kembali pada *register* dalam satu *clock cycle*. Operasi aritmatik dan *logic* pada ALU akan mengubah *bit-bit* yang terdapat pada *Status Register* (SREG). Proses pengambilan instruksi dan pengeksekusian instruksi berjalan secara *paralell*[6].

Konfigurasi *pin* ATMega16 dengan kemasan 40 *pin* DIP (*Dual In-line Package*) dapat dilihat pada gambar 2.16.



Gambar 2.16 Konfigurasi pin ATMega16

Dari gambar 2.16 dapat dijelaskan fungsi dari masing-masing *pin* ATMega16 sebagai berikut:

1. VCC merupakan *pin* yang berfungsi sebagai masukan catu daya.
2. GND merupakan *pinGround*.
3. *Port* A(PA0..PA7) merupakan *pininput*/*output* dua arah dan *pin* masukan ADC.
4. *Port* B(PB0..PB7) merupakan *pininput*/*output* dua arah dan *pin* memiliki fungsi khusus.

2.6.1 Pemrograman Mikrokontroler ATMega 16

Pengembangan sebuah sistem menggunakan mikrokontroler AVR buatan ATMEL menggunakan *software* CodeVisionAVR. CodeVisionAVR merupakan *software C-cross compiler*, dimana program dapat ditulis dalam bahasa C, CodeVisionAVR memiliki IDE (*Integrated Development Environment*) yang

lengkap, dimana penulisan program, *compile*, *link*, pembuatan kode mesin (*assembler*) dan *download* program ke *chip* AVR dapat dilakukan pada *codevision*. Selain itu terdapat fasilitas terminal untuk melakukan komunikasi serial dengan mikrokontroler. Proses *download* program ke IC mikrokontroler AVR dapat menggunakan *system download* secara ISP (*In-System Programming*). *In-System Programming Flash on-chip* mengizinkan memori program untuk diprogram ulang dalam sistem menggunakan hubungan serial SPI[6].

Bentuk dasar program dalam bahasa C setidaknya harus memiliki sebuah fungsi. Fungsi dasar ini disebut dengan fungsi utama (fungsi *main*) dan memiliki kerangka program sebagai berikut:

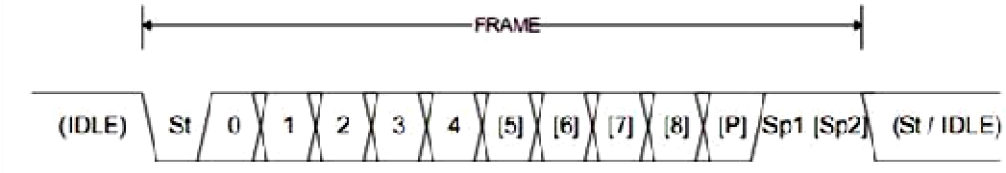
```
void main (void)
{
// pernyataan-pernyataan
}
```

Jika terdapat beberapa fungsi yang lain maka fungsi inilah yang memiliki kedudukan paling tinggi dibandingkan fungsi-fungsi lain sehingga setiap kali program dijalankan akan selalu dimulai dari memanggil fungsi utama terlebih dahulu. Fungsi-fungsi yang lain dapat dipanggil setelah fungsi utama dijalankan melalui pernyataan-pernyataan yang berada di dalam fungsi utama.

2.6.2 USART (*Universal Synchronous and Asynchronous Receiver-Transmitter*)

USART merupakan protokol komunikasi yang umum digunakan dalam pengiriman data serial antara *device* satu dengan yang lainnya. USART menjadi bagian perangkat keraskomputer yang menerjemahkan antara bit-bit paralel data dan bit-bit serial. USART biasanya berupa sirkuit terintegrasi yang digunakan untuk komunikasi serial pada komputer atau port serial perangkat periperal.

USART ini telah terdapat sebagai fitur di dalam mikrokontrolerATMega16.USART memungkinkan transmisi data baik secara *synchronous* maupun*asynchronous* sehingga dengan demikian USART pasti kompatibel dengan UART. Pada ATMega16, untuk mengatur mode dan prosedur komunikasi baik *synchronous* atau*asynchronous* dilakukan melalui register UCSRA, UCSRB, UCSRC, UBRRH, UBRRL, dan UDR.



