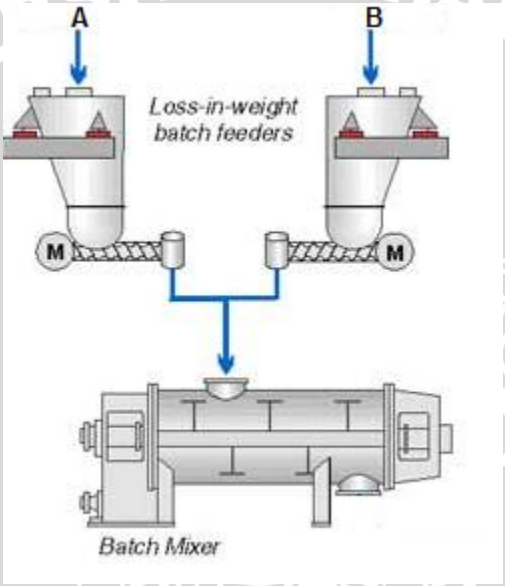
BAB II

TINJAUAN PUSTAKA

2.1 *Mixing and Blending System*

Dalam sistem *mixing* dan *blending* ada pilihan jenis *mixing* dan *blending* yang dapat dipilih suatu perusahaan pada proses produksinya. Pilihan tersebut adalah *batch* dan *continuous mixing and blending system*.

2.1.1 *Batch mixing*



Gambar 2.1 Ilustrasi *Batch mixing*
(Sumber : Anindya & Rissa, 2011)

Pada Gambar 2.1 adalah salah satu proses yang paling umum digunakan dalam industri farmasi yaitu *batch mixing*. Optimalisasi proses pencampuran membutuhkan pemahaman tentang mekanisme dan variabel kritis. Meskipun bubuk kohesi dan ukuran *mixer* dan geometri mungkin tidak dimodifikasi karena kendala lain, kondisi operasi seperti tingkat rotasi dan tingkat mengisi lebih mudah untuk dimodifikasi. Dengan demikian, pemahaman tentang interaksi antar variabel penting untuk dilakukan.

Dengan *Batch mixing*, bahan-bahan dimasukkan secara semi-otomatis. Dimisalkan akan dilakukan pencampuran 2 macam bahan yaitu A 1 liter, B 2 liter. Ketika *mixing* dilakukan secara *batch* maka A diukur terlebih dahulu sebanyak 1 liter, B diukur juga sebanyak 2 liter kemudian kedua bahan tersebut dimasukkan ke mesin *mixing* dan dilakukan proses *mixing*.

2.1.2 *Continuous Mixing*

Pada *continuous mixing*, bahan mengalir dari proses hulu ke dalam *mixer*, kemudian ditahan dalam bejana *mixer* selama waktu *mixing* yang ditentukan, dan dihentikan saat rasio aliran yang sama pada hilir. Pada beberapa aplikasi, *continuous mixing* memiliki kelebihan yang jelas dibandingkan dengan *batch mixing*. Untuk kapasitas proses yang tinggi, *continuous mixer* lebih ringkas daripada *batch mixer*. *Continuous mixing* digunakan untuk mencampur bahan-bahan secara kontinyu di dalam *mixer*. Pada proses *continuous mixing*, langkah-langkah untuk menimbang, memasukkan, mencampur, dan menghentikan pencampuran terjadi secara kontinyu dan serempak.

Bahan-bahan yang akan dicampur secara kontinyu dimasukkan ke dalam *mixer* sesuai perumusan yang telah ditentukan. Proses pemberian bahan dalam *continuous mixer* sangat kritis dan dapat secara signifikan mempengaruhi kualitas hasil akhir pencampuran. Gambar 2.2 menunjukkan proses *radial mixing* dan *axial mixing* mengambil tempat sebagai perjalanan bahan-bahan dari titik pengisian ke titik penghentian. Waktu yang diberikan oleh bahan untuk berjalan dari titik pengisian ke titik penghentian diketahui sebagai waktu memori bahan dalam *mixer*. Tidak seperti pada *batch mixer* yang mana waktu memori produk secara hati-hati dikontrol, dengan *continuous mixer*, waktu memori bahan tidak diseragamkan dan dapat secara langsung mempengaruhi kecepatan *mixer*, *feed rate*, *mixer geometry*, dan desain internal dari *mixer*. Bahan-bahan secara kontinyu berhenti saat rasio konstan yang secara umum dimasukkan sebagai kapasitas dari

continuous mixer. Kapasitas tersebut dihitung dalam kg/jam dari produk yang dicampur. Untuk mempertahankan *track* dari kualitas pencampuran, baiknya didefinisikan contoh dan testing prosedur bahan untuk diamati.

Dalam disertasi Patricia Maribel Portillo (2008) menyatakan bahwa *Continuous mixing* adalah alternatif pertimbangan efektif dari proses-proses *batch mixing* yang dalam prinsipnya memberikan control secara *online* yang lebih mudah dan optimasi dari performansi mixing.

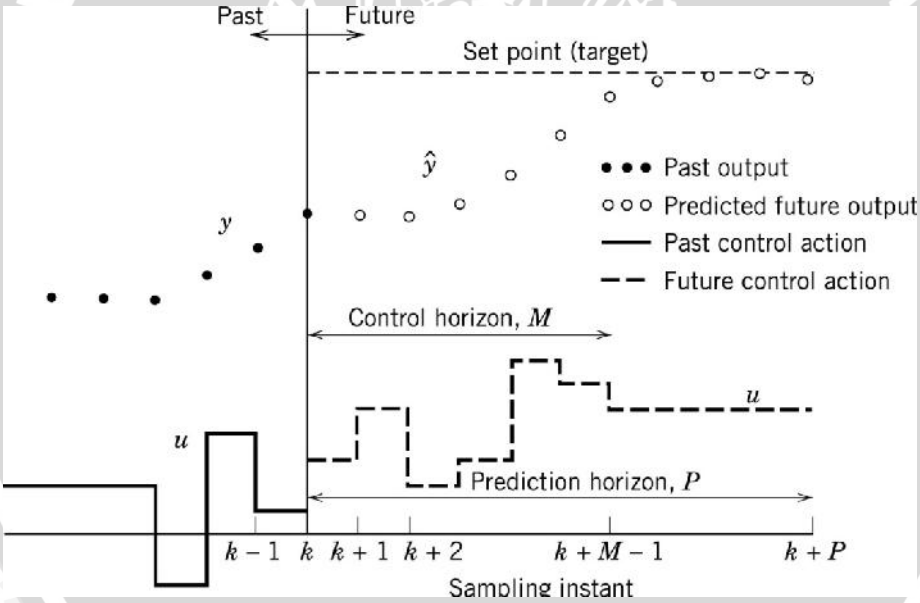
Gambar 2.2 Ilustrasi *Continuous Mixing*
(Sumber : Anindya & Rissa, 2011)

2.2 Model Predictive Control (MPC)

MPC adalah metode aljabar linear untuk memprediksi sinyal urutan manipulasi variabel kontrol. Setelah hasil manipulasi telah diperkirakan, kontroler kemudian dapat dilanjutkan dengan sinyal yang memberikan hasil yang diinginkan. MPC adalah sarana yang digunakan secara luas untuk menangani masalah besar kontrol multivariabel yang disertai dengan kendala-kendala di industri. Tujuan utama dari MPC adalah untuk minimalkan kriteria kinerja di masa mendatang yang mungkin akan tunduk pada kendala pada input dan output termanipulasi, di mana perilaku masa mendatang dihitung berdasarkan model plant. Kontroler model prediktif menggunakan model dan pengukuran plant sekarang untuk menghitung gerak masa mendatang dalam variabel independen yang akan menghasilkan

operasi yang menerima semua kendala variabel independen dan dependen. MPC kemudian mengirimkan variabel ini bergerak bebas ke *controller regulatory* yang sesuai set-point yang akan dilaksanakan dalam proses.

MPC pada dasarnya bekerja pada tiga horizon berurutan: horizon optimalisasi, horizon kendali dan horizon prediksi. Sebuah optimalisasi on-line menempati pada horizon optimalisasi sambil menghitung masukan kendali mendatang pada horizon kendali dan lintasan keluaran yang akan datang dihitung pada horizon prediksi. Kemudian kontroler melaksanakan masukan kendali yang pertama, dan menghitung kembali deretan masukan kendali berikutnya pada langkah waktu selanjutnya. Karena intensitas perhitungan akibat optimalisasi *on-line*, maka MPC secara tradisional dimanfaatkan untuk proses dengan lebar jalur (*bandwith*) yang rendah.



Gambar 2.3 Konsep dasar untuk Model Predictive Control

(Sumber : Tarmukan & Donny, 2005)

Pada Gambar 2.3 menjelaskan konsep dasar kontrol prediksi untuk plant dengan *single-input, single output* (SISO), diasumsikan dalam waktu diskrit, waktu sekarang dilabelkan sebagai waktu langlah k, *output* plant waktu sekarang y(k), dan juga menunjukkan sejarah masa lampau lintasan

output juga menunjukkan lintasan setpoint yang mana idealnya *output* mengikuti lintasan tersebut. Nilai dari lintasan setpoint pada suatu waktu t dinotasikan sebagai s(t). Lintasan selain lintasan setpoint adalah lintasan referensi. Lintasan referensi dimulai pada *output* masa sekarang y(k), dan didefiniskan lintasan ideal sepanjang plant seharusnya kembali pada lintasan setpoint, misalnya setelah gangguan terjadi. Maka lintasan referensi didefinisikan sebagai aspek penting perilaku loop tertutup plant terkontrol.

Sebuah hukum *model predictive control* memiliki komponen-komponen dasar prediksi, optimasi, dan implementasi *receding horizon*. (Tarmukan & Donny, 2005)

Masukan kendali mendatang serta respons plant yang akan datang diprediksi dengan menggunakan model dari sistem dan dioptimalkan dalam interval tetap berkaitan dengan indeks kinerja. Meskipun menjadi sangat sederhana untuk desain dan implementasi, algoritma MPC bisa mengendalikan sistem skala besar dengan banyak variabel kendali, dan yang lebih penting MPC memberikan sebuah metoda yang sistematik berkenaan dengan kendala-kendala pada masukan dan keadaan. Kendala seperti itu ada dalam semua aplikasi teknik kendali dan merupaka batasan pada aktuator dan keadaan plant yang timbul dari fisik, ekonomi atau kendala keamanan. Di MPC kendala ini diperhitungkan secara eksplisit dengan menyelesaikan dalam waktu nyata untuk menentukan masukan prediksi optimal. Dinamika plant non-linier juga bisa dilakukan dalam model prediksi.

