BAB III

METODE PENELITIAN

Metode penelitian yang digunakan untuk tugas akhir ini dimulai dengan studi kepustakaan, kemudian dilanjutkan dengan perancangan perangkat lunak, perancangan perangkat keras sampai akhirnya pembuatan alat.

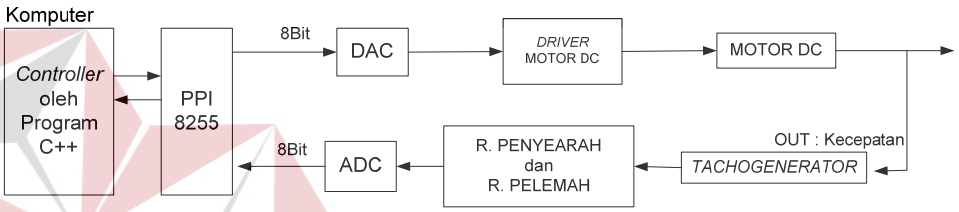
Dengan studi kepustakaan akan didapat data dan informasi yang memadai secara teori maupun praktek. Secara teori semua data yang dibutuhkan dapat dilihat dari buku, *datasheet*, ataupun dari internet. Sedangkan secara praktek data yang dibutuhkan dapat dilihat dari buku laporan tugas akhir mahasiswa yang sudah lulus, modul praktikum yang berkenaan dengan tugas akhir ini, ataupun dari *help* program yang akan digunakan. Kedua hal ini diperlukan untuk diintegrasikan membentuk sistem yang diusulkan untuk dibangun, seperti pada gambar 3.1 dibawah.

Perancangan perangkat lunak dilakukan setelah memperoleh informasi yang dibutuhkan untuk membangun suatu program *controller* yang berkenaan dengan tugas akhir yang dibuat. Program yang dibuat untuk tugas akhir ini dibangun dengan menggunakan bahasa pemrograman Borland Turbo C++ 3.0 yang akan membahas perencanaan dan pembuatan perangkat lunak untuk identifikasi karakteristik pada Motor DC. Program lain yang digunakan adalah MATLAB V6.5.

Perancangan perangkat keras dilakukan setelah memperoleh data-data teoritis dan praktis diseputar komponen – komponen elektronika yang akan digunakan. Komponen- komponen yang dipergunakan antara lain PPI 8255, DAC

0808, ADC 0804, TIP 3055, Resistor, Kapasitor, Dioda, Potensio multitune, lampu LED, dan lain sebagainya.

Setelah perancangan perangkat keras, maka dilanjutkan dengan pembuatan perangkat keras sehingga menjadi sebuah modul yang siap pakai, tapi sebelumnya harus melewati pengujian dengan program sederhana, Multimeter, Osiloskop, Rangkaian I/O ataupun RPMmeter pada bagian yang berkaitan dengan sistem kerja perangkat keras yang dibuat.



Gambar 3.1 Blok diagram alat pengidentifikasi karakteristik Motor DC

3.1 Perancangan Aplikasi Identifikasi Karakteristik Motor DC

Didalam pembuatan program aplikasi, desain tampilan yang *user friendly* adalah merupakan suatu hal yang sangat penting, selain dari tujuan utama pembuatan program tersebut. Tujuan utama program ini adalah mendapatkan suatu fungsi alih dalam *domain* s yang mewakili karakteristik dari Motor DC. Program ini menggunakan 2 pilihan metode identifikasi sistem, yaitu secara statis dan secara dinamis.

Identifikasi sistem secara statis bekerja sebagai berikut, langkah pertama adalah memberikan tegangan *unit step* untuk Motor sesuai *input*-an dari *user* dengan nilai minimal 5V dan maksimal 12V, kemudian mengambil data dari *output* Motor DC sebanyak 4 kali, setiap 1 kali pengambilan dilakukan *sampling*

data sebanyak 30 kali, dari keempat data tersebut di rata – rata kemudian akan di analisa sehingga didapatkan nilai fungsi alihnya dalam *domain* s.

Identifikasi sistem yang dinamis bekerja dengan memberikan masukan tegangan *input* yang berubah – ubah sesuai dengan deret PRBS (*Pseudo Random Binary Sequence*) yang dibangkitkan dari proses *shift register* dengan *feedback*. Disini digunakan deret PRBS dengan panjang deret 31 sel, dengan masukan 5 sel. Kemudian keluaran dari Motor DC disimpan ke sebuah *file*, dari *file* ini nantinya akan diteruskan dengan menggunakan program MATLAB untuk bisa dianalisa sehingga akan menghasilkan fungsi alih sistem atau *plant*.

3.1.1 Perencanaan dan pembuatan program

Karena Borland Turbo C++ 3.0 berbasis pada DOS, maka ukuran program yang akan dihasilkan kecil dan komputer yg di butuhkan juga tidak memerlukan spesifikasi yang tinggi. Pemrograman Turbo C++ ini mendukung mode grafik yang akan menghasilkan suatu tampilan yang lebih bagus di bandingkan dengan mode teks. Program yang dibuat menggunakan dua mode yg ada. Mode teks digunakan untuk menampilkan nilai – nilai yg langsung di dapat dari pengukuran maupun nilai – nilai hasil analisis, sedangkan mode grafik di gunakan untuk tampilan program, menu, grafik, menuliskan fungsi alih, dan sebagainya. Program ini menggunakan pengoperasian *file* sebagai media penyimpanan data. Sehingga data yg di dapat, dapat di analisa secara manual dan dapat di ketahui ke absahan hasil akhirnya, yaitu fungsi alih dalam *domain* s. *File* yang dihasilkan dapat di akses langsung dengan Microsoft Excel sehingga dapat diketahui grafiknya sesuai atau tidak dengan hasil pengukuran.

Program lain yang digunakan adalah program MATLAB, program ini umum digunakan di dalam analisis dan desain sistem kontrol, karena kemudahan penggunaannya dan mempunyai banyak fungsi yang mendukung. Meski termasuk program kompleks yang disederhanakan seperti pembuatan grafik dari suatu model matematika, pembuatan fungsi dari data – data, dan lain sebagainya. Program ini berjalan dibawah sistem operasi keluarga Windows, Unix, Linux, dll. Untuk tugas akhir ini digunakan Windows 98, sehingga komputer yang digunakan tidak memerlukan spesifikasi *hardware* yang tinggi.

