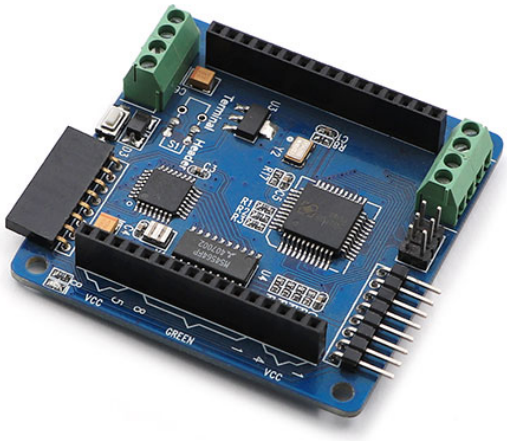
Arduino RGB Colorduino
User's Manual



Overview:

Colorsduino is a magic RGB LED dot-matrix driver compatible with Arduino . Colorsduino pairs the M54564 with a single DM163 constant current driver. By using the DM163, Colorduino gains three 8+6-bit channels of hardware PWM control of the LED's freeing up the MCU from having to implement it in software. This gives the ATmega328 more CPU bandwidth for performing other tasks. Colorduino is easy to cascade by IIC and Power interface.

Features:

- 8bits colors support with 6bits correction for each color in every dots
- Hardware 16MHz PWM support
- Without any external circuits, play and shine
- Dedicated GPIO and ADC interface
- Hardware UART and IIC communication with easy cascading
- 24 constant current channels of 100mA each
- 8 super source driver channels of 500mA each

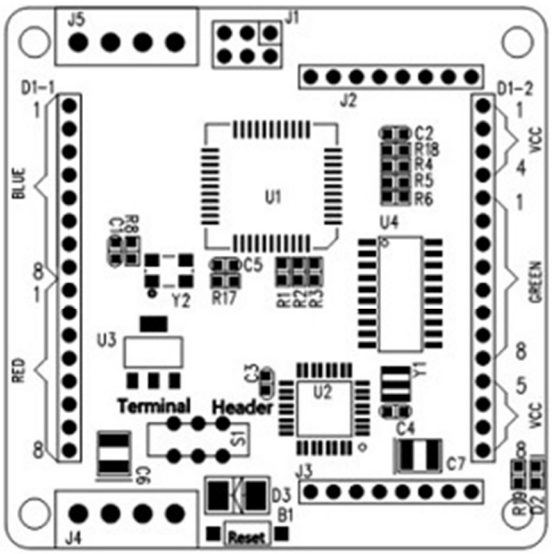
Specifications:

PCB size	60mm X 60mm X 1.6mm
Microprocessor	Atmega328P
Indicator	PWR State
Power supply	5V~7.5V DC(7.5V Max)
Cascade power connector	Terminal Blocks
Program interface	UART/ISP
Expansion socket	100mil bended pin header pair
Communication protocols	UART/IIC
RoHS	Yes

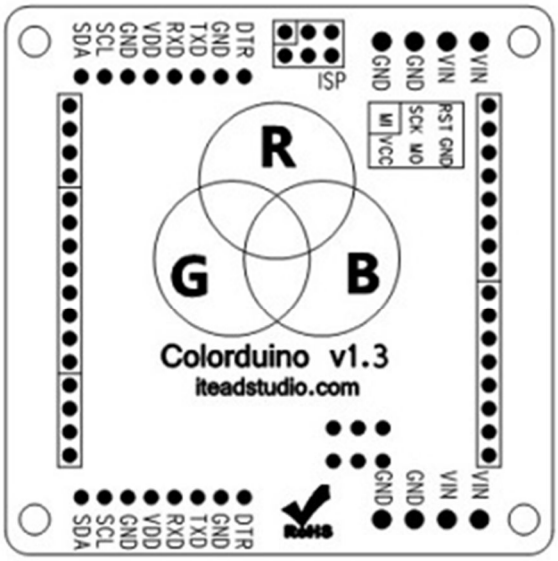
Electrical Characteristics

Specification	Min.	Typical Value	Max.	Unit
Power Voltage	3.5	5	7.5	VDC
Input Voltage VH:	4.5	5	5.5	
Input Voltage VL:	-0.3	0	0.5	V
Current Consumption(Except LED matrix)	-	20	40	mA
Drive current(Every channel)			500	mA
Drive current(Every dot)			58	mA
Circuit response time	10			ns
RGB LED-Matrix color resolution per dot			16M	
Uart Baud rate	9600		115200	bps