Gambar 2.17 Transmisi data USART

Gambar 2.17 di atas merupakan ilustrasi pengiriman data secara serial. Pada pengiriman data UART terdapat beberapa parameter yang dapat diatur yaitu start bit, parity bit, dan stop bit. Pengaturan ini harus sama antara pengirim dan penerima karena jika tidak maka data tidak akan diterima. Data yang dikirim adalah data berukuran 8 bit atau 1 byte. Jika ditambah dengan 3 parameter diatas

maka total bit data yang dikirim adalah 11 bit. Dari format data inilah setiap data yang terbaca dapat diterjemahkan menjadi bit-bit yang merepresentasikan data tertentu.

Karena protokol USART ini universal, maka *baudrate* yang ada adalah nilai-nilai tetap yang tidak bisa diubah ubah dari kisaran nilai 110 sampai 11059200 bps (bit/second) atau lebih. Semakin cepat clock yang digunakan mikrokontroler maka *baudrate* akan semakin cepat juga. Umumnya *baudrate* yang digunakan dalam pengiriman data antar mikrokontroler adalah 9600 bps.

2.6.3 ADC (*Analog to Digital Converter*)

ADC adalah pengubah *input* analog menjadi kode-kode digital. ADC banyak digunakan sebagai Pengatur proses industri, komunikasi digital dan rangkaian pengukuran/ pengujian. Umumnya ADC digunakan sebagai perantara antara sensor yang kebanyakan analog dengan sistim komputer seperti sensor suhu, cahaya, tekanan/berat, aliran dan sebagainya kemudian diukur dengan menggunakan sistim digital (komputer). ADCmemiliki 2 karakter prinsip, yaitu kecepatan sampling dan resolusi . Kecepatan sampling suatu ADC menyatakan seberapa sering sinyal analog dikonversikan ke bentuk sinyal digital pada selang waktu tertentu. Kecepatan sampling biasanya dinyatakan dalam *sample per second* (SPS).

Resolusi ADC menentukan ketelitian nilai hasil konversi ADC. Sebagai contoh: ADC 8 bit akan memiliki *output* 8 bit data digital, ini berarti sinyal *input* dapat dinyatakan dalam 255 (2n–1) nilai diskrit. ADC 12 bit memiliki 12 bit

output data digital, ini berarti sinyal *input* dapat dinyatakan dalam 4096 nilai diskrit. Dari contoh diatas ADC 12 bit akan memberikan ketelitian nilai hasil konversi yang jauh lebih baik daripada ADC 8 bit.

Prinsip kerja ADC adalah mengkonversi sinyal analog ke dalam bentuk besaran yang merupakan rasio perbandingan sinyal *input* dan tegangan referensi. Sebagai contoh, bila tegangan referensi 5 volt, tegangan *input* 3 volt, rasio *input* terhadap referensi adalah 60%. Jadi, jika menggunakan ADC 8 bit dengan skala maksimum 255, akan didapatkan sinyal digital sebesar 60% x 255 = 153 (bentuk decimal) atau 10011001 (bentuk biner). Contoh perhitungannya sebagai berikut.

signal = (sample/max_value) * reference_voltage

= (153/255) * 5

= 3 Volts

Mikrokontroler ATMega16 memiliki fasilitas Analog to Digital Converter yang sudah *built-in* dalam *chip*. ATMega16 memiliki resolusi ADC 10-bit dengan 8 *channelinput* dan mendukung 16 macam penguat beda. ADC ini bekerja dengan teknik *successive approximation*. Rangkaian internal ADC ini memiliki catu daya tersendiri yaitu pin AVCC. Tegangan AVCC harus sama dengan VCC ± 0,3 V.

Data hasil konversi ADC dirumuskan sebagai berikut:

$$ADC = \frac{V_{in}.1024}{V_{REF}} (2.88)$$

Untuk mengatur mode dan cara kerja ADC pada ATMega16 dilakukan melalui register ADMUX, ADCSRA, ADCL, ADCH, dan SFIOR.