2.2.1 Strategi Kendali Prediktif

Asas *model predictive control* mengandung komponen dasar prediksi, implementasi optimisasi dan *receding horizon*.

2.2.1.1 Prediksi

Respons plant yang akan datang diprediksi dengan menggunakan model dinamik. Dari persamaan ruang-keadaan berikut ini,

$$x(k+1) = Ax(k) + Bu(k) \tag{2.1}$$

Dimana x(k) dan u(k) adalah vektor-vekor model *state* dan masukan pada saat *sampling* yang ke-k. Diketahui urutan masukan terprediksi, maka urutan keadaan prediksi yang bersesuaian dibangkitkan dengan mensimulasikan model maju di atas horizon prediksi, dalam interval sampling N.

$$\mathbf{u}(k) = \begin{bmatrix} u(k \mid k) \\ u(k+1 \mid k) \\ \text{M} \\ u(k+N-1 \mid k) \end{bmatrix}; \qquad \mathbf{x}(k) = \begin{bmatrix} x(k+1 \mid k) \\ x(k+2 \mid k) \\ \text{M} \\ x(k+N \mid k) \end{bmatrix} \tag{2.2}$$

$u(k+i \mid k)$ dan $x(k+i \mid k)$ menyatakan vektor masukan dan keadaan pada saat k + i yang diprediksi pada saat k, dan $x(k+i \mid k)$ berkembang menurut model prediksi:

$$x(k+i+1 \mid k) = Ax(k+i \mid k) + Bu(k+i \mid k), \qquad i = 0, 1, \ldots \tag{2.3}$$

Dengan kondisi awal (pada awal horizon prediksi) didefinisikan $x(k \mid k) = x(k)$. Dengan demikian,

$$x(k+1 \mid k) = Ax(k) + Bu(k \mid k) \tag{2.4}$$

$$x(k+2 \mid k) = A^2x(k) + ABu(k \mid k) + Bu(k+1 \mid k) \tag{2.5}$$

Atau

$$x(k+i \mid k) = A^i x(k) + C_i \mathbf{u}(k) \ , \quad \text{i} = 0, 1, \ldots, \text{N} \tag{2.6}$$

$$\text{x(k)} = \text{Mx(k)} + \text{Cu(k)}, \text{ dimana } \ M = \begin{bmatrix} A \\ A^2 \\ \text{M} \\ A^N \end{bmatrix} \tag{2.7}$$

$$C = \begin{bmatrix} B & 0 & \Lambda & 0 \\ AB & B & \Lambda & 0 \\ \text{M} & \text{M} & \text{O} & \text{M} \\ A^{N-1}B & A^{N-2}B & \Lambda & B \end{bmatrix} \quad \begin{matrix} C_0 = 0, \\ C_i = \text{blok ke - i baris } C \end{matrix} \tag{2.8}$$

2.2.1.2 Optimasi

Asas *feedback* kendali prediktif dihitung dengan meminimalkan *cost* kinerja prediksi, yang didefinisikan menurut urutan prediksi u, x. Dalam kasus ini dinyatakan dalam bentuk umum *cost* kuadratik:

$$J(k) = \sum_{i=0}^{N-1} \left[x^T(k+i \mid k) Q x(k+i \mid k) + u^T(k+i \mid k) R u(k+i \mid k) \right] + x^T(k+N \mid k) \overline{Q} x(k+N \mid k)$$

(2.9)

Dengan Q dan R adalah matriks definit positif (Q boleh semi- defitit positif).Dalam hal ini memungkinkan untuk membuat matriks bobot $\overline{Q}$ yang berbeda untuk mengikut sertakan keadaan prediksi akhir x(k+N|k). Dengan memilih nilai $\overline{Q}$ yang tepat, hal ini membuat *cost* pada prediksi tak hingga untuk diikutkan dalam J(k). Jelaslah, J(k) adalah fungsi dari u(k), urutan masukan optimal untuk meminimalkan J(k) dinyatakan u*(k).

$$\mathbf{u}^*(k) = \arg\min_{\mathbf{u}} J(k) \tag{2.10}$$

Jika plant bermasalah dengan kendala masukan dan keadaan, maka ini bisa dilibatkan dalam optimalisasi sebagai kendala ekivalen pada u(k).

Substitusi untuk $x(k+i \mid k)$ akan memberikan,

Dengan demikian,

$$J(k) = \mathbf{u}^T(k) H \mathbf{u}(k) + 2x^T(k) F^T \mathbf{u}(k) + x^T(k) G x(k) \tag{2.11}$$

Dimana

$$H = C^T \tilde{Q} C + \tilde{R} \quad ; \quad F = C^T \tilde{Q} M \tag{2.12}$$

$$G = M^T \tilde{Q} M + Q \tag{2.13}$$

Dengan $\tilde{Q} = \begin{bmatrix} Q & 0 & \Lambda & 0 \\ 0 & O & & M \\ M & M & Q & 0 \\ 0 & \Lambda & 0 & \overline{Q} \end{bmatrix}$ dan $\tilde{R} = \begin{bmatrix} R & 0 & \Lambda & 0 \\ 0 & O & & M \\ M & M & R & 0 \\ 0 & \Lambda & 0 & R \end{bmatrix}$ (2.14)

Catatan bahwa matriks H, F, dan G bisa dihitung offline.

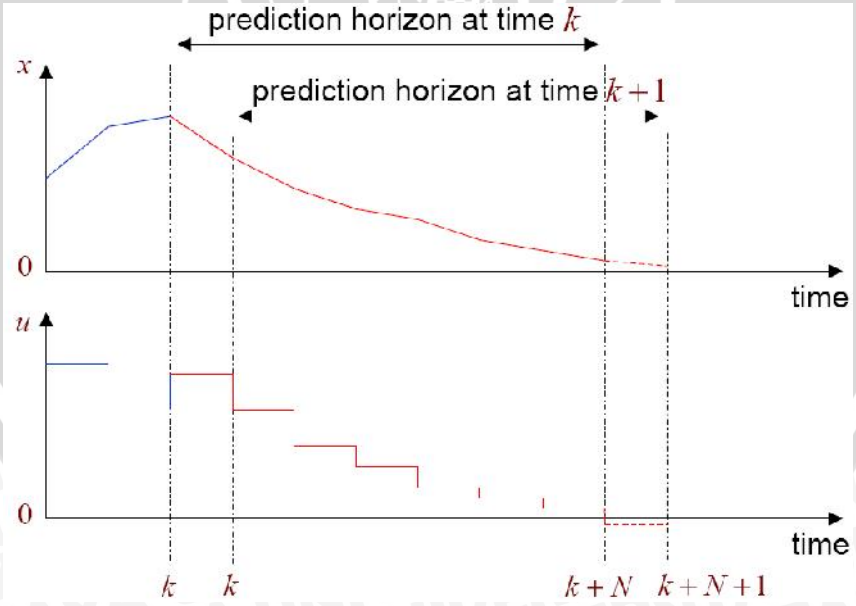
2.2.1.3 Implementasi Receding Horizon

Hanya elemen pertama dari urutan masukan prediksi optimal u*(k) adalah masukan untuk plant.

$$u(k) = u^*(k \mid k) \tag{2.15}$$

Proses penghitungan u*(k) dengan meminimalkan *cost* prediksi dan mengimplementasikan elemen pertama dari u* kemudian diulangi pada setiap sampling k = 0, 1, … . Untuk kejadian ini optimasi dengan menetapkan u* dikenal sebagai optimasi online.

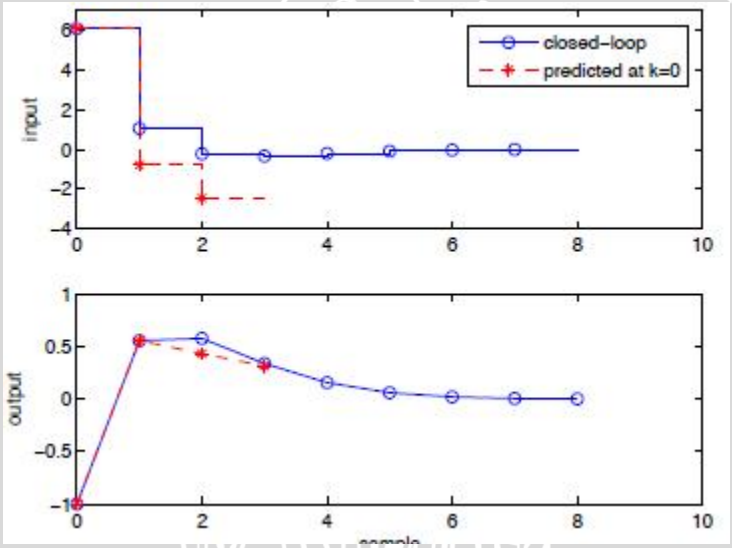
Horizon prediksi tetap dengan panjang yang sama walaupun optimasi berulang pada waktu yang akan dating dan pendekatan ini dikenal sebagai strategi *receding horizon* seperti ditunjukkan pada Gambar 2.4.



Gambar 2.4 Strategi receding horizon

(Sumber : Mark Cannon, 2012)

Karena prediksi keadaan x dan dari sini urutan masukan optimal u* tergantung pada pengukuran keadaan sekarang x(k), prosedur ini mengenalkan *feedback* menjadi asas MPC, jadi memberikan derajat kekokohan untuk galat pemodelan dan ketidakpastian. Keistimewaan kedua dari pendekatan *receding horizon* yaitu secara terus menerus menggeser horizon selama masukan yang akan datan dioptimalkan, ia mencoba untuk mengkompensasi suatu kenyataan bahwa horizon ini terbatas. Pada Gambar 2.5, menjelaskan bahwa respons prediksi bisa berbeda signifikan dari respons lup-tertutup. *Cost* dan kendala yang diberikan akan dirancang dengan benar, strategi *receding horizon* bisa memastikan bahwa kinerja dari system lup-tertutup setidaknya sebaik prediksi optimal.