Hardware:



Top View



Bottom View

Pad Name	Type	Description
SDA	I/O	Data wire of IIC Bus
SCL	I/O	Clock wire of IIC Bus
GND	G	Ground plane
VDD	P	Power wire for all digital components
RXD	I/O	Data wire of UART Bus
TXD	I/O	Data wire of UART Bus
DTR	I	Special reset wire of Arduino program
VIN	P	Power wire for all LEDs and Super current driver
MI	I/O	Data input of ISP
MO	I/O	Data output of ISP
SCK	I/O	Clock input of ISP
RST	I	Reset input of ISP

Pin Configuration:

Mount the 8X8 Matrix on the COLORDUINO

Arduino	412 ARDUINO RGB COLORDUINO
RX	RX
TX	TX
RESET	DTR
5V	5V
GND	GND

Library found in Software

Demo Code 1:

```
/*
  Colorduino - Colorduino DEMO for Arduino.
  Copyright (c) 2010 zzy@IteadStudio.  All right reserved.

*/
#include <Wire.h>
#include <avr/pgmspace.h>
/*****************************
define the operate commands
*****************************/

/*****************************
define the status
*****************************/

/*****************************
define the IO
*****************************/
#define RST_BIT 0x04
#define LAT_BIT 0x02
#define SLB_BIT 0x01
#define SCL_BIT 0x40
#define SDA_BIT 0x80

#define RST PORTC
#define LAT PORTC
#define SLB PORTC
#define SDA PORTD
#define SCL PORTD
```

```
#define open_line0   {PORTB=0x01;}
#define open_line1   {PORTB=0x02;}
#define open_line2   {PORTB=0x04;}
#define open_line3   {PORTB=0x08;}
#define open_line4   {PORTB=0x10;}
#define open_line5   {PORTB=0x20;}
#define open_line6   {PORTD=0x08;}
#define open_line7   {PORTD=0x10;}
#define close_all_line          {PORTD=0x00;PORTB=0x00;}

/****************************************
define the data zone
****************************************/
/*
//Test dots
unsigned char Tdots[8][8][3]=    {{{0,0,255},    {0,0,255},    {0,0,255},    {0,0,255},   {0,0,255},
{0,0,255},     {0,0,255},      {0,0,255}},
                    {{0,165,255},  {0,165,255},   {0,165,255},  {0,165,255}, {0,165,255},   {0,165,255},
{0,165,255},    {0,165,255}},
                    {{0,255,255},  {0,255,255},   {0,255,255},  {0,255,255}, {0,255,255},   {0,255,255},
{0,255,255},    {0,255,255}},
                    {{0,255,0},    {0,255,0},     {0,255,0},    {0,255,0},   {0,255,0},     {0,255,0},
{0,255,0},      {0,255,0}},
                    {{255,127,0},  {255,127,0},   {255,127,0},  {255,127,0}, {255,127,0},   {255,127,0},
{255,127,0},    {255,127,0}},
                    {{255,0,0},    {255,0,0},     {255,0,0},    {255,0,0},   {255,0,0},     {255,0,0},
{255,0,0},      {255,0,0}},
                    {{255,0,139},  {255,0,139},   {255,0,139},  {255,0,139}, {255,0,139},   {255,0,139},
{255,0,139},    {255,0,139}},
                    {{255,255,255}, {255,255,255}, {255,255,255}, {255,255,255},{255,255,255},
{255,255,255},  {255,255,255},   {255,255,255}}
                    };
*/
unsigned char dots[2][8][8][3] = {0};
//dots matrix
//[2]:Page:one for display, one for receive data
//[8]:Row:8 row in LED plane
//[8]:Column:8 column in one row
//[3]:Color:RGB data: 0 for Red; 1 for green, 2 for Blue
unsigned char Gamma_Value[3] = {10,63,63};
//Gamma correctly value, every LED plane is different.value range is 0~63
//[3]:RGB data, 0 for Red; 1 for green, 2 for Blue
unsigned char Page_Index = 0; // the index of buffer
unsigned char row = 0;//the value of row in LED plane, from 0~7
unsigned char column = 0;//the value of every row, from 0~7
unsigned char color = 0;//the value of every dots, 0 is Red, 1 is Green, 2 is Blue

unsigned char line = 0;

/************************************************