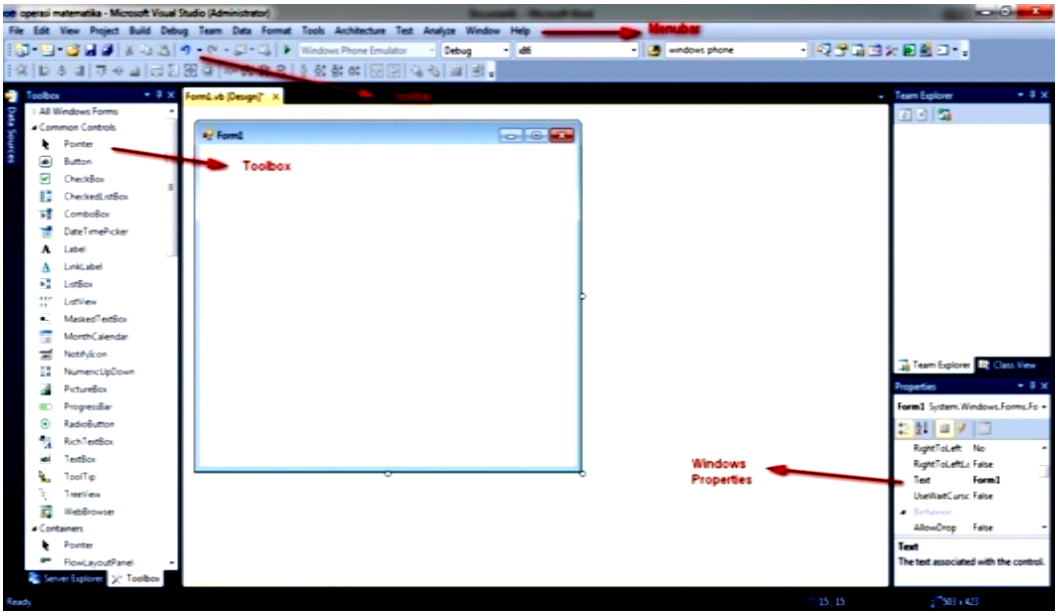
2.7 *Visual Basic* 2010

Visual Basic 2010 merupakan suatu lingkungan pemrograman bagian dari *Microsoft Visual Studio* 2010 dimana programmer dapat menggunakan *Graphical User Interface* (GUI) untuk memilih dan memodifikasi kode yang ditulis dalam bahasa BASIC.

Bahasa *Visual Basic* adalah sebuah bahasa pemrograman yang menggunakan *Integrated Development Environment* (IDE) *visual* yang digunakan untuk membuat program berbasis *Microsoft Windows* dengan menggunakan model pemrograman (COM).

Perangkat lunak ini digunakan untuk menyusun program aplikasi yang bekerja dalam lingkungan sistem operasi windows. Seperti program berbasis windows lainnya, *visual basic* terdiri dari banyak windows. Ketika akan memulai *visual basic* sekumpulan windows yang paling berkaitan inilah yang disebut dengan *Integrated Development Environtment* (IDE). Program yang berbasis windows bersifat *event driven*, artinya program bekerja berdasarkan *event* dalam objek trsebut, misalnya jika seorang user mengklik sebuah tombol maka program akan memberikan reaksi terhadap *event* klik tersebut. Program akan memberikan reaksi sesuai dengan kode-kode program yang dibuat untuk suatu *event* pada objek tertentu.

*Toolbox*berada disebelah kiri jendela kerja sedangkan*Window Project* dan *Widow Properties* berada disebelah kanan jendela kerja. Ketika akan membuat sebuah project baru, maka pada bagian tengah jendela akan terdapat sebuah *form*. *Form* tersebut akan digunakan sebagai tampilan (jendela) untuk program yang dibuat. Gambar 2.18 adalah *screenshot* tampilan *Visual Basic* 2010.



Gambar 2.18*ScreenshootVisual Basic* 2010

*Visual basic*2010 mempunyai beberapa tambahan komponen, antara lain:

1. *Toolbox*

   Toolbox merupakan komponen lingkungan kerja VB yang berisikan *tool*s untuk ditempatkan di *form*. Dalam membuat sebuah aplikasi, maka komponen-komponen tersebut ditempatkan pada*form* dan menjadi komponen jendela program.

