


# 点亮WS2812 LED设计文档(完结)

## 一、工具介绍

 陈柏良

电脑,STM32F103C8T6,WS2812\*3,ST-LINK.....

## 二、需求与分析

### 1任务目标

使用软件 ~~STM32CUBEMX~~和 ~~KEIL~~,,,后来改用STM32CUBEIDE, (LDE=MX+KEIL)

**目标一：**使用MCU (STM32F103C8T6) 和ws2812, 实现使用一或多个开关, 控制RGB灯光颜色以及亮灭

**目标二：**实现呼吸灯

**目标三：**实现ADC检测外部电压, 当检测的外部电压低于1V时, ws2812发绿色光; 当检测的外部电压高于1V但低于2V时, ws2812发蓝色光; 当检测的外部电压高于2V时, ws2812发红色光。

## 2 极简版原理分析

### 2.1信息传输

2.1.1 使用PWM驱动TIM+PWM+DMA驱动WS2812 (脉宽调制)

2.1.2 每经过一个像素点的传输, 信号减少24bit;

2.1.3 800K的传输速率, 为传输一个bit位的速率, 换算成时间为1.25us, 周期T1=1.25us

2.1.4 COUNT 决定脉冲周期, DUMMY 决定占空比高电平时间

2.1.5 使用一个周期内的占空比来确定接收到的是1值还是0值。根据时序波形图来确定是0还是1。

### 2.2 灯的点亮与RGB色 (GRB)

## 24bit 数据结构

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

注：高位先发，按照 GRB 的顺序发送数据。

绿色0xff0000

红色0x00ff00

蓝色0x0000ff

## 三、设计流程暨正常版原理拆解

### 1时钟配置和脉冲周期

TO 先导版 —— 在单片机中的1/0指高低电平，但WS2812 1/0指不同波型

参考阅读（[WS2812与STM32的不解之缘](#)）

#### 1.1

时钟信号选用72Mhz，时钟周期约为13.9ns（减少时长，使控制更为精确,毕竟n大更可控）

时钟周期13.89ns，脉冲周期1.25us。

$T0 \cdot N = T1$ ； $N=90$ ；即count=90；

### 1.2 1/0传输

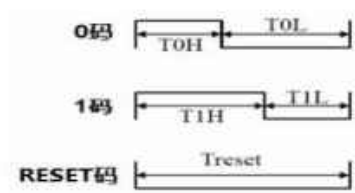
#### 1.2.1

数据传输时间

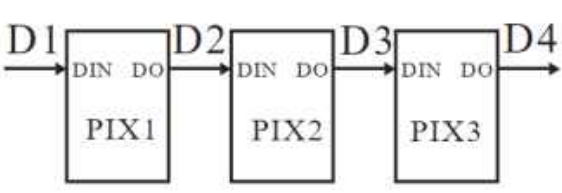
T0H	0 码, 高电平时间	220ns~380ns
T1H	1 码, 高电平时间	580ns~1μs
T0L	0 码, 低电平时间	580ns~1μs
T1L	1 码, 低电平时间	580ns~1μs
RES	帧单位, 低电平时间	280μs 以上

时序波形图

输入码型:



连接方法:



$(T_{0H})\ 25 \times 13.9 = 347.5;$

$(T_{0L})\ (90 - 25) \times 13.9 = 903.5;$

所以在单片机里高电平持续 25个时钟周期 ( $T_{0H}$ )，低电持续65个周期 ( $T_{0L}$ ) 为ws2812的0码

高电平持续 25个时钟周期( $T_{1H}$ )，低电持续65个周期 ( $T_{1L}$ ) 为ws2812的1码

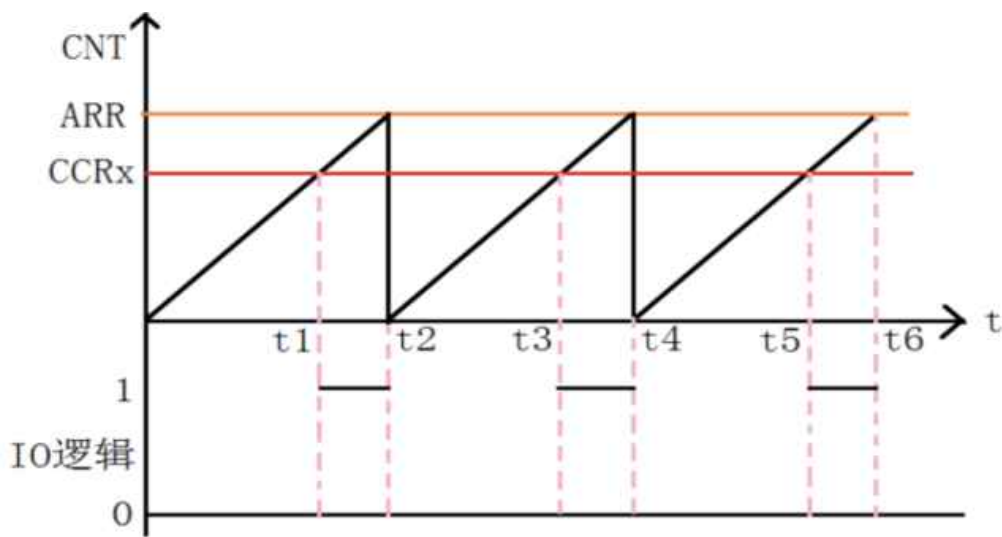
两者均需RES复位信号 低电平  $T_1 \times 240 = 300\mu s;$

即 复位信号为90\*240个时钟型号；

1.3 脉冲宽度调制(PWM- Pulse Width Modulation)

问题： 如何实现1/0之间的比例关系？

答： 1.3.1 TIM 计数PWM



当 $CNT < CCRx$ 时,输出0,当 $CNT \geq CCRx$ 时输出1。

当CNT值小于CCRx的时,IO输出低电平(0)

当CNT值大于等于CCRx的时,输出高电平(1),当CNT达到ARR值的时候,重新归零,然后重新向上计数,依次循环。

改变CCRx的值,就可以改变PWM输出的占空比,改变ARR的值,就可以改变PWM输出的频率

### 1.3.2 水桶短板效应

即可以简单的认为 CCRx为水桶的短板，ARR为桶高。

往水桶中加水后，水是0，无水为1，无水高/桶长=占空比。

两种不同规格的水桶在ws2812中一个代表1码，一个是0码

在进行CNT与CCR比较后，其结果在经过输出比较电路后，通过TIM\_CH? 输出到GPIO引脚上

## 1.4 DMA(Direct Memory Access)

### 1.4.1

DMA输出传输方向从内存到外设，内存里面是需要我们自己定义一个存储空间，比如一个数组，然后将24bit的颜色数据存到这个数组中，在配置DMA的时候这里使用定义的该数组的地址。外设就是TIM产生PWM的捕获/比较寄存器CCR。

## 四、MX配置过程以及代码编程（测试版）

### 1准备工作

//配置可以参考（[MX配置C8T6](#)）

1.打开STM32F103CUBEMX

2 选取单片机并配置时钟周期72MHZ 过程参考（[72MHZ](#)）

3 配置定时器 TIM3 , PWM Generation CH1 , DMA配置 选择TIM3\_CH1/TRIG 参考 ([时钟源、通道、T0与T1](#))

4 GRB信息存储于内存中，输出到CCR，产生pwm 参考 ([DMA \(Direct Memory Access\)](#))

5 MX generate code ，使用keil打开 编译运行-正常

## 2 开始测试

开始进行输入数字测试检测思路是否有问题 参考 ([代码测试](#))

### 2.1 小尝试

机制:

