“微控致用”智能垃圾分类机器人

上位机开发报告书

组号：60

负责人：许佳上

一、 项目简介

安卓上位机负责与用户交互以及与MCU传输数据。上位机通过语音和文字提示用户（可切换语音、用户距离、系统状态......）检测用户使用行为（分类开始命令、距离、用户错误报告、分类提示......），同时获取MCU数据（颜色、距离、状态、返回命令......），并且当维护人员维护时除了用户可获得信息也提供更多信息支持（维护命令、返回信息、各传感器状态......），通过这些讲这些数据统合帮助人员更快实现自己的目的、器件更便捷地获取和发送数据。

开发信息：

外接：STM32L476

平台：Smart4418（OS：Android5），CP2102 USB to UART Bridge

软件：Android Studio（Java），Serial Port Utility(串口调试)

管脚：（CON3）+5V、GND、TXD3、RXD3

器件文件：/dev/ddyAMA3

传输协议：UART（8位数据位 1位终止位 无校验位 9600比特率）

1. 功能及原理介绍

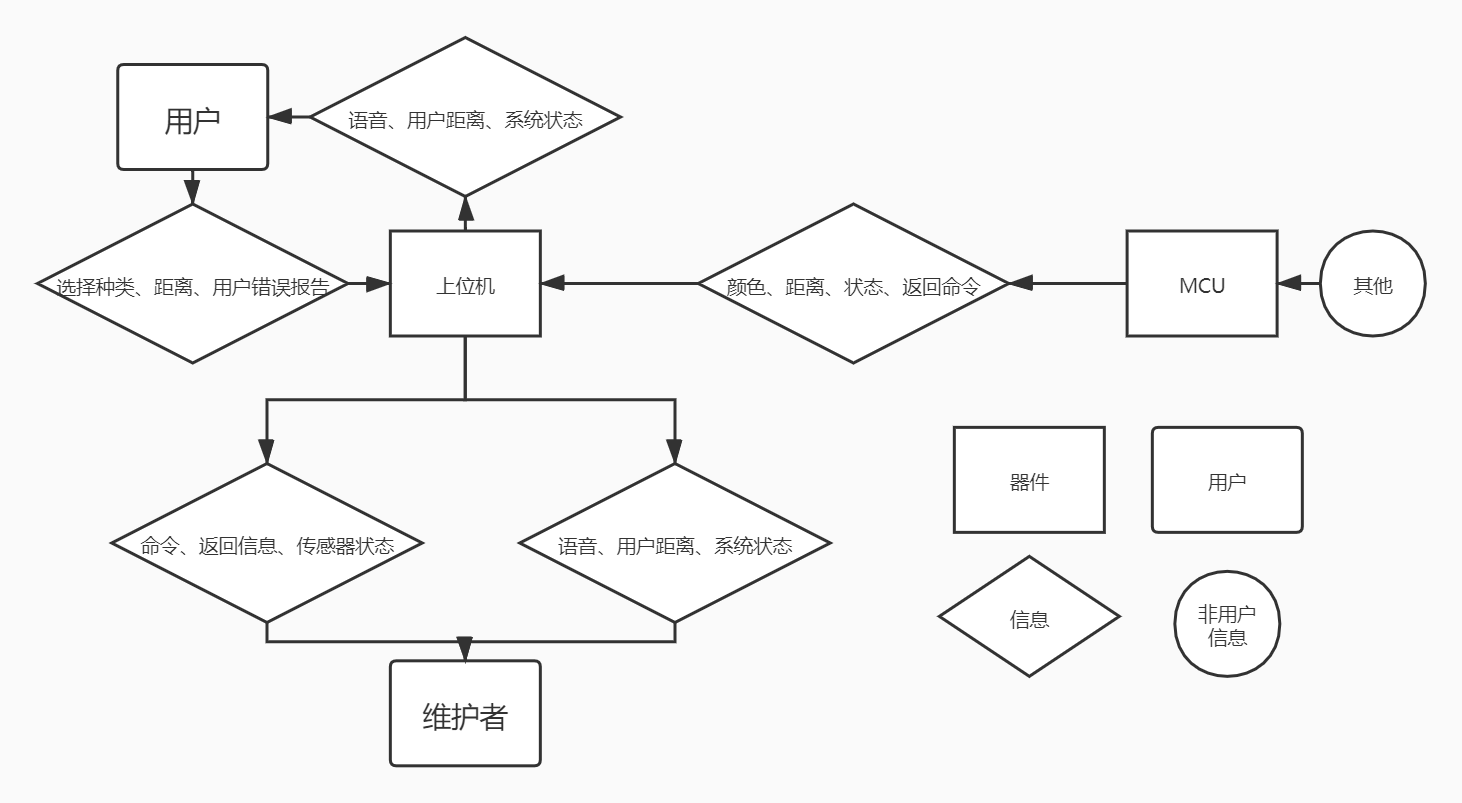


图 1

具体功能如下：

1. MainActivity主页面

* “维护模式”按钮可以切换到维护模式DebugActivity页面，进入密码已设置为“a”，同时由于维护人员在调试维护，右上角错误数会清零；
* “故障反馈”按钮用于记录用户主动报告错误，错误数在右上角在“错误报告：”中显示，当错误数多于20时界面下方“工作状态”将会变为“可能异常”；
* 当上位机与MCU通信过程中出现异常，“工作状态”会变成“异常”并且上方错误报告+1；
* “Go运行”会发送“g”到MCU运行一系列用户距离检测、垃圾识别分类程序；
* “En/中文”用来切换中英文提示语音；
* “语音提示”用来获取语音使用提示；
* 图片点击可以获取对应垃圾种类的例子，帮助用户了解垃圾分类方式；同时图片下文字在垃圾满溢时会接收MCU信息变为红色“……（已满）”
* 实时获取用户距离数据，分析用户需要位于10 - 80 cm范围内才显示“用户距离”为“正常”，负责为“异常”；系统状态根据用户故障反馈和串口回馈数据改变。



图 2

相关代码：

1. **package** trash\_can.jashshor.myapplication;
3. **import** android.annotation.SuppressLint;
4. **import** android.app.AlertDialog;
5. **import** android.content.DialogInterface;
6. **import** android.content.Intent;
7. **import** android.graphics.Color;
8. **import** android.media.MediaPlayer;
9. **import** android.os.Bundle;
10. **import** android.os.Handler;
11. **import** android.os.Message;
12. **import** android.view.View;
13. **import** android.widget.Button;
14. **import** android.widget.EditText;
15. **import** android.widget.ImageButton;
16. **import** android.widget.ImageView;
17. **import** android.widget.TextView;
18. **import** android.widget.Toast;
20. **import** androidx.appcompat.app.AppCompatActivity;
22. **import** com.friendlyarm.FriendlyThings.HardwareControler;
24. **import** java.util.Timer;
25. **import** java.util.TimerTask;

