```
#include<LedControl.h>
#include<math.h>
#include<LedControl.h>


class HarmonicOscillator //Harmonic Oscillator Class: Every SHO related
property and function is here
{ public:

float x; //variable x at any given time in a SHO
float y;  //variable dx/dt at any given time in a SHO
float b; //damping constant b
float w; //angular frequency w

HarmonicOscillator() //Constructor: creates a "Harmonic Oscillator Object with
all the properties and function list"
{
  x=0;
  y=0;
  b=0;
  w=0;
}
void change(float xin, float yin)//updates x and dx/dt at every instant of
time
{
  x=x+xin;
  y=y+yin;

}


float yder (float x, float y, float b, float w) //function dy/dt = f(x,y),
used in solving the differential equation to find find x and y=(dx/dt) at
every point of time (refer section: )
{
 return (-b*y-w*w*x) ;
}

float xder (float x, float y, float b, float w) //function dx/dt = g(x,y)=y,
used in solving the differential equation to find find x and y=(dx/dt) at
every point of time (refer section: )
{
 return (y) ;
}

float calculateOscillator(float h, float choice) //solving the values of x and
y=dx/dt at a given time, to plot it on the LEDMatrix (refer section: )
{
  float dx1= h*y;
```

```
  float dy1= h* yder(x,y,b,w);
  float dx2 = h*(y+(dy1/2));
  float dy2 = h*yder(x+(dx1/2), y+(dy1/2),b,w);
  float dx3 = h*(y+(dy2/2));
  float dy3 = h*yder(x+(dx2/2), y+(dy2/2),b,w);
  float dx4 = h*(y+(dy3));
  float dy4 = h*yder(x+(dx3), y+(dy3),b,w);

  float dx= ((dx1+2*dx2+2*dx3+dx4))/6.00;
  float dy= (dy1+2*dy2+2*dy3+dy4)/6.00;
   if(choice==1)
   return dx;
   else
   return dy;
}
float getAmplitude(float h)//finding the Amplitude of the oscillations, so
that we can appopriately scale the values to fit well in our LED grid
{

  float Amplitude=0;
    float xcopy=x;
    float ycopy=y;
    float wnew2 =(w*w - (b*b)/4.0);
        if(wnew2<=0)
          {
           return x;
           }
    float wnew = sqrt(wnew2);
    float Timeperiod= (2*3.1415926536)/wnew;


    for(float k=0; k<Timeperiod; k=k+h)
    {



    float addx = calculateOscillator(h,1);
    float addy = calculateOscillator(h,2);
    if(abs(x)>Amplitude)
    {
      Amplitude=x;

    }

    x=x+addx;
    y=y+addy;
    }
  x=xcopy;
```

```
  y=ycopy;
  return Amplitude;
}


};




class MasterSlave //Master Slvae Oscillator Class: Every SHO related property
and function is here
{public:
float mu;//parameter mu in VDP Mater Slave Oscillations (kept constant so that
oscillations don't become unstable due to incorrect user input)
float K;//parameter K in VDP Master Slave Oscillations
float xm;//Value of Masters x in VDP oscillations
float ym;//Value of Masters y in VDP oscillations
float xs;//Value of Slave x in VDP oscillations
float ys;//Value of Slave y in VDP oscillations


MasterSlave()//Constructor: creates a "Harmonic Oscillator Object with all the
properties and function list"
{
 xm=0;
 xs=0;
 ym=0;
 ys=0;
 mu=0;
 K=0;

}

//The four coupled differential equations required to obtain the solutions of
Master x,y and Slave x,y are obtained from the four given differential
equations (refer section: )
float MasterX ( float xm, float ym, float mu, float K)
{
  float r=mu*(xm - ym -(xm*xm*xm)/3);
 return r;
}

float MasterY ( float xm, float ym, float mu, float K)
{
  float r=xm/mu;
 return r;
}
```

```
float SlaveX ( float xm, float ym, float mu, float K, float xs, float ys)
{
  float r=mu*(xs - ys- (xs*xs*xs)/3)+ K*(xm-xs);
 return r;
}

float SlaveY ( float xs, float ys, float mu, float K)
{
  float r=xs/mu;
 return r;
}

//Calculates Master x,y and Slave x,y at every instant of time (refer section:
)
float RK4Calculator(float choice, float h)//The value of choice helps us
choose which one of the 4 calculated values we want to obtain 1:xm, 2 ym, 3
xs, 4 ys
{
  float dxm1=h*MasterX(xm, ym, mu, K);
  float dxs1=h*SlaveX(xm,ym,mu,K,xs,ys);
  float dym1 = h*MasterY(xm,ym,mu,K);
  float dys1= h*SlaveY(xs,ys,mu,K);
  float dxm2=h*MasterX(xm+(dxm1/2), ym+(dym1/2), mu, K);
  float dxs2=h*SlaveX(xm+(dxm1/2),ym+(dym1/2),mu,K,xs+(dxs1/2),ys+(dys1/2));
  float dym2 = h*MasterY(xm+(dxm1/2),ym+(dym1/2),mu,K);
  float dys2= h*SlaveY(xs+(dxs1/2),ys+(dys1/2),mu,K);
  float dxm3=h*MasterX(xm+(dxm2/2), ym+(dym2/2), mu, K);
