Anexo 3. Software usado para la caracterización de los motores

1. lectura de datos para la prueba estática

```
program leedatos.pas;

uses crt;
{$I c:\ernest\pascal\analog\analog.bib}

const
    digioutaddrlow : word =$220+13;
    digioutaddrhigh: word =$220+14;

var
    lastk,k,kk,j,l,ll,jj     : integer;
    portlow   : word;
    porthigh  : word;
    nom_fitxer: string;
    nom_aux   : string;
    fitx      : text;
    aa,bb,f     : real;
    size      : word;
    espai     : longint;
    lect      : word;



procedure inicialize;
begin
clrscr;
{port[digioutaddrlow]:=0;
portlow:=0;
port[digioutaddrhigh]:=0;
porthigh:=0;}

end;



begin
 clrscr;
 inicialize;
 writeln('escriu el nom del fitxer sense extensi¢.');
 writeln(' Es genera un fitxer extensi¢   . dat a c:\pepe');
 readln(nom_fitxer);
 nom_aux:= 'c:\pepe\'+nom_fitxer+'.dat';
 assign(fitx,nom_aux);

 rewrite(fitx);
 nosound;
 kk:=0;
 repeat
 kk:=kk+1;
 aa:=leevolt(0,100);
 bb:=leevolt(1,100);
 writeln(aa,'  ',bb);
 writeln(fitx,aa,'  ',bb);
 delay(100);
 gotoxy(10,10);
 writeln('mesura numero:',kk);
```

```
 until keypressed;
 close(fitx);
 end.
```

2. programa para la excitación de la etapa de potencia y lectura del par , la posición i la velocidad del motor en la prueba dinàmica

program trifasic.pas;

```
uses crt, dos;
{$I c:\ernest\pascal\analog\analog.bib}

type
    binari = string;

const
    digioutaddrlow : word =$220+13;
    digioutaddrhigh: word =$220+14;
    modecountaddr  : word =$220+3 ;
    countreadaddr  : word =$220 ;

var
    ona, estat     : array[1..12] of byte;
    pas            : byte;
    wait           : word;
    a              : real;

function pasabin(numero:byte):binari;
 var
   j  : byte;
   pasabi:binari;

 begin
 pasabi:='';
  for j:=1 to 8 do
   begin
     pasabi:=pasabi+char(48+numero mod 2);
     numero:=numero div 2;
   end;
  pasabin:=pasabi;

end;

procedure inicialize;
 var
   j:byte;
   ona1,ona2,ona3 : byte;

 begin
   clrscr;
   for j :=1 to 3 do ona[j]:=0;
   for j :=4 to 6 do ona[j]:=2;
   for j :=7 to 9 do ona[j]:=0;
   for j :=10 to 12 do ona[j]:=1;
   for j:=1 to 12 do
      begin
        ona1:=ona[j];
```

```
        ona2:=ona[(j+4)mod 12];
        ona3:=ona[(j+8)mod 12];
        estat[j]:=ona1 or 4*ona2 or 16*ona3 ;
      end;
   wait:=120;
   port[modecountaddr]:=$38;
   port[countreadaddr]:=$ff;
   port[countreadaddr]:=$ff;
   delay(20000);
end;

function readcount:word;
var
   prim, second :byte;

  begin
   prim:=port[countreadaddr];
   second:=port[countreadaddr];
   readcount:=prim+ 256*second;
  end;

procedure cicle(espera: word);
 var
   j        :byte;

 begin
  for j:= 1 to 12 do
    begin
       port[digioutaddrhigh]:=estat[j];
       {gotoxy(1,15);
       writeln(pasabin(estat[j]),'  ');
       a:=leevolt(0,100);
       gotoxy(1,17);
       write(a :4 :3);
       gotoxy(1,19);
       write(readcount :5);}
       delay(espera);
    end;
end;



begin
  inicialize;
  repeat
   cicle(wait);
  until keypressed;
  port[digioutaddrhigh]:=0;
end.
```

3. programa para el control de l motor paso a paso de frenado .

```
program frena;
uses crt,dos;
const    prt :word =$378;
var   j,k,d : integer;
procedure frenar(x:integer);

var:  integer;
begin
for n:=0 to x do
     begin
     port[prt]:=3;
     delay(k);
     port[prt]:=9;
     delay(k);
     port[prt]:=12;
     delay(k);
     port[prt]:=6;
     end;
end;

procedure desfrenar(x:integer);
var
n:integer;
begin
for n:=0 to x do
     begin
     port[prt]:=3;
     delay(k);
     port[prt]:=12;
     delay(k);
     port[prt]:=6;
     delay(k);
     port[prt]:=9;
     end;
end;

begin
clrscr;

 k:=30;
     repeat
     gotoxy(20,20);
     writeln('desfrenar= ');
     readln(d);
     desfrenar(d);
     gotoxy(20,20);
     writeln('frenar= ');
     readln(d);
     frenar(d);
     until 2=4;
end.
```

4. movimiento axial del estator, y adquisición y almacenamiento de los datos de fuerzas de levitación.