Program Borland C++ 3.0 mendukung untuk pengoperasian *hardware* di luar komputer, tapi tentunya harus menggunakan suatu media penghubung, dalam Tugas Akhir ini dipakai PPI sebagai media penghubung antara komputer dan *hardware* di luar komputer atau *plant*.

3.1.2 Inisialisasi pada program

Komponen – komponen yang membangun program ini haruslah di inisialisasi pada bagian awal program, inisialisasi yang diperlukan adalah inisialisasi *port* PPI, inisialisai grafik untuk penggunaan mode grafik dari komputer serta inisialisasi *file* untuk output dari alat ini.

Untuk dapat digunakan, PPI tersebut terlebih dahulu di inisialisasi dengan memberikan alamat I/O (*Input* / *Output*) yang tidak dipakai oleh komputer. Biasanya digunakan alamat 0300H – 0303H atau 03E0H – 03E3H. Alamat ini juga di gunakan di PPI dengan mengatur *DIP Switch* yang disediakan sehingga sama dengan alamat yang kosong yang digunakan untuk program ini.

Inisialisasi *port* – *port* PPI di C++ sangatlah mudah, untuk alat ini digunakan alamat 03E0H – 03E3H, inisialisasinya seperti berikut

```
#define porta 0x3E0       :   Port A menggunakan alamat 03E0h
#define portb 0x3E1       :   Port B menggunakan alamat 03E1h
#define portc 0x3E2       :   Port C menggunakan alamat 03E2h
#define Cw 0x3e3          :   CW (control word) menggunakan alamat 03e3h

Outportb (Cw,0x89)        :   mengeset Port A sebagai output PPI, Port B
                              sebagai output PPI, port C sebagai input PPI, mode
                              operasi 0.
```

Karena program ini beroperasi dalam mode grafik juga, maka harus diadakan inisialisasi terhadap grafik dari komputer yang digunakan. Adapun inisialisasinya sebagai berikut :

```
/* request auto detection */
 int gdriver = DETECT, gmode, errorcode;

/* initialize graphics and local variables */
 initgraph(&gdriver, &gmode, "c:\\tc\\bgi");

/* read result of initialization */
 errorcode = graphresult();

/* an error occurred */
 if (errorcode != grOk)
 {
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  getch();
  exit(1);
 }
```

Pertama pembuatan variabel *gdriver*, *gmode*, *errorcode* dengan tipe integer, kemudian pendeteksi otomatis ke grafik komputer apakah VGA, EGA, CGA, atau yang lain, untuk komputer sekarang ini biasanya terdeteksi VGA dengan kelas *high* yang mempunyai ruang resolusi 480 kolom x 640 baris.

Inisialisasi yang lain adalah inisialisasi terhadap *file output* dari program. Disini berguna untuk memeriksa apakah *file output* tersebut sudah ada, kalau belum ada maka akan dibuat, jika sudah ada maka isi dari *file* tersebut akan dihapus semuanya. Inisialisasinya sebagai berikut :

```
/* Inisialisasi file */
 if ((MotorDC=fopen("C:\\Tc\\Data\\Motor.csv","w")) == NULL)
 { printf("KESALAHAN : File Tidak Dapat di buka !!\7\n");
   exit(1);
 }
```

Pengecekan *file* dilakukan dengan *option* w diatas, *option* lain yang ada adalah r, yaitu operasi yang dilakukan hanyalah sebatas membaca isi dari *file*

tersebut, untuk mengisi *file* tersebut dapat digunakan *option* a seperti pada prosedur untuk *entry* data ke *file*.

3.2 Perancangan Perangkat Lunak Untuk Identifikasi Sistem Secara Statis

Identifikasi sistem secara statis mempunyai alur kerja yang lebih rumit daripada identifikasi sistem secara dinamis. Untuk identifikasi stastis dilakukan 4 kali pengambilan data, dalam 1 kali pengambilan data diambil 30 kali *sampling* data dengan periode *sampling* = 0.2 detik. Dari ke empat kali percobaan tadi diambil rata – rata untuk dijadikan data ke 5 sebagai data yang mewakili dari identifikasi statis yang didasarkan pada ke empat data tersebut.

Flowchart untuk program identifikasi secara statis bisa dilihat pada gambar 3.2 dibawah. Beberapa prosedur penting untuk identifikasi statis adalah prosedur ambil data, prosedur pengambilan rata – rata dari keempat hasil pengambilan data, prosedur *mid* yang berguna untuk mencari suatu nilai dalam sebuat *file* dan yang terpenting adalah prosedur analisa data terhadap data yang diambil dan data hasil rata – rata.

3.2.1 Prosedur ambil data

Pengambilan data dilakukan sebanyak 30 kali dalam 1 kali eksperimen. Pengambilan data diambil dari output Motor DC melalui *Port* C PPI yang beralamat 3E2H, di program ini hanya data yang mempunyai nomor urut genap saja yang di tampilkan di layar, dikarenakan keterbatasan ruang *layout* program, tapi semua nilai yang diambil tadi ditampilkan dalam bentuk grafik kartesian ke semuanya secara *real time*, sehingga dapat dilihat proses pengambilan datanya. Untuk prosedurnya sebagai berikut :

```
Vout = (inportb(PortC)/255.0)*12.0;
  q=(i%2);
  if (p==q)
  {
   gotoxy(7,(i*2)-(((i-2)*3)/2));
   printf("%0.3f\n",Vout);
  }
  if (i==1)
   line(a-30 ,b+40 ,a-20 ,(b-Vout*16)+40);
  else
   line(a+(-40+y) ,(b-z*16)+40 ,a+(-30+y) ,(b-Vout*16)+40);
```

untuk proses *inportb(portC)* hasil yang didapat masih berupa data desimal, jadi harus dikonversi ke tegangan dahulu supaya sesuai dengan apa yang di inginkan, untuk itu harus dibagi dengan 255 lalu dikalikan dengan 12V, karena tegangan maksimal dari Motor 12V.

