BAB II

LANDASAN TEORI

A. Tinjauan Pustaka

Sistem kontrol cerdas dibutuhkan untuk dapat mempermudah manusia untuk melakukan suatu pekerjaan tertentu secara otomatis. Itulah tujuan mendasar dari mempelajari bidang studi Sistem Kendali, yaitu untuk dapat menciptakan sebuah sistem yang berjalan otomatis untuk mengeksekusi suatu tugas tertentu. Sehingga *output* sistem tersebut bisa dikendalikan kuantitas, kualitas dan konsistensinya.

Arduino Uno adalah salah satu alternatif *microcontroller* yang dapat digunakan untuk membangun suatu sistem kontrol yang dapat bekerja secara otomatis. Penelitian mengenai aplikasi Arduino Uno pada sistem kontrol telah banyak dilakukan.

AJ. Taufiq (2017) dari jurnalnya yang dipublikasikan pada Seminar Nasional SNTT V telah melakukan penelitian mengenai "Kontrol PID Pengaturan Temperatur Inkubator Sebagai Sarana Belajar Kontroler PID Digital" yang menjelaskan tentang implementasi kontrol PID dengan *plant* berupa pengaturan temperature inkubator dengan menterjemahkan rumus matematis kontrol PID menggunakan Bahasa pemrograman C Arduino. Tujuan penelitian ini adalah sebagai sarana pembelajaran mahasiswa tingkat awal Teknik Elektro untuk mempelajari kontrol PID melalui praktek implementasi kontrol PID pada *plant.*

Dinesh Bista (2016) dari jurnalnya yang diterbitkan *Virginia Commonwealth University* telah melakukan penelitian mengenai "*Understanding and Design of an Arduino-based PID Controller*" yang menjelaskan tentang penggunaan algoritma PID menggunakan platform Arduino. Algoritma PID berbasis Arduino ini digunakan pada sistem kontrol yang ada pada laboratorium dengan subjek penelitian adalah sistem pengendalian suhu pada *thermoelectric cooler.* Sistem kontrol tersebut deprogram untuk menjaga suhu sesuai dengan *set point* yang diinginkan.

Thirupathi Allam (2016) dari jurnalnya yang diterbitkan pada *International Research Journal of Engineering and Technology (IRJET)* telah melakukan penelitian mengenai "*Design of PID Controller for DC Motor Speed Control using Arduino Microcontroller*" yang menjelaskan tentang dengan PID *controller* untuk mengendalikan kecepatan putar motor tegangan DC menggunakan Arduino *microcontroller.*

Adhi Ksatria Theopaga (2014) dari jurnalnya yang diterbitkan pada *Journal of Theoretical and Applied Information Technology (JATIT)* telah melakukan penelitian mengenai "*Design and Implementation of PID Control Based Baby Incubator*" yang menjelaskan tentang penggunaan Arduino Uno untuk membuat sistem kontrol suhu untuk mengendalikan suhu inkubator bayi. Dalam artikel ini juga menjelaskan mengenai *tunning controller* berbasis Arduino menggunakan metode *Ziegler-Nichols.*

Ardiansyah (2016) telah melakukan penelitian tentang "Sistem Monitoring Air Layak Konsumsi Berbasis Arduino (Studi Kasus PDAM Patalassang)"

menjelaskan tentang penggunaan Arduino untuk mengetahui kualitas air PDAM berdasarkan nilai pH yang terukur. Dengan ketentuan apabila nilai pH ≥ 6,5 dan ≤ 8,5 akan dinyatakan air memiliki kualitas normal. Sedangkan apabila nilai pH ≤ 6,5 dan ≥ 8,5 akan dinyatakan air memiliki kualitas tidak normal.

M Syukur Budiawan H (2017) telah melakukan penelitian tentang "Sistem Pengendali Beban Arus Listri Berbasis Arduino" menjelaskan tentang bagaimana membangun sistem pengendali beban arus listrik untuk mengendalikan pemakaian bebas listrik pada rumah tangga menggunakan sensor arus ACS712. Pemakaian melebihi *setting* akan memutus arus menggunakan *relay*.

Muchammad Nur Fatah Muiz (2019) telah melakukan penelitian tentang "Rancang Bangun Pengendalian *Level* Air Otomatis pada Tangki dengan *Servo Valve* Berbasis PID *Controller*" menjelaskan tentang aplikasi PID pada pengendalian *level* dengan menggunakan simulasi pada aplikasi LabView 2014 dan membandingkannya dengan sistem tanpa *controller*. Penelitian tersebut menunjukkan bahwa respon pengendalian *level* air dengan *Servo Valve* menggunakan PID mampu mencapai *set point* yang diinginkan dengan τr = 37,04 detik dan *error steady state* 8,3% (dengan fungsi alih 12 cm) dan τr = 38,427 detik dan *error steady state* 2,715%(dengan fungsi alih 4 cm). Respon ini menunjukkan perbaikan dari respon sistem tanpa kontroler dengan τr = 73,998 detik dan *error steady state* 12,507%.

Penelitian-penelitian sebelumnya menunjukkan aplikasi PID *controller* berbasis Arduino dapat digunakan untuk mengendalikan suatu proses secara otomatis. Sedangkan pada skripsi ini, penulis akan menjelaskan aplikasi PID

controller berbasis Arduino untuk mengendalikan *level* suatu bejana secara otomatis namun menggunakan *hardware* berupa peralatan instrumentasi berskala industri untuk meneliti kemungkinan penggunaan *microcontroller* sebagai alternatif subtitusi dari *industrial base controller.*

B. Landasan Teori

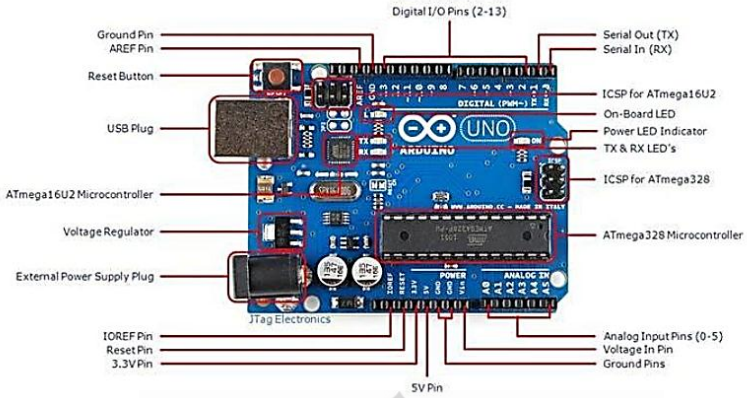
1. *Controller*

Controller atau pengendali adalah sebuah alat dimana sebuah instruksi ditempatkan. Pengendali mengendalikan *Input/Output* sesuai dengan instruksi tersebut sehingga terbentuk sebuah sistem yang bekerja secara otomatis, dan konsisten. Berikut adalah penjelasan beberapa jenis Pengendali yang akan banyak dibahas dalam skripsi ini.

a. Arduino Uno

Arduino Uno adalah papan sirkuit *microcontroller* dengan basis Atmega328P. Arduino Uno memiliki 14 Digital Pin (*input/output*) dengan 6 diantaranya dapat digunakan sebagai *PWM output*, memiliki 6 *Analog Input*, 16MHz kristal kwarsa, koneksi *USB*, *power jack*, *ICPS header* dan tombol *reset.* Alat ini memiliki semua yang diperlukan untuk menyokong kerja *microcontroller*nya. Hanya dengan menghubungkannya dengan komputer melalui koneksi *USB* atau hanya dengan menghubungkannya dengan adaptor *AC-to-DC* atau baterai, alat ini siap digunakan. Siapapun dapat mendesain apapun dengan Arduino Uno tanpa perlu takut salah, untuk skenario terburuk dapat diatasi dengan

mengganti *microcontroller*nya hanya dengan harga puluhan ribu rupiah (Arduino.cc. 2019).



