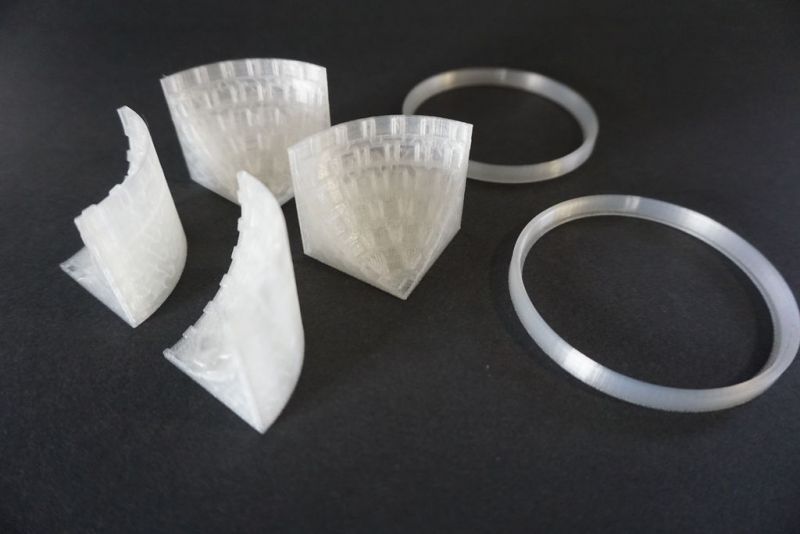
# **用 ESP32 制造炫彩 LED 球**

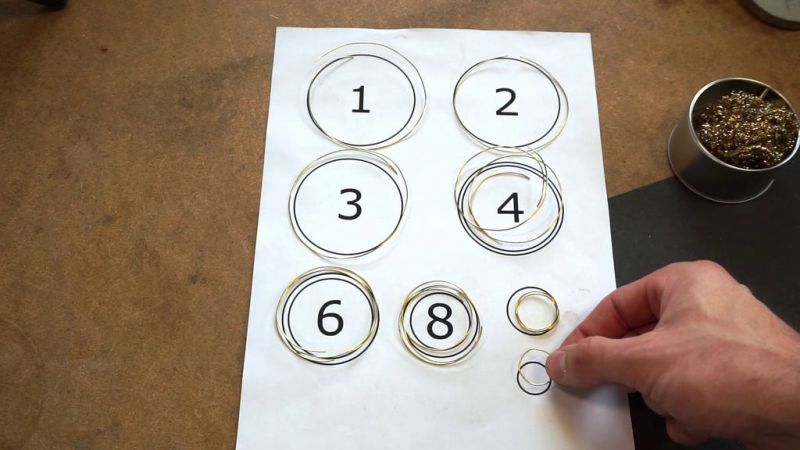
  
MAKER:jiripraus／译：趣无尽 Cherry（转载请注明出处）

本期带大家制作一个 LED 的炫彩发光球，使用 3D 打印外壳搭建球体并将 LED 灯嵌入在球体表面，最后再上传代码，一个颜色绚丽的 LED 球就完成了。用它来做装饰品，效果异常炫酷！  
焊接球体会比较耗时，大家一定要有耐心。  