Fuerzas.pas

```
Program forces;

Uses
  Crt,t8255,sensors,motors,grafics,control;

Type
  senyuelo = Integer;

Const
  tmp_const: Real = 210;        {210 pasos por segundo; retardo=1}
  sec: Real = 1.2;        {margen de securitad de puntos medidos}
  n_mot: Integer = 2;       {número de motores}
  n_finestres: Integer = 4;
  max_bucles_petits: Integer = 4;
Var
  Psi:                  Rsistema;
  retard:               g_int;
  maxims,minims:g_int;
  ms:                   g_mot;
  busca_fi:            g_bool;
  cnls:                 g_int;          {canales de los sensores}
  cont:                 Integer;
  process:              Boolean;
  relax:                Integer;
  N:                    Integer;
  nom_fitxer:           String;
  nom_aux:              String;
  fminx,fmaxx:          Integer;
  fminz,fmaxz:          Integer;
  espaiat,tram:         Integer;
  vuit:                 Tgraf;
  claus:                Array[1..2] of conj_claus;
  marge:                Array[1..4] of conj_limits;
  maxtmp:               Real;
  primer,ultim:         Integer;
  n_bucles_petits:      Integer;
  incd:                 Integer;        {incremento densidad puntos
bucles pequeños}
  interval:             Integer;        {relación puntos medida-pasos}
  int_aux:              Real;
  cicles:               Integer;
  desti:                LongInt;
  referencia:           Integer;
  fitxer:               Text;

Procedure prepara_fitxer(nom_fitxer:String;Var fitxer:Text);
  Var
    nom_aux:    String;
  Begin
    nom_aux:=Concat('c:\tp60\isabel\',nom_fitxer,'.m');
    Assign(fitxer,nom_aux);
    Rewrite(fitxer);
    nom_aux:=Concat('%',nom_fitxer,';');
```

```
    Writeln(fitxer,nom_aux);
    Close(fitxer);
  End;

Function cas_fc(Var mot:Rmotor):Boolean;
  Var
    resposta:   Char;
  Begin
    mot.retard:=mot.retard*proporcio;
    mot.maxpos:=mot.maxpos*proporcio;
    mot.minpos:=mot.minpos*proporcio;
    While (mot.mou_motor(DAVANT) AND (mot.pos<mot.maxpos) AND
(mot.estat=DAVANT)) Do;
    Writeln('FC Preparat. Continuem? (S/N)');
    Readln(resposta);
    If ((resposta<>'s') AND (resposta<>'S')) Then
      cas_fc:=False
    Else
      cas_fc:=True;
    mot.pos:=mot.pos div proporcio;
    mot.maxpos:=mot.maxpos div proporcio;
    mot.minpos:=mot.minpos div proporcio;
    mot.retard:=mot.retard div proporcio;
  End;

Begin
{  writeln(26946*14/60);}
  process:=True;        {True: ZFC; False:FC}
  relax:=0;             {minutos a esperar}
  N:=1;                   {ciclos de medida}
  nom_fitxer:='cal';
  fminx:=7000;        {fuerzas estimadas según x}
  fmaxx:=9000;
  fminz:=8400;       { fuerzas estimadas según z}
  fmaxz:=9000;

  espaiat:=2200;    { 2200 pasos=5 mm retardo=1)}
  tram:=2*625;      { 440 pasos=1mmz (retardo=1); 420 pasos=1mmx
(retardo=2)}
                    { 625 pasos=1mmx (retardo=3)}
  incd:=4;          {incremento densidad puntos bucles pequeños}

  vuit.init('c:\tp60\bgi',0);
  If vuit.acces Then
    Writeln('Acceso a opciones gráficas');

  retard[1]:=1;              retard[2]:=3;
  ms[1]:=MZ;                 ms[2]:=MX;
  busca_fi[1]:=false;         busca_fi[2]:=False;
  minims[1]:=0;              minims[2]:=-28716;
  maxims[1]:=37733;           maxims[2]:=21345;

  cnls[1]:=0;  cnls[2]:=8;
  If
Psi.crea_sistema(n_mot,retard,ms,busca_fi,maxims,minims,cnls,n_finestr
es) Then
    Writeln('Ok. Motor amb retard=',retard[1]);

  If N>2 Then
    If Not(process) Then
```

```
      int_aux:=sec*((2*N-1)*Psi.mot[1].maxpos+4*(N-
2)*int_aux)+relax*60*tmp_const
    Else
      int_aux:=sec*(2*N*Psi.mot[1].maxpos+4*(N-
2)*int_aux)+relax*60*tmp_const
  Else
    If Not(process) Then
      int_aux:=sec*((2*N-1)*Psi.mot[1].maxpos+4*(N-1)*int_aux)
    Else
      int_aux:=sec*(2*N*Psi.mot[1].maxpos+4*(N-1)*int_aux);
  maxtmp:=int_aux/tmp_const;

  {x's}              {min(x)}                   {max(x)}
  claus[1][1]:=1;  marge[1][1]:=0;
marge[3][1]:=Psi.mot[1].maxpos;
  claus[1][2]:=1;  marge[1][2]:=0;
marge[3][2]:=Psi.mot[1].maxpos;
  claus[1][3]:=3;  marge[1][3]:=0;          marge[3][3]:=maxtmp;
  claus[1][4]:=2;  marge[1][4]:=-1.1*tram;  marge[3][4]:=1.1*tram;

  {y's}              {min(y)}                   {max(y)}
  claus[2][1]:=5;  marge[2][1]:=fminx;  marge[4][1]:=fmaxx;
  claus[2][2]:=4;  marge[2][2]:=fminz;  marge[4][2]:=fmaxz;
  claus[2][3]:=4;  marge[2][3]:=fminz;  marge[4][3]:=fmaxz;
  claus[2][4]:=5;  marge[2][4]:=fminx;  marge[4][4]:=fmaxx;

  Psi.graf.asigna_indexs(claus[1],claus[2]);

  Psi.graf.fin[1].title('         Psi1');
  Psi.graf.fin[1].xnom('dist…ncia Z (pasos)');
  Psi.graf.fin[1].ynom('forza X (A/D)');
  Psi.graf.fin[2].title('         Psi2');
  Psi.graf.fin[2].xnom('dist…ncia Z (pasos)');
  Psi.graf.fin[2].ynom('forza Z (A/D)');
  If Process Then
    nom_aux:='Cas Zero Field Cooled'
  Else
    nom_aux:='Cas Field Cooled';
  Psi.graf.fin[3].title(nom_aux);
  Psi.graf.fin[3].xnom('temps (segons)');
  Psi.graf.fin[3].ynom('forza Z (A/D)');
  Psi.graf.fin[4].title('         Psi3');
  Psi.graf.fin[4].xnom('dist…ncia X (pasos)');
  Psi.graf.fin[4].ynom('forza X (A/D)');
  Psi.graf.asigna_limits(marge[1],marge[2],marge[3],marge[4]);

  prepara_fitxer(nom_fitxer,fitxer);

  If Not(process) Then
    If Not(cas_fc(Psi.mot[1])) Then
      Exit;

  primer:=1;

{*********************************}

  Psi.tmp.posa_zero;

  desti:=trunc(Psi.mot[1].maxpos/80)*77;
  interval:=2000;
  if Psi.cami(1,desti,true,primer,ultim,interval) then
```

```
  primer:=1;
  delay(3000);

  desti:=trunc(Psi.mot[1].maxpos/80)*6;
  interval:=5;
  if Psi.cami(1,desti,true,primer,ultim,interval) then
  primer:=ultim+1;

  desti:=trunc(Psi.mot[1].maxpos/80)*7;
  interval:=5;
  if Psi.cami(1,desti,true,primer,ultim,interval) then
  primer:=ultim+1;

 desti:=trunc(Psi.mot[1].maxpos/80)*67;
  interval:=6;
  if Psi.cami(1,desti,true,primer,ultim,interval) then
  primer:=ultim+1;

  desti:=trunc(Psi.mot[1].maxpos);
  interval:=5;
  if Psi.cami(1,desti,true,primer,ultim,interval) then
  primer:=ultim+1;

  desti:=trunc(Psi.mot[1].maxpos/80)*77;
  interval:=5;
  if Psi.cami(1,desti,true,primer,ultim,interval) then
  writeln('ultim=',ultim);

     Str(cicles,nom_aux);
     nom_aux:=Concat(nom_fitxer,nom_aux,'=[');
     escriu(fitxer,Psi.resultat,ultim,0,nom_aux);

     primer:=1;
     desti:=0;
     interval:=2000;
  if Psi.cami(1,desti,true,primer,ultim,interval) then
  primer:=1;

  For cont:=1 To n_motors Do
    Psi.mot[cont].tanca_port;

  Repeat Until KeyPressed;
  vuit.fi;
End.
```