define the extern data zone
************************************************/
extern unsigned char font8_8[92][8];
```

```
extern unsigned char pic[4][8][8][3];
/************************************************
all parts inition functions zone
************************************************/
void _IO_Init()
{
  DDRD = 0xff; // set all pins direction of PortD
  DDRC = 0xff; // set all pins direction of PortC
  DDRB = 0xff; // set all pins direction of PortB

  PORTD = 0x00; // set all pins output is low of PortD
  PORTC = 0x00; // set all pins output is low of PortC
  PORTB = 0x00; // set all pins output is low of PortB
}
void _LED_Init()
{
  /*
  LED_RST(1);
  LED_Delay(1);
  LED_RST(0);
  LED_Delay(1);
  LED_RST(1);
  LED_Delay(1);
  */
  LED_RST(1);
  SetGamma();
  line = 0;

}
void _TC2_Init()
{
  TCCR2A |= (1 << WGM21) | (1 << WGM20);
  TCCR2B |= ((1<<CS22)|(1<<CS20));   // by clk/64
  TCCR2B &= ~((1<<CS21));   // by clk/64
  TCCR2A &= ~((1<<WGM21) | (1<<WGM20));   // Use normal mode
  ASSR |= (1<<AS2);      // Use internal clock - external clock not used in Arduino
  TIMSK2 |= (1<<TOIE2) | (0<<OCIE2B);   //Timer2 Overflow Interrupt Enable
  TCNT2 = 0xff;
  sei();
}


/************************************************
the timer2 operate functions zone
************************************************/
ISR(TIMER2_OVF_vect)        //Timer2  Service
{
  cli();
  TCNT2 = 0x64;     //flash a led matrix frequency is 100.3Hz,period is 9.97ms
  //TCNT2 = 0x63;     //flash a led matrix frequency is 99.66Hz,period is 10.034ms
    if(line > 7) line = 0;
    close_all_line;
    run(line);
    open_line(line);
```

```
   line++;
   sei();
}
/*****************************************************
the LED Hardware operate functions zone
*****************************************************/
void LED_SDA(unsigned char temp)
{
 if (temp)
   SDA|=SDA_BIT;
 else
   SDA&=~SDA_BIT;
}

void LED_SCL(unsigned char temp)
{
 if (temp)
   SCL|=SCL_BIT;
 else
   SCL&=~SCL_BIT;
}

void LED_RST(unsigned char temp)
{
 if (temp)
   RST|=RST_BIT;
 else
   RST&=~RST_BIT;
}

void LED_LAT(unsigned char temp)
{
 if (temp)
   LAT|=LAT_BIT;
 else
   LAT&=~LAT_BIT;
}

void LED_SLB(unsigned char temp)
{
 if (temp)
   SLB|=SLB_BIT;
 else
   SLB&=~SLB_BIT;
}
/*****************************************************
the LED datas operate functions zone
*****************************************************/
void SetGamma()
{
 unsigned char i = 0;
 unsigned char j = 0;
 unsigned char k = 0;
 unsigned char temp = 0;
```

```
 LED_LAT(0);
 LED_SLB(0);
 for(k=0;k<8;k++)
  for(i = 3;i > 0 ;i--)
  {
   temp = Gamma_Value[i-1]<<2;
   for(j = 0;j<6;j++)
   {
    if(temp &0x80)
     LED_SDA(1);
    else
     LED_SDA(0);

    temp =temp << 1;
    LED_SCL(0);
    LED_SCL(1);
  }
 }
 LED_SLB(1);
}
void run(unsigned char k)
{
 unsigned char i = 0;
 unsigned char j = 0;
 unsigned char p = 0;
 unsigned char temp = 0;
 LED_SLB(1);
 LED_LAT(0);
 for(i = 0;i<8;i++)
 {

  for(j=0;j<3;j++)
  {
   temp = dots[Page_Index][k][i][2-j];
   for(p=0;p<8;p++)
   {
    if(temp & 0x80)
     LED_SDA(1);
    else
     LED_SDA(0);

    temp = temp<<1;
    LED_SCL(0);
    LED_SCL(1);
   }
  }
 }
 LED_LAT(1);
 LED_LAT(0);
}
void open_line(unsigned char x)
{
 switch (x)
 {
```

```
  case 0 :open_line0;
        break;
  case 1 :open_line1;
        break;
  case 2 :open_line2;
        break;
  case 3 :open_line3;
        break;
  case 4 :open_line4;
        break;
  case 5 :open_line5;
        break;