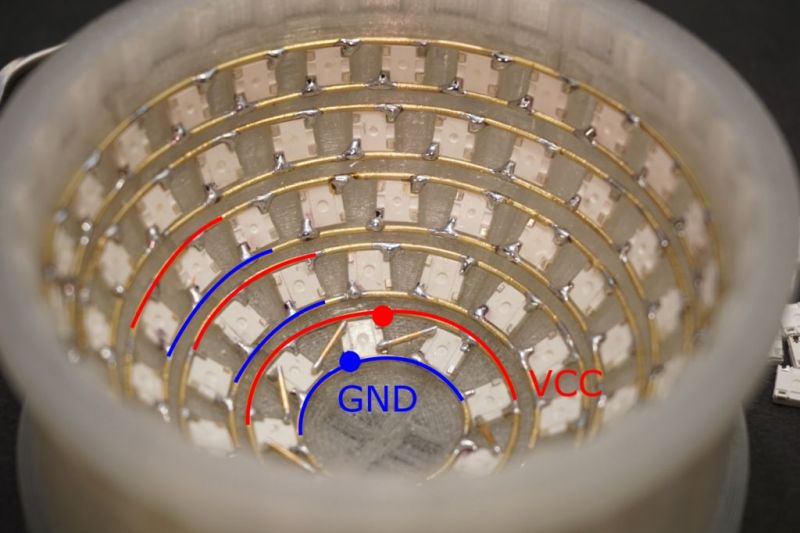
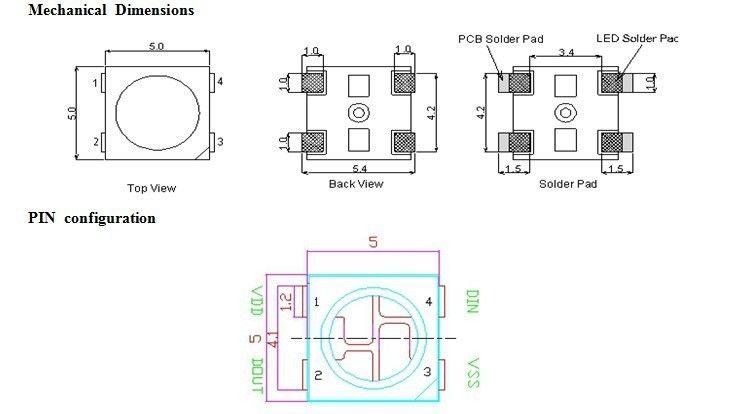
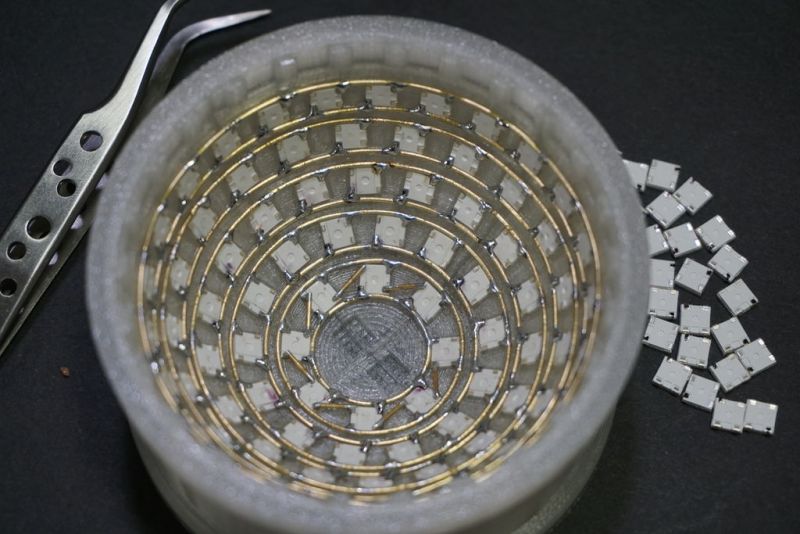

### **材料清单**

  
ESP32 开发板（带电池） × 1  
WS2812b RGB LED × 200  
1000mAh 锂电池 × 1  
0.8mm 黄铜线 × 若干  
锡焊 × 若干  
助焊剂 × 若干  
3D 打印机 × 1  
窄钳 × 1  
圆嘴钳 × 1  
侧切刀 × 1  
焊台 × 1  
镊子 × 1  
**[自制弯曲机](https://www.instructables.com/id/Spring-Making-Wire-Bending-Machine/" \t "https://www.quwj.com/2020/01/01/_blank)**×1  
**3D 打印外壳**1、设计出 3D 打印的球体模板，球体的模板是由四个部分组成的半圆。  
  
2、再由两个环固定到位。这四个组成部分很重要，注意这四个部分分为 templateA 和 templateB 两种，因为球体不是完全对称的。请分别打印两次这两个部分就可以打印出一个半圆球体。  
  
3D 打印文件请在项目文件库中下载。  
**[https://make.quwj.com/project/186](https://make.quwj.com/project/186" \t "https://www.quwj.com/2020/01/01/_blank)**