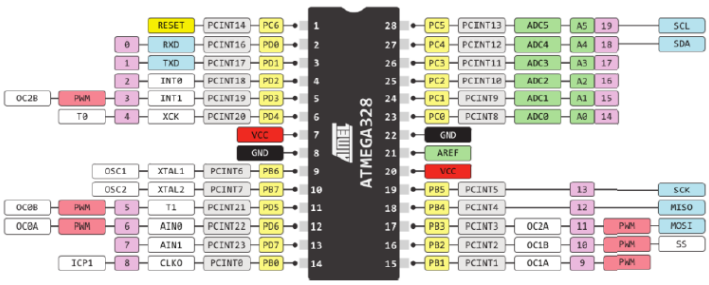
Gambar 2.1 Arduino Uno berbasis ATMEL Atmega 328P

(Sumber : Project Sistem Kendali Elektronik Berbasis Arduino.

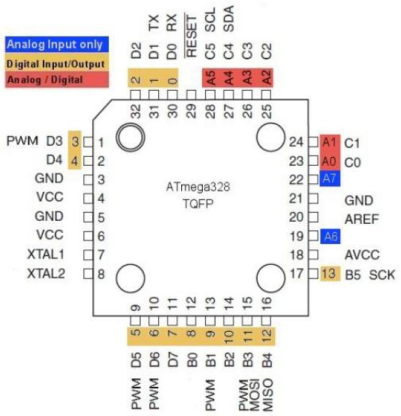
Dr. Junaidi, S.Si., M.Sc. & Yulian Dwi Prabowo. 2018)

Uno dalam bahasa Italia memiliki arti 'satu' dipilih sebagai nama untuk menandai rilisnya *Arduino Software (IDE)* generasi pertama (1.0). Sekaligus menjadi Arduino pertama yang memiliki koneksi *USB* dan menjadi model referensi untuk Arduino generasi selanjutnya hingga yang terbaru.

ATMega328 adalah mikrokontroller keluaran dari atmel yang mempunyai arsitektur RISC (*Reduce Instruction Set Computer*) dimana setiap proses eksekusi data lebih cepat dari pada arsitektur CISC (*Completed Instruction Set Computer*).



(a)



(b)

Gambar 2.2 (a) PinOut ATMega328 Model DIP dan (b) PinOut ATMega328 Model SMD

(Sumber : Project Sistem Kendali Elektronik Berbasis Arduino. Dr. Junaidi, S.Si., M.Sc. & Yulian Dwi Prabowo. 2018)

ATMega 328 memiliki beberapa fitur antara lain :

1) Memiliki 130 macam instruksi yang hampir semuanya dieksekusi dalam satu siklus clock.
2) Memiliki 32 x 8-bit register serba guna.
3) Kecepatan akses mencapai 16 MIPS dengan clock 16 MHz.
4) Memiliki 32 KB Flash memory dan pada Arduino memiliki bootloader yang menggunakan 2 KB dari flash memori sebagai bootloader.
5) Memiliki EEPROM (*Electrically Erasable Programmable Read Only Memory*) sebesar 1KB sebagai tempat penyimpanan data semi permanen karena EEPROM tetap dapat menyimpan data meskipun catu daya dimatikan.
6) Memiliki SRAM (*Static Random Access Memory*) sebesar 2KB.

7) Memiliki pin I/O digital sebanyak 14 pin, 6 pin diantaranya dapat digunakan sebagai pin PWM (*Pulse Width Modulation*).

8) Memiliki Master/Slave SPI Serial interface.

Mikrokontroler ATmega328 memiliki arsitektur Harvard, yaitu pemisah antara memori untuk kode program dan memori untuk data sehingga dapat memaksimalkan kerja dari mikrokontroler. Instruksi – instruksi dalam memori program dieksekusi dalam satu alur tunggal, dimana pada saat satu instruksi dikerjakan instruksi berikutnya sudah diambil dari memori program.

Konsep inilah yang memungkinkan instruksi – instruksi dapat dieksekusi dalam setiap satu siklus clock. Sebanyak 32 x 8-bit register serba guna digunakan untuk mendukung operasi pada ALU (*Arithmatic Logic Unit*) yang dapat dilakukan dalam satu siklus. Sebanyak 6 dari register serbaguna ini dapat digunakan sebagai 3 buah register pointer 16-bit pada mode pengalamatan tidak langsung untuk mengambil data pada ruang memori data. Ketiga register pointer 16-bit ini disebut dengan register X (gabungan R26 dan R27), register Y (gabungan R28 dan R29), dan register Z (gabungan R30 dan R31). Hampir semua instruksi AVR memiliki format 16-bit. Setiap alamat memori program terdiri dari instruksi 16-bit atau 32-bit. Selain register serba guna di atas, terdapat register lain yang terpetakan dengan teknik *memory mapped* I/O berukuran 64 byte.

Beberapa register ini digunakan untuk fungsi khusus antara lain sebagai register control Timer/Counter, Interupsi, ADC, USART, SPI, EEPROM, dan fungsi I/O lainnya. Register – register ini menempati memori pada alamat 0x20h – 0x5Fh. Arduino Uno R3 memiliki 14 pin digital I/O (dimana 6 pin dapat

digunakan sebagai *Output* PWM), 6 pin analog *input*, 2x3 pin ICSP (untuk memprogram Arduino dengan software lain), dan kabel USB. Untuk menghidupkannya cukup dengan menghubungkan kabel USB ke komputer atau menggunakan adaptor 5 VDC. Berikut adalah spesifikasi Arduino Uno :

Tabel 2.1 Spesifikasi Arduino Uno

Tegangan Operasi 5V	Flash Memory 32 KB (ATmega328P)
Tegangan Input (Rekomendasi) 7-12V	SRAM 2 KB (ATmega328P)
Tegangan Input (Limit) 6-20V	EEPROM 1 KB (ATmega328P)
Pin Digital I/O 14 (6 PinOutput PWM)	Clock Speed 16 MHz
Pin Digital PWM I/O 6	LED_BUILTIN 13
Pin Analog Input 6	Panjang 68,6 mm
Arus DC tiap Pin I/O 20 mA	Lebar 53,4 mm
Arus DC Pin 3,3V 50 mA	Berat 25 g