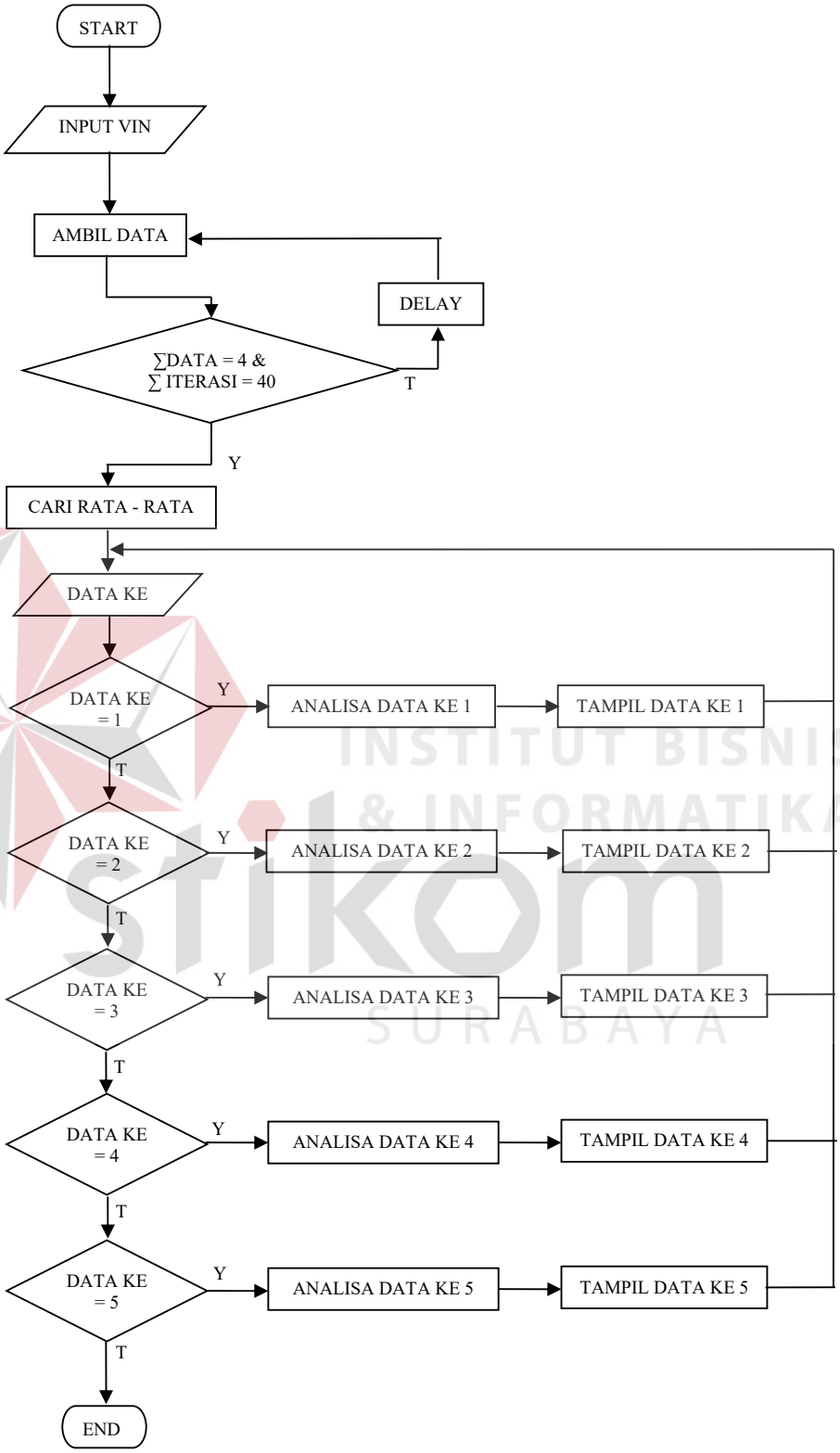
3.2.2 Prosedur rata – rata data

Prosedur ini berguna untuk mencari nilai rata - rata dari ke empat data yang diperoleh. Data ini yang nantinya dijadikan patokan sebagai hasil dari identifikasi secara statis, adapun prosedurnya seperti berikut :

```
for (int i=0;i< 30;i++)
{
 mid(string1, string5, urut1, 5);
 urut1 = urut1 + 6;
 tes01 = atof(string1);
 mid(string2, string5, urut2, 5);

 urut2 = urut2 + 6;
 tes02 = atof(string2);
 mid(string3, string5, urut3, 5);
 urut3 = urut3 + 6;
 tes03 = atof(string3);
 mid(string4, string5, urut4, 5);
 urut4 = urut4 + 6;
 tes04 = atof(string4);
 tes05 = ((tes01+tes02+tes03+tes04)/4.0);
 fprintf(MotorKU,"%0.3f;",tes05);
 fprintf(MotorDC,"%0.3f;",tes05);
}
```

Disini proses yang terjadi adalah pengambilan data lewat prosedur *mid* dari data ke 1 sampai 4 untuk data nomor urut 1 sampai 30, dalam satu kali proses dilakukan pengambilan data dari *database* yang ditulis di sebuah *file* menurut nomor urut data kemudian dijumlahkan lalu dibagi 4, kemudian hasilnya disimpan lagi ke *database* yang sama.



Gambar 3.2 *Flowchart* untuk identifikasi statis

3.2.3 Prosedur *mid* untuk mencari suatu nilai dalam *file*

Prosedur ini sangat penting dikarenakan sangat dibutuhkan untuk membaca teks dari suatu *file* yang terletak secara terstruktur seperti *database* sederhana, disini dibutuhkan nomor urut teks tersebut, kemudian panjang teksnya. Prosedurnya tidak begitu panjang, tetapi inilah salah satu kunci yang sangat penting dalam perangkat lunak di tugas akhir ini, prosedurnya seperti berikut ini :

```
void mid (char *s1, char *s2, int m, int p)
{
  int i,j,n;
  for (n=0;s2[n];n++);
  if ((m>=0) && (m<=n-1))
  {
    for (i=m,j=0;(j<p&&s2[i]);i++,j++)
    s1[j] = s2[i];
    s1[p] = '\0';
  }
  else
  s1[0] = '\0';
}
```

Prosesnya adalah sebagai berikut, pertama kali dicari panjang isi *file* tersebut, kemudian ditentukan urutan teks yang mau di ambil serta panjang teks tersebut. Demikian seterusnya proses pencarian dan pengambilan teks yang di kehendaki dalam proses pengulangan sampai selesai.

3.2.4 Prosedur analisa data

Prosedur ini merupakan prosedur yang paling penting, dan mempunyai proses yang sangat panjang, karena disinilah semua nilai – nilai yang membangun suatu fungsi alih dicari. Urutan proses yang terjadi secara garis besar sebagai berikut :

1. Temukan *Time Delay System*.

   Disini dicari nilai L (*time delay system*), yaitu waktu yang dibutuhkan oleh *plant* tersebut untuk bereaksi dari nol. Algoritmanya seperti berikut :

```
for (int i=0;i< 30;i++)
{
       mid(string1, string2, z1, 5);
```

```
        z1 = z1 + 6;
        Vout = atof(string1);
        if ((Vout > 0) && (iL == 0))
                iL = i;
)
L = iL * 0.20;
```

Pertama data sebanyak 30 tersebut dibaca melewati prosedur *mid* kemudian tiap data tersebut saling dibandingkan. Data yang memenuhi syarat adalah bernilai lebih dari nol dan nilai *iL* masih nol. Data inilah yang merupakan data pertama yang lebih dari nol, kemudian *iL* diisi dengan nilai i yang merupakan urutan data tersebut kemudian dikalikan dengan ts (*time sampling*) dan itulah L yang dicari.