### **制作黄铜圈**

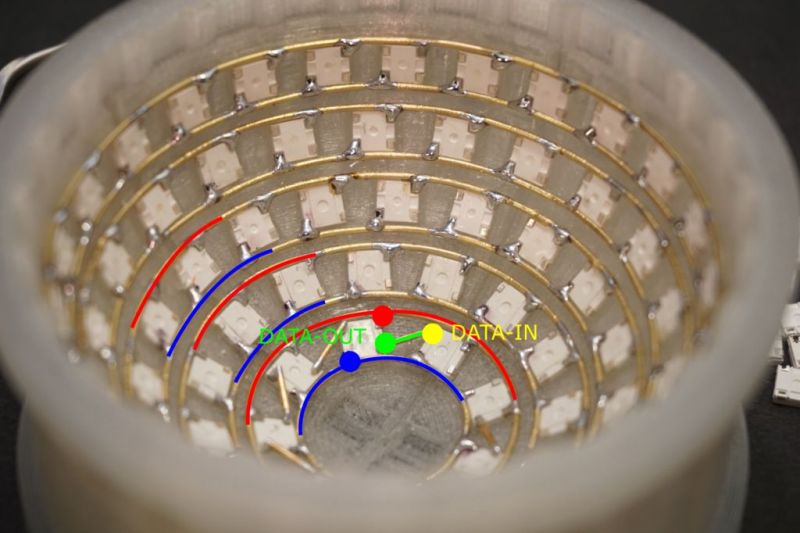
  
整个球体由十一个 LED 环组成，每个环由两条圆形的线和一定数量的 LED 组成。整个球体共需要二十二个黄铜圈。如图所示，纸上画出半个球体所需的黄铜圈的大小。具体的大小还是根据你的实际情况来。  
最后，球体的最终形状主要取决于黄铜弯曲的准确性。如果黄铜圈的形状不是很圆，那么可能在末端弯曲。你可以参考我的另一个项目：金属材料弯曲机**[https://www.instructables.com/id/Spring-Making-Wire-Bending-Machine/](https://www.instructables.com/id/Spring-Making-Wire-Bending-Machine/" \t "https://www.quwj.com/2020/01/01/_blank)**， 打造出完美的形状。  
关于环形的文件请在项目文件库中下载。  
**[https://make.quwj.com/project/186](https://make.quwj.com/project/186" \t "https://www.quwj.com/2020/01/01/_blank)**

### **放置 LED 灯**

1、从模板的最底部处开始安放 LED。第一环由八个 LED 灯组成。用黑色记号笔在底部标记出 LED 灯的所有 GND 引线，以便能够识别 GND（VSS）引脚的位置。放置所有 LED 灯时，应将 GND 引脚朝向环内，VCC 引脚朝向环外。如图所示，之后的每一环都是如此，内线接地，外线通电。  
  
  
2、每个 LED 灯都会涂抹一些助焊剂中，以便插入它们时，能够更好的固定在插槽内。  
  
3、插入内环并所有的 GND 引脚焊接到环上。插入外环并将所有的 VCC 引脚焊接到环上。其中 VCC 引脚位于 GND 引脚的对角线上。确保环上没有连接 LED 灯的任何 DATA（DIN 和 DOUT）引线。  
  
4、模板上一共六个环，LED 灯分布情况如下（从底部到顶部依次排列）：  
第一个环为八个 LED 灯，第二个环为十四个 LED 灯，第三个环为十八个 LED 灯，第四个环为二十个 LED 灯，第五个环为二十四个 LED 灯，第六个环为二十六个 LED 灯。  
  
请注意整个球体只需要一个第六个环，因为它是中间的一圈。因此，球体一半带有五个环的 LED 灯，另一半带有六个环的 LED 灯。

****WS2812b LED 说明****  
这款 LED 灯是可独立寻址的 RGB LED 灯。你只需一根电线就可以点亮它，颜色可选。与经典的通过电流量来调节亮度的 LED 灯不同，WS2812b 会始终处于供电状态并且通过数字信号来控制光。你可以任意选择灯光的颜色。你可能会问，它需要三根线吗？再用第四根线为控制链中的下一个 LED 供电。它最大的特点是将所有灯都放在一起，只用一根线来控制它们。你只需按照说明中的将第一个 LED 灯连接到其他 LED 灯上，如此继续即可。  
**[https://www.seeedstudio.com/document/pdf/WS2812B%20Datasheet.pdf](https://www.seeedstudio.com/document/pdf/WS2812B Datasheet.pdf" \t "https://www.quwj.com/2020/01/01/_blank)**

