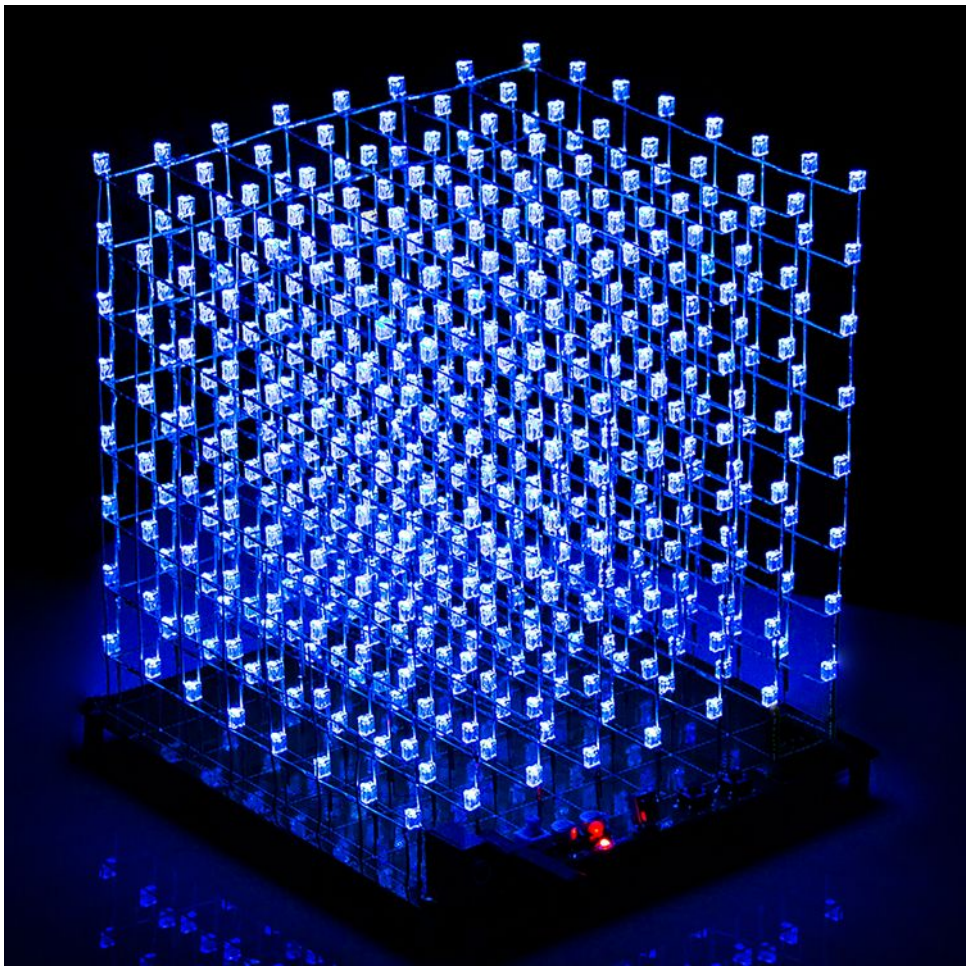
Touch Your Future!
www.yfrobot.com

光立方制作手册

（YFRobot Cube8 LedDisp）

编著：产品部



2013 年 12 月 3 日

目 录

1 绪论

重磅升级的我来啦，现在的我只需要 3 根控制线就可以控制了，使得控制更加灵活，更加节约控制端口。在驱动板的底部设计了氛围灯，接通电源时就可以点亮。丰富了动画，完成了频谱显示，缺点就是程序没有自适应能力，信号过弱时频谱跳跃不明显，在今后中我们将重点升级。

焊接方法与 V1.0 类似，在 8 个“面”都焊接好后，将 64 个个镀金圆孔 PIN 焊如 L1~L8 中，将每个“面”的下部引脚剪切一样长，这样可以保证每个“面”的高度一致。

套件清单：

1、驱动板；

2、一包灯珠（530 个）；

3、10 根杜邦线（两头母）；

4、M3*8mm 螺丝，8mm、30mm 铜柱各 4 个；

5、20 针直排针；

6、灰色细导线（60cm）；

7、64 个镀金圆孔 PIN;

8、4 个 3mm 红发红 LED 灯；

（送：1、音频线、一分二音频头，各一个；2、列子。）

2013 年 12 月 3 日

1.1 V1.0 绪论

我们设计了一种全新的焊接方式，不需要额外的模版，只是利用我们的驱动板，和几个单排针，就可以焊接出四方四正的一个面，看完第二章，一定会给你耳目一新的感觉。设计驱动板的同时，我们还考虑了板子的通用性，可以使不同的控制器来驱动光立方，同时减少控制端口，现在是 8 个端口，在今后的升级中我们会再次减少控制端口。

驱动板上从左往右依次有 3.5mm 音频插座，5.5*2.1mmDC 插座，电源开关，电源指示灯，红外接收头，两个按键，8 个数据控制端，2 个按键信号输出端，红外信号输出端，音频模拟信号采集端。

