PC-BASED STEPPER MOTOR CONTROLLER

■ ASHUTOSH M. BHATT

This stepper motor controller is perhaps the cheapest, smallest and simplest. A pair of H-bridges with a software program written in 'C++' is used to control the bipolar stepper motor with a step resolution of 18 degree per pulse.

The controller is a combination of driver and switching circuits. The driver is the actual circuit that drives the stepper motor and the switching circuit decides how the motor should be driven. So, it is basically the switching circuit that controls the motor. The transistors (T1 through T8) act as switches. The switching of these transistors is controlled by the software via data pins D0 through D7.

You can control three parameters of the stepper motor: speed, direction and number of steps. To vary the speed of the motor, you have to vary the pulse repetition frequency (PRF). To change the direction of the motor, you have to change the sequence of pulses applied to its coils. By limiting the number of applied pulses, you can restrict the motor to complete the desired number of steps.

Specifications of the stepper motor

Stepper motors of various ratings/specifications are available in the market for different applications. Here, the stepper motor is taken from an 8.9cm (3.5-inch) floppy drive. It's a bipolar stepper motor rated at 5V DC with step resolution of 18° per pulse. The motor has two coils inside and four terminals (colour-coded, but not always) for external connections. Stepper motors rated at 5V and up to 1 ampere of current and different step size (e.g., 1.8° per pulse) may also be used with this circuit and control software.

Circuit description

H-Bridge driver. H-Bridge is a standard, well-known circuit widely used as stepper motor driver. It is a bridge connection of four transistors (see Fig. 1). Because there are two coils in the bipolar stepper motor, two H-bridge circuits, one for each coil, have been used. One H-bridge is formed by transistors T1 through T4 and the other bridge is formed by transistors T5 through T8.

Transistors T1 through T8 are BD139 type and should be used with heat-sinks. Pin details of BD139 and regulator IC 7805 are shown in Fig. 2. The bases of all the eight transistors are connected to data pins (D0 through D7) of the 25-pin, D-type male connector through 1-kilo-ohm current limiting resistors R1 through R8.

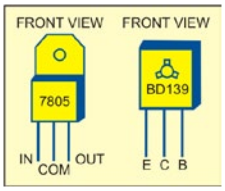


Fig. 2: Pin details of BD139 transistor and regulator IC 7805

The bases of transistors T1 and T4 are connected to parallel-port pins 2 (D0) and 3 (D1) through resistors R1 and R2, respectively, and the bases of transistors T2 and T3 are connected to parallel-port pins 4 (D2) and 5 (D3) through resistors R3 and R4, respectively. The red and orange terminals of the first coil (COIL1) are connected to the first H-bridge section as shown in Fig. 1.

The bases of transistors T5 and T8 are connected to pins 6 (D4) and 7 (D5) through resistors R5 and R6, respectively, and the bases of transistors T6 and T7 are connected to pins 8 (D6) and 9 (D7) through resistors R7 and R8, respectively. The yellow and green terminals of the second coil (COIL2) are connected to the second H-bridge section as shown in Fig. 1.

Power supply. The power supply section is shown in Fig. 3. It consists of a 230V AC to 9V AC, 1A secondary transformer (X1), filter, bridge rectifiers and 5V DC regulator 7805 (IC1). The regulated 5V DC is connected to the H-bridge circuits. The circuit ground is shorted to pins 18 through 25 of the D-type parallel-port connector. When switch S1 is closed, LED1 glows

