物联网智能小灯

设计说明书

版本号	作者	修改日期	修改内容
V1.0	胡健、张凡、肖责	2017.11.13	对总体设计和功能模块设计做了必要的说明

一、 作品简介

选题	智能家居
功能说明	将全彩 LED 灯连入互联网，并通过手机客户端对其进行控制
设计思路	灯是家居中不可或缺的元素，如果给所有的灯配备一个可以“思考”的组件，比如语音控制、定时开关、自动调整亮度等，再如果这些灯可以接入互联网，那我们就可以随时随地的控制家里的灯了。

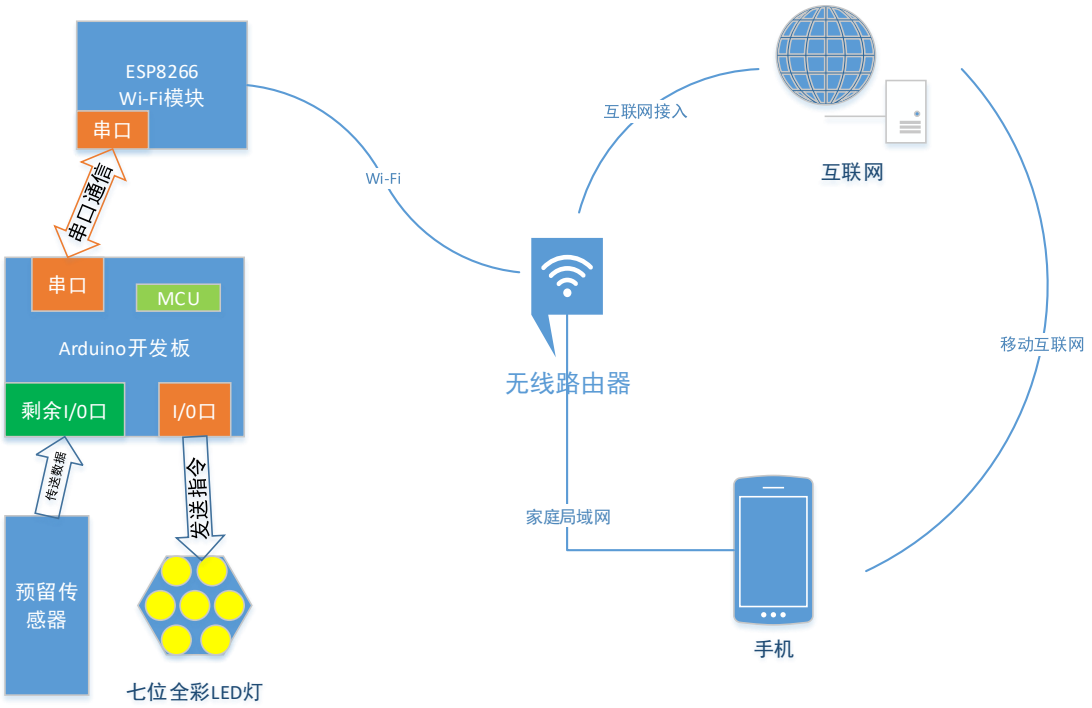
1.1 作品创意

①将全彩 LED 灯连入互联网，可通过手机实时对其操控（包括开关、调整色彩和亮度、定时关开灯等等），改善人与家庭设备的交互。

②操作简单。布置简单，连接方式采用 Wi-Fi,只需有家庭无线路由器，便能让智能灯连入网络，无需其他网关。配对方便，采用 UDP 广播方式，不需复杂的操作就能完成手机 APP 和设备的配对。