2. Temukan *Time* Konstan dari Sistem

   T = waktu untuk mencapai tegangan sebesar [63,2 % * Respon Stabil]

   Respon stabil disini berarti respon dari Motor DC untuk mencapai nilai *output* tegangan yang stabil. Pertama kali dicari nilai respon stabil tersebut, adapun urutan program nya sebagai berikut :

```
for (int i=0;i< 30;i++)
 {
        mid(string1, string2, z1, 5);
        z1 = z1 + 6;
        Vout = atof(string1);
        temp  = Vout;
        if ((Vout > 0) && (iL == 0))
                iL = i;

        /* Cari Jumlah Tiap Vout yg di hasilkan */
        if (i == 1)
        { V01 = temp;
          i1 = 1;
        }
        else
         if (temp == V01)
          i1++;
         else
          if (i2 == 0)
          { V02 = temp;
                i2++;
          }
          else
                if (temp == V02)
                 i2++;
                else
                 if (i3 == 0)
                 { V03 = temp;
                        i3++;
                 }

                 else
                  if (temp == V03)
```

```
                    i3++;
                else   ..............

    if (i1>i2)    { Goodi = i1; VTemp= V01;}
    if (i2>Goodi) { Goodi = i2; VTemp= V02;}
    if (i3>Goodi) { Goodi = i3; VTemp= V03;}
    if (i4>Goodi) { Goodi = i4; VTemp= V04;}
    ...........
```

Inti dari penggalan program diatas adalah membuat variabel – variabel yang di isi dengan tiap nilai yang ada di data kemudian di cari nilai tegangan *output* atau data yang paling sering muncul, yang dapat di anggap sebagai nilai stabil tegangan yang disebut respon stabil.

Kemudian langkah selanjutnya adalah mengalikan respon stabil dengan 63,2% sehingga di peroleh tegangan untuk *time* konstan, langkah selanjutnya adalah mencari letak *time* konstan tersebut di detik ke berapa, adapun caranya seperti berikut :

```
temp = 0.632 * VTemp;
fgets(string2, sizeof(string2), MotorKU);
    for (int ii=1;ii<= 30;ii++)
    {  mid(string1, string2, z2, 5);
           z2 = z2 + 6;
           Vout = atof(string1);
           if (temp >= Vout)
           { i4TLow = ii;
             V4TLow = Vout;
           }
           else if (V4TUp == 0)
                  { V4TUp = Vout;
                        i4TUp = i4TLow + 1;
                  }
           if ((VTemp == Vout)&&(i4RS == 0)) i4RS  = ii;
           if (ii == 30) V4End = Vout;
    }
    UP  = V4TUp - temp;
    LOW = temp - V4TUp;
    ALL = UP+LOW;
    Nemo = LOW/ALL;
    i4T = i4TLow + Nemo;
    float T4T  = i4T*0.22;
    float T4Ts = i4RS*0.22;
```

3. Temukan *Time Settling* dari Sistem

   Ts = waktu yang dibutuhkan untuk mencapai nilai respon stabil pertama kali

   Disini dicari waktu dimana *Vout* (tegangan *output*) pertama kali mencapai nilai respon stabil, untuk caranya bisa dilihat program pada nomor 2 (digabung dengan program untuk mencari nilai *time* konstan) :

```
if ((VTemp == Vout)&&(i4RS == 0)) i4RS  = ii;
float T4Ts = i4RS*0.20;
```

4. Temukan Konstanta Penguatan (K)

   Disini dipakai rumus : K = (*Output* akhir – *Output* awal) / Amplitudo dari *Input Unit Step*. Untuk bagian ini tidaklah sulit dan bisa digabung di program no 2, caranya sebagi berikut :

```
if (ii == 30) V4End = Vout;
if (ii == 1) V4Start = Vout;
Gain = (VTemp-V4Start)/Vin;
```

5. Temukan *Ratio* antara *Time Delay* dengan *Time Settling*

   Rumus yang digunakan sebgai berikut : L/Ts. Disini tidaklah sulit karena nilai L dan Ts sudah didapat dan langsung dapat dipakai, penggalan programnya seperti berikut :

```
LDivTs = L / T4Ts
```

6. Lihat Tabel *STREJC* untuk menentukan letak *Ratio* yang didapat dari Langkah 5: sebagai jawaban dari pertanyaan ’Lebih dekat ke nilai manakah nilai *Ratio* ?’ (dibandingkan dengan kolom Tu/Ta)

Tabel 3.1 Tabel *STREJC* ($\tau = 0$)

N	T_a/T	T_u/T	T_u/T_a
1	1	0	0
2	2.718	0.282	0.104
3	3.695	0.805	0.218
4	4.463	1.425	0.319
5	5.119	2.100	0.410

Tabel diatas di dapat dari www.eudil.fr/eudil/belk/pra1.htm. Dari Langkah ini didapat nilai-nilai: N = Orde Sistem, [Ta/T], [Tu/T], dan [Tu/Ta]. Langkah untuk mencari nilai yang mendekati dengan yang ada di tabel ini membutuhkan urutan program yang panjang, nilai *ratio* (langkah 5) di bandingkan dengan nilai pada kolom Tu/Ta yang mendekati dari Tabel *STREJC* tersebut. Adapun programnya sebagai berikut :

```
Key = LDivTs;
LowGate = 0; UpGate = 0;

if (Key >= 0)
       LowGate = 0;
else
 { if (UpGate < Key)  UpGate = 0;
       else
       if (UpGate > 0) UpGate = 0; }
       ..................

                Left = UpGate - Key; Right = Key - LowGate;
if (Left < Right)  NewKey = UpGate;
else
NewKey = LowGate;
if (NewKey == No1)
{ Ta4T = 2.718; Tu4T = 0.282; Tu4Ta = 0.104; Orde = 2; }
if (NewKey == No2)
{ Ta4T = 3.695; Tu4T = 0.805; Tu4Ta = 0.218; Orde = 3; };
if (NewKey == No3)
{ Ta4T = 4.463; Tu4T = 1.425; Tu4Ta = 0.319; Orde = 4; };
if (NewKey == No4)
{ Ta4T = 5.119; Tu4T = 2.100; Tu4Ta = 0.410; Orde = 5; };
if (NewKey == No5)
{ Ta4T = 1; Tu4T = 0; Tu4Ta = 0; Orde = 1; };
```