28. **public** **class** MainActivity **extends** AppCompatActivity {
29. **public** **final** **static** **double** max= 100;
30. **public** **int** bugs = 0;
31. **private** TextView bug;
32. **private** TextView julidui;
33. **private** Button button5;
34. **private** Button button4;
35. **private** Button button3;
36. **private** Button button1;
37. **private** ImageButton arc1;
38. **private** ImageButton arc2;
39. **private** ImageButton arc3;
40. **private** ImageButton arc4;
41. **private** String lan;
42. **private** TextView zhuangtai;
43. **private** MediaPlayer mp\_zh;
44. **private** MediaPlayer mp\_en;
46. **private** String devName = "/dev/ttyAMA3";             //用哪个串口，需要改成对应串口文件
47. **private** **int** speed = 9600;       //波特率
48. **private** **int** dataBits = 8;       //数据位
49. **private** **int** stopBits = 1;       //停止位
50. **private** **int** devfd = -1;         //devfd表示串口打开（成功）与否，初始关闭
51. **private** **final** **int** BUFSIZE = 512;
52. **private** **byte**[] buf = **new** **byte**[BUFSIZE];
53. **private** Timer timer = **new** Timer();
55. @Override
56. **public** **void** onDestroy() {       //周期末收尾工作
57. timer.cancel();
58. **if** (devfd != -1) {
59. HardwareControler.close(devfd);
60. devfd = -1;
61. }
62. **super**.onDestroy();      //父类继承
63. }
65. @Override
66. **protected** **void** onCreate(Bundle savedInstanceState) {
67. **super**.onCreate(savedInstanceState);
68. setContentView(R.layout.activity\_main);
69. Button button = (Button) **this**.findViewById(R.id.button2);   //维护模式按钮
70. zhuangtai =(TextView)findViewById(R.id.zhuangtai);
71. button.setOnClickListener(**new** View.OnClickListener() {
72. @Override
73. **public** **void** onClick(View arg0) {            //切换activity
74. AlertDialog.Builder builder = **new** AlertDialog.Builder(MainActivity.**this**);       //弹对话框管理员身份认证
75. builder.setTitle("请输入管理员密码");
76. EditText editText = **new** EditText(MainActivity.**this**);
77. builder.setView(editText);
78. builder.setPositiveButton("确认", **new** DialogInterface.OnClickListener() {
79. @Override
80. **public** **void** onClick(DialogInterface dialog, **int** which) {
81. **if**(which==-1){
82. String key=editText.getText().toString();
83. **if**(key.equals("a"))                                                 //密码匹配后，跳转界面
84. {
85. Intent intent = **new** Intent();
86. intent.setClass(MainActivity.**this**, DebugActivity.**class**);
87. startActivity(intent);
88. bugs = 0;
89. bug.setText("错误报告："+ bugs);//维护后bugs清零，状态变正常
90. zhuangtai.setText("系统状态：正常");}
91. }}
92. });
93. builder.setNegativeButton("取消", **null**);
94. builder.show();
95. }});
96. lan = "zh";                                                         //中英文讲解切换以及语音提示功能
97. mp\_zh = MediaPlayer.create(**this**, R.raw.chinese);
98. mp\_en = MediaPlayer.create(**this**, R.raw.english);
99. button1=(Button)**this**.findViewById(R.id.button1);
100. button1.setOnClickListener(**new** View.OnClickListener(){
101. @Override
102. **public** **void** onClick(View v){
103. **if**(lan.equals("zh"))
104. lan="en";
105. **else** **if**(lan.equals("en"))
106. lan="zh";
107. }
108. });
109. button4=(Button)**this**.findViewById(R.id.button4);
110. button4.setOnClickListener(**new** View.OnClickListener(){
111. @Override
112. **public** **void** onClick(View v){
113. **if**(lan.equals("zh")){
114. mp\_zh.start();}
115. **else** **if**(lan.equals("en")){
116. mp\_en.start();}
117. }
118. });
119. arc1=(ImageButton)**this**.findViewById(R.id.arc1);
