Grove - Chainable RGB LED

Grove - Chainable RGB LED 是基于 P9813 芯片的，它是全彩 LED 灯的驱动器。 它提供 3 个恒流的驱动器以及 256 个灰度调制的输出信号。 它使用两线传输（数据和时钟）能够与 MCU 进行通信。 可以使用这种两线传输的功能实现 **Grove - Chainable RGB LED** 模块的多个连接。 内置的时钟传输能够增大传输距离。 该 Grove 模块能够适用于任何使用彩色 LED 灯项目。

版本信息

版本调整	描述	发布日期	购买方式
v1	初次公开发布（测试版）	2011年5月5日	无
v2	用 P9813S14 替换 P9813S16，并将 Grove 连接器从垂直更改为水平	2016年4月19日	马上购买

产品特性

- 工作电压：5V
- 工作电流：20mA
- 通讯协议：串口

!!!Tip 关于Grove模块的更多细节请参考 Grove System

Platforms Supported

使用方式

使用 Arduino

当您获得 Grove - Chainble RGB LED 时，您可能会考虑如何点亮它。 现在我们将向您展示这个演示：使 RGB 灯里的颜色循环的发光。

要完成此演示，您可以使用一个或多个 Grove - Chainable RGB LED。 请注意，一个Grove - Chainable RGB LED 的 **IN** 接口应连接到 Grove - Base Shield的 **D7 / D8**，其 **OUT** 接口连接到另一个 Grove - Chainable RGB LED 的 **IN** 接口，这样可以连接更多的 LED。

- 下载 Chainable LED 库 并将其安装到 Arduino 库。 如果您是第一次安装 Arduino 库文件，请点击 这里 查看库文件的安装方法。
- 通过路径打开示例 CycleThroughColors：**File**（文件）**-> Examples**（示例）**-> ChainableLED_master** 并将其上传到 Seeeduino。

```
    /*
     * Example of using the ChainableRGB library for controlling a Grove RGB.
     * This code cycles through all the colors in an uniform way. This is
accomplished using a HSB color space.
     */
    #include <ChainableLED.h>

    #define NUM_LEDS  5

    ChainableLED leds(7, 8, NUM_LEDS);

    void setup()
    {
    }

    float hue = 0.0;
    boolean up = true;

    void loop()
    {
      for (byte i=0; i<NUM_LEDS; i++)
        leds.setColorHSB(i, hue, 1.0, 0.5);

      delay(50);

      if (up)
        hue+= 0.025;
      else
        hue-= 0.025;

      if (hue>=1.0 && up)
        up = false;
```

```
        else if (hue<=0.0 && !up)
          up = true;
      }
```

您可以看到这个情况：两个LED的颜色将一致地渐变。

扩展设计： 基于Chainable LED 库，我们设计了这个演示：RGB 灯的颜色随 Grove - temperature 测量的温度值而发生变化。 当温度为25至32℃时，RGB 颜色从绿色变为红色。测试代码如下所示。

```
    // demo of temperature -> rgbLED
    // temperature form 25 - 32, rgbLed from green -> red
    // Grove-temperature plu to A0
    // LED plug to D7,D8

    #include <Streaming.h>
    #include <ChainableLED.h>

    #define TEMPUP 32
    #define TEMPDOWN 25

    ChainableLED leds(7, 8, 1); // connect to pin7 and pin8 , one led

    int getAnalog() // get value from A0
    {
        int sum = 0;
        for(int i=0; i<32; i++)
        {
            sum += analogRead(A0);
        }

        return sum>>5;
    }

    float getTemp() // get temperature
    {
        float temperature = 0.0;
        float resistance = 0.0;
        int B = 3975; //B value of the thermistor

        int a = getAnalog();

        resistance = (float)(1023-a)*10000/a; //get the resistance of the sensor;
        temperature = 1/(log(resistance/10000)/B+1/298.15)-273.15; //convert to
temperature via datasheet ;
        return temperature;
    }

    void ledLight(int dta) // light led
    {

        dta = dta/4; // 0 - 255
```

```
        int colorR = dta;
        int colorG = 255-dta;
        int colorB = 0;

        leds.setColorRGB(0, colorR, colorG, colorB);
    }

    void setup()
    {
        Serial.begin(38400);
        cout << "hello world !" << endl;
    }

    void loop()
    {
        float temp = getTemp();
        int nTemp = temp*100;

        nTemp = nTemp > TEMPUP*100 ? TEMPUP*100 : (nTemp < TEMPDOWN*100 ?
TEMPDOWN*100 : nTemp);
        nTemp = map(nTemp, TEMPDOWN*100, TEMPUP*100, 0, 1023);
        ledLight(nTemp);
        delay(100);
    }
```