7. Temukan nilai *Time* Konstan yang Baru

   Disini mempunyai rumus sebagai berikut Tbaru = *Time Settling* / [Ta/T], nilai - nilai yang dibutuhkan untuk langkah ini sudah didapat tinggal menghitungnya, potongan programnya adalah sebagai berikut :

```
NewT = T4Ts/Ta4T;
```

8. Temukan nilai *Time Delay* yang Baru

   Disini mempunyai rumus sebagai berikut Lbaru = *Time Settling* x [Tu/T], nilai yang dibutuhkan sudah ada tinggal dimasukkan dalam perhitungan, algoritmanya sebagai berikut :

```
NewL = T4Ts/Tu4T;
```

9. Temukan nilai Lamda

   Disini mempunyai rumus sebagai berikut Lamda = *Time Delay – Time Delay* Baru. Potongan programnya sebagai berikut :

```
Lamda = L - NewL;
```

10. Pilih Struktur Model yang sesuai sesuai orde yang di dapat.

$$G(s) = \frac{K}{(Ts+1)^n} e^{-sL} \tag{3.1}$$

Disinilah langkah dari analisa data berakhir, semua komponen yang membangun suatu persamaan fungsi alih sudah didapat tinggal menempelkan pada bentuk persamaan yang ada, disini bentuk persamaan ditentukan oleh *orde* hasil perhitungan dari tabel *Strejc*.

3.2.5 Prosedur untuk menampilkan data yang diinginkan

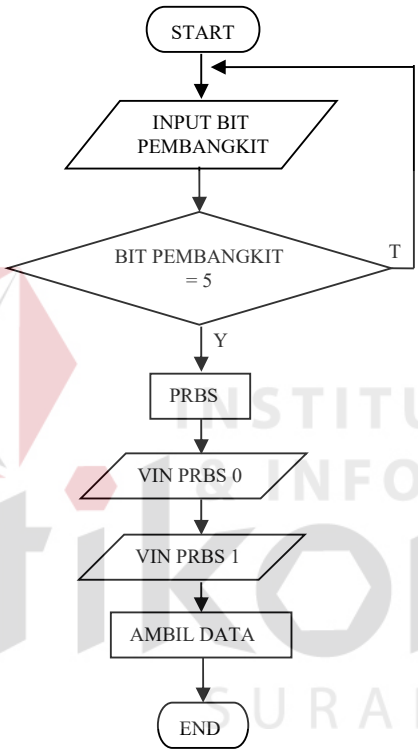
Disinilah akhir dari program untuk identifikasi sistem secara statis, dimana *user* dapat melihat hasil dari keempat percobaan dan hasil dari rata – rata secara grafik dan beserta semua nilai dari hasil analisa dan juga hasil akhir berupa persamaan fungsi alih dalam *domain* s. Hasil dari tiap percobaan serta hasil rata - rata juga dapat dilihat langsung melewati *file* bertipe *csv* yang langsung dapat diakses dengan Microsoft Excel, hal ini sebagai pembanding untuk hasil penyajian secara grafik di program dengan yang ada di Microsoft Excel.

3.3 Perancangan Perangkat Lunak Untuk Identifikasi Secara Dinamis

Identifikasi sistem secara dinamis disini bersifat sebagai pembanding untuk hasil dari perhitungan identifikasi secara statis, dan juga proses yang di lakukan program ini tidak sampai analisis data untuk menemukan persamaan

fungsi alihnya. Disini hanya sebatas mengambil data sebanyak 105, sedangkan deret minimal PRBS untuk nilai N = 5 adalah 31. Proses selanjutnya akan dilanjutkan menggunakan program MATLAB yang berfungsi untuk menganalisa sehingga mendapatkan persamaan fungsi alih yang dikehendaki..

Flowchart untuk identifikasi secara dinamis digambarkan dibawah ini :



Gambar 3.3 *Flowchart* untuk identifikasi secara dinamis

3.3.1 Prosedur pembangkitan PRBS

Disini pembangkitan deret PRBS menggunakan nilai N = 5, programnya sebagai berikut :

```
// Menggeser & menambahkan Bit ke1 = bit3+bit5
 for (int j=1;j<=iterasi;j++)
  {     buffer = A[3] + A[5];
          if ((buffer == 2)||(buffer == 0))
                buffer = 0;
          else
                buffer = 1;
```

```
        for (int k=1;k<i;k++)
        {
              A[1+(i-k)] = A [(i-k)];
        }
        i++;
        A[1] = buffer;
}
for (i=1;i<=5;i++)
{ A[i+26]=B[i];
}
```

Dari deret yang berhasil dibangkitkan diatas maka tinggal memakainya untuk masukan Motor DC dengan memberi nilai tegangan *input Robust* bernilai 0 sebesar tegangan *input* untuk Motor DC dapat bergerak untuk bit yang bernilai 0 dan tegangan *input Robust* bernilai 1, sebesar tegangan *input* untuk Motor DC bergerak secara bagus untuk bit yang bernilai 1. kemudian di ambil data tegangan *output*-nya untuk disimpan ke sebuah *file*.