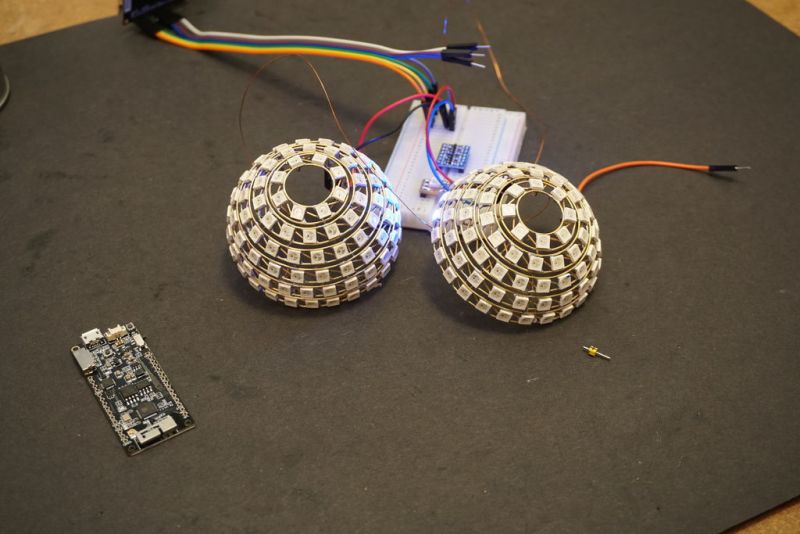
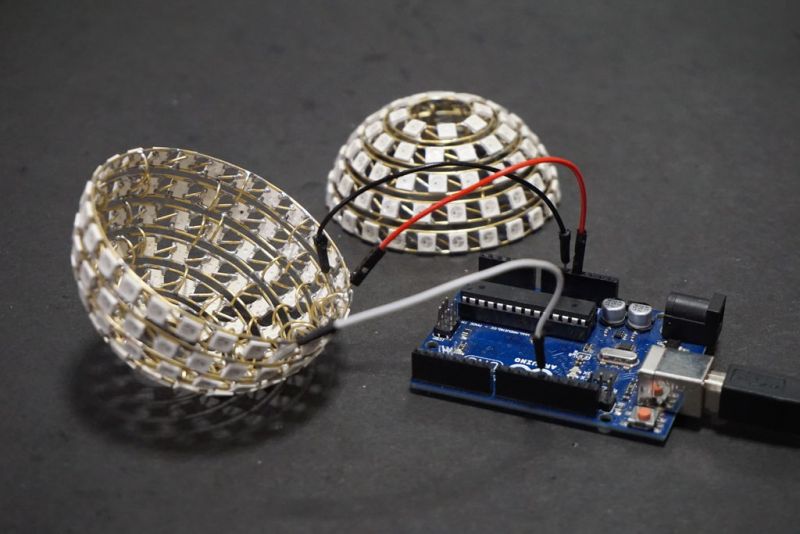
### **架接数据线**

电源线是环形的，这个部分已经完成。现在，需要连接 LED 灯的其他两条引线——DIN 和 DOUT。DIN 位于 GND 引脚旁，DOUT 位于 VCC 引脚旁。使用 5mm 长的线来连接。期间，务必确保数据线不接地或电源环，务必确保不要将数据引脚焊接到电源环。请正确操作，这很重要。  
  
从最里一环开始一圈一圈地连接。当你连接完成并达到与上一环的交叉点时，就可以跳到上一环了。我使用 C 形跳线来连接两个后续环的数据引脚。因为 C 形跳线可以绕过接地线和电源线。然后再继续连接下去直到所有 LED 灯都串连起来。  
  
