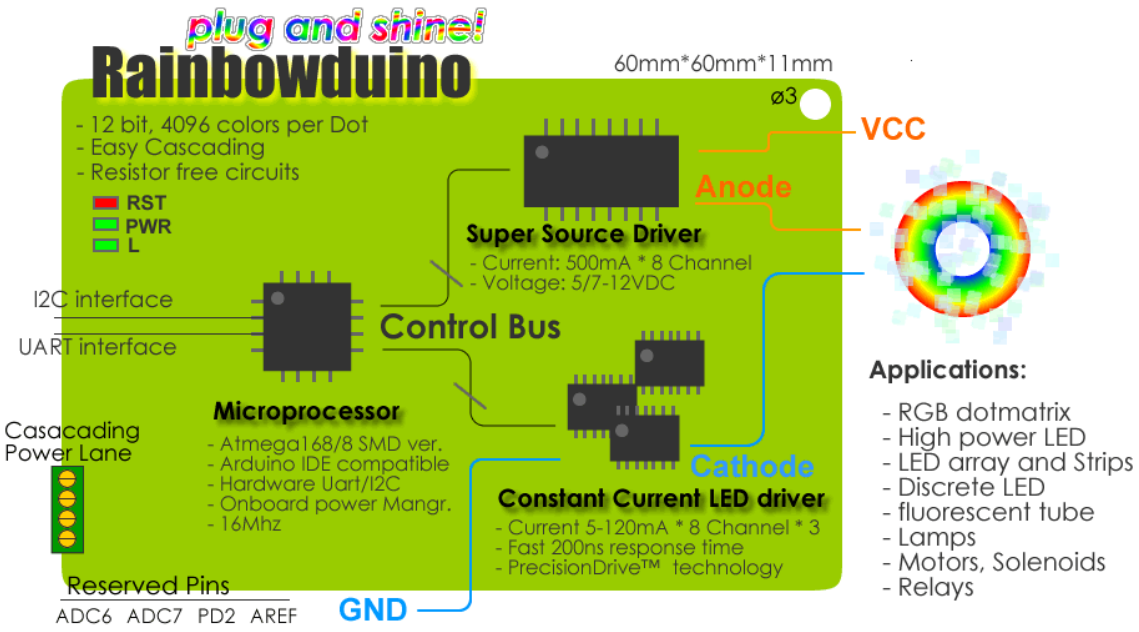
Rainbowduino

LED controller manual and datasheet v1.1

Hardware: Albert Miao
Software: Freezing Xie, Albert Miao
Graphic: Xiang Fan
Documentation: Eric Pan

Index

Overview



Rainbowduino is an Arduino compatible controller board with professional LED driving capacity. By including various user friendly features from hardware setup to programming, it makes LED applications easier and more reliable!

- Arduino compatible
- Open source library and community contributed examples
- No external circuit required, plug and shine!
- 24 constant current channels of 120mA each
- 8 super source driver channel of 500mA each
- Wide output voltage adaption from 5V-12VDC
- Dedicated GPIO and ADC
- Hardware Uart and I2C communication
- Easy cascading
- Small form and light weight

