BAB III

PERENCANAAN DAN PERANCANGAN ALAT

Weighfeeder conveyor digunakan untuk memindah material dengan putaran dari motor sebagai pengerak utama yang terhubung dengan *drum*/*pulley* yang diselubungi oleh sabuk/*belt*. Untuk mengukur berat material menggunakan sensor berat *load cell*, sedangkan sebagai pengukur kecepatan *belt* menggunakan sensor *tachometer.* Dari variabel proses berupa berat dan kecepatan tersebut maka dapat diketahui laju material yang melalui *belt conveyor*. Laju material pada *weighfeeder conveyor* dapat diatur sesuai kebutuhan proses dengan adanya sistem pengendali.

Sebagai pengendali utama *weighfeeder conveyor* menggunakan mikrokontroler ATMega 16 untuk memproses variabel-variabel proses. Variabel proses diterima mikrokontroler melalui rangkaian pengkondisi sinyal dan rangkaian op-amp *comparator.* Mikrokontroler mengeluarkan sinyal aktuasi yang diumpankan ke *driver* motor penggerak *belt-conveyor*, dan mentransmisikan data hasil pemrosesan ke *notebook*/*laptop* untuk pemantauan/*monitoring* dengan rangkaian komunikasi serial RS232. Mikrokontroler sebagai pengendali utama dan *notebook* sebagai pemantauan bekerja berdasarkan algoritma pemrograman yang dibuat dengan menggunakan bahasa *Visual Basic* 2010 dan *Borland* C++.

3.1 Perancangan Sistem

Sistem *weighfeeder conveyor* ini secara keseluruhan terdiri dari beberapa bagian, antara lain :

1. *Notebook*/*laptop*

   Notebook/laptop digunakan untuk memonitor dan mengkontrol *weighfeeder*. Pada *notebook* ini dibuat sebuah program aplikasi yang dapat menampilkan data yang diambil dari mikrokontroler. Program aplikasi dibuat dengan bantuan perangkat lunak *Visual Basic* 2010 sehingga terbentuk sebuah GUI (*Graphic User Interface*).

2. Mikrokontroler

   Mikrokontroler sebagai *master control* akan mengolah variabel-variabel data masukan-keluaran yang dikalkulasi dengan menggunakan FLC (*Fuzzy Logic Controller*). Mikrokontroler juga mentransmisikan data ke *notebook*, dan memberikan sinyal aktuasi ke *driver*.

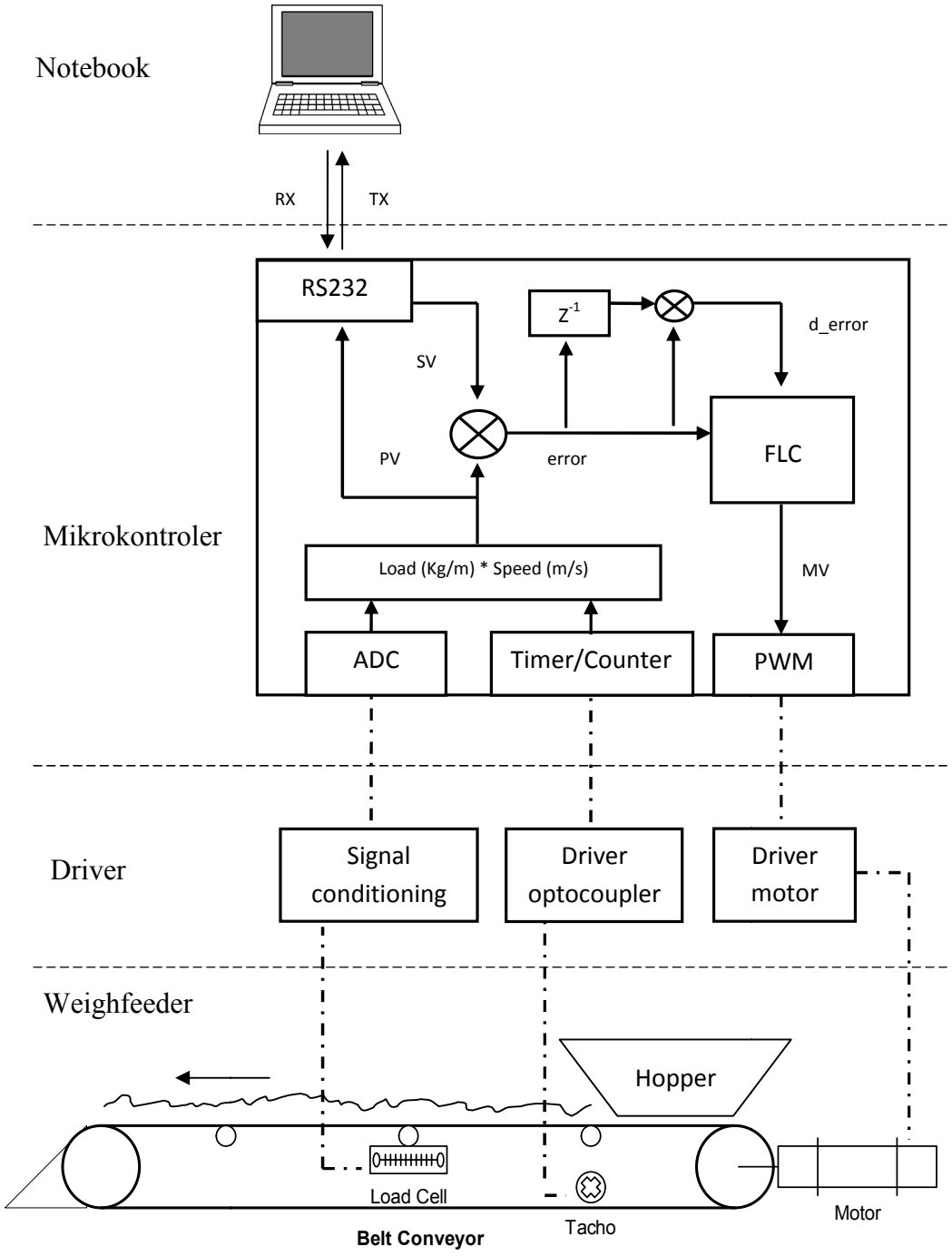
3. *Driver* dan pengkondisi sinyal

   Driver menerima sinyal aktuasi dari mikrokontroler untuk menggerakan motor *conveyor*. Pengkondisi sinyal merubah tegangan keluaran *load cell* kedalam rentang tegangan input ADC mikrokontroler (0 - 5 Volt).

4. *Weighfeeder*

   Weighfeeder berupa ban berjalan/*conveyor* yang digerakkan oleh motor untuk mengalirkan material, dengan sinyal aktuasi yang diterima dari mikrokontroler.

Dalam bentuk diagram sistem *weighfeeder conveyor* dapat dilihat pada gambar 3.1.