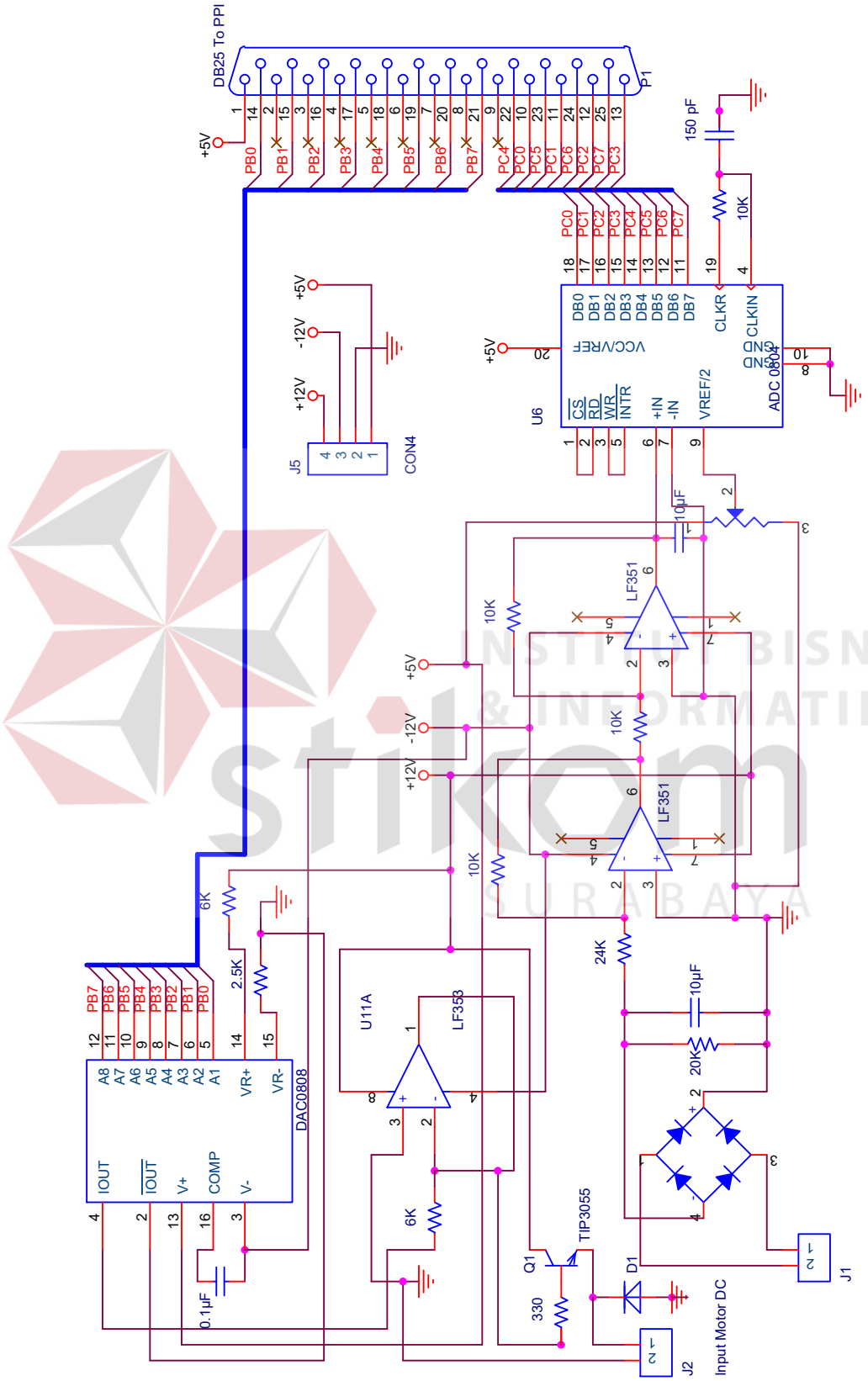
File yang di dapat tersebut diolah dengan program MATLAB untuk dianalisa sampai diketemukan fungsi alihnya. Programnya dalam MATLAB sebagai berikut :

```
IN=input('inputkan harga n :');
inp=dinamis(1:105,2);
outp=dinamis(1:105,3);
nn=[IN 1 1];
z=[outp inp];
th=arx(z,nn);
[numd,dend]=th2tf(th);
sys=tf(numd,dend,0.2);
[num,den]=d2cm(numd,dend,0.2,'zoh');
sys=tf(num,den)
```


3.4 Perancangan Dan Pembuatan Perangkat Keras

Perangkat keras yang dipakai untuk tugas akhir ini sudah umum digunakan dalam praktikum dan perkuliahan. Peralatan yang dipakai meliputi PPI, DAC, ADC, *Driver* Motor DC, Rangkaian penyearah dan Rangkaian pelemah.

Disini akan diuraikan secara jelas tentang rangkaian perangkat keras, yang digunakan untuk alat yang dibuat dalam tugas akhir ini. Disertakan juga gambar rangkaian dari tiap rangkaian serta *foto* dari modul rangkaian yang telah dibuat.



Gambar 3.4 Rangkaian perangkat keras

3.4.1 PPI

PPI di konfigurasikan dengan memberikan nilai CW sebesar 89H, yang artinya *Port* C difungsikan untuk ambil data dari luar PPI, *Port* B dan *Port* A untuk kirim data keluar dari PPI, serta beroperasi pada mode 0. PPI bekerja pada alamat 3E0H untuk *Port* A, 3E1H untuk *Port* B, 3E2H untuk *Port* C serta 3E3H untuk *Control Word*. IC PPI yang digunakan adalah produksi dari NEC JAPAN.

Tabel 3.2 Konfigurasi *Control Word* PPI 8255

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	1	0	0	1
8H				9H			

Dari tabel diatas dapat dijelaskan sebagai berikut :

D7 : Untuk mengaktifkan PPI.

D6 – D5 : Untuk memilih mode yang digunakan 0 – 2.

D4 : *Port* A untuk mengirimkan data.

D3 : *Port* C *upper* mengambil data.

D2 : Untuk memilih mode yang digunakan 0 - 1

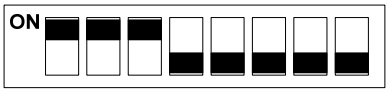
D1 : *Port* B untuk mengirimkan data.

D0 : *Port* C *lower* mengambil data.

Selain s*etting* alamat pada perangkat lunak, *setting* alamat juga diberlakukan pada perangkat keras PPI yang tertanam pada slot ISA komputer, karena untuk mencocokkan alamat *address bus* di komputer dengan alamat PPI, untuk *setting* alamat PPI produksi dari EL-TECH digunakan *DIP Switch* 8 saklar Aktif *Low* yang mewakili alamat 3E0H, konfigurasinya sebagai berikut :