  case 6 :open_line6;
        break;
  case 7 :open_line7;
        break;
  default: close_all_line;
        break;
 }
}
/*******************************************************
Name:DispShowChar
Function:Display a English latter in LED matrix
Parameter:chr :the latter want to show
        R: the value of RED.   Range:RED 0~255
        G: the value of GREEN. Range:RED 0~255
        B: the value of BLUE.  Range:RED 0~255
        bias: the bias of a letter in LED Matrix.Range -7~7
*******************************************************/
void DispShowChar(char chr,unsigned char R,unsigned char G,unsigned char B,char bias)
{
 unsigned char i,j,Page_Write,temp;
 unsigned char Char;
 unsigned char chrtemp[24] = {0};

 if ((bias > 8) || (bias < -8))
  return;


 Char = chr - 32;

 if(Page_Index == 0)
  Page_Write = 1;
 if(Page_Index == 1)
  Page_Write = 0;
 j = 8 - bias;
 for(i = 0;i< 8;i++)
 {
  chrtemp[j] = pgm_read_byte(&(font8_8[Char][i]));
  j++;
 }
 for(i = 0;i < 8;i++)
 {
```

```
   temp = chrtemp[i+8];
   for(j = 0;j < 8;j++)
   {
    if(temp & 0x80)
    {
     dots[Page_Write][j][i][0] = B;
     dots[Page_Write][j][i][1] = G;
     dots[Page_Write][j][i][2] = R;
    }
    else
    {
     dots[Page_Write][j][i][0] = 0;
     dots[Page_Write][j][i][1] = 0;
     dots[Page_Write][j][i][2] = 0;
    }
    temp = temp << 1;
   }
  }
  Page_Index = Page_Write;
}
/*****************************************************
Name:DispShowColor
Function:Fill a color in LED matrix
Parameter:R: the value of RED.   Range:RED 0~255
        G: the value of GREEN. Range:RED 0~255
        B: the value of BLUE.  Range:RED 0~255
*****************************************************/
void DispShowColor(unsigned char R,unsigned char G,unsigned char B)
{
  unsigned char Page_Write,i,j;

  if(Page_Index == 0)
    Page_Write = 1;
  if(Page_Index == 1)
    Page_Write = 0;

  for (i = 0;i<8;i++)
    for(j = 0;j<8;j++)
    {
     dots[Page_Write][i][j][2] = R;
     dots[Page_Write][i][j][1] = G;
     dots[Page_Write][i][j][0] = B;
    }

  Page_Index = Page_Write;
}
/*****************************************************
Name:DispShowPic
Function:Fill a picture in LED matrix from FLASH
Parameter:Index:the index of picture in Flash.
*****************************************************/
void DispShowPic(unsigned char Index)
{
  unsigned char Page_Write,i,j;
```

```
  if(Page_Index == 0)
    Page_Write = 1;
  if(Page_Index == 1)
    Page_Write = 0;

  for (i = 0;i<8;i++)
  {

    for(j = 0;j<8;j++)
    {

      dots[Page_Write][i][j][0] = pgm_read_byte(&(pic[Index][i][j][2]));
      dots[Page_Write][i][j][1] = pgm_read_byte(&(pic[Index][i][j][1]));
      dots[Page_Write][i][j][2] = pgm_read_byte(&(pic[Index][i][j][0]));
    }
  }
  Page_Index = Page_Write;

}

/****************************************
the other operate functions zone
****************************************/
void LED_Delay(unsigned char i)
{
  unsigned int y;
  y = i * 10;
  while(y--);
}
/*************************************************
Main Functions zone
*************************************************/
void setup()
{
  _IO_Init();          //Init IO
  _LED_Init();         //Init LED Hardware
  _TC2_Init();         //Init Timer/Count2
}
void loop()
{
  unsigned int i = 1000;
  char j=0;
  /*
  DispShowChar('A',255,0,0,0);
  delay(i);
  DispShowChar('B',0,255,0,0);
  delay(i);
  DispShowChar('C',0,0,255,0);
  delay(i);
  DispShowChar('D',255,255,255,0);
  delay(i);
  DispShowChar('E',0,255,255,0);
  delay(i);
```

```