2. *Windows Project*

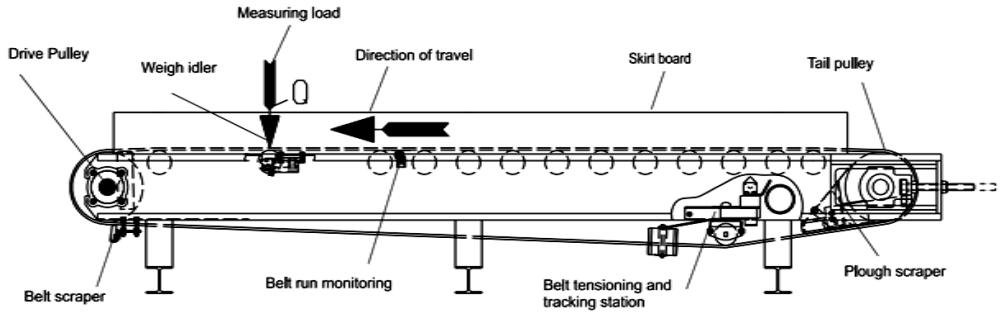
   Window Project berfungsi untuk menampilkan daftar *form* dan modul yang terdapat di *project* aplikasi yang sedang dikerjakan.

3. *Windows Properties*

Window Properties berfungsi untuk menampilkan daftar properti dari sebuah komponen yang sedang aktif. Kita dapat mengubah properti dari sebuah komponen dengan cara mengaktifkan (mengklik/memilih) komponen tersebut, kemudian mengubah nilai propertinya di Window Properties.

2.8 *Weighfeeder Conveyor*

Weighfeeder conveyor dipakai untuk memindah material dengan putaran dari motor sebagai pengerak utama yang terhubung dengan *drum*/*pulley* yang diselubungi oleh sabuk/*belt*. Lebar dan panjangnya *belt* menyesuaikan dengan kebutuhan atau kapasitas angkut serta jarak angkut material tersebut. *Weighfeeder conveyor* seperti terlihat pada gambar 2.19, memiliki instrumentasi pendukung yang dapat mengukur laju aliran material. Untuk mengukur berat material menggunakan sensor berat *load cell*, sedangkan sebagai pengukur kecepatan *belt*menggunakan *tachometer*. Dari berat dan kecepatan tersebut maka dapat diketahui laju material yang melalui *belt conveyor*.



Gambar 2.19 Ilustrasi*weighfeeder conveyor*[7]

Untuk mengukur laju aliran material dihitung dengan formula sebagai berikut:

$$I = Q * V \quad (2.89)$$

Dengan

I : Laju material dalam kilogram per menit (kg/m).

Q : *Belt load*/beban sabuk dalam kilogram per meter (kg/m).

V : *Belt speed*/kecepatan sabuk dalam meter per *second* (m/s)

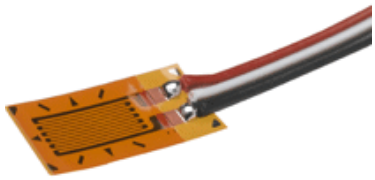
2.9 *Load cell*

*Load cell*seperti terlihat pada gambar 2.20 adalah suatu alat transduser yang menghasilkan *output* proporsional dengan beban atau gaya yang diberikan. *Load cell* dapat memberikan pengukuran yang akurat dari gaya dan beban. *Load cell* digunakan untuk mengkonversikan regangan pada logam ke tahanan variabel (*variable resistance*)[8].

Gambar 2.20*Load cell*

Load cell dikelompokkan sebagai transduser gaya (*force transducer*). Alat ini mengubah gaya atau beban/berat menjadi sinyal elektrik. Bagian utama dari

*load cell*adalah *strain gage*.*Strain gage* adalah sebuah alat yang memiliki nilai tahanan yangdapat berubah apabila alat mengalami penekanan.



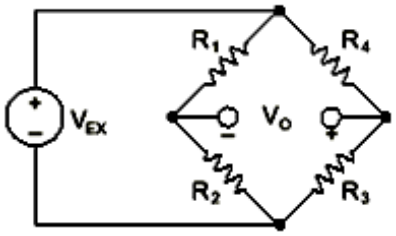
Gambar 2.21*Strain gage*

*Strain gage*seperti terlihat pada gambar 2.21 adalah bagian yang sangat penting dari sebuah *load cell*. Fungsi dari*strain gage* adalah untuk mendeteksi besarnya perubahan, dalam hal ini berupa dimensi jarak, yang disebabkan oleh suatu elemen gaya. *Strain gage*menghasilkan perubahan nilai tahanan yang proporsional dengan perubahan panjang atau jarak (*length*). Pada umumnya *strain gage* dipasang sebagaibagiandari rangkaian jembatan *Wheatstone* untuk aplikasi sirkuit elektrik.