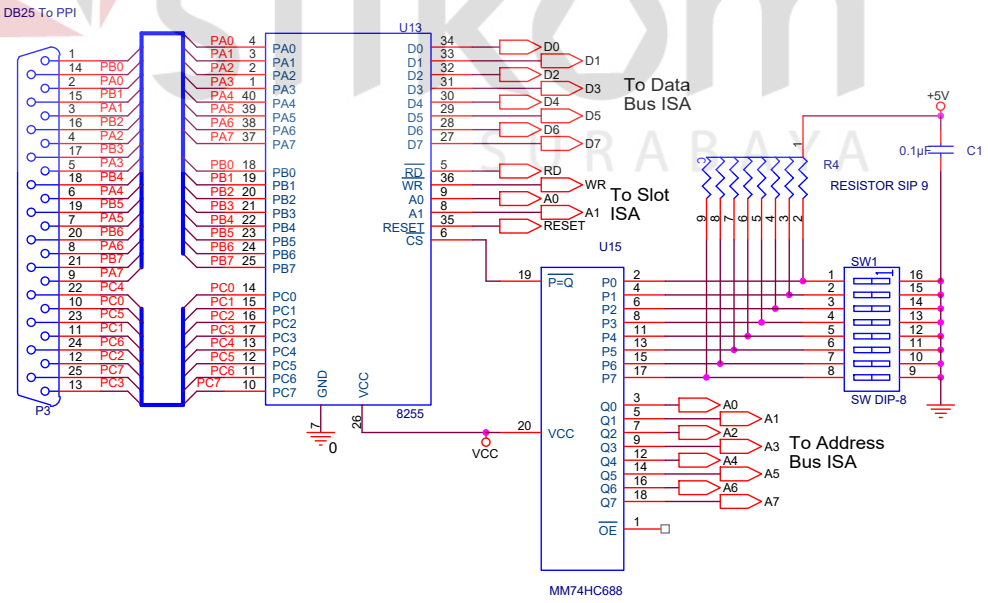
Tabel 3.3 Konfigurasi *address bus* PPI

A01	A02	A03	A04	A05	A06	A07	A08	A09	A10	A11	A12
X	X	0	0	0	1	1	1	1	1	X	x
0				E				3			



Gambar 3.5 Konfigurasi *DIP Switch* PPI untuk 3E0H

PPI disini berfungsi sebagai perantara untuk mengirimkan data dari komputer untuk *input* Motor DC melalui *Port* B dan menerima data dari *Tachogenerator* melewati *Port* C. Rangkaian PPI yang digunakan sudah terangkai dari pembuatnya dan diproduksi oleh EL-TECH.



Gambar 3.6 Rangkaian PPI

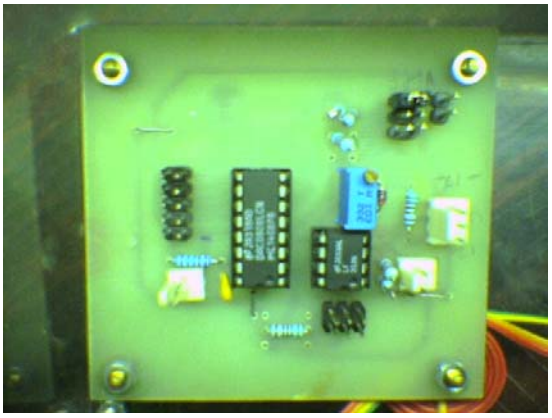
3.4.2 DAC

Dalam tugas akhir ini DAC 0808 berfungsi sebagai media untuk merubah sinyal digital dari PPI menjadi sinyal analog yang akan diteruskan oleh *Driver* Motor DC, sinyal tersebut merupakan tegangan yang besarnya antara 5 V sampai 12V, yang akan digunakan sebagai tegangan *input* Motor DC.

DAC 0808 yang digunakan diproduksi oleh National Semiconductor, yang mempunyai perubahan arus tiap ± 1 LSB. Untuk rangkaiannya dapat dilihat pada Gambar 3.4.

Dari Gambar 3.4 dapat dijelaskan sebagai berikut, kaki nomor 1 *no connect* tidak dihubungkan, kaki nomor 2 sebagai *ground* serta juga sebagai kaki *output* kutup negatif, antara kaki nomor 3 dan kaki nomor 16 diberikan Kapasitor kemudian dihubungkan dengan Vee -12V untuk mendapatkan *clock* yang dibutuhkan oleh DAC, kaki nomor 4 sebagai *output* dari DAC dihubungkan dengan Op-Amp untuk merubah arus menjadi tegangan, kaki nomor 5 sampai nomor 12 berfungsi sebagai *input* data digital DAC yang dihubungkan ke PPI 8255 melewati *connector* DB25, kaki nomor 13 merupakan Vcc yang dihubungkan dengan tegangan 5V, kaki nomor 14 sebagai Vref dihubungkan dengan tegangan +12V melewati sebuah R yang diperhitungkan untuk mendapatkan Iref sebesar 2mA.

Untuk rangkaian DAC 0808 sangatlah peka terhadap Vcc dan Vee, untuk itu diusahakan memberikan Vcc yang berimbang, seperti jika Vcc 11,98V maka Vee haruslah -11,98V dengan toleransi tidak boleh lebih atau kurang 1 Volt.



Gambar 3.7 Modul Rangkaian DAC

3.4.3 ADC

ADC yang digunakan adalah ADC 0804 dari National Semiconductor yang memiliki pendekatan berturut – turut (*Succesive Aproximation*) untuk mengkonversi masukan analog dari 0V sampai 5V menjadi data digital 8 bit.. Pada rangkaian ini ADC di set *free running*, artinya pengoperasian tidak memerlukan sinyal kontrol seperti *start converting* (SC) ataupun *end of converting* (EOC). Kaki Vref diberi tegangan 5V sama dengan sumber 5V sehingga akan memiliki total *error* ± 1LSB, dengan resolusi sebesar $\frac{1}{255}$ x 5V = 19,53125 mV yang menyebabkan perubahan 1 bit pada keluaran ADC. Dikarenakan keluaran dari ADC adalah biner 8 bit dengan desimal 0 – 255 maka konversi dari keluaran ADC ke tegangan adalah :

$$\text{Vout} = \text{desimal x } \frac{Vref}{255} \tag{3.1}$$

ADC ini memiliki *internal clock* yang harus diaktifkan dengan menghubungkan sebuah tahanan *eksternal* (R) antara pin CLK OUT dan CLK IN