DispShowChar('F',255,255,0,0);
delay(i);
DispShowChar('G',255,0,255,0);
delay(i);
DispShowChar('H',255,127,0,0);
delay(i);
DispShowChar('I',127,255,0,0);
delay(i);
DispShowChar('J',127,0,255,0);
delay(i);
DispShowChar('K',255,127,127,0);
delay(i);
DispShowChar('L',127,127,255,0);
delay(i);
DispShowChar('M',127,255,127,0);
delay(i);
DispShowChar('N',255,127,255,0);
delay(i);
DispShowChar('O',200,120,120,0);
delay(i);
  DispShowChar('P',120,200,120,0);
delay(i);
DispShowChar('Q',200,120,120,0);
delay(i);
DispShowChar('R',120,120,200,0);
delay(i);
DispShowChar('S',120,120,120,0);
delay(i);
DispShowChar('T',127,0,100,0);
delay(i);
DispShowChar('U',255,0,200,0);
delay(i);
 DispShowChar('V',200,255,120,0);
delay(i);
DispShowChar('W',120,200,200,0);
delay(i);
DispShowChar('X',200,200,120,0);
delay(i);
DispShowChar('Y',127,0,180,0);
delay(i);
DispShowChar('Z',0,1800,200,0);
delay(i);
DispShowChar('a',255,0,0,0);
delay(i);
DispShowChar('b',0,255,0,0);
delay(i);
DispShowChar('c',0,0,255,0);
delay(i);
DispShowChar('d',255,255,255,0);
delay(i);
DispShowChar('e',0,255,255,0);
delay(i);
DispShowChar('f',255,255,0,0);
delay(i);
```

```
DispShowChar('g',255,0,255,0);
delay(i);
DispShowChar('h',255,127,0,0);
delay(i);
DispShowChar('i',127,255,0,0);
delay(i);
DispShowChar('j',127,0,255,0);
delay(i);
DispShowChar('k',255,127,127,0);
delay(i);
DispShowChar('l',127,127,255,0);
delay(i);
DispShowChar('m',127,255,127,0);
delay(i);
DispShowChar('n',255,127,255,0);
delay(i);
DispShowChar('0',200,120,120,0);
delay(i);
DispShowChar('p',120,200,120,0);
delay(i);
DispShowChar('q',200,120,120,0);
delay(i);
DispShowChar('r',120,120,200,0);
delay(i);
DispShowChar('s',120,120,120,0);
delay(i);
DispShowChar('t',127,0,100,0);
delay(i);
DispShowChar('u',255,0,200,0);
delay(i);
 DispShowChar('v',200,255,120,0);
delay(i);
DispShowChar('w',120,200,200,0);
delay(i);
DispShowChar('x',200,200,120,0);
delay(i);
DispShowChar('y',127,0,180,0);
delay(i);
DispShowChar('z',0,1800,200,0);
delay(i);
DispShowChar('0',200,120,120,0);
delay(i);
DispShowChar('1',120,120,200,0);
delay(i);
DispShowChar('2',120,120,120,0);
delay(i);
DispShowChar('3',127,0,100,0);
delay(i);
DispShowChar('4',255,0,200,0);
delay(i);
 DispShowChar('5',200,255,120,0);
delay(i);
DispShowChar('6',120,200,200,0);
delay(i);
```

```
DispShowChar('7',200,200,120,0);
delay(i);
DispShowChar('8',127,0,180,0);
delay(i);
DispShowChar('9',0,1800,200,0);
delay(i);


DispShowPic(0);
delay(i);
DispShowPic(1);
delay(i);
DispShowPic(2);
delay(i);
DispShowPic(3);
delay(i);


for(j = -7;j < 7;j++)
{
 DispShowChar('I',255,255,0,j);
 delay(i);
}
for(j = -7;j < 7;j++)
{
 DispShowChar('t',0,255,255,j);
 delay(i);
}
for(j = -7;j < 7;j++)
{
 DispShowChar('e',255,0,255,j);
 delay(i);
}
for(j = -7;j < 7;j++)
{
 DispShowChar('a',255,255,255,j);
 delay(i);
}
for(j = -7;j < 7;j++)
{
 DispShowChar('d',255,255,255,j);
 delay(i);
}
for(j = -7;j < 7;j++)
{
 DispShowChar('s',255,255,255,j);
 delay(i);
}
for(j = -7;j < 7;j++)
{
 DispShowChar('t',255,255,255,j);
 delay(i);
}
for(j = -7;j < 7;j++)
```

```
 {
  DispShowChar('u',255,255,255,j);
  delay(i);
 }
 for(j = -7;j < 7;j++)
 {
  DispShowChar('d',255,255,255,j);
  delay(i);
 }
 for(j = -7;j < 7;j++)
 {
  DispShowChar('i',255,255,255,j);
  delay(i);
 }
 for(j = -7;j < 7;j++)
 {
  DispShowChar('o',255,255,255,j);
  delay(i);
 }

 */
 DispShowPic(0);
 delay(i);
 DispShowPic(1);
 delay(i);
 DispShowPic(2);
 delay(i);
 /*
 DispShowPic(3);
 delay(i);
 DispShowPic(4);
 delay(i);
 DispShowPic(5);
 delay(i);
 */
}
```