Gambar 2.5 Respons lup-tertutup dan respons yang diprediksi

(Sumber : Mark Cannon, 2012)

Gambar 2.5 menunjukkan respons lup-tertutup dan respons yang diprediksi pada k = 0 untuk tanpa sistem orde kedua kendala dengan horizon N = 3.

2.2.2 Penanganan Kendala

Sangat luas kelas dari model plant yang bisa disatukan dengan strategi kendali prediktif. Ini termasuk model linier, nonlinier, waktu-diskrit dan waktu-kontinyu. Model prediksi bisa deterministic, stokastik, atau

fuzzy. Namun dalam tugas akhir ini, kamia hanya membatasi untuk model plant linier.

Kendala ketidaksamaan pada keadaan dan masukan seharusnya memenuhi dinamika model, kendala ketidaksamaan pada variabel masukan dan keadaan ditemui dalam setiap masalah kendali. Sambil menangani kendala ketidaksamaan ditangani secara implisit yaitu model plant digunakan untuk menuliskan trayektori keadaan prediksi sebagai fungsi dari kendala eksplisit dalam masalah optimisasi *online.*

2.2.2.1 Optimasi Tanpa Kendala

Tiadanya kendala, maka optimasi $\mathbf{u}^*(k) = \arg\min_{\mathbf{u}} J(k)$ mempunyai solusi bentuk tertutup yang bisa diturunkan dengan menganggap gradient J terhadap u:

$$\nabla_{\mathbf{u}} J = 2H\mathbf{u} + 2Fx \tag{2.16}$$

Jelaslah $\nabla_{\mathbf{u}} J = 0$ harus dipenuhi pada titik minimum *J*, dan karena *H* adalah definit positif (atau semi-definit positif), adanya u seperti itu $\nabla_{\mathbf{u}} J = 0$ adalah perlu sebuah titik minimum. Oleh karena itu u* adalah unik hanya bila H non-singular dan kemudian diketahui

$$\mathbf{u}^*(k) = -H^{-1}Fx(k) \tag{2.17}$$

Jika H adalah singular (semi-definit positif lebih baik daripada definit positif), maka u* adalah non-unik, dan solusi khusus dari $\nabla_{\mathbf{u}} J = 0$ harus didefinisikan sebagai $\mathbf{u}(k) = -H^{+}Fx(k)$ dimana H+ adalah *inverse* kiri dari H (sehingga H+H = I).

Implementasi elemen pertama dari prediksi optimal u*(k) pada tiap sampling k mendefinisikan asas *receding horizon control*. Karena H dan F adalah konstanta, ini adalah pengendali *feedback* linear *time-invariant* u(k) = K_Nx(k). dimana matriks penguatan K_N adalah baris pertama dari $-H^{-1}F$

untuk kasus satu-masukan (atau baris nu pertama untuk kasus u mempunyai dimensi nu), yakni

$$u(k) = u^*(k \mid k) = K_N x(k),\ K_N = -\begin{bmatrix} I_{n_u} & 0 & \Lambda & 0 \end{bmatrix} H^{-1} F \qquad (2.18)$$

Untuk sistem linier, ketergantungan prediksi x(k) pada u(k) adalah linier. *Cost* prediksi kuadratik adalah fungsi kuadratik dari urutan masukan u(k). Jadi J(k) bisa dinyatakan sebagai fungsi dari u dalam bentuk

$$J(k) = \mathbf{u}^T(k) H \mathbf{u}(k) + 2 f^T \mathbf{u}(k) + g \qquad (2.19)$$

Dengan H adalah matriks konstanta definit positif (atau memungkinkan semidefinit positif), dan f, g merupakan vektor dan scalar yang tergantung pada x(k). Kendala masukan dan keadaan linier demikian juga secara tidak langsung menyatakan bahwa u(k) yang bisa dinyatakan

$$A_C \mathbf{u}(k) \le b_C \qquad (2.20)$$

Dengan A_C adalah matriks konstanta dan tergantung pada bentuk kendala, vektor b_C bisa menjadi fungsi dari x(k). Optimalisasi MPC online terdiri dari minimalisasi di atas u dari obyektif kuadratik adalah untuk (tnduk kepada) kendala linier:

$$\underset{\mathbf{u}}{\text{minimisasi}}\ \mathbf{u}^T H \mathbf{u} + 2 f^T \mathbf{u} \qquad (2.21)$$

Tunduk kepada $A_C \mathbf{u}(k) \le b_C$

Kelas dari masalah optimalisasi ini dikenal sebagai masalah pemrograman kuadratik (QP), dan diketahui bahwa H adalah matriks definit positif dan kendala adalah linier.

2.2.2.2 Optimasi dengan Kendala

Macam-macam kendala dalam optimasi, yaitu :

- Kendala Masukan, umumnya timbul sebagai hasil dari aktuator, yakni saturasi.

  Kendala absolut, $\underline{u} \le u(k) \le \overline{u}$ (2.22)

  Atau kendala laju, $\underline{\Delta u} \le u(k) - u(k-1) \le \overline{\Delta u}$ (2.23)

- Kendala Keadaan. Kendala keadaan linier mempunyai bentuk umum sebagai berikut,

$$\underline{g_C} \le G_C x(k) \le \overline{g_C} \tag{2.24}$$

Dengan G_C adalah matriks konstanta dan $\underline{g_C}, \overline{g_C}$ adalah vector konstanta. Kendala keadaan bisa menjadi aktif selama transient.

- Kendala *hard/soft*. Kendala diklasifikasikan sebagai *hard* atau *soft*. Kendala *hard* harus selalu dipenuhi, dan jika ini tidak mungkin masalah *infeasible*. Dengan kata lain, kendala *soft* bisa dilanggar bila perlu untuk menghindari *infeasibility*.

2.2.3 Algoritma Pengendalian Plant dengan *Model Predictive Control*

Algoritma pengendalian plant sistem dengan metode *Model Predictive Control* adalah sebagai berikut,

1. Pemodelan plant sistem. Dalam hal ini sistem plant model linier,

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \tag{2.25}$$

Sistem dengan kendala,

$$\begin{cases} u_{\min} \le u(t) \le u_{\max} \\ y_{\min} \le y(t) \le y_{\max} \end{cases} \tag{2.26}$$

2. Mengukur dan mengoptimasikan *state* $x(k)$ pada saat k.

$$J(k) = \sum_{i=0}^{N-1} \left[x^T(k+i \mid k) Q x(k+i \mid k) + u^T(k+i \mid k) R u(k+i \mid k) \right] + x^T(k+N \mid k) P x(k+N \mid k)$$

$$\text{dengan } u_{\min} \le u(k) \le u_{\max}, k = 0,\ldots,N-1 \qquad y_{\min} \le y(k) \le y_{\max}, k = 1,\ldots,N \tag{2.27}$$

Dengan factor pembobotan, $Q = Q^T \ge 0$, $R = R^T > 0$, $P \ge 0$ dan P memenuhi persamaan Riccati,

$$K = -(R + B'PB)^{-1} B'PA$$
$$P = (A + BK)'P(A + BK) + K'RK + Q \tag{2.28}$$

$$J(x(0),U) = x'(0)Qx(0) + \begin{bmatrix} x^T(1) & x^T(2) & \text{K} & x^T(N-1) & x^T(N) \end{bmatrix} \underbrace{\begin{bmatrix} Q & 0 & 0 & \text{K} & 0 \\ 0 & Q & 0 & \text{K} & 0 \\ \text{M} & \text{M} & \text{O} & \text{M} & \text{M} \\ 0 & \text{K} & 0 & Q & 0 \\ 0 & 0 & \text{K} & 0 & P \end{bmatrix}}_{\bar{Q}} \begin{bmatrix} x(1) \\ x(2) \\ \text{M} \\ x(N-1) \\ x(N) \end{bmatrix} + \begin{bmatrix} u^T(0) & u^T(1) & \text{K} & u^T(N-1) \end{bmatrix} \underbrace{\begin{bmatrix} R & 0 & \text{K} & 0 \\ 0 & R & \text{K} & 0 \\ \text{M} & \text{M} & \text{O} & \text{M} \\ 0 & \text{K} & 0 & R \end{bmatrix}}_{\bar{R}} \begin{bmatrix} u(0) \\ u(1) \\ \text{M} \\ u(N-1) \end{bmatrix} \quad (2.29)$$

Model prediksi (*time invariant*):

$$x(k+i+1 \mid k) = Ax(k+i \mid k) + Bu(k+i \mid k) \quad (2.30)$$

Prediksi keadaan (*state*):

$$x(k \mid k) = x(k) \quad (2.31)$$

$$x(k+1 \mid k) = Ax(k) + Bu(k \mid k) \quad (2.32)$$

$$x(k+2 \mid k) = A^2 x(k) + ABu(k \mid k) + Bu(k+1 \mid k) \quad (2.33):$$

Jadi untuk waktu awal $k = 0$, maka estimasi *state* sekarang $x(k)$ menjadi

$$\begin{bmatrix} x(1) \\ x(2) \\ \text{M} \\ x(N) \end{bmatrix} = \underbrace{\begin{bmatrix} B & 0 & \text{K} & 0 \\ AB & B & \text{K} & 0 \\ \text{M} & \text{M} & \text{O} & \text{M} \\ A^{N-1}B & A^{N-2}B & \text{K} & B \end{bmatrix}}_{\bar{T}} \begin{bmatrix} u(0) \\ u(1) \\ \text{M} \\ u(N-1) \end{bmatrix} + \underbrace{\begin{bmatrix} A \\ A^2 \\ \text{M} \\ A^N \end{bmatrix}}_{\bar{T}} x(0) \quad (2.34)$$

$$J(x(0),U) = x'(0)Qx(0) + \left(\bar{S}U + \bar{T}x(0)\right)'\bar{Q}\left(\bar{S}U + \bar{T}x(0)\right) + U'\bar{R}U$$

$$= \frac{1}{2}U'\underbrace{2(\bar{R} + \bar{S}'\bar{Q}\bar{S})}_{H}U + x'(0)\underbrace{2\bar{T}'\bar{Q}\bar{S}}_{F}U + \frac{1}{2}x'(0)\underbrace{2(Q + \bar{T}'\bar{Q}\bar{T})}_{Y}x(0)$$

$H(k), F(k), Y(k)$ harus dihitung secara *online.*

3. Menentukan $U_0^*(x(k))$ dengan menyelesaikan problema optimasi.

$$J(x(0),U) = \frac{1}{2}x^T(0)Yx(0) + \min_U \frac{1}{2}U^T HU + x^T(0)F^T U$$
$$\text{dengan } GU \le W + Sx(0) \tag{2.35}$$

Optimal diperoleh dengan me-nolkan gradient

$$\nabla_U J(x(0),U) = HU + Fx(0) = 0 \tag{2.36}$$

Akhirnya didapatkan solusi $U^* = \begin{bmatrix} u^*(0) \\ u^*(1) \\ \text{M} \\ u^*(N-1) \end{bmatrix} = -H^{-1}Fx(0)$ (2.37)

4. Jika $U_0^*(x(k)) = 0$, maka problema tidak layak berhenti.
5. Gunakan elemen pertama u_0^* dari U_0^* ke sistem.
   Kemudian hanya memakai $u(k) = u^*(0)$ dan mengabaikan inputan optimal yang tersisa. Jadi, $u(k) = -[I \quad 0 \quad \text{K} \quad 0]H^{-1}Fx(k) \cong Kx(k)$.
6. Langkah ini kemudian dilanjutkan untuk waktu $k + 1$, dan seterusnya ….