  float dxs3=h*SlaveX(xm+(dxm2/2),ym+(dym2/2),mu,K,xs+(dxs2/2),ys+(dys2/2));
  float dym3 = h*MasterY(xm+(dxm2/2),ym+(dym2/2),mu,K);
  float dys3= h*SlaveY(xs+(dxs2/2),ys+(dys2/2),mu,K);
  float dxm4=h*MasterX(xm+(dxm3), ym+(dym3), mu, K);
  float dxs4=h*SlaveX(xm+(dxm3),ym+(dym3),mu,K,xs+(dxs3),ys+(dys3));
  float dym4 = h*MasterY(xm+(dxm3),ym+(dym3),mu,K);
  float dys4= h*SlaveY(xs+(dxs3),ys+(dys3),mu,K);

 float dxm = (1/6.00)*(dxm1+2*dxm2+2*dxm3+dxm4);

 float dxs = (1/6.00)*(dxs1+2*dxs2+2*dxs3+dxs4);

 float dym = (1/6.00)*(dym1+2*dym2+2*dym3+dym4);
 float dys = (1/6.00)*(dys1+2*dys2+2*dys3+dys4);

 if(choice==1)
 return dxm;

 else if( choice==2)
 return dym;
```

```
 else if(choice==3)
 return dxs;
 else
 return dys;

}

//updates all variables to their new values at every instant of time
void change(float xmin, float ymin, float xsin, float ysin)
{
  xm=xmin+xm;
  ym=ymin+ym;
  xs=xsin+xs;
  ys=ysin+ys;
}

float getAmplitude(float h)//Amplitude helps us ensure the display stays
within the bounds of the led matrix
{
  float xmcopy=xm;
  float ymcopy=ym;
  float xscopy=xs;
  float yscopy=ys;
 float Amplitude=0;

      for(int i=0;i<500;i++)
      {
      float addxm=RK4Calculator(1,h);
      float addym=RK4Calculator(2,h);
     float addxs=RK4Calculator(3,h);
      float addys=RK4Calculator(4,h);
      if(abs(xm)>Amplitude)
      {
        Amplitude=abs(xm);
      }

      xm=xm+addxm;
      ym=ym+addym;
      xs=xs+addxs;
      ys=ys+addys;

    }
     xm=xmcopy;
     ym=ymcopy;
     xs=xscopy;
     ys=yscopy;
    return Amplitude;
```

```
}
//end of Oscillator code

};




LedControl lc=LedControl(12,11,10,4); //object for LEDControl, needed for code systax

class LEDlight//Class to do various precise LED controls
{ public:
static void pinControl (int x, int y, bool state)//Allows us to map the matrix to a x-y coordinate system of our choice (O at bottom right, 8*32 system without device input requirment)
{
  int arr[5]={0,1,2,3,4};
  //Bottom left corner held horizontally is (0,0)
  int addr;
  int row=y;
  int column;

  if(x>=0 && x<8)
  {
    addr=0;
    column=7-x;
  }
  else if(x>=8 && x<16)
  {
    addr=1;
    column=15-x;
  }
  else if(x>=16 && x<24)
  {
    addr=2;
    column=23-x;
  }
  else if(x>=24 && x<32)
  {
    addr=3;
    column=31-x;
  }
  else
  {
    addr=4;
  }
  lc.setLed(addr,row,column,state);
```

```
}

static void DecimalPoint(int state)//Displays the "decimal point" where needed
{
  pinControl(16,0,state);
}


static void sign(int state, int sign)//Displays the + or - sign where needed
{
  for(int i=1; i<=5; i++)
  {
    pinControl(i,3,state);
  }
  if(sign==1)
  {
    for(int i=1;i<=5;i++)
    {
      pinControl(3,i,state);
    }
  }
}

static void clear() //clears all the thurned on lights
{
  lc.clearDisplay(0);
  lc.clearDisplay(1);
  lc.clearDisplay(2);
  lc.clearDisplay(3);
}

  private:
static void SegmentLight(int frame, int segment, bool state) //top-1 topleft-2
topright-3 middle-4 bottom-left 5 bottom right6, lights the segments of the
"7-segment display system"
{
      switch(segment)
      {
        case 1: for(int i=2;i<=5;i++)
                {pinControl(i+(frame*8),6,state);
                }
                break;
        case 2: for(int i=4;i<=6;i++)
                {
                  pinControl(2+(frame*8),i, state);
                }
                break;
        case 3: for(int i=4;i<=6;i++)
                {
```

```
                  pinControl(5+(frame*8),i,state);
                }
                break;
        case 4: for(int i=2;i<=5;i++)
                {
                  pinControl(i+(frame*8),3,state);
                }
                break;
        case 5:for(int i=0;i<=2;i++)
                {
                  pinControl(2+(frame*8),i,state);
                }
                break;
        case 6:for(int i=0;i<=2;i++)
                {
                  pinControl(5+(frame*8),i,state);
                }
                break;
        case 7:for(int i=2;i<=5;i++)
                {
                  pinControl(i+(frame*8),0,state);
                }
                break;

      }


}

static void letter_X(int a) //Prints Letter X
{
  lc.setLed(a,0,6,1);
  lc.setLed(a,0,0,1);
  lc.setLed(a,1,1,1);
  lc.setLed(a,1,5,1);
  lc.setLed(a,2,2,1);