Demo Code 2:

```
#include <Colorduino.h>

typedef struct
{
  unsigned char r;
  unsigned char g;
  unsigned char b;
} ColorRGB;

//a color with 3 components: h, s and v
typedef struct
{
  unsigned char h;
  unsigned char s;
  unsigned char v;
```

```
} ColorHSV;

unsigned char plasma[ColorduinoScreenWidth][ColorduinoScreenHeight];
long paletteShift;


//Converts an HSV color to RGB color
void HSVtoRGB(void *vRGB, void *vHSV)
{
 float r, g, b, h, s, v; //this function works with floats between 0 and 1
 float f, p, q, t;
 int i;
 ColorRGB *colorRGB=(ColorRGB *)vRGB;
 ColorHSV *colorHSV=(ColorHSV *)vHSV;

 h = (float)(colorHSV->h / 256.0);
 s = (float)(colorHSV->s / 256.0);
 v = (float)(colorHSV->v / 256.0);

 //if saturation is 0, the color is a shade of grey
 if(s == 0.0) {
  b = v;
  g = b;
  r = g;
 }
 //if saturation > 0, more complex calculations are needed
 else
 {
  h *= 6.0; //to bring hue to a number between 0 and 6, better for the calculations
  i = (int)(floor(h)); //e.g. 2.7 becomes 2 and 3.01 becomes 3 or 4.9999 becomes 4
  f = h - i;//the fractional part of h

  p = (float)(v * (1.0 - s));
  q = (float)(v * (1.0 - (s * f)));
  t = (float)(v * (1.0 - (s * (1.0 - f))));

  switch(i)
  {
   case 0: r=v; g=t; b=p; break;
   case 1: r=q; g=v; b=p; break;
   case 2: r=p; g=v; b=t; break;
   case 3: r=p; g=q; b=v; break;
   case 4: r=t; g=p; b=v; break;
   case 5: r=v; g=p; b=q; break;
   default: r = g = b = 0; break;
  }
 }
 colorRGB->r = (int)(r * 255.0);
 colorRGB->g = (int)(g * 255.0);
 colorRGB->b = (int)(b * 255.0);
}

unsigned int RGBtoINT(void *vRGB)
{
```

```
  ColorRGB *colorRGB=(ColorRGB *)vRGB;

  return (((unsigned int)colorRGB->r)<<16) + (((unsigned int)colorRGB->g)<<8) + (unsigned
int)colorRGB->b;
}


float
dist(float a, float b, float c, float d)
{
  return sqrt((c-a)*(c-a)+(d-b)*(d-b));
}


void
plasma_morph()
{
  unsigned char x,y;
  float value;
  ColorRGB colorRGB;
  ColorHSV colorHSV;

  for(y = 0; y < ColorduinoScreenHeight; y++)
    for(x = 0; x < ColorduinoScreenWidth; x++) {
      {
          value = sin(dist(x + paletteShift, y, 128.0, 128.0) / 8.0)
            + sin(dist(x, y, 64.0, 64.0) / 8.0)
            + sin(dist(x, y + paletteShift / 7, 192.0, 64) / 7.0)
            + sin(dist(x, y, 192.0, 100.0) / 8.0);
          colorHSV.h=(unsigned char)((value) * 128)&0xff;
          colorHSV.s=255;
          colorHSV.v=255;
          HSVtoRGB(&colorRGB, &colorHSV);

          Colorduino.SetPixel(x, y, colorRGB.r, colorRGB.g, colorRGB.b);
      }
  }
  paletteShift++;

  Colorduino.FlipPage(); // swap screen buffers to show it
}

/*********************************************************
Name: ColorFill
Function: Fill the frame with a color
Parameter:R: the value of RED.   Range:RED 0~255
        G: the value of GREEN. Range:RED 0~255
        B: the value of BLUE.  Range:RED 0~255
*********************************************************/
void ColorFill(unsigned char R,unsigned char G,unsigned char B)
{
  PixelRGB *p = Colorduino.GetPixel(0,0);
  for (unsigned char y=0;y<ColorduinoScreenWidth;y++) {
    for(unsigned char x=0;x<ColorduinoScreenHeight;x++) {
```

```
    p->r = R;
    p->g = G;
    p->b = B;
    p++;
   }
  }

  Colorduino.FlipPage();
}

void setup()
{
  Colorduino.Init(); // initialize the board

  // compensate for relative intensity differences in R/G/B brightness
  // array of 6-bit base values for RGB (0~63)
  // whiteBalVal[0]=red
  // whiteBalVal[1]=green
  // whiteBalVal[2]=blue
  unsigned char whiteBalVal[3] = {36,63,63}; // for LEDSEE 6x6cm round matrix
  Colorduino.SetWhiteBal(whiteBalVal);


  // start with morphing plasma, but allow going to color cycling if desired.
  paletteShift=128000;
  unsigned char bcolor;

  //generate the plasma once
  for(unsigned char y = 0; y < ColorduinoScreenHeight; y++)
   for(unsigned char x = 0; x < ColorduinoScreenWidth; x++)
   {
    //the plasma buffer is a sum of sines
    bcolor = (unsigned char)
    (
        128.0 + (128.0 * sin(x*8.0 / 16.0))
      + 128.0 + (128.0 * sin(y*8.0 / 16.0))
    ) / 2;
    plasma[x][y] = bcolor;
   }

 // to adjust white balance you can uncomment this line
 // and comment out the plasma_morph() in loop()
 // and then experiment with whiteBalVal above
 // ColorFill(255,255,255);
}

void loop()
{
  plasma_morph();
}
```