两组数组，一个是1/0，一个是占空比

准备了data作为GRB(Green,Red,Blue的数值范围0~255)

pwmData为PWM(65/25)

seed函数的作用为将data的1/0转换为pwmData中的(65/25)

用 HAL\_TIM\_PWM\_Start\_DMA 函数 来输出pwm

输入seed(0, 255, 0)，测试灯是否有效

个人在main.c 中添加的代码:

```
1  /* USER CODE BEGIN 0 */
2
3  void HAL_TIM_PWM_PulseFinshedCallback(TIM_HandleTypeDef *htim)
4  {
5      HAL_TIM_PWM_Stop_DMA(&htim3,TIM_CHANNEL_1);
6
7
8  } //中断DMA与PWM的传输
9
10 uint16_t pwmData[24];
11
12 void send (int Green,int Red,int Blue)
13 {
14     uint32_t data=(Green<<16) | (Red<<8) | Blue;
15
16     for(int i=23;i>=0;i--){
17
18         if(data&(1<<i)) pwmData[i]=65;
```

```

19         else pwmData[i]=35;
20
21
22     }
23     HAL_TIM_PWM_Start_DMA(&htim3,TIM_CHANNEL_1,(uint32_t *)pwmData,24);
24
25
26 }
27 /* USER CODE END 0 */
28 /* USER CODE BEGIN 2 */
29     send(0,255,0);
30 /* USER CODE END 2

```

再次烧录，接电源，发现正常？（少reset，失败的亮度）😄

数值测试工作结束，本身配置无问题，正片开始:

## 2.2 控制RGB颜色以及开关亮灭

封装想法:把每一个灯记录序号，并赋予其data值，让每个灯有自己的光。

灯的颜色函数

```

1 void Set_LED (int LEDnum,int Green,int Red,int Blue)
2 {
3
4     LED_Data[LEDnum][0] = LEDnum;
5     LED_Data[LEDnum][1] = Green;
6     LED_Data[LEDnum][2] = Red;
7     LED_Data[LEDnum][3] = Blue;
8
9 }
10

```

将灯的数据发送到并且转变为pwm

```

1 uint16_t pwmData[24*LED_MAX+240];
2 void WS2812_Send (void)
3 {
4     uint32_t indx = 0;
5     uint32_t color;
6
7     for (int i=0;i<LED_MAX;i++)

```

```

8      {
9          color = ((LED_Data[i][1]<<16)|(LED_Data[i][2]<<8)|
10             (LED_Data[i][3]));
11      }
12      for(int i=23;i>=0;i--)
13      {
14          if(color&(1<<i))
15          {
16              pwmData[indx] = 65;
17          }
18          else pwmData[indx]= 25;
19      }
20      indx++;
21  }
22  }
23  }

```

开始输出pwm     //可以直接放在WS2812\_Send函数末尾，因为转变与亮灯是同一个步骤

```

1      HAL_TIM_PWM_Start_DMA(&htim3,TIM_CHANNEL_1,(uint32_t *)pwmData,indx);

```

在这里的pwm中会发现有一个+240（手动跳转 三 1.2）

```

1  uint16_t pwmData[24*LED_MAX+240];

```

这是因为每一次发送完一整个的pwm后，还需要有起码280us的reset码

所以在每次传输pwm的尾部，提前存240的空间用来放占空比0；

```

1  for ( int i=0;i<240;i++)
2      {
3          pwmData[indx]=0;
4          indx++;
5      }

```

把这几个模块放到一起,完成通电发光的效果函数，在mian函数中加上 **Set\_LED**和**WS2812\_Send**函数

```

1  /* USER CODE BEGIN 0 */
2  #define LED_MAX 8
3      uint8_t LED_Data[LED_MAX][4];
4      uint8_t LED_Mod[LED_MAX][4];
5  void HAL_TIM_PWM_PulseFinshedCallback(TIM_HandleTypeDef *htim)
6  {
7      HAL_TIM_PWM_Stop_DMA(&htim3,TIM_CHANNEL_1);
8  }
9      void Set_LED (int LEDnum,int Green,int Red,int Blue)
10     {
11
12         LED_Data[LEDnum][0] = LEDnum;
13         LED_Data[LEDnum][1] = Green;
14         LED_Data[LEDnum][2] = Red;
15         LED_Data[LEDnum][3] = Blue;
16
17     }
18  uint16_t pwmData[24*LED_MAX+240];
19  void WS2812_Send (void)
20  {
21      uint32_t indx = 0;
22      uint32_t color;
23  /* for (int i=0;i<240;i++)
24      {
25          pwmData[index] = 0;
26          index++;
27
28      }*/
29      for (int i=0;i<LED_MAX;i++)
30      {
31          color = ((LED_Data[i][1]<<16)|(LED_Data[i][2]<<8)|
32          (LED_Data[i][3]));
33
34          for(int i=23;i>=0;i--)
35          {
36              if(color&(1<<i))
37              {
38                  pwmData[indx] = 65;
39              }
40              else pwmData[indx]= 25;
41
42              indx++;
43          }
44      }
45

```



```

46         for ( int i=0;i<240;i++)
47         {
48             pwmData[indx]=0;
49             indx++;
50         }
51
52         HAL_TIM_PWM_Start_DMA(&htim3,TIM_CHANNEL_1,(uint32_t
53 *)pwmData,indx);
54     }
55     /* USER CODE END 0 */
56     /* USER CODE BEGIN 2 */
57     Set_LED(0,255,0,0);
58     Set_LED(1,0,255,0);
59     Set_LED(2,0,0,255);
60
61     WS2812_Send();
62     /* USER CODE END 2 */