最后，只剩下两条引线为连接，一根位于最顶端的环上，那是整个半球的主导 DIN 引线。另一个在底部，那是一个 DOUT 引线。  

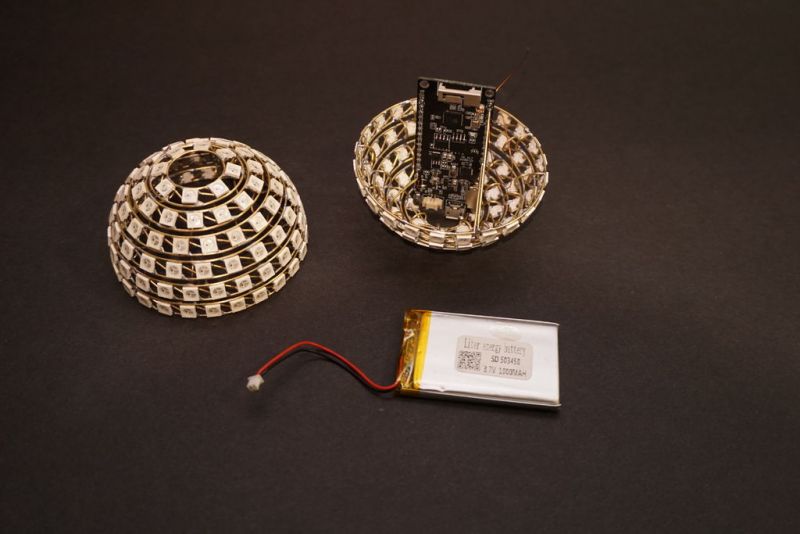
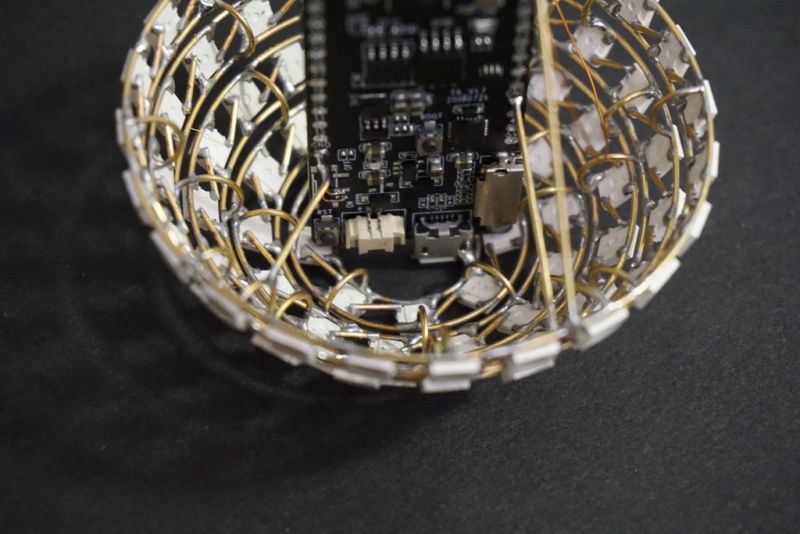

### **连接电源线并制作另一半**

将半球上每个环之间的电源线互连。连接它们时可以创建某种结构让它们串在一起。我再次使用 C 形跳线的方式来连接电源和相邻圈的 GND 线。前面三个较小的环上至少使用四根跳线，在较大的三个环上使用至少八根跳线。这样结构就会比较稳固。  
  
取下模具上固定四个部分的塑料环。并开始加固每个零件。请注意在过程中，不要损坏任何一个 LED 灯。  
  
现在半球就完成了，可以开始制作另一个半。再制作的半球的环数会不一样。就如之前提到的那样，如果你的第一个半球是六环，那么第二个半球就是五环，反之亦然。

### **测试两个半球**

在将两个半球组装前，你最好将两个半球测试一下。  
1、使用万用表进行简单测试，并检查电源线和地线之间是否短路。  
  
2、将两条短线焊接到每个半球接地线和电源线上。将电源线连接至 ESP32 板的 3.3V 引脚并接地。将所有 LED 灯都连接到一条线上。  
  
3、取较小的半球（五个环），将它的 DIN 连接到 ESP32 板的 IO21 引脚上，将 DOUT 连接到第二个半球的 DIN 引脚上。现在，只剩下一个未连接的 LED 引脚——大半球的 DOUT 引脚。那是链中的最后一个 LED 灯。然后接通 ESP32 电路板的电源，再上传代码。  
  
4、如果所有的灯都点亮就成功了。如果没有，并且终止在某处，请仔细检查 LED 灯上是否有冷接点或短路。  


### **插入微控制器和电池**

1、处理 ESP32 开发板。将开关从板子上移除（你无法接触到它，因为它将嵌入在球体内）并用两条铜线代替它。将其悬空，将微控制器放入球体中时，将其焊接。  
2、将电路板的角落的三或四个螺孔填充封闭，因为不会使用到，而且可以使它们更易于连接到球体上。  
  