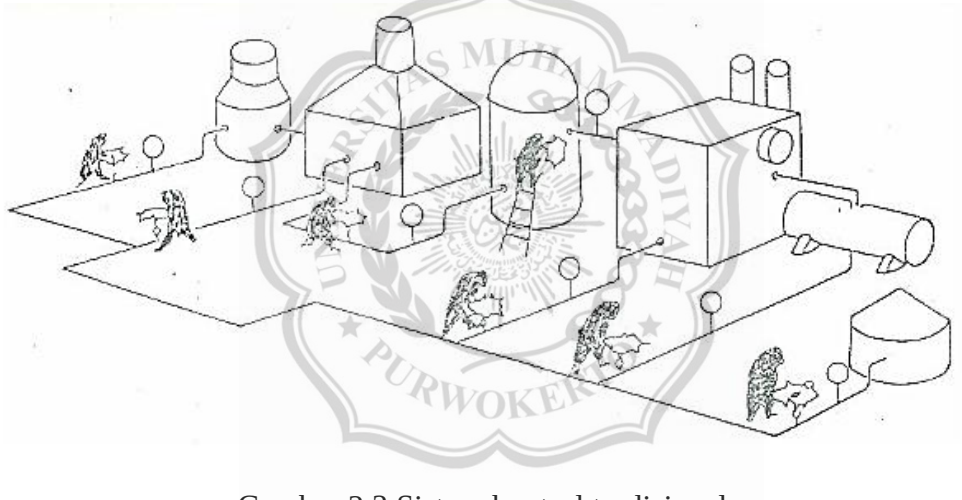
Dan beberapa keunggulan dari Arduino Uno adalah sebagai berikut :

1) Pengembangan *project* mikrokontroler akan menjadi lebih mudah dan menyenangkan. Pengguna dapat langsung menghubungakan board Arduino ke komputer atau laptop
2) melalui kabel USB. Board Arduino juga tidak membutuhkan *downloader* untuk *download* program yang telah dibuat dari computer ke mikrokontrolernya.
3) Didukung oleh Arduino IDE dangan bahasa pemrograman dengan library yang lengkap.
4) Terdapat modul yang siap pakai/shield sehingga dapat langsung dipasang pada board Arduino

Sumber : (Project Sistem Kendali Elektronik Berbasis Arduino. Dr. Junaidi, S.Si., M.Sc. & Yulian Dwi Prabowo. 2018).

a. *Distributed Control System* (DCS)

Pada mulanya, untuk mengendalikan sebuah proses/plant digunakan konsep Sistem Kendali Tradisional. Pada konsep ini, peralatan instrumentasi dan sistem kontrol didistribusikan di seluruh plant, dimana operator dapat membaca set point dan mengatur keluaran. Namun antara satu sistem kontrol dengan sistem kontrol yang lain tidak dihubungkan, sehingga operator bertugas mengkoordinasikan sistem kontrol yang terdistribusi tersebut. Komunikasi yang digunakan untuk mengintegrasikan plant dilakukan dengan komunikasi verbal antar operator.



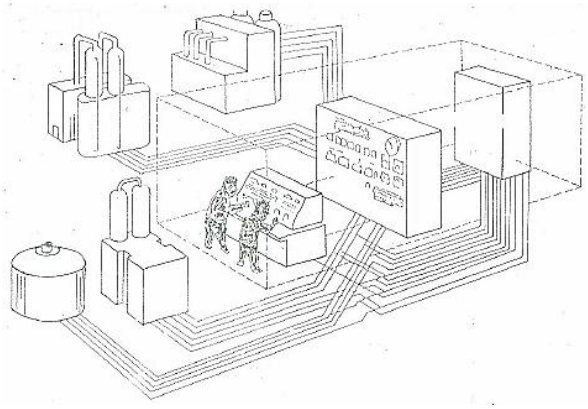
Gambar 2.3 Sistem kontrol tradisional

(Sumber : Dasar Instrumentasi dan Sistem Kontrol. BPST. 2007)

Namun seiring dengan meningkatnya ukuran dan tingkat kerumitan suatu proses, konsep ini semakin tidak mungkin untuk diterapkan. Dan dengan ditemukannya sistem instrumentasi berbasis *pneumatic* pada tahun 1960 pengendalian proses tidak lagi dilakukan secara tersebar melainkan terpusat dalam suatu ruang kendali (*control room*). Mekanismenya : pengukuran proses variabel dilakukan oleh sensor di lapangan, kemudian hasil pengukuran ditransmisikan oleh transmitter (dalam sinyal pneumatic 0,2-1,0 kg/cm^2 atau 3-15 psi) ke

controller yang berlokasi di ruang kendali. Selanjutnya sinyal control yang diinginkan ditransmisikan kembali menuju *actuator* pada unit proses. Semakin jauh jarak plant dengan *controller* semakin besar pula waktu tunda (*lag time*) menjadi kekurangan utama sistem pneumatic. Namun pada tahun 1970 seiring ditemukannya sistem kendali berbasis elektronik/digital waktu tunda ini berhasil ditangani dengan aplikasi sinyal elektronik 4-20 mA menggantikan sinyal *pneumatic*. Konsep kendali seperti ini disebut sebagai Sistem Kendali Langsung (*Direct Control System*).

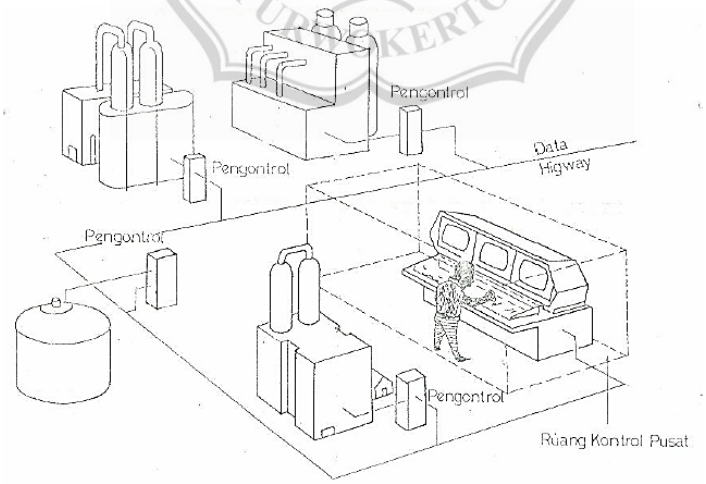
Gambar 2.4 Sistem kontrol *pneumatic* terpusat di *control room*



Gambar 2.5 Sistem kontrol *Direct Digital Control*

(Sumber : Dasar Instrumentasi dan Sistem Kontrol. BPST. 2007)