License

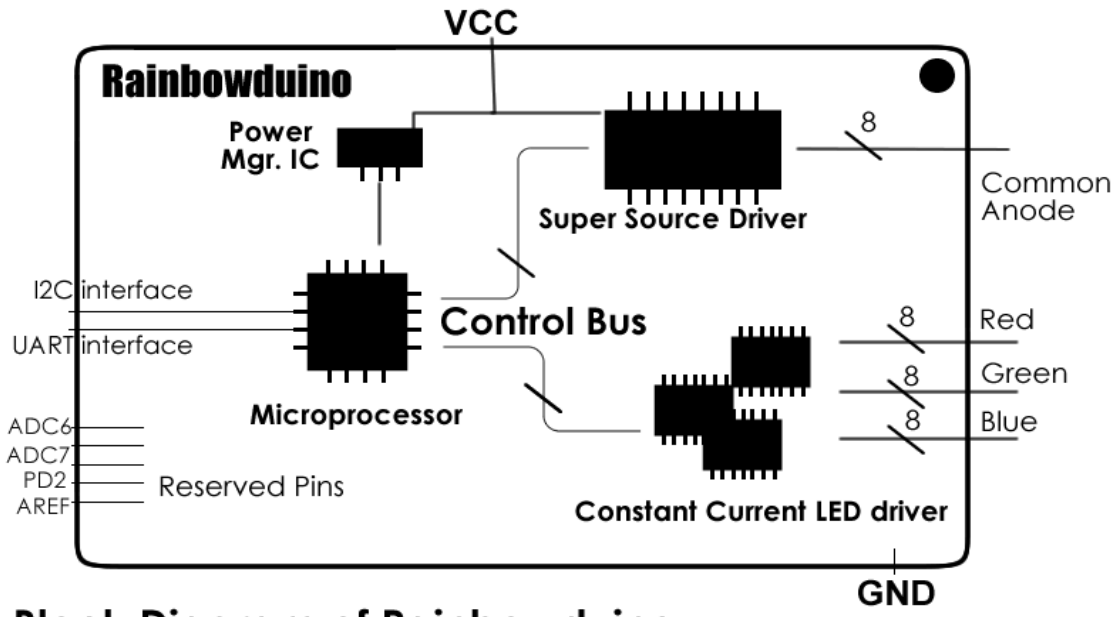
Specifications

Microprocessor	Atmega8/168
PCB size	60mm*60mm*1.6mm
Indicators	Reset, Power, Pin13 LED
Power supply	5 or 7-12 VDC (9 VDC recommended)
Power Connector	2pin JST/ Terminal Blocks/ 3mm DC jacks
Casacading Power Connector	Teminal Blocks
Program interface	Uart/ISP
LED dotmatrix sockets	32
Expansion socket	2.54mm bended pinheader pair
Program interface	Uart/ISP
Communication Protocals	I2C, Uart
ADC input	Dedicated 2 channel 10bit resolution
Outline Dimension	60mm*60mm*10mm (2.36inch * 2.36inch * 0.39inch)
RHOS	Yes

Electrical Characterstics

Specification	Min	Typ	Max	Unit
Input voltage	5	9	12	VDC
Global Current Consumption		600	4000	mA
Constant Current Channels (Cathode)			24	
Constant Current per channel (Cathode)		20	120	mA
Source Driver Current per channel (Common-Anode)			500	mA
Source Driver Voltage per channel (Common-Anode)		9	12	VDC
Source Driver Channels (Common-Anode)			8	
Drive LED count			192	
Circuit Response Time	10			ns
RGB LED matrix color resolution per dot			4096	
Uart Baud Rate			115200	bps

System Block Diagram



Block Diagram of Rainbowduino

Microprocessor - Atmega 8/168/328

Atmega 8/168/328 has enough resources to generate 4096 color for 64 dots while providing complete I2C and Uart communication. More importantly they are the most popular MCU among open source hardware community, making it compatible to Arduino IDE and the vast knowledge pool.

>>Datasheet: http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

Super Source Driver - M54564P

M54564P is an 8 circuit output-sourcing Darlington transistor arrays, widely used with proven performance. The most critical feature meeting our requirement is its fast turn-off time of 4.3ms which guarantee a vivid rendering.

>>Datasheet:
http://www.mitsubishichips.com/Global/content/product/power/transistorarray/tarray/tarray/m54564fp_e.pdf

Constant Current LED driver – MBI5168

MBI5168 is designed for LED display applications, with PrecisionDrive™ technology to enhance its output characteristics. MBI5168 contains a serial buffer and data latches, which convert serial input data into parallel output format, eight regulated current ports are designed to provide uniform and constant current sinks for driving LEDs within a large range of Vf variations.

>>Datasheet:
http://www.mblock.com.tw/en_download_file/datasheet/MBI5168%20Datasheet%20VA.02-English.pdf

Hardware Installation

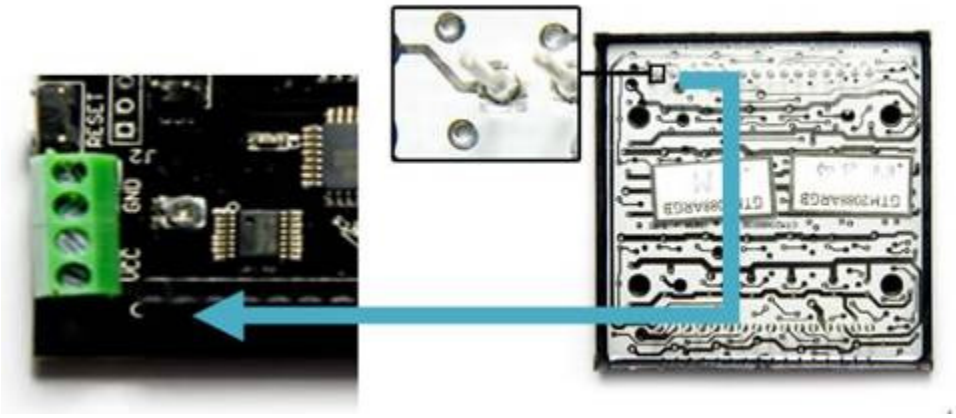
Standalone

The basic setup is a Rainbowduino equiped with RGB dotmarix, all parameters and programs are preset. You just need plug everything easily out of box and it will shine!