使用 Raspberry Pi

1.你应该有一个 raspberry pi 和一个 grovepi 或 grovepi +。

2.您需要完成配置开发环境，否则遵循说明 完成配置。

3.硬件连接

- 使用 grove 连接线将传感器插入 grovepi socket **D7**。

4.跳转到演示目录：

```
    cd yourpath/GrovePi/Software/Python/
```

- 找到这行代码

```
     nano grove_chainable_rgb_led.py    # "Ctrl+x" to exit #
```

```
    import time
    import grovepi
```

```
    # Connect first LED in Chainable RGB LED chain to digital port D7
    # In: CI,DI,VCC,GND
    # Out: CO,DO,VCC,GND
    pin = 7

    # I have 10 LEDs connected in series with the first connected to the GrovePi
and the last not connected
    # First LED input socket connected to GrovePi, output socket connected to
second LED input and so on
    numleds = 1

    grovepi.pinMode(pin,"OUTPUT")
    time.sleep(1)

    # Chainable RGB LED methods
    # grovepi.storeColor(red, green, blue)
    # grovepi.chainableRgbLed_init(pin, numLeds)
    # grovepi.chainableRgbLed_test(pin, numLeds, testColor)
    # grovepi.chainableRgbLed_pattern(pin, pattern, whichLed)
    # grovepi.chainableRgbLed_modulo(pin, offset, divisor)
    # grovepi.chainableRgbLed_setLevel(pin, level, reverse)

    # test colors used in grovepi.chainableRgbLed_test()
    testColorBlack = 0   # 0b000 #000000
    testColorBlue = 1    # 0b001 #0000FF
    testColorGreen = 2   # 0b010 #00FF00
    testColorCyan = 3    # 0b011 #00FFFF
    testColorRed = 4     # 0b100 #FF0000
    testColorMagenta = 5 # 0b101 #FF00FF
    testColorYellow = 6  # 0b110 #FFFF00
    testColorWhite = 7   # 0b111 #FFFFFF

    # patterns used in grovepi.chainableRgbLed_pattern()
    thisLedOnly = 0
    allLedsExceptThis = 1
    thisLedAndInwards = 2
    thisLedAndOutwards = 3

    try:

        print "Test 1) Initialise"

        # init chain of leds
        grovepi.chainableRgbLed_init(pin, numleds)
        time.sleep(.5)

        # change color to green
        grovepi.storeColor(0,255,0)
        time.sleep(.5)

        # set led 1 to green
        grovepi.chainableRgbLed_pattern(pin, thisLedOnly, 0)
        time.sleep(.5)
```

```
        # change color to red
        grovepi.storeColor(255,0,0)
        time.sleep(.5)

        # set led 10 to red
        grovepi.chainableRgbLed_pattern(pin, thisLedOnly, 9)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 2a) Test Patterns - black"

        # test pattern 0 - black (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(1)


        print "Test 2b) Test Patterns - blue"

        # test pattern 1 blue
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlue)
        time.sleep(1)


        print "Test 2c) Test Patterns - green"

        # test pattern 2 green
        grovepi.chainableRgbLed_test(pin, numleds, testColorGreen)
        time.sleep(1)


        print "Test 2d) Test Patterns - cyan"

        # test pattern 3 cyan
        grovepi.chainableRgbLed_test(pin, numleds, testColorCyan)
        time.sleep(1)


        print "Test 2e) Test Patterns - red"

        # test pattern 4 red
        grovepi.chainableRgbLed_test(pin, numleds, testColorRed)
        time.sleep(1)


        print "Test 2f) Test Patterns - magenta"

        # test pattern 5 magenta
```

```
        grovepi.chainableRgbLed_test(pin, numleds, testColorMagenta)
        time.sleep(1)


        print "Test 2g) Test Patterns - yellow"

        # test pattern 6 yellow
        grovepi.chainableRgbLed_test(pin, numleds, testColorYellow)
        time.sleep(1)


        print "Test 2h) Test Patterns - white"

        # test pattern 7 white
        grovepi.chainableRgbLed_test(pin, numleds, testColorWhite)
        time.sleep(1)


        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 3a) Set using pattern - this led only"

        # change color to red
        grovepi.storeColor(255,0,0)
        time.sleep(.5)

        # set led 3 to red
        grovepi.chainableRgbLed_pattern(pin, thisLedOnly, 2)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 3b) Set using pattern - all leds except this"

        # change color to blue
        grovepi.storeColor(0,0,255)
        time.sleep(.5)