```

烧录后，通电后符合预期，关于开关我是直接在中路卡上，控制DIN信号传递来实现开关的，暂时不整花活了

```

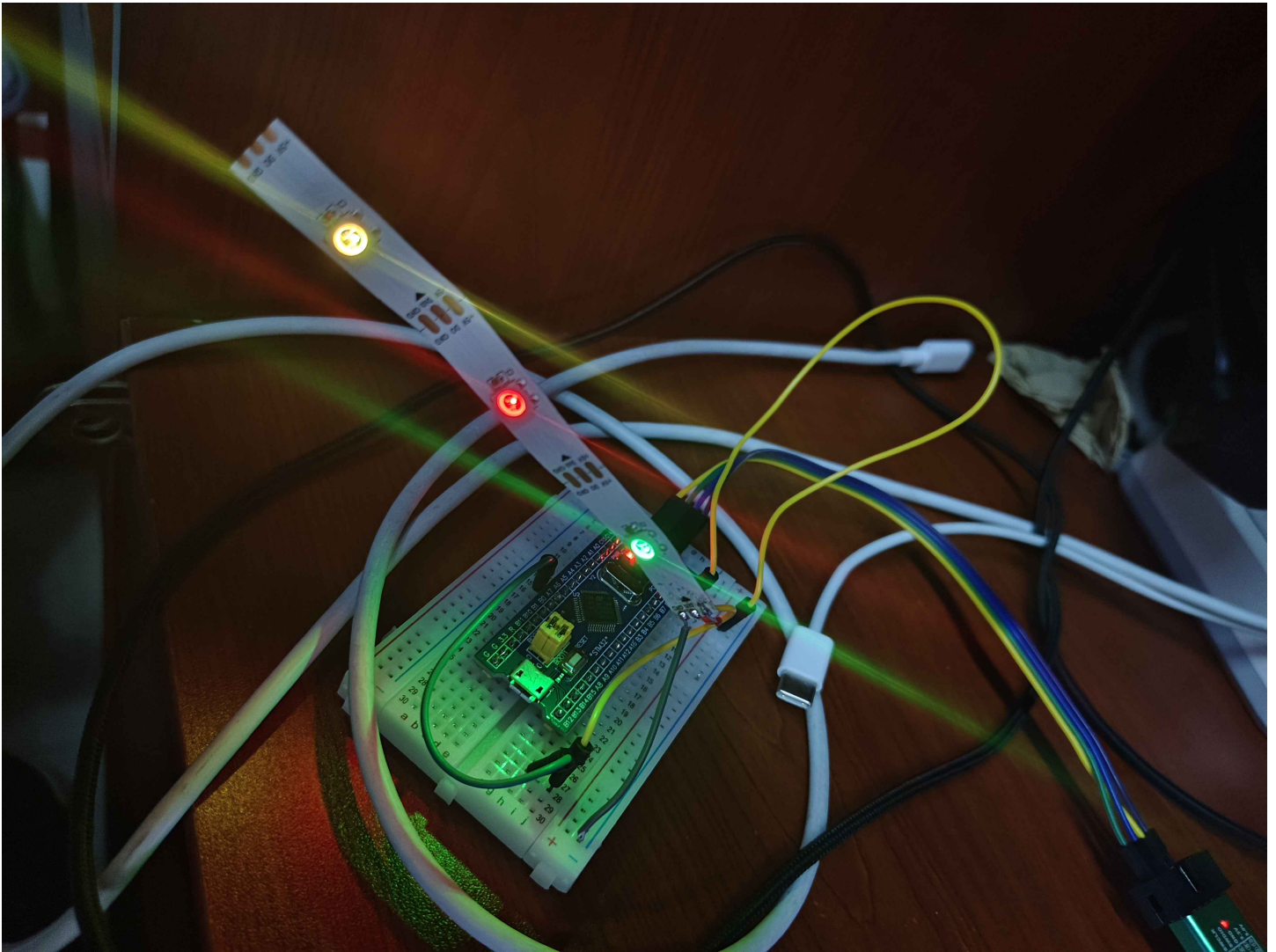
1  /* USER CODE BEGIN 2 */
2  Set_LED(0,255,0,0);
3  Set_LED(1,0,255,0);
4  Set_LED(2,255,255,255);
5
6  WS2812_Send();
7
8  /* USER CODE END 2 */
9

```

当检测到引脚被拉低后，开始WS2812\_Send();函数，发送pwm灭灯

开始测试，烧录接电

结果(照片为调低亮度版，不然太亮看不清色)



## 2.3 呼吸灯

接下来逐个在原本的点灯函数基础上修改，使其能够实现灯的亮度不断改变

通过一个亮度值来调节每一个灯的亮度，同时尽可能地实现线性亮度变化，这里需要一个亮度转换并设置函数

```
1  uint8_t LED_Mod[LED_MAX][4];
2  /**
3   * @brief 设置LED亮度
4   *
5   * 该函数调整LED的颜色亮度，基于给定的亮度等级，将亮度应用到所有的LED灯上。
6   *
```

```

7  * @param brightness 亮度等级，范围为0到50。更高的值表示更高的亮度。
8  */
9  void Set_Brightness(int brightness) {
10     // 计算亮度系数，将输入的亮度等级映射到0.0到1.0的范围内
11     float factor = (float)brightness / 50.0f;
12
13     // 遍历所有LED灯，调整其亮度
14     for(int i = 0; i < LED_MAX; i++) {
15         for(int j = 1; j < 4; j++) {
16             // 根据亮度系数调整LED颜色的强度
17             LED_Mod[i][j] = (uint8_t)(LED_Data[i][j] * factor);
18         }
19     }
20 }
21 //别忘了在开始时，#include "math.h"

```

在这里使用了LED\_Mod来取代原本的LED\_Data，所以修改原本的WS2812\_Send的部分

```

1  for (int i=0;i<LED_MAX;i++)
2      {
3          color = ((LED_Mod[i][1]<<16)|(LED_Mod[i][2]<<8)|(LED_Mod[i]
4              [3]));
5
6          // 对每个颜色位，根据值生成对应的PWM脉冲长度
7          for(int i=23;i>=0;i--)
8              {
9                  if(color&(1<<i))
10                     {
11                         pwmData[indx] = 65;
12                     }
13                     else pwmData[indx]= 25;
14                     indx++;
15             }
16     }

```

在main函数中while（1）使用 Set\_Brightness 改变亮度

```

1  /* USER CODE BEGIN 3 */
2      /* if (GPIO_PIN_RESET == HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0)) {
3          // 按下按键，关闭LED亮度。。。这个是目标1对于开关的要求，暂时注释掉
4          Set_Brightness(0);
5          WS2812_Send();
6      }

```

```

7         HAL_Delay(100);
8         */
9         // 循环改变亮度并发送数据
10        for(int i=0;i<50;i++){
11            Set_Brightness(i);
12            WS2812_Send();
13            HAL_Delay(50);
14        }
15        for(int i=0;i<50;i++){
16            Set_Brightness(45-i);
17            WS2812_Send();
18            HAL_Delay(50);
19        }
20    }
21    /* USER CODE END 3 */

```

## 2.4 任务1, 2的完整版main.c代码

```

1  /* USER CODE BEGIN Header */
2  /**
3
4      *****
5      * @file           : main.c
6      * @brief          : Main program body
7      *****
8
9      * @attention
10     *
11     * Copyright (c) 2024 STMicroelectronics.
12     * All rights reserved.
13     * This software is licensed under terms that can be found in the LICENSE file
14     * in the root directory of this software component.
15     * If no LICENSE file comes with this software, it is provided AS-IS.
16     *
17     *****
18     */
19 /* USER CODE END Header */
20 /* Includes -----
21 */
22 #include "main.h"
23 #include "dma.h"
24 #include "tim.h"
25 #include "gpio.h"

```

```

24
25 /* Private includes -----
   */
26 /* USER CODE BEGIN Includes */
27 #include "math.h"
28 /* USER CODE END Includes */
29
30 /* Private typedef -----
   */
31 /* USER CODE BEGIN PTD */
32
33 /* USER CODE END PTD */
34
35 /* Private define -----
   */
36 /* USER CODE BEGIN PD */
37
38 /* USER CODE END PD */
39
40 /* Private macro -----
   */
41 /* USER CODE BEGIN PM */
42
43 /* USER CODE END PM */
44
45 /* Private variables -----
   */
46
47 /* USER CODE BEGIN PV */
48
49 /* USER CODE END PV */
50
51 /* Private function prototypes -----
   */
52 void SystemClock_Config(void);
53 /* USER CODE BEGIN PFP */
54
55 /* USER CODE END PFP */
56
57 /* Private user code -----
   */
58 /* USER CODE BEGIN 0 */
59
60
61 #define LED_MAX 8
62 #define USE_BRIGHTNESS 1
63

```

```

64
65     uint8_t LED_Data[LED_MAX][4];
66     uint8_t LED_Mod[LED_MAX][4];
67
68
69     int datasentflag = 0;
70
71
72 void HAL_TIM_PWM_PulseFinshedCallback(TIM_HandleTypeDef *htim)
73 {
74     HAL_TIM_PWM_Stop_DMA(&htim3,TIM_CHANNEL_1);
75     datasentflag = 1;
76
77 }
78
79 void Set_LED (int LEDnum,int Green,int Red,int Blue)
80 {
81
82     LED_Data[LEDnum][0] = LEDnum;
83     LED_Data[LEDnum][1] = Green;
84     LED_Data[LEDnum][2] = Red;
85     LED_Data[LEDnum][3] = Blue;
86
87 }
88
89 #define PI 3.14159265
90
91 /**
92  * 设置亮度
93  *
94  * 本函数用于根据输入的亮度值调整LED的亮度。该函数仅在定义了USE_BRIGHTNESS宏的情况下有
    效。
95  * 对每个LED，函数会根据给定的亮度值调整其亮度。如果输入的亮度值大于45，则会将亮度值固定为
    45。
96  * 调整亮度的算法是通过改变LED颜色的强度来实现的。
97  *
98  * @param brightness 亮度值，一个整数，范围0到45。
99  */
100 void Set_Brightness (int brightness)
101 {
102 #if USE_BRIGHTNESS
103
104     // 如果亮度值大于45，则将其限制为45。
105     if (brightness >45 ) brightness = 45;
106     // 遍历所有LED，调整它们的亮度。
107     for(int i=0;i<LED_MAX;i++)
108     {