具体的焊接方式、控制方式、程序，会在下面几个章节中详细讲解。

套件清单：

1、驱动板；

2、一包灯珠（530 个）；

3、15 根杜邦线（两头母）；

4、8mm 螺丝，8mm、30mm 铜柱各 4 个；

5、20 针直排针；

6、灰色细导线（80cm）。

（送：1、音频线、一分二音频头，各一个；2、列子。）

2013 年 10 月 10 日

2 灯的搭建

这一章节中我们介绍了两种方法来焊接体，总体的思想是一样的，只是在弯引脚方式中有所不同，建议把这一整章看完后，选择合适方式，再开始灯的搭建。

把搭建的过程分为四个过程，点、线、面、体。

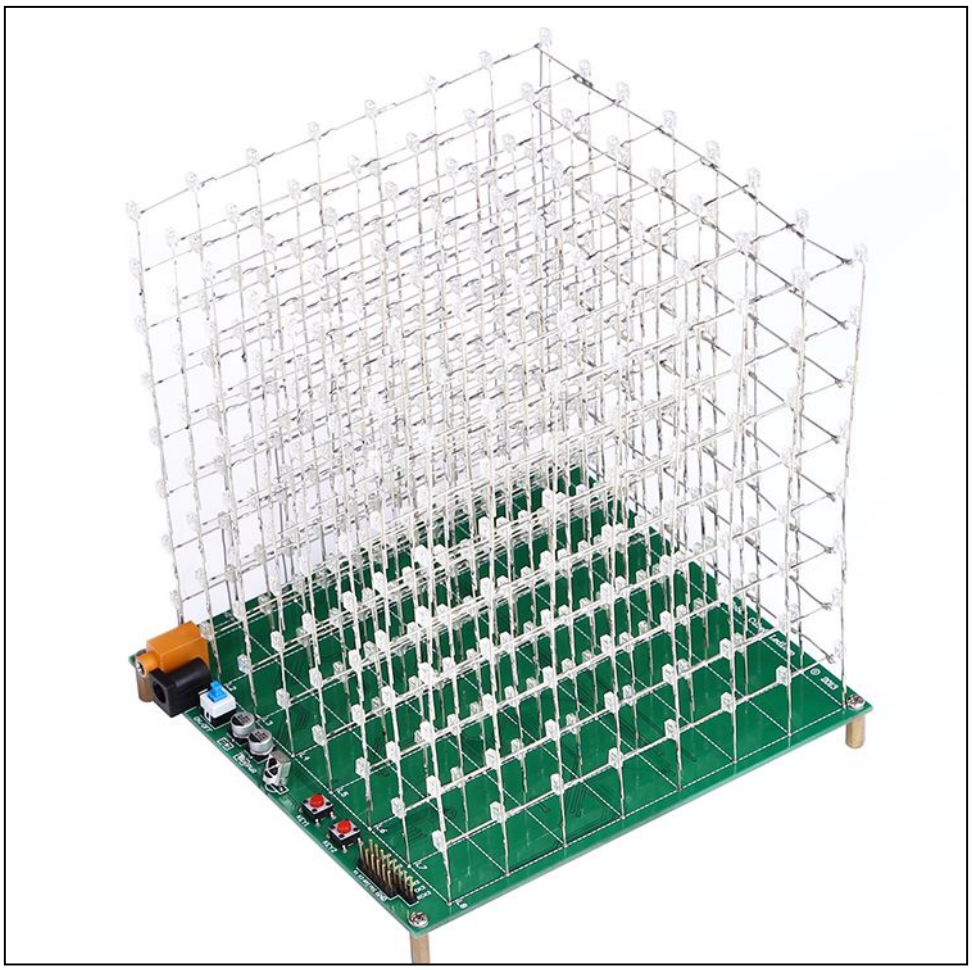


图 2.1 完整的光立方

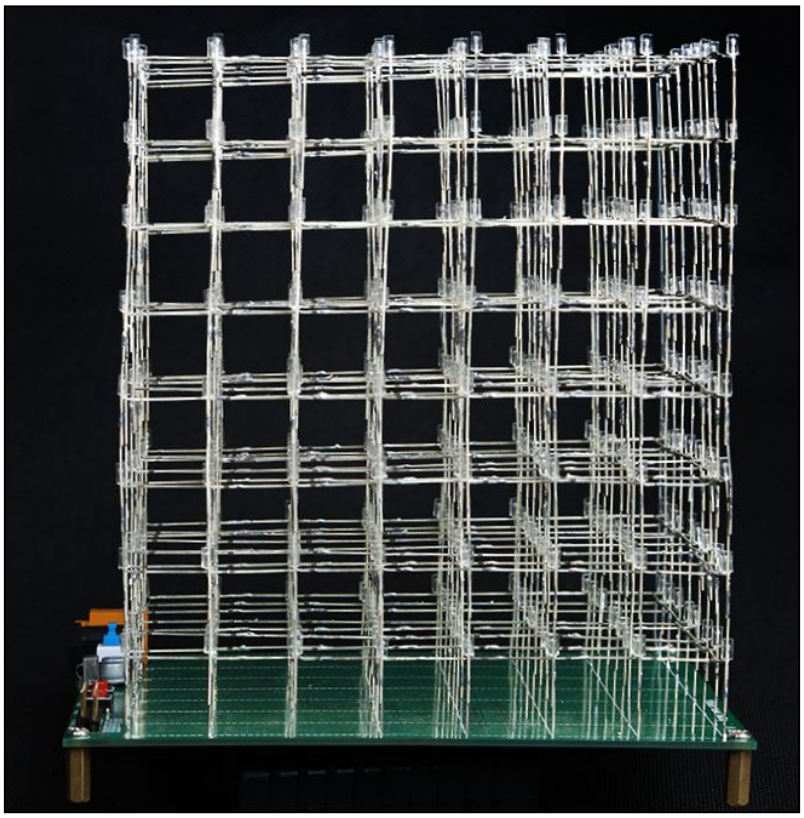


图 2.2　右视图（层与驱动板平行）

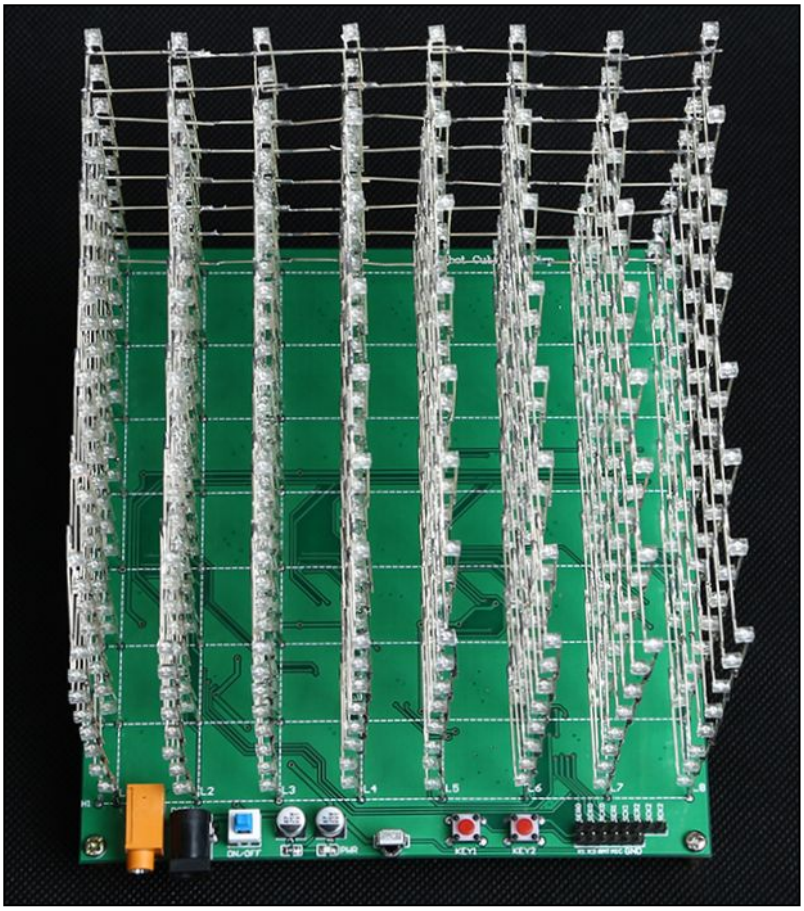


图 2.3　俯视图（面与驱动板垂直）

2.1 点

“点”就是灯，我们选择的灯是 2*3*4 高亮蓝色长脚雾灯，参数：VF：3.0~3.2；IV：550~650。引脚的弯曲非常的重要，它将直接影响线是否直，面是否方正水平。