120. arc1.setOnClickListener(**new** View.OnClickListener(){
121. @Override
122. **public** **void** onClick(View v){
123. AlertDialog.Builder builder = **new** AlertDialog.Builder(MainActivity.**this**);       //弹可回收垃圾 介绍对话框
124. builder.setTitle("可回收垃圾");
125. ImageView kehui = **new** ImageView(MainActivity.**this**);
126. kehui.setImageResource(R.drawable.kehui);
127. builder.setView(kehui);
128. builder.show();
129. }
130. });
131. arc2=(ImageButton)**this**.findViewById(R.id.arc2);
132. arc2.setOnClickListener(**new** View.OnClickListener(){
133. @Override
134. **public** **void** onClick(View v){
135. AlertDialog.Builder builder = **new** AlertDialog.Builder(MainActivity.**this**);
136. builder.setTitle("其他垃圾");
137. ImageView qi = **new** ImageView(MainActivity.**this**);
138. qi.setImageResource(R.drawable.qi);
139. builder.setView(qi);
140. builder.show();
141. }
142. });
143. arc3=(ImageButton)**this**.findViewById(R.id.arc3);
144. arc3.setOnClickListener(**new** View.OnClickListener(){
145. @Override
146. **public** **void** onClick(View v){
147. AlertDialog.Builder builder = **new** AlertDialog.Builder(MainActivity.**this**);
148. builder.setTitle("厨余垃圾");
149. ImageView chu = **new** ImageView(MainActivity.**this**);
150. chu.setImageResource(R.drawable.chu);
151. builder.setView(chu);
152. builder.show();
153. }
154. });
155. arc4=(ImageButton)**this**.findViewById(R.id.arc4);
156. arc4.setOnClickListener(**new** View.OnClickListener(){
157. @Override
158. **public** **void** onClick(View v){
159. AlertDialog.Builder builder = **new** AlertDialog.Builder(MainActivity.**this**);
160. builder.setTitle("有害垃圾");
161. ImageView hai = **new** ImageView(MainActivity.**this**);
162. hai.setImageResource(R.drawable.hai);
163. builder.setView(hai);
164. builder.show();
165. }
166. });
167. button3 = (Button) **this**.findViewById(R.id.button3);      //故障反馈按钮
168. bug = (TextView)findViewById(R.id.bug);
169. julidui = (TextView)findViewById(R.id.julidui);
170. button3.setOnClickListener(**new** View.OnClickListener() {
171. @Override
172. **public** **void** onClick(View v) {
173. bugs = bugs +1;
174. bug.setText("错误报告："+ bugs);
175. **if**(bugs ==20 ) zhuangtai.setText("系统状态：可能异常");
176. }
177. });
178. devfd = HardwareControler.openSerialPort( devName, speed, dataBits, stopBits );     //重写 打开串口 方法，设定传输参数，获取串口打开状态
179. **if** (devfd >= 0) {
180. zhuangtai.setText("系统状态：正常");
181. timer.schedule(task, 0, 500);  //设置 timer ：0延时，500周期——重复task
182. } **else** {
183. devfd = -1;
184. Toast.makeText(**this**,"出现错误，已记录",Toast.LENGTH\_SHORT).show();
185. bugs = bugs+1;
186. bug.setText("错误报告："+ bugs);
187. zhuangtai.setText("系统状态：异常");           //状态异常为真异常（串口异常），bugs更偏向用户反馈情况
188. }
189. button5 =(Button)findViewById(R.id.button5);
190. button5.setOnClickListener(**new** View.OnClickListener() {
191. @Override
192. **public** **void** onClick(View v) {           //GO命令运转电机
193. **int** ret = HardwareControler.write(devfd, "g".getBytes());
195. **if** (ret > 0) {
196. Toast.makeText(MainActivity.**this**,"Succeed in sending!",Toast.LENGTH\_SHORT).show();
197. } **else** {
198. Toast.makeText(MainActivity.**this**,"Fail to send!",Toast.LENGTH\_SHORT).show();
199. }
200. }
201. });
202. }
204. **private** TimerTask task = **new** TimerTask() {
205. **public** **void** run() {
206. Message message = **new** Message();
207. message.what = 1;               //接收标志位
208. handler.sendMessage(message);           //触发信息接收handler ，转至上一段handler处理
209. }
210. };