```

```

109         // 初始化LED的修改值为原始值。
110         LED_Mod[i][0] = LED_Data[i][0];
111         // 对每个LED的颜色通道进行亮度调整。
112         for( int j=1;j<4;j++)
113         {
114             // 计算角度，并转换为弧度。
115             float angle =90-brightness;// 度数，用于计算亮度调整系数。
116             angle = angle*PI /180;// 转换为弧度。
117             // 根据角度计算新的亮度值。
118             LED_Mod[i][j] = (LED_Data[i][j])/(tan(angle));
119         }
120     }
121 #endif
122 }
123
124 uint16_t pwmData[24*LED_MAX+240];
125 void WS2812_Send (void)
126 {
127     uint32_t indx = 0;
128     uint32_t color;
129     /* for (int i=0;i<240;i++)
130     {
131         pwmData[indx] = 0;
132         indx++;
133     }*/
134     for (int i=0;i<LED_MAX;i++)
135     {
136         color = ((LED_Mod[i][1]<<16)|(LED_Mod[i][2]<<8)|(LED_Mod[i]
137 [3]));
138
139         for(int i=23;i>=0;i--)
140         {
141             if(color&(1<<i))
142             {
143                 pwmData[indx] = 65;
144
145             }
146             else pwmData[indx]= 25;
147
148             indx++;
149         }
150     }
151
152     for ( int i=0;i<240;i++)
153     {
154         pwmData[indx]=0;

```

```

155         indx++;
156     }
157
158     HAL_TIM_PWM_Start_DMA(&htim3,TIM_CHANNEL_1,(uint32_t *)pwmData,indx);
159     while(datasentflag){};
160     datasentflag = 0;
161
162 }
163
164
165
166 /*void send (int Green,int Red,int Blue)
167 {
168         uint32_t data=(Green<<16)|(Red<<8)|Blue;
169
170     for(int i=23;i>=0;i--){
171
172         if(data&(1<<i)) pwmData[i]=60;
173         else pwmData[i]=30;
174
175
176     }
177     HAL_TIM_PWM_Start_DMA(&htim3,TIM_CHANNEL_1,(uint32_t *)pwmData,24);
178
179
180     }*/
181 /* USER CODE END 0 */
182
183 /**
184  * @brief The application entry point.
185  * @retval int
186  */
187 int main(void)
188 {
189     /* USER CODE BEGIN 1 */
190
191     /* USER CODE END 1 */
192
193     /* MCU Configuration-----
194
195     /* Reset of all peripherals, Initializes the Flash interface and the
196     Systick. */
197     HAL_Init();
198
199     /* USER CODE BEGIN Init */
200
201

```



```

200  /* USER CODE END Init */
201
202  /* Configure the system clock */
203  SystemClock_Config();
204
205  /* USER CODE BEGIN SysInit */
206
207  /* USER CODE END SysInit */
208
209  /* Initialize all configured peripherals */
210  MX_GPIO_Init();
211  MX_DMA_Init();
212  MX_TIM3_Init();
213  /* USER CODE BEGIN 2 */
214  Set_LED(0,255,0,0);
215  Set_LED(1,0,255,0);
216  Set_LED(2,255,255,255);
217
218  Set_Brightness(45);
219  WS2812_Send();
220
221
222
223  /* USER CODE END 2 */
224
225  /* Infinite loop */
226  /* USER CODE BEGIN WHILE */
227  while (1)
228  {
229      /* USER CODE END WHILE */
230
231      /* USER CODE BEGIN 3 */
232          for(int i=0;i<45;i++){
233
234              Set_Brightness(i);
235              WS2812_Send();
236              HAL_Delay(50);
237          }
238          for(int i=45;i>0;i--){
239
240              Set_Brightness(i);
241              WS2812_Send();
242              HAL_Delay(50);
243          }
244      }
245      /* USER CODE END 3 */
246  }

```

```

247
248 /**
249  * @brief System Clock Configuration
250  * @retval None
251  */
252 void SystemClock_Config(void)
253 {
254     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
255     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
256
257     /** Initializes the RCC Oscillators according to the specified parameters
258     * in the RCC_OscInitTypeDef structure.
259     */
260     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
261     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
262     RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
263     RCC_OscInitStruct.HSIState = RCC_HSI_ON;
264     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
265     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
266     RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
267     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
268     {
269         Error_Handler();
270     }
271
272     /** Initializes the CPU, AHB and APB buses clocks
273     */
274     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
275                                     |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
276     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
277     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
278     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
279     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
280
281     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
282     {
283         Error_Handler();
284     }
285 }
286
287 /* USER CODE BEGIN 4 */
288
289 /* USER CODE END 4 */
290
291 /**
292  * @brief This function is executed in case of error occurrence.
293  * @retval None

```

```

294  */
295 void Error_Handler(void)
296 {
297     /* USER CODE BEGIN Error_Handler_Debug */
298     /* User can add his own implementation to report the HAL error return state
        */
299     __disable_irq();
300     while (1)
301     {
302     }
303     /* USER CODE END Error_Handler_Debug */
304 }
305
306 #ifdef USE_FULL_ASSERT
307 /**
308  * @brief Reports the name of the source file and the source line number
309  *        where the assert_param error has occurred.
310  * @param file: pointer to the source file name
311  * @param line: assert_param error line source number
312  * @retval None
313  */
314 void assert_failed(uint8_t *file, uint32_t line)
315 {
316     /* USER CODE BEGIN 6 */
317     /* User can add his own implementation to report the file name and line
        number,
318     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
        */
319     /* USER CODE END 6 */
320 }
321 #endif /* USE_FULL_ASSERT */
322

```

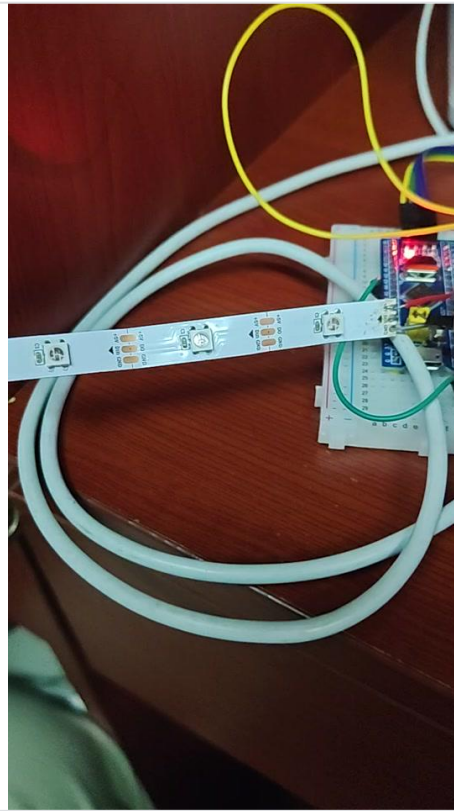
~~失败了，程序陷入了一种很诡吊的情况，单片机的pwm线一旦断开，ws2812灭灯。但是，后面的for循环不断改变灯的亮度，却无法实现，我猜测应该是原本的pwm码的波形一直是重复输出第一次的pwmData，后面的要么是终止reset符没做好，要么是所谓的回调函数没又正确的终止pwm的输出，或者二者兼有？~~

~~现在是0:27，我已经被这个问题困扰两天了，pwm线也断了，只能手动连接，带来些许麻烦，祝愿第二天的修改能顺利进行，~~

~~第二天我要在PA0处加上一个开关以此来控制灯的亮灭（复位建无法，nm电脑5，4，3没电要）~~

~~然后再加上一个ADC模拟数字信号，不过这样的话~~

~~纠错方法，改变回调函数的触发条件，使其datasentflag更符合条件~~

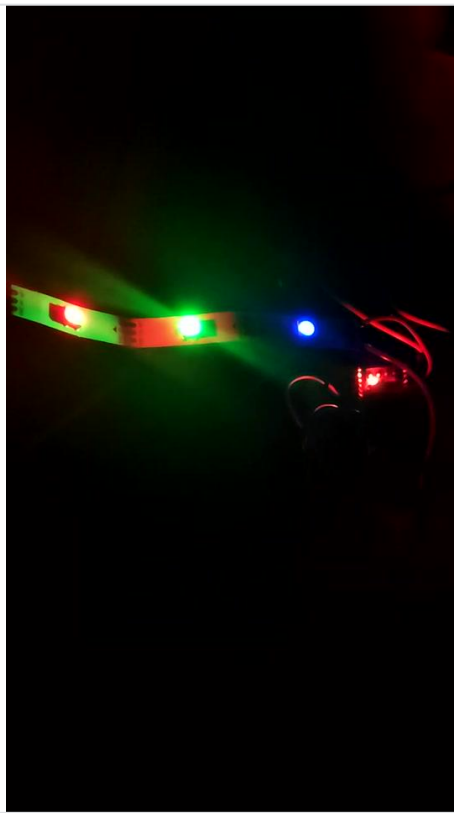


视频系手动开关din口，成功实现呼吸灯与手动开关

只有呼吸灯呼吸灯没啥意思，再额外加上几行流水灯特色

```
1  do {
2      t++;
3      if (t % 3 == 0) {
4          Set_LED(0, 255, 0, 0);
5          Set_LED(1, 0, 255, 0);
6          Set_LED(2, 0, 0, 255);
7      } else if (t % 3 == 1) {
8          Set_LED(1, 255, 0, 0);
9          Set_LED(2, 0, 255, 0);
10         Set_LED(0, 0, 0, 255);
11     } else {
12         Set_LED(0, 255, 0, 0);
13         Set_LED(2, 0, 255, 0);
14         Set_LED(1, 0, 0, 255);
15     }
16 } while (!t); //记得提前int t;该循环函数写在呼吸灯特效下面或上面都行，或者把呼吸灯特效删了也行。
```

这是有呼吸的流水灯(注意看每种颜色位置变化)



## 2.5 使用ADC检查外部电压

本MCU使用ADC1, MODE IN0, PA0作为ADC的配置

`HAL_ADC_Start(&hadc1);` 用于启动ADC1的转换过程。而  
`HAL_ADC_PollForConversion(&hadc1, 300);` 则是阻塞式等待转换完成，这里的 `300` 表示最大等待时间（单位为微秒），超过这个时间后，函数将不再等待并返回当前状态。

```
1  HAL_ADC_Start(&hadc1);  
2  HAL_ADC_PollForConversion(&hadc1, 300);
```

1. **读取转换结果：** 转换完成后，通过下面这行代码获取ADC的转换结果：

```
1  raw = HAL_ADC_GetValue(&hadc1);
```

2. **处理转换结果：** 获取到ADC转换得到的原始数据 `raw` 之后，代码进行了如下处理：

```
1 float vin = raw*(3.3/4096);
2 sprintf(msg2,"vol = %.2f\r\n",vin);
```

将原始数值转化为实际电压值，STM32F103系列ADC的满量程范围是0-3.3V，分辨率为12位，因此可以通过 `raw * (3.3 / 4096)` 计算出对应的电压值 `vin`。然后将这个电压值格式化字符串存储在 `msg2` 中，以便通过串口发送出去

### 3. 串口发送电压信息：最后，使用串口发送电压值：

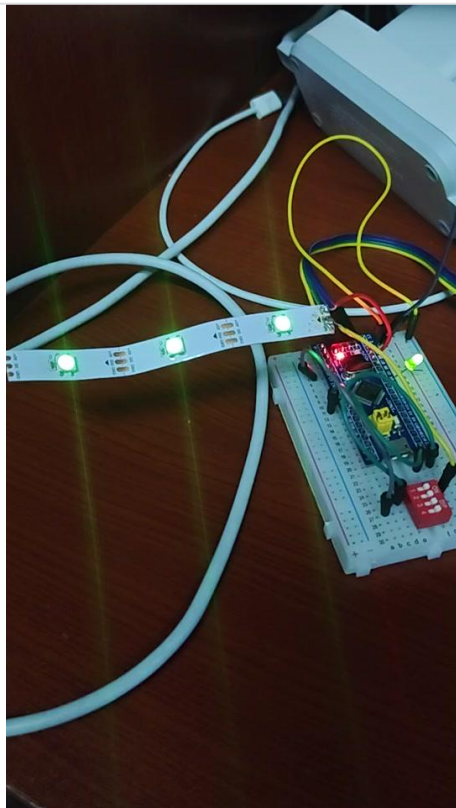
```
1 HAL_UART_Transmit(&huart1,(uint8_t *)msg2,strlen(msg2),300);
```

这行代码将包含电压信息的字符串通过串口发送出去。

### 4. 选择结构：实现<1、1~2、>2三种情况

```
1 if (vin < 1) // 小于1v
2     {
3
4         Set_LED(0, 255, 0, 0);
5         Set_LED(1, 255, 0, 0);
6         Set_LED(2, 255, 0, 0); // 绿光
7         Set_Brightness(50);
8         WS2812_Send();
9     }
10    else if(vin < 2)//1~2
11    {
12        Set_LED(0, 0, 0, 255);
13        Set_LED(1,0, 0, 255);
14        Set_LED(2, 0, 0, 255); // 蓝光
15        Set_Brightness(50);
16        WS2812_Send();
17    }
18    else // >2
19    {
20        Set_LED(0,0,255, 0);
21        Set_LED(1,0, 255,0);
22        Set_LED(2,0,255,0); // 红光
23        Set_Brightness(50);
24        WS2812_Send();
25    }
26    HAL_Delay(50);
27
```

ADC检测效果，我右手拿到是3.3v左右（3.22~3.23）的电压，可以看到，给一个大于二信号后变色，实现了目标



## 2.6 三个效果的独立函数

截至到这一行，我已经完成了前面的三个目标

但是由于目标二和三有点冲突，所以我决定做一个函数，让整个程序更美观

### 1. 呼吸灯效

```
1  ///speed为变换速度，数字越小越快
2  void Breathing_light(int speed) {
3      for(int i = 0; i < 45; i++) {
4          Set_Brightness(i);
5          WS2812_Send();
6          HAL_Delay(speed);
7      }
8      for(int i = 45; i > 0; i--) {
9          Set_Brightness(i);
10         WS2812_Send();
11         HAL_Delay(speed);
12     }
```

```

13     WS2812_Send();
14     HAL_Delay(speed);
15 }
16