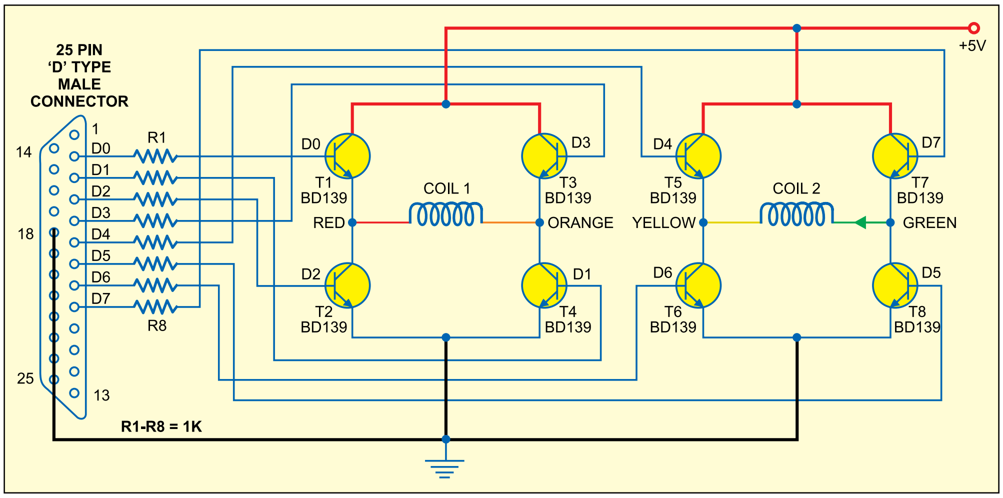


Fig. 1: Circuit of PC-based stepper motor controller

to indicate the presence of power in the circuit.

Operation

Specific sequence of pulses are given to the red and orange terminals of COIL1 and yellow and green terminals of COIL2 to rotate the motor either in clockwise or anticlockwise direction as explained in the following paragraph.

Direction control. In Tables I and II, '0' indicates low logic and '1' indicates high logic. We know that the current flows from high to low. Changing the direction of rotation is nothing but changing the direction of current that flows through the coils.

Speed control. To vary the speed, you have to vary the pulse repetition frequency (PRF). The PRF of 20 Hz means 20 pulses will be given to the stepper motor in one second. Since the step resolution of the motor is 18°/pulse, the motor will rotate 20x18° = 360° (one complete revolution) in one second. So the speed of the motor is one revolution per second (RPS) or 60 RPM. Now if you increase the PRF from 20 Hz to 40 Hz, the RPS will also double to 2 RPS (120 RPM).

Number of rotations. The step resolution of 18°/pulse means if you apply only one pulse, the motor will rotate by 18°. If you apply 10 pulses sequentially, the motor will rotate 180° (half of a revolution). So if you limit the number of pulses applied to the motor, you can stop it at any angular position (multiple of 18°) after completing the desired number of full revolutions. Thus if you apply only 25 pulses, the motor will complete one full revolution and rotate further by 90° (¼ revolution) and stop.

H-bridge. The transistors in the circuit act as switches. When high logic (3.49V) is applied to any data pin of the port, the transistor connected to it conducts and acts as a closed switch, allowing the current to pass through it. When low logic (0.09V) is applied, the transistor stops conducting and acts as an open switch, so the current cannot pass through it.

The pulse sequences to be given to switch the transistors are shown in tables. The current will flow into/out of the coils through the four terminals of the motor (red, orange, green and yellow).

Clockwise rotation. In the first phase, orange and green terminals should be high and red and yellow terminals should be low. To achieve this, out of the eight transistors, four transistors (T2, T3, T6 and T7) should conduct. For this, you have to output hex data word 'CC' (1100 1100) from the LPT port.

In the second phase, red and yellow terminals should be high, while orange and green terminals should be low. To achieve this, only transistors T2, T3, T5 and T8 should conduct. For this, you have to output hex data word '3C' (0011 1100) from the LPT port.

In the next phase, red and green terminals should be high, while orange and yellow terminals should be low. To achieve this, transistors T1, T4, T5 and T8 should conduct. For this, hex data word will be '33' (0011 0011).

In the next phase, red and yellow terminals should be low, while orange and green terminals should be high. To achieve this, transistors T1, T4, T6 and T7 should conduct. For the purpose, hex data word 'C3' (1100 0011) has to be output from the LPT port.

Thus the data sequence to be fed to the port for clockwise rotation of the motor is CC-3C-33-C3.

Anticlockwise rotation. To rotate the motor in anti-clockwise direction, the sequence of hex data to be output from the LPT port will be CC-C3-33-3C.

The software

All the controlling actions are performed by the software program. The program is written in 'C++' language and compiled in Turbo C++ Version 3. The complete software program (STEPCNT.CPP) is given at the end of this article along with necessary comments. You require the egavga.bgi graphic file to be in the same directory as the application program to run the program. The output of the program is shown in Fig. 4.

The main functions of the software are:

1. Change the direction of rotation of the stepper motor by switching the eight transistors in proper manner.

2. Vary the RPM of the stepper motor accurately.

3. Stop the motor at a given angular position after the desired number of complete rotations

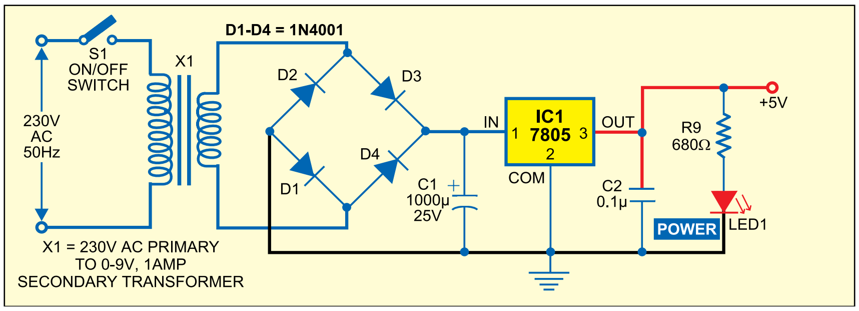


Fig. 3: Power supply for the circuit

TABLE I
Pulse Sequence to Rotate the Motor Clockwise

Data bit	D7	D6	D5	D4	D3	D2	D1	D0
Port pin	9	8	7	6	5	4	3	2
Phase 1	1	1	0	0	1	1	0	0
Phase 2	0	0	1	1	1	1	0	0
Phase 3	0	0	1	1	0	0	1	1
Phase 4	1	1	0	0	0	0	1	1

1= High, 0 = low

TABLE II
Pulse Sequence to Rotate the Motor Anti-clockwise

Data bit	D7	D6	D5	D4	D3	D2	D1	D0
Port pin	9	8	7	6	5	4	3	2
Phase 1	1	1	0	0	1	1	0	0
Phase 2	1	1	0	0	0	0	1	1
Phase 3	0	0	1	1	0	0	1	1
Phase 4	0	0	1	1	1	1	0	0

1= High, 0 = low

The software is divided into three parts: graphics, stepper motor control and mouse interfacing.

Graphics. The graphics part generates complete view of the control panel. It draws buttons like clockwise, anticlockwise and RPM increase/decrease, displays instructions, draws borderline, writes text like 'RPM,' 'rotations,' 'number of rotations,' etc. Graphic functions are used to make the program output screen visually appealing.

Stepper motor control. To change the direction of rotation of the motor, the program generates the desired pulse sequence, either CC-3C-33-C3 (to rotate the motor in clockwise direction) or CC-C3-33-3C (to rotate the motor in anticlockwise direction), on the parallel port with appropriate delay. The delay adjustment is done depending upon the RPM.

To vary the RPM, the program varies the PRF. First, the current RPM (S) is converted into RPS (S1) by dividing it by '60' as follows:

$$S1 = \frac{S}{60}$$

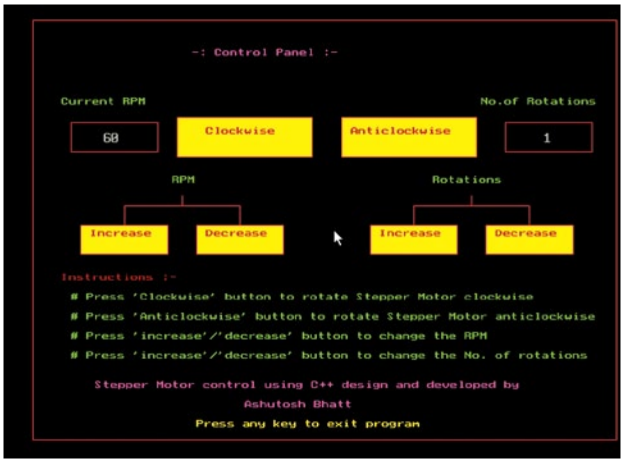
Now for one complete revolution, you have to apply 20 pulses. So the RPS factor (S1) multiplied by '20' will give you the desired PRF.

The delay (d) between the pulse sequences is given by:

$$\boxed{d = \frac{1000}{20 \times S1} = \frac{50}{S1} \text{ milliseconds}}$$

When RPM is greater than '10,' you can increase or decrease the RPM by a factor of ±10. For RPM less than '10,' you can increase or decrease it by ±1 only. There is no limit on the maximum RPM but the minimum limit is 1 RPM.

As stated earlier, 20 sequential pulses are required for a complete revolution of the stepper motor. Since a sequence of four pulses is repeated (for clockwise or anticlockwise movement), we may say that a revolution of the stepper motor involves five identical sequences of four pulses. You can increase or decrease the number of rotations linearly by ±1. For 'N' below '1,' you can decrease or increase 'N' by a factor of '0.5.' The minimum limit is 0.25 (quarter revolution), but there is no maximum limit.



```
# Press 'Clockwise' button to rotate Stepper Motor clockwise
# Press 'Anticlockwise' button to rotate Stepper Motor anticlockwise
# Press 'increase'/'decrease' button to change the RPM
# Press 'increase'/'decrease' button to change the No. of rotations

Stepper Motor control using C++ design and developed by
Ashutosh Bhatt
Press any key to exit program
```

Fig. 4: Screenshot of the program output

Mouse interfacing. This is the most interesting part of the program. It enables you to perform a task at a click of mouse. To understand how the mouse is interfaced, you have to go through the entire theory of hardware interfacing using 'C++.' Here, only some references have been made. For details, refer to the 'Mouse Interfacing' chapter of 'Let Us C' book by Kanitkar.

In this program, the functions that handle the mouse event are initmouse( ), resmptr(int p, int q, int r, int s), showmptr( ) and getmpos(int *t, int *u, int *v).

The initmouse( ) function loads mouse driver into the program. You've to pass '0' value through input union REGS to the int86( ) function. This function will return some non-zero value through output union REGS to the main program. If this function returns '0,' it means the mouse driver is not loaded. So the program displays the message "mouse driver is not loaded" and shuts the screen off using the exit( ) function.

The resmptr(int p, int q, int r, int s) function restricts mouse movement within the boundary specified by the four variables passed to it. Pass all these boundary limits through input union REGS to the int86( ) function. So the resmptr(int p, int q, int r, int s) function will restrict mouse movement out of this boundary.

The showmptr( ) function displays mouse pointer on the program screen. For this, you just have to pass value '1' through input union REGS to the int86( ) function. The showmptr( ) function will now show mouse pointer on the screen.

The getmpos(int *t, int *u, int *v) function performs two tasks: determines whether the mouse button is pressed or not, and captures the current mouse pointer position from the screen. You have to pass value '3' through input union REGS to the int86( ) function. This function will return 'x' and 'y' coordinates of the mouse pointer and also return value '1' if the mouse button (left) is pressed or '0' if the button is not pressed.

How the program works?

The program output screen includes the control panel for speed, direction and number of rotations of the stepper motor.

The program continuously checks for mouse-click event. Whenever there is a mouse click, the getmpos( ) function instantly captures 'x' and 'y' coordinates of the mouse pointer and passes them to the main program.

The main program decides on which position the click event has occured. If the click event occured on any button (clockwise, anti-clockwise, etc), it performs the desired task. For example, if you click the 'RPM increase' button, the program gets the coordinates and directly switches them to 'if' loop, increases the RPM and also displays it on the screen.

Construction and operation

Construct the hardware on a breadboard or on a PCB. An actual-size, single-side PCB layout is shown in

Fig. 5 and its component layout in Fig. 6. Connect the bases of the transistors to the respective data pins of the port DB25 (25-pin, D type male connector) as shown in Fig. 1. Insert DB25 into the PC's LPT-port female connector. Connect all the coil terminals (red, orange, green and yellow) to respective points as shown in the schematic.

Apply 5V DC supply to the circuit and connect the 5V stepper motor with its terminals as shown in Fig. 1. Now run the program on a computer powered by Windows 95/98 operating system. You will see the control panel on the computer screen. Switch on the 5V supply and LED1 will glow. Move the mouse pointer to any of the buttons as desired.

To rotate the motor clockwise, press and hold 'clockwise' button with left mouse button on the control panel. Similarly, for anticlockwise rotation, press and hold 'anticlockwise' button. The motor will rotate in the desired direction along with the beep sound as long as the button is kept pressed. When you release the button, the beep sound as well as the motor will stop.

If the motor rotates in anticlockwise direction when you press 'clockwise' button, just reverse any pair of terminals of the motor coils.

Run the stepcnt.exe file on your computer. You will see the control panel for the stepper motor controller on your screen. Default RPM and number of rotations are 60 RPM and 1, respectively. If you press 'clockwise' or 'anticlockwise' button, the motor will rotate until the desired rotations complete. You can increase or decrease the RPM or even the number of rotations by simply left-clicking that button once. Pressing these buttons more than once will increase/decrease the RPM/number of rotations by the same amount.

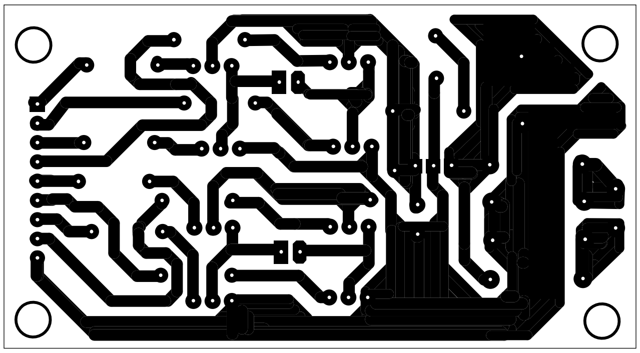


Fig. 5: Actual-size, single-side PCB layout for PC-based stepper motor controller

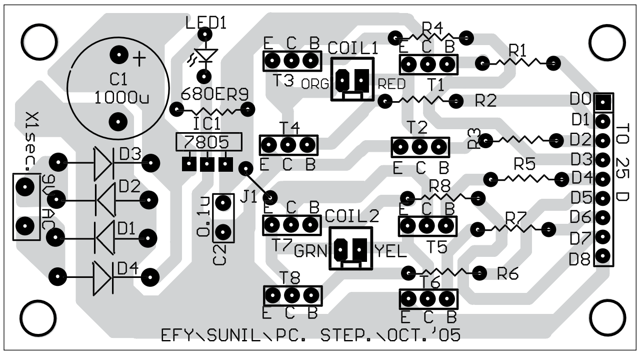


Fig. 6: Component layout for the PCB

The RPM and the number of rotations are perfectly calibrated for this particular stepper motor and you will get the accurate result for the RPM. That means if you choose 1 RPM, the motor will complete one revolution in one minute exactly. For the number of rotations also, if you choose one rotation, the motor will complete only one rotation. If you choose 0.25, the motor will complete only quarter revolution (90°).

Note. The stepper motor you choose should have the same specifications as given in this project. In the program, first enter RPM as '60' and then the number of rotations as '1.' Select clockwise or anticlockwise direction and then press 'Enter' key. The program will now display the PRF (=20 Hz) and the current RPM (=60) with a melodious sound output.

If the motor doesn't rotate, it means you have connected its four terminals wrongly. You can correct this using trial-and-error method. If the motor runs in a direction opposite to that you have selected, just reverse one pair of the coil terminals.

Because you have entered one rotation with 60 RPM, the motor will complete only one rotation in one second. If it doesn't, your stepper motor has some different specification. To check the specifications of the stepper motor, in the sample program select RPM as '5' and change the number of rotations (like 1.25, 1.5, etc) to get the step resolution. The delay factor (d) can be changed in the software. When you are satisfied with the result, switch to main program 'stepcnt.exe'

In case you wish to use the stepper motor with a step size of 1.8° per pulse, the software program for the same will differ. The same is included in the source code download folder.

Download source code: http://www.efymag.com/admin/issuepdf/Stepper%20Motor%20Control.zip