5. software necesario para el funcionamiento de fuerzas.pas

Motores.pas

```
Unit motors;

Interface

Uses
  Crt,t8255;

Type
  tipus_motor = (MX,MY,MZ);
  cicle_motor = Array[1..8] of Byte;
  estat_motor = (DAVANT,ENRERA,DESACT);
  mascares = Array[1..2] of Byte;

{*********************************}
  Rmotor = Object(Tt8255)
             pos :              LongInt;
             maxpos,minpos :    LongInt;
             posicio:           Real;
             estat :            estat_motor;
             retard :           Integer;
             digfi :            Tt8255;
             fi :               mascares;

             Function fi_carrera:Byte;
             Function mou_motor(sentit:estat_motor):Boolean;
             Function ini_motor:Boolean;
             Function crea_motor(rtard:Integer;m:tipus_motor;
                                 opcio:Boolean;maxim,minim:LongInt):
Boolean;
             Function pas: LongInt;
             Function situacio: Real;
             Function max_pos: LongInt;
             Procedure vis_motor;
             Procedure mata_motor;
           End;

Const
 cicle : cicle_motor = (7,5,13,9,11,10,14,6);}
  cicle : cicle_motor = (6,14,10,11,9,13,5,7);
  pas_micra: Real = 2;                          {Conversión paso-
micra}

Implementation


Function Rmotor.fi_carrera:Byte;
Var
    fi_aux:Byte;
    aux:Byte;
  Begin
    fi_aux:=0;
    aux:=Port[digfi.base];
    If (Port[digfi.base] And fi[1])=0 Then
      fi_aux:=1;
    If (Port[digfi.base] And fi[2])=0 Then
      fi_carrera:=fi_aux+2
```

```
    Else
      fi_carrera:=fi_aux;
  End;

Function Rmotor.mou_motor(sentit:estat_motor):Boolean;
Var
    index_motor, index_motor_bis :Byte;
    aux:Byte;
  Begin
    mou_motor:=True;
    estat:=sentit;
    aux:=fi_carrera;
    Case aux of
      0: Begin
           Case sentit of
             DAVANT: pos:=pos+1;
             ENRERA: pos:=pos-1;
           End;
         End;
      1: Begin
           estat:=DAVANT;
           minpos:=pos;
           pos:=pos+1;
         End;
      2: Begin
           estat:=ENRERA;
           maxpos:=pos;
           pos:=pos-1;
         End;
    Else
      Begin
        Writeln('Error al motor. Problemas con los fines de
carrera.');
        estat:=DESACT;
        mou_motor:=False;
      End;
    End;
    If estat<>DESACT Then
      Begin
        aux:=round(pos/retard+0.01*abs(pos/retard));
        index_motor:=(aux And $7)+1;
        index_motor_bis:=((aux-2) And $7)+1;

        Port[base]:=cicle[index_motor]+cicle[index_motor_bis]*16;

{        posicio:=(2+((pos-1) div retard))/pas_micra;
        index_motor:=(1+((pos-1) div retard) And $7)+1;
        index_motor_bis:=(1+(((pos-1) div retard)-2) And $7)+1;
        Port[base]:=cicle[index_motor]+cicle[index_motor_bis]*16;}
      End;
  End;

Function Rmotor.ini_motor:Boolean;
Var
    sort : Boolean;
    i : Word;
    c: Char;
  Begin
    ini_motor:=True;
    sort:=False;
    estat:=DAVANT;
```

```
    pos:=0;
    While ((estat=DAVANT) And Not(sort)) Do
      sort:=Not(mou_motor(DAVANT));
    If sort Then
      Begin
        Writeln('Error al motor. Problemas con el segundo fin de
carrera.');
        ini_motor:=False;
      End
    Else
      Begin
        Writeln('Detectado primer fin de carrera. Ok. Pulsa una
tecla');
        Read(c);
        maxpos:=pos;
        While ((estat=ENRERA) And Not(sort)) Do
          sort:=Not(mou_motor(ENRERA));
        If sort Then
          Begin
            Writeln('Error al motor. Problemas con el segundo fin de
carrera.');
            ini_motor:=False;
          End
        Else
          minpos:=pos;
      End;
  End;

Function Rmotor.crea_motor(rtard:Integer;m:tipus_motor;
                           opcio:Boolean;maxim,minim:LongInt):
Boolean;
Begin
    crea_motor:=True;
    retard:=rtard;
    Case m of
      MX: Begin
            base:=$1b5;
            fi[1]:=4;
            fi[2]:=2;
          End;
      MY: Begin
            base:=$1b4;
            fi[1]:=4;
            fi[2]:=2;
          End;
      MZ: Begin
            base:=$1b4;
            fi[1]:=8;
            fi[2]:=1;
          End;
    End;
    digfi.base:=$1b0;
    digfi.ini_port;
    If opcio Then
      crea_motor:=ini_motor
    Else
      Begin
        maxpos:=maxim;
        minpos:=minim;
        pos:=0;
      End;
```

```
  End;

Function Rmotor.pas: LongInt;
Begin
    pas:=pos;
  End;

Function Rmotor.situacio: Real;
Begin
    situacio:=posicio;
  End;

Function Rmotor.max_pos: LongInt;
Begin
    max_pos:=maxpos;
  End;

Procedure Rmotor.vis_motor;
Begin
    Write('Pos= ',pos,' MaxPos= ',maxpos);
  End;

Procedure Rmotor.mata_motor;
Begin
    tanca_port;
    digfi.tanca_port;
  End;

End.
```