Step 1: Get everything ready

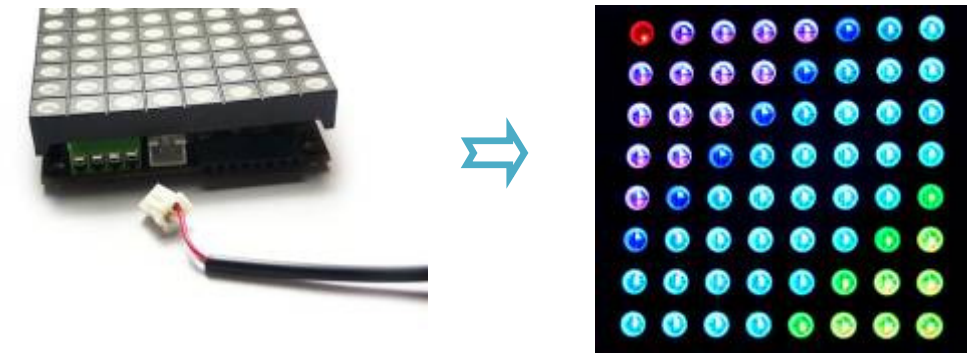
Rainbowduino *1
Compatible RGB dotmatrix *1
Compatible power supply *1
-- Then let's get started!

Step 2: Plug the dot matrix



Find the **Squre pin** of the RGB dotmatrix, plug it to **the Circle marked pin header** on rainbowduino**.** Mistaken orientation will not damage anything, thought ☺

Step 3: Connect Rainbowduino to power



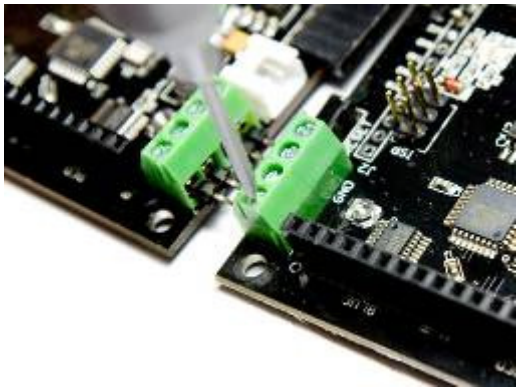
When powerred up, Rainbowduino will output default color pattern instantly. Then you could move on the next actions.

Step Later. Program Rainbowduino

. ➔See programming section.

Casacading

Rainbowduino is designed for easy casacading. After physically connected, power is passed on, and you may control the chain by I2C. Please note that each Rainbowduino must be assigned for a uniqe address for I2C communication.



Some Dual male pin header would be handy, while there are many alternatives. After you fasten the connection, two rainbowduinos are firmly connected.

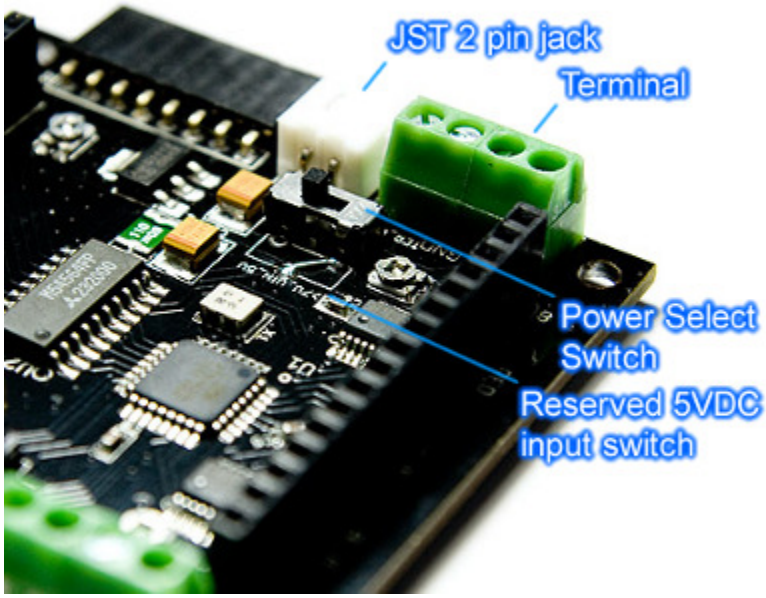
3. Note:

Power Supply Compatability

You may choose various power supplies between 5.5V to 9V for its supply, however the best balance is between 6V-6.5V where the color is best and the board does not give out much heat.

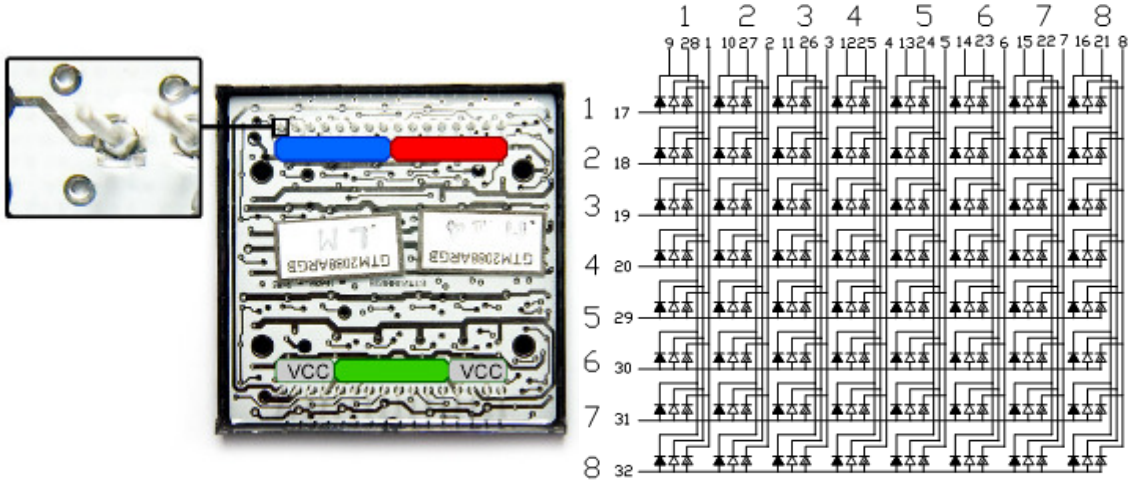
There are two types of connectors available on board, which could be selected by the VIN switch: JST 2pin and 3.0mm Terminal. For normal power adapters you could use a converter for easy installation. Or you might want to adopt power supplies by wire or jumper wire, then terminal would be a convenient way.

Also, we have reserved a 5VDC switch place, incase you would want Rainbowduino to work under 5VDC. The performance of Super bright RGB dotmatrix might be fully driven, but good enough if you want lid other LED devices. Please **NOTE SERIOUSLY**: If you supply rainbowduino with power over 7VDC, and select 5V channel, Rainbowduino will be fried very fast. That's why we decided to reserve the switch instead of defaultly provided.



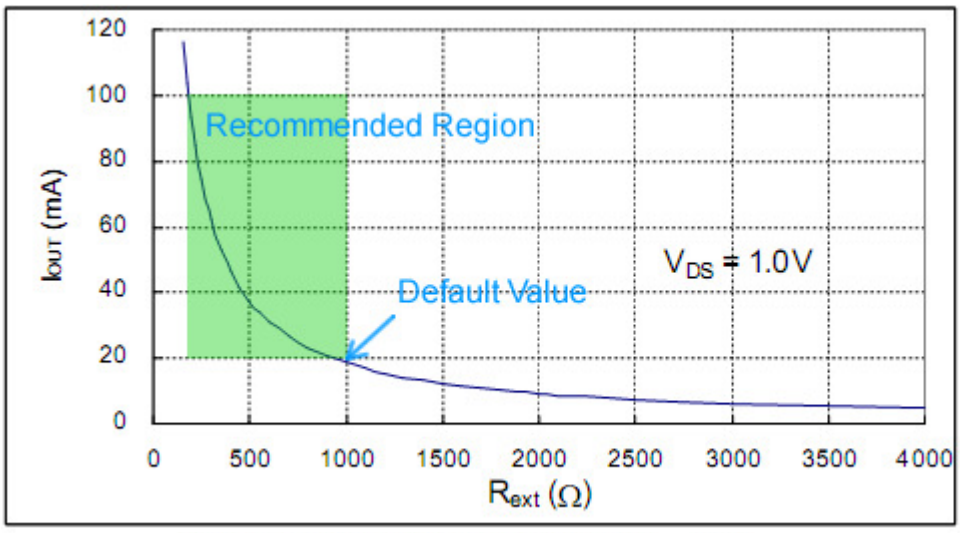
LED devices Compatability

Before direct plug into the female pinheaders, please verify if the RGB dot matrix are proven compatible. The concern is mainly on the pin out, where same color LEDs are in culster, here we attach the scheme and photo demostration. The color sequence might change, since the controlling logic are open source and easily reprogrammable.