213. @SuppressLint("HandlerLeak")
214. **private** Handler handler = **new** Handler() {
215. **public** **void** handleMessage(Message msg) {
216. **switch** (msg.what) {
217. **case** 1:
218. **if** (HardwareControler.select(devfd, 0, 0) == 1) {       //以下：timer 通过调用 select 接口轮询串口设备是否有数据到来
219. **int** retSize = HardwareControler.read(devfd, buf, BUFSIZE);      //读取串口状态，传入数据，数据位数
220. **if** (retSize > 0) {
221. String str = **new** String(buf, 0, retSize); //buf二进制数 offset解码偏移量 retSize位数
222. **double** in;
223. **if**(str.contains("juli")) {                              //距离传入
224. str=strDel(str, "juli");
225. **if**(str.contains("dianjibugongzuo"))
226. str=strDel(str, "dianjibugongzuo");
227. **if**(str.contains("dianjigongzuo"))                           //判断有没有再删除，否则会闪退
228. str=strDel(str, "dianjigongzuo");
229. str= str.replaceAll("\\D+","");
230. in= Integer.parseInt(str);               //String字面转Int
231. **if**(in >= 100 & in< 800) julidui.setText("用户距离：正常");
232. **else**  {
233. julidui.setText("用户距离：异常");
234. Toast.makeText(MainActivity.**this**,"Fail to send!",Toast.LENGTH\_SHORT).show();     //TODO 不表明事件会当成子控件
235. }
236. }
237. **switch**(str)                         //处理传入数据                       //TODO 测试传输数据，约定数据内容
238. {
239. **case** "f":
240. TextView ke=(TextView)findViewById(R.id.kehuishou);
241. ke.setText("可回收垃圾(已满)");
242. ke.setTextColor(MainActivity.**this**.getResources().getColor(R.color.red));
243. **break**;
244. **case** "e":
245. TextView ke2= findViewById(R.id.kehuishou);
246. ke2.setText("可回收垃圾(已满)");                           //TODO  目前垃圾满程度只有可回收垃圾
247. ke2.setTextColor(Color.GRAY);**break**;
248. }
249. }
250. }
251. **break**;          //标准格式 跳出switch case语句
252. }
253. **super**.handleMessage(msg);       //父类继承
254. }
255. };
257. **public** **static** String strDel(String str, String indexStr){
258. **if**(str == **null**){
259. **return** **null**;
260. }
261. StringBuilder newStr = **new** StringBuilder(str);
262. **if**(newStr.indexOf(indexStr) == 0){
263. newStr = **new** StringBuilder(newStr.substring(indexStr.length()));
265. }**else** **if**(newStr.indexOf(indexStr) == newStr.length() - indexStr.length()){
266. newStr = **new** StringBuilder(newStr.substring(0,newStr.lastIndexOf(indexStr)));//在结尾
268. }**else** **if**(newStr.indexOf(indexStr) < (newStr.length() - indexStr.length())){
269. newStr =  **new** StringBuilder(newStr.substring(0,newStr.indexOf(indexStr))+newStr.substring(newStr.indexOf(indexStr)+indexStr.length(),newStr.length()));
271. }
272. **return** newStr.toString();
273. }
274. //    public static String getNumberText(String str){
275. //        if(str.isEmpty()){
276. //            throw new RuntimeException("参数str不能为空");
277. //        }
278. //        StringBuffer number = new StringBuffer("");
279. //
280. //        String[] strArray = str.split("");
281. //        for (String string : strArray) {
282. //            if(!string.isEmpty() && string.contains){
283. //                number.append(string);
284. //            }
285. //        }
286. //        return number.toString();
287. //    }
288. }
289. DebugActivity维护模式页面