---

6. programa para la lectura de datos de los sensores

sensores.pas

```
Unit sensors;
Interface
Uses
  Crt, basics;

Type
Rsensor=Object(Tbase)
      analog : Word;
      canal: Byte;
      valor,zero: Integer;

      Function leevolt(promig:Integer):Integer;
      Procedure ini_sensor;
      Procedure crea_sensor(ab:Word;cnl:Byte);
    End;

Implementation

Function Rsensor.leevolt(promig:Integer):Integer;
Const
    espera : Integer = 1;
  Var
    i,k : Integer;
    j,Hsal,Lsal : Byte;
    parcial : Real;
    q : Byte;
  Begin
    parcial:=0;
    For i:=1 To promig Do
      Begin
        Port[analog+1]:=0;
        Port[analog]:=canal;
        For j:=1 to 8 Do
          Hsal:=Port[analog+8];
        For j:=1 to 8 Do
          Lsal:=Port[analog+12];
        Lsal:=Port[analog+2];
        Hsal:=Port[analog+3] And $3F;
        parcial:=parcial+Lsal+256*Hsal;
        for q:=1 to espera Do;
      End;
    valor:=Trunc(parcial/promig);{-zero;}
    leevolt:=valor;
  End;

Procedure Rsensor.ini_sensor;
Begin
    zero:=0;
    zero:=leevolt(200);
  End;

Procedure Rsensor.crea_sensor(ab:Word;cnl:Byte);
Begin
    analog:=ab;
    canal:=cnl;
    ini_sensor;
  End;
```

```
End.
```