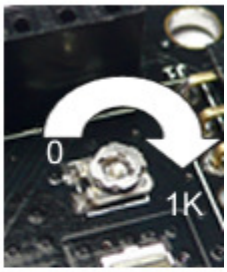


The power of Rainbowduino is well beyond driving RGB dotmatrix. With 192 output count, and up to 120mA constant current capacity, you may easily populate massive LED setups.

The output current of each channel (IOUT) is set by an external resistor, R_{ext}. The relationship between Iout and Rext is shown in the following figure. Please refer to MBI5168 datasheet for more details. Adjusting the 1k Potienmeter clockwise to reduce the output current (defautly minimal 20mA for RGB dotmatrix), rotating counter-clockwise to increase the output current. The potienmeters are single circle, please NOTE that strong force will break it into unlimit rotatable, then you would need a multimeter to adjust. :)



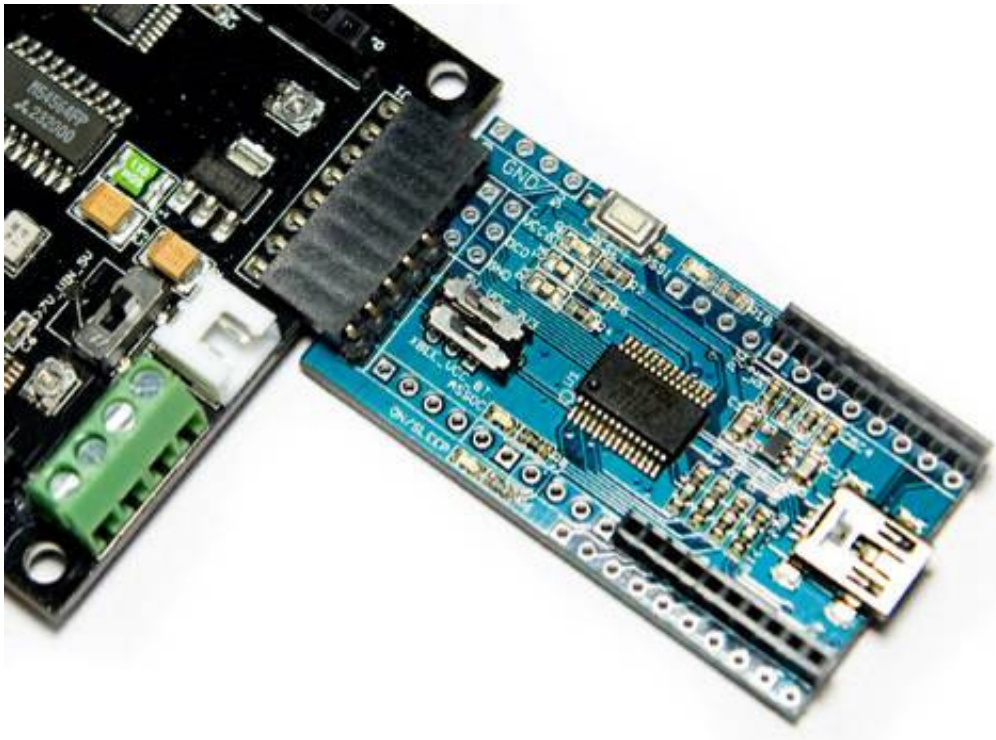
Resistance of the external resistor, R_{ext}, in Ω