③可拓展性强。MCU 组件预留了很多接口，接入传感器后能实现语音交互，并能让设备通过感知周围的环境自行调整自身的状态，实现更友好的智能交互。

1.2 总体设计图

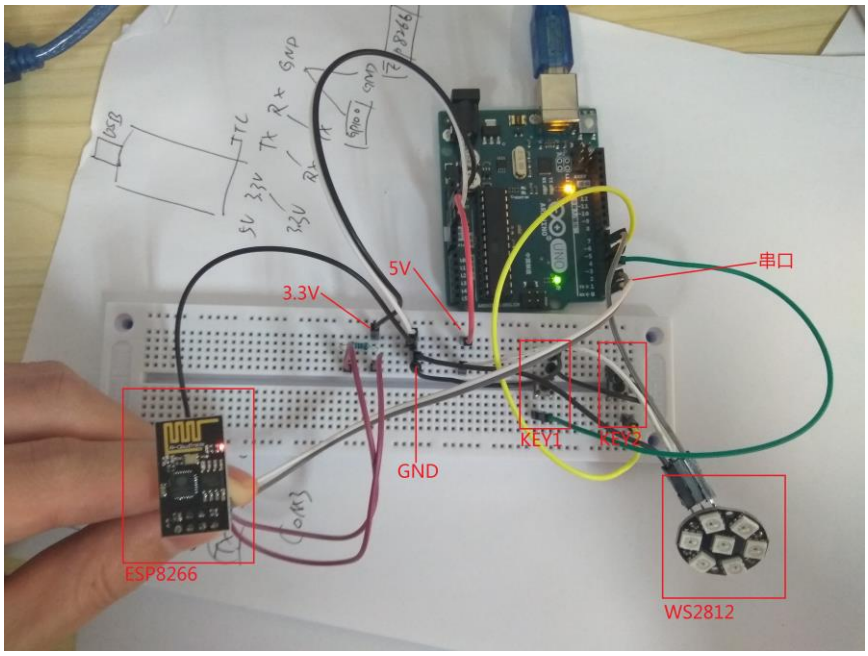


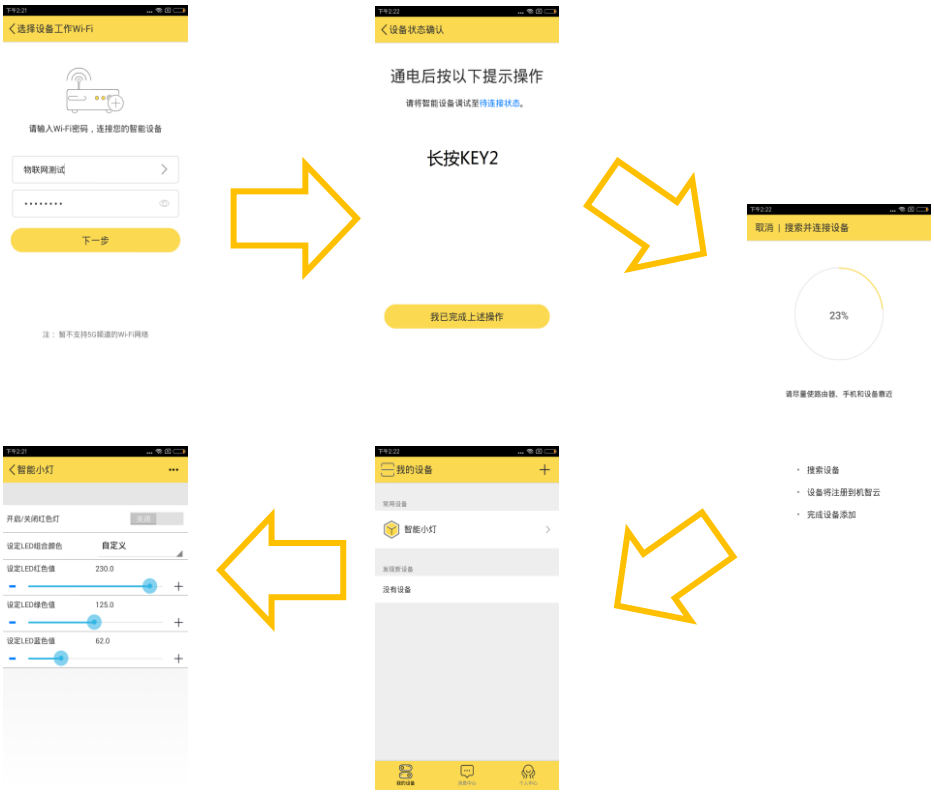
1.3 操作说明

第一步：首次使用需要完成设备配对，给 Arduino 开发板上电，安卓手机安装“智能小灯”APP 并启动，选择添加设备；