  lc.setLed(a,2,4,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,4,2,1);
  lc.setLed(a,4,4,1);
  lc.setLed(a,5,1,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,0,1);
  lc.setLed(a,6,6,1);
}
static void letter_Y(int a)//Prints Letter Y
{
  lc.setLed(a,0,3,1);
  lc.setLed(a,1,3,1);
```

```
  lc.setLed(a,2,3,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,4,2,1);
  lc.setLed(a,4,4,1);
  lc.setLed(a,5,1,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,1,1);
  lc.setLed(a,6,5,1);

}
static void letter_b(int a)////Prints Letter b
{
  lc.setLed(a,0,5,1);
  lc.setLed(a,0,4,1);
  lc.setLed(a,0,3,1);
  lc.setLed(a,1,5,1);
  lc.setLed(a,1,2,1);
  lc.setLed(a,2,5,1);
  lc.setLed(a,2,2,1);
  lc.setLed(a,3,5,1);
  lc.setLed(a,3,4,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,4,5,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,5,1);


}
static void letter_omega(int a)////Prints Letter omega
{
  lc.setLed(a,1,5,1);
  lc.setLed(a,1,4,1);
  lc.setLed(a,1,2,1);
  lc.setLed(a,1,1,1);
  lc.setLed(a,2,6,1);
  lc.setLed(a,2,3,1);
  lc.setLed(a,2,0,1);
  lc.setLed(a,3,6,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,3,0,1);
  lc.setLed(a,4,6,1);
  lc.setLed(a,4,3,1);
  lc.setLed(a,4,0,1);


}
static void letter_K(int a)////Prints Letter K
{
  lc.setLed(a,0,5,1);
```

```
  lc.setLed(a,1,5,1);
  lc.setLed(a,2,5,1);
  lc.setLed(a,3,5,1);
  lc.setLed(a,4,5,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,5,1);
  lc.setLed(a,6,2,1);
  lc.setLed(a,2,4,1);
  lc.setLed(a,4,4,1);
  lc.setLed(a,1,3,1);
  lc.setLed(a,5,3,1);
  lc.setLed(a,0,2,1);
  lc.setLed(a,0,2,1);


}
static void letter_S(int a)////Prints Letter S
{
  lc.setLed(a,0,4,1);
  lc.setLed(a,0,3,1);
  lc.setLed(a,1,5,1);
  lc.setLed(a,1,2,1);
  lc.setLed(a,5,2,1);
  lc.setLed(a,3,4,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,4,5,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,5,2,1);
  lc.setLed(a,6,4,1);
  lc.setLed(a,6,3,1);
  lc.setLed(a,2,2,1);


}
static void letter_M(int a)//Prints Letter M
{
  lc.setLed(a,0,6,1);
  lc.setLed(a,1,6,1);
  lc.setLed(a,2,6,1);
  lc.setLed(a,3,6,1);
  lc.setLed(a,4,6,1);
  lc.setLed(a,5,6,1);
  lc.setLed(a,6,6,1);
  lc.setLed(a,0,0,1);
  lc.setLed(a,1,0,1);
  lc.setLed(a,2,0,1);
  lc.setLed(a,3,0,1);
  lc.setLed(a,4,0,1);
  lc.setLed(a,5,0,1);
```

```
  lc.setLed(a,6,0,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,4,4,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,4,2,1);
  lc.setLed(a,5,1,1);


}
static void letter_P(int a)//Prints Letter X
{
  lc.setLed(a,0,5,1);
  lc.setLed(a,1,5,1);
  lc.setLed(a,2,5,1);
  lc.setLed(a,3,5,1);
  lc.setLed(a,4,5,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,5,1);
  lc.setLed(a,6,4,1);
  lc.setLed(a,6,3,1);
  lc.setLed(a,5,2,1);
  lc.setLed(a,4,2,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,3,4,1);



}
static void letter_R(int a)//Prints Letter R
{
  lc.setLed(a,0,5,1);
  lc.setLed(a,1,5,1);
  lc.setLed(a,2,5,1);
  lc.setLed(a,3,5,1);
  lc.setLed(a,4,5,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,5,1);
  lc.setLed(a,6,4,1);
  lc.setLed(a,6,3,1);
  lc.setLed(a,5,2,1);
  lc.setLed(a,4,2,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,3,4,1);
  lc.setLed(a,2,4,1);
  lc.setLed(a,1,3,1);
  lc.setLed(a,0,2,1);

}
static void letter_L(int a)//Prints Letter l
```

```
{
  lc.setLed(a,0,5,1);
  lc.setLed(a,1,5,1);
  lc.setLed(a,2,5,1);
  lc.setLed(a,3,5,1);
  lc.setLed(a,4,5,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,5,1);
  lc.setLed(a,0,4,1);
  lc.setLed(a,0,3,1);
  lc.setLed(a,0,2,1);
}
static void letter_V(int a)//Prints Letter V
{
  lc.setLed(a,6,6,1);
  lc.setLed(a,5,6,1);
  lc.setLed(a,4,5,1);
  lc.setLed(a,3,5,1);
  lc.setLed(a,2,4,1);
  lc.setLed(a,1,4,1);
  lc.setLed(a,0,3,1);
  lc.setLed(a,1,2,1);
  lc.setLed(a,2,2,1);
  lc.setLed(a,3,1,1);
  lc.setLed(a,4,1,1);
  lc.setLed(a,5,0,1);
  lc.setLed(a,6,0,1);
}
static void letter_O(int a)//Prints Letter O
{
  lc.setLed(a,1,4,1);
  lc.setLed(a,1,3,1);
  lc.setLed(a,2,5,1);
  lc.setLed(a,2,2,1);
  lc.setLed(a,3,6,1);
  lc.setLed(a,3,1,1);
  lc.setLed(a,4,6,1);
  lc.setLed(a,4,1,1);
  lc.setLed(a,5,2,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,4,1);
  lc.setLed(a,6,3,1);
}
static void letter_A(int a)//Prints Letter A
{
  lc.setLed(a,0,5,1);