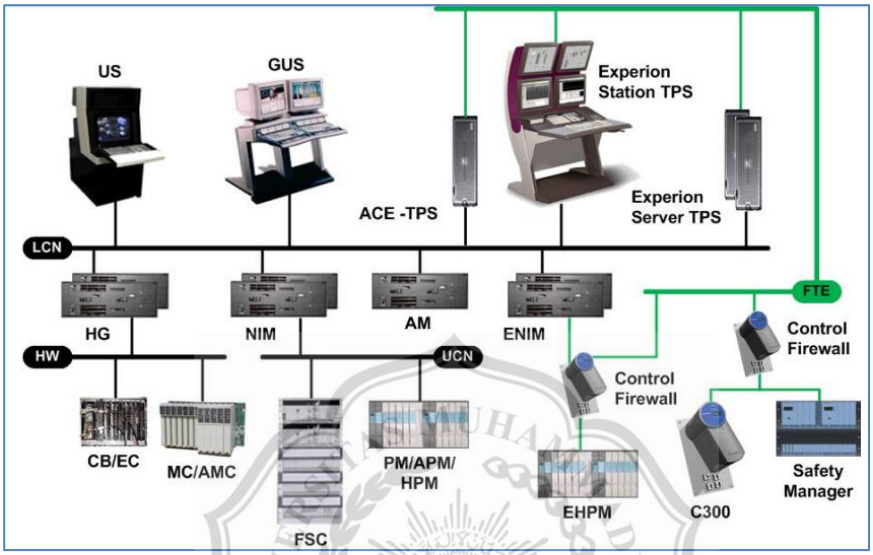
Arsitektur sistem kontrol proses berbasis *Distributed Control System* (DCS) mulai diperkenalkan dalam era industri proses sekitar tahun 1976. Dari perkembangan DCS pertama kali hingga tahun 1995, telah terjadi penambahan fungsi dan modifikasi sehingga penggunaannya menjadi lebih *user friendly* dan perawatan yang lebih mudah.



Gambar 2.6 Sistem kontrol *Distributed Control System*

(Sumber : Dasar Instrumentasi dan Sistem Kontrol. BPST. 2007)

DCS adalah suatu jaringan komputer yang menjalankan fungsi kendali untuk tujuan monitoring dan mengendalikan proses variabel pada sebuah proses industri.



Gambar 2.7 Arsitektur DCS Honeywell Experion PKS

(Sumber : www.honeywellprocess.com)

Secara umum arsitektur DCS terbagi ke dalam 5 (lima) bagian utama, yaitu :

1) *Human Machine Interface* (HMI)

Gambar 2.8 HMI DCS Honeywell Orion *Console*

HMI merupakan antar muka antara operator dengan proses. Seluruh data ditampilkan pada HMI merupakan data yang diambil langsung dari controller yang berisi seluruh parameter yang mewakili kondisi proses secara aktual. Operator dapat menetapkan set point dan mode kendali melalui HMI.

2) *Controller Unit*

Gambar 2.9 Honeywell C300 *Controller*

Sebuah komputer kontrol yang memiliki algoritma kendali yang *dedicated*. *Controller* mengeksekusi seluruh algoritma kendali dengan fungsi utama untuk membandingkan parameter proses dengan *set point* yang ditetapkan oleh operator untuk kemudian mengeluarkan sinyal kendali kepada *final control element.*

3) *Server*

Sebuah komputer yang berfungsi untuk melakukan *pulling data* dari controller untuk ditampilkan ke dalam HMI.

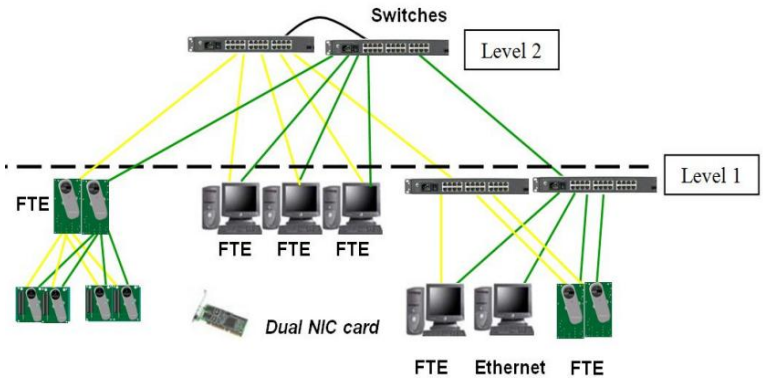
Gambar 2.10 Honeywell *Virtualization Server*

5) Input/Output

Input DCS mampu membaca masukkan berupa sinyal analog 4-20mA, field bus, HART, dan digital, sedangkan Output DCS dapat mengeluarkan sinyal analog 4-20mA dan digital sesuai dengan jenis I/O yang digunakan.

6) Network

Seluruh sistem distribusi tersebut terhubung dalam satu jaringan data, yang berbeda-beda untuk setiap merk DCS. DCS keluaran terbaru (s/d 2019) menggunakan FTE (*Fault Tolerant Ethernet*) sebagai basis Network.



Gambar 2.11 FTE *Network*

(Sumber : www.honeywellprocess.com)

2. *Input/Output*

Guna mempermudah pemahaman mengenai skripsi ini akan dijabarkan penjelasan mengenai I/O yang akan digunakan dalam skripsi ini. Batasan perlu diterapkan karena pembahasan teknologi I/O DCS dan Arduino sangat luas, sehingga skripsi ini hanya akan membahas garis besar *Input & Ouput* Arduino, *Input* DCS (*Transmitter*), *Output* DCS (*Control Valve & Positioner*).

a. *Input & Output* Arduino

Masing-masing dari 14 *pin digital* di Uno dapat digunakan sebagai *input* atau *output*, dengan menggunakan fungsi *pinMode ()*, *digitalWrite ()*, dan *digitalRead ()*, beroperasi dengan daya 5 volt. Setiap pin dapat memberikan atau menerima maksimum 40 mA dan memiliki internal pull-up resistor (secara default terputus) dari 20-50 kOhms. Selain itu, beberapa pin memiliki fungsi khusus :

1) *Serial*: 0 (RX) dan 1 (TX). Digunakan untuk menerima (RX) dan mengirimkan (TX) TTL data serial. Pin ini dihubungkan ke pin yang berkaitan dengan chip Serial ATmega8U2 USB-to-TTL.
2) *Eksternal menyela*: 2 dan 3. Pin ini dapat dikonfigurasi untuk memicu interrupt pada nilai yang rendah, dengan batasan tepi naik atau turun, atau perubahan nilai.
3) *PWM*: 3, 5, 6, 9, 10, dan 11. Menyediakan output PWM 8-bit dengan fungsi *analogWrite ()*.
4) *SPI*: 10 (SS), 11 (Mosi), 12 (MISO), 13 (SCK). Pin ini mendukung komunikasi SPI menggunakan *SPI library*.

5) *LED*: 13. Ada built-in LED terhubung ke pin digital 13. Ketika pin bernilai nilai HIGH, LED on, ketika pin bernilai LOW, LED off.