Pada*strain gage* terjadi perubahan tahanan R yang akan diukur dan perubahan ini biasanya kecil. Jika mengukur sebuah perubahan tahanan yang kecilmaka perubahan tegangan juga kecil. Sebagai contoh, jika dialirkan arus sebesar 5 mA melalui sebuah *strain gage* bertahanan 120 Ω, tegangan yang melintasi*gage* akan dihasilkan 0,6 V. Jika tahanan berubah sebesar 1 mΩ, tegangan akan berubah menjadi 5 μV. Untuk menampilkan perubahan itu, diutuhkan penguatantegangan dengan sebuah faktor pengali. Karena sangat sulit untuk mendeteksi perbedaan tegangan yang sangat kecil,oleh karena itu

dibutuhkan sirkuit yang memungkinkan untuk hanya memperkuat perbedaan dalam tegangan yang melintasi *strain gage* disebabkan oleh perubahan tahanan. Solusinya ditemukan dalam *bridge circuit*.

Strain gage dalam jumlah banyak disambungkan untuk menciptakan keempat kakidari konfigurasi jembatan *wheatstone*. Ketika tegangan*input* dikenakan pada jembatan, timbul tegangan *output* yang proporsional atau sesuai dengan beban yang ditimbulkan atau diberikan. *Output* ini dapat diperkuat dan diproses dengan menggunakan peralatan-peralatan elektrik konvensional.



Gambar 2.22 *Wheatstone bridge*

Tegangan *output* Vo pada gambar 2.22 dapat di hitung dengan formula:

$$Vo = \left[\frac{R_3}{R_3 + R_4} - \frac{R_2}{R_1 + R_2}\right] V_{EX} (2.90)$$

Konfigurasi *bridge* dan *strain gage* dapat menghasilkan *output* yang berupa tegangan tingkat rendah (*low-level voltage*) yang nilainya tergantung pada nilai tegangan *input* (yang dipertahankan pada level yang konstan dan stabil) dan besarnya regangan atau beban yang diberikan pada *load cell* tersebut. Pada kondisi beban penuh, nilai nominal tegangan *output* dengan menggunakan *input* tegangan bernilai 10 volt dc adalah berkisar antara 20 milivolt sampai 30 milivolt.

Untuk menguji tahanan *bridge* pada *load cell*, dapat dilakukan dengan mengukur tahanan pada setiap pasang kabel *input* dan *output*. Nilai tahanan *output* biasanya berkisar pada 350 ± 3 ohm. Nilai tahanan *input* biasanya berkisar pada 415 ± 15 ohm.

Load cell memiliki ciri kerusakan fisik berupa perubahan pada kesetimbangan nol (*zero balance*) disebabkan karena tegangan sisa pada daerah penginderaan (*sensing area*) dari kelenturan (*flexure*), tegangan sisa terjadi sebagai akibat dari kelebihan beban (*overload*) pada *load cell*, dan/atau dari siklus pemakaian yang berjumlah jutaan. Kerusakan ini dapat diidentifikasi dengan milivoltmeter, dengan mengukur *outputload cell* pada kondisi tanpa beban (*no load*). *Output* yang dihasilkan seharusnya berada pada ± 10% dari *output* rata-rata (*rated output*).

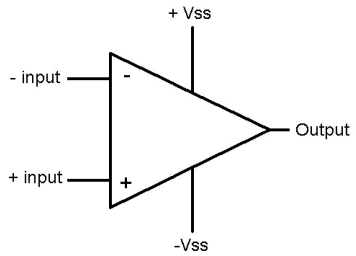
Apabila *outputload cell* tertentu = 2 mV/V dengan *power supplyoutput* = 10V, maka *output* dengan skala penuh bernilai 20 V. Apabila*zero balance* untuk *load cell* tersebut adalah ± 10 % dari *rated output*, maka pembacaan nol (*zero reading*) seharusnya berada pada ± 2 mV/V.Apabila *output* lebih besar dari toleransi *zero balance*, maka *load cell* tersebut rusak.