---

7. programa para la representación de gráficos en la pantalla

Graficos.pas

```
Unit grafics;

Interface

Uses
  Crt,basics,Graph;

Type

  conj_claus= Array[1..4] of Integer;
  conj_limits= Array[1..4] of Real;

{**********************************}
  Tgraf=Object(Tbase)
      Mx,My: Integer;
      PathToDriver:  String;
      GraphMode:     Integer;
      Constructor init(S:string;I:Integer);
      Function acces: Boolean;
      Procedure fi;
    End;

{**********************************}
  Rseccio=Object(Tbase)
      buffer: Pointer;
      tamany: Word;
      limits: Array[1..4] of Integer;
      Constructor init(x1,y1,x2,y2:Integer);
      Destructor fi;
      Procedure agafa;
      Procedure posa;
    End;

{**********************************}
  Rpantalla=Object(Tbase)
      bloc: Array[1..4] of Rseccio;
      Constructor init;
      Destructor fi;
    End;

{**********************************}
  Rgrafica=Object(Tbase)
      int_limits: Array[1..4] Of Integer;
      real_limits: Array[1..4] Of Real;
      Procedure init(x1,y1,x2,y2:Integer);
      Procedure canvi_escala(x1,y1,x2,y2:Real);
      Function obte_punt(x,y:Real;Var xp,yp:Integer):Boolean;
      Procedure dibuixa_punt(x,y:Real);
    End;

{**********************************}
  Reix=Object(Rgrafica)
      incx,incy: Real;
```

```
      grid:      Boolean;
      t,x,y:     String;
      Procedure activa;
      Procedure desactiva;
      Procedure canvi_escala(x1,y1,x2,y2:Real);
      Procedure nou(clau:Integer);
      Procedure xarxa(g:Boolean);
      Procedure title(nom:String);
      Procedure xnom(xtitle:String);
      Procedure ynom(ytitle:String);
    End;

{**********************************}
  Rfinestra=Object(Tbase)
      n_finestres :      Integer;
      fin :              Array[1..4] of Reix;   {n_ventanas=4}
      cx,cy :            conj_claus;

      Procedure defineix_finestres(claus:conj_claus);
      Procedure asigna_indexs(claux,clauy:conj_claus);
      Procedure asigna_limits(minx,miny,maxx,maxy:conj_limits);
    End;

Implementation
Constructor Tgraf.init(S:String;I:Integer);
Begin
    PathToDriver:=S;
    GraphMode:=I;
  End;

Function Tgraf.acces: Boolean;
Var
    GraphDriver:      Integer;
    ErrorCode:        Integer;
    LowMode:          Integer;
    HighMode:         Integer;

  Begin { acces }
    DirectVideo:=False;
    acces:=True;
    GraphDriver:=Detect;
    InitGraph(GraphDriver,GraphMode,PathToDriver);
    GetModeRange(GraphDriver,LowMode,HighMode);
    SetGraphMode(HighMode);
    ErrorCode:=GraphResult;
    ClrScr;
    If ErrorCode<>grOk Then
      Begin
        Writeln('Error en gr…fics: ',GraphErrorMsg(ErrorCode));
        Writeln('Sin acceso a las opciones gráficas');
        Delay(7500);
        acces:=False;
      End
    Else
      ClearDevice;
    Mx:=GetMaxX;
    My:=GetMaxY;
  End;  { acces }

Procedure Tgraf.fi;
  Begin
```

```
    CloseGraph;
  End;

Constructor Rseccio.init(x1,y1,x2,y2:Integer);
Begin
    limits[1]:=x1;
    limits[2]:=y1;
    limits[3]:=x2;
    limits[4]:=y2;
    tamany:=Imagesize(x1,y1,x2,y2);
    GetMem(buffer,tamany);
  End;

Destructor Rseccio.fi;
Begin
    FreeMem(buffer,tamany);
  End;

Procedure Rseccio.agafa;
Begin
    GetImage(limits[1],limits[2],limits[3],limits[4],buffer^);
  End;

{-------------------------------------------------------}
Procedure Rseccio.posa;
Begin
    PutImage(limits[1],limits[2],buffer^,NormalPut);
  End;

Constructor Rpantalla.init;
Begin
    bloc[1].init(0,0,GetMaxX div 2,GetMaxY div 2);
    bloc[2].init(GetMaxX div 2,0,GetMaxX,GetMaxY div 2);
    bloc[3].init(0,GetMaxY div 2,GetMaxX div 2,GetMaxY);
    bloc[4].init(GetMaxX div 2,GetMaxY div 2,GetMaxX,GetMaxY);
  End;

Destructor Rpantalla.fi;
Var
    i:Integer;
  Begin
    For i:=1 To 4 Do
      bloc[i].fi;
  End;

Procedure Rgrafica.init(x1,y1,x2,y2:Integer);
Begin
    int_limits[1]:=x1;
    int_limits[2]:=y1;
    int_limits[3]:=x2;
    int_limits[4]:=y2;
    real_limits[1]:=0.0;
    real_limits[2]:=0.0;
    real_limits[3]:=1.0;
    real_limits[4]:=1.0;
  End;

Procedure Rgrafica.canvi_escala(x1,y1,x2,y2:Real);
Begin
    real_limits[1]:=x1;
    real_limits[2]:=y1;
```

```
    real_limits[3]:=x2;
    real_limits[4]:=y2;
  End;

Function Rgrafica.obte_punt(x,y:Real;Var xp,yp:Integer):Boolean;
Begin
    obte_punt:=True;
    xp:=Round((x-real_limits[1])*(int_limits[3]-int_limits[1])
        /(real_limits[3]-real_limits[1])+int_limits[1]);
    yp:=Round((y-real_limits[2])*(int_limits[2]-int_limits[4])
        /(real_limits[4]-real_limits[2])+int_limits[4]);
    If (x<real_limits[1]) Or (x>real_limits[3]) Then
      obte_punt:=False;
    If (y<real_limits[2]) Or (y>real_limits[4]) Then
      obte_punt:=False;
  End;

Procedure Rgrafica.dibuixa_punt(x,y:Real);
Var
    auxx,auxy:Integer;
  Begin
    If obte_punt(x,y,auxx,auxy) Then
      PutPixel(auxx,auxy,LightGray);
  End;

Procedure Reix.activa;
Var
    p:     Real;
    xx,yy: Integer;
    s:     String;
  Begin
    SetColor(LightGray);
    SetTextStyle(TriplexFont,HorizDir,1);
    OutTextXY(int_limits[1]+15,int_limits[2]-30,t);
    SetTextStyle(SmallFont,HorizDir,4);
    OutTextXY(int_limits[3]-8*length(x),int_limits[4]+15,x);
    SetTextStyle(SmallFont,VertDir,4);
    OutTextXY(int_limits[1]-50,int_limits[2]+8*length(y),y);
    SetTextStyle(SmallFont,HorizDir,2);
    SetColor(DarkGray);

Rectangle(int_limits[1],int_limits[2],int_limits[3],int_limits[4]);
    SetColor(LightGray);
    p:=real_limits[1];
    While obte_punt(p,real_limits[2],xx,yy) Do
      Begin
        PutPixel(xx,yy+1,LightGray);