        # set all leds except for 3 to blue
        grovepi.chainableRgbLed_pattern(pin, allLedsExceptThis, 3)
        time.sleep(.5)
```

```
        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 3c) Set using pattern - this led and inwards"

        # change color to green
        grovepi.storeColor(0,255,0)
        time.sleep(.5)

        # set leds 1-3 to green
        grovepi.chainableRgbLed_pattern(pin, thisLedAndInwards, 2)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 3d) Set using pattern - this led and outwards"

        # change color to green
        grovepi.storeColor(0,255,0)
        time.sleep(.5)

        # set leds 7-10 to green
        grovepi.chainableRgbLed_pattern(pin, thisLedAndOutwards, 6)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 4a) Set using modulo - all leds"

        # change color to black (fully off)
        grovepi.storeColor(0,0,0)
        time.sleep(.5)

        # set all leds black
        # offset 0 means start at first led
        # divisor 1 means every led
        grovepi.chainableRgbLed_modulo(pin, 0, 1)
```

```
        time.sleep(.5)

        # change color to white (fully on)
        grovepi.storeColor(255,255,255)
        time.sleep(.5)

        # set all leds white
        grovepi.chainableRgbLed_modulo(pin, 0, 1)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 4b) Set using modulo - every 2"

        # change color to red
        grovepi.storeColor(255,0,0)
        time.sleep(.5)

        # set every 2nd led to red
        grovepi.chainableRgbLed_modulo(pin, 0, 2)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)


        print "Test 4c) Set using modulo - every 2, offset 1"

        # change color to green
        grovepi.storeColor(0,255,0)
        time.sleep(.5)

        # set every 2nd led to green, offset 1
        grovepi.chainableRgbLed_modulo(pin, 1, 2)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 4d) Set using modulo - every 3, offset 0"

        # change color to red
        grovepi.storeColor(255,0,0)
```

```
        time.sleep(.5)

        # set every 3nd led to red
        grovepi.chainableRgbLed_modulo(pin, 0, 3)
        time.sleep(.5)

        # change color to green
        grovepi.storeColor(0,255,0)
        time.sleep(.5)

        # set every 3nd led to green, offset 1
        grovepi.chainableRgbLed_modulo(pin, 1, 3)
        time.sleep(.5)

        # change color to blue
        grovepi.storeColor(0,0,255)
        time.sleep(.5)

        # set every 3nd led to blue, offset 2
        grovepi.chainableRgbLed_modulo(pin, 2, 3)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 4e) Set using modulo - every 3, offset 1"

        # change color to yellow
        grovepi.storeColor(255,255,0)
        time.sleep(.5)

        # set every 4nd led to yellow
        grovepi.chainableRgbLed_modulo(pin, 1, 3)
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)


        print "Test 4f) Set using modulo - every 3, offset 2"

        # change color to magenta
        grovepi.storeColor(255,0,255)
        time.sleep(.5)

        # set every 4nd led to magenta
        grovepi.chainableRgbLed_modulo(pin, 2, 3)
        time.sleep(.5)
```

```
        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 5a) Set level 6"

        # change color to green
        grovepi.storeColor(0,255,0)
        time.sleep(.5)

        # set leds 1-6 to green
        grovepi.write_i2c_block(0x04,[95,pin,6,0])
        time.sleep(.5)

        # pause so you can see what happened
        time.sleep(2)

        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        time.sleep(.5)


        print "Test 5b) Set level 7 - reverse"

        # change color to red
        grovepi.storeColor(255,0,0)
        time.sleep(.5)

        # set leds 4-10 to red
        grovepi.write_i2c_block(0x04,[95,pin,7,1])
        time.sleep(.5)


    except KeyboardInterrupt:
        # reset (all off)
        grovepi.chainableRgbLed_test(pin, numleds, testColorBlack)
        break
    except IOError:
        print "Error"
```