2.10 Pengkondisi Sinyal

Sensor berat*load cell* membutuhkan rangkaian elektronik sebagai pengkondisi sinyal untuk mengkonversi kedalam sinyal elektrik. Metode paling sederhana dari pengkondisi sinyal adalah pengubahan level sinyal. Rangkaian pengkondisi sinyal ini menggunakan IC jenis*Operationalamplifier*(op –amp)yang

dapat menguatkan atau melemahkan level tegangan. Suatu faktor penting dalam pemilihan sebuah *amplifier* adalah impedansi *input* yang dimiliki *amplifier* atau elemen-elemen lain yang menjadi *input*.

Op - amp merupakan jenis IC yang memiliki spesifikasi dan karakteristik khusus dalam operasinya. Op - amp memiliki satu *output* dan dua *input,* yaitu *input* negatif dan *input* positif (lihat gambar 2.23). Dengan spesifikasi dan rangkaian tertentu op- amp dapat menguatkan atau melemahkan level tegangan hingga ribuan kali dari tegangan *input*-nya.



Gambar 2.23 Skema/simbol op-amp

Terdapat berbagai macam jenis rangkaian *amplifier* yang dapat diaplikasikan sebagai pengkondisi sinyal diantarnya :

1. *Inverting amplifier*.
2. *Non - inverting amplifier*.
3. *Adder* / penjumlah.
4. *Differential amplifier*
5. *Instrument amplifier*.

Op-amp pada dasarnya adalah sebuah *differential amplifier* (penguat diferensial) yang memiliki dua masukan. *Input* op-amp seperti yang telah

dimaklumi ada yang dinamakan *input* inverting dan non-inverting. Op-amp ideal memiliki *open loop gain* (penguatan *loop* terbuka) yang tak terhingga besarnya. Seperti misalnya op-amp LM741 yang sering digunakan oleh banyak praktisi elektronika, memiliki karakteristik tipikal *open loop gain* sebesar $10^4 \sim 10^5$. Penguatan yang sebesar ini membuat op-amp menjadi tidak stabil, dan penguatannya menjadi tidak terukur (*infinite*). Disinilah peran rangkaian *negative feedback* (umpanbalik negatif) diperlukan, sehingga op-amp dapat dirangkai menjadi aplikasi dengan nilai penguatan yang terukur (*finite*). Impedasi *input* op-amp ideal mestinya adalah tak terhingga, sehingga mestinya arus *input* pada tiap masukannya adalah 0. Sebagai perbandingan praktis, op-amp LM741 memiliki impedansi *input* $Z_{in} = 10^6$ Ohm. Nilai impedansi ini masih relatif sangat besar sehingga arus *input* op-amp LM741 mestinya sangat kecil.

Ada dua aturan penting dalam melakukan analisa rangkaian op-amp berdasarkan karakteristik op-amp ideal. Aturan ini dalam beberapa literatur dinamakan *golden rule*, yaitu :

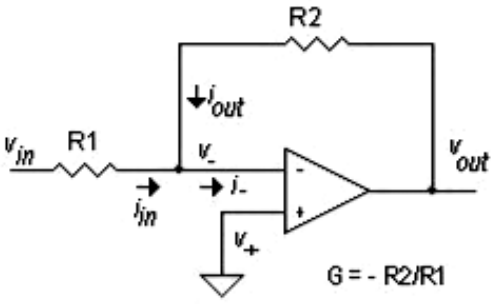
Aturan 1 : Perbedaan tegangan antara *input* v_+ dan v_- adalah nol ($v_+ - v_- = 0$ atau $v_+ = v_-$).

Aturan 2 : Arus pada *input* Op-amp adalah nol ($i_+ = i_- = 0$).

2.10.1 *Inverting Amplifier*

Rangkaian dasar penguat inverting adalah seperti yang ditunjukkan pada gambar 2.24, dimana sinyal masukannya dibuat melalui *inputinverting*. Seperti tersirat pada namanya, pembaca tentu sudah menduga bahwa fase keluaran dari

penguat inverting ini akan selalu berbalikan dengan *input*-nya. Pada rangkaian ini, umpanbalik negatif di bangun melalui resistor R2.



Gambar 2.24*Inverting amplifier*

Inputnon-inverting pada rangkaian ini dihubungkan ke *ground,* atau $v_+ = 0$. Dengan mengingat dan menimbang aturan 1 (lihat aturan 1), maka akan dipenuhi $v_- = v_+ = 0$. Karena nilainya = 0 namun tidak terhubung langsung ke ground, *input* op-amp v_- pada rangkaian ini dinamakan *virtual ground*. Dengan fakta ini, dapat dihitung tegangan jepit pada R1 adalah $v_{in} – v_- = v_{in}$ dan tegangan jepit pada reistor R_2 adalah $v_{out} – v_- = v_{out}$. Kemudian dengan menggunakan aturan 2, di ketahui bahwa :

$i_{in} + i_{out} = i_- = 0$, karena menurut aturan 2, arus masukan op-amp adalah 0.

$i_{in} + i_{out} = v_{in}/R_1 + v_{out}/R_2 = 0$

Selanjutnya,

$v_{out}/R_2 = - v_{in}/R_1$ .... atau$v_{out}/v_{in} = - R_2/R_1$

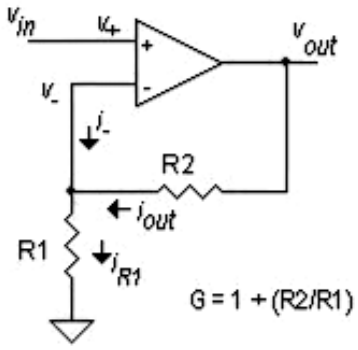
Jika penguatan G didefenisikan sebagai perbandingan tegangan keluaran terhadap tegangan masukan, maka dapat ditulis

$$G = v_{out}/v_{in} = - R_2/R_1 \quad (2.91)$$

Impedansi rangkaian inverting didefenisikan sebagai impedansi *input* dari sinyal masukan terhadap ground. Karena *inputinverting* (-) pada rangkaian ini diketahui adalah 0 (*virtual ground*) maka impendasi rangkaian ini tentu saja adalah $Z_{in} = R_1$.

2.10.2 *Non-Inverting amplifier*

Prinsip utama rangkaian penguat *non-inverting* adalah seperti yang diperlihatkan pada gambar 2.25 berikut ini. Seperti namanya, penguat ini memiliki masukan yang dibuat melalui *input non-inverting*. Dengan demikian tegangan keluaran rangkaian ini akan satu fasa dengan tegangan *input*-nya. Untuk menganalisa rangkaian penguat op-amp non inverting (lihat gambar 2.25), caranya sama seperti menganalisa rangkaian *inverting*.



Gambar 2.25*Non-inverter amplifier*

Dengan menggunakan aturan 1 dan aturan 2, kita uraikan dulu beberapa fakta yang ada, antara lain :

$v_{in} = v_+$

$v_+ = v_- = v_{in}$ ..... lihat aturan 1.

Dari sini ketahui tegangan jepit pada R_2 adalah $v_{out} - v_- = v_{out} - v_{in}$, atau $i_{out} = (v_{out}-v_{in})/R_2$. Lalu tegangan jepit pada R_1 adalah $v_- = v_{in}$, yang berarti arus $i_{R1} = v_{in}/R_1$.

Hukum Kirchof pada titik *inputinverting* merupakan fakta yang mengatakan bahwa :

$i_{out} + i_{(-)} = i_{R1}$

Aturan 2 mengatakan bahwa $i_{(-)} = 0$ dan jika disubsitusi ke rumus yang sebelumnya, maka diperoleh $i_{out} = i_{R1}$ dan Jika ditulis dengan tegangan jepit masing-masing maka diperoleh $(v_{out} - v_{in})/R_2 = v_{in}/R_1$ yang kemudian dapat disederhanakan menjadi :