6) Uno memiliki 6 masukan analog, berlabel A0 sampai dengan A5, yang masing-masing menyediakan 10 bit dengan resolusi (yaitu 1024 nilai yang berbeda). Selain itu, beberapa pin memiliki fungsi khusus:

7) *I2C: A4 (SDA) dan A5 (SCL)*. Dukungan I2C (TWI) komunikasi menggunakan perpustakaan Wire.

8) *Aref*. Tegangan referensi (0 sampai 5V saja) untuk input analog. Digunakan dengan fungsi *analogReference ()*.

9) *Reset*. Bawa baris ini LOW untuk me-reset *microcontroller*.

ATmega328 mempunyai 32 KB (dengan 0,5 KB digunakan untuk bootloader). ATmega 328 juga mempunyai 2 KB SRAM dan 1 KB EEPROM (yang dapat dibaca dan ditulis (RW/read and written) dengan EEPROM library().Setiap 14 pin digital pada Arduino Uno dapat digunakan sebagai input dan output, menggunakan fungsi pinMode(),digitalWrite(), dan digitalRead(). Fungsi-fungsi tersebut beroperasi di tegangan 5 Volt. Setiap pin dapat memberikan atau menerima suatu arus maksimum 40 mA dan mempunyai sebuah resistor pull-up (terputus secara default) 20-50 kOhm. Selain itu, beberapa pin mempunyai fungsi-fungsi spesial:

1) **Serial: 0 (RX) dan 1 (TX)**. Digunakan untuk menerima (RX) dan memancarkan (TX) serial data TTL (Transistor-Transistor Logic). Kedua pin ini dihubungkan ke pin-pin yang sesuai dari chip Serial Atmega8U2 USB-ke-TTL.

2) **External Interrupts: 2 dan 3**. Pin-pin ini dapat dikonfigurasikan untuk dipicu sebuah interrupt (gangguan) pada sebuah nilai rendah, suatu kenaikan atau penurunan yang besar, atau suatu perubahan nilai. Lihat fungsi attachInterrupt() untuk lebih jelasnya.

3) **PWM: 3, 5, 6, 9, 10, dan 11**. Memberikan 8-bit PWM output dengan fungsi analogWrite().

4) **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)**. Pin-pin ini mensupport komunikasi SPI menggunakan SPI library.

5) **LED: 13**. Ada sebuah LED yang terpasang, terhubung ke pin digital 13. Ketika pin bernilai HIGH LED menyala, ketika pin bernilai LOW LED mati.

Arduino UNO mempunyai 6 input analog, diberi label A0 sampai A5, setiapnya memberikan 10 bit resolusi (contohnya 1024 nilai yang berbeda). Secara default, 6 input analog tersebut mengukur dari ground sampai tegangan 5 Volt, dengan itu mungkin untuk mengganti batas atas dari rangenya dengan menggunakan pin AREF dan fungsi analogReference(). Di sisi lain, beberapa pin mempunyai fungsi spesial: **TWI: pin A4 atau SDA dan pin A5 atau SCL**. Mensupport komunikasi TWI dengan menggunakan Wire library.

Ada sepasang pin lainnya pada board yaitu :

1) **AREF**. Referensi tegangan untuk input analog. Digunakan dengan analogReference().

2) **Reset**. Membawa saluran ini LOW untuk mereset mikrokontroler. Secara khusus, digunakan untuk menambahkan sebuah tombol reset untuk melindungi yang memblock sesuatu pada board.

Lihat juga pemetaan antara pin Arduino dengan port Atmega328. Pemetaan untuk Atmega8, 168, dan 328 adalah identik (lihat Gambar 2.1).

Sumber : (Project Sistem Kendali Elektronik Berbasis Arduino. Dr. Junaidi, S.Si., M.Sc. & Yulian Dwi Prabowo. 2018).

a. *Input* DCS

DCS memiliki beberapa jenis inputan dan dibagi berdasarkan jenis sinyal yang digunakan, yaitu :

1) Analog Input : 4-20mA signal (dari Analog atau STIM/Digital Transmitter)
2) Digital Input : Discrete Open/Close Contact (dari Switch atau Contact)
3) Low Level Analog Input : mV signal (dari Thermocouple)
4) Resistance Input : Ω signal (dari RTD)

Dalam skripsi ini digunakan sensing element berupa elektronik *level transmitter* (tipe *Differential Pressure*) yang merubah besaran proses (0 s/d 100% *Level* Bejana) menjadi sinyal standar yang sebanding dengan nilai pengukuran. Dengan konversi sebagai berikut :

Tabel 2.2 Konversi Sinyal Standar

Procentage	***Signal Standard***			
	Pneumatic		***Electronic***	
%	**kg/cm^2**	**psi**	**mA**	**V**
0	0,2	3	4	0

10	0,28	4,2	5,6	0,5
20	0,36	5,4	7,2	1
30	0,44	6,6	8,8	1,5
40	0,52	7,8	10,4	2
50	0,6	9	12	2,5
60	0,68	10,2	13,6	3
70	0,76	11,4	15,2	3,5
80	0,84	12,6	16,8	4
90	0,92	13,8	18,4	4,5
100	1	15	20	5

Gambar 2.12 *Differential Pressure Transmitter* Honeywell

(Sumber : www.honeywellprocess.com)

b. Output DCS

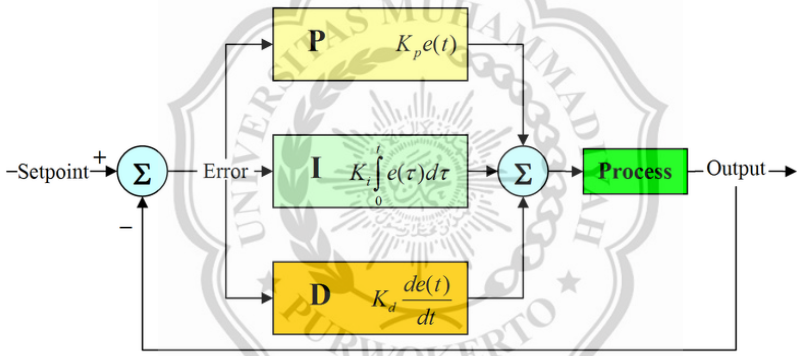
DCS memiliki beberapa jenis output dan dibagi berdasarkan jenis sinyal yang dihasilkan, yaitu :

1) *Analog Output* : 4-20mA *signal* (ke *Positioner Control Valve*)
2) *Digital Output* : *Dry* (*Open/Close* ke *Contact* atau *Coil Relay*), *Wet* (0 atau 24VDC ke *Coil Relay* atau I)

Dalam skripsi ini digunakan sinyal *Analog Output* (4-20mA) yang dihubungkan dengan *Control Valve* yang dilengkapi dengan *Positioner.*

Gambar 2.13 *Control Valve* dengan *Positioner*

3. Algoritma PID



Gambar 2.14 Blok diagram PID *Controller*

Kontroler PID (*Proportional, Integral, Derivative Controller*) merupakan mekanisme umpan balik (*feedback*) yang dipakai dalam pengendalian proses. Kontroler PID secara terus menerus menghitung nilai kesalahan (*error*) dari beda antara nilai yang dikehendaki (*set point*) dengan variabel proses terukur. Kontroler berusaha meminimalkan beda dengan cara mengendalikan posisi aktuator atau *final control element* lainnya dengan mengeluarkan sinyal kendali (*manipulated variabel/controlled variable*) sesuai dengan hasil penjumlahan :

$$u(t) = P(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

$with\ e = Y_{sp} - Y_m$ ..... (2.2)

Dengan keterangan :

K_p : *Gain Proportional* (parameter tuning)

K_i : *Gain Integral* (parameter tuning)

K_d : *Gain Derivative* (parameter tuning)

e : *Error*

Y_{sp} : *Set Point*

Y_m : *Process Variable*

t : Waktu

τ : Variabel Integrasi; nilainya diambil dari nol sampai t

Sedangkan transfer fungsi dalam bentuk domain *Laplace* kontroler PID :

$L(s) = K_p + \frac{K_i}{s} + K_d s$ .....(2.3)

K_p : *Gain Proportional* (parameter tuning)

K_i : *Gain Integral* (parameter tuning)

K_d : *Gain Derivative* (parameter tuning)

s : frekuensi bilangan kompleks

Dengan nilai K_p, K_i, dan K_d semuanya positif. Maka nilai P, I dan D melambangkan koefisien dengan ketentuan :

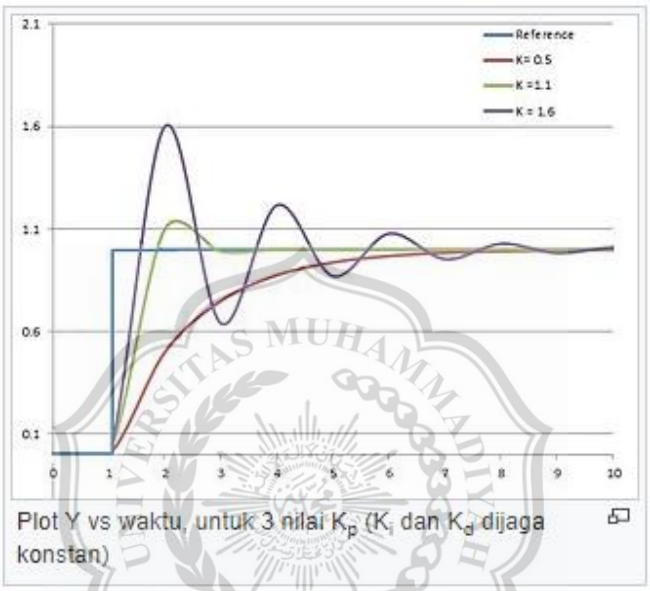
1) *P* : bertanggung jawab untuk nilai kesalahan saat ini. Contohnya, jika nilai kesalahan besar dan positif, maka keluaran kontrol juga besar dan positif.

   Term proporsional akan menghasilkan nilai keluaran yang berbanding lurus dengan nilai kesalahan. Responnya dapat diatur dengan mengalikan

kesalahan (*error*) dengan konstanta Kp, disebut konstanta *gain proporsional* atau *gain* kontroler. *Term proporsional* dirumuskan :

$P_{out} = K_p e(t)$ ……(2.4)

$P_{out} = K_p(Y_{sp} - Y_m)$ ……..(2.5)



Gambar 2.15 Respon kontrol terhadap nilai P pada I & D konstan

(Sumber : Wikipedia)

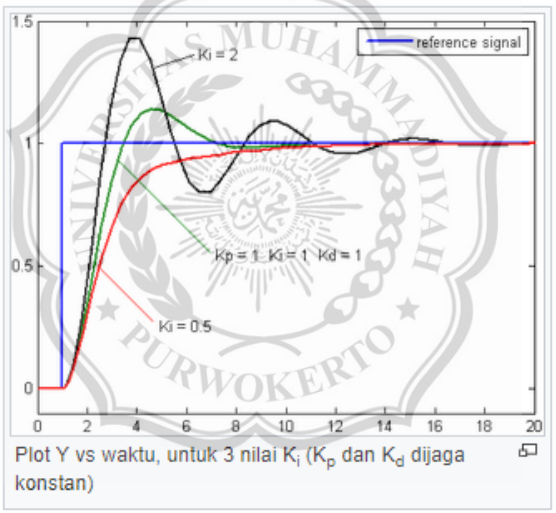
Gain yang besar menghasilkan perubahan yang besar pada keluaran untuk suatu nilai kesalahan tertentu. Jika gain terlalu besar, sistem akan membutuhkan waktu yang lama untuk mencapai kondisi *steady-state*. Sebaliknya, gain yang bernilai kecil maka respon keluaran juga kecil, sehingga kontroler menjadi kurang responsif/sensitif, hal ini akan mengakibatkan respon kontroler akan lebih lambat jika mendapatkan gangguan. Berikut adalah respon control terhadap perubahan nilai P. *I* : bertanggung jawab untuk nilai kesalahan sebelumnya. Contoh, jika keluaran

saat ini kurang besar, maka kesalahan akan terakumulasi terus menerus, dan kontroler akan merespon dengan keluaran lebih tinggi.

Peranan dari term integral berbanding lurus dengan besar dan lamanya *error*. *Integral* dalam kontroler PID adalah jumlahan *error* setiap waktu dan mengakumulasi *offset* yang sebelumnya telah dikoreksi. *Error* terakumulasi dikalikan dengan *gain integral* (K_i) dan menjadi keluaran kontroler.

Term integral dirumuskan dengan :

$$I_{out} = K_i \int_0^t e(\tau)d\tau \quad \text{.....(2.6)}$$



Gambar 2.16 Respon kontrol terhadap nilai I pada P & D konstan

(Sumber : Wikipedia)

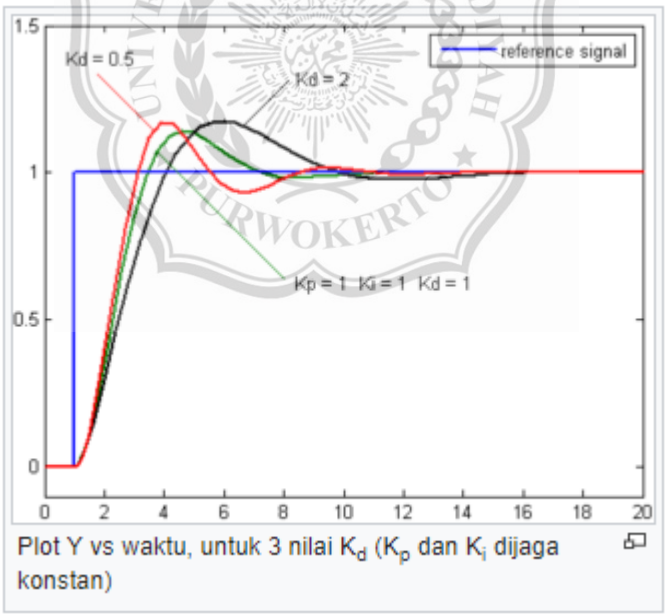
Term integral mempercepat perpindahan proses menuju *Set Point* dan menghilangkan *steady-state error* yang muncul pada kontroler *proportional*. Namun, karena *integral* merespon terhadap *error* terakumulasi dari sebelumnya, maka dapat menyebabkan *overshoot*. Berikut adalah respon kontrol terhadap perubahan nilai I.