* 在黑色指令框输入维护命令，点击左上角“运行命令”即可将命令发送到MCU进行维护操作，所有返回数据将会显示在右侧青色反馈命令输出框，同时输入命令也会在右侧框内，所有命令都可以直接在维护命令输入框输入测试，其中“>”后为调试命令，“<”为返回信息；
* “测试电机”用来发送“g”执行系列命令；
* “距离传感器”显示距离测试数据（mm），“颜色传感器”显示颜色传感；
* 当伺服电机工作时会接收MCU数据，“伺服电机”会显示“工作”，反之收到未工作数据时也会显示“未工作”帮助维护者确定问题位置；
* “获取颜色”、“获取距离”便捷地进入对应调试部分；
* “清除输出”可以清除清空输出框。



图 3

部分相关代码：

1. **package** trash\_can.jashshor.myapplication;
3. **import** android.annotation.SuppressLint;
4. **import** android.app.Activity;
5. **import** android.os.Bundle;
6. **import** android.os.Handler;
7. **import** android.os.Message;
8. **import** android.util.Log;
9. **import** android.view.View;
10. **import** android.widget.Button;
11. **import** android.widget.EditText;
12. **import** android.widget.TextView;
13. **import** android.widget.Toast;
15. **import** com.friendlyarm.FriendlyThings.HardwareControler;
17. **import** java.util.Timer;
18. **import** java.util.TimerTask;
20. **public** **class** DebugActivity **extends** Activity **implements** View.OnClickListener {
22. **private** **static** **final** String TAG = "SerialPort"; //串口通信
24. **private** TextView fromTextView = **null**;
25. **private** EditText toEditor = **null**; //初始化收、发栏
27. **private** TextView dianji = **null**;
28. **private** TextView yanse = **null**;
29. **private** TextView juli = **null**;
31. **private** **final** **int** MAXLINES = 200; //收发极限
32. **private** StringBuilder remoteData = **new** StringBuilder(256 \* MAXLINES); //初始化收发变量，规定字节数
34. **private** String devName = "/dev/ttyAMA3";             //用哪个串口，需要改成对应串口文件
35. **private** **int** speed = 9600;       //波特率
36. **private** **int** dataBits = 8;       //数据位
37. **private** **int** stopBits = 1;       //停止位
38. **private** **int** devfd = -1;         //devfd表示串口打开（成功）与否，初始关闭
40. @Override
41. **public** **void** onDestroy() {       //周期末收尾工作
42. timer.cancel();
43. **if** (devfd != -1) {
44. HardwareControler.close(devfd);
45. devfd = -1;
46. }
47. **super**.onDestroy();      //父类继承
48. }
50. @Override
51. **public** **void** onCreate(Bundle savedInstanceState) {
52. **super**.onCreate(savedInstanceState);
53. setContentView(R.layout.activity\_debug);
54. String winTitle = devName + "," + speed + "," + dataBits + "," + stopBits;
55. //        setTitle(winTitle);                                                           //改变标题 暂定
57. ((Button)findViewById(R.id.sendButton)).setOnClickListener(**this**);   //发送监听
58. ((Button)findViewById(R.id.back)).setOnClickListener(**this**);
59. ((Button)findViewById(R.id.test)).setOnClickListener(**this**);
60. ((Button)findViewById(R.id.testc)).setOnClickListener(**this**);
61. ((Button)findViewById(R.id.testj)).setOnClickListener(**this**);
62. ((Button)findViewById(R.id.clear)).setOnClickListener(**this**);
63. fromTextView = (TextView)findViewById(R.id.fromTextView);       //接收栏
64. toEditor = (EditText)findViewById(R.id.toEditor);       //发送栏
65. dianji = (TextView)findViewById(R.id.dianji);
66. yanse = (TextView)findViewById(R.id.yanse);
67. juli = (TextView)findViewById(R.id.juli);
68. /\* no focus when begin \*/
69. toEditor.clearFocus();
70. toEditor.setFocusable(**false**);
71. toEditor.setFocusableInTouchMode(**true**);
72. devfd = HardwareControler.openSerialPort( devName, speed, dataBits, stopBits );     //重写 打开串口 方法，设定传输参数，获取串口打开状态
73. **if** (devfd >= 0) {
74. timer.schedule(task, 0, 100);  //设置 timer ：0延时，500周期——重复task
75. } **else** {
76. devfd = -1;
77. fromTextView.append("Fail to open " + devName + "!");
78. }
79. }
81. **private** **final** **int** BUFSIZE = 512;
82. **private** **byte**[] buf = **new** **byte**[BUFSIZE];
83. **private** Timer timer = **new** Timer();
84. @SuppressLint("HandlerLeak")
85. **private** Handler handler = **new** Handler() {
86. **public** **void** handleMessage(Message msg) {
87. **switch** (msg.what) {
88. **case** 1:
89. **if** (HardwareControler.select(devfd, 0, 0) == 1) {       //以下：timer 通过调用 select 接口轮询串口设备是否有数据到来