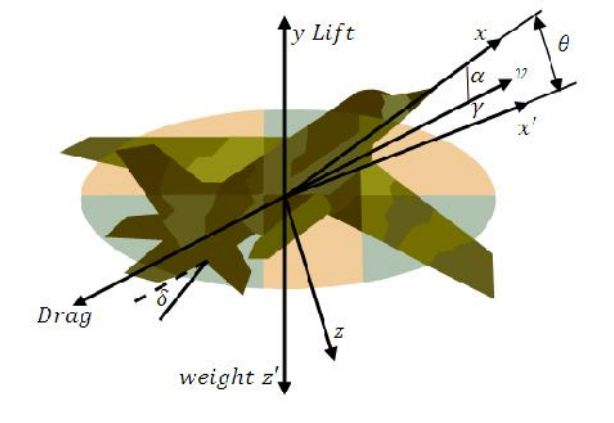
Sebagai contoh, plant sistem pengaturan *pitch* pesawat yang dikontrol dengan mengatur sudut (dan kemudian gaya angkat) belakang pesawat. Gaya aerodinamik (*Lift* dan *Drag*) maupun kelembaman pesawat yang diambil. Sistem ini adalah sistem orde tiga dan sistem nonlinear yang tidak stabil memiliki integrator bebas. Sumbu koordinat dan gaya yang terjadi pada pesawat ditunjukkan pada Gambar 2.6. Diasumsikan bahwa tinggi dan kecepatan pesawat dalam keadaan mantap.

Dalam bentuk State-space persamaan plant sistem sebagai berikut,

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \beta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} [\delta]$$

Output sistem adalah sudut *pitch*, persamaan output adalah,

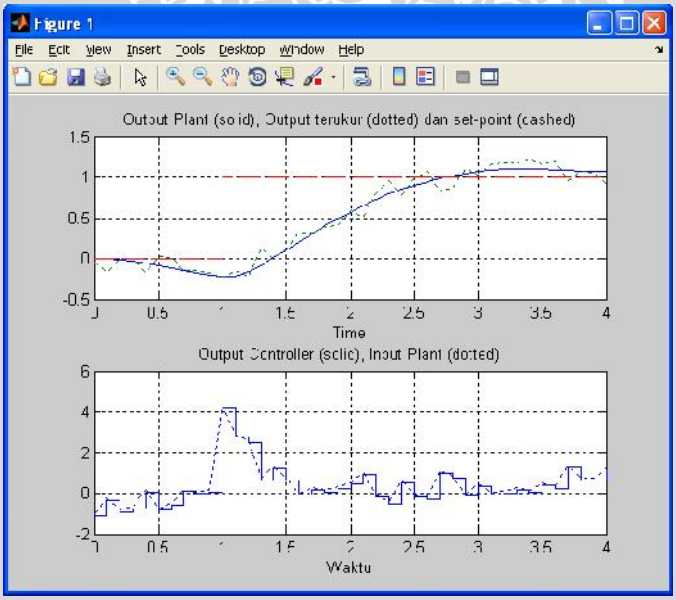
$$y = [0 \quad 0 \quad 1] \begin{bmatrix} \alpha \\ q \\ \beta \end{bmatrix} + [0][\delta]$$



Gambar 2.6 Sumbu Koordinat dan Gaya pada Pesawat

(Sumber : Carnegie Mellon, 1996)

Dengan menggunakan program Matlab pengendalian menggunakan *Model Predictive Control* yang terlampir, didapatkan plot yang ditunjukkan pada Gambar 2.7 sebagai berikut,



Gambar 2.7 Hasil Plot Plant Pengaturan Sudut *Pitch* Pesawat

(Sumber : Pengujian)

2.2.4 Perbedaan MPC dengan Kontroler Lain

Model kontrol prediktif menggunakan model matematis untuk mensimulasikan proses. Model ini kemudian mencoba input untuk

memprediksi perilaku sistem. Dengan cara ini, MPC adalah jenis kontrol umpan maju. Menggunakan masukan sistem sebagai dasar kontrol. MPC adalah lebih kompleks daripada kebanyakan kontrol umpan maju lainnya karena cara prediksi ini digunakan untuk mengoptimalkan proses dalam jumlah waktu tertentu. Kebanyakan tipe kontrol umpan maju tidak memperhitungkan *output* proses yang melewati waktu tinggal. Algoritma MPC akan membandingkan *output* yang diperkirakan dengan *output* yang diinginkan dan memilih sinyal yang akan meminimalkan perbedaan ini selama waktu yang dipilih. Jenis kontrol ini dapat melihat ke depan menjadi langkah-langkah beberapa waktu di masa depan untuk mengoptimalkan proses. Jenis pengendali PID normal menggunakan ekspresi matematika berdasarkan *error* dari *set point*. Pengaturan persamaan untuk pengendali MPC didasarkan pada *set point*, sifat sistem, dan hasil yang diinginkan serta optimasi.

MPC adalah sangat spesifik untuk proses itu pemodelan. Tidak seperti *set up rasio* atau kaskade kontrol, di mana ia mudah untuk mengimplementasikan dan mengubah setpoint dalam berbagai situasi, MPC akan memodelkan satu proses tertentu dan mengoptimalkannya. Seperti yang disebutkan sebelumnya ini dapat menjadi keuntungan atau kerugian. MPC sangat bagus untuk memilih salah satu jenis operasi pada satu sistem dan menyempurnakan ke kondisi yang diinginkan. Ini juga memiliki sisi buruk dalam persamaan model yang akan bekerja untuk satu dan hanya satu situasi.

2.2.5 Kelebihan *Model Predictive Control* (MPC)

Beberapa kelebihan yang didapatkan dalam menggunakan kontroler MPC adalah sebagai berikut,

1. MPC dapat digunakan untuk menangani program pengendalian multivariabel.
2. MPC dapat mempertimbangkan keterbatasan aktuator.

3. MPC dapat meningkatkan keuntungan dengan memungkinkan untuk operasi dekat dengan kendala sistem.
4. MPC dapat melakukan perhitungan *online* dengan cepat.
5. MPC dapat digunakan untuk fasa non-minimal dan proses yang tidak stabil.
6. MPC mampu menangani perubahan struktural.

2.2.6 Batasan *Model Predictive Control* (MPC)

Kekurangan yang dimiliki oleh kontroler MPC adalah sebagai berikut,

1. Karena metodenya linier, maka tidak disarankan untuk dipakai pada sistem non linier tinggi, misalnya kendali pH.
2. Tidak mengadaptasi perubahan proses. Jika proses berubah secara signifikan maka model MPC harus diidentifikasi kembali.
3. Untuk kasus-kasus tertentu *economic linier programming* bisa jadi tidak tepat untuk kasus-kasus ini karena perlu menambah optimalisasi *non linear* di atas *linear programming*.
4. Beberapa model MPC dibatasi hanya untuk proses stabil, loop terbuka.
5. MPC sering membutuhkan sejumlah besar koefisien model untuk menggambarkan tanggapan.
6. Beberapa model MPC diformulasikan untuk *output* gangguan, dan mereka tidak dapat menangani masukan gangguan juga.
7. Beberapa bentuk MPC menggunakan asumsi keluaran gangguan konstan. Ini mengoreksi kenyataannya bahwa *output* yang diprediksi oleh model ini tidak persis sama dengan *output* yang diukur sebenarnya. Metode ini mengasumsikan istilah koreksi adalah konstan di masa depan, yang mungkin tidak menghasilkan kinerja yang baik jika ada gangguan nyata pada masukan plant.

8. Jika horizon prediksi tidak dirumuskan dengan benar, kinerja kontrol akan menjadi miskin bahkan meskipun model sudah benar.
9. Beberapa sistem memiliki berbagai kondisi operasi yang sering berubah. Beberapa contoh ini termasuk reaktor eksotermik, proses *batch*, dan setiap sistem di mana konsumen yang berbeda memiliki spesifikasi produk yang berbeda. Model linier MPC tidak akan mampu menangani perilaku dinamis dari proses ini. Sebuah model *nonlinier* harus digunakan untuk kinerja kontrol yang lebih baik.

2.3 *Distributed Control System* (DCS)

2.3.1 Pendahuluan

Aplikasi awal komputer dalam bidang kontrol proses adalah sistem kontrol suvervisi dan monitoring pada stasiun pembangkit sistem tenaga sekitar tahun 1958. Evolusi selanjutnya adalah aplikasi komputer pada loop kontrolnya itu sendiri yang dikenal dengan nama *Direct Digital Control* (DDC) yang pertama kali diinstal di perusahaan petrokimia di Inggris sekitar tahun 1962. Pada sistem DDC tersebut, ada 224 variabel proses yang diukur dan 129 *valve* yang dikontrol secara langsung oleh komputer. Sekitar tahun 1975, untuk pertama kalinya Honeywell meluncurkan produk kontrol proses komersil yang diistilahkan sebagai *Distributed Control System* (DCS).

DCS ini pada dasarnya merupakan pengembangan sistem kontrol DDC. Perbedaan utama DCS dengan DDC terutama terletak dalam arsitektur sistem kontrol, dalam hal ini DCS adalah sistem kontrol yang mempunyai kesatuan kontrol yang terdistribusi dalam keseluruhan proses Industri. Pengertian terdistribusi dalam DCS meliputi terdistribusi secara geografis, resiko kegagalan operasi, dan fungsional.