        Str(p:1:1,s);
        SetTextStyle(SmallFont,HorizDir,2);
        OutTextXY(xx-15,yy+3,s);
        If grid Then
          Begin
            SetColor(DarkGray);
            SetLineStyle(DottedLn,0,0);
            Line(xx,int_limits[4],xx,int_limits[2]);
            SetLineStyle(SolidLn,0,0);
            SetColor(LightGray);
          End;
        p:=p+incx;
      End;
    p:=real_limits[2];
```

```
    While obte_punt(real_limits[1],p,xx,yy) Do
      Begin
        PutPixel(xx-1,yy,LightGray);
        Str(p:1:1,s);
        SetTextStyle(SmallFont,HorizDir,2);
        OutTextXY(xx-30,yy-7,s);
        If grid Then
          Begin
            SetColor(DarkGray);
            SetLineStyle(DottedLn,0,0);
            Line(int_limits[1],yy,int_limits[3],yy);
            SetLineStyle(SolidLn,0,0);
            SetColor(LightGray);
          End;
        p:=p+incy;
      End;
    Line(int_limits[1]-5,int_limits[4],int_limits[3]+5,int_limits[4]);
    Line(int_limits[1],int_limits[4]+5,int_limits[1],int_limits[2]-5);
  End;

Procedure Reix.desactiva;
Var
    p:     Real;
    xx,yy: Integer;
    s:     String;
  Begin
    SetColor(Black);
    SetTextStyle(TriplexFont,HorizDir,1);
    OutTextXY(int_limits[1]+15,int_limits[2]-30,t);
    SetTextStyle(SmallFont,HorizDir,4);
    OutTextXY(int_limits[3]-8*length(x),int_limits[4]+15,x);
    SetTextStyle(SmallFont,VertDir,4);
    OutTextXY(int_limits[1]-50,int_limits[2]+8*length(y),y);
    SetTextStyle(SmallFont,HorizDir,3);

Rectangle(int_limits[1],int_limits[2],int_limits[3],int_limits[4]);
    p:=real_limits[1];
    While obte_punt(p,real_limits[2],xx,yy) Do
      Begin
        PutPixel(xx,yy+1,Black);
        Str(p:1:1,s);
        SetTextStyle(SmallFont,HorizDir,2);
        OutTextXY(xx-15,yy+3,s);
        If grid Then
          Begin
            SetLineStyle(DottedLn,0,0);
            Line(xx,int_limits[4],xx,int_limits[2]);
            SetLineStyle(SolidLn,0,0);
          End;
        p:=p+incx;
      End;
    p:=real_limits[2];
    While obte_punt(real_limits[1],p,xx,yy) Do
      Begin
        PutPixel(xx-1,yy,Black);
        Str(p:1:1,s);
        SetTextStyle(SmallFont,HorizDir,2);
        OutTextXY(xx-30,yy-7,s);
        If grid Then
          Begin
            SetLineStyle(DottedLn,0,0);
```

```
            Line(int_limits[1],yy,int_limits[3],yy);
            SetLineStyle(SolidLn,0,0);
          End;
        p:=p+incy;
      End;
    Line(int_limits[1]-5,int_limits[4],int_limits[3]+5,int_limits[4]);
    Line(int_limits[1],int_limits[4]+5,int_limits[1],int_limits[2]-5);
  End;

Procedure Reix.canvi_escala(x1,y1,x2,y2:Real);
Begin
    desactiva;
    real_limits[1]:=x1;
    real_limits[2]:=y1;
    real_limits[3]:=x2;
    real_limits[4]:=y2;
    incx:=(real_limits[3]-real_limits[1])/5;
    incy:=(real_limits[4]-real_limits[2])/5;
    activa;
  End;

Procedure Reix.nou(clau:Integer);
Const
    lc: Array[1..4] Of Integer=(10,45,60,95);
  Var
    aux: Array[1..4] Of Real;
  Begin
    Case clau Of
      111:Begin
            aux[1]:=lc[1];
            aux[2]:=lc[1];
            aux[3]:=lc[4];
            aux[4]:=lc[4];
          End;
      121:Begin
            aux[1]:=lc[1];
            aux[2]:=lc[1];
            aux[3]:=lc[2];
            aux[4]:=lc[4];
          End;
      122:Begin
            aux[1]:=lc[3];
            aux[2]:=lc[1];
            aux[3]:=lc[4];
            aux[4]:=lc[4];
          End;
      211:Begin
            aux[1]:=lc[1];
            aux[2]:=lc[1];
            aux[3]:=lc[4];
            aux[4]:=lc[2];
          End;
      212:Begin
            aux[1]:=lc[1];
            aux[2]:=lc[3];
            aux[3]:=lc[4];
            aux[4]:=lc[4];
          End;
      221:Begin
            aux[1]:=lc[1];
            aux[2]:=lc[1];
```

```
            aux[3]:=lc[2];
            aux[4]:=lc[2];
          End;
      222:Begin
            aux[1]:=lc[3];
            aux[2]:=lc[1];
            aux[3]:=lc[4];
            aux[4]:=lc[2];
          End;
      223:Begin
            aux[1]:=lc[1];
            aux[2]:=lc[3];
            aux[3]:=lc[2];
            aux[4]:=lc[4];
          End;
      224:Begin
            aux[1]:=lc[3];
            aux[2]:=lc[3];
            aux[3]:=lc[4];
            aux[4]:=lc[4];
          End;
      Else
          Begin
            aux[1]:=lc[1];
            aux[2]:=lc[1];
            aux[3]:=lc[4];
            aux[4]:=lc[4];
          End;
    End;

init(Trunc(aux[1]*GetMaxX/100),Trunc(aux[2]*GetMaxY/100),Trunc(aux[3]*
GetMaxX/100),Trunc(aux[4]*GetMaxY/100));
    incx:=(real_limits[3]-real_limits[1])/5;
    incy:=(real_limits[4]-real_limits[2])/5;
    grid:=False;
    t:='';
    x:='';
    y:='';
  End;


Procedure Reix.xarxa(g:Boolean);
Begin
    grid:=g;
  End;

Procedure Reix.title(nom:String);
Begin
    t:=nom;
  End;

Procedure Reix.xnom(xtitle:String);
Begin
    x:=xtitle;
  End;

Procedure Reix.ynom(ytitle:String);
Begin
    y:=ytitle;
  End;
```

```
Procedure Rfinestra.defineix_finestres(claus:conj_claus);
Var
    cont: Integer;
  Begin
    For cont:=1 To n_finestres Do
      Begin
        fin[cont].nou(claus[cont]);
        fin[cont].xarxa(True);
      End;
  End;

Procedure Rfinestra.asigna_indexs(claux,clauy:conj_claus);
Var
    cont: Integer;
  Begin
    For cont:=1 To n_finestres Do
      Begin
        cx[cont]:=claux[cont];
        cy[cont]:=clauy[cont];
      End;
  End;

Procedure Rfinestra.asigna_limits(minx,miny,maxx,maxy:conj_limits);
Var
    cont: Integer;
  Begin
    For cont:=1 To n_finestres Do

fin[cont].canvi_escala(minx[cont],miny[cont],maxx[cont],maxy[cont]);
  End;

End.
```