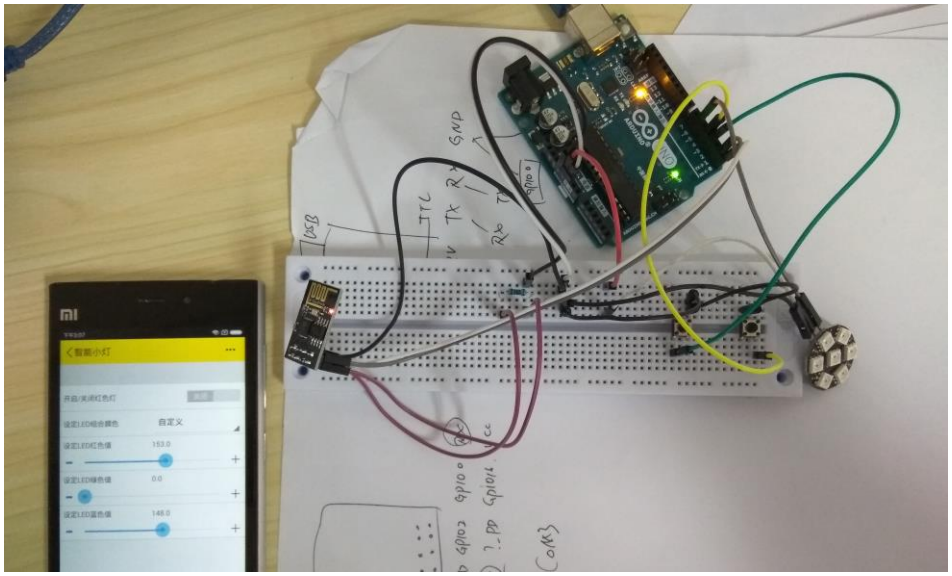
第二步：按 APP 提示，长按 KEY2 3s 以上使设备进入配对模式后，APP 点击已完成进行配对。

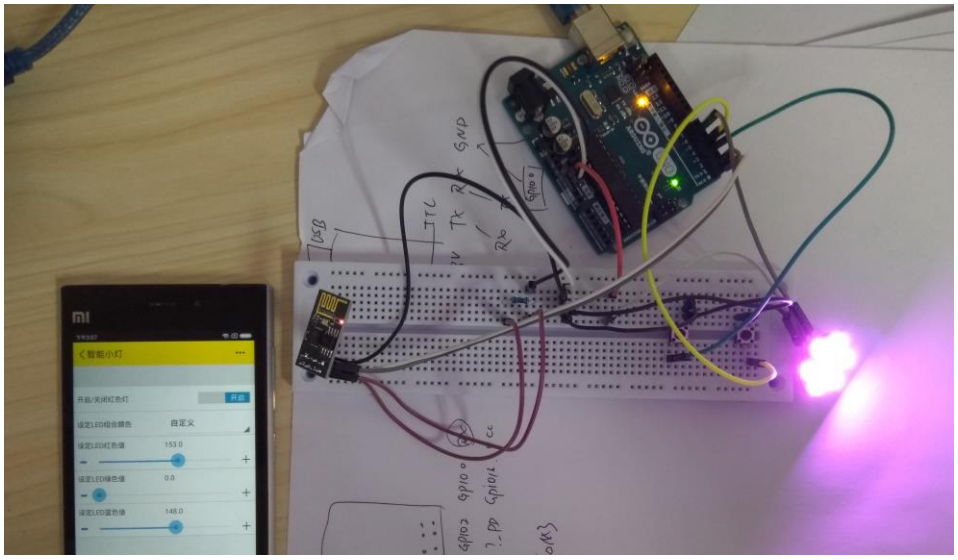
第三步：配对完成后，进入智能小灯的控制界面（下图为目前已经实现的功能，更多的功能还在开发中）。