3、取较小的半球，将微控制器放置其底部的中心处，并将 USB 和电池连接器朝向球部的外侧。将角孔焊接到最小的环上，它是接地的，将两根开关线穿过孔，然后将之前移除的开关焊接到它们上。开关是金属主体，也可以焊接到最小的环上。请确保不要让开关线与结构短路，也不要让电源开关超出球体。  
  
4、取一根线将其焊接到 ESP32 板上的 3.3V 引脚上。然后将另一端焊接到球体的最后一个环上，它是接电源的。我通过接地和电路板 GND 引脚之间焊接的直线来增强内部结构。  
  
5、确保在板子背面留出一块可容纳电池的空间。用铜线将主导的 DIN 引脚连接到 ESP32 的 IO21 引脚上。现在半球已经设置好，连接电池即可进行测试。  
6、将电池连接到 ESP32 板的背面，然后通过孔将其连接。

### **焊接球体**

1、焊接前，测试 ESP32 板上的螺孔是否接地。同时在它和电路板的最近的 GND 引脚之间加一条短线，它将为第二个半球提供接地。  
  
2、焊接短线，将 DOUT 从第一个半球连接到第二个半球的 DIN 引脚上（与测试阶段相同）。取一根 2mm 长的电线，并将其焊接到第二个半球的最上面的环上。我一共使用了八根这样的线。最上面的两个环都通电。因此将每个半球的两个最上面的环焊接在一起是安全的。将两个半球放置在一起并焊接。确保它们之间的空间均匀且平滑。最后，将 ESP32 板的螺丝孔与最小的接地环线连接起来。  


### **编程**

Arduino IDE 源代码：  
#include <NeoPixelBus.h>

#include <NeoPixelAnimator.h>

#define PIN\_LEDS 21

#define NUMPIXELS 194

#define LIGHTNESS 0.05f

#define BRIGHTNESS 64

RgbColor BLACK(0, 0, 0);

RgbColor RED(BRIGHTNESS, 0, 0);

RgbColor GREEN(0, BRIGHTNESS, 0);

RgbColor BLUE(0, 0, BRIGHTNESS);

RgbColor YELLOW(BRIGHTNESS, BRIGHTNESS, 0);

RgbColor WHITE(BRIGHTNESS, BRIGHTNESS, BRIGHTNESS);

NeoPixelBus<NeoGrbFeature, Neo800KbpsMethod> leds(NUMPIXELS, PIN\_LEDS);

NeoPixelAnimator animations(10); // NeoPixel animation management object

#define NUM\_RINGS 11

#define RING\_MAX\_PIXELS 26

#define RING\_MIN\_PIXELS 8

// all rings starts at 0 ray

byte RINGS[NUM\_RINGS][RING\_MAX\_PIXELS] = {

  {78, 79, 80, 81, 82, 83, 84, 77}, // 8

  {65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 63, 64}, // 14

  {48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 45, 46, 47}, // 18

  {28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 25, 26, 27}, // 20

  {5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 1, 2, 3, 4}, // 24

  {102, 101, 100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 110, 109, 108, 107, 106, 105, 104, 103}, // 26

  {127, 126, 125, 124, 123, 122, 121, 120, 119, 118, 117, 116, 115, 114, 113, 112, 111, 134, 133, 132, 131, 130, 129, 128}, // 24

  {148, 147, 146, 145, 144, 143, 142, 141, 140, 139, 138, 137, 136, 135, 154, 153, 152, 151, 150, 149}, // 20

  {167, 166, 165, 164, 163, 162, 161, 160, 159, 158, 157, 156, 155, 172, 171, 170, 169, 168}, // 18

  {182, 181, 180, 179, 178, 177, 176, 175, 174, 173, 186, 185, 184, 183}, // 14

  {192, 191, 190, 189, 188, 187, 194, 193} // 8

};

byte RING\_SIZES[NUM\_RINGS] = {8, 14, 18, 20, 24, 26, 24, 20, 18, 14, 8};

unsigned **long** startMillis = 0;

**short** animation = 0;

**void** setup() {

  Serial.begin(115200);

  initRandom();

  leds.Begin();

  leds.ClearTo(BLACK);

  leds.Show();