90. **int** retSize = HardwareControler.read(devfd, buf, BUFSIZE);      //读取串口状态，传入数据，数据位数
91. **if** (retSize > 0) {
92. String str = **new** String(buf, 0, retSize); //buf二进制数 offset解码偏移量 retSize位数
93. remoteData.append(str);             //最终传入数据增添，                 //TODO str为获取数据
94. **switch**(str)                         //处理传入数据                       //TODO 测试传输数据，约定数据内容
95. {
96. **case** "red": yanse.setText("颜色传感器：红色");**break**;
97. **case** "blue": yanse.setText("颜色传感器：蓝色");**break**;
98. **case** "green": yanse.setText("颜色传感器：绿色");**break**;
99. **case** "none": yanse.setText("颜色传感器：其他");**break**;
100. }
101. **if**(str.contains("ig")) dianji.setText("伺服电机：工作");
102. **if**(str.contains("ibug")) dianji.setText("伺服电机：未工作");
104. **if**(str.contains("juli")) {                              //距离传入
105. str=strDel(str, "juli");
106. **if**(str.contains("dianjibugongzuo"))
107. str=strDel(str, "dianjibugongzuo");
108. **if**(str.contains("dianjigongzuo"))                           //判断有没有再删除，否则会闪退
109. str=strDel(str, "dianjigongzuo");
110. **if**(str!=""){
111. Log.e("初步净化，接收的",str);
112. str= str.replaceAll("\\D+","");
113. Log.e("完全净化，接收的",str);
114. **double** in = Integer.parseInt(str);                //String字面转Int
115. juli.setText("距离传感器："+String.valueOf(in)+"mm");
116. }**else** Log.e("输入","没有实质性数据");
117. }
118. //Log.d(TAG, "#### LineCount: " + fromTextView.getLineCount() + ", remoteData.length()=" + remoteData.length());
119. **if** (fromTextView.getLineCount() > MAXLINES) {       //输入栏数据行数判断大于最大值
120. **int** nLineCount = fromTextView.getLineCount();
121. **int** i = 0;
122. **for** (i = 0; i < remoteData.length(); i++) {     //i从0位到总数据最后一位，可能是截断溢出值 //TODOing
123. **if** (remoteData.charAt(i) == '\n') {
124. nLineCount--;                       //识别 '\n'
125. **if** (nLineCount <= MAXLINES) {
126. **break**;
127. }
128. }
129. }
130. remoteData.delete(0, i);
131. //Log.d(TAG, "#### remoteData.delete(0, " + i + ")");
132. fromTextView.setText(remoteData.toString());
133. } **else** {
134. fromTextView.append("< "+str);       //行数正常就直接加到输出框就得了 //TODO EditText增加string
135. }
136. **if**(str!="") {
137. remoteData.append('\n');
138. fromTextView.append("\n");
139. }
140. }
141. }
142. **break**;          //标准格式 跳出switch case语句
143. }
144. **super**.handleMessage(msg);       //父类继承
145. }
146. };
147. **private** TimerTask task = **new** TimerTask() {
148. **public** **void** run() {
149. Message message = **new** Message();
150. message.what = 1;               //接收标志位
151. handler.sendMessage(message);           //触发信息接收handler ，转至上一段handler处理
152. }
153. };
155. **public** **void** onClick(View v)         //发送信息监听 线程
156. {
157. **switch** (v.getId()) {                    //识别点击按钮点击按钮ID
158. **case** R.id.sendButton:
159. String str = toEditor.getText().toString();         //获取发送文
160. **if** (str.length() > 0) {
161. **if** (str.charAt(str.length()-1) != '\n') {
162. str = str + "\n";           //增加换行符
163. }
164. **int** ret = HardwareControler.write(devfd, str.getBytes());       //转化二进制调用write接口发送 //TODO 发送数据方式
165. Log.i("write","str");
166. **if** (ret > 0) {              //依据发送成功与否分类
167. toEditor.setText("");           //TODO setText("") 清空EditText
168. str = "> " + str;
169. **if** (remoteData.length() > 0) {
170. **if** (remoteData.charAt(remoteData.length()-1) != '\n') {
171. remoteData.append('\n');
172. fromTextView.append("\n");
173. }
174. }
175. remoteData.append(str);
176. fromTextView.append(str);       //在显示上和数据存储上按'“>>>”+?'格式增加发送值（实际上也没必要区分成两个量）
177. } **else** {
178. Toast.makeText(**this**,"Fail to send!",Toast.LENGTH\_SHORT).show();
179. }
180. }**break**;
181. **case** R.id.back:
182. finish();
183. **break**;
184. **case** R.id.test:
185. String str2 = "g";         //获取发送文
186. **if** (str2.length() > 0) {
187. **if** (str2.charAt(str2.length()-1) != '\n') {
188. str2 = str2 + "\n";           //增加换行符
189. }
190. **int** ret = HardwareControler.write(devfd, str2.getBytes());       //转化二进制调用write接口发送 //TODO 发送数据方式
