未保存,另注意,由于文件传输是长连接里传输文件的 base64 转码后的 String,并非流形式的传输,所以根据 Android 内存管理和实际测试,目前 2M 及以上的文件未作下载处理.

五 详细开发参考代码示例

以最快上手控制一台 Robot 动起来为例:

1. 配置工作.

```
1  第一步:导入 arr 包 ， 设置相关配置文件(如上文介绍)
2  第二步: 生成 eventbus 的 compile 依赖---均可参考 Demo 中示例
```

2.搜索服务器.

2.1 原理

开启 UDP 连接,并接收 UDP 广播,选取需要操控的 server(主键为 serverIP).

2.2 代码

```
1  //开启执行 UDP 的线程
2  public void startUDP() {
3      new Thread(new Runnable() {
4          @Override
5          public void run() {
6              TCP_CONN.isUDP = false;
7              TCP_CONN.getUDPs();
8          }
9      }).start();
10 }
11                  (发送开启 UDP 线程,接收 SDK 封装的 UDP 广播)
```

```
1  @Subscribe(threadMode = ThreadMode.MAIN)
2      public void getUDPInfo(UDPList udpList) {
3          if (udpList != null) {
4              Log.i("eventBus 测试", "SearchServer==>" + udpList.getList().toString());
5              String serverName = udpList.getList().get(0);
6              String serverIP = udpList.getList().get(1);
7              String displayStr = serverName + "" + serverIP;
```

```
8
9                   int a = compare(SearchServerActivity.UDPServerInfoList, displayStr);
10                  if (a == 0) {                                   //UDP 传来的 server 信息与原有
11  的不相同,添加上
12                      SearchServerActivity.UDPServerInfoList.add(udpList.getList());
13                      if (SearchServerActivity.searchServeradapter != null) {
14                          Log.i("searchActivity", "searchServer=" + displayStr);
15                          SearchServerActivity.searchServeradapter.notifyDataSetChanged();
16                      }
17                  }
18                  if ((SearchServerActivity.UDPServerInfoList.size() == 0)) {
19                      SearchServerActivity.UDPServerInfoList.add(udpList.getList());
20                      if (SearchServerActivity.searchServeradapter != null) {
21                          SearchServerActivity.searchServeradapter.notifyDataSetChanged();
22                          Log.i("searchActivity", "searchServer=" + displayStr);
23                      }
24                  }
25
26              }
27          }
28          在 ListView 中选择确定的 serverIP(为后续 TCP 连接获取 IP 参数)
```

3 建立 TCP 连接

```
1   @Override
2   public int onStartCommand(Intent intent, int flags, int startId) {
3
4       new Thread(new KEEPCONN()).start();
5       return super.onStartCommand(intent, flags, startId);
6   }
7
8
9   class KEEPCONN implements Runnable {
10
11      @Override
12      public void run() {
13          TCP_CONN.initFiles(getApplicationContext());
14          TCP_CONN.doTCPLoop(LoginEntity.serverIP,    LoginEntity.user_name,    LoginEntity.password,
15  LoginEntity.salt);
16      }
17  }
18                                  开启服务,执行 TCP 连接线程
```

4 发送命令

```
1  class Move implements Runnable{
2
3          @Override
4          public void run() {
5              TCP_CONN.sendMove(LoginEntity.robotMac,0,0,0.3);
6          }
7      }
8      new Thread(new Move()).start();(开启子线程发送命令)
```

六 备注

1. Android 向 Server 发送命令均为网络操作,请在子线程内调用.
2.长连接 TCP_CONN.doTCPLoop(serverIP,user_name,password,salt)放入 service 里进行执行,保持服务后台持续运行.
3.Navi++ Android SDK 手册基于 Navi++ 开发手册,如想深入了解协议设计及相关内容,请参照该开发手册.