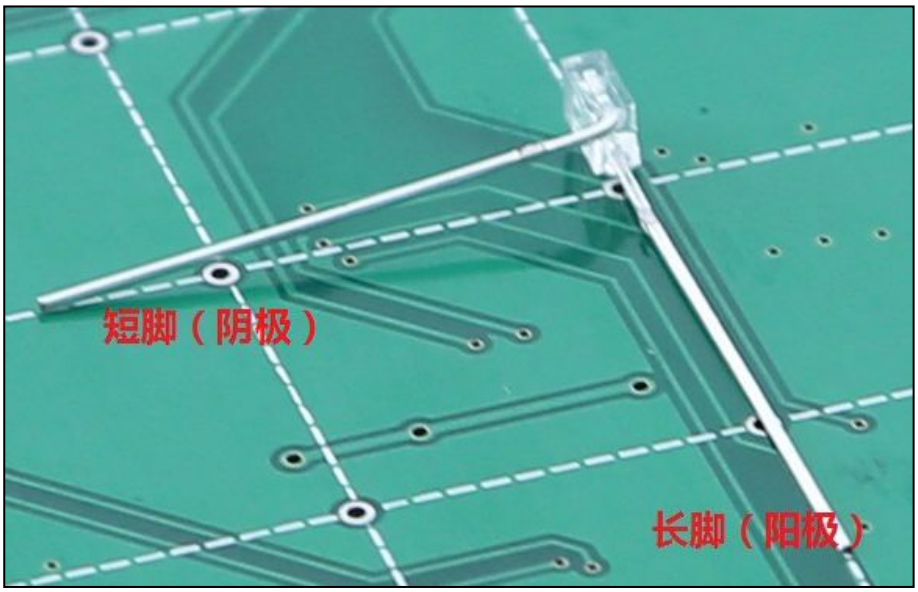


图 2.1.1 弯灯引脚（ I 型）

弯灯的方式如图 2.2.1 所示，当你拿起灯看的时候，你看到的只有 90 度和 180 度，这是焊接出一个完美光立方的前提。以上图为标准，弯出 512 个灯，用我们送的列子作为工具，更加便捷。