2) *D* : bertanggung jawab untuk kemungkinan nilai kesalahan mendatang, berdasarkan pada *rate* perubahan tiap waktu.

Turunan *error* pada proses dihitung dengan menentukan kemiringan *error* setiap waktu dan mengalikan perubahan tiap waktu dengan *gain derivative* (K_d). Term derivatif dirumuskan dengan :

$$D_{out} = K_d \frac{de(t)}{dt} \quad \text{..... (2.7)}$$

Aksi derivatif memprediksi perilaku sistem dan kemudian memperbaiki waktu tinggal dan stabilitas sistem. Aksi derivatif jarang digunakan pada industri - diperkirakan hanya 25% kontroler - karena akibatnya pada stabilitas sistem pada aplikasi dunia nyata.



Gambar 2.17 Respon kontrol terhadap nilai D pada P & I konstan

(Sumber : Wikipedia)

Kontroler PID bekerja hanya dengan hasil pengukuran langsung variabel proses tidak terhadap sifat/karakteristik prosesnya secara langsung, sehingga untuk mendapatkan respon PID yang sesuai dengan proses dibutuhkan proses *adjustment* parameter PID (*tuning*). Respon kontroler dapat dijelaskan dengan bagaimana responnya terhadap *error*, besarnya *overshoot* terhadap *set point* dan derajat osilasi sistem.

Ada beberapa cara untuk menentukan nilai K_p, K_i, K_d. Salah satunya adalah dengan cara tunning nilainya satu persatu. dimulai dengan nilai K_p (*Gain proporsional*) terlebih dahulu, hal ini dikarenakan kita perlu mencari respon sistem yang paling cepat dengan cara meminimalkan nilai *rise time*, jangan memberikan nilai K_p terlalu besar atau terlalu kecil. Setelah respon dirasa cukup tepat hal selanjutnya yang dapat dilakukan adalah dengan memberikan nilai pada K_d (*Gain Derivative*), hal ini bertujuan untuk mengecilkan nilai amplitudo sehingga osilasi dapat diredam atau bahkan dihilangkan. Kemudian proses terakhir pada tunning nilai *Gain* adalah dengan mencari nilai K_i (*Gain Integral*), tunning K_i diperlukan jika kondisi sistem memiliki *steady state error*, yakni terjadi selisih antara nilai set point dengan nilai sistem saat mencapai kondisi *steady state*.

a. Algoritma PID DCS

Algoritma PID pada DCS mengadaptasi dari rumus dasar perhitungan PID konvensional. PID pada DCS memiliki 4 (empat) buah kombinasi yaitu :

1) *Equation* A : P + I + D bereaksi terhadap *error* (PV – SP).

2) *Equation* B : P + I bereaksi terhadap *error* (PV – SP) dan D bereaksi terhadap perubahan nilai PV. *Equation* B digunakan untuk mengeliminasi lonjakan derivative pada aksi kontrol yang muncul akibat perubahan SP yang terlalu jauh dari PV.

3) *Equation* C : I bereaksi terhadap *error* (PV – SP) dan P + D bereaksi terhadap perubahan nilai PV. *Equation* C memiliki respon yang paling halus dan paling pelan dalam merespon perubahan nilai SP.

4) *Equation* D : Hanya menyediakan parameter I saja.

Algoritma PID pada DCS digunakan untuk tujuan mengendalikan output secara langsung atau menjadi inputnya dari kontrol lainnya (*cascade control*). Berikut adalah algoritma PID pada DCS :

Equation* A – P, I dan D bereaksi terhadap *error

$$CV_s = K * (\frac{1+T1*s}{T1*s} * \frac{1+T2*s}{1+a*T1*s} * [PVP_s - SPP_s]) \quad \text{.....(2.8)}$$

***Equation* B – P dan I bereaksi terhadap *error*, D terhadap PV**

$$CV_s = K * (\frac{1+T1*s}{T1*s} * \frac{1+T2*s}{1+a*T1*s} * PVP_s - \frac{1+T1*s}{T1*s} * SPP_s) \quad \text{..... (2.9)}$$

***Equation* C – I bereaksi terhadap *error*, P dan D terhadap PV**

$$CV_s = K * (\frac{1+T1*s}{T1*s} * \frac{1+T2*s}{1+a*T1*s} * PVP_s - \frac{1}{T1*s} * SPP_s) \quad \text{.....(2.10)}$$

***Equation* D – Kontrol Integral saja**

$$CV_s = \frac{1}{T1*s} * [PVP_s - SPP_s] \quad \text{.....(2.11)}$$

Dengan ketentuan :

CV : *Output* dari PID (% *Full Range*)

K : *Proportional Gain*

T1 : *Time Integral* (dalam menit per repeat)

T2 : *Time Derivative* (dalam menit per repeat)

PVP : Proses Variabel dalam persen (%)

SPP : *Set Point* dalam persen (%)

Algoritma PID pada DCS memiliki 3 (tiga) mode eksekusi, yaitu :

1) *MANUAL* : Operator langsung menentukan besarnya nilai *Output* (%).
2) *AUTO* : Operator menentukan nilai SP, OP ditentukan oleh algoritma.
3) *CASCADE* : SP ditentukan oleh *master control* dan OP ditentukan oleh algoritma.

b. Algoritma PID Arduino

Arduino memiliki *library* tersendiri yang dapat mengakomodir fungsi PID Kontrol. *Library* PID pada Arduino dikembangkan dengan filosofi sederhana : PID diperintahkan untuk membaca Input (*Process Variable*/PV), lalu memerintahkan Output (*Manipulated Variable*/MV/OP) agar nilainya sama dengan Nilai yang diinginkan (*Set Point*/SP). *Library* PID Arduino juga mendukung *tunning* dengan parameter Kp (mewakili *Proportional*), Ki (mewakili *Integral*) dan Kd (Mewakili *Derivative*). Namun berdasarkan keterangan resmi dari website Arduino ada beberapa hal penting yang perlu diketahui yaitu :

1) Ada banyak cara untuk menuliskan algoritma PID ke dalam Arduino. *Library* yang telah tersedia dalam Arduino telah dikembangkan oleh banyak pihak dan memiliki kemampuan yang sama dengan industri.

2) Ketika menggunakan *library* PID pada Arduino, sifatnya *self contained.* Artinya adalah fungsi penuh PID *built in* di dalam source code. Sehingga lebih mudah digunakan, dan mampu mendukung hingga 8 (delapan) PID dalam program yang sama.