  lc.setLed(a,0,1,1);
  lc.setLed(a,1,5,1);
```

```
  lc.setLed(a,1,1,1);
  lc.setLed(a,2,5,1);
  lc.setLed(a,2,1,1);
  lc.setLed(a,3,5,1);
  lc.setLed(a,3,1,1);
  lc.setLed(a,4,5,1);
  lc.setLed(a,4,1,1);
  lc.setLed(a,5,4,1);
  lc.setLed(a,5,2,1);
  lc.setLed(a,6,3,1);
  lc.setLed(a,3,4,1);
  lc.setLed(a,3,3,1);
  lc.setLed(a,3,2,1);
}
static void letter_C(int a)//Prints Letter C
{
  lc.setLed(a,0,4,1);
  lc.setLed(a,0,3,1);
  lc.setLed(a,1,5,1);
  lc.setLed(a,2,6,1);
  lc.setLed(a,3,6,1);
  lc.setLed(a,4,6,1);
  lc.setLed(a,5,5,1);
  lc.setLed(a,6,3,1);
  lc.setLed(a,6,4,1);
  lc.setLed(a,5,2,1);
  lc.setLed(a,1,2,1);

}
public:
static void LightAll(int frame, bool state)//Lights all 7 segments of the 7
segment-like coded for the LED Matrix
{
  for(int i=1;i<=7;i++)
  {
    SegmentLight(frame,i,state);
  }
}

static void selector(int a, int b, int c)//a is flag 4, b is flag 2, c is flag
3, prints the appropriate text on the LED Matrix depending on whichparmater we
are the entering, which in tur  is controlled by 3 flags
{
  if (a==0)
  {
    if(b==0)
    {
      if(c==0)
```

```
      {
        letter_M(0);
        letter_S(1);
        letter_R(2);
        letter_X(3);
      }
      else if (c==1)
      {

        letter_M(0);
        letter_S(1);
        letter_R(2);
        letter_Y(3);
      }
    }
    else if(b==1)
    {
      if(c==0)
      {
        letter_S(0);
        letter_L(1);
        letter_V(2);
        letter_X(3);
      }
      else if(c==1)
      {
        letter_S(0);
        letter_L(1);
        letter_V(2);
        letter_Y(3);
      }

    }
    else if(b==2)
    {
      letter_P(0);
      letter_A(1);
      letter_R(2);
      letter_K(3);
    }
  }
  else if(a==1)
  {
    if (b==0)
    {
      if(c==0)
      {
        letter_O(0);
```

```
        letter_S(1);
        letter_C(2);
        letter_X(3);
      }
      else if(c==1)
      {
        letter_O(0);
        letter_S(1);
        letter_C(2);
        letter_Y(3);
      }
    }
    else if (b==1)
    {
      if(c==0)
      {
        letter_P(0);
        letter_A(1);
        letter_R(2);
        letter_omega(3);
      }
      else if(c==1)
      {
        letter_P(0);
        letter_A(1);
        letter_R(2);
        letter_b(3);
      }
    }
  }
}
static void numberdisp (int n, int frame)//Displays the numbers using the
above written 7 segment display code
{
      switch(n)
      {
        case 0:
                SegmentLight(frame,1,true);
                SegmentLight(frame,2,true);
                SegmentLight(frame,3,true);
                SegmentLight(frame,5,true);
                SegmentLight(frame,6,true);
                SegmentLight(frame,7,true);
                pinControl((frame*8)+2,3,true);
                pinControl((frame*8)+5,3,true);
                break;
        case 1:
                SegmentLight(frame,3, true);
```

```
            SegmentLight(frame,6,true);
            pinControl((frame*8)+5,3,true);

            break;
        case 2:
            SegmentLight(frame,1,true);
            SegmentLight(frame,3,true);
            SegmentLight(frame,4,true);
            SegmentLight(frame,5,true);
            SegmentLight(frame,7,true);
            pinControl((frame*8)+2,6,true) ;
            pinControl((frame*8)+5,0,true);

            break;
        case 3:
            SegmentLight(frame,1,true);
            SegmentLight(frame,3,true);
            SegmentLight(frame,4,true);
            SegmentLight(frame,6,true);
            SegmentLight(frame,7,true);
            pinControl((frame*8)+2,6,true) ;
            pinControl((frame*8)+2,0,true);
            break;
        case 4:
            SegmentLight(frame,2,true);
            SegmentLight(frame,4, true);
            SegmentLight(frame,3,true);
            SegmentLight(frame,6,true);
            break;
        case 5:
            SegmentLight(frame,1,true);
            SegmentLight(frame,2,true);
            SegmentLight(frame,4,true);
            SegmentLight(frame,6,true);
            SegmentLight(frame,7,true);
            pinControl((frame*8)+5,6,true) ;
            pinControl((frame*8)+2,0,true);
            break;
        case 6:
            SegmentLight(frame,1,true);
            SegmentLight(frame,2,true);
            SegmentLight(frame,4,true);