```

## 2. 流水灯效

```

1  //speed为变换速度，数字越小越快
2  void Water_light(int speed) {
3
4      t++;
5      if (t % 3 == 0) {
6          Set_LED(0, 255, 0, 0);
7          Set_LED(1, 0, 255, 0);
8          Set_LED(2, 0, 0, 255);
9      } else if (t % 3 == 1) {
10         Set_LED(1, 255, 0, 0);
11         Set_LED(2, 0, 255, 0);
12         Set_LED(0, 0, 0, 255);
13     } else {
14         Set_LED(0, 255, 0, 0);
15         Set_LED(2, 0, 255, 0);
16         Set_LED(1, 0, 0, 255);
17     }
18     Set_Brightness(50);
19     WS2812_Send();
20     HAL_Delay(speed);
21
22 } //t为一个全局变量  int t = 0;

```

## 3. ADC检测灯效

```

1  void ADC_V(void)
2  {
3      // 开始ADC采样
4      HAL_ADC_Start(&hadc1);
5      HAL_ADC_PollForConversion(&hadc1, 300);
6      // 获取ADC转换结果并计算电压
7      raw = HAL_ADC_GetValue(&hadc1);
8      float vin = raw*(3.3/4096);
9      // 创建包含电压值的字符串消息
10     sprintf(msg2, "vol = %.2f\r\n", vin);
11     // 通过串口发送电压值信息

```



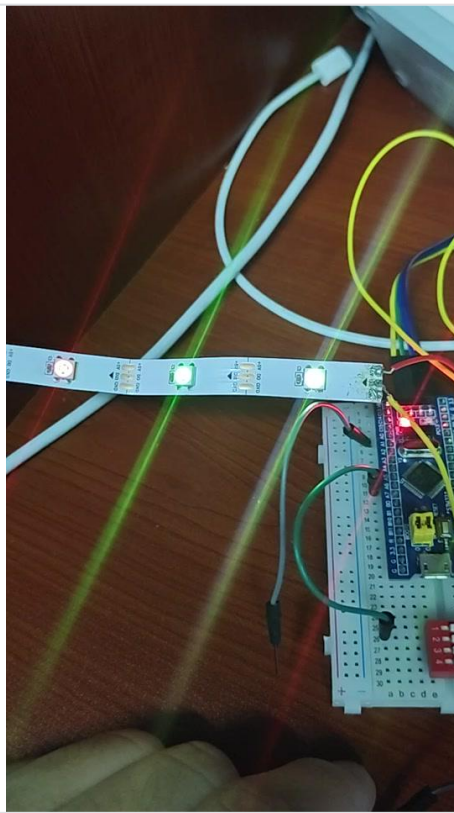
```

12          HAL_UART_Transmit(&huart1,(uint8_t
    *)msg2,strlen(msg2),300);
13          HAL_Delay(100);
14          if (vin < 1) // 小于1v
15          {
16
17              Set_LED(0, 255, 0, 0);
18              Set_LED(1, 255, 0, 0);
19              Set_LED(2, 255, 0, 0); // 绿光
20              Set_Brightness(50);
21              WS2812_Send();
22          }
23          else if(vin < 2)//1~2
24          {
25              Set_LED(0, 0, 0, 255);
26              Set_LED(1,0, 0, 255);
27              Set_LED(2, 0, 0, 255); // 蓝光
28              Set_Brightness(50);
29              WS2812_Send();
30          }
31          else// >2
32          {
33              Set_LED(0,0,255, 0);
34              Set_LED(1,0, 255,0);
35              Set_LED(2,0,255,0); // 红光
36              Set_Brightness(50);
37              WS2812_Send();
38          }
39          HAL_Delay(50);
40
41      }
42

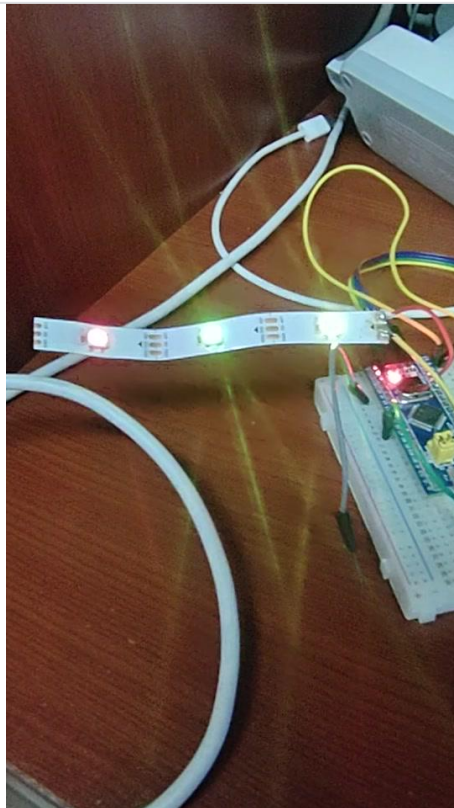
```