Gambar 3.1 Diagram sistem *weighfeeder conveyor*

3.2 Perangkat Mekanik

Perangkat mekanik pada *weighfeeder conveyor* ini terdiri dari beberapa bagian yang memiliki fungsi masing-masing, antara lain :

1. *Frame*

Frame merupakan besi *plate* siku yang dirangkai membentuk seperti meja. *Frame* ini memiliki fungsi utama sebagai tumpuan *roll*, *hopper* dan *load cell*.

2. *Roll*

Roll merupakan roda panjang yang dapat berputar karena terdapat *bearing* pada poros roda. Beberapa r*oll* ditempatkan sejajar pada *frame* sehingga *belt* yang berada diatasnya dapat berjalan untuk mengalirkan material.

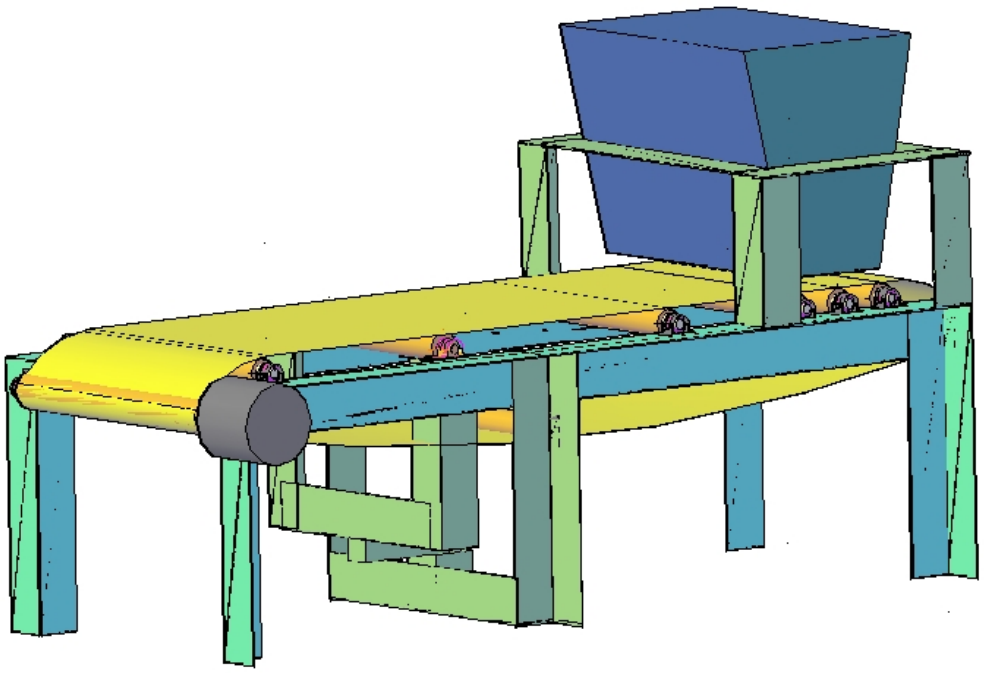
3. *Belt conveyor*

Belt conveyor merupakan lembaran kain berlapis karet panjang melingkari susunan *roll*. *Belt conveyor* menjadi tempat jatuhnya material dari *hopper* menuju penimbangan dan diumpankan ke wadah selanjutnya.

4. *Hopper*

Hopper merupakan wadah penampung material berbentuk kerucut memiliki pintu/*gate* yang terbuka dibagian bawah untuk jatuhnya material.

Bagian – bagian mekanik tersebut dirangkai sedemikian rupa sehingga *belt conveyor* dapat berputar dengan baik. Dalam gambar 3.2 terlihat ilustrasi desain konstruksi *weighfeeder conveyor*.



Gambar 3.2 Desain konstruksi *weighfeeder conveyor*

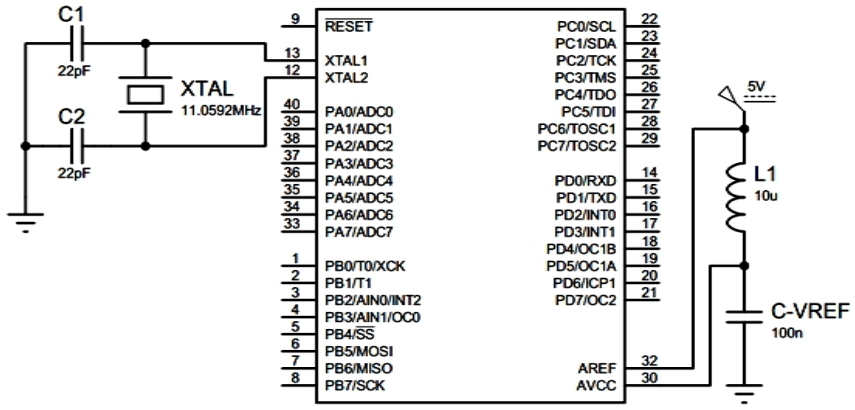
3.3 Perangkat Elektronik

Perangkat elektronik pada sistem ini bekerja dengan diberikan sumber tegangan 12, 10, dan 5 volt DC. Sumber tegangan tersebut didistribusikan oleh sebuah *multivariable power supply* pada masing-masing perangkat sesuai kebutuhan tegangan kerjanya. Perangkat elektronik yang mendukung sistem *weighfeeder conveyor* ini terdiri dari beberapa bagian yang memiliki fungsi masing-masing, antara lain sistem minimum ATMega 16, *driver* RS232, *loadcell*, pengkondisi sinyal *loadcell*, *driver motor*, *tachometer*, *motor*.

3.3.1 Sistem Minimum ATMega 16

Pada bagian ini mikrokontroler ATMega 16 dirangkai dengan beberapa komponen pendukung sehingga siklus pengendali dapat bekerja secara kontinyu.

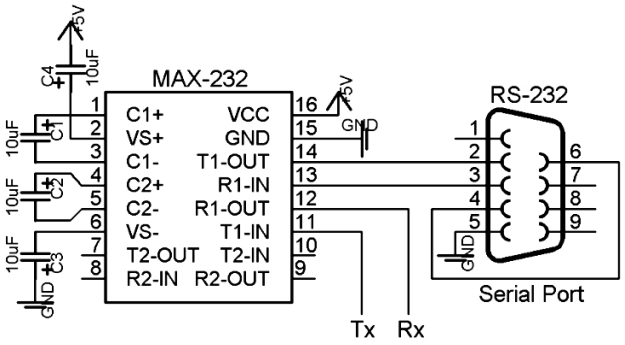
Komponen minimal yang digunakan agar prosesor dapat bekerja yaitu, kristal 11.0592 MHz dan kapasitor 22 pF sebagai sumber *clock*, dan catu daya 5 volt DC. Beberapa komponen lain juga digunakan seperti induktor dan kapasitor sebagai penyetabil tegangan referensi. Skema sistem minimum ATMega 16 dapat dilihat pada gambar 3.3 berikut.



Gambar 3.3 Sistem minimum ATMega16

3.3.2 *Driver* RS-232

Untuk dapat melakukan transmisi data antara *notebook* dengan mikrokontroler, maka sistem minimum ATMega 16 ditambahkan rangkaian driver berupa IC max232 sebagai *interface*. Karena *notebook* tidak memiliki port serial RS232, maka ditambahkan sebuah kabel konverter *usb to serial*. Dengan adanya rangkaian driver IC max232 pada sistem minimum ATMega 16 dan kabel konverter *usb to serial* pada *notebook* maka *intefacing* antara *notebook* dengan mikrokontroler dapat berlangsung. Skema *driver* RS-232 dapat dilihat pada gambar 3.4 berikut.



Gambar 3.4 Skema *driver* RS-232

3.3.3 *Load Cell*

Load cell disini digunakan sebagai sensor penimbang berat material yang dialirkan oleh *belt conveyor*. Kapasitas maksimum *load* cell yang digunakan yaitu 10 kg dengan karakteristik tegangan sebesar 2 mV/V. Apabila tegangan *supply load cell* yang diberikan yaitu sebesar 20 V, maka dalam kondisi beban *load cell* maksimal tegangan *output* sebesar 40 mV. Dalam tabel 3.1 dapat diketahui hasil perhitungan antara beban dan tegangan *output load cell* dengan tegangan eksitasi sebesar 20 V.

Tabel 3.1 Perhitungan *output load cell*

Beban (kg)	***Output* (mV)**
0	0
1	4
2	8
3	12
4	16
5	20
6	24
7	28
8	32
9	36
10	40

3.3.4 Pengkondisi Sinyal *Load Cell*

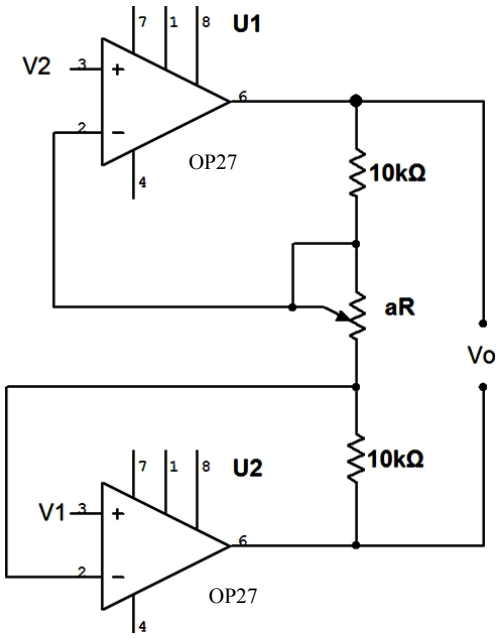
Tegangan *output load cell* yang masih terlalu kecil dikuatkan lagi pada rangkaian pengkondisi sinyal. Pengkondisian sinyal menggunakan IC jenis op - amp yang dirangkai dalam bentuk *instrument amplifier*, *differential amplifier*, dan *inverting adder*.

Rangkaian pengkondisi sinyal berfungsi sebagai penguat sinyal *output* yang dihasilkan *loadcell*. Besar penguatan pada pengkondisi sinyal ini dapat diatur sesuai keinginan dengan merubah nilai tahanan yang ada pada rangkaian *instrument amplifier*. Penguatan ini ditujukan agar tegangan yang akan dimasukkan ke *input* ADC mikrokontroler masih dalam rentang tegangan yang diizinkan, atau dapat diatur lagi menurut beban efektif.

Tabel 3.2 Perhitungan *output instrument amplifier* (gain = 125)

Beban (kg)	***Loadcell* (mV)**	***Instrument Amplifier* (V)**
0	0	0
1	4	0,5
2	8	1
3	12	1,5
4	16	2
5	20	2,5
6	24	3
7	28	3,5
8	32	4
9	36	4,5
10	40	5

Sesuai tabel 3.2 diatas diketahui tegangan *output loadcell* dalam keadaan terbebani maksimal yaitu 40 mV. Untuk dapat menyesuaikan rentang tegangan ADC mikrokontroler antara 0 – 5 volt, maka *gain* dari rangkaian *instrument amplifier* diatur sebesar 125.



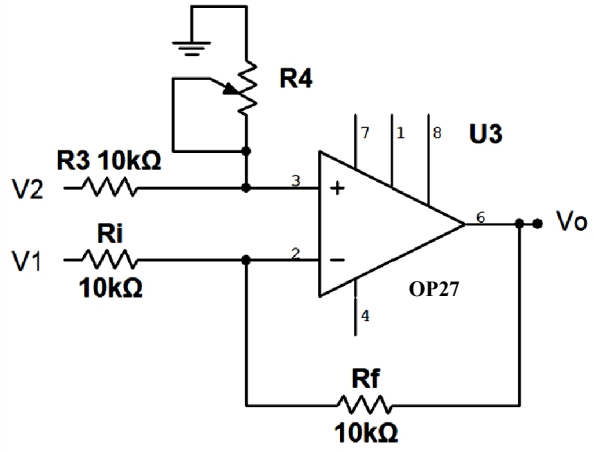
Gambar 3.5 Rangkaian *instrument amplifier*

Seperti terlihat pada gambar 3.5 dapat diketahui *output instrument amplifier* adalah $V_o = (V_2 - V_1)(1 + 2R/aR)$. Dimana *gain* adalah $(1 + 2R/aR)$, aR adalah sebuah potensiometer yang digunakan untuk mengatur nilai *gain* yang diinginkan. Jika nilai R dipilih sebesar 10 KOhm, maka :

$$1 + 2R/aR = 125$$

$$aR \cong 161$$

Maka untuk mendapatkan nilai *gain* sebesar 125, maka aR harus diatur sebesar 161 Ohm. Sinyal keluaran dari rangkaian *nstrument amplifier* ini kemudian masuk pada rangkaian *differential amplifier*, seperti terlihat pada gambar 3.6.



Gambar 3.6 Rangkaian *differential amplifier*

Rangkaian *differential amplifier* ini berfungsi untuk mencari selisih tegangan V1 dan V2 yang dideferensialkan terhadap *ground.* Potensiometer yang ada pada gambar 3.6 tersebut berfungsi sebagai penyeimbang rangkaian. Dapat diketahui bahwa $Vo = \left(\left(\frac{Rf}{Ri}+1\right)\left(\frac{R2}{R1+R2}\right)V2\right) - \left(\frac{Rf}{Ri}V1\right)$.