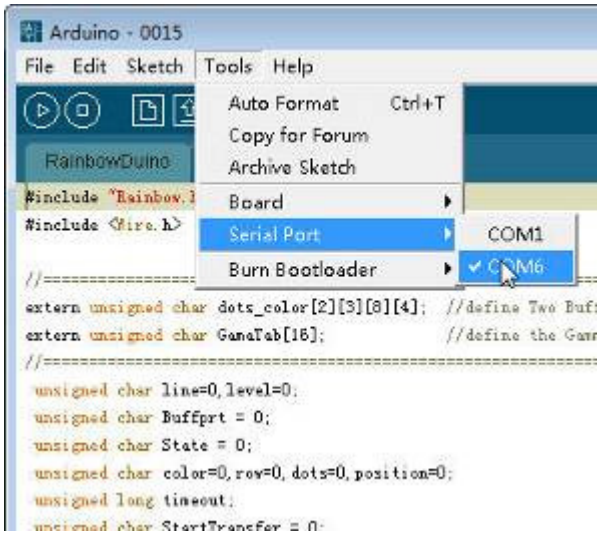


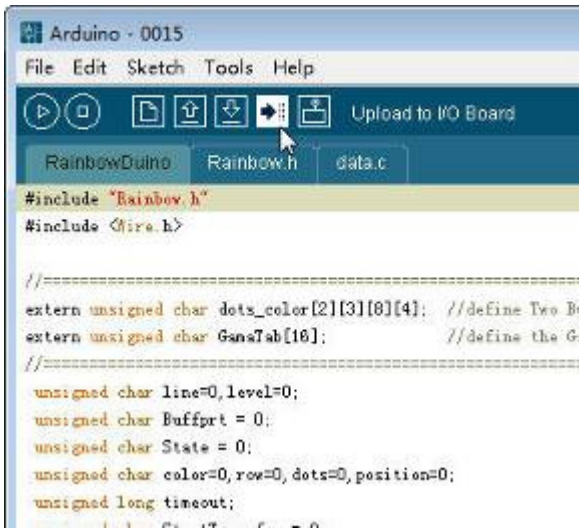
Software Installation

Programming Rainbowduino

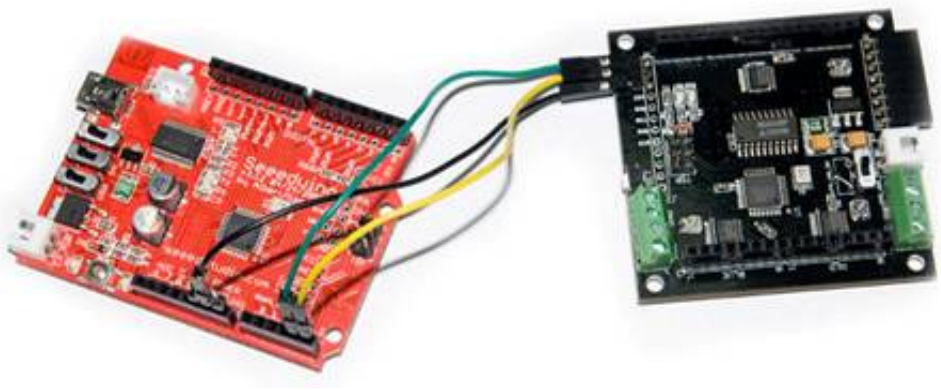
Though you can control Rainbowduino by I2C directly with default sketch, there is obvious need for rewriting the program. To do this, you may use any UART TTL adapter, e,g. UartSB v2.1, USB-Uart cable, Serial-TTL cable… or even a Seeeduino it self. Just make sure you have the right pins connected for a succesful uploading.







Communicating with Rainbowduino



I2C bus

The default communication setup is by I2C bus. After programming different address for individual Rainbowduinos, you are able to control indivisually on the whole chain. Please note that the I2C bus is not capable of high speed data transferring, dynamic data transfer would be archived differently.

Serial Port

Aside from I2C bus, we reserved the Uart pins on each Rainbowduino, you are able to communicate with every Rainbowduino individually by Serial port after their casacading. This will greatly increase the data transfer rate up to 115200Kbps.