8. control del sistema

Control.pas

```
Unit control;

Interface

Uses
  Crt,basics,t8255,sensors,motors,grafics;

Type
  g_bool =       Array[1..2] of Boolean;        {n_motores =2}
  g_mot =        Array[1..2] of tipus_motor;    {n_motores =2}
  g_int =        Array[1..2] of LongInt;        {n_sensores=2}
  grup_sensors = Array[1..2] of Rsensor;        {n_sensores=2}
  grup_dades =   Array[1..5] of Real;
{x,z,tiempo,s1,s2=1+n_motores+n_sensores}
  dades =Array[1..1500] of grup_dades;  {limite=1500, cambiar abajo,
8000 / (1+ n_motores + n_sensores)}

{**********************************}
  Rdevice = Object(Tbase)
            mot :             Array[1..2] of Rmotor;    {n_motores =2}
            lectors :         grup_sensors;
            conjunt :         grup_dades;
            promig :          Integer;

            Function crea_device(n_mot:Integer;dretard:g_int;m:g_mot;

ini:g_bool;dmaxpos,dminpos:g_int;cnl:g_int):Boolean;
            Function
mou_i_llegeix(motor:Integer;sentit:estat_motor;pasos:Integer):Boolean;
          End;

{**********************************}
  Rsistema = Object(Rdevice)
             tmp :            Ttemps;
             graf :           Rfinestra;
             resultat :       dades;
             primer :         Integer;

             Function
crea_sistema(n_mot:Integer;dretard:g_int;m:g_mot;ini:g_bool;

dmaxpos,dminpos:g_int;cnl:g_int;n_fin:Integer):Boolean;
             Procedure dibuixa_fin(index:Integer);
             Function
cami(motor:Integer;final:LongInt;llegeix:Boolean;
                          Var
actual,ultim:Integer;pasos:Integer):Boolean;
             Procedure espera(minuts:Integer;Var actual: Integer);
          End;

Procedure escriu(Var fitxer:Text;Var
valors:dades;max,tipus:Integer;cadena:String);
Function existfitxer(fitxer : String): Boolean;
Procedure emergencia(recupera:dades;index:Integer);

Const
```

```
  proporcio:              Integer = 4;           {relación retardo
motor sin leer/motor leyendo}
  limit:                  Integer = 1500;
{8000/(1+n_motor+n_sensores)}
  n_motors:               Integer = 2;
  n_sensors:              Integer = 2;
  punts_per_minut:        Integer = 60;
  promig_ini:             Integer = 5;

Implementation

Function
Rdevice.crea_device(n_mot:Integer;dretard:g_int;m:g_mot;ini:g_bool;
                            dmaxpos,dminpos:g_int;cnl:g_int):Boolean;
Var
    cont: Integer;
  Begin
    crea_device:=True;
    n_motors:=n_mot;
    For cont:=1 To n_mot Do
      Begin
        If
Not(mot[cont].crea_motor(dretard[cont]*proporcio,m[cont],ini[cont],
               dmaxpos[cont]*proporcio,dminpos[cont]*proporcio)) Then
          crea_device:=False;
        mot[cont].maxpos:=mot[cont].maxpos div mot[cont].retard;
        mot[cont].minpos:=mot[cont].minpos div mot[cont].retard;
        mot[cont].pos:=mot[cont].pos div mot[cont].retard;
        mot[cont].retard:=dretard[cont];
        mot[cont].maxpos:=mot[cont].maxpos*mot[cont].retard;
        mot[cont].minpos:=mot[cont].minpos*mot[cont].retard;
        mot[cont].pos:=mot[cont].pos*mot[cont].retard;
      End;
    For cont:=1 To n_sensors Do
      lectors[cont].crea_sensor($170,cnl[cont]);
    promig:=promig_ini;
  End;

Function
Rdevice.mou_i_llegeix(motor:Integer;sentit:estat_motor;pasos:Integer):
Boolean;
Var
    aux: Array[1..10] of Real;
    cont,cont2,cont3: Integer;
  Begin
    For cont:=1 To n_sensors Do
      aux[cont]:=0;
    cont3:=0;
    For cont:=1 To pasos Do
      Begin
        mou_i_llegeix:=mot[motor].mou_motor(sentit);
        If sentit<>mot[motor].estat Then cont:=pasos;
        For cont2:=1 To n_sensors Do
          aux[cont2]:=aux[cont2]+lectors[cont2].leevolt(promig);
        Inc(cont3);
      End;
    For cont:=1 To n_sensors Do
      conjunt[1+n_motors+cont]:=aux[cont]/cont3;
  End;
```

```
Function
Rsistema.crea_sistema(n_mot:Integer;dretard:g_int;m:g_mot;ini:g_bool;

dmaxpos,dminpos:g_int;cnl:g_int;n_fin:Integer):Boolean;
Var
    claus: conj_claus;
    cont: Integer;
  Begin

crea_sistema:=crea_device(n_mot,dretard,m,ini,dmaxpos,dminpos,cnl);
    Case n_fin Of
       1:claus[1]:=111;
       2:Begin
           claus[1]:=211;
           claus[2]:=212;
         End;
       3:Begin
           claus[1]:=221;
           claus[2]:=222;
           claus[3]:=212;
         end;
       4:Begin
           claus[1]:=221;
           claus[2]:=222;
           claus[3]:=223;
           claus[4]:=224;
         End
    End;
    graf.n_finestres:=n_fin;
    graf.defineix_finestres(claus);
  End;

Procedure Rsistema.dibuixa_fin(index:integer);
Var
    cont,ix,iy: Integer;
    x,y: Real;
  Begin
    For cont:=1 To graf.n_finestres Do
      Begin
        ix:=graf.cx[cont];
        iy:=graf.cy[cont];
        x:=resultat[index][ix];
        y:=resultat[index][iy];
        graf.fin[cont].dibuixa_punt(x,y);
      End;
  End;

Function Rsistema.cami(motor:Integer;final:LongInt;llegeix:Boolean;Var
actual,ultim:Integer;pasos:Integer):Boolean;
Var
    index:   Integer;
    cont:    Integer;
    min_pas: Integer;
    sentit0: estat_motor;
    sort:    Boolean;
  Begin
    sort:=False;
    If final=mot[motor].pos Then
      sort:=True
    Else
      Begin
```

```
        If final<mot[motor].pos Then
          sentit0:=ENRERA
        Else
          sentit0:=DAVANT;
      End;
    mot[motor].estat:=sentit0;
    index:=actual;
    While ((mot[motor].estat=sentit0) And Not(sort)) Do
      Begin
        If llegeix Then
          Begin
            If abs(final-mot[motor].pos)<pasos Then
              min_pas:=abs(final-mot[motor].pos)
            Else
              min_pas:=pasos;
            cami:=mou_i_llegeix(motor,mot[motor].estat,min_pas);
            For cont:=1 To n_motors Do
              resultat[index][cont]:=mot[cont].pos;
            tmp.actualizar;
            resultat[index][1+n_motors]:=tmp.cputime/100;
            For cont:=1 To n_sensors Do

resultat[index][1+n_motors+cont]:=conjunt[1+n_motors+cont];
            dibuixa_fin(index);
            Inc(index);
            If index>limit Then
              Begin
                Writeln('Buffer excedit');
                emergencia(resultat,index);
                index:=1;
              End;
          End
        Else
          cami:=mot[motor].mou_motor(mot[motor].estat);
        If (sentit0=DAVANT) AND (mot[motor].pos>=final) Then
          sort:=True;
        If (sentit0=ENRERA) AND (mot[motor].pos<=final) Then
          sort:=True;
      End;
    ultim:=index-1;
  End;

Procedure Rsistema.espera(minuts:Integer;Var actual:Integer);
Var
    nsegons, conta: Integer;
  Begin
    For nsegons:=1 To minuts*60 Do
      Begin
        For conta:=1 To n_motors Do
          resultat[actual][conta]:=mot[conta].pos;
        tmp.actualizar;
        resultat[actual][1+n_motors]:=tmp.cputime/100;
        For conta:=1 To n_sensors Do

resultat[actual][1+n_motors+conta]:=lectors[conta].leevolt(promig);
        dibuixa_fin(actual);
        Delay(1000);
        Inc(actual);
      End;
  End;
```

```
{ ******************************** }
  Procedure escriu(Var fitxer:Text;Var
valors:dades;max,tipus:Integer;cadena:String);
  Var
    aux,cont,cont2:Integer;
  Begin
    Append(fitxer);
    If tipus=0 Then
      Writeln(fitxer,cadena);
    For cont:=1 To max-1 Do
      Begin
        For cont2:=1 To 1+n_motors+n_sensors Do
          Write(fitxer,valors[cont][cont2],' ');
        If ((tipus=0) And (cont=max-1))  Then
          Writeln(fitxer,'];')
        Else
          Writeln(fitxer,';');
      End;
    Close(fitxer);
  End;
{**** ***************************** }


{ Torna True si existe el fichero y False si hay error }
Function existfitxer(fitxer : String): Boolean;
{ función booleana que torna True si existe el fichero }
  Var
     f:File;
  Begin
     {$I-}
     Assign(f,fitxer);
     Reset(f);
     Close(f);
     {$I+}
     existfitxer := (IOResult=0) And (fitxer <> '');
  End;

Procedure emergencia(recupera:dades;index:Integer);
  Var
    fitxer_emergencia: Text;
  Begin
    Assign(fitxer_emergencia,'c:\tmp\tmp.m');
    If existfitxer('c:\tmp\tmp.m') Then
      Append(fitxer_emergencia)
    Else
      Rewrite(fitxer_emergencia);
    Writeln(fitxer_emergencia,'Buffer excedit');
    Close(fitxer_emergencia);
    escriu(fitxer_emergencia,recupera,index,0,'tmp=[');
  End;
End.
```