191. Log.i("write","str");
192. **if** (ret > 0) {              //依据发送成功与否分类
193. toEditor.setText("");           //TODO setText("") 清空EditText
194. str2 = "> " + str2;
195. **if** (remoteData.length() > 0) {
196. **if** (remoteData.charAt(remoteData.length()-1) != '\n') {
197. remoteData.append('\n');
198. fromTextView.append("\n");
199. }
200. }
201. remoteData.append(str2);
202. fromTextView.append(str2);       //在显示上和数据存储上按'“>>>”+?'格式增加发送值（实际上也没必要区分成两个量）
203. } **else** {
204. Toast.makeText(**this**,"Fail to send!",Toast.LENGTH\_SHORT).show();
205. }
206. }**break**;
207. **case** R.id.clear:
209. fromTextView.setText("");
210. **break**;
211. **case** R.id.testc:
212. String str3 = "c";         //获取发送文
213. **if** (str3.length() > 0) {
214. **if** (str3.charAt(str3.length()-1) != '\n') {
215. str3 = str3 + "\n";           //增加换行符
216. }
217. **int** ret = HardwareControler.write(devfd, str3.getBytes());       //转化二进制调用write接口发送 //TODO 发送数据方式
218. Log.i("write","str");
219. **if** (ret > 0) {              //依据发送成功与否分类
220. toEditor.setText("");           //TODO setText("") 清空EditText
221. str3 = "> " + str3;
222. **if** (remoteData.length() > 0) {
223. **if** (remoteData.charAt(remoteData.length()-1) != '\n') {
224. remoteData.append('\n');
225. fromTextView.append("\n");
226. }
227. }
228. remoteData.append(str3);
229. fromTextView.append(str3);       //在显示上和数据存储上按'“>>>”+?'格式增加发送值（实际上也没必要区分成两个量）
230. } **else** {
231. Toast.makeText(**this**,"Fail to send!",Toast.LENGTH\_SHORT).show();
232. }
233. }**break**;
234. **case** R.id.testj:
235. String str4 = "j";         //获取发送文
236. **if** (str4.length() > 0) {
237. **if** (str4.charAt(str4.length()-1) != '\n') {
238. str4 = str4 + "\n";           //增加换行符
239. }
240. **int** ret = HardwareControler.write(devfd, str4.getBytes());       //转化二进制调用write接口发送 //TODO 发送数据方式
241. Log.i("write","str");
242. **if** (ret > 0) {              //依据发送成功与否分类
243. toEditor.setText("");           //TODO setText("") 清空EditText
244. str4 = "> " + str4;
245. **if** (remoteData.length() > 0) {
246. **if** (remoteData.charAt(remoteData.length()-1) != '\n') {
247. remoteData.append('\n');
248. fromTextView.append("\n");
249. }
250. }
251. remoteData.append(str4);
252. fromTextView.append(str4);       //在显示上和数据存储上按'“>>>”+?'格式增加发送值（实际上也没必要区分成两个量）
253. } **else** {
254. Toast.makeText(**this**,"Fail to send!",Toast.LENGTH\_SHORT).show();
255. }
256. }**break**;
257. }
258. }
259. **public** **static** String strDel(String str, String indexStr){
260. **if**(str == **null**){
261. **return** **null**;
262. }
263. StringBuilder newStr = **new** StringBuilder(str);
264. **if**(newStr.indexOf(indexStr) == 0){
265. newStr = **new** StringBuilder(newStr.substring(indexStr.length()));
267. }**else** **if**(newStr.indexOf(indexStr) == newStr.length() - indexStr.length()){
268. newStr = **new** StringBuilder(newStr.substring(0,newStr.lastIndexOf(indexStr)));//在结尾
270. }**else** **if**(newStr.indexOf(indexStr) < (newStr.length() - indexStr.length())){
271. newStr =  **new** StringBuilder(newStr.substring(0,newStr.indexOf(indexStr))+newStr.substring(newStr.indexOf(indexStr)+indexStr.length(),newStr.length()));
273. }
274. **return** newStr.toString();
275. }
276. }
277. 项目结构

app //忽略 /build，/release，/androidTest等次要文件夹，仅主要java,xml文件列出，次要文件忽略，文件前仅顺延上级文件夹竖线但后置

├─libs //so文件所在文件夹

│ ├─arm64-v8a

│ │ libfriendlyarm-things.so

│ │

│ ├─armeabi

│ │ libfriendlyarm-things.so

│ │

│ └─armeabi-v7a

│ libfriendlyarm-things.so

│

└─src

└─main

│ Android.mk

│ AndroidManifest.xml //软件信息、权限配置、activity等

│

├─java

│ ├─com

│ │ └─friendlyarm

│ │ ├─FriendlyThings //硬件功能类

│ │ │ BoardType.java

│ │ │ FileCtlEnum.java

│ │ │ GPIOEnum.java

│ │ │ HardwareControler.java

│ │ │ SPI.java

│ │ │ SPIEnum.java

│ │ │ WatchDogEnum.java

│ │ │

│ │ └─Utils //功能类

│ │ CommonFuncs.java

│ │

│ └─trash\_can

│ └─jashshor

│ └─myapplication

│ DebugActivