WHADDA

EXCITING ELECTRONICS

USER MANUAL

**EN** RGB dot matrix board and driver board based on ATmega328

WPB439

*whadda.com*

# Introduction

**To all residents of the European Union**
**Important environmental information about this product**
This symbol on the device or the package indicates that disposal of the device after its lifecycle could harm the environment. Do not dispose of the unit (or batteries) as unsorted municipal waste; it should be taken to a specialized company for recycling. This device should be returned to your distributor or to a local recycling service. Respect the local environmental rules.
**If in doubt, contact your local waste disposal authorities.**

Thank you for choosing Whadda! Please read the manual thoroughly before bringing this device into service. If the device was damaged in transit, do not install or use it and contact your dealer.

# Safety Instructions

Read and understand this manual and all safety signs before using this appliance.

For indoor use only.

- This device can be used by children aged from 8 years and above, and persons with reduced physical, sensory or mental capabilities or lack of experience and knowledge if they have been given supervision or instruction concerning the use of the device in a safe way and understand the hazards involved. Children shall not play with the device. Cleaning and user maintenance shall not be made by children without supervision.

# General Guidelines

- Refer to the Velleman® Service and Quality Warranty on the last pages of this manual.
- All modifications of the device are forbidden for safety reasons. Damage caused by user modifications to the device is not covered by the warranty.
- Only use the device for its intended purpose. Using the device in an unauthorized way will void the warranty.
- Damage caused by disregard of certain guidelines in this manual is not covered by the warranty and the dealer will not accept responsibility for any ensuing defects or problems.
- Nor Velleman Group nv nor its dealers can be held responsible for any damage (extraordinary, incidental or indirect) – of any nature (financial, physical...) arising from the possession, use or failure of this product.
- Keep this manual for future reference.

## What is Arduino®

Arduino® is an open-source prototyping platform based on easy-to-use hardware and software. Arduino® boards are able to read inputs – light-on sensor, a finger on a button or a Twitter message – and turn it into an output – activating of a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so, you use the Arduino® programming language (based on Wiring) and the Arduino® software IDE (based on Processing). Additional shields/modules/components are required for reading a twitter message or publishing online. Surf to www.arduino.cc for more information.

## Product Overview

This Arduino®-based (ATmega328P) RGB LED matrix driver platform gains three 8+6-bit channels of hardware PWM control thanks to its LED driver. Its design allows you to easily change or write firmware using the Arduino® IDE.

## Specifications

- module dimensions: 60 x 60 x 16 mm
- microprocessor: ATmega328P
- indicator: PWR state
- power supply: 5-7.5 VDC
- cascade power connector: terminal blocks
- program interface: UART/ISP (WPI440)
- expansion socket: 100 mil bended pin header pair
- communication protocols: UART/IIC
- current consumption (except LED matrix): max. 40 mA
- drive current (every channel): max. 500 mA
- drive current (every dot): max. 58 mA
- circuit response time: 10 ns
- RGB LED matrix colour resolution per dot: 16 M
- UART baud rate: 9600-115200

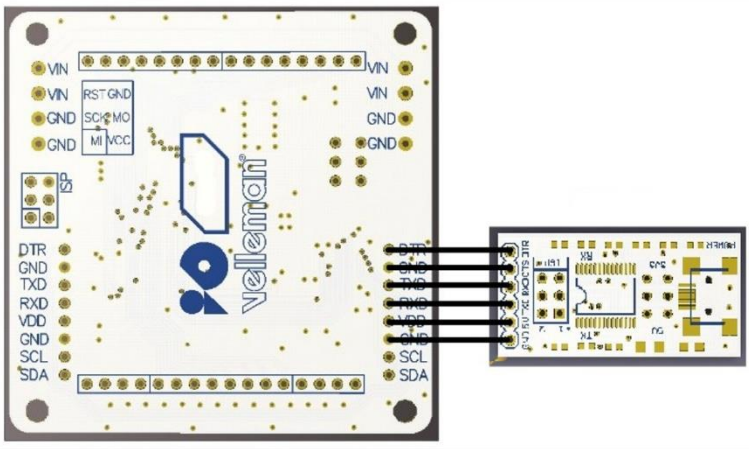
# Features

- 8 bits colours support with 6 bits correction for each colour in every dot
- hardware 16 MHz 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 100 mA each
- 8 super source driver channels of 500 mA each
- driver board is based on ATmega328 and works with IDE

# Example

## Using the WPI440 TTL-USB Interface

Insert the WPI440 TTL-USB into the WPB439 controller board. Be sure that the pins correspond (DTR to DTR). Connect the USB connection to the USB port of your laptop/desktop and start the Arduino® IDE.



## Connecting the WPB439 by Using an Arduino® UNO without WPI440

Carefully remove the ATmega chip from your Arduino® (be careful not to bend the pins).

Make following connections between the WPB439 and the UNO board:

| Arduino® | | WPB439 |
|---|---|---|
| RX | ▶ | RX* |
| TX | ▶ | TX* |
| RST | ▶ | DTR |
| +5V | ▶ | +5V |
| GND | ▶ | GND |

* Indeed, RX to RX and TX to TX, this is not a mistake!

Now, plug in the Arduino® into the USB port of your computer. The Arduino® board will act as a USB-TTL converter.

The RGB display can run on the USB 5 V power. However, we recommend applying a 5 VDC to the green power connectors on the controller board. When using the ArduinoPlasma example, the maximum current drawn is about 150 mA.

Once the WPI439 is connected to your computer, you can start the IDE. Under Tools → Board, select the "Arduino Duemilanova or Diecimila".

Import the Colorduino library, open the ColorduinoPlasma example sketch... and start enjoying your display.
The Colorduino library, ColorduinoPlasma example and 2 other small examples are available from our website.

*** CODE BEGIN ***

```
ColorduinoPlasma - Plasma demo using Colorduino Library for Arduino
 Copyright (c) 2011 Sam C. Lin lincomatic@hotmail.com ALL RIGHTS RESERVED

 based on  Color cycling plasma
  Version 0.1 - 8 July 2009
  Copyright (c) 2009 Ben Combee.  All right reserved.
  Copyright (c) 2009 Ken Corey.  All right reserved.
  Copyright (c) 2008 Windell H. Oskay.  All right reserved.
  Copyright (c) 2011 Sam C. Lin All Rights Reserved

 This demo is free software; you can redistribute it and/or
 modify it under the terms of the GNU Lesser General Public
 License as published by the Free Software Foundation; either
 version 2.1 of the License, or (at your option) any later version.

 This demo is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 Lesser General Public License for more details.

 You should have received a copy of the GNU Lesser General Public
 License along with this library; if not, write to the Free Software
 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/

#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);
}

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();
}
*** CODE END ***
```


whadda.com