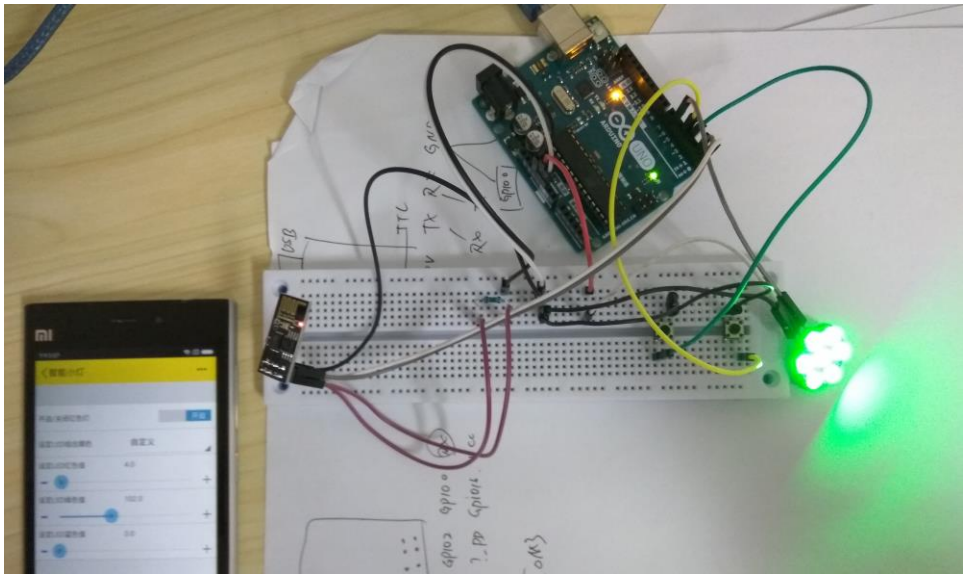




演示结果如下图：







二、各部分设计说明

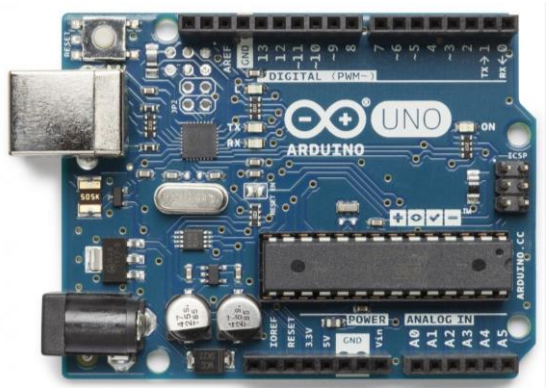
2.1 MCU 部分

MCU 采用的是 Arduino UNO R3 开发板，板载 MCU 为 Atmel Atmega328P-PU 单片机，开发环境为其对应的 ArduinoIDE，

使用 C++来编写程序。

开发板的串口（TX/RX）与 ESP8266 的串口相连，PIN 6 与 WS2812LED 的 DIN 相连，同时给二者供电，PIN 3 和 PIN 4f 分别接了一个按键开关，用于控制 ESP8266 的状态，如下表

5V	LED-VCC
3.3V	ESP8266-VCC
GND	LED-GND,ESP8266-GND
TX	ESP8266-URXD
RX	ESP8266-UTXD
PIN 6	LED-DIN
PIN 3	KEY1
PIN 4	KEY2



MCU 主要完成两方面的工作：

①完成与 ESP8266 的串口通信，这里使用了机智云提供的 Wi-Fi 类设备接入协议函数库；

②驱动 WS2812LED，根据数据手册，编写了相应的驱动类及成员函数；

③按键检测程序

下图为调用的库以及用类建立的实例：

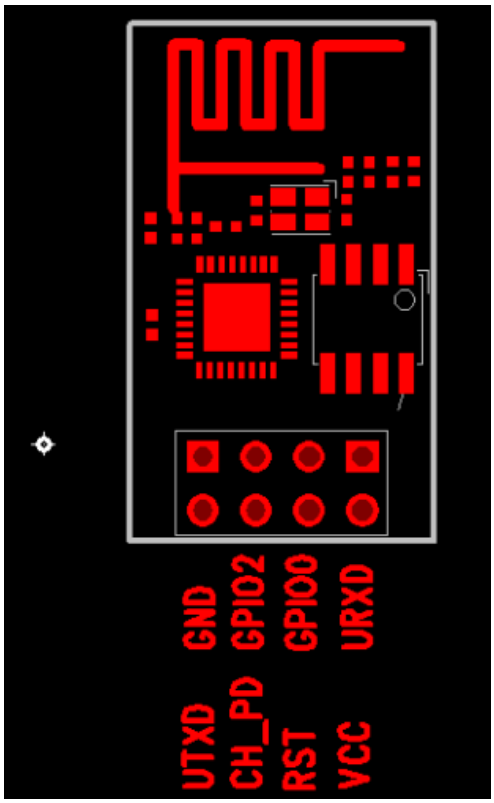
```
#include <Gizwits.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <Adafruit_NeoPixel.h>
```

```
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUM_LEDS,PIN,NEO_GRB+NEO_KHZ800);
SoftwareSerial mySerial(A2, A3); // A2 -> RX, A3 -> TX
Gizwits myGizwits;
```

2.2 Wi-Fi 模块

Wi-Fi 模块采用的是 ESP8266-01 模块。ESP8266 是一款超低功耗的 UART-WiFi 透传模块，拥有业内极富竞争力的封装尺寸和超低能耗技术，专为移动设备和物联网应用设计，可将用户的物理设备连接到 Wi-Fi 无线网络上，进行互联网或局域网通信，实现联网功能。

接口定义如下：



VCC	3.3V，模块供电；
RST	外部 Reset 信号，低电平复位，高电平工作（默认高）；
CH_PD	1)高电平工作； 2)低电平模块供电关掉；
UTXD	1)UART_TXD，发送； 2)GPIO1； 3)开机时禁止下拉；
URXD	1)UART_RXD，接收； 2)GPIO3；
GPIO0	1)默认WiFi Status：WiFi工作状态指示灯控制信号； 2)工作模式选择： 上拉：工作模式； 下拉：下载模式；
GPIO2	1)开机上电时必须为高电平，禁止硬件下拉； 2)内部默认已拉高
GND	GND