$v_{out} = v_{in} (1 + R_2/R_1)$

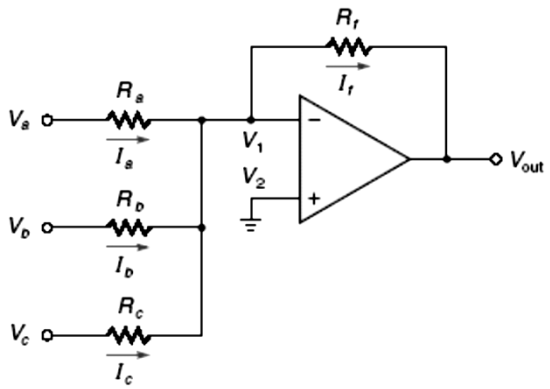
Jika penguatan G adalah perbandingan tegangan keluaran terhadap tegangan masukan, maka didapat penguatan op-amp *non-inverting*:

$G = v_{out}/v_{in} = 1 + (R_2/R_1)$ (2.92)

Impendasi untuk rangkaian op-amp *non-inverting* adalah impedansi dari *inputnon-inverting* op-amp tersebut.

2.10.3 *Adder*/Penjumlah

Rangkaian penjumlah atau rangkaian *adder* adalah rangkaian penjumlah yang dasar rangkaiannya adalah rangkaian *inverting amplifier* dan hasil *output*-nya adalah dikalikan dengan penguatan seperti pada rangkaian *inverting*. Pada dasarnya nilai *output*-nya adalah jumlah dari penguatan masing-masing dari *inverting*, seperti terlihat pada gambar 2.26:



Gambar 2.26*Inverting adder*

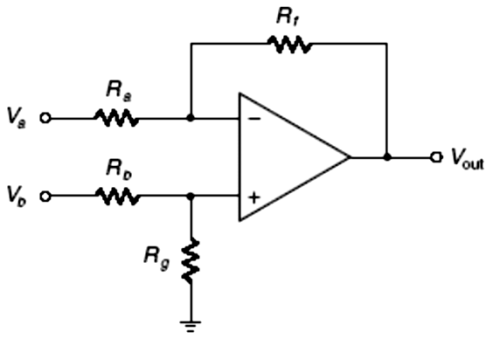
Dari rangkaian pada gambar 2.26 dapat dirumuskan sebagai berikut:

$$V_{out} = -\left(\frac{R_f}{R_a}V_a + \frac{R_f}{R_b}V_b + \frac{R_f}{R_c}V_c\right) \quad (2.93)$$

2.10.4 *Differential Amplifier*

Differential amplifier merupakan *amplifier* yang digunakan untuk mencari selisih tegangan dari dua sinyal yang masuk. Dari rangakaian pada gambar 2.27 persamaan tegangan *output*-nya dinyatakan sebagai berikut:

$$V_{out} = \frac{R_f}{R_a}(V_b - V_a) \quad (2.94)$$



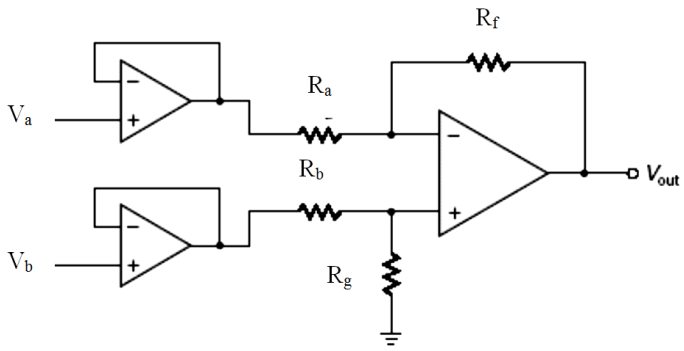
Gambar 2.27*Differential amplifier*

Tegangan *output* merupakan tegangan tunggal yang mengacu pada *ground*biasanya disebut *single-ended-voltage*.

2.10.5 *Instrument Amplifier*

Instrumentation amplifier adalah jenis d*ifferential amplifier* yang pada *input*-nya dihubungkan dengan sebuah *voltage follower*. Kegunaan dari *voltage follower*ini adalah:

1. Meningkatkan tahanan *input* sehingga sumber (seperti sensor) tidak mengalami beban turun.
2. Membuat kedua tahanan *input* sama.
3. Penguatan tidak ditentukan oleh tahanan dari sumber.



Gambar 2.28*Instrument amplifier*

Instrument amplifier pada gambar 2.28 dapat dirumuskan sebagai berikut:

$$V_{out} = (V_b - V_a)\frac{R_f}{R_a} \quad (2.95)$$

Untuk $R_a = R_b = R_f = R_g$.