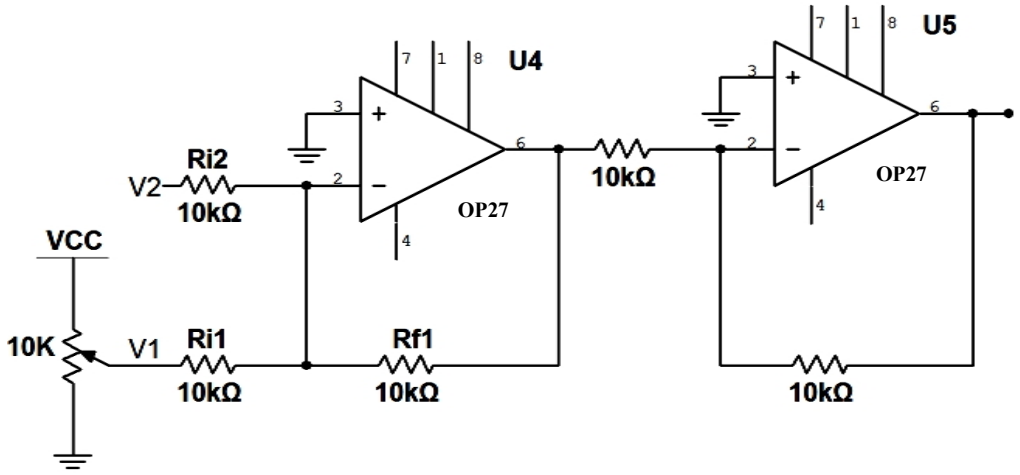
Dimana Rf=Ri=R2=R1=R, akan dihasilkan,

$$Vo = \left(\left(\frac{10}{10}+1\right)\left(\frac{10}{10+10}\right)V2\right) - \left(\frac{10}{10}V1\right)$$

$$Vo = V2 = V1$$

Penggunaan potensiometer tersebut berfungsi untuk mendapatkan selisih tegangan *input* yang seimbang.

Output dari rangkaian *differential amplifier* ini kemudian dijadikan masukan rangkaian *inverting adder* seperti terlihat pada gambar 3.7.



Gambar 3.7 Rangkaian *inverting adder*

Rangkaian *inverting adder* ini berfungsi sebagai penjumlah *input*. Dalam hal ini *inverting adder* bekerja sebagai rangkaian kalibrasi (untuk mengurangi atau menurunkan beban awal yang terdapat pada *belt conveyor*). Dengan mengubah nilai potensiometer yang terdapat pada rangkaian *inverting adder*, maka beban terhitung mulai dari nol.

3.3.5 *Tachometer*

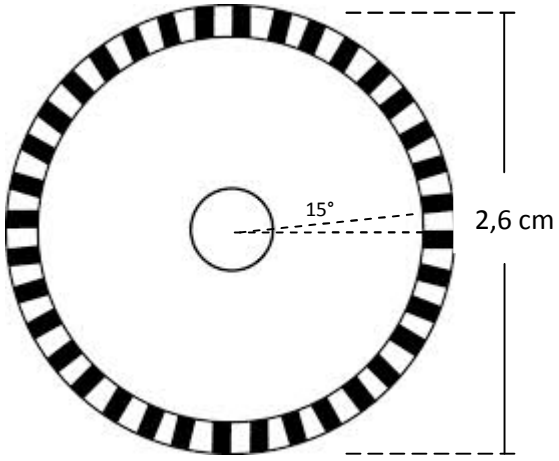
Tachometer ini terdiri dari dua bagian, yaitu piringan berlubang dan sensor *optocoupler*. Sensor *optocoupler* digunakan untuk mendeteksi adanya lubang ketika piringan berputar. Kemudian *optocoupler* tersebut mengirim sinyal ke rangkaian op-amp komparator sebagai pembangkit pulsa. Sinyal pulsa tersebut dimasukkan ke mikrokontroler dan dihitung untuk dikonversi menjadi data kecepatan.

Piringan pada *tachometer* ini berdiameter 2,6 cm, sehingga dapat diketahui kelilingnya dengan menggunakan rumus keliling lingkaran, yaitu :

Keliling $= 2\pi r = \pi D$

$= 3{,}14 \times 2{,}6$

$= 8{,}164\ cm$



Gambar 3.8 Gambar ilustrasi *tachometer discs*

Seperti yang diilustrasikan pada gambar 3.8, lubang *tachometer discs* berjumlah 24, dengan jarak sudut 15° antara lubangnya. Sehingga apabila piringan berputar satu putaran penuh (24 lubang) sama dengan bergerak sejauh 8,164 cm. Dengan diketahuinya keliling piringan ini, maka kecepatan *belt conveyor* dapat dihitung komparasi antara jarak dengan waktu. Perhitungan ini dilakukan pada satu *coding* penghitung kecepatan yang terdapat pada program mikrokontroler.

Berikut dicontohkan apabila *belt conveyor* berputar sehingga dideteksi putaran *tachometer* sebanyak setengah putaran (12 lubang = 4,082 cm) dalam satu detik, maka kecepatan *belt conveyor* adalah :

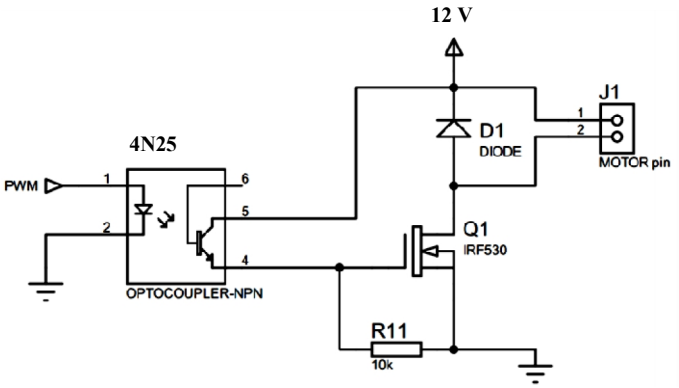
$$v = \frac{s}{t}$$

$$v = \frac{4{,}082}{1}$$

$v = 4{,}082$ cm/detik

3.3.6 *Driver Motor*

Driver motor ini sebagai aktuator yang dapat meregulasi tegangan *supply* motor DC sehingga kecepatan dapat dikendalikan. Rangkaian ini menerima sinyal PWM dari mikrokontroler melalui *optocoupler*. *Optocoupler* ini akan menyalurkan variabel tegangan pada kaki *gate* MOSFET sehingga arus IDS berubah pada daerah kerjanya. *Optocoupler* juga berfungsi sebagai isolator untuk mengamankan mikrokontroler. Gambar 3.9 berikut menujukkan rangkaian driver motor DC.



Gambar 3.9 Rangkaian *driver* motor DC

3.3.7 Motor

Jenis motor yang digunakan pada sistem ini yaitu motor DC. Dipilih motor DC karena lebih mudah untuk diatur kecepatannya. Jenis motor DC yang dipilih ini juga memiliki *gear box* yang dapat menambah torsi, sehingga mampu menggerakkan *roll* dan belt conveyor.