2.3.2 Perbedaan DCS dan PLC

Tabel 2.1 Perbedaan PLC dan DCS untuk proses produksi

PLC	DCS
Tipikal produk: Barang (automasi manufacture dan perakitan)	Tipikal produk: Bahan (melibatkan kombinasi dan transformasi bahan mentah)
Proses produksi termonitor secara visual oleh operator	Proses produksi seringkali tidak terlihat oleh operator
Membutuhkan kontrol logika	Membutuhkan sistem kontrol regulator
Kontrol Batch sederhana: menghasilkan satu jenis produk, prosedur tetap,resep konstan -tidak pernah berubah	Kontrol Batch kompleks menghasilkan berbagai jenis produk, prosedur dapat berubah,resep variabel

Tabel 2.2 Perbedaan PLC dan DCS saat downtime

PLC	DCS
Nilai individual produk relative murah	Nilai material yang akan diolah dan hasil produk relative mahal
Downtime hanya menyebabkan kehilangan produksi	Downtime tidak hanya menyebabkan kehilangan produksi tetapi dapat menyebabkan keadaan yang berbahaya
Downtime tidak menyebabkan kerusakan peralatan proses	Downtime umumnya menyebabkan kerusakan peralatan proses dan produk
Jika terjadi kerusakan, waktu pemulihan relative cepat	Jika terjadi kerusakan, waktu pemulihan relative lambat

Pada tabel 2.1 adalah perbedaan antara pengendalian PLC dan DCS yang dapat dipertimbangkan oleh suatu industri dalam memproduksi suatu bahan. Pada tabel 2.2 menjelaskan perbedaan antara pengendalian PLC dan DCS dalam mempertimbangkan nilai atau harga yang harus dibayar saat terjadi downtime.

Jantung sistem pada PLC adalah kontroler sedangkan pada DCS adalah HMI (*Human Machine Interface*). Tabel 2.3 menunjukkan yang harus dilakukan oleh operator agar proses berjalan dengan lancar dalam menggunakan PLC dan DCS. Tabel 2.4 menggambarkan performansi yang

diharapkan pada PLC dan DCS. Juga perbedaan-perbedaan lainnya pada PLC dan DCS yang ditunjukkan oleh tabel 2.5.

Tabel 2.3 Tugas Operator mengoprasikan PLC dan DCS

PLC	DCS
Tugas utama operator adalah mengatasi proses abnormal	Interaksi operator dan proses sangat ketat: operator harus dapat membuat keputusan dan secara terus menurus berinteraksi dengan proses (terkait dengan target)
Informasi status (ON/Off, Run/Stop) sangat kritis bagi operator	Trend sinyal analog pada HMI memberikan informasi kritis bagi operator: Apa yang sedang terjadi pada proses?
Alarm proses produksi abnormal adalah informasi kunci bagi operator	Managemen alarm adalah kunci keamanan operasi proses

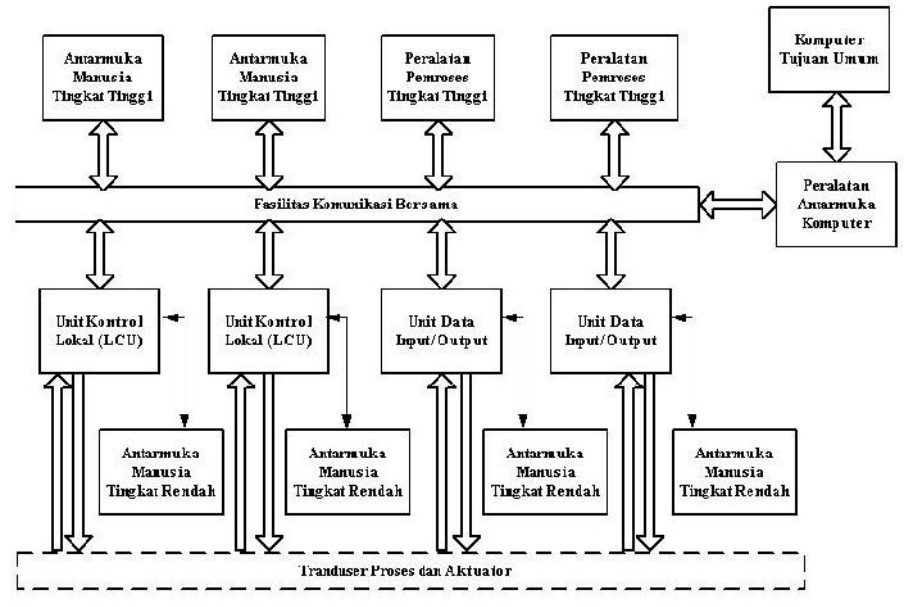
Tabel 2.4 Performansi pada PLC dan DCS

PLC	DCS
Eksekusi program kontrol relative cepat (10 ms)	Eksekusi program loop kontrol relative lambat (100-500 ms)
Sistem redudansi tidak selalu diperlukan	Sistem redudansi sangat penting
Untuk merubah konfigurasi sistem dapat dilakukan secara offline	Untuk merubah konfigurasi sistem harus dilakukan secara online
Sistem kontrol Analog: PID sederhana	Sistem kontrol Analog: sederhana sampai kompleks

Tabel 2.5 Perbedaan PLC dan DCS

PLC	DCS
Perancangan sistem kontrol bersifat Bottom Up	Perancangan sistem kontrol bersifat Top -down
Program aplikasi: diagram ladder	Program apliksi: blok fungsional
Rutin-rutin customisasi biasanya telah tersedia	Rutin-rutin umumnya bersifat kompleks
Training staff operator: less	Training staff/ operator :more (front up training)
Protokol komunikasi: Open	Protokol komunikasi: closed (propietary)
Pemeliharaan, jika terjadi kerusakan: Operator	Pemeliharaan, jika terjadi kerusakan: vendor

2.3.3 Konsep Distributed Control System (DCS)



Gambar 2.8 Diagram Blok DCS

Komponen-komponen yang digunakan pada DCS adalah sebagai berikut,

- Unit kontrol lokal *(Local Control Unit, LCU) :*
  Merupakan unit terkecil pada DCS yang mampu melaksanakan kontrol kalang tertutup. Contoh: PLC, mikroprosesor dll.
- Antarmuka manusia tingkat rendah *(Low-Level Human Interface, LLHI):*
  Peralatan yang digunakan untuk antarmuka antara operator atau orang instrumen dengan LCU secara langsung, misalnya mengubah *set-point*, mode kontrol, konfigurasi kontrol atau menala parameter kontrol.
- *Data Input/Output Unit (DI/OU):*
  Digunakan untuk antarmuka dengan proses dengan tujuan mengambil atau mengeluarkan data dan tidak melakukan aksi kontrol.
- Antarmuka manusia tingkat tinggi *(High-Level Human Interface, HLHI):*

Seperangkat peralatan yang mempunyai fungsi sama seperti LLHI tetapi dengan kemampuan lebih bagus dan pemakaian lebih nyaman untuk pengguna *(user friendliness).*

- Peralatan pemroses tingkat tinggi *(High-Level Computing Device, HLCD):*

  Sekumpulan perangkat berbasis mikroprosesor yang melakukan fungsi pengendalian terhadap plant.
- Peralatan antarmuka komputer *(Computer Interface Device, CID):*

  Sekumpulan perangkat yang digunakan untuk interaksi antara komputer dengan komponen DCS lain.
- Fasilitas komunikasi bersama *(Shared Communication Facilities):*

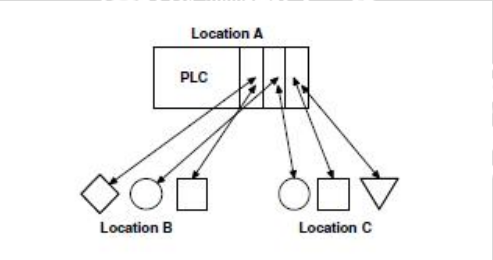
  Digunakan sebagai alat komunikasi data bersama antara peralatan DCS.

2.4 FIELDBUS

Teknologi *fieldbus* adalah langkah utama dalam industri kontrol proses. Dalam sistem yang lebih lama, sinyal-sinyal dikirim dulu menjadi unidirectional dan secara tradisional analog 4-20 mA. Teknologi *fieldbus* lainnya adalah pengganti tipe tersebut dan memperpanjang seluruh konsep kontrol proses dari pendekatan bawah-atas ke pemikiran atas-bawah yang lebih canggih. *Fieldbus* modern saat ini beraksi sebagai seluruh jaringan, kurang lebih seperti internet. *Fieldbus* tersebut menyambungkan semua dari PLC, *engineering* dan operator station ke peralatan *low level* dalam pabrik yang dikenal sebagai aktuator, sensor, tranduser, dan *drive*.

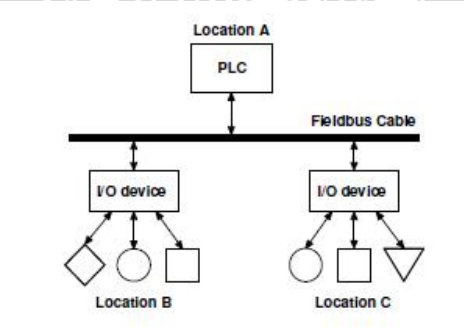
Gambar 2.5 menunjukkan jaringan *fieldbus* lama. Jaringan *fieldbus* membuat standar komunikasi modern seperti TCP/IP, IPX, dan Ethernet. Jaringan-jaringan ini saling berhubungan secara luas dimana operator dapat berada di temppat yang jauh dari proses yang sedang dikontrol olehnya. Hal ini membuka kemungkinan baru secara luas seperti pembangunan terkendali dan meningkatkan mutu sistem, pemantauan proses-poses pabrik dan bantuan di tempat operator dalam jaringan proses yang dapat dicapai.