要想让 Wi-Fi 模块能正常工作，还需要刷入相应的固件，本设计采用的是乐鑫官方烧写工具，烧入的固件为机智云提供的：

GAgent_00ESP826_04020025_8Mbit_201708301926_combine.bin

该固件是应用程序 user1 区域固件、boot1.6 固件、esp_init_data_default、blank 四个文件合一的固件，适用于 8Mbit flash 硬件。

我们将 ESP8266 的串口与 Arduino 开发板的串口相连，可实现将从 Wi-Fi 端接受的指令通过串口传递给 MCU，同时也接受从 MCU 串口传递过来的数据，并完成指令对应的操作。

2.3 LED 部分

这里使用了 7 位 WS2812LED 模块，WS2812 是一个集控制电路与发光电路于一体的智能外控 LED 光源。数据协议采用单线归零码的通讯方式，像素点在上电复位以后，DIN 端接受从控制器传输过来的数据，首先送过来的 24bit 数据被第一个像素点提取后，送到像素点内部的数据锁存器，剩余的数据经过内部整形处理电路整形放大后通过 DO 端口开始转发输出给下一个级联的像素点，每经过一个像素点的传输，信号减少 24bit。下图为 24bit 的组成：

Composition of 24bit data:

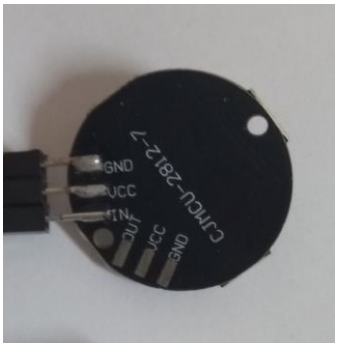
G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Follow the order of GRB to sent data and the high bit sent at first.

这里将该模块的 DIN 脚与 Arduino 的 PIN6 相连，通过 PIN6 将每次更新后的数据发送给 LED 模块。调用了 Adafruit_NeoPixel 库，实现方式如下

```
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUM_LEDS,PIN,NEO_GRB+NEO_KHZ800);
```

```
void setPixelColorAll(uint8_t r,uint8_t g,uint8_t b)
{
  for(int i=0;i<pixels.numPixels();i++)
  {
    pixels.setPixelColor(i,pixels.Color(r,g,b));
    pixels.setPixelColor(i,pixels.Color(r,g,b));
    pixels.show();
  }
}
```



2.4 按键部分

按键用于控制 Wi-Fi 模块的状态，功能定义如下：

KEY1	短按	Production Test Mode
	长按	Wifi Reset
KEY2	短按	Soft AP mode
	长按	AirLink mode

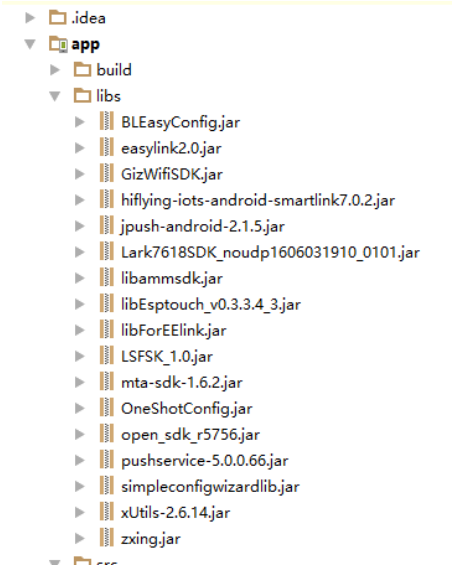
下图为按键处理函数：

```
void KEY_Handle(void)
{
  /*  Press for over than 3 second is Long Press  */
  switch (gokit_keydown())
  {
    case KEY1_SHORT_PRESS:
      mySerial.println(F("KEY1_SHORT_PRESS , Production Test Mode "));
      myGizwits.setBindMode(WIFI_PRODUCTION_TEST);
      break;
```

2.5 手机客户端部分

开发环境为 Android Studio,基于机智云提供的客户端开源框架二次开发，使用了该开源框架提供的大量类库。目前实现了对智能灯的开关和 RBG 值的读写，更友好的界面以及其他的功能还在开发中。

下图为开源框架提供的类库：



二次开发后编译成功：