            SegmentLight(frame,5,true);
            SegmentLight(frame,6,true);
            SegmentLight(frame,7,true);
            pinControl((frame*8)+5,6,true) ;
            break;
        case 7:
```

```
                SegmentLight(frame,3, true);
                SegmentLight(frame,6,true);
                SegmentLight(frame,1,true);
                pinControl((frame*8)+5,3,true) ;
                break;
        case 8:LightAll(frame, true);
                break;
        case 9:
                SegmentLight(frame,1,true);
                SegmentLight(frame,2,true);
                SegmentLight(frame,3,true);
                SegmentLight(frame,4,true);
                SegmentLight(frame,6,true);
                SegmentLight(frame,7,true);
                pinControl((frame*8)+2,0,true) ;

                break;
      }


  }

};




  float min=0;//When a value is being read from the user, these sets the range of the minimum and maximum value they can enter, depending on what paraeter the value is being entered to
  float max=0;

  float valuearray[40];//Since values can fluctuate very rapidly due to noise, we average the value over 40 clock cylces, and display them for the next 40 cycles, This array holds the 40 values to be averaged
  int valuearraycounter=0;//This is a counter which lets the boards know when all 40 values have been received for averaging

  int valuearraycounterflag=0;//Since we display the input mode and the value in a 3:1 duty cycle, the flag lets the board know when the mode has to be shown, and when the value of the input has to be shown

  float h=0.02;//The value of h used to calculate all variables at various times using Runge-Kutta function (refer section: )
float Amplitude;//The actual variable where Amplitude will be stored

//list of Master SLave Variables
  float xm = 0;
  float xs = 0;
```

```
  float ym = 0;
  float ys = 0;
  float k = 0;
  float nu=0;

//list of Harmonic Oscialttor Variables
  int x=0;
  int y=0;
  int w=0;
  int b=0;
void setup() {
//initialising our pins and awakening our LEDMaatrix
  Serial.begin(9600);
  pinMode(A0, INPUT);//potentiometer
  pinMode(A1,INPUT);
  pinMode(3, INPUT);//Mode + Oscillator selection
  pinMode(4, INPUT);// Initial value selector
  pinMode(5, OUTPUT);//Clock pulse
   lc.shutdown(0,false);
 lc.shutdown(1,false);
 lc.shutdown(2,false);
lc.shutdown(3,false);
//intialising our above declared averaging array
for(int j=0; j<40;j++)
{
  valuearray[j]=0;
}
}


float normalise2(int x,float start, float end)//normalizes the DAC 0-1023
Input to the range between min and max (declared above)
{
  float answer = start + (end - start)*(((float)(x))/1023.00);
  return answer;
}

int normalize( float x, float Amplitude) //Normalizes the value of both
osicllators based on the Amplitude calculted for them
{
  float a = 16*(x/Amplitude+1);

  int y = (int)a;
  return y;
}


HarmonicOscillator obj1=HarmonicOscillator();//creating the objets we will
work with
```

```
MasterSlave obj2=MasterSlave();

int AmpFlag =0;//Since our implemwntation is in a loop, and we don't want
Amplitude to be recalculated every time (time sonsuming process), we have a
flag to check
int OscillatorInit=0;//Since our implemwntation is in a loop, and we don't
want the initial conditions to be reinitialized every time since that will
make the oscillations stationary, we have a flag to check

//four flags control what to display on the board
int flag1 = 0;//controls if we want to display the oscilations, or allow the
user to view and change parameters
int flag2 = 0;//controls which paramter to imput along with flag3
int flag3 = 0;//controls which paramter to imput along with flag2
int flag4 =0;//controls which osicllator the user wants to see

int flag1flag=0; //When button 1 is held for 16 cycles, the flag1 value is
changed, but to prevent it from changing after it has changed once, this flag
is set to 1, blocking any updates to flag1 till the button has been released
for sufficient amount of time
int flag2flag=0; //When button 1 is pressed, this updates, waiting for button
1 to be released
int flag2flag1=0; //If button 1 is released within 6 cycles, thus flag
updates. These two flags together allows flag2 to switch only in short button
pulses.