Gambar 2.6 menunjukkan jaringan *fieldbus* modern dapat menjadi cepat dan mengatasi masalah kemacetan dalam kaitan dengan padatnya *traffic* pada jaringan atau kabel yang sama. Teknologi yang lebih lama mengandalkan beberapa ratus kabel, satu dari sekian sinyal, menunjang PLC dan *Operator station/Engineering Station*, sementara itu jaringan *fieldbus* modern menghubungkan objek proses dengan beberapa yang lain di central access point. Objek biasanya terhubung dalam sebuah mode serial maka tujuan tercapai jika banyak traffic dan informasi saling bertukar, hal ini dibutuhkan antara objek proses dan kontrol proses itu sendiri. Meskipun hal tersebut dapat terbantu dengan teknologi baru seperti kabel fiber yang mengirim informasi menggunakan cahaya sebagai ganti sinyal elektris atau menginstall sebuah kabel kedua maka objek terhubung secara paralel.



Gambar 2.9 Jaringan FIELDBUS lama

(Sumber : The Siemens SIMIT)



Gambar 2.10 Jaringan FIELDBUS Modern

(Sumber : The Siemens SIMIT)

Secara ekstensif properti positif dari teknologi *fieldbus* adalah sebagai berikut,

- Desain dan *Engineering*

  Membangun jaringan-jaringan yang terdefinisi dengan baik dan mengetahui standar dan konsep antara desain dan waktu yang direncanakan, maupun membantu meminimalkan kebutuhan dokumentasi

- Instalasi dan Pengawasan

  Sejak I/O *fieldbus* dapat lebih bermacam-macam dengan kabel yang terbatas dibutuhkan dalam proses, antara harga dan waktu diturunkan selama pengawasan dan instalasi jaringan dan melingkupi objek proses, PLC dan peralatan pemantauan yang lebih tinggi.

- Operasi dan Perawatan yang lebih mudah

  Data dikirim dalam sebuah jaringan dapat lebih bermacam-macam daripada jaringan-jaringan proses analog tradisional sejak standar-standar layanan pengiriman paket terbentuk sama halnya dengan TCP/IP. Ini berarti, tidak hanya informasi sinyal digital dan analog dapat terkirim, melainkan informasi diagnosa ekstensif dari beberapa objek proses yang berlokasi di sistem proses juga dapat terkirim. Informasi dapat secara mudah terkirim ke beberapa titik akhir dalam jaringan pabrik maupun jaringan yang lebih luas seperti internet. Ini membuat kooperatif dan pencarian *error* atau solusi masalah saat ini lebih mudah ditangani.

Beberapa fieldbus standar seperti PROFIBUS, INTERBUS, MPI, dan Industrial Ethernet. Umumnya semua digunakan untuk aplikasi industri dan menjamin dapat bekerja di bawah lingkungan yang sangat berbeda antara sebuah plant ataupun perkantoran. Perbedaannya pada bagaimana protokol *software* diimplementasikan, maupun properti fisik dari kabel dan level sinyal.

2.5 PROFIBUS

PROFIBUS menentukan seluruh media properti elektris sampai spesifikasi protokol. PROFIBUS adalah sebuah *two wire buss* sederhana,

membuat tegar di lingkungan yang tidak datar, menggunakan implementasi protokol minimal dan mudah, dengan keseluruhan yang minim. Hal tersebut mengurangi *down time* dan kebutuhan untuk perawatan. PROFIBUS adalah standar terbuka dan dibangun sebagai proyek penelitian, menyertakan beberapa badan hukum besar dan penelitan institut pada tahun 1987-1990. PROFIBUS adalah komunikasi fieldbus pertama dengan lebih dari 20% digunakan di dunia. Kira-kira 500.000 plant dan pabrik menggunakan PROFIBUS, dengan lebih dari 5 juta titik terinstall.

2.6 PROFIBUS-DP

PROFIBUS-DP adalah ekstensi dari PROFIBUS standar yang didesain untuk pertukaran data yang cepat pada level medan. PROFIBUS-DP bekerja seperti "*token ring*", token adalah melewati antara stasiun yang berbeda dan mengambil dari stasiun khusus jika informasi terkirim olehnya. Terdapat tiga perbedaan protokol DP. Pertama digunakan pada sistem SIMATIC adalah DP-V1 yang mensupport fitur seperti *fail-safe*.

Beberapa objek terhubung dengan jaringan PROFIBUS-DP memiliki identitas unik diantara 1-127 yang mana 0 dipesan untuk operasi broadcast (mengirimkan informasi kepada semua titik dalam sekali).

2.7 MPI

MPI atau *Multipoint Interface* adalah standar Siemens tertutup. MPI digunakan sebagai antarmuka program S7 PLC, akan tetapi juga dapat digunakan sebagai antarmuka untuk panel operator secara serempak. Setiap antarmuka MPI, atau titik, diidentifikasi sebagai alamat.

2.8 Siemens SIMATIC

Konsep Siemens SIMATIC adalah sebuah merk produk *hardware* dan *software* untuk aplikasi kontrol plant. Sistem SIMATIC itu sendiri bebas digunakan untuk mengirim informasi, tetapi hal itu harus mendukung standar MPI dan PROFIBUS maupun. Berbagai antarmuka komunikasi dapat digunakan pada waktu yang sama, menyatukan semua, tergantung pada kebutuhan konfigurasi dan sistem *Industrial Ethernet*.

Inti produk dari sistem SIMATIC adalah PCS 7 *software suite* (*Process Control System* 7) yang dibutuhkan untuk membuat program yang digunakan pada kontrol proses plant. Setiap sistem kontrol kemudian diinputkan ke dalam S7 PLC box. S7 PLC diatur kedalam mode run dan kemudian mulai untuk menukar informasi dengan objek sekitar hingga bermacam antarmuka tersedia.

2.9 SIMATIC *hardware*

SIMATIC *hardware suite* terdiri dari modul *hardware* yang digunakan untuk membuat sistem kontrol *plant* keseluruhan dari atas ke bawah. Intinya adalah S7 PLC box yang digunakan untuk mengontrol proses plant. Jaringan S7 PLC box dibangun menggunakan Ethernet, PROFIBUS, dan antarmuka MPI, tergantung dengan apa dan dimana untuk menghubungkan peralatan sekeliling seperti *Operator Stations* (OS) dan *Engineering Stations* (ES) driver DP dan PLC lain jika dibutuhkan. *Driver* DP mengatur objek proses dari plant dan komunikasi dengan PLC melalui jaringan.

2.10 *Engineering Stations* (ES) dan *Operator Stations* (OS)

Operator Stations (OS) adalah sebuah komputer yang mengontrol dan memonitor sebuah plant. Komputer yang ada biasanya lebih dari satu sehingga teknisi dapat melihat seluruh proses secara serempak. *Engineering Stations* (ES) adalah layanan personal dapat membuat perubahan untuk mengatur program dan menguji perubahan barunya. Biasanya ES hanya terdiri dari satu unit saja.

2.10.1 S7 PLC

S7 PLC berbeda dalam fungsi, kecepatan, dan kapasitas memori. Kebutuhan konfigurasi tergantung berdasarkan aplikasi. Proses-proses seperti sistem konveyor dengan beberapa pos dapat berjalan dengan S7 PLC yang lebih sederhana. Sistem yang lebih rumit seperti turbin dan mesin kertas membutuhkan seluruh kekuatan S7 PLC untuk bekerja sama sebagai

kelompok ataupun tersendiri, membagi kendali *driver* DP dan perhitungan dibutuhkan untuk menjalankan proses plant.

Konfigurasi *hardware* S7 PLC dasar terdiri dari,

- *Rack* (UR2) berguna untuk tempat komponen *hardware*,
- *Power Supply* (PS) berguna untuk menyediakan daya dan mengadakan persediaan baterai jika sistem terjadi kehilangan daya,
- *Central Processor Unit* (CPU) adalah unit utama untuk menjalankan program PLC yang ditempatkan pada kartu memori,
- *Communication Processor* (CP) adalah modul komunikasi prosesor yang digunakan untuk antarmuka dengan Ethernet.

PS, CPU, dan CP ditempatkan di UR2 Rak. UR2 rak adalah mekanik dan elektrik *backplane*. Rak membiarkan komunikasi komponen berbeda dengan komponen lainnya maupun menyediakan landasan mekanis stabil untuk digunakan pada peralatan industri yang keras. Bergantung pada tipe rak dapat menyusun PLC sel ganda, dimana setiap kelompok terdiri dari PS, CPU, dan CP.

PS menyuplai daya dari rak menuju backplane. PS hanya terdiri dari dua baterai (*rechargeable*), yang digunakan jika daya regular dari jaringan listrik hilang. Program S7 PLC itu sendiri ditempatkan pada kartu memori *flash*, yang memegang informasi selama dayanya hilang. Walaupun data konfigurasi yang memegang informasi dari macam-macam komponen terbingkai dalam rak, yang ditempatkan dalam *volatile* RAM jadi membuat redundansi PS penting melalui baterai.

Sumber lain dari redundansi dapat dibuat dengan menambah PS ekstra ke dalam rak. Dengan tujuan ini, dan bergantung pada kebutuhan keselamatan, ada perbedaan tipe dari PS seperti *redundant* dan *non-redundant*. Sebenarnya membuat solusi *non-redundant* adalah solusi dengan harga awal yang rendah, tetapi dapat membuktikan keputusan fatal setelahnya, dalam kaitan dengan kehilangan produksi disebabkan oleh *downtime* sistem PLC.

Modul CP seperti *network interface card* (NIC) yang ditemukan dalam modern PC, memiliki hubungan untuk jaringan ethernet. Sejak Industrial Ethernet (subset Ethernet) sama seperti Ethernet normal, pengkabelannya dapat dihubungkan ke *switch* atau hub seperti jaringan komputer lain. Teknisi dapat kemudian memrogram PLC lalu kartu antarmuka CP maupun antarmuka MPI disebutkan. Antarmuka Ethernet juga biasanya dihubungkan ke ES atau OS dimana operator dapat mengatur *plant* dengan S7 PLC. Modul CP yang memegang konektor fiber optic juga mengendalikan jaringan *fiber optic* jika S7 PLC membutuhkan data kritis dan objek proses selama waktu proses.

Jantung dari kontrol plant adalah modul CPU. Modul CPU terdiri dari CPU dan modul memori. Ukuran modul memori yang dibutuhkan tergantung dari ukuran program kontrol plant yang dibutuhkan untuk disimpan juga tergantung dari tipe modul CPU yang dapat diganti dengan modul yang lebih besar antara 1-16 MB. Modul CPU juga memegang antarmuka DP dan menggabungkan antarmuka DP/MPI. MPI dapat digunakan untuk program dalam kartu flash modul CPU melalui kartu ekspansi dalam PC. Pada umumnya, kartu memori diprogram melalui modul CP koneksi Ethernet atau melalui jaringan. Hal tersebut menyebabkan antarmuka DP dan DP-slave terhubung. Antarmuka DP memegang komunikasi objek proses.