Berikut adalah fungsi PID yang didukung oleh *library* Arduino :

1) PID()

2) Compute()

3) SetMode()

4) SetOutputLimits()

5) SetTunings()

6) SetSampleTime()

7) SetControllerDirection()

8) Display Functions

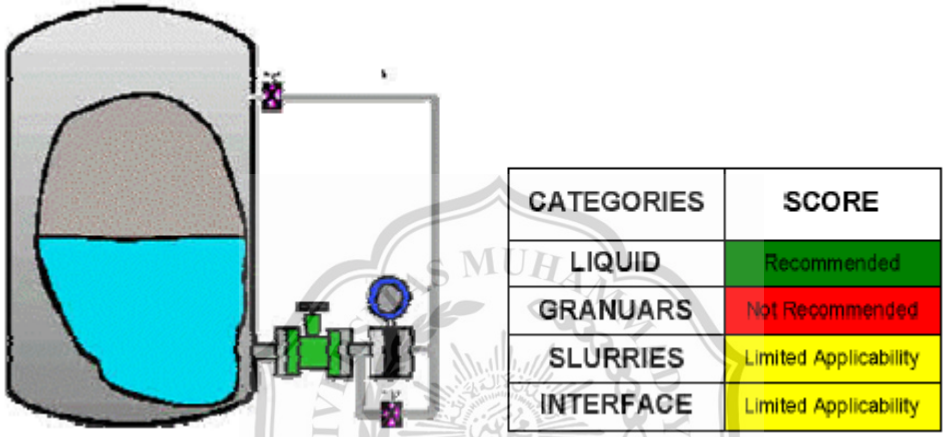
Sumber : Arduino.cc

4. Prinsip Pengukuran *Level* Fluida

Pada skripsi ini, pengukuran fluida sebagai contoh pengendalian proses dengan pertimbangan : pengendalian *level* relatif mudah karena memiliki prinsip kerja yang sederhana, waktu tunda rendah, akurasi tinggi mudah didapat, dan dapat dibandingkan dengan kondisi sebenarnya dengan mudah sehingga *error* sistem mudah dianalisa.

Pengukuran level jenis *differential pressure* (DP) didasarkan pada prinsip *hydrostatic head*. Prinsip ini mengatakan bahwa pada setiap titik di dalam fluida

yang diam (*static*), gaya yang bekerja padanya adalah sama untuk semua arah dan tidak tergantung pada volume fluida maupun bentuk ruang atau tempat dimana fluida berada, tetapi hanya bergantung pada tinggi kolom fluida di atas titik yang bersangkutan. Oleh karena itu *hydrostatic head* sering dinyatakan dalam satuan tekanan.



CATEGORIES	SCORE
LIQUID	Recommended
GRANUARS	Not Recommended
SLURRIES	Limited Applicability
INTERFACE	Limited Applicability

Gambar 2.18 Pengukuran *level* dengan prinsip beda tekanan

Hydrostatic Head dapat dinyatakan dengan persamaan :

$P = \rho * g * h$ …… (2.12)

Dimana :

P : Tekanan *hydrostatic head*

ρ : *fluid density*

g : *gravity acceleration constant* (9,81m/s^2 atau 32,2ft/s^2)

h : *level* fluida

Aplikasi pengukuran *level* dengan menggunakan metoda perbedaan tekanan atau tekanan hidrostatik telah mengalami kemajuan yang signifikan beberapa tahun lalu. Peralatan D/P ini memungkinkan untuk mengukur *level* dengan *range* yang lebar pada *services* yang bersih, korosif, *slurry* dan *high viscous*. Hampir semua

jenis peralatan D/P dapat digunakan untuk mengukur *level* jika peralatan tersebut tersedia dalam *range* yang diperlukan untul *level* yang dimaksud. Pada umumnya range D/P untuk *level* adalah sekitar (10 ~ 150) inches H_20.

Peralatan *differential pressure* diklasifikasin menjadi 2 (dua) yaitu :

1. Non Sealed System

Peralatan *differential pressure* (D/P *cell transmitter*) seperti pada gambar di bawah biasanya digunakan untuk mengukur *flow*, namun dapat juga digunakan untuk mengukur *level*. Peralatan D/P ini dalam aplikasinya digunakan secara kontak langsung dengan fluida dan dapat dibersihkan dengan gas atau liquid yang sesuai.

(a) (b)

Gambar 2.19 (a) *Differential pressure transmitter* dan (b) D/P untuk aplikasi pengukuran *level non sealed*

(Sumber : Dasar Instrumentasi dan Sistem Kontrol. BPST. 2007)

Kelebihan *Non Sealed Level Measurement*

1) Akurasi baik.
2) Dapat digunakan pada *range level* yang lebar.
3) Tersedia didalam banyak material konstruksi.

4) Dapat dibersihkan (*purge*) untuk penggunaan *service* yang korosif dan *slurry*.

5) Biaya pengadaan awal : sedang (moderat).

6) Dapat diisolasi dan *zero* ditempat.

Kekurangan *Non Sealed Level Measurement*

1) Kesalahan (*error*) disebabkan oleh *density* yang bervariasi.

2) *Lead line/impuls line* (*low pressure*) tidak dibutuhkan pada aplikasi *atmospheric*.

3) Pemanasan (*heating*) pada *lead line/impuls line* kadang-kadang dibutuhkan.

4) Problem operasi dan *maintenance* sering terjadi disebabkan kegagalan *purged lines.*

5) Perbersihan material sering dilakukan pada servis proses yang sulit.

2. Non Sealed System

(a) (b)

Gambar 2.20 (a) *Built in sealed Differential Pressure Transmitter* dan (b) D/P untuk Aplikasi Pengukuran *Level Sealed*

(Sumber : Dasar Instrumentasi dan Sistem Kontrol. BPST. 2007)

Untuk memenuhi persyaratan aplikasi pengukuran level yang sulit misalnya pada material seperti *slurry* dan *high viscous, sealed system* sering memberikan solusi yang sesuai untuk pengukuran level tersebut. Gambar di bawah memperlihatkan D/P cell jenis *sealed system*, di mana *measuring element* terisolasi dari cairan proses (*process liquid*).

Kelebihan *Sealed Level Measurement*

1) Purge tidak diperlukan
2) Baik untuk *slurry* dan material yang korosif.
3) *Range* pengukuran : lebar.
4) Akurasi : sedang ~ tinggi
5) Dapat digunakan untuk *vessel* yang terbuka atau tertutup.
6) Baik untuk *temperature* relatif tinggi.
7) Pemasangan *simple* dan mudah.

Kekurangan *Sealed Level Measurement*

1) Unit tidak dapat dilepas untuk tujuan *maintenance* tanpa *shutdown* peralatan (*equipment*).
2) *Density* yang bervariasi menyebabkan *error.*
3) Letak pemasangan harus dipertimbangkan sehubungan dengan pengaruh pada kalibrasi.
4) Perubahan *temperature ambient* menyebabkan *error* pada jenis "*capillary filled system*".

Sumber : Dasar Instrumentasi dan Sistem Kontrol. BPST. 2007