一．ConfigWIFI 类

```
/**
 * 初始化ConfigWifi对象,返回一个 ConfigWifi 对象
 * @return
 */
1.public static ConfigWifi init();

/**
 * 初始化配置的目的IP和端口,用户手机在连接上控制设备后,需要将要连接的WIFI名和密码
 * 发送给设备,然后手机也连上WIFI让设备也连上WIFI,这样设备和手机都在同一局域网下.
 * 函数调用后返回一个 ConfigWifi 对象
 * @param Server_IP 手机先连接设备,获取设备的IP地址
 * @return ConfigWifi
 */
2.public static ConfigWifi initWifi(String Server_IP,int Server_Port);

/**
 * 初始化配置的目的IP,建议使用此函数,使用默认端口,用户手机在连接上控制设备后,需要
 * 将要连接的WIFI名和密码发送给设备,让设备连上WIFI,然后手机也连上WIFI
 * 这样设备和手机都在同一局域网下. 函数调用后返回一个 ConfigWifi 对象
 * @param Server_IP 手机先连接设备,获取设备的IP地址
 * @return ConfigWifi
 */
3.public static ConfigWifi initWifi(String Server_IP);

/**
 * 初始化IP和端口之后,扫描WIFI(也可以手动输入),将wifi名和密码发送到站点
 * 让设备连上WIFI,让手机和设备在同一局域网下
 * @param SSID 获取的即将要连的WIFI名
 * @param Password 即将要连的WIFI的密码
 */
4.public abstract void config(String SSID,String Password);

/**
* 启动监听UDP消息的线程,参数handler需要自己写的
* 当监听的是wifi配置是否成功的消息时,从Bundle中获取字符串方法:
*   getString("config_receive")
* 当监听的是控制开关是否成功时,从Bundle获取byte数组方法:
*   getByteArray("bulb_receive")
* @param handler
*/
```

```
5.public abstract void startListen(Handler handler);

/**
 * 关闭监听消息线程,监听消息设置了线程标志位,
 * 当为false时,停止执行循环体代码, 停止监听
 */
6.public abstract void stopListen();

/**
 * 判断是否配置成功,传入的参数是通过上面函数5.所写的Handler里面通过Bundle的函
 * 数:getString("config_receive")来获取,返回为true则配置成功,否则配置失败
 * @return boolean
 */
7.public abstract boolean isConfigSuccess(String message);

/**
* 扫描mDNS服务
* 在调用之前需要打开组播锁
*/
8.public abstract void scanMDNS();

/**
 * 获取扫描到的服务,返回一个List对象,List中每个值存储的是一个Map
 *
 * 每个Map存放扫描到的服务
 * 键：Service_IP 值：扫描到的服务IP
 * 键：Service_Port 值：扫描到的服务端口
 * 键：service_Name 值：扫描到的服务名称
 * @return List
 */
public abstract List<Map<String, Object>> getMdnsList();
```

二.BulbControl 类

```
/**
 * 灯泡控制初始化方法, 设置灯泡IP和端口,这两个参数需要从mDNS服务扫描得到
 * Server_IP是灯泡所在的网络IP,也就是进行mDNS扫描获得的IP,Server_Port
 * 是接收灯泡的控制信息的端口,函数返回的是一个 BulbControl 对象
 * @param  Server_IP
 * @param Server_Port
 * @return BulbControl
 */
1.public static BulbControl init(String Server_IP, int Server_Port);
```

```
/**
 * 打开指定ID的灯泡,输入参数为整型,调用之前请判断参数类型,灯泡的ID不能超过
 * 32767
 * @param bulbID
 */
2.public abstract void openBulb(int bulbID);

/**
 * 关闭指定ID的灯泡,输入参数为整型,调用之前请判断参数类型,灯泡的ID不能超过
 * 32767
 * @param bulbID
 */
3.public abstract void closeBulb(int bulbID);

/**
* 启动监听UDP消息的线程,参数handler需要自己写的
* 当监听的是wifi配置是否成功的消息时,从Bundle中获取字符串方法:
*   getString("config_receive")
* 当监听的是控制开关是否成功时,从Bundle获取byte数组方法:
*   getByteArray("bulb_receive")
* @param handler
*/
4.public abstract void startListen(Handler handler);

/**
 * 关闭监听消息线程,监听消息设置了线程标志位,
 * 当为false时,停止执行循环体代码, 停止监听
 */
5.public abstract void stopListen();

/**
 * 判断灯泡是否控制成功,传入的参数是通过上面函数4.所写的Handler里面通过Bundle的
 * 函数:getByteArray("bulb_receive")来获取,返回为true则控制成功,否则控制失
 * 败
 * @return boolean
 */
public abstract boolean isControlSuccess(byte[] message);
```

三．例子

1.ConfigWifi 类

① 初始化IP和端口,建议使用第二个

```
ConfigWifi configWifi = ConfigWifi.initWifi("192.168.0.100", 9090);
```
或者
```
ConfigWifi configWifi = ConfigWifi.initWifi("192.168.0.100");
```

② 将wifi名和密码发送给设备

```
private String SSID = "test";
private String Password = "123456";
new Thread(){
  @Override
   public void run()
   {
       configWifi.config(ssid, Password);
  }
}.start();
```

③ 监听返回的消息，可选项。如果需要监听是否成功的话可以调用以下函数,message是自己定义的String类型，用于存储监听的消息

```
Handler controlHandler = new Handler(){
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        Bundle bundle = new Bundle();
        bundle = msg.getData();
        message = bundle.getString("config_receive");

    }
   };

/**启动线程监听**/
new Thread(){
  @Override
   public void run()
   {
       configWifi. startListen(controlHandler);
  }
}.start();

/**判断配置是否成功,成功返回true,否则为false, message 是从Handler获取的
**/
isConfigSuccess(message);
```