STEPCNT.CPP

```
#include<graphics.h>
#include<conio.h>
#include<dos.h>
#include<process.h>
#include<iostream.h>
union REGS i,o;
void main()
 {
   int driver,mode,x,y,but;      //intialitions of all vari-
ables and functions
   driver = DETECT;
   int initmouse();           // to load mouse driver
    int resmptr(int p,int q,int r,int s); //restric mouse
pointer within boundry
   int showmptr();                // shows mouse pointer
   int getmpos(int *t,int *u, int *v);  // captures the cur-
rent position of mouse pointer
    int text(int e,int f);  // changes the size and color
of text
    float s1,d=50,s=60;  // default RPM=60 and no. of
rotations = 1
   float r=1,n=5;
   initgraph(&driver, &mode, "C:\\tc\\bgi"); //initial-
ize graphics mode
   outport(0x0378,0x00);        // clear parallel port
   if(initmouse() == 0)
   {                       // load mouse driver if not
          closegraph();                        // exit the
program
          restorecrtmode();
```

```
            cout<<"\nMouse driver not loaded";
            exit(1);
    }
    gotoxy(14,10);
    cout<<s;                    // display current RPM
    gotoxy(71,10);          // and no. of rotations
    cout<<r;
    showmptr();
    resmptr(30,30,635,460);
    setcolor(LIGHTRED);
    rectangle(30,30,635,460);      //Border line
    rectangle(70,135,160,165);      //RPM Box
    rectangle(520,135,610,165);    //No.of Rotation box
    setfillstyle(SOLID_FILL,YELLOW);
    rectangle(180,130,320,170);
    floodfill(202,132,LIGHTRED);      //clockwise button
    rectangle(80,240,170,270);
    floodfill(82,242,LIGHTRED);      //RPM inc button
    rectangle(350,130,490,170);
      floodfill(352,132,LIGHTRED);       //anticlockwise
button
    rectangle(200,240,290,270);
    floodfill(202,242,LIGHTRED);     //RPM dec button
    rectangle(380,240,470,270);
    floodfill(382,242,LIGHTRED);     //rotation inc button
    rectangle(500,240,590,270);
    floodfill(502,242,LIGHTRED);    //rotation dec button
    line(125,220,245,220);
    line(425,220,545,220);
    line(185,220,185,210);
    line(485,220,485,210);
    line(245,220,245,240);
    line(545,220,545,240);
    line(125,220,125,240);
    line(425,220,425,240);
    text(8,13);
    outtextxy(195,60,"-: Control Panel :-");
    text(6,4);
    outtextxy(60,290,"Instructions :-");
    outtextxy(210,140,"Clockwise");
    outtextxy(360,140,"Anticlockwise");
    outtextxy(90,245,"Increase");
    outtextxy(210,245,"Decrease");
    outtextxy(390,245,"Increase");
    outtextxy(510,245,"Decrease");
    setcolor(10);
    outtextxy(175,190,"RPM");
    outtextxy(445,190,"Rotations");
    outtextxy(60,110,"Current RPM");
    outtextxy(495,110,"No.of Rotations");
    text(5,10);
    outtextxy(70,310,"# Press 'Clockwise' button to rotate
Stepper Motor clockwise");
      outtextxy(70,330,"# Press 'Anticlockwise' button to
rotate Stepper Motor anticlockwise");
    outtextxy(70,350,"# Press 'increase'/'decrease' button
to change the RPM");
    outtextxy(70,370,"# Press 'increase'/'decrease' button
to change the No. of rotations");
    setcolor(13);
      outtextxy(95,400,"Stepper Motor control using C++
design and developed by");
    outtextxy(250,420,"Ashutosh Bhatt");
    setcolor(YELLOW);
    outtextxy(200,440,"Press any key to exit program");
    while(!kbhit())         // loop until any key is pressed
    {
      getmpos(&but,&x,&y);                      // capture
current pointer position when click event happens
     if(x>=200 && x<=300 && y>=130 && y<=170 && (but
& 1) == 1)  // and switch to that if loop
                {
                  text(6,13);
                  outtextxy(210,140,"Clockwise");
                  for(int i=1;i<=n;i++)
                          {
                           sound(500);
                           outport(0x0378,0xcc);
                           delay(d);
                           outport(0x0378,0x3c);
                           delay(d);
                           outport(0x0378,0x33);
                           delay(d);
                           outport(0x0378,0xc3);
                           delay(d);
                           nosound();
                          }                //for loop ends
                  text(6,4);
                  outtextxy(210,140,"Clockwise");
                }                     // first if ends

      else if(x>=350 && x<=490 && y>=130 && y<=170
&& (but & 1) == 1)
                {
                  text(6,13);
                  outtextxy(360,140,"Anticlockwise");
                  for(int i=1;i<=n;i++)
                          {
                           sound(750);
                           outport(0x0378,0xcc);
                           delay(d);
                           outport(0x0378,0xc3);
                           delay(d);
                           outport(0x0378,0x33);
                           delay(d);
                           outport(0x0378,0x3c);
                           delay(d);
                           nosound();
                          }        // for loop ends
                          text(6,4);
                          outtextxy(360,140,"Anticlockw
ise");
                 }            //second if ends