Dari berbagai jenis motor DC yang ada di pasaran, dipilihlah motor DC yang telah memiliki *gear box* dengan spesifikasi di dalam tabel 3.3 sebagai berikut:

Tabel 3.3 Spesifikasi motor

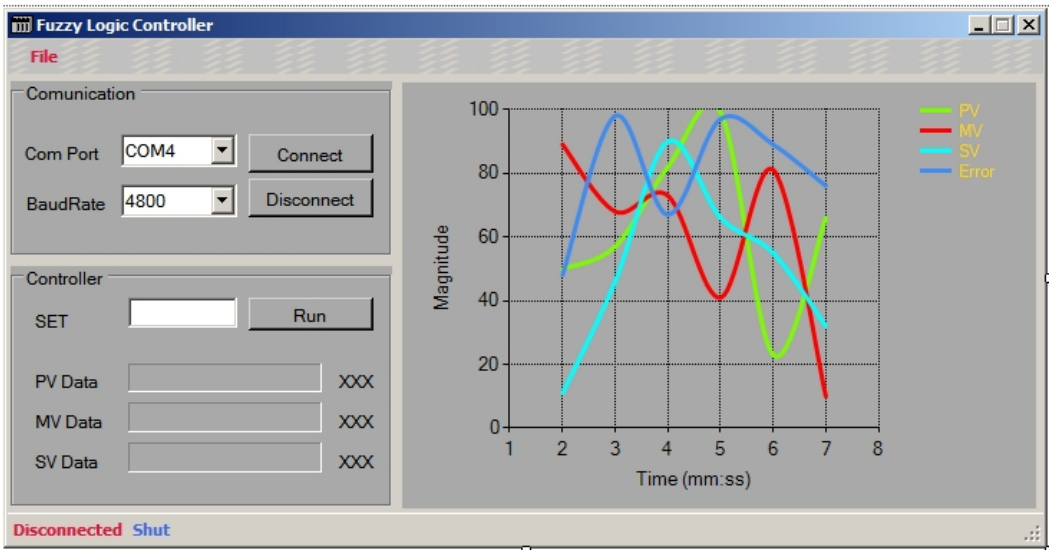
Spesifikasi	Nilai
Tegangan	18 VDC
Kecepatan	79 Rpm
Torsi	> 50 Kg

3.4 Perangkat Lunak

Perangkat lunak pada sistem *weighfeeder conveyor* ini menggunakan dua macam basis permrogaman yaitu bahasa basic dan bahasa C. Pemrograman bahasa basic terdapat pada *notebook*/*laptop*, dan bahasa C pada mikrokontroler. Pemrograman bahasa basic dibuat melalui aplikasi pendukung *visual basic* 2010, untuk menghasilkan GUI (*Graphic User Interface*). Pemrograman bahasa C dibuat dengan bantuan aplikasi pendukung *code vision* AVR, untuk menghasilkan file *.hex yang di-*download* ke mikrokontroler.

3.4.1 Pemrograman *Visual Basic* 2010

Pemrograman *visual basic* 2010 menghasilkan GUI yang digunakan sebagai antarmuka pusat operasional dalam mengendalikan *weighfeeder conveyor*. Melalui GUI ini *weighfeeder conveyor* dapat dijalankan dan dihentikan dengan nilai *set-point* yang diberikan. Konfigurasi komunikasi serial dapat disesuaikan berdasar alamat *port* dan nilai *baudrate*. Variabel proses pada *weighfeeder conveyor* ditampilkan dalam bentuk grafik dan angka, sehingga pemantauan proses dapat dilakukan dengan mudah. Desain dari perancangan GUI dapat dilihat pada gambar 3.10 berikut.



Gambar 3.10 Desain GUI

Semua variabel data proses ditransmisikan melalui komunikasi serial dengan membuat suatu protokol komunikasi, sehingga penyajian data dapat berlangsung dengan baik. Protokol komunikasi yang digunakan dapat dilihat pada tabel 3.4 berikut.

Tabel 3.4 Protokol komunikasi serial

Protokol	Definisi/fungsi data
v	Mengaktifkan PWM
w	Menghentikan PWM
x	Membaca data PV
y	Membaca data MV

Algoritma pemrograman pada *visual basic* 2010 ini adalah sebagai berikut:

a. *Setting* konfigurasi dan hubungkan *serial port*.
b. Masukkan nilai *set-point flow rate.*
c. Kirim protokol "v" untuk mengaktifkan PWM.
d. Kirim nilai *set-point.*
e. Aktifkan *timer*.
f. Kirim protokol "x" untuk data PV.
g. Baca data PV dan tampilkan.
h. Kirim protokol "y" untuk data MV.
i. Baca data MV dan tampilkan.
j. Kirim protokol "w" untuk menghentikan PWM.

3.4.2 Desain FIS

Dengan menggunakan struktur sistem kendali kalang tertutup, dapat didefinisikan beberapa variabel yaitu perbedaan nilai acuan (*Setting Value*, SV) dengan keluaran aktual (*Process Value*, PV), sehingga terdapat galat (*Error*, E). Kemudian, *Error* (E) dan perubahan galat (*Change of Error*, CE) dijadikan

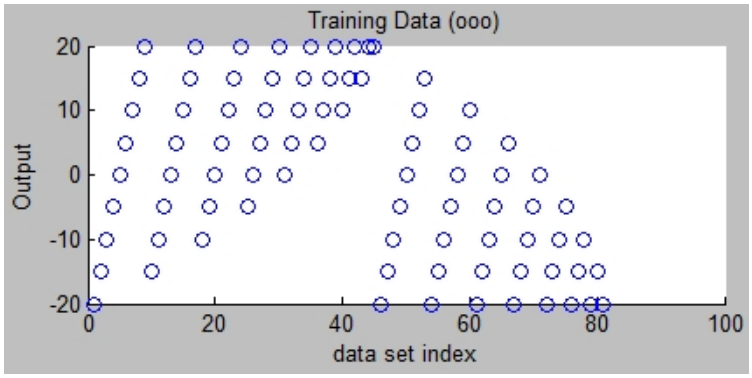
sebagai variabel masukan bagi FIS. Untuk keluaran FIS merupakan aksi kontrol (U).