2.2 线

“线”是由 8 个“点”组成。用 30mm 铜柱与螺丝，将驱动板垫高，将灯珠的长脚插在孔中。如图 2.2.1 所示：

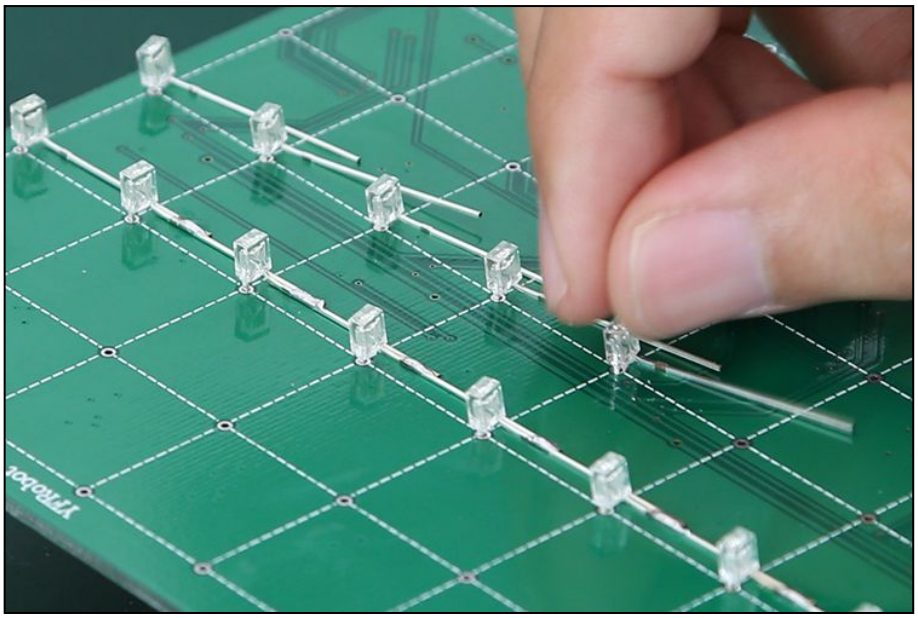


图 2.2.1　插灯脚

将 8 个灯珠插好后，将灯的短脚排列整齐，引脚需紧靠在一起。如图 2.2.2 所示：

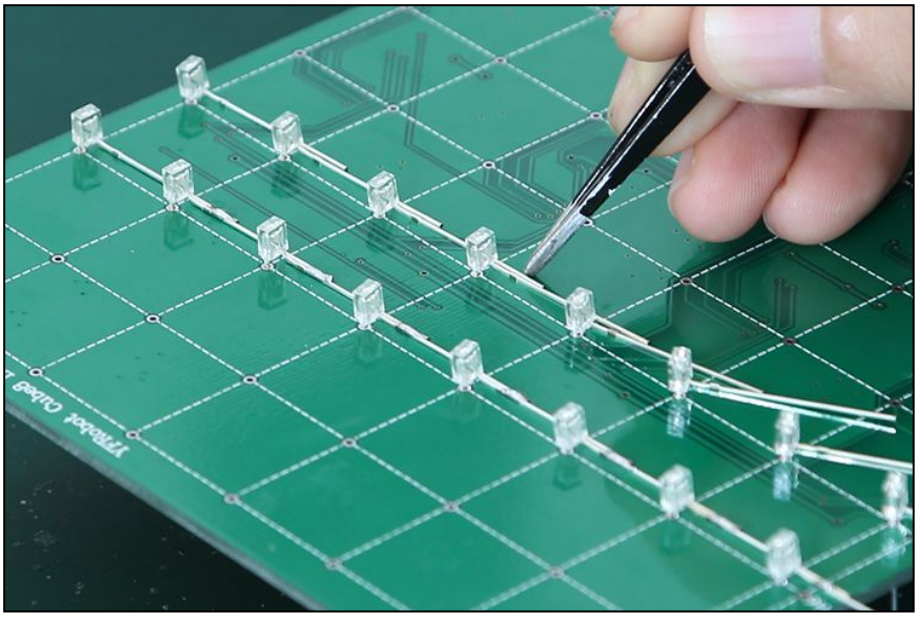


图 2.2.2　引脚排列整齐

引脚排好后，就开始焊接了（如图 2.2.3），焊接的时候尽量做到焊点小，不虚焊，当立方体焊成后，虚焊处理起来就麻烦了。

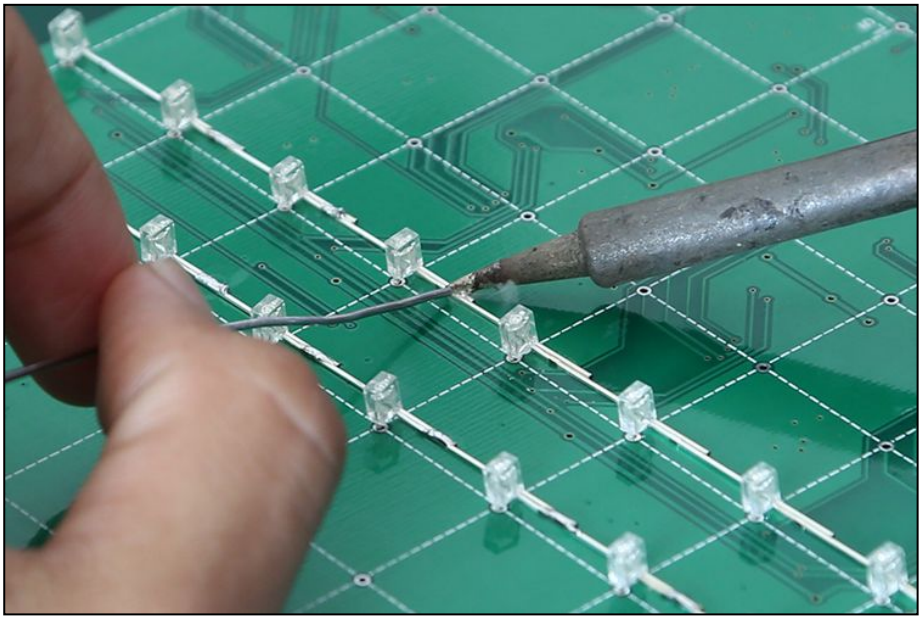


图 2.2.3　“线”焊接

一条“线”焊好后，它的引脚以任何角度看都是 90 度或 180 度，焊接“面”的时候您就会体会到它的好处了。就以这种方式焊接好 64 条“线”。

2.3　面

将 8 条“线”焊接一起就成为一个“面”。先在控制板上插些单直排针，起

固定中用，使得“线”与“线”保持平行，并使得每条“线”之间距离相等。如图 2.3.1：

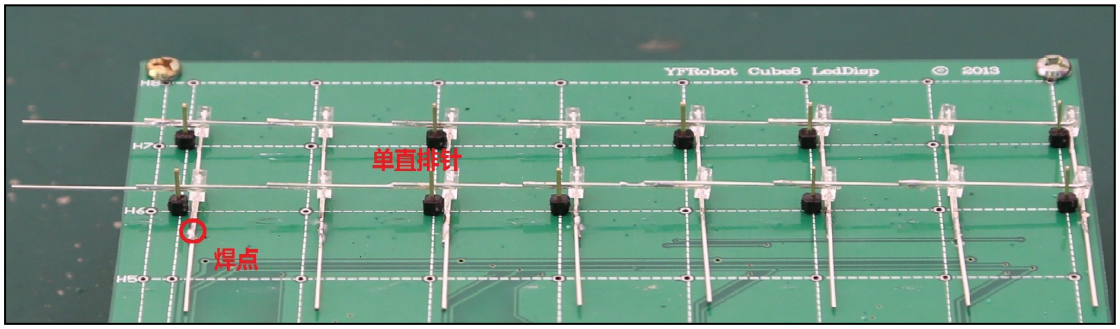


图 2.3.1 “线”动成“面”

如上图所示，将 8 条“线”焊接在一起，这样就可以成“面”了，焊好后将上图中左面长出的引脚向下弯曲垂直于“面”（此引脚为阴极，在“体”焊接的时候，需将每一层的阴极脚相连，使得每层可分别控制）。

测试是非常重要的环节，如图 2.3.2 所示，用红圈标出的引脚为被弯曲的引脚。“面”焊接好后，用 3V 电源测试，正极接在方框标出的那排第一个引脚，负极接在圆圈标出的那排任意第一个引脚，看对应的灯是否亮，不亮，检测是否虚焊、断路，再检测是否是因为灯的本身质量问题。我们一列一列检测，检测每个灯是否都能正常点亮。

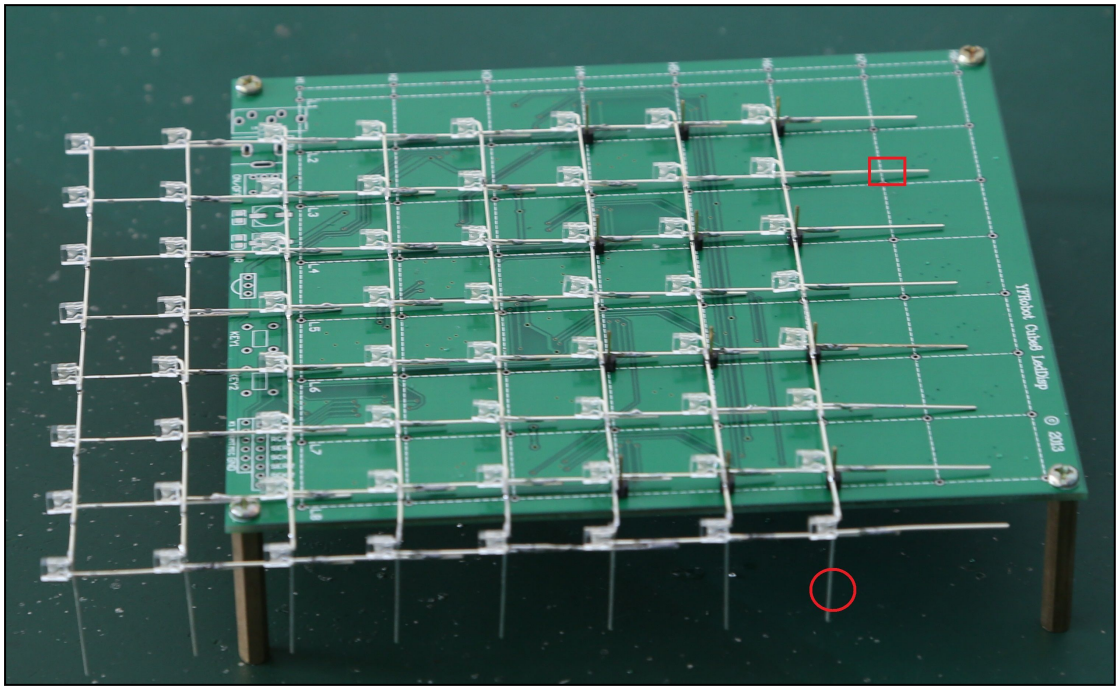


图 2.3.2 “面”