General IO and ADC pins

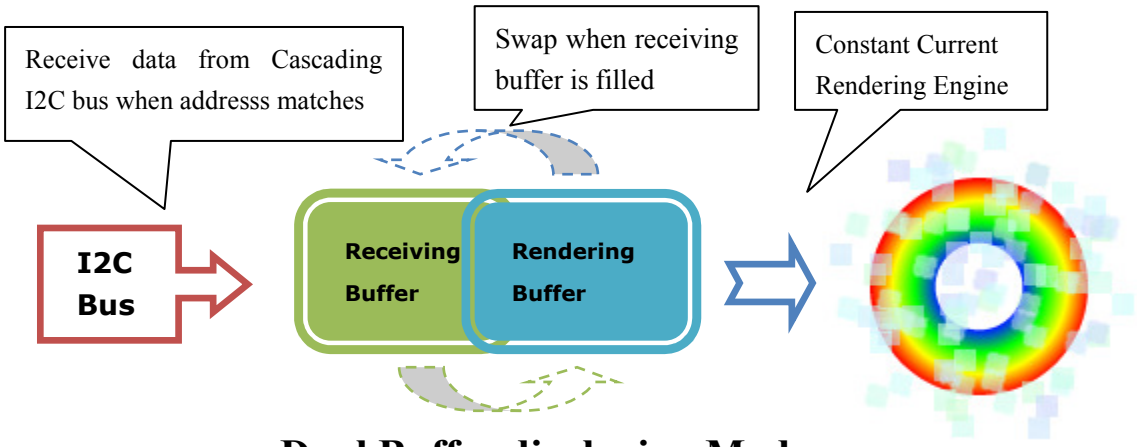
ADC6, ADC7 and PD2 are reserved for Rainbowduino to interact with additional sensors, modules, or switches to form various flexible projects.

Programming Ideas

Rainbowduino would have two working mode: 1) direct buffer rendering mode, 2) command mode. Both modes are host/client mode where an I2C host device is needed to coordinate all Rainbowduino clients. Of cause you could reprogram Rainbowduinos to self-initiative modes under Arduino programming.

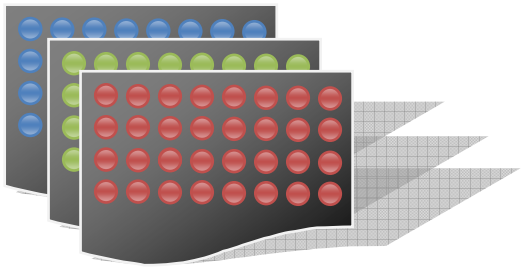
Direct Buffer Rendering Mode

In buffer rendering mode, bitmap-like raw data is received via I2C bus and rendered onto RGB Dotmatrix directly. With PingPong Buffer (swappable double buffer), one buffer is used to catch received data, while the other renders. If valid incoming data transfer is completed, the two buffers are swapped to refresh display.



Dual Buffer displaying Mode

Data structure



For example, dotmatrix is consisted of 8*8 RGB Leds, represented by 3 arrays (Red, Green, Blue). Each array is 4*8, while one byte represents two adjacing dots. You may setup the data structure like:

unsigned char dots_color[2][3][8][4]

Host API reference:

Programming on the host side is basically preparing the data and transfer to Rainbowduino via I2C or Uart. We have prepared some sample Arduino API you may start with like below.

```
//------------------------------------------------------------
//Name:SetDots
//Function:    change one dot data of    the master's buff
//------------------------------------------------------------
//Name:SentNum
//Function: Sent a Numeber to RainbowDuino to show
//Parameter:      ADD: RainbowDuino IIC Address
//                    Num: the Num want to show
//Other: the Number character color definet by the Global color Variables which set up by SetRGB() function
//------------------------------------------------------------
//Name:SentChar
//Function: Sent a    English Latter character to RainbowDuino to show
//Parameter:      ADD: RainbowDuino IIC Address
//                    Chr: the Latter want to show
//Other: the Latter character color definet by the Global color Variables which set up by SetRGB() function
//------------------------------------------------------------
//Name:Shift_Num_Right (Left)
//Function: Sent a Numeber with shift to RainbowDuino to show
//Parameter:      ADD: RainbowDuino IIC Address
//                    Num: the Num want to show
//                    Shift: Shift right bits(0-8)
//Other: the Number character color definet by the Global color Variables which set up by SetRGB() function
 //------------------------------------------------------------
//Name: Shift_Char_Right(Right)
//Function: Sent a Latter character with shift to RainbowDuino to show
//Parameter:      ADD: RainbowDuino IIC Address
//                    Chr: the Latter want to show
//                    Shift: Shift right bits (0-8)
//Other: the Latter character color definet by the Global color Variables which set up by SetRGB() function
//------------------------------------------------------------
//Name:FillBuff
//Function: Fill the master's buff with color data
//Parameter: Re, Gr, Bl    : RGB color level data
```

***Please visit our product page for the latest API samples.** Rainbowduino has open source structure, we have finished the preliminary coding for basic functionality. A library will be integrated and published later.