int counterflag=0;//flags which controls if counter should be upcounting or
downcounting depending on button pressed down or relaesed. This flag updates
when the button is pressed
int counterflag2=0; //This updates when the button is releaed, indicitating
that the counter should downcount now
int counter=0;//the counter controlled by the above flags, counts for how many
cycles a button has been presses

int ic_counter =0;//same as counter for the other button
int ic_counterflag=0;
int ic_counterflag2=0;

//same as flag1flag and flag2flags, but for the flag3 and flag4, controlled by
the other button
int flag3flag=0;
int flag3flag1=0;
int flag4flag=0;

int prime=0;//sets to 1 when button1 is released, preventing any new actions
from occuring, till counter reaches 0
int prime2=0;//sets to 1 when button1 is released, preventing any new actions
from occuring, till counter reaches 0
```

```
int flaglight=0;//flag to detect if the board needs to be cleared when there
is a mode switch


void loop() {



  int mode = digitalRead(3); //two inputs for the two switches
  int input_value = digitalRead(4);




// prevents users from entering buttons before the counter is 0, prevents
button mashing issues (button 1)
  if(counterflag==0 && mode==1)
  {
    counter++;

    counterflag2=1;
  }

  if(counterflag2==1 && mode==0)
  {
  counterflag=1;
  }
  if(counterflag==1)
  {
    counter--;
    prime=1;
  }

//maximum bound on counter, to prevent very high counter decrement time
(button 1)

  if(counter>20)
  {
    counter=20;
  }



// prevents users from entering buttons before the counter is 0, prevents
button mashing issues (button 2)
if(ic_counterflag==0 && input_value==1)
  {
    ic_counter++;
```

```
    ic_counterflag2=1;
  }

  if(ic_counterflag2==1 && input_value==0)
  {
    ic_counterflag=1;

  }

if(ic_counterflag==1)
  {
    ic_counter--;
    prime2=1;


  }

 //maximum bound on counter, to prevent very high counter decrement time
(button 2)
  if(ic_counter>20)
  {
    ic_counter=20;
  }

  // end of the button mashing protection, as long as the array doesnt reach
0, no new inputs will be taken, for both buttons


  // resets all the flags of flags, when counter is 0, implying system is
primed to take new inputs
  if(counter==0)
  {
     flag1flag=0;
     flag2flag=0;
     flag2flag1=0;
     counterflag=0;
     counterflag2=0;
     prime=0;
  }
  if(ic_counter==0)
  {
    ic_counterflag=0;
    flag3flag=0;
    flag3flag1=0;
    flag4flag=0;
    ic_counterflag2=0;
    prime2=0;
  }
```

```
  //switches mode from display mode to variable selection mode
  if(counter>=16 && flag1flag==0)
  {flag1flag=1;//prevents constant flipping of mode from display to input
    if(flag1 == 0)
    {
      flag1 = 1;

    }
    else if(flag1 == 1)
    {
      flag1 = 0;
    }
  }
  //end of switch mode segment

  //switches MasterSlave and Harmonic Oscillator Mode
if(ic_counter>=16 && flag4flag==0)
  {flag4flag=1;//prevents constant flipping of mode from SHO to Master Slave
   AmpFlag=0;
   OscillatorInit=0;
    if(flag4 == 0)
    {
      flag4 = 1;
      if(flag2==2)
      {
        flag2=0;
      }

    }
    else if(flag4 == 1)
    {
      flag4 = 0;
    }
  }
  // end of change oscillator segement

  if (flag1 == 0  && flag1flag==0)//display mode active
    {
      if(flaglight==1)//ressting the display if newly switched from input mode
      {flaglight=0;

    LEDlight::clear();


      }

    if(flag4==1)//Simple Harmonic Oscillator mode
```

```
{
      if(OscillatorInit==0)//Initialising the oscillator if unitialised
      {
        obj1.x=x;
        obj1.y=y;
        obj1.w=w;
        obj1.b=b;
        OscillatorInit=1;
      }
        if(AmpFlag==0)//Calculating Amplitude if uncalculated
  {AmpFlag=1;

  Amplitude=obj1.getAmplitude(h);


  }

    int xmatrix = normalize(obj1.x, Amplitude);//normalizing the x value to be