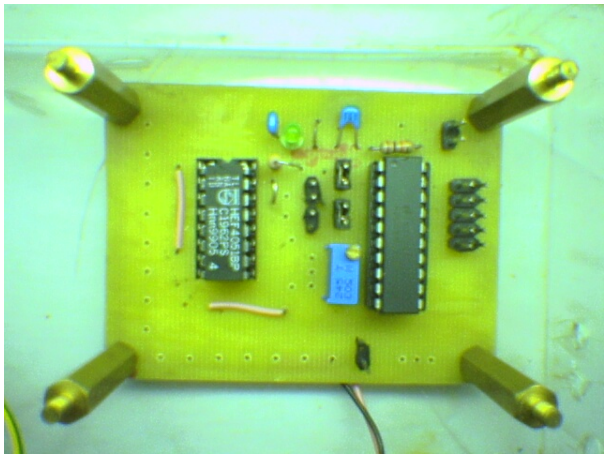
serta sebuah Kapasitor *eksternal* (C) antara CLK IN dan *ground digital*. Maka frekuensi *clock* yang digunakan adalah :

$$f = \frac{1}{1.1xRxC} \tag{3.2}$$

Untuk memperoleh *clock* sebesar 600KHz. ditentukan dulu C yang di pakai, disini digunakan C dengan nilai 150pF maka sesuai rumus diatas nilai R yang dibutuhkan adalah 10KΩ:

$$R = \frac{1}{1.1x600KHzx150\,pF} \tag{3.3}$$

Cara kerja ADC 0804 untuk satu kali proses konversi sebaga berikut, CS (*chip select*) terlebih dahulu harus diaktifkan, CS selalu aktif dengan menghubungkannya pada *ground*. *Start of Convertion* dilakukan dengan memberi logika *High – Low – High* pada pin WR dan RD harus tetap *High*. INTR secara langsung menuju ke logika *High* jika sebelumnya berlogika *Low*. Begitu data telah siap pada *output lacth*, INTR menuju ke logika *Low*, ini menjadi pertanda bahwa data telah siap.



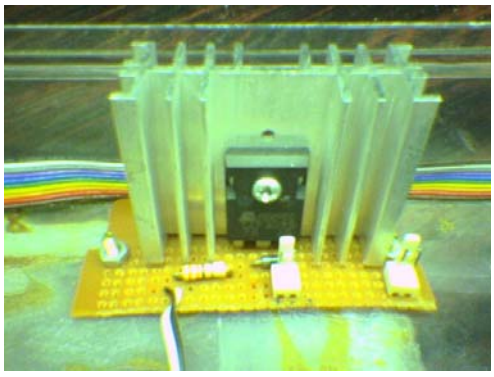
Gambar 3.8 Modul Rangkaian ADC.

Pengambilan data dilakukan pertama kali dengan mengaktifkan CS yang kemudian diikuti dengan memberi logika *High – Low* pada pin RD (*Read*). Tunggu sinyal INTR sampai *High*. Jika INTR sudah *High*, maka data telah siap pada pin – pin *output* digital dan data siap diambil. Setelah data diambil, sinyal RD diberi logika *High*, data kembali ke *high impedance.*

3.4.4 *Driver* Motor DC

Rangkaian *Driver* untuk Motor DC dalam tugas akhir ini sangatlah sederhana, karena hanya digunakan untuk memperkuat arus tegangan *input* untuk Motor DC, dengan mengabaikan arah putaran Motor, pengaturan kecepatan Motor, ataupun posisi putaran Motor. Untuk itu digunakan rangkaian sederhana dengan menggunakan sebuah TIP 3055 NPN Silicon Power Transistor, sebuah Resistor serta sebuah Dioda.

Dari Gambar 3.4 diatas, dapat dijelaskan prinsip kerja dari driver motor yang digunakan, pertama arus yang masuk ke basis akan diperoleh setelah melewati tahanan 330Ω, kemudian arus tersebut akan dikuatkan sebanyak 70 kali untuk maksimalnya, dan untuk minimalnya dikuatkan sebanyak 20 kali.



Gambar 3.9 Modul Rangkaian *Driver* Motor DC.

3.4.5 Rangkaian penyearah dan rangkaian pelemah

Rangkaian Penyearah dibutuhkan untuk merubah tegangan AC dari *output Tachogenerator* ke tegangan DC untuk dapat di proses oleh ADC, sedangkan rangkaian pelemah digunakan untuk melemahkan tegangan DC sebelum masuk ke ADC.

Rangkaian penyearah menggunakan rangkaian Dioda Jembatan yang komponennya adalah 4 buah Dioda dan ditambah sebuah Resistor 20KΩ serta sebuah Kapasitor 10µF.

Rangkaian Pelemah disini dibutuhkan karena Motor yang digunakan adalah Motor DC 12 Volt yang memungkinkan *output*-nya sebesar 12 Volt, untuk itu digunakanlah Rangkaian Pelemah sehingga data yang masuk ke ADC nantinya dapat dilemahkan hingga nilai maksimal datanya adalah 5 Volt.

Rangkaian pelemah menggunakan 2 buah Op-Amp LF351 atau UA741 yang terkonfigurasi sama, tapi berbeda dalam R yang digunakannya. Disini digunakan rangkaian penguat inverting, penguatan disini nantinya dibalik menjadi pelemahan, tapi pelemahan tersebut masih bernilai negatif, untuk itu digunakan 1 buah lagi Om-Amp dengan penguatan 1 kali sehingga nilai yang didapat bernilai positif.

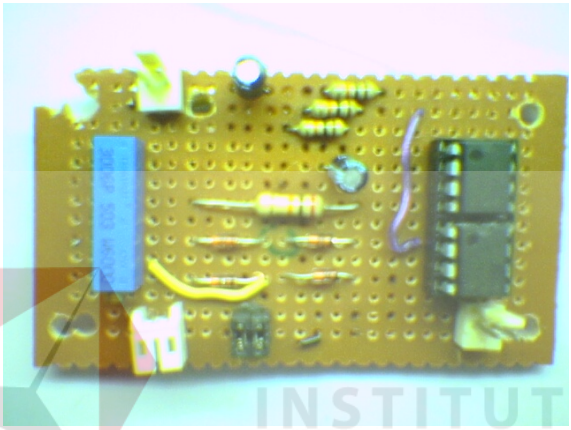
Untuk mendapatkan pelemahan dari 12 V menjadi 5V digunakan rumus dari penguatan *inverting* ini yaitu:

$$Vout = -\frac{R2}{R1}Vin \qquad (3.4)$$

Dalam rangkaian pelemah ini R2 bernilai 10KΩ, untuk mendapatkan pelemahan dari 12V menjadi 5V, sesuai rumus diatas maka R1 yang dibutuhkan adalah 24KΩ :

$$R1 = -\frac{Vin}{Vout} R2 \tag{3.5}$$

Setelah selesai masuk rangkaian pertama yang nilai keluarannya masih bernilai negatif kemudian haruslah dilemahkan lagi sebesar 1 kali sehingga nilai keluaran dari rangkaian pelemah ini bernilai positif.



Gambar 3.10 Modul Rangkaian Penyearah dan Rangkain Pelemah.