9. definición de los parámetros

Basics.pas

```
Unit basics;
Interface
Type
Tbase=Object
    End;
Implementation
End.
```

---

10. control de puertos de entrada-salida

T8255.pas

```
Unit t8255;

Interface

Uses
  Dos,basics;

Type
Ttemps=Object(Tbase)
      t_ini:LongInt;
      cputime:LongInt;
      Procedure posa_zero;
      Procedure actualizar;
    End;

Tt8255=Object(Tbase)
      base: Word;
      Procedure ini_port;
      Procedure tanca_port;
    End;

Implementation

Procedure Ttemps.posa_zero;
Var
    timecpu:DateTime;
    time_aux:Word;
  Begin
    GetTime(timecpu.Hour,timecpu.Min,timecpu.Sec,time_aux);

t_ini:=time_aux+100*(LongInt(timecpu.Sec+60*(timecpu.Min+60*timecpu.Ho
ur)));
  End;

Procedure Ttemps.actualizar;
Var
    timecpu:DateTime;
    time_aux:Word;
  Begin
    GetTime(timecpu.Hour,timecpu.Min,timecpu.Sec,time_aux);
```

```
cputime:=time_aux+100*(LongInt(timecpu.Sec+60*(timecpu.Min+60*timecpu.
Hour)))-t_ini;
  End;

Procedure Tt8255.ini_port;
Begin
    Port[base+11]:=$36;
    Port[base+11]:=$76;
    Port[base+11]:=$b6;
    Port[base+8]:=$2;
    Port[base+8]:=$0;
    Port[base+9]:=$32;
    Port[base+9]:=$0;
    Port[base+10]:=$64;
    Port[base+10]:=$0;
    Port[base+3]:=$80;
    Port[base+7]:=$80;
    Port[base]:=255;
  End;

Procedure Tt8255.tanca_port;
Begin
    Port[base]:=255;
  End;

End.
```

---

3. programa de lectura de datos para la caracterización estática

Dades.txt.

```
clear
a=path
path(a,'d:\usuarios\granados\joan')
load joan30_1.dat
load joan40_1.dat
load joan50_1.dat
load joan60_1.dat

f30= joan30_1;
f40= joan40_1;
f50= joan50_1;
f60= joan60_1;

joan30_1=[];
joan40_1=[];
joan50_1=[];
joan60_1=[];

esc=-1.1733;
```

```
f30(:,1)=(f30(:,1)+4.37)/0.0134;
f30(:,2)=f30(:,2)*esc;
f40(:,1)=(f40(:,1)+4.2)/0.0134;
f40(:,2)=f40(:,2)*esc;
f50(:,1)=(f50(:,1)+4.59)/0.0134;
f50(:,2)=f50(:,2)*esc;
f60(:,1)=(f60(:,1)+4.5)/0.0134;
f60(:,2)=f60(:,2)*esc;


save 'd:\usuarios\granados\joan\f30.dat' f30 -ascii
save 'd:\usuarios\granados\joan\f40.dat' f40 -ascii
save 'd:\usuarios\granados\joan\f50.dat' f50 -ascii
save 'd:\usuarios\granados\joan\f60.dat' f60 -ascii


[Y,I]=sort(f40);

for k=1 :500
40
k
[Y,I]=sort(Y);
Y(:,2)=f40(I(:,1),2);
n=size(Y);
for j=1 : n(1)-1
  if Y(j+1,1)==Y(j,1)
          Y(j+1,1)=Y(j+1,1)+0.0001;
  end
end
end
Xi=linspace(-20,338,180);
Yi=interp1(Y(:,1),Y(:,2),Xi);
F40=Yi;

[Y,I]=sort(f50);

for k=1 :500
50
k
[Y,I]=sort(Y);
Y(:,2)=f50(I(:,1),2);
n=size(Y);
for j=1 : n(1)-1
  if Y(j+1,1)==Y(j,1)
          Y(j+1,1)=Y(j+1,1)+0.0001;
  end
end
end
Xi=linspace(-20,338,180);
```

```
Yi=interp1(Y(:,1),Y(:,2),Xi);
F50=Yi;

[Y,I]=sort(f60);

for k=1 :500
60
k
[Y,I]=sort(Y);
Y(:,2)=f60(I(:,1),2);
n=size(Y);
for j=1 : n(1)-1
  if Y(j+1,1)==Y(j,1)
          Y(j+1,1)=Y(j+1,1)+0.0001;
  end
end
end
Xi=linspace(-20,338,180);
Yi=interp1(Y(:,1),Y(:,2),Xi);
F60=Yi;

cur=linspace(30,60,4)
surf(cur,Xi,mat)
shading interp
view(-85,20)

zlabel('torque (Nxm)')
title('torque vs current & slip angle')
xlabel('current (Amps)')
ylabel('angle (degrees)')
```