displayed

  //Switching on the appropriate LEDs
    LEDlight::pinControl(xmatrix,0,true);
    LEDlight::pinControl(xmatrix,1,true);
    LEDlight::pinControl(xmatrix,2,true);
    LEDlight::pinControl(xmatrix,3,true);
    LEDlight::pinControl(xmatrix,4,true);
    LEDlight::pinControl(xmatrix,5,true);
    LEDlight::pinControl(xmatrix,6,true);
    LEDlight::pinControl(xmatrix,7,true);

    delay(50);

  float addx=obj1.calculateOscillator(h,1);

  float addy=obj1.calculateOscillator(h,2);
  obj1.change(addx,addy);//updating the x and y=dx/dt values of the oscillator
for the next loop iteration

  //Switching of the LEDs
  LEDlight::pinControl(xmatrix,0,false);
  LEDlight::pinControl(xmatrix,1,false);
  LEDlight::pinControl(xmatrix,2,false);
  LEDlight::pinControl(xmatrix,3,false);
  LEDlight::pinControl(xmatrix,4,false);
  LEDlight::pinControl(xmatrix,5,false);
  LEDlight::pinControl(xmatrix,6,false);
  LEDlight::pinControl(xmatrix,7,false);
```

```
    }

    if(flag4==0)//Master Slave Oscillator Mode
    {
        if(OscillatorInit==0)//Initialising the oscillator if unitialised
        {

           obj2.xm=xm;
            obj2.ym=ym;
            obj2.xs=xs;
           obj2.ys=ys;
           obj2.mu=0.01;
           obj2.K=k;
          OscillatorInit=1;

        }



  if(AmpFlag==0)//Calculating Amplitude if uncalculated
  {AmpFlag=1;
  Amplitude=obj2.getAmplitude(h);
  }

 int xmatrix = normalize(obj2.xm, Amplitude);//normalizing the Master and
Slave Oscillators X values to be displayed
int xsmatrix = normalize(obj2.xs, Amplitude);

//Switching on the appropriate LEDs
LEDlight::pinControl(xmatrix,5,true);
LEDlight::pinControl(xmatrix,6,true);
LEDlight::pinControl(xsmatrix,1,true);
LEDlight::pinControl(xsmatrix,2,true);

//calculating the values of x and y for the next oscillation
float xmadd=obj2.RK4Calculator(1,h);
float ymadd=obj2.RK4Calculator(2,h);
float xsadd=obj2.RK4Calculator(3,h);
float ysadd=obj2.RK4Calculator(4,h);

delay(10);

obj2.change(xmadd, ymadd, xsadd, ysadd );//Updating the values of x and y for
master and slave oscillators for next cycle

//Switching of the LEDs
LEDlight::pinControl(xmatrix,5,false);
LEDlight::pinControl(xmatrix,6,false);
LEDlight::pinControl(xsmatrix,1,false);
```

```
LEDlight::pinControl(xsmatrix,2,false);

    }

     }
  else if (flag1 == 1)//variable slection mode
  { AmpFlag=0;

  OscillatorInit=0;//restting flags for display mode so that next time display
mode is switched into, the values are recalculated and reinitialised
    if(flaglight==0)//clearing the LEDMatrix from display mode LEDs
    {
  LEDlight::clear();

    flaglight=1;
    }


    int digital_value = analogRead(A0);//Reading the coarse knob value
    int digital_value2 = analogRead(A1);//Reading the fine knob vake



    float value = normalise2(digital_value, min, max);//Normalising the
readings to the required range
    float max2=(max-min)/10;


    float value2 = normalise2(digital_value2, 0, max2);

    value=value+value2;
    //the final value of the reading is the sum of the total normalised
reading of both knobs

    if(value>max)
    {
      value=max;
    }
    if(value<min)
    {
      value=min;
    }//check in case the fine knob pushes value outside our desginated range

    valuearray[valuearraycounter]=value;//filling the value array, whose
average will be calculated after some cycles
    valuearraycounter++;
    float average=0;
```

```
    if(valuearraycounter==40 && valuearraycounterflag<=2)//40 values filled,
hence average of the 40 values is calculated
    { LEDlight::clear();
      LEDlight::DecimalPoint(1);//printing the decimal point

      //printing the sign
      if(value>0)
      {
        LEDlight::sign(1,1);
      }
      else if(value<0)
      {
        LEDlight::sign(1,0);
      }
      //computing average abd clearing the array
      for(int i=0; i<40; i++)
      {