- 请注意，您有以下需要关注的事项：

```
    pin = 7         #setting up the output pin
    numleds = 1     #how many leds you plug
```

- 您还可以在 grovepi.py 中看到所有的方法：

```
storeColor(red, green, blue)
chainableRgbLed_init(pin, numLeds)
chainableRgbLed_test(pin, numLeds, testColor)
chainableRgbLed_pattern(pin, pattern, whichLed)
chainableRgbLed_modulo(pin, offset, divisor)
chainableRgbLed_setLevel(pin, level, reverse)
```

5.运行这个示例

```
sudo python grove_chainable_rgb_led.py
```

6.如果您的 Grove pi 没有最新的固件，请更新固件，否则此演示可能无法正常工作。

```
cd yourpath/GrovePi/Firmware
sudo ./firmware_update.sh
```

使用 Beaglebone Green

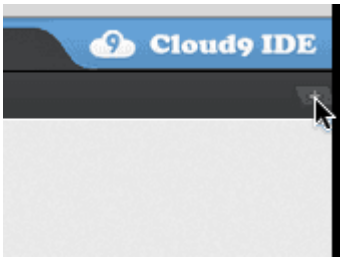
要开始编辑 BBG 上的程序，可以使用 Cloud9 IDE。

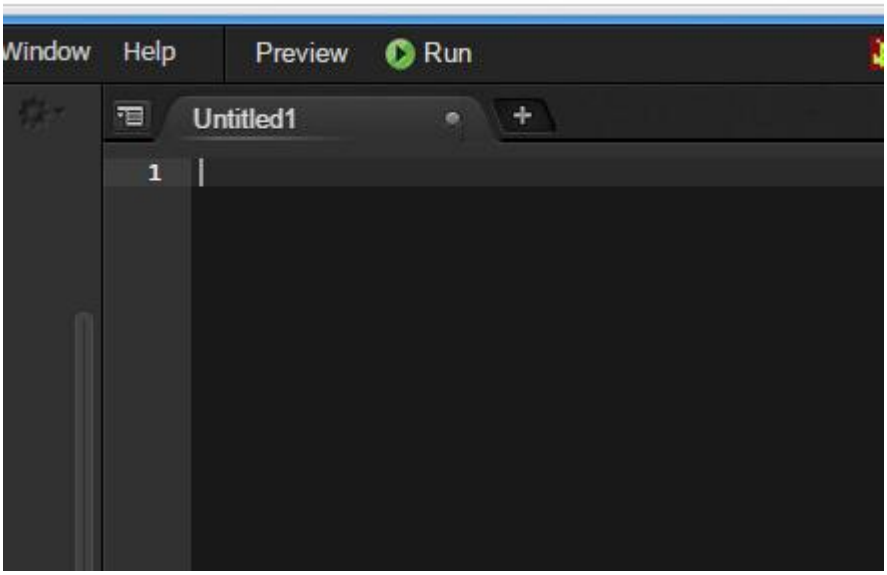
作为熟悉 Cloud9 IDE 的简单练习，一个好的开始是创建一个简单的应用程序来闪烁 BeagleBone 上的 4 个可编程的 LED 灯。

如果这是您第一次使用 Cloud9 IDE，请按照 **link**.