Command Mode

In command mode, host send out commands to manipulate client Rainbowduinos to team up a coordinated show. This is done by a concise and extensible command protocal:

["R", Command, Shift-R, G-B, Index]

"R" – Header of an ASCII "R" to determine if this is a Rainbowduino command packet
Command – Command type, 0x01=show image, 0x02=show character, 0x03= show color, the functionality will be extended over time, and customizable.
Please refer to following table for the notation in different command mode.

	Show Image	Show Character	Show Color	Customized
"R"	Header	Header	Header	Header
Command	0x01	0x02	0x03	0x04 or beyond
Shift	Shift left or right	Shift left or right	N/A	Customizable
R	N/A	Red value	Red value	Customizable
G	N/A	Green value	Green value	Customizable
B	N/A	Blue value	Blue value	Customizable
Index	Preset pic index	ASCII Character	N/A	Customizable

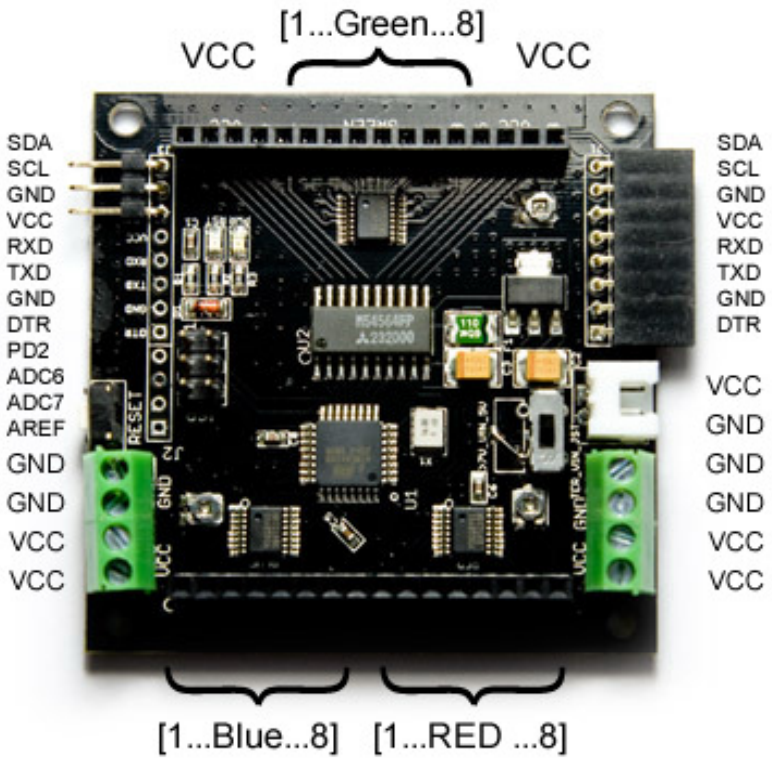
Host API reference:

```
//------------------------------------------------------------------------
//Name:ShowColor
//function: Send a conmand to Rainbowduino for showing a color
//parameter: Address: rainbowduino IIC address
//                    red,green,blue:    the color RGB
//------------------------------------------------------------------------
void ShowColor(int Address,unsigned char red , unsigned char green, unsigned char blue)

//------------------------------------------------------------------------
//Name:ShowImage
//function: Send a conmand to Rainbowduino for showing a picture which was pre-set in Rainbowduino Flash
//parameter: Address: rainbowduino IIC address
//                    number:    the pre-set picture position
//                    shift: the picture    shift bit for display
//------------------------------------------------------------------------
void ShowImage(int Address, unsigned char number,unsigned char shift)

//------------------------------------------------------------------------
//Name:ShowColor
//function: Send a conmand to Rainbowduino for showing a color
//parameter: Address: rainbowduino IIC address
```

```
//                    red,green,blue:   the color RGB
//                    shift: the picture   shift bit for display
//                    ASCII:the char or Number want to show
//-------------------------------------------------------------------------
```

void ShowChar(int Address,unsigned char ASCII,unsigned char red, unsigned char blue ,unsigned char green,unsigned char shift)

Pinout



Rev.	Descriptions	Release date
V1.1	Added Command mode program and pinout	2009/0623
V1.0 Beta	Added more detailed illstrations and photos	2009/0612
V0.8 Beta	First public version	2009/06/11
V0.8 Alpha	Initial design	2008/11/06

More info and projects availabe at:
http://www.seeedstudio.com/wiki/index.php?title=Rainbowduino
http://rainbowduino.seeedstudio.com