Dengan mengacu pada nilai acuan *flow rate* 1000 dan 2000 gram/menit, maka dapat diperkiraan dinamika proses yang akan terjadi dalam masing-masing variabel yaitu; (E) dan (CE) pada rentang [-2000 2000]; dan (U) pada rentang [-20 20]. Dimana (E) merupakan selisih antara *set-point* dengan pembacaan *flow rate*, dan (U) adalah nilai kompensasi untuk nilai PWM. Variabel tersebut dikomposisikan berdasar kemungkinan dinamika proses yang dapat ditunjukkan pada tabel 3.5 berikut :

Tabel 3.5 Komposisi variabel proses

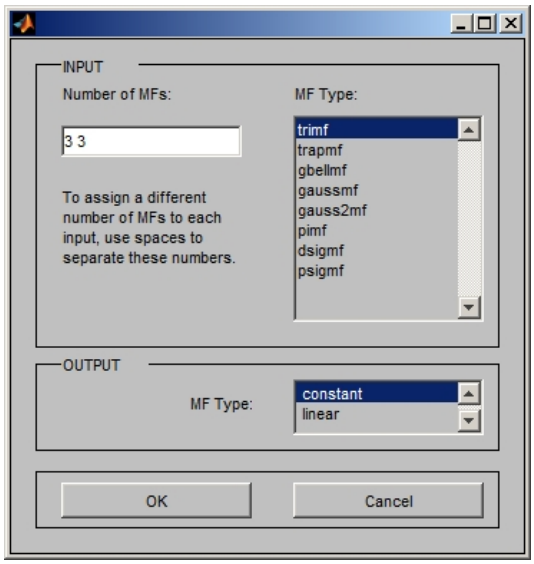
U		**CE (gr/mnt)**								
		-2000	**-1500**	**-1000**	**-500**	**0**	**500**	**1000**	**1500**	**2000**
E (gr/mnt)	**-2000**	-20	-20	-20	-20	-20	-20	-20	-20	-20
	-1500	-15	-15	-15	-15	-15	-15	-15	-15	-15
	-1000	-10	-10	-10	-10	-10	-10	-10	-10	-10
	-500	-5	-5	-5	-5	-5	-5	-5	-5	-5
	0	0	0	0	0	0	0	0	0	0
	500	5	5	5	5	5	5	5	5	5
	1000	10	10	10	10	10	10	10	10	10
	1500	15	15	15	15	15	15	15	15	15
	2000	20	20	20	20	20	20	20	20	20

Tabel 3.5 terdapat 9 baris dan 9 kolom yang menghasilkan 81 komposisi variabel untuk digunakan sebagai data *training* ANFIS. Data *training* tersebut dimasukkan pada *toolbox Matlab* ANFIS *editor* sehingga dipetakan target *output* dalam gambar 3.11 berikut.



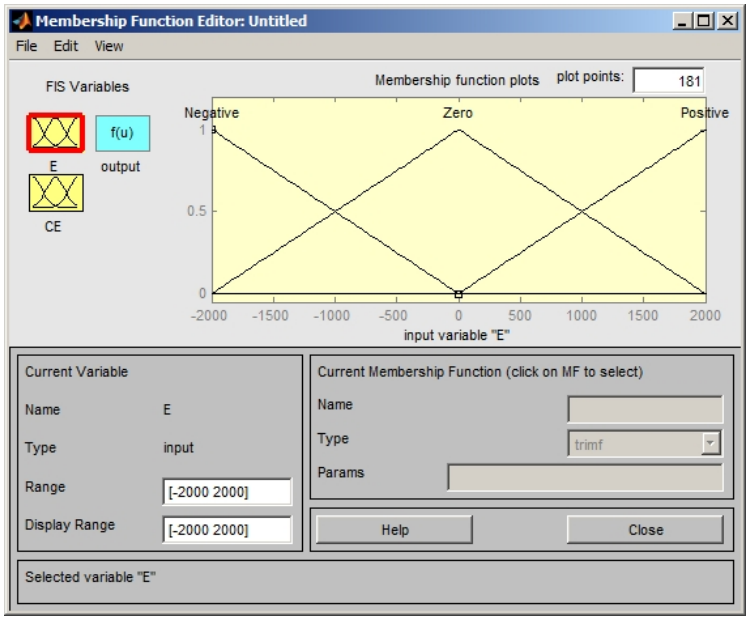
Gambar 3.11 *Plot training data*

Variabel E dan CE dibentuk dalam tiga himpunan fuzzy yaitu; *Negative* (N), *Zero* (Z), dan *Positive* (P), dengan tipe *membership fuction* yang digunakan adalah trimf. Gambar 3.12 menunjukkan konfigurasi *membership fuction input* yang melalui proses *generate* FIS *grid partition*.



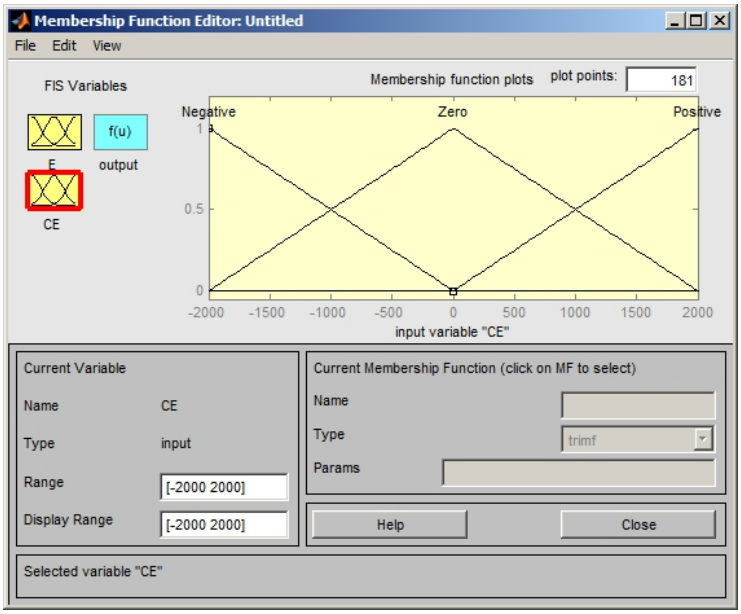
Gambar 3.12 Konfigurasi *membership fuction input*

Melalui proses konfigurasi *generate* FIS pada toolbox *matlab* ANFIS *editor* dihasilkan *plot membership function* variabel *input* E dan CE seperti pada gambar 3.13 dan 3.14 berikut.



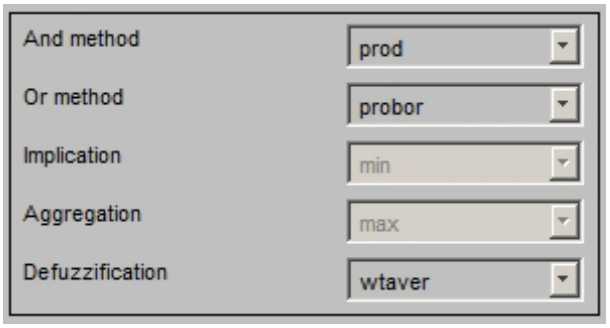
Gambar 3.13 *Plot membership function* variabel E

Diketahui sesuai pada gambar 3.13 dan gambar 3.14 bahwa, tipe *membership function negative*, *zero*, dan *positive* menggunakan trimf.