      else if(x>=80 && x<=170 && y>=240 && y<=270 &&
(but & 1) == 1)
                 {
                  gotoxy(10,10);
                  cout<<"        ";
                  text(6,2);
                  outtextxy(90,245,"Increase");
                  sound(1000);
                  delay(200);
                  nosound();
                  if(s>=10) s=s+10;
                  else s++;                  // when this button
is pressed
                  s1 = s/60;                // increase current
RPM and also
                  d = 50/s1;                // change delay
                  gotoxy(14,10);
                  cout<<s;
                  text(6,4);
                  outtextxy(90,245,"Increase");
                }                // third if ends

                else if(x>=200 && x<=290 && y>=240 &&
y<=270 && (but & 1) == 1)
                 {
                  gotoxy(10,10);
                  cout<<"        ";
                  text(6,2);
                  outtextxy(210,245,"Decrease");
                  sound(1000);
                  delay(200);
                  nosound();
                  if(s>10)
                          {
                           s=s-10;                  // when
this button is pressed
                            gotoxy(14,10);                  //
decrease it till s>1
                            cout<<s;                      // if
s<=1 stop decreasing
                          }                              // and
display massage
                  else
                          {
                           if(s>1)
                             {
                              s--;
                              gotoxy(14,10);
                              cout<<s;
                             }
                           else
                             {
                              gotoxy(11,10);
                              cout<<"min limit";
                             }
                          }
                  s1 = s/60;
                  d = 50/s1;
                  text(6,4);
                  outtextxy(210,245,"Decrease");
                }                     // forth if ends

                else if(x>=380 && x<=470 && y>=240 &&
y<=270 && (but & 1) == 1)
                 {
                  gotoxy(67,10);
                  cout<<"        ";
                  text(6,2);
                  outtextxy(390,245,"Increase");
                  sound(1000);
                  delay(200);
                  nosound();
                  if(r<1) r=r*2;
// when this button is pressed
                  else r++;            // increase no. of rota-
tion
                  gotoxy(71,10);         // if rotations are <
1 then
                  cout<<r;                   // double it every
time
                  n=r*5;               // otherwise increase it
linearly
                  text(6,4);
                  outtextxy(390,245,"Increase");
                }              // fifth if ends

              else if(x>=500 && x<=590 && y>=240 &&
y<=270 && (but & 1) == 1)
                {
                  gotoxy(67,10);
                  cout<<"        ";
                  text(6,2);
                  outtextxy(510,245,"Decrease");
                  sound(1000);
                  delay(200);
                  nosound();
                  if(r>1)
                          {
                           r--;       // when this button
is pressed
              gotoxy(71,10);                // decrease No. of
rotations
                cout<<r;                  // till r=0.25 if r<0.25
stop decreasing
                          }    //and display a massage
                  else
                          {
                           if(r>0.25)
                             {
                                       r=r/2;
                                       gotoxy(71,10);
                                       cout<<r;
                             }
                           else
                             {
                                       gotoxy(67,10);

cout<<"Oopppps...";
                             }
                          }
                  n=r*5;
                  text(6,4);
                  outtextxy(510,245,"Decrease");
                }          // last if ends
    }      // while loop ends
  }        // main ends

  getmpos(int *but,int *x,int *y)
             {
               i.x.ax = 3;
               int86(0x33,&i,&o);
               *but = o.x.bx;
               *x = o.x.cx;
               *y = o.x.dx;
             }
   initmouse()
    {
              i.x.ax = 0;
              int86(0x33,&i,&o);
              return(o.x.ax);
             }
   showmptr()
             {
               i.x.ax = 1;
               int86(0x33,&i,&o);
             }
   resmptr(int a,int b,int c,int d)
             {
               i.x.ax = 7;
               i.x.cx = a;
               i.x.dx = c;
               int86(0x33,&i,&o);
               i.x.ax = 8;
               i.x.cx = b;
               i.x.dx = d;
               int86(0x33,&i,&o);
             }
             text(int e,int f)
             {
              setcolor(f);
              settextstyle(SMALL_FONT,HORIZ_DIR,e);
             }
```