2.10.2 *DP-slaves*

DP-slave adalah *hardware* objek proses yang memegang komunikasi antara S7 PLC dan sensor, aktuator, dan *drive* yang ditempatkan dalam *plant* atau *hierarchy* proses. Setiap *DP-slave* memiliki identitas node PROFIBUS yang unik., antara 1-127 seperti yang telah disebutkan sebelumnya. Identitas itu sendiri biasanya dapat diatur oleh *DIP-switches* secara langsung pada alat. Namanya berasal dari fakta bahwa objek menggunakan protokol PROFIBUS-DP untuk komunikasi antar satu atau lebih S7 PLC. Setiap PLC box juga mempertimbangkan untuk jadi sebuah

node dalam sistem dan oleh karena itu menunjuk sebuah nomor node. S7 PLC biasanya memberikan nilai 2 untuk node id (id 0 dipesan untuk paket-paket *broadcast*).

DP-slave terhubung melalui antarmuka PROFIBUS (kabel) ke S7 PLC. Setiap *driver* dapat menangani nomor variabel dari I/O dan tipe-tipe seperti 16 *input* sinyal analog atau 4 output sinyal digital. *Slaves* dapat terhubung dalam serial S7 PLC yang sama menggunakan kabel dan antarmuka yang sama seperti ditunjukkan oleh Gambar 2.7. Sama dengan S7 CPU ada *fail-safe, redundant*, dan versi *fiber-optic* dari modul DP lain.

Proses-proses industri seperti turbin melibatkan lebih dari 20 *DP-slave* dalam jaringan plant berisi 100 sinyal-sinyal, sementara itu proses-proses plant yang lebih rumit seperti mesin kertas melibatkan ribuan sinyal-sinyal dan jumlah slave yang lebih besar.

Gambar 2.11 Dua Siemens SIMATIC S7 PLC *Boxes* terkoneksi ke *DP-slaves*
(Sumber : The Siemens SIMIT)

2.11 SIMATIC *Software*

Software SIMATIC dikembangkan dan digunakan sejak kemunculan DOS (sekitar tahun 1980) sebelum Windows. Setiap revisi baru dari sistem, sistem tersebut *updated*, *upgraded*, dan dikerjakan ulang. *Software* yang diprogram adalah sebagai berikut,

- SIMATIC Manager, seluruh pusat kontrol untuk program SIMATIC.
- WinCC, pembuatan dan pemakaian OS dan ES.
- *Hardware Config*, Konfigurasi susunan *hardware*, digunakan dalam proses pabrik pada semua tingkat.

- *Continuous Function Chart* (CFC) Editor, blok fungsi ditarik dan ditaruh juga dihubungkan untuk membuat fungsi PLC.
- *Sequential Function Chart* (SFC) Editor, program digunakan untuk membentuk sekuensial menggunakan blok fungsi, sama halnya dengan sebuah *flowchart*.
- PG\PC *Interface*, mengatur antarmuka digunakan saat memrogram modul CPU.
- NetPro, program yang dibuat untuk komunikasi antara PLC dengan OS.

2.11.1 SIMATIC Manager

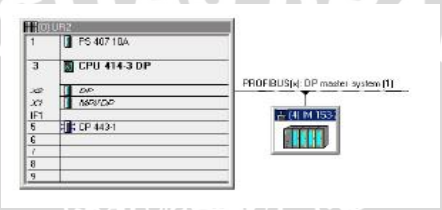
SIMATIC Manager adalah pusat kendali dalam SIMATIC yang digunakan untuk mengatur, membangun, dan membentuk proyek *software* dari titik pusat. Setiap satu dari program lain dalam SIMATIC suite dimulai dari manager itu sendiri. Pembangunan sebuah sistem kontrol biasanya membutuhkan sekelompok orang untuk bekerjasama menggunakan SIMATIC manager membuat konsep multi proyek yang membolehkan teknisi untuk membangun secara serempak model dan sistem kontrol, yang diketahui sebagai teknik yang bersamaan.

Manager menunjukkan proyek yang sedang dibuka, ringkasan konfigurasi *hardware*, dan komponen-komponen yang berbeda di dalam program pengaturan. Manager juga menunjukkan titik-titik DP yang terhubung (*master* dan *slave*) untuk kemudahan peninjauan luas peralatan yang terhubung, membuat diagnosa dan deteksi *error* lebih dapat diatur. Aplikasi *plant* termasuk lebih dari satu pemrograman proses menggunakan PLC, lebih dari satu program PLC, pada suatu waktu dapat dimasukkan ke dalam manager.

2.11.2 Konfigurasi *Hardware*

Hardware configuration editor didapatkan dalam SIMATIC manager. Pada *hardware configuration editor* teknisi dapat membuat seluruh jaringan *hardware*, mengenai seluruh macam PLC yang digunakan, tipe sambungan media antara titik-titik, dan dimana OS dan ES seharusnya

dihubungkan dalam jaringan sistem kontrol otomatis. Dari Gambar 2.8 pengaturan *hardware* dapat dilihat sebagai definisi oleh *hardware configuration editor*. *Window* sebelah kiri pada Gambar 2.8 menunjukkan komponen-komponen yang digunakan teknisi yang terbingkai secara fisik dalam rak UR2. Ada macam-macam perbedaan versi dari S7 CPU, CP, dan PS yang harus dikonfigurasi. Setiap nomor di dalam window menunjukkan slot dalam rak. Seperti dalam Gambar 2.8, CPU dan PS terbingkai dalam 2 slot yang berbeda.



Gambar 2.12 Konfigurasi *hardware*

(Sumber : The Siemens SIMIT)

Sebelah kanan, terdapat kabel PROFIBUS, yang mana terdapat DP-slave. Dalam kasus ini IM153-2 slave, yang memiliki 16 sinyal analog out (tidak terlihat dalam Gambar 2.8). Contoh tersebut mengilustrasikan sistem minimum dengan sinyal yang sangat sedikit tapi meskipun begitu Gambar 2.8 tersebut contoh yang baik jaringan sebuah plant yang dapat dibangun menggunakan antarmuka konfigurasi *hardware*.

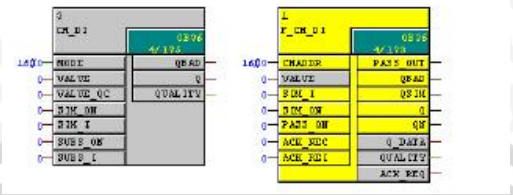
2.11.3 SFC Editor

Dalam *sequential function chart editor* teknisi membuat sekuensial plant berguna untuk menjalankan program. Dengan kata lain, urutan CFC *chart* harus dieksekusi dalam urutan untuk memulai plant dengan dikontrol. *Editor* juga mencakup langkah-langkah transisi yang dipenuhi sebelum sekuensial berjalan.

2.11.4 CFC Editor

Continuous function chart editor adalah tempat dimana teknisi membuat program kontrol sebenarnya. Komponen-komponen seperti PID-regulator, OR-block, dan fungsi waktu dapat ditarik dan ditaruh ke dalam

editor. Komponen-komponen tersebut kemudian dikonfigurasi dan dihubungkan dengan port I/O masing-masing.



Gambar 2.13 Blok CFC

(Sumber : The Siemens SIMIT)

Terdapat dua macam blok *low-level* dasar dalam editor yaitu blok *fail-safe* dan blok normal. Blok *fail-safe* memiliki warna kuning menyala sementara itu blok normal berwarna abu-abu seperti terlihat pada gambar 2.9. blok-blok tersebut memiliki fungsi untuk menjadi blok out dan input keluar dan masuk ke PLC itu sendiri. Blok-blok tersebut jadi batasan-batasan program PLC.

2.11.5 NetPro

Program ini digunakan untuk mengatur komunikasi antara PLC dengan *Operator Station* (OS). Tambahan untuk konfigurasi *hardware* dimana komunikasi sudah diatur pada awalnya, program tersebut mengubah blok-blok yang dibutuhkan untuk memastikan bahwa komunikasi antara PLC dan OS bekerja secara benar mengenai pesan dan alarm terkirim antara proses, PLC, dan OS.

2.11.6 Fail-safe dan non fail-safe

Blok *fail-safe* berbeda dari *non fail-safe* dalam beberapa aspek penting. Pertama, semua blok *fail-safe* membutuhkan CPU khusus yang dikenali sebagai modul H-series. Jika hal ini bukan kasus program kendali maka tidak dapat dimasukkan, dari semua, ke dalam modul memori CPU. Secara tipikal, modul *fail-safe-able* lebih mahal dari *non fail-safe*, ini alasan mengapa teknisi seharusnya mempertimbangkan pro dan kontra menggunakan *fail-safe* dibanding dengan yang lebih murah.

Fail-safe sebagian besar digunakan pada proses-proses plant dengan komponen sensitif yang dapat membahayakan jika sinyal hilang. Sebagai contoh sistem kimia seperti gas turbin. *Fail-safe* juga digunakan pada peralatan yang gampang meledak seperti landasan gas dan minyak.

2.11.7 WinCC

WinCC atau *Windows Control Center* adalah program gambar operator dibuat. Hubungan sinyal-sinyal dari *CFC-editor* terhubung ke komponen GUI dalam WinCC seperti tombol, diagram, gambar, dan *slide*. WinCC dapat dijalankan pada PC ganda yang dikenali sebagai *Operator Stations* (OS) yang membuat proses plant lebih mudah untuk diamati dan dikontrol.