Gambar 3.14 *Plot membership function* variabel CE

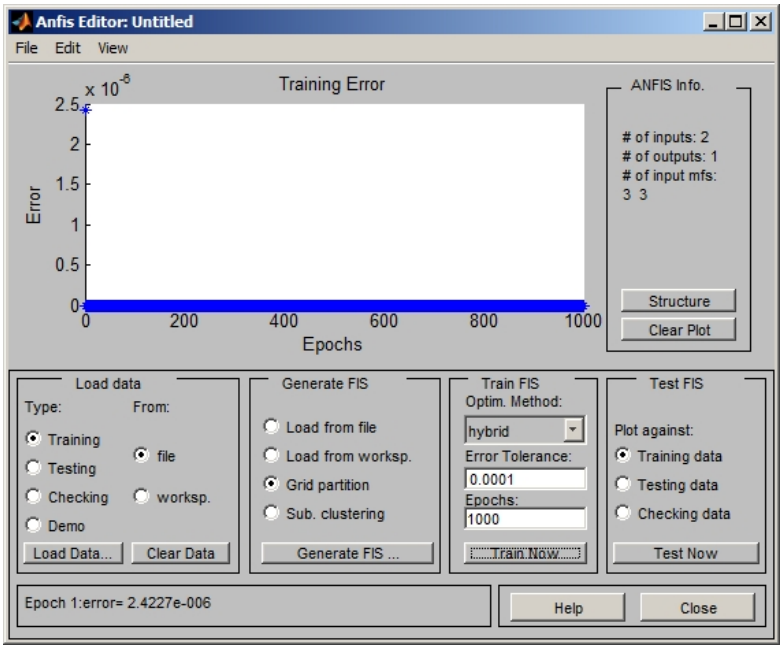
Dengan menggunakan metode fuzzy Sugeno orde-satu, maka nilai *output* (konsekuen) merupakan tipe *constant*. Pada FIS *editor* ditentukan konfigurasi sistem inferensi fuzzy yaitu; operator menggunakan operator *Product.AND*, implikasi menggunakan operator min, dan defuzifikasi menggunakan *weight average*. Bentuk konfigurasi FIS *editor* seperti pada gambar 3.15 berikut.



Gambar 3.15 Konfigurasi FIS *editor*

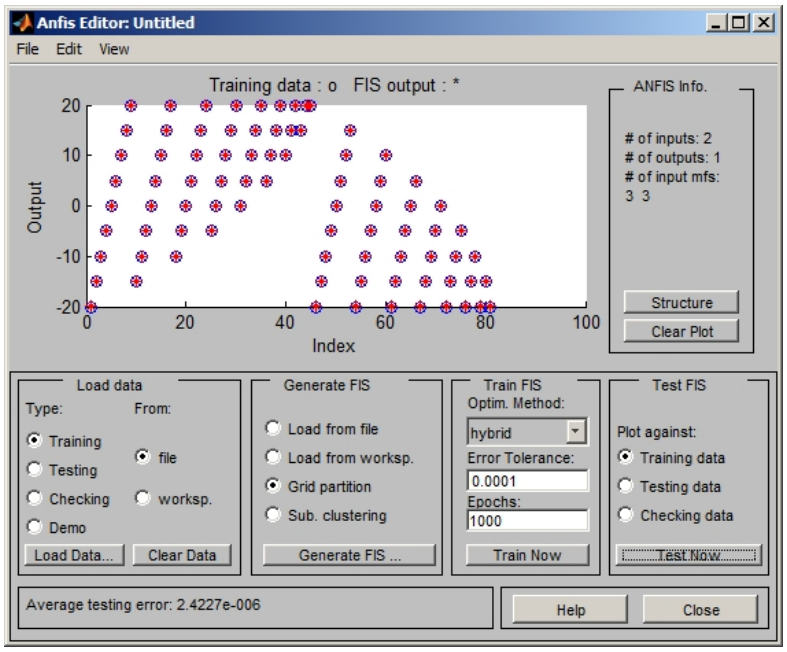
Selanjutnya konfigurasi sistem inferensi fuzzy dilatih berdasar data *input - output* dengan menggunakan ANFIS pada *toolbox matlab,* sehingga dapat menghasilkan aturan fuzzy dengan nilai-nilai konstanta dalam konsekuen/*output*.

Dari proses *training* berdasar data *input/output* dengan *error tolerance* 0.0001 dan *epoch* 1000 menghasilkan *training error* sebesar 2,4227e-06. Dalam bentuk grafik dapat dilihat *plot* dari *training error* pada gambar 3.16.



Gambar 3.16 *Plot training error*

Dengan melakukan *test* FIS dapat diketahui *plot training data* dan *output* FIS seperti pada gambar 3.17 berikut.



Gambar 3.17 *Plot training data* dan *output* FIS

Proses *training* ANFIS berdasar data *input/output* menghasilkan adaptasi terhadap parameter dari *membership function* dan 9 aturan fuzzy dengan parameter - parameter pada tabel 3.6 dan 3.7 sebagai berikut :

Tabel 3.6 Parameter *input* FIS

Variabel *Input*	**Himpunan fuzzy**	**Parameter anteseden**		
		a	**b**	**c**
E	Negative	-4000	-2000	0
	Zero	-2000	0	2000
	Positive	0	2000	4000
CE	Negative	-4000	-2000	0
	Zero	-2000	0	2000
	Positive	0	2000	4000

Dengan memilih tipe *output constant* maka diperoleh parameter konsekuen seperti pada tabel 3.7 sebagai berikut.

Tabel 3.7 Parameter *output* FIS

Aturan ke-	**k**
1	-20
2	-20
3	-20
4	-1,521e-011
5	6,471e-012
6	-5,987e-013
7	20
8	20
9	20

Parameter – parameter FIS seperti pada tabel 3.6 dan 3.7 tersebut akan diimplementasikan pada FLC dalam bentuk pemrograman mikrokontroler yang melakukan komputasi secara *real-time* sesuai keadaan *input* yang diterima dari komponen sistem *weighfeeder conveyor*.

3.4.3 Pemrograman Mikrokontroler ATMega 16

Mikrokontroler ATMega 16 sebagai *master control* diprogram menggunakan bahasa pemrograman C++ dengan *software* pendukung *code vision AVR*. Terdapat beberapa bagian *source code* yang didefinisikan sebagai variabel proses antara lain; ADC, *timer*/*counter*, PWM, USART, dan FLC.

3.4.3.1 *Source Code* ADC

Source code ADC ini berfungsi untuk menghasilkan nilai berat material yang melalui *weighfeeder conveyor*. Data ADC ini merupakan masukan tegangan 0 – 5 volt dari pengkondisi sinyal *load cell*. Melalui proses kalibrasi dengan memberikan beban acuan 200 gram, maka data ADC tersebut dimasukkan kedalam persamaan linier garis lurus $y = mx + c$ untuk menghasilkan nilai berat sebagai berikut :

0 gram = 0.5 volt = 25 ADC

200 gram = 1.6 volt = 81 ADC

y = 200

x = 81 - 25 = 56

$m = 200/56 = 3.5714$

$c = 200 - (3.5714 \times 81) = -89.285$

Jadi variabel berat dapat diketahui dengan persamaan $y = 3.5714x - 89.285$.