步骤1 将 Grove-UART 插座设置为 Grove - GPIO Socket，只需按照 链接. 。

步骤2 点击右上角的 “+” 创建一个新文件。





步骤3 将以下代码复制并粘贴到新的编辑器中

```
import time
import Adafruit_BBIO.GPIO as GPIO

CLK_PIN = "P9_22"
DATA_PIN = "P9_21"
NUMBER_OF_LEDS = 1

class ChainableLED():
    def __init__(self, clk_pin, data_pin, number_of_leds):
        self.__clk_pin = clk_pin
        self.__data_pin = data_pin
        self.__number_of_leds = number_of_leds

        GPIO.setup(self.__clk_pin, GPIO.OUT)
        GPIO.setup(self.__data_pin, GPIO.OUT)

        for i in range(self.__number_of_leds):
            self.setColorRGB(i, 0, 0, 0)

    def clk(self):
        GPIO.output(self.__clk_pin, GPIO.LOW)
        time.sleep(0.00002)
        GPIO.output(self.__clk_pin, GPIO.HIGH)
        time.sleep(0.00002)

    def sendByte(self, b):
        "Send one bit at a time, starting with the MSB"
        for i in range(8):
            # If MSB is 1, write one and clock it, else write 0 and clock
            if (b & 0x80) != 0:
                GPIO.output(self.__data_pin, GPIO.HIGH)
            else:
                GPIO.output(self.__data_pin, GPIO.LOW)
```

```
            self.clk()

            # Advance to the next bit to send
            b = b << 1

    def sendColor(self, red, green, blue):
        "Start by sending a byte with the format '1 1 /B7 /B6 /G7 /G6 /R7 /R6' "
        #prefix = B11000000
        prefix = 0xC0
        if (blue & 0x80) == 0:
            #prefix |= B00100000
            prefix |= 0x20
        if (blue & 0x40) == 0:
            #prefix |= B00010000
            prefix |= 0x10
        if (green & 0x80) == 0:
            #prefix |= B00001000
            prefix |= 0x08
        if (green & 0x40) == 0:
            #prefix |= B00000100
            prefix |= 0x04
        if (red & 0x80) == 0:
            #prefix |= B00000010
            prefix |= 0x02
        if (red & 0x40) == 0:
            #prefix |= B00000001
            prefix |= 0x01
        self.sendByte(prefix)

        # Now must send the 3 colors
        self.sendByte(blue)
        self.sendByte(green)
        self.sendByte(red)

    def setColorRGB(self, led, red, green, blue):
        # Send data frame prefix (32x '0')
        self.sendByte(0x00)
        self.sendByte(0x00)
        self.sendByte(0x00)
        self.sendByte(0x00)

        # Send color data for each one of the leds
        for i in range(self.__number_of_leds):
            '''
            if i == led:
                _led_state[i*3 + _CL_RED] = red;
                _led_state[i*3 + _CL_GREEN] = green;
                _led_state[i*3 + _CL_BLUE] = blue;
            sendColor(_led_state[i*3 + _CL_RED],
                      _led_state[i*3 + _CL_GREEN],
                      _led_state[i*3 + _CL_BLUE]);
            '''
            self.sendColor(red, green, blue)
```

```
        # Terminate data frame (32x "0")
        self.sendByte(0x00)
        self.sendByte(0x00)
        self.sendByte(0x00)
        self.sendByte(0x00)


# Note: Use P9_22(UART2_RXD) and P9_21(UART2_TXD) as GPIO.
# Connect the Grove - Chainable RGB LED to UART Grove port of Beaglebone Green.
if __name__ == "__main__":
    rgb_led = ChainableLED(CLK_PIN, DATA_PIN, NUMBER_OF_LEDS)

    while True:
        # The first parameter: NUMBER_OF_LEDS - 1; Other parameters: the RGB
values.
        rgb_led.setColorRGB(0, 255, 0, 0)
        time.sleep(2)
        rgb_led.setColorRGB(0, 0, 255, 0)
        time.sleep(2)
        rgb_led.setColorRGB(0, 0, 0, 255)
        time.sleep(2)
        rgb_led.setColorRGB(0, 0, 255, 255)
        time.sleep(2)
        rgb_led.setColorRGB(0, 255, 0, 255)
        time.sleep(2)
        rgb_led.setColorRGB(0, 255, 255, 0)
        time.sleep(2)
        rgb_led.setColorRGB(0, 255, 255, 255)
        time.sleep(2)
```

步骤4 通过单击磁盘图标保存文件，并为文件提供.py 扩展名。

步骤5 将 Grove Chainable RGB LED 连接到 BBG 上的 **Grove UART** 端口。

步骤6 运行代码。 您会发现 RGB LED 每 2 秒钟更改一次颜色。

资源下载

- **[Library]**Chainable RGB LED Library for the P9813
- **[Library]**Github repository for Chainable RGB LED Library (new)
- **[Eagle]**Chainable RGB LED eagle file V1
- **[Eagle]**Chainable RGB LED eagle file V2
- **[PDF]**Chainable RGB LED SCH file V1
- **[PDF]**Chainable RGB LED SCH file V2
- **[PDF]**Chainable RGB LED PCB file V1
- **[PDF]**Chainable RGB LED PCB file V2
- **[Datasheet]**P9813 Datasheet

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Emulators/Simulators ***category:***

Click to view products by Seeed Studio ***manufacturer:***

Other Similar products are found below :

AC244062 AC244064 SPC563M64CAL144 SPC563M64CAL176 ST7MDT2-EMU2B IM3316 IM1281B IM1275 IM1227 IM1259G IM1253B IM1253B(D) MJYS-QKJL-40/380V MJYS-QKJL-75/380V MJYD-JL-75/380V MJYD-JL-40/380V CI-B02CS01S CI-B03CS01S CI-BO3GS01S GD10PJX120L2S HEDS-9730#Q50 HEDS-9700#F50 L-MZ07 L-MZ02 L-MZ021 TXVT4G6M-S JL_MOD_FH_V1.0 MKSDSOCKET-Pinboard V1 CY3250-24X33 AC244060 7027-D-350 DS1747WP-120IND+ AC244061 S5U1C31W74T1300 S5U1C17M13T2100 S5U1C17M13T1100 J-Link ULTRA+ AFM201TI-AY2LED2 AFW121T-EVB CP2102 CE118M12 ESP32-A1S ESP32-CAM ESP-32S ADZS-ICE-1000 ADZS-ICE-2000 USB-EA-CONVZ BH-USB-100v2-ARM BH-USB-100v2D BH-USB-200