2.4 体

在本章开头我们已经将“立方”进行了图片展示，

2.5 搭建方法二

在这里我们介绍另外的弯引脚方式，我个人认为这样弯曲后焊接的“体”，后面看的视觉效果会更好些，就是阴极连接的时候会相对麻烦些。上图说话：

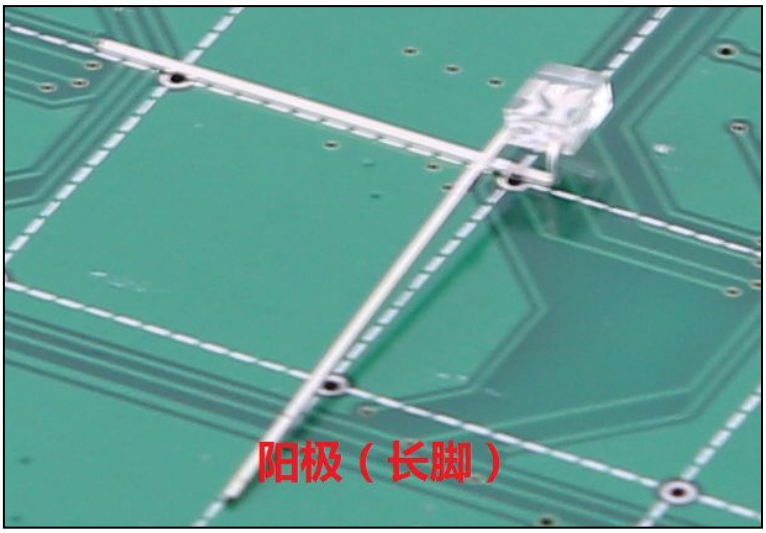


图 2.5.1 弯灯引脚（II 型）

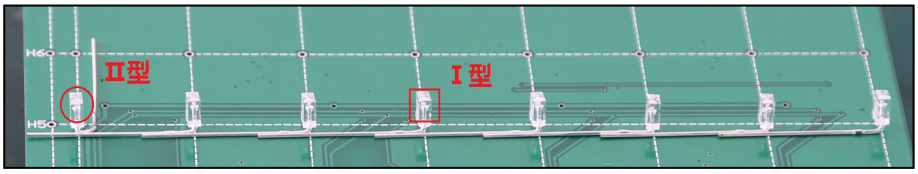


图 2.5.2 引脚焊接方式

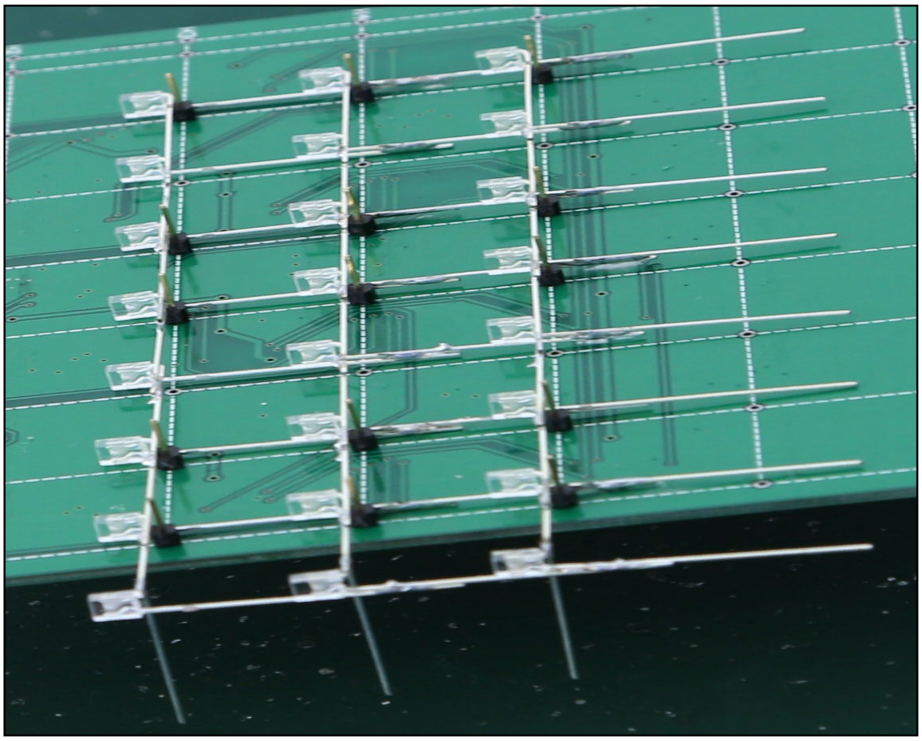


图 2.5.3 面的焊接

这种“面”焊好后就不需像第一种方法那样弯引脚了，我们制作的样品就是

以这种方法来焊接的，在下图（图 2.5.4）中可以明显的看出，这样焊接阴极的连线是在“立方体”的里面。

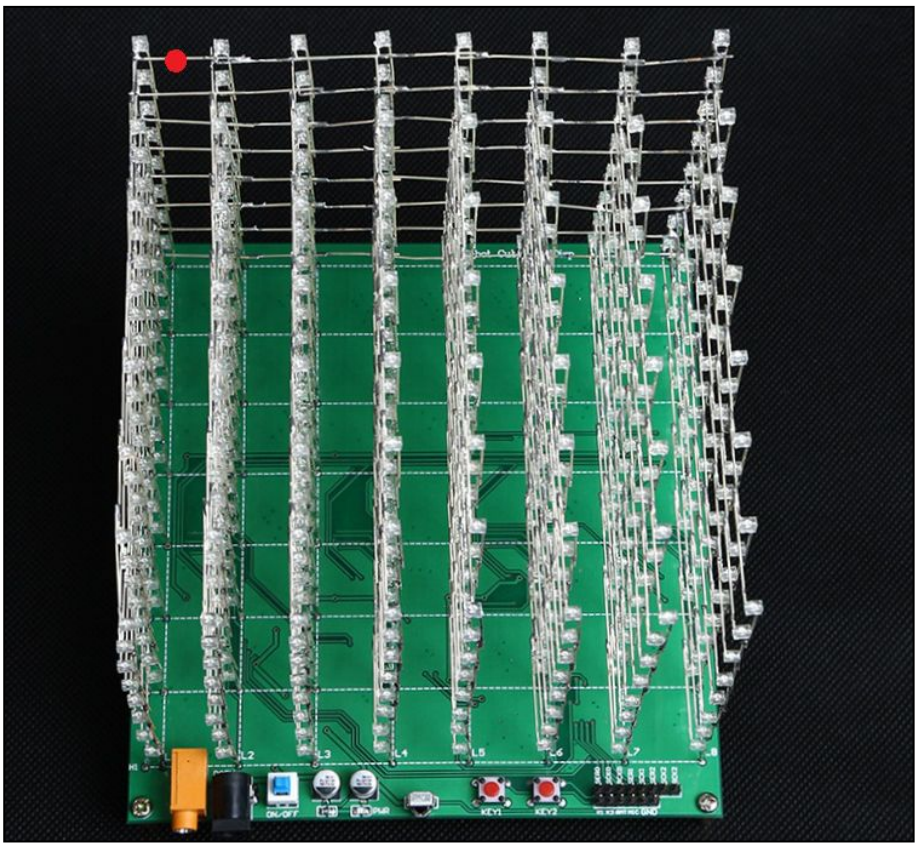
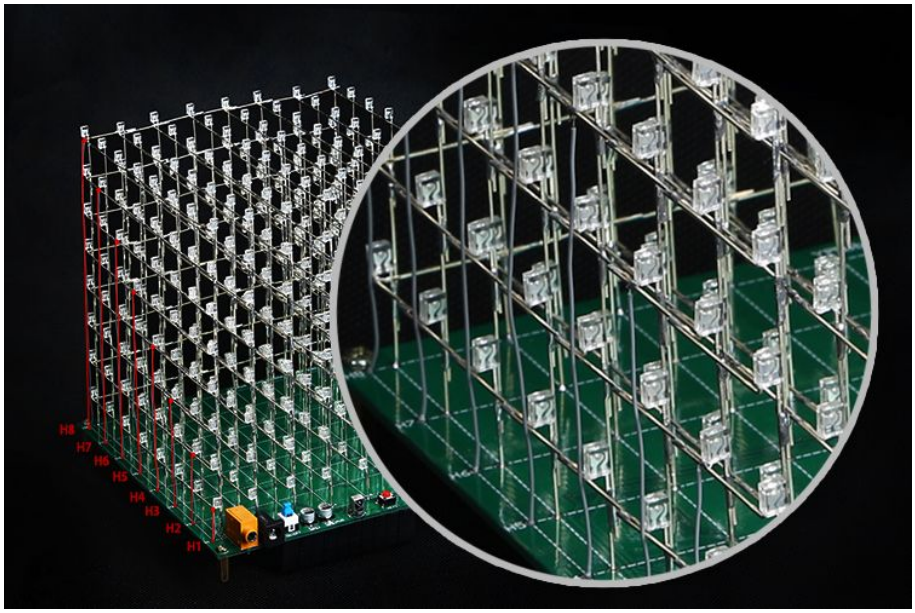