Konfigurasi ADC mikrokontroler ATMega 16 yang digunakan adalah ADC *interrupt* dengan kapasitas 10 bit. Dalam pemrograman *code vision AVR* dapat ditulis *source code* untuk menghasilkan nilai berat sebagai berikut :

```
interrupt [ADC_INT] void adc_isr(void)
{
unsigned int adc_data;
adc_data = ADCH;                          //menyimpan data ADCH
berat = (adc_data * 3.5714) – 89.285;     //menghitung nilai berat
}
```

3.4.3.2 *Source Code Timer/Counter*

Fasilitas *timer*/*counter* pada ATMega 16 ini digunakan untuk menghitung kecepatan *weighfeeder conveyor*. *Timer*/*counter* ini berfungsi untuk membandingkan sinyal pulsa yang dibangkitkan oleh dari *tachometer*. Dengan mengatur *time sampling* dalam satu detik maka dapat dihitung nilai kecepatan dalam satuan *meter per second* (m/s).

Untuk menghasilkan waktu satu detik perlu dilakukan *setting* register dari *timer*/*counter*, dalam hal ini yang digunakan adalah *timer2*. *Setting* registernya yaitu; ASSR, TCCR2, TCNT2, dan OCR2. *Setting* register *timer2* tersebut menentukan mode, sumber dan nilai *clock* yang digunakan.

Mode *sampling* yang digunakan pada *timer2* ini yaitu *compare match interrupt*. Dimana waktu interupsi ditentukan berdasar sistem dan nilai *clock.* Nilai *clock* ditentukan sebesar 10800 Hz, sehingga dapat ditentukan banyaknya pengulangan interupsi untuk menghasilkan waktu satu detik dalam perhitungan sebagai berikut :

$$T = \frac{1}{10800} = 9{,}26 \times 10^{-5}$$

Dimisalkan periode (T) diulang 200 kali, maka : $9{,}26 \times 10^{-5} \times 200 = 0{,}01852$ detik. Sehingga untuk menghasilkan waktu satu detik, pengulangan interupsi *timer2* sebanyak; $\frac{1}{0{,}01852} = 54$ kali. Pemisalan pada pengulangan periode komparasi sebanyak 200 kali dijadikan sebagai nilai pada register OCR2 dalam bilangan hexa yaitu C8.

Pada *timer/counter1* digunakan untuk menghitung jumlah *pulse* yang dibangkitkan oleh *tachometer*. Jumlah *pulse* tersebut disimpan pada register TCNT1 *timer/counter1*. Dengan diketahui jumlah *pulse* maka dapat dihitung kecepatan dengan mengalikan antara jumlah *pulse* dengan jarak tiap lubang pada piringan *tachometer*.

Pada *interrupt sub-routine timer2* diisikan formula penghitung nilai kecepatan sebagai berikut :

```
if (++ms>=54)          //komparasi pencacah nilai ms untuk hasil waktu 1 detik
{
ms = 0;                //reset nilai ms
pulse = TCNT1;         //menyimpan data TCNT1 ke variabel pulse
MperS = 0.34 * (float)pulse;  //nilai pulsa dikalikan jarak satu lubang tacho disc
TCNT1=0;               //reset nilai TCNT1
}
```

3.4.3.3 *Source Code* PWM (*Pulse Width Modulation*)

Source code PWM digunakan untuk menghasilkan suatu nilai yang akan memberikan sinyal kontrol kecepatan motor. Rentang nilai PWM yang dimiliki

mikrokontroler ATMega 16 yaitu 0 – 255. Nilai PWM tersebut dihasilkan melalui proses akhir penghitungan oleh FLC yaitu defuzifikasi.

Sebagai pembangkit sinyal PWM digunakan fasitilitas *timer0*. Konfigurasi register *timer0* adalah dengan memberikan nilai register TCCR0=0x65. Dimana register tersebut merupakan pemilihan *mode phase correct* PWM dan *non-inverted* PWM. Konfigurasi PWM ini juga menggunakan mode *interrupt* yang menghasilkan sinyal PWM melalui register OCR0. Dalam pemrograman *code vision AVR* dapat ditulis *source code* untuk menghasilkan PWM sebagai berikut:

```
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
OCR0=defuzy;          //output PWM dari nilai output fuzzy
}
```

3.4.3.4 *Source Code* USART

Dalam proses transmisi data antara *notebook* dan mikrokontroler perlu dilakukan penyesuaian parameter satu sama lain. Konfigurasi parameter komunikasi serial pada mikrokontroler yaitu dengan mengisi register UCSRA, UCSRB, UCSRC, UBRRH, dan UBRRL. Konfigurasi parameter komunikasi serial yang digunakan adalah sesuai pada tabel 3.8 berikut:

Tabel 3.8 Parameter komunikasi serial

Parameter	Nilai data
Baudrate	4800
Data bits	8
Stop bits	1
Parity	None

Untuk menentukan konfigurasi sesuai parameter pada tabel 3.8 maka register USART di berikan sebagai berikut :

```
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x06;
UBRRH=0x00;
UBRRL=0x8F;
```

Protokol yang dikirim dari *notebook* diterima oleh mikrokontroler pada register UDR. Struktur transmisi data dalam *source code* adalah sebagai berikut :

```
while(UCSRA.7)                  //Menunggu data baru
{ dari_VB = toascii(UDR);     //konversi data UDR dan simpan
if (dari_VB == 'x')
{ printf("%ux",(unsigned int)(PVX*10)); }   //kirim data PV
else if (dari_VB == 'y')
{ printf("%uy",(unsigned int)OCR0); }         //Kirim data MV
else if (dari_VB == 'v')
{ TCCR0 = 0x65; }                                  //Start PWM
else if (dari_VB == 'w')
```

```
{ TCCR0 = 0x00; }              //Stop PWM
else
{ SV = dari_VB*100; }}         //Nilai set-point
```

3.4.3.5 *Source Code* FLC

Source code FLC ini merupakan representasi satu bentuk *membership function* dari desain FIS yang telah melalui proses *training* pada *toolbox matlab* ANFIS *editor*. Parameter-parameter *input*/*output* FIS diaplikasikan pada beberapa blok fungsi. *Source code* dalam bahasa *basic* yang dibuat melalui *software visual basic* 2010 adalah sebagai berikut:

```
Private Sub input1()
    'representasi membership function
    If i <= a1 Then
       alpA = 0
    ElseIf i >= a1 And i <= b1 Then
       alpA = (i - a1) / (b1 - a1)
    ElseIf i >= b1 And i <= c1 Then
       alpA = (c1 - i) / (c1 - b1)
     ElseIf i > c1 Then
       alpA = 0
     End If
End Sub
```