2.12 Sensor Aliran

Sensor aliran air pada Gambar 2.10 ini terdiri terbuat dari plastik yang didalamnya terdapat rotor dan sensor *hall effect*. Saat air mengalir melewati rotor, rotor akan berputar. Kecepatan putaran ini akan tergantung dengan kecepatan aliran air. *Hall Effect* sensor akan mengeluarkan output pulsa sesuai dengan besarnya aliran air. Spesifikasi yang dimiliki oleh sensor aliran ini adalah sebagai berikut,

1. *Working voltage* 5V-24V
2. *Maximum current* 15 mA（DC 5V）
3. *Weight* 43 g
4. *External diameters* 20mm
5. *Flow rate range* 1～30 L/min
6. *Operating temperature* 0□ ～80□
7. *Liquid temperature* <120□
8. *Operating humidity* 35%～90%RH O
9. *perating pressure under* 1.2Mpa
10. *Store temperature* -25□ ～+80□

Adapun karakteristik sinyal yang dimiliki oleh sensor aliran ini adalah sebagai berikut,

1. *Output pulse high level Signal voltage* >4.5 V(*input* DC 5 V)
2. *Output pulse low level Signal voltage* <0.5V(*input* DC 5V)
3. *Precision* 3% (*Flow rate from* 1L/min to 10L/min)
4. *Output signal duty cycle* 40%～60%

Untuk mendapatkan nilai aliran dalam L/menit bisa didapat dengan rumusan berikut,

Q= (jumlah pulsa per menit) / 7.5 (2.22)

Jika diambil sampling waktu 1 detik setiap pembacaan pulsa, maka hasil pembacaan pulsa dikalikan 60 dan seterusnya.

Gambar 2.14 Sensor Aliran

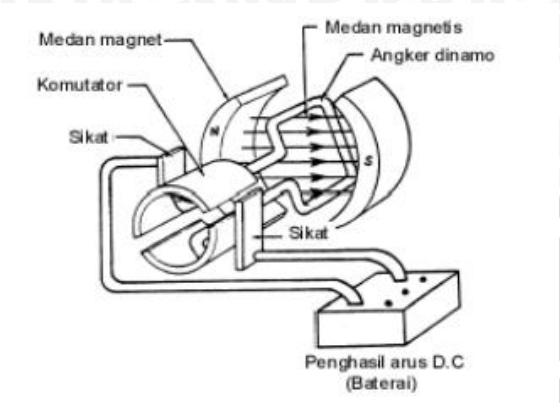
(Sumber : Indoware, 2012)

2.13 Motor DC

Motor DC pada Gambar 2.11 memerlukan suplai tegangan yang searah pada kumparan medan untuk diubah menjadi energi mekanik. Kumparan medan pada motor dc disebut stator (bagian yang tidak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar). Jika terjadi putaran pada kumparan jangkar dalam pada medan magnet, maka akan timbul tegangan (GGL) yang berubah-ubah arah pada setiap setengah putaran, sehingga merupakan tegangan bolak-balik. Prinsip kerja dari arus searah adalah membalik *phasa*

tegangan dari gelombang yang mempunyai nilai positif dengan menggunakan komutator, dengan demikian arus yang berbalik arah dengan kumparan jangkar yang berputar dalam medan magnet. Bentuk motor paling

sederhana memiliki kumparan satu lilitan yang bisa berputar bebas di antara kutub-kutub magnet permanen.



Gambar 2.15 Motor DC Sederhana

(Sumber : Sumanto, 1994)

2.14 Termokopel

Termokopel adalah sensor suhu yang banyak digunakan untuk mengubah perbedaan suhu dalam benda menjadi perubahan tegangan listrik (*voltage*). Termokopel yang sederhana dapat dipasang, dan memiliki jenis konektor standar yang sama, serta dapat mengukur temperatur dalam jangkauan suhu yang cukup besar dengan batas kesalahan pengukuran kurang dari 1 °C.

2.14.1 Prinsip Operasi

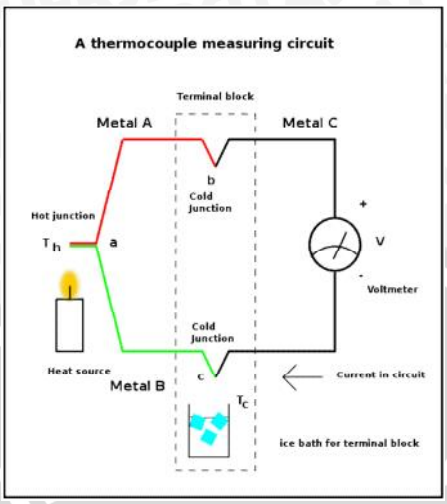
Pada banyak aplikasi, salah satu sambungan (sambungan yang dingin) dijaga sebagai temperatur referensi, sedang yang lain dihubungkan pada objek pengukuran. contoh, pada gambar di atas, hubungan dingin akan ditempatkan pada tembaga pada papan sirkuit. Sensor suhu yang lain akan mengukur suhu pada titik ini, sehingga suhu pada ujung benda yang diperiksa dapat dihitung. Termokopel dapat dihubungkan secara seri satu sama lain untuk membuat *termopile*, dimana tiap sambungan yang panas diarahkan ke suhu yang lebih tinggi dan semua sambungan dingin ke suhu yang lebih rendah ditunjukkan oleh Gambar 2.12. Dengan begitu, tegangan

pada setiap termokopel menjadi naik, yang memungkinkan untuk digunakan pada tegangan yang lebih tinggi.

Dengan adanya suhu tetapan pada sambungan dingin, yang berguna untuk pengukuran di laboratorium, secara sederhana termokopel tidak mudah dipakai untuk kebanyakan indikasi sambungan lansung dan instrumen kontrol. Mereka menambahkan sambungan dingin tiruan ke sirkuit mereka yaitu peralatan lain yang sensitif terhadap suhu (seperti termistor atau diode) untuk mengukur suhu sambungan input pada peralatan, dengan tujuan khusus untuk mengurangi gradiasi suhu di antara ujung-ujungnya.

Tegangan yang berasal dari hubungan dingin yang diketahui dapat disimulasikan, dan koreksi yang baik dapat diaplikasikan. Hal ini dikenal dengan kompensasi hubungan dingin. Biasanya termokopel dihubungkan dengan alat indikasi oleh kawat yang disebut kabel ekstensi atau kompensasi.

Kabel-kabel ekstensi biasanya memiliki spesifikasi untuk rentang suhu yang lebih besar dari kabel termokopel. Kabel ini direkomendasikan untuk keakuratan tinggi. Kabel kompensasi pada sisi lain, kurang presisi, tetapi murah biasanya campuran material konduktor yang murah yang memiliki koefisien termoelektrik yang sama dengan termokopel (bekerja pada rentang suhu terbatas), dengan hasil yang tidak seakurat kabel ekstensi. Kombinasi ini menghasilkan output yang mirip dengan termokopel, tetapi operasi rentang suhu pada kabel kompensasi dibatasi untuk menjaga agar kesalahan yang diperoleh kecil. Kabel ekstensi atau kompensasi harus dipilih sesuai kebutuhan termokopel. Pemilihan ini menghasilkan tegangan yang proporsional terhadap beda suhu antara sambungan panas dan dingin, dan kutub harus dihubungkan dengan benar sehingga tegangan tambahan ditambahkan pada tegangan termokopel, menggantikan perbedaan suhu antara sambungan panas dan dingin.



Gambar 2.16 Rangkaian Pengukuran Termokopel
(Sumber: Wikipedia, 2013)

2.15 Valve

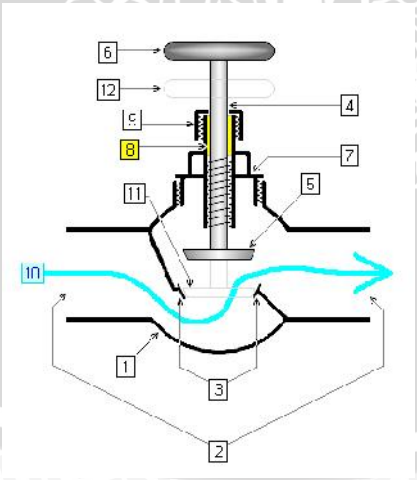
Valve atau katup adalah perangkat yang mengatur, mengarahkan atau mengontrol aliran cairan (gas, cairan, padatan fluidized, atau bubur) dengan membuka, menutup, atau sebagian menghalangi berbagai lorong-lorong. Katup secara teknis katup fitting , tapi biasanya dibahas sebagai kategori yang terpisah. Dalam sebuah katup terbuka, aliran fluida dalam arah dari tekanan tinggi ke tekanan rendah. Dalam skripsi ini digunakan katup 2 arah.

2.15. 1 Prinsip Operasi

Operasi posisi untuk katup 2 arah dapat menutup baik (tertutup) sehingga tidak ada aliran sama sekali melewati, sepenuhnya terbuka untuk aliran maksimum, atau kadang-kadang sebagian terbuka untuk setiap tingkat di antara keduanya. Banyak katup tidak dirancang untuk secara tepat mengontrol tingkat menengah aliran, katup tersebut dianggap baik membuka atau menutup. Beberapa katup yang dirancang khusus untuk mengatur jumlah yang bervariasi aliran. Katup tersebut telah disebut dengan berbagai nama, seperti mengatur, throttling, metering, atau katup jarum . Misalnya, katup jarum telah memanjang conically-meruncing cakram dan kursi yang cocok untuk kontrol aliran baik. Untuk beberapa katup, mungkin ada

mekanisme untuk menunjukkan dengan berapa banyak katup terbuka, tetapi dalam banyak kasus indikasi lain dari laju aliran yang digunakan, seperti terpisah *flow meter* .

Dalam *plant* dengan proses operasi *remote control*, seperti kilang minyak dan pabrik petrokimia, beberapa katup 2 arah dapat ditunjuk sebagai normal tertutup (NC) atau normal terbuka (NO) selama operasi rutin. Contoh katup normal-tertutup adalah pengambilan sampel katup, yang hanya dibuka ketika sampel diambil. Contoh lain dari katup normal-tertutup yaitu katup shut-down, yang tetap terbuka ketika sistem dalam operasi dan secara otomatis akan menutup dengan mengambil catu daya. Hal ini terjadi ketika ada masalah dengan unit atau bagian dari suatu sistem fluida seperti kebocoran untuk mengisolasi masalah dari seluruh sistem. Meskipun banyak katup 2 arah yang dibuat di mana aliran dapat masuk di kedua arah antara dua *port*, ketika katup ditempatkan ke dalam aplikasi tertentu, aliran sering diharapkan untuk pergi dari satu *port* tertentu pada sisi hulu dari katup, ke *port* lainnya di sisi hilir. Regulator tekanan adalah variasi dari katup di mana aliran dikontrol untuk menghasilkan hilir dengan tekanan tertentu jika memungkinkan. (Wikipedia, 2013)



Gambar 2.17 Katup 2 arah

(Sumber : Wikipedia, 2013)