  //animations.StartAnimation(0, 6000, rainbowAnimation);

  animations.StartAnimation(0, 2000, raysRainbow);

  //animations.StartAnimation(0, 2000, ringsRainbow);

  //animations.StartAnimation(0, 500, xmasOrbAnimation);

  //animations.StartAnimation(0, 1000, ringAnimation);

  //animations.StartAnimation(0, 500, rayAnimation);

  //animations.StartAnimation(0, 100, randomAnimation);

}

**void** loop() {

  animations.UpdateAnimations();

  leds.Show();

  delay(10);

**if** (**false** && (startMillis == 0 || startMillis + 20000 < millis())) { // 30s

    startMillis = millis();

**switch** (animation) {

**case** 0:

        animations.StartAnimation(0, 500, rayAnimation);

**break**;

**case** 1:

        animations.StartAnimation(0, 1000, ringAnimation);

**break**;

**case** 2:

        animations.StartAnimation(0, 2000, ringsRainbow);

**break**;

**case** 3:

        animations.StartAnimation(0, 2000, raysRainbow);

**break**;

**case** 4:

        animations.StartAnimation(0, 6000, rainbowAnimation);

**break**;

**case** 5:

        animations.StartAnimation(0, 100, randomAnimation);

        animation = -1;

**break**;

    }

    animation ++;

  }

}

**void** randomAnimation(**const** AnimationParam& param) {

**float** hue;

  HslColor color;

**if** (param.state == AnimationState\_Completed) {

**for** (byte i = 0; i < 194; i ++) {

      hue = random(0, 1000) / 1000.0f;

      color = HslColor(hue, 1.0f, LIGHTNESS);

      leds.SetPixelColor(i, color);

    }

    animations.RestartAnimation(0);

  }

}

**void** rainbowAnimation(**const** AnimationParam& param) {

  HslColor color = HslColor(param.progress, 1.0f, LIGHTNESS);

  leds.ClearTo(color);

**if** (param.state == AnimationState\_Completed) {

    animations.RestartAnimation(0);

  }

}

**void** raysRainbow(**const** AnimationParam& param) {

  HslColor color;

**float** hue;

**for** (**int** i = 0; i < RING\_MAX\_PIXELS; i++) {

    hue = param.progress + (**float**) i / (**float**) RING\_MAX\_PIXELS;

**if** (hue > 1.0f) {

      hue -= 1.0f;

    }

    color = HslColor(hue, 1.0f, LIGHTNESS);

    rayColor(i, RgbColor(color));

  }

**if** (param.state == AnimationState\_Completed) {

    animations.RestartAnimation(0);

  }

}

**void** ringsRainbow(**const** AnimationParam& param) {

  HslColor color;

**float** hue;

**for** (**int** i = 0; i < NUM\_RINGS; i++) {

    hue = param.progress + (**float**) i / (**float**) NUM\_RINGS;

**if** (hue > 1.0f) {

      hue -= 1.0f;

    }

    color = HslColor(hue, 1.0f, LIGHTNESS);

    ringColor(i, RgbColor(color));

  }

**if** (param.state == AnimationState\_Completed) {

    animations.RestartAnimation(0);

  }

}

**void** xmasOrbAnimation(**const** AnimationParam& param) {

  ringColor(0, WHITE);

  ringColor(1, RED);

  ringColor(2, RED);

  ringColor(3, RED);

  ringColor(4, RED);

  ringColor(5, WHITE);

  ringColor(6, RED);

  ringColor(7, RED);

  ringColor(8, RED);

  ringColor(9, RED);

  ringColor(10, WHITE);

  byte offset = round(param.progress);

**for** (byte i = offset; i < RING\_SIZES[3]; i+=2) {

    leds.SetPixelColor(RINGS[3][i] - 1, WHITE);

  }

**for** (byte i = offset; i < RING\_SIZES[7]; i+=2) {

    leds.SetPixelColor(RINGS[7][i] - 1, WHITE);

  }

**if** (param.state == AnimationState\_Completed) {

    animations.RestartAnimation(0);

  }

}

**void** ringAnimation(**const** AnimationParam& param) {

**int** index = param.progress \* (NUM\_RINGS \* 2 - 2);

  leds.ClearTo(BLACK);