④ 获取 mDNS 服务。如果配置完成后没有跳转页面,可以直接使用 configWifi 对象,否则使用下面的方法得到服务存放在 mdnsList 中，然后从 mdnsList 获取 Map 的

键”Service_IP”、 ”Service_Port”得到服务的 IP 和端口。连接之后就可以控制灯泡了。注意，扫描 mDNS 服务之前要打开组播服务，Android 默认是关闭的。

```
new Thread(){
    @Override
    public void run()
    {
     //打开组播
     openBroadcast();

     //加载扫描到的服务
     ConfigWifi configWifi = ConfigWifi.init();
     configWifi.scanMDNS();
     mdnsList = configWifi.getMdnsList();
     }
}.start();
```

2.BulbControl 类

① 初始化 IP 和端口，其中 SERVER_IP 和 SERVER_PORT 是通过上面的 mDNS 扫描服务得到的

```
BulbControl bulbControl = BulbControl.init(SERVER_IP,SERVER_PORT);
```

② 打开灯泡，注意灯泡 ID 需要判断

```
new Thread(){
   @Override
   public void run()
   {
      bulbControl.openBulb(bulbID);
    }
}.start();
```

③ 关闭灯泡，注意灯泡 ID 需要判断

```
new Thread(){
   @Override
   public void run()
   {
      bulbControl.closeBulb(bulbID);
}
}.start();
```

④ 监听控制灯泡是否成功，可选项。如果需要监听是否成功的话可以使用以下方

法,message是自己定义的byte[]数组类型，用于存储监听的消息

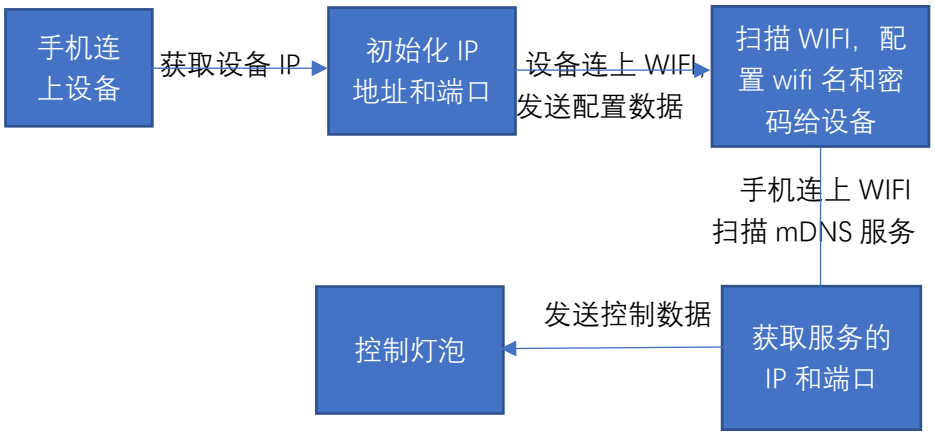
```
Handler controlHandler = new Handler(){
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        Bundle bundle = new Bundle();
        bundle = msg.getData();
        message = bundle.getString("bulb_receive");

    }
    };

/**启动线程监听**/
new Thread(){
  @Override
   public void run()
   {
       configWifi. startListen(controlHandler);
  }
}.start();

/**判断配置是否成功,成功返回true,否则为false, message 是从Handler获取的
**/
isControlSuccess(message);
```

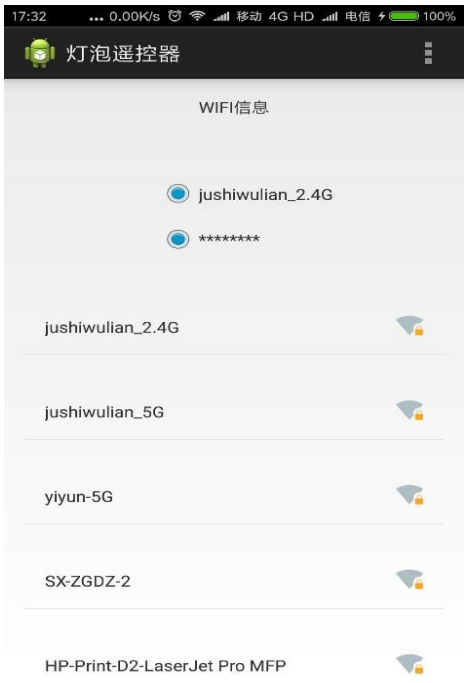
四．流程图



五．DEMO

1.扫描附近的 WIFI

2.获取 WIFI 信息列表



3.配置到站点

4.扫描 mDNS 服务

5.获取 mDNS 服务，得到 IP 和端口


17:27 0.00K/s 移动 4G HD 电信 100%
灯泡遥控器
站点IP
Please input IP
站点端口
Port
扫描
连接
192.168.0.100
点击获取
取消
获取


17:27 1.95K/s 移动 4G HD 电信 100%
灯泡遥控器
站点IP
192.168.0.100
站点端口
9091
扫描
连接
light udpServer

6.连接到设备，控制灯泡


17:30 0.00K/s 移动 4G HD 电信 100%
灯泡遥控器
22
打开
information:
目的IP：192.168.0.100
目的端口：9091
本地端口：13689
接收的信息