图 2.5.4　阴极连接线

2.6　层控制线

以安装示意图的方式将层控制线焊好就 OK 了。



2.6.2　层控制线

3 光立方驱动板

驱动板上用到的几个芯片具体使用方法请看“光立方制作手册 V1.0\芯片资料”，驱动板原理图请看“cube8sch.pdf”。

3.1 坐标系

建立“体”的空间坐标系，如图 3.1.1。

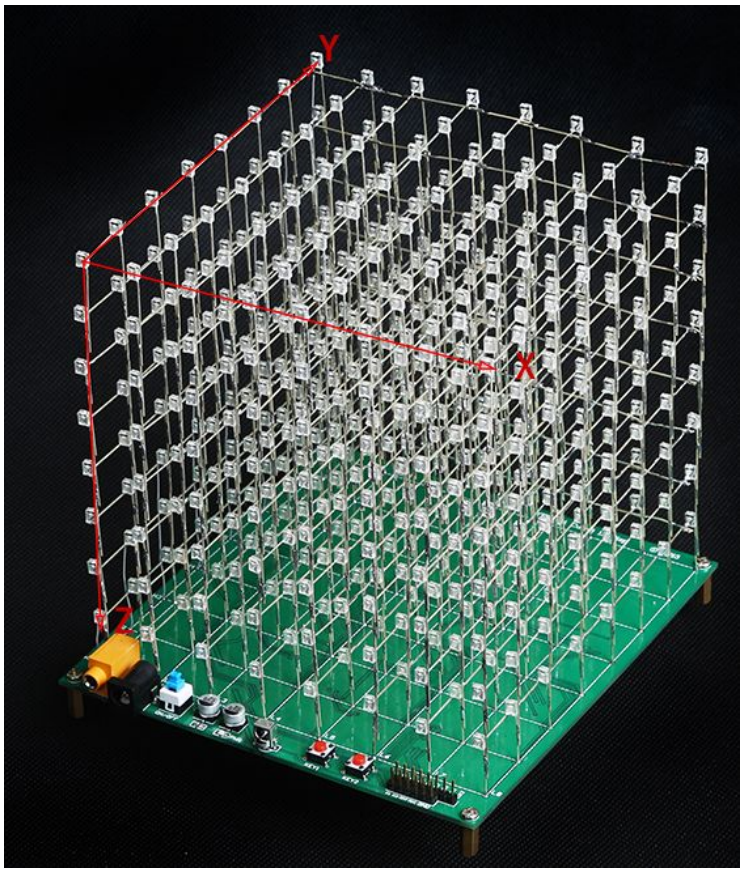


图 3.1.1 建立空间坐标系

Z=0 为第 1 层、Z=1 为第 2 层、•••Z=7 为第 8 层，在每一层中有 8 束光（就是我们前面焊接的“线”），一个立方共有 64 束，我们将他们分别编号为 0、1、3、••• 63（图 3.1.2），每束光由 8 个灯组成。例如，第一层，我们分别编号 0、1、2、••• 7。在 X=0 这个面，有 0、8、••• 56 这 8 束光，这 8 束光由 U4（74hc595）控制，每束光由 8 个灯组成，(0,0,0)灯由 U4->QA 控制、(0, 1, 0)灯由 U4->QB 控制、••• (0, 7, 0)灯由 U4->QH 控制。

我们数组的编号方式正好对应着数组中数，one[0]控制这光束 0 上面的 8 个灯，a[0]=0x01，用二进制表示就是 00000001，控制（0，0，0）这个灯被点亮。如果 a[1]=0x88，用二进制表示就是 10001000，控制灯（1，3，0）和（1，7，0）两个灯亮，这样编号的优点，您会在今后的编程过程中慢慢体会到。

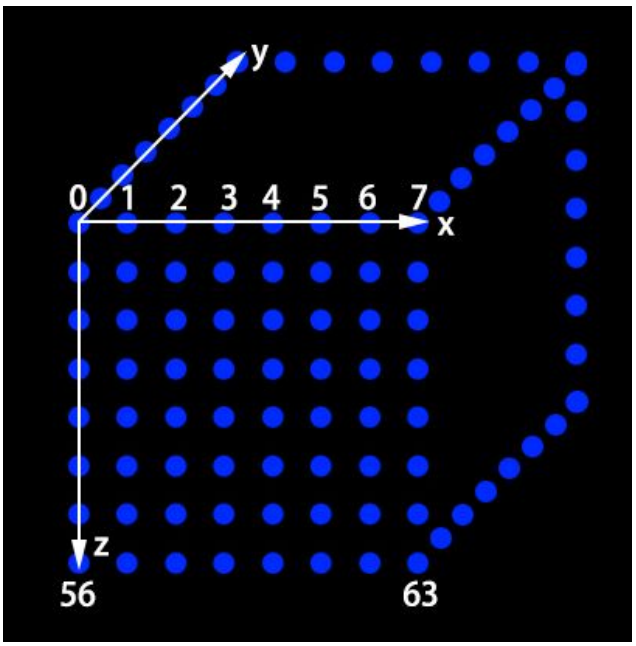


图 3.1.2　坐标系

例如一个数组

u8 one[]={

ox01, ox01, ox01, ox01, ox01, ox01, ox01, ox01,

ox01, ox01, ox01, ox01, ox01, ox01, ox01, ox01,

ox01, ox01, ox01, ox01, ox01, ox01, ox01, ox01,

ox01, ox01, ox01, ox01, ox01, ox01, ox01, ox01,

ox01, ox01, ox01, ox01, ox01, ox01, ox01, ox01,

ox01, ox01, ox01, ox01, ox01, ox01, ox01, ox01,

ox01, ox01, ox01, ox01, ox01, ox01, ox01, ox01,

ox01, ox01, ox01, ox01, ox01, ox01, ox01, ox01,

}

这个数组选择的是 Y=0 这个面的灯“同时”亮。“同时”我们加双引号，实际上在某一时刻只有一层灯是被点亮，对 8 层灯进行扫描，利用人眼的视觉停留效果，给人一种同时亮的错觉。

3.2　控制方式

U4、U5、U6、U7、U10、U11、U12、U13 灯束控制，选择每束光中不同的灯。

U15 控制 8 个 595 的存储寄存器输出并锁存。

U8 层控制，选通某一层，是某一时刻，只有一层灯亮，U9 是 ULN2803 驱动芯片，输入为 1 时，输出为 0。

4 库语法介绍

我们的程序是从 建立在控制器为“YFstm32 核心控制板 V1.0”的基础移植过来的，程序为 C 语言编程，兼容 Arduino，我们将其整理成库文件“LightCube”，方便大家调用。

下面我们来介绍下函数库的语法使用：

1、LightCube()：端口初始化

语法：

LightCube lightCube(int SER0 ,int SCK0 ,int RCK0)

参数：

SER0：Arduino 引脚编号，串行数据输入

SCK0：Arduino 引脚编号，输入到移位寄存器

RCK0：Arduino 引脚编号，上升沿是串行输入，并行输出并锁存

2、storey()：层填充函数，控制某层灯点亮方式

语法：

lightCube.storey(*a)

参数：

a：一帧，a 是一帧编码起始地址

3、frame()：显示函数

语法：

lightCube.frame(a,v)

参数：

a：一帧，a 是一帧编码起始地址

v：表示一帧画面扫描的次数

4、clc()：全局清屏

语法：

lightCube.clc()

参数：

无

以下为我们为大家提供的动画效果函数：

5、All()：全局清屏

语法：

lightCube.All(v)

参数：

v：表示一帧画面扫描的次数

6、dot()：依次点亮 00-1,01-1,02-1,03-1....63-1

语法：

lightCube.dot(v)

参数：

v：表示一帧画面扫描的次数

7、层层点亮显示效果

语法：

```
lightCube.LtoR_scan(v);   //侧面从左向右依次点亮
lightCube.RtoL_scan(v);   //侧面从右向左依次点亮
lightCube.FtoB_scan(v);   //从前排向后排依次点亮
lightCube.BtoF_scan(v);   //从后排向前排依次点亮
lightCube.UtoD_scan(v);   //从上向下依次点亮
lightCube.DtoU_scan(v);   //从下向上依次点亮
```

参数：

v：表示一帧画面扫描的次数

8、层旋转显示效果

语法：

```
lightCube.Acw(num, v);     // anticlockwise 逆时针
lightCube.Cw(num,v);       // clockwise 顺时针
lightCube.RCw(num,v);      //右侧看顺时针参数：
```

参数：

num：旋转周数

v：速度

9、其他动画显示效果

1）语法：

lightCube.cube(empty,kind,v);

参数：

empty = 0 空，1 实

kind = 0 左上角，1 右上角，2 左下角，3 右下脚

2）语法：

lightCube.rain(menu, num, v);

参数：

menu：0 -- down

1 -- up

num：循环次数

3）语法：

lightCube.up(num,v); //上移

参数：

num：数量

v：速度

4）语法：

lightCube.xuanzhuantiao(kind,cw,num,v);

参数：

kind：1 实心，否则空心

cw：是否正转

5）语法：

lightCube.qumianxuanzhuan(cw, num, v);　　曲面旋转效果

参数：

cw ：1 逆时针，否则顺时针

6）语法：

lightCube.sandglass(s8 v);　　沙漏效果

参数：

v：速度

例程中给出所有显示效果，我们也在努力更新，希望能给大家提供更好的体验~~~

5 程序理解

下面为库文件中重要的几个函数（有兴趣可以理解下，然后可以亮出自己想要的画面）：

端口定义函数：

SER0 --- 串行数据输入

SCK0 --- 输入到移位寄存器

RCK0 --- 上升沿是串行输入，并行输出并锁存

根据端口设置，将控制器输出端口对应接到驱动板的 SER、SCK、RCK 上，控制板不驱动板用不同电源时，控制板上 GND 与驱动板上 GND 相连。

5.1 层填充函数

```
void storey(u8 *a)//层填充函数，控制某层灯点亮方式
{
    u8 i,j,num;
    for(i=0;i<8;i++)
    {
        num=a[i];           //将数组中数输入寄存器
        for(j=0;j<8;j++) //串行数据输入
        {
            if(num&0x80)
                SER0=1;       // SER 串行输入端口
            else
                SER0=0;

            SCK0=0;           //上升沿，输入到移位寄存器
            delay_us(1);
            SCK0=1;
            num<<=1;
        }
    }
```

}通过上面的函数，我们就可以将 8 个寄存器分别写入数值，可以控制某一层灯的点亮方式。

5.2 帧函数

```
void frame(u8 *a,u8 v)//一帧,a 是一帧编码起始地址
                      //一个画面，v 表示一帧画面扫描的次数
                      //可以看作这帧显示的时间
{
    s8 i,j,num;   //s8  有符号定义
    while(v--)
    {
        num=0x01;
        for(i=0;i<8;i++) //层数层控制，选通某一层，
                         //使得第 1 层到第 8 层，依次点亮
        {
            num<<=i;
            RCK0=0;
            for(j=0;j<8;j++) //串行数据输入
            {
                if(num&0x80)
                    SER0=1;        // SER 串行输入端口
                else
                    SER0=0;
                SCK0=0;            //上升沿，输入到移位寄存器
                delay_us(1);
                SCK0=1;
                num<<=1;
            }
            storey(a+i*8);//层填充函数，控制某层灯点亮方式
            RCK0=1;
            num=0x01;
            delay_ms(2);     //  层显示时间
        }
    }
}
```

帧——就是影像动画中最小单位的单幅影像画面，相当于电影胶片上的每一格镜头。一帧就是一副静止的画面，连续的帧就形成动画，

5.3 动画

动画内容是：左侧面灯全亮，并从左向右依次点亮。

```
void LtoR_scan(u8 v) //侧面从左向右依次点亮
{
    u8 b[64]={0};
    s8 x,z;
    for(z=0;z<8;z++)
    {
        b[z*8]=0xff;
    }
    frame(b,v);
    for(x=1;x<8;x++)
    {
        for(z=0;z<8;z++)
        {
            b[z*8+x]=0xff;
            b[z*8+(x-1)]=0;
        }
        frame(b,v);
    }
}
```

利用帧函数，发挥自己的想象，写不同的数组，就可以制作出属于自己的动画了。

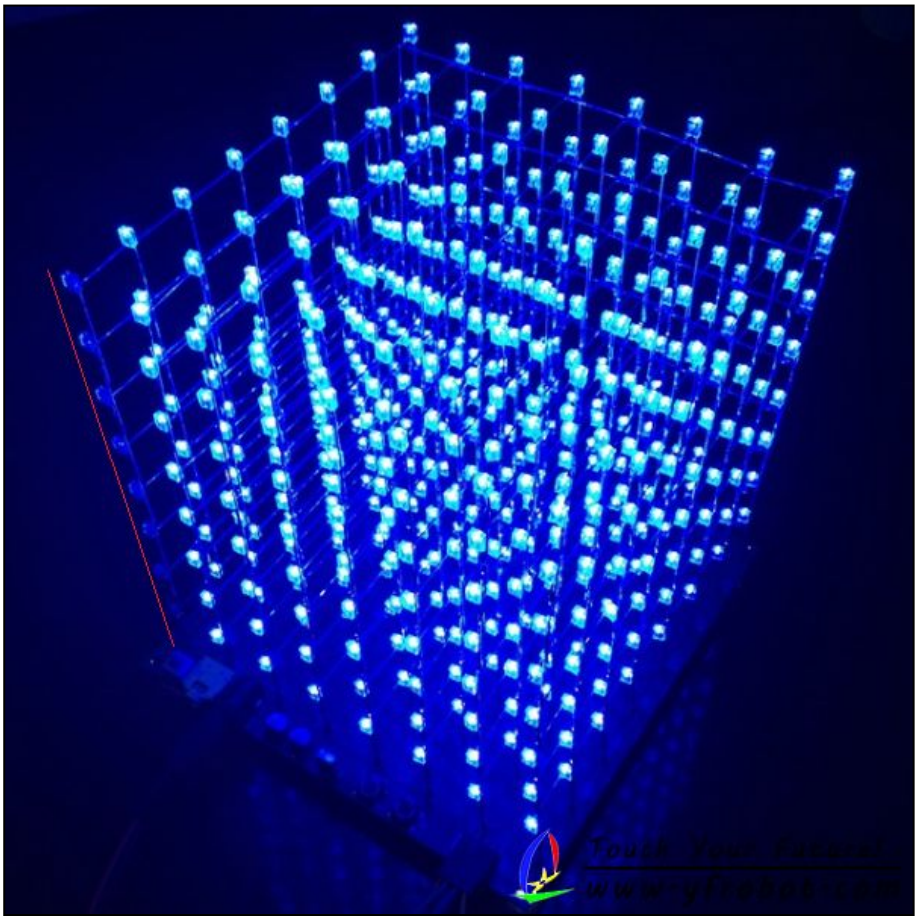
6 疑难排解

光立方是一个焊接的大工程，在焊接的时候难免会出现虚焊的问题了，此贴将贴出一些常见问题，帮助大家快速找到问题。

在制作手册中我们已经提到，在“面”焊接好后，我们要对“面”进行检测，检测是否有虚焊状况或坏灯现象，及时处理，“面”检测好后在进行“体”的焊接，“体”焊接好后，我们再进行“全身体检”。

运行“test”例程，使光立方的灯全亮，根据灯亮的状况，来寻找问题点。以下为大家列出 4 种可能出现的问题；

情况一：一列灯未亮

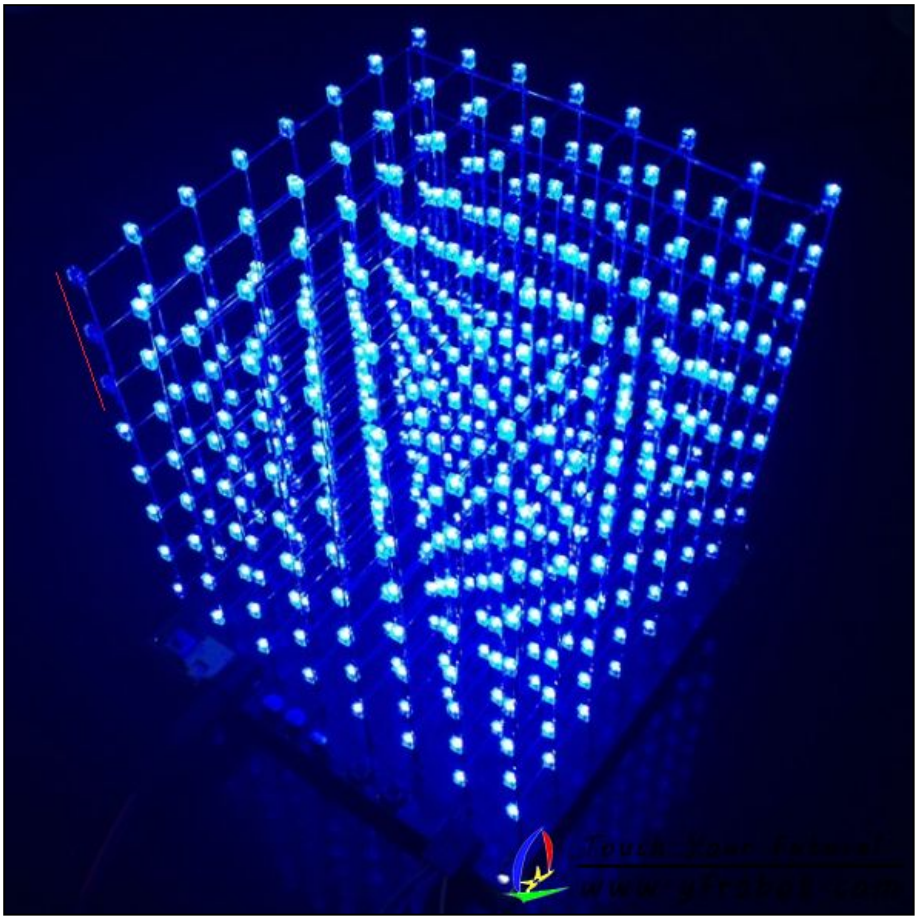


如上图所示，X=0，Y=0 这列灯未亮。

1：检查这列灯的下方引脚是否插入圆孔 PIN 中；

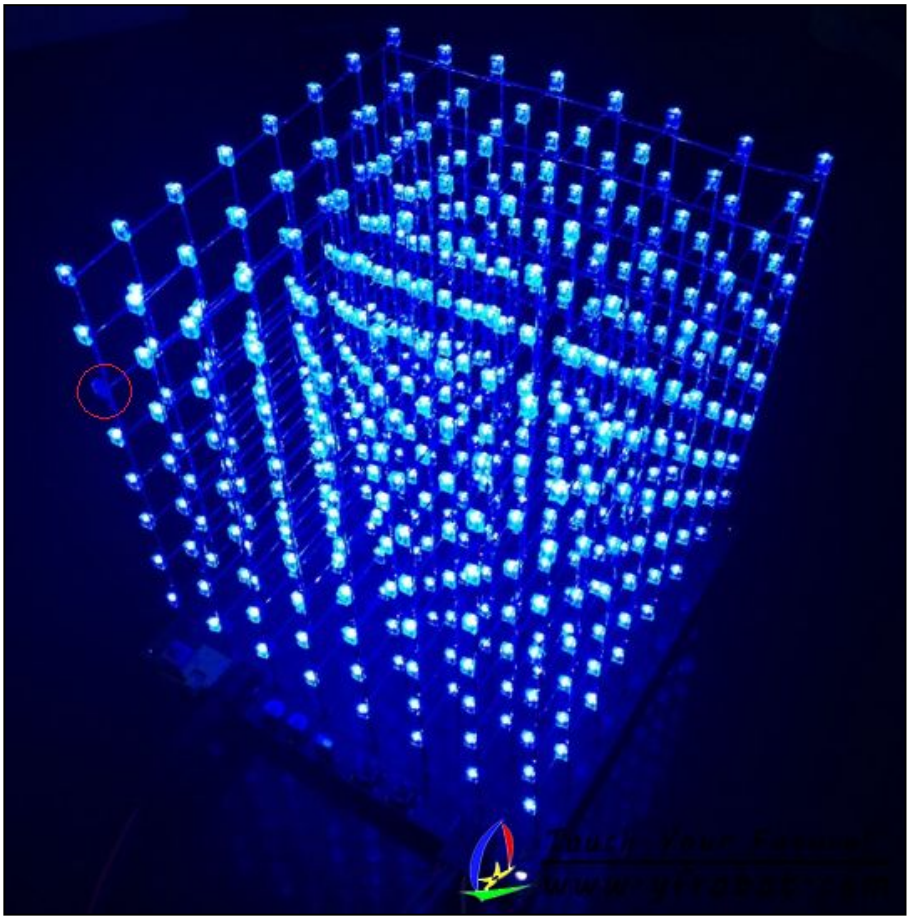
2：检查控制这列灯的 595 是否有虚焊、串联的电阻是否有虚焊。

情况二：“三”个点未亮



如图，可以看到一列灯的上面三个灯未亮。出现这种状况的只有一个原因，灯（0，0，2）与灯（0，0，3 ）之间虚焊。

情况三：单个灯未亮



如图，一个灯未亮，在这种状况下，还是先保佑灯没坏吧。

1：检测不亮的灯引脚与周围灯连接是否正常；

2：确保引脚焊接正常后，再 次负 full bright，如果灯还是未亮，关闭电源，拔掉控制线，用 3.0（不可超过 3.3 伏）左右的电源直接接在不亮灯的引脚上，看灯是否能被点亮，还是未亮，确定灯已坏，只有换灯了。

情况四：一层灯未亮

1：检测层控制线与驱动板是否有虚焊；

2：检测 2803 驱动这层灯的引脚是否有虚焊状况，以及 2803 与 595 之间是否有虚焊。

以上为最常见的状况，如有其它状况，可以论坛 www.yfrobot.com 请留言，大家一起解决。