**if** (index < NUM\_RINGS) {

    ringColor(index, BLUE);

  }

**else** {

    ringColor(NUM\_RINGS - (index - NUM\_RINGS) - 2, BLUE);

  }

**if** (param.state == AnimationState\_Completed) {

    animations.RestartAnimation(0);

  }

}

**void** rayAnimation(**const** AnimationParam& param) {

**int** index = param.progress \* (RING\_MAX\_PIXELS / 2);

**if** (index > 12) {

    index = 12;

  }

  leds.ClearTo(BLACK);

  rayColor(index, BLUE);

  rayColor(index + (RING\_MAX\_PIXELS / 2), BLUE);

**if** (param.state == AnimationState\_Completed) {

    animations.RestartAnimation(0);

  }

}

**void** rayColor(byte rayIndex, RgbColor color) {

**int** pixelIndex;

  byte pixel;

**if** (rayIndex >= RING\_MAX\_PIXELS) {

**return**; // prevents out of bounds

  }

**for** (byte i = 0; i < NUM\_RINGS; i ++) {

    pixelIndex = round((**float**) RING\_SIZES[i] / (**float**) RING\_MAX\_PIXELS \* rayIndex);

    pixel = RINGS[i][pixelIndex];

**if** (pixel == 0) {

**continue**; // skip condition

    }

    leds.SetPixelColor(pixel - 1, color); // index starts from 1 (0 is stop condition)

  }

}

**void** ringColor(byte ringIndex, RgbColor color) {

  byte pixel;

**if** (ringIndex >= NUM\_RINGS) {

**return**; // prevents out of bounds

  }

**for** (byte i = 0; i < RING\_MAX\_PIXELS; i ++) {

    pixel = RINGS[ringIndex][i];

**if** (pixel == 0) {

**return**; // end condition

    }

    leds.SetPixelColor(pixel - 1, color); // index starts from 1 (0 is stop condition)

  }

}

**void** initRandom() {

  // random works best with a seed that can use 31 bits

  // analogRead on a unconnected pin tends toward less than four bits

  uint32\_t seed = analogRead(0);

  delay(1);

**for** (**int** shifts = 3; shifts < 31; shifts += 3) {

    seed ^= analogRead(0) << shifts;

    delay(1);

  }

  // Serial.println(seed);

  randomSeed(seed);

}

RgbColor colorWheel(byte wheelPos) {

  // Input a value 0 to 255 to get a color value.

  // The colours are a transition r - g - b - back to r.

  wheelPos = 255 - wheelPos;

**if** (wheelPos < 85) {

**return** RgbColor(255 - wheelPos \* 3, 0, wheelPos \* 3);

  }

**if** (wheelPos < 170) {

    wheelPos -= 85;

**return** RgbColor(0, wheelPos \* 3, 255 - wheelPos \* 3);

  }

  wheelPos -= 170;

**return** RgbColor(wheelPos \* 3, 255 - wheelPos \* 3, 0);

}

请在项目文件库中下载。  
**[https://make.quwj.com/project/186](https://make.quwj.com/project/186" \t "https://www.quwj.com/2020/01/01/_blank)**  
同时配合使用 Neopixel BUS 库来控制 LED 条。它具有非常好的界面并动画支持的效果。  
**[https://github.com/Makuna/NeoPixelBus](https://github.com/Makuna/NeoPixelBus" \t "https://www.quwj.com/2020/01/01/_blank)**  
为了控制动画，我将 LED 映射设置为 11 行 26 列的表（数组数列）。通过这种方式，我能够确切地了解 LED 灯如何放置在球体上的位置，并且可以点亮我想点亮的 LED 灯。  
  
可以变化的 6 个动画为：1、垂直圆模式；2、水平圆模式；3、垂直彩虹模式；4、水平彩虹模式；5、彩虹模式；6、随机模式。  
  
注意：LED 的亮度限制为全功率的大约 20％。因为其中有 194 个灯是全功率，它们将消耗 10 安的电流。板子将无法处理。如果电量过大，也不用担心，该板子会重新启动，不会损坏板子。

### **完成**

  
现在整个项目就完成了，这将是一个挑战自己耐心的项目，非常具有实践的意义。漂亮的 LED 炫彩球可以装饰圣诞树，也可以用来点亮户外广告。