## 五、完结（附上测试视频和所有代码）

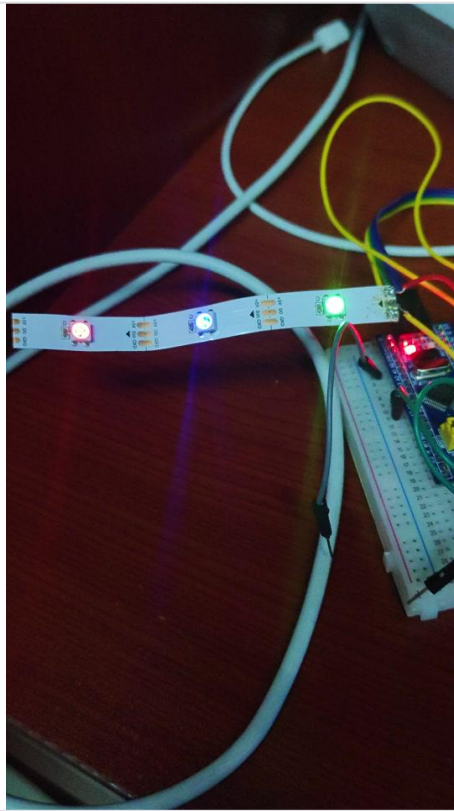
### 1. 开关以及控制颜色



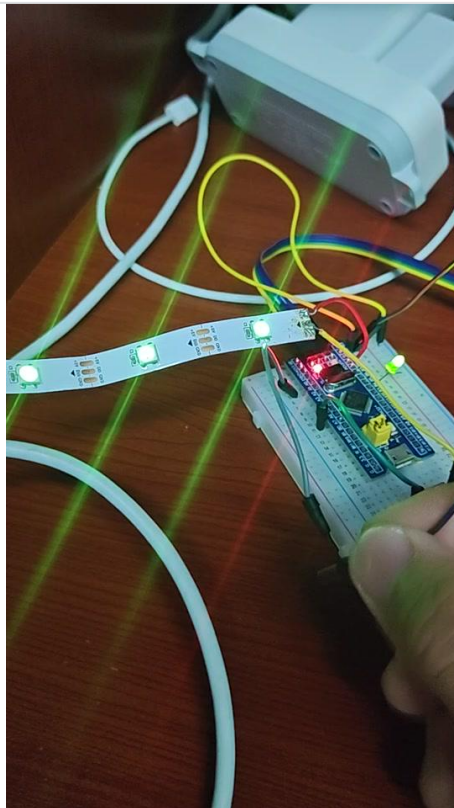
## 2. 呼吸灯



## 3. 流水+呼吸灯



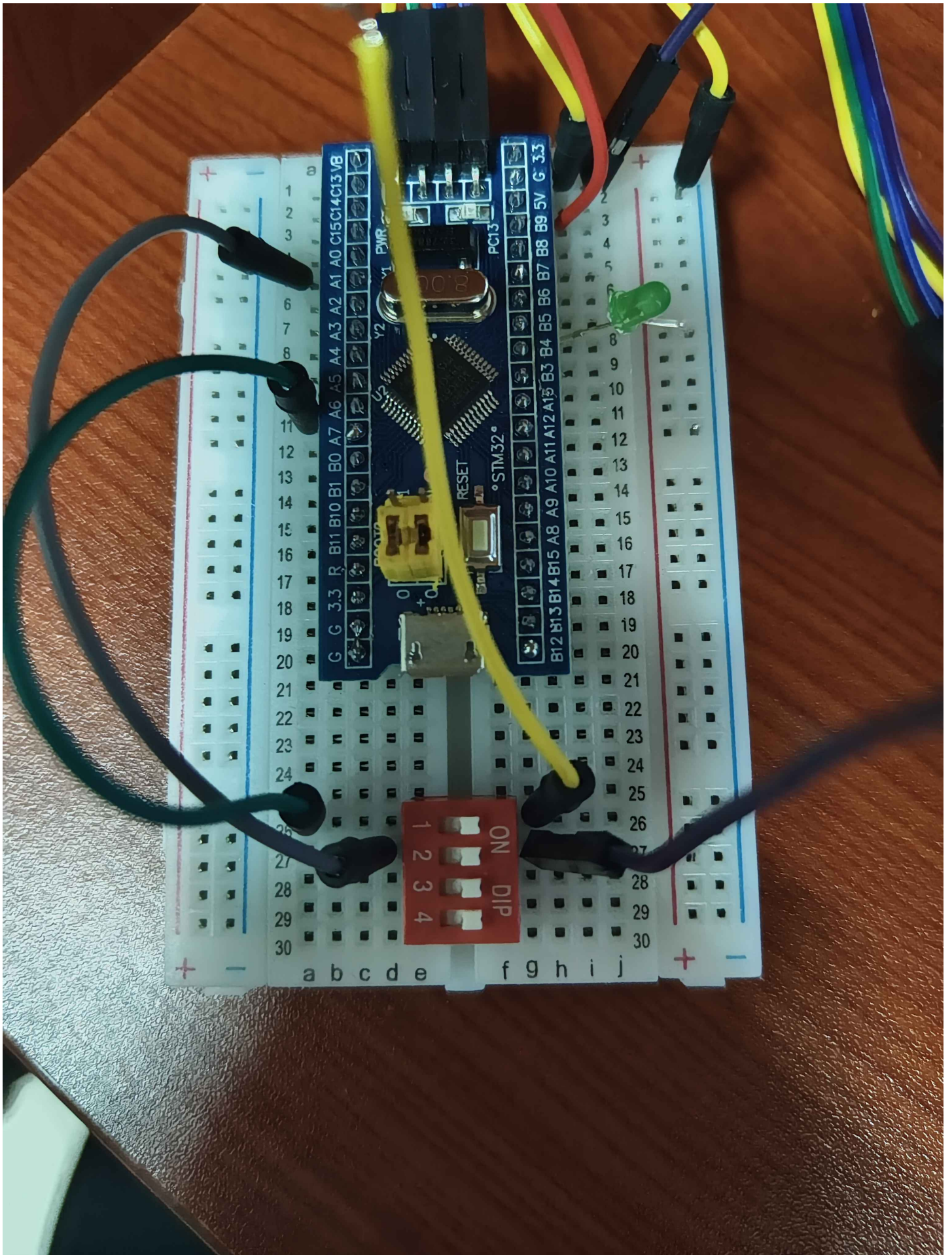
#### 4. ADC检测



#### 5. 配置过程参考 [驱动WS2812](#)（仅供参考如何配置MCU，代码看我下方）

#### 6. 接线图片





我的评价为不需要再外接5V稳压，单片机也足够3个LED供电

## 7. main.c代码

```

1  /* USER CODE BEGIN Header */
2  /**
3
4      *****
5      * @file           : main.c
6      * @brief          : Main program body
7
8      *****
9      * @attention
10     *
11     * Copyright (c) 2024 STMicroelectronics.
12     * All rights reserved.
13     *
14     * This software is licensed under terms that can be found in the LICENSE file
15     * in the root directory of this software component.
16     * If no LICENSE file comes with this software, it is provided AS-IS.
17     *
18     *****
19     */
20  /* USER CODE END Header */
21  /* Includes -----
22     */
23  #include "main.h"
24  #include "adc.h"
25  #include "dma.h"
26  #include "tim.h"
27  #include "usart.h"
28  #include "gpio.h"
29
30  /* Private includes -----
31     */
32  /* USER CODE BEGIN Includes */
33  #include "math.h"
34  #include <stdio.h>
35  #include <string.h>
36  /* USER CODE END Includes */
37
38  /* Private typedef -----
39     */
40  /* USER CODE BEGIN PTD */
41
42  /* USER CODE END PTD */
43
44  /* Private define -----
45     */

```

```

40  /* USER CODE BEGIN PD */
41
42  /* USER CODE END PD */
43
44  /* Private macro -----
    */
45  /* USER CODE BEGIN PM */
46
47  /* USER CODE END PM */
48
49  /* Private variables -----
    */
50
51  /* USER CODE BEGIN PV */
52  void Breathing_light(int);
53  void Water_light(int);
54  void ADC_V(void);
55  /* USER CODE END PV */
56
57  /* Private function prototypes -----
    */
58  void SystemClock_Config(void);
59  /* USER CODE BEGIN PFP */
60
61  /* USER CODE END PFP */
62
63  /* Private user code -----
    */
64  /* USER CODE BEGIN 0 */
65
66
67  #define LED_MAX 3
68  #define USE_BRIGHTNESS 1
69  int t=0;
70
71      uint8_t LED_Data[LED_MAX][4];
72      uint8_t LED_Mod[LED_MAX][4];
73      uint16_t raw;
74      char msg2[50];
75
76      int datasentflag = 0;
77
78
79  void HAL_TIM_PWM_PulseFinshedCallback(TIM_HandleTypeDef *htim)
80  {
81      HAL_TIM_PWM_Stop_DMA(&htim3,TIM_CHANNEL_1);
82      datasentflag = 1;

```

```

83
84 }
85
86 void Set_LED (int LEDnum, int Green, int Red, int Blue)
87 {
88
89     LED_Data[LEDnum][0] = LEDnum;
90     LED_Data[LEDnum][1] = Green;
91     LED_Data[LEDnum][2] = Red;
92     LED_Data[LEDnum][3] = Blue;
93
94 }
95
96 void Set_Brightness(int brightness) {
97     float factor = (float)brightness / 45.0f; // 计算亮度系数
98     for(int i = 0; i < LED_MAX; i++) {
99         for(int j = 1; j < 4; j++) {
100             // 根据亮度系数调整颜色 ??
101             LED_Mod[i][j] = (uint8_t)(LED_Data[i][j] * factor);
102         }
103     }
104 }
105
106
107 uint16_t pwmData[24*LED_MAX+240];
108 void WS2812_Send(void) {
109     uint32_t indx = 0;
110     uint32_t color;
111
112     for(int i = 0; i < LED_MAX; i++) {
113         color = ((LED_Mod[i][1] << 16) | (LED_Mod[i][2] << 8) | (LED_Mod[i]
114 [3]));
115
116         for(int i = 23; i >= 0; i--) {
117             if(color & (1 << i)) {
118                 pwmData[indx] = 65;
119             } else {
120                 pwmData[indx] = 25;
121             }
122             indx++;
123         }
124
125         for(int i = 0; i < 240; i++) {
126             pwmData[indx] = 0;
127             indx++;
128         }

```

```

129
130     HAL_TIM_PWM_Start_DMA(&htim3, TIM_CHANNEL_1, (uint32_t *)pwmData, indx);
131 }
132
133 void HAL_TIM_PWM_PulseFinishedCallback(TIM_HandleTypeDef *htim) {
134     if(htim->Instance == TIM3) {
135         HAL_TIM_PWM_Stop_DMA(&htim3, TIM_CHANNEL_1);
136         datasentflag = 1;
137     }
138 }
139 void Breathing_light(int speed) {
140     for(int i = 0; i < 45; i++) {
141         Set_Brightness(i);
142         WS2812_Send();
143         HAL_Delay(speed);
144     }
145     for(int i = 45; i > 0; i--) {
146         Set_Brightness(i);
147         WS2812_Send();
148         HAL_Delay(speed);
149     }
150     WS2812_Send();
151     HAL_Delay(speed);
152 }
153
154 //speed为变换速度，数字越小越快
155 void Water_light(int speed) {
156
157     t++;
158     if (t % 3 == 0) {
159         Set_LED(0, 255, 0, 0);
160         Set_LED(1, 0, 255, 0);
161         Set_LED(2, 0, 0, 255);
162     } else if (t % 3 == 1) {
163         Set_LED(1, 255, 0, 0);
164         Set_LED(2, 0, 255, 0);
165         Set_LED(0, 0, 0, 255);
166     } else {
167         Set_LED(0, 255, 0, 0);
168         Set_LED(2, 0, 255, 0);
169         Set_LED(1, 0, 0, 255);
170     }
171     Set_Brightness(50);
172     WS2812_Send();
173     HAL_Delay(speed);
174
175 }

```



```

176
177 void ADC_V(void)
178 {
179     // 开始ADC采样
180     HAL_ADC_Start(&hadc1);
181     HAL_ADC_PollForConversion(&hadc1,300);
182     // 获取ADC转换结果并计算电压
183     raw = HAL_ADC_GetValue(&hadc1);
184     float vin = raw*(3.3/4096);
185     // 创建包含电压值的字符串消息
186     sprintf(msg2,"vol = %.2f\r\n",vin);
187     // 通过串口发送电压值信息
188     HAL_UART_Transmit(&huart1,(uint8_t
*)msg2,strlen(msg2),300);
189     HAL_Delay(100);
190     if (vin < 1) // 小于1v
191     {
192
193         Set_LED(0, 255, 0, 0);
194         Set_LED(1, 255, 0, 0);
195         Set_LED(2, 255, 0, 0); // 绿光
196         Set_Brightness(50);
197         WS2812_Send();
198     }
199     else if(vin < 2)//1~2
200     {
201         Set_LED(0, 0, 0, 255);
202         Set_LED(1,0, 0, 255);
203         Set_LED(2, 0, 0, 255); // 蓝光
204         Set_Brightness(50);
205         WS2812_Send();
206     }
207     else // >2
208     {
209         Set_LED(0,0,255, 0);
210         Set_LED(1,0, 255,0);
211         Set_LED(2,0,255,0); // 红光
212         Set_Brightness(50);
213         WS2812_Send();
214     }
215     HAL_Delay(50);
216
217 }
218
219 /*void send (int Green,int Red,int Blue)
220 {
221     uint32_t data=(Green<<16)|(Red<<8)|Blue;

```

```

222
223     for(int i=23;i>=0;i--){
224
225         if(data&(1<<i)) pwmData[i]=60;
226         else pwmData[i]=30;
227
228
229     }
230     HAL_TIM_PWM_Start_DMA(&htim3,TIM_CHANNEL_1,(uint32_t *)pwmData,24);
231
232
233     }*/
234 /* USER CODE END 0 */
235
236 /**
237  * @brief The application entry point.
238  * @retval int
239  */
240 int main(void)
241 {
242     /* USER CODE BEGIN 1 */
243
244     /* USER CODE END 1 */
245
246     /* MCU Configuration-----
247
248     /* Reset of all peripherals, Initializes the Flash interface and the
249     Systick. */
250
251     HAL_Init();
252
253     /* USER CODE BEGIN Init */
254
255     /* USER CODE END Init */
256
257     /* Configure the system clock */
258     SystemClock_Config();
259
260     /* USER CODE BEGIN SysInit */
261
262     /* Initialize all configured peripherals */
263     MX_GPIO_Init();
264     MX_DMA_Init();
265     MX_TIM3_Init();
266     MX_ADC1_Init();

```

```

267  MX_USART1_UART_Init();
268  /* USER CODE BEGIN 2 */
269
270  // () 说要有光, 于是便有了光
271  for(int i=0;i<=LED_MAX;i++){
272      if(i%3==1) Set_LED(i,255,0,0);
273      else if(i%3==2) Set_LED(i,0,255,0);
274      else if (i%3==0) Set_LED(i,255,255,266);
275  }
276
277  Set_Brightness(50);
278  WS2812_Send();
279
280
281
282  /* USER CODE END 2 */
283
284  /* Infinite loop */
285  /* USER CODE BEGIN WHILE */
286  while (1)
287  {
288      /* USER CODE END WHILE */
289
290      /* USER CODE BEGIN 3 */
291      //三种灯效, 想用那个, 删掉前面的“//”就行, 1和2可以组合用, speed为变换速度, 数
      字越小越快
292          //Breathing_light(10);
293          //Water_light(100) ;
294          // ADC_V();
295      }
296      /* USER CODE END 3 */
297  }
298
299  /**
300   * @brief System Clock Configuration
301   * @retval None
302   */
303  void SystemClock_Config(void)
304  {
305      RCC_OscInitTypeDef RCC_OscInitStruct = {0};
306      RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
307      RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
308
309      /** Initializes the RCC Oscillators according to the specified parameters
310       * in the RCC_OscInitTypeDef structure.
311       */
312      RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;

```

```

313  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
314  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
315  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
316  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
317  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
318  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
319  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
320  {
321      Error_Handler();
322  }
323
324  /** Initializes the CPU, AHB and APB buses clocks
325  */
326  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
327                               |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
328  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
329  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
330  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
331  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
332
333  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
334  {
335      Error_Handler();
336  }
337  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC;
338  PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
339  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
340  {
341      Error_Handler();
342  }
343 }
344
345 /* USER CODE BEGIN 4 */
346
347 /* USER CODE END 4 */
348
349 /**
350   * @brief This function is executed in case of error occurrence.
351   * @retval None
352   */
353 void Error_Handler(void)
354 {
355     /* USER CODE BEGIN Error_Handler_Debug */
356     /* User can add his own implementation to report the HAL error return state
357     */
357     __disable_irq();
358     while (1)

```

```

359  {
360  }
361  /* USER CODE END Error_Handler_Debug */
362  }
363
364  #ifdef  USE_FULL_ASSERT
365  /**
366   * @brief Reports the name of the source file and the source line number
367   *        where the assert_param error has occurred.
368   * @param file: pointer to the source file name
369   * @param line: assert_param error line source number
370   * @retval None
371   */
372  void assert_failed(uint8_t *file, uint32_t line)
373  {
374      /* USER CODE BEGIN 6 */
375      /* User can add his own implementation to report the file name and line
376         number,
377         ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
378         */
379      /* USER CODE END 6 */
380  }
381  #endif /* USE_FULL_ASSERT */

```

## 8. 全部配置和代码



WS2812\_STM32F103C8T6.zip

7.02MB



解压，用CUBEIDE打开，按照我上面的连线，阅读main.c的注释，选择自己需要的功能后，烧录即可使用🤖

//别忘了在main.c里把有关所有灯带数量的函数改成你的\

[Github](#)