          average=average+valuearray[i];
          valuearray[i]=0;

      }
      average=average/40;



 //finding the first few digits (ones plays, one tenths place and one
hundredths place), to display on the LED Matirx
    float value2=abs(average);
  value2=value2*100;
  int value3=(int)(value2);
  int digit1=value3/100;
  value3=value3%100;
  int digit2=value3/10;
  int digit3=value3%10;

//diplaying the digits of LED Matrix
   LEDlight::numberdisp(digit1,1);

   LEDlight::numberdisp(digit2,2);
    LEDlight::numberdisp(digit3,3);

    valuearraycounter=0;
    valuearraycounterflag++;//three cycles of display to allow user to see the
reading clearly, after which the mode is displayed
    }
    if(valuearraycounter==30 && valuearraycounterflag==3)
    { LEDlight::clear();
        for(int i=0; i<30; i++)
```

```
      {

          valuearray[i]=0;

      }
      LEDlight::selector(flag4, flag2, flag3);//displays the appropriate input
reading mode
      valuearraycounterflag=0;
      valuearraycounter=0;
    }


    if(flag2 == 0)
    {
      if(flag4==0)
      {
      if (flag3 == 0) // master oscillator x value
      {xm = value;
        min=-5.00;
        max=5.00;
      }
      else if (flag3 == 1)//master oscillator y value
      {ym = value;
         min=-5.00;
        max=5.00;
      }
      }
      if(flag4==1)
      {
        if(flag3==0) // SHO x value
        {x=value;
        min=-5.00;
        max=5.00;
        }
        else if(flag3==1) //SHO y value
        {
          y=value;
          min=-5.00;
          max=5.00;
        }
      }
    }
    else if(flag2 == 1)
    {

      if(flag4==0)
      {
      if (flag3 == 0)//slave oscillator x value
```

```
      {xs = value;

      min=-abs(xm);
      max=abs(xm);
      }
      else if (flag3 == 1)//slave oscillator y value
      {ys = value;

      min=-5.00;
      max=5.00;
      }
      }
      else if(flag4==1)
      {
        if (flag3==0)
       { w=value; //SHO omega value
        min=0;
        max=M_PI;
       }

        else if(flag3==1)
        {b=value;//SHO damping b value
        min=0;
        max=3*M_PI;
        }

      }
    }
    else if(flag2 == 2)
    {
      k = value;//k value, mu is fixed
      min=0;
      max=1;
    }
//switching varibale modes
    if(counter>0 && counter<=6 && flag2flag==0 && prime==0)
    {
      flag2flag=1; //button has been presses

    }

    if(counter>6 && flag2flag==1 && prime==0)
    {
      flag2flag=0;
    }

    if(flag2flag==1 && flag2flag1==0 && counter<=6 && mode==0)
```

```
       { flag2flag1=1;//button has been released after button was pressed
within 6 ticks

    }
     //button press has been detected



    if(flag2flag==1 && flag2flag1==1)//we change flag2 so we change which mode
we are in
    {
      if(flag4==0)
      {
        flag2++;
        flag2=flag2%3;
      }

      else if(flag4==1)
      {
        flag2=abs(flag2-1);
      }
      flag2flag=0;
      flag2flag1=0;

    }
    //mode has been switched

    //flag3 only has value in x-y master slave modes
    //repeating flag 3 button press check
    if(ic_counter>0 && ic_counter<=6 && flag3flag==0 && prime2==0)
     {
      flag3flag=1; //button has been presses

    }
    if(ic_counter>6 && flag3flag==1 && prime==0)
    {
      flag3flag=0;
    }


    if(flag3flag==1 && flag3flag1==0 && ic_counter<=6 && input_value==0)
       {
          flag3flag1=1;//button has been released after button was pressed
within 6 ticks
    }
     //button press has been detected

    if(flag3flag==1 && flag3flag1==1)
```

```
    {
      flag3=abs(flag3-1);
      flag3flag1=0;
      flag3flag=0;

    }


  }
  //reset();




  digitalWrite(5, HIGH);//Artificial Clock Pulse
  digitalWrite(5, LOW);

if(flag1==1 || (flag1==0 && counterflag2==1) || (flag1==0 &&
ic_counterflag2==1))//Activate Clock Pulse only if we are in input mode
{
delay(50);

}
}
```