How to add Library:

Manual installation

To install the library, first quit the Arduino application.

Then uncompress the ZIP file containing the library. For example, if you're installing a library called "ArduinoParty", uncompress ArduinoParty.zip. It should contain a folder called ArduinoParty, with files like ArduinoParty.cpp andArduinoParty.h inside. (If the .cpp and .h files aren't in a folder, you'll need to create one. In this case, you'd make a folder called "ArduinoParty" and move into it all the files that were in the ZIP file, like ArduinoParty.cpp and ArduinoParty.h.)

Drag the ArduinoParty folder into this folder (your libraries folder). Under Windows, it will likely be called "My Documents\Arduino\libraries". For Mac users, it will likely be called "Documents/Arduino/libraries". On Linux, it will be the "libraries" folder in your sketchbook.

Your Arduino library folder should now look like this (on Windows):

My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.cpp
My Documents\Arduino\libraries\ArduinoParty\ArduinoParty.h
My Documents\Arduino\libraries\ArduinoParty\examples
....

or like this (on Mac):

Documents/Arduino/libraries/ArduinoParty/ArduinoParty.cpp
Documents/Arduino/libraries/ArduinoParty/ArduinoParty.h
Documents/Arduino/libraries/ArduinoParty/examples
...

or similarly for Linux.

There may be more files than just the .cpp and .h files, just make sure they're all there.

(The library won't work if you put the .cpp and .h files directly into the libraries folder or if they're nested in an extra folder. For example:

Documents\Arduino\libraries\ArduinoParty.cpp and
Documents\Arduino\libraries\ArduinoParty\ArduinoParty\ArduinoParty.cpp

won't work.)

Restart the Arduino application. Make sure the new library appears in the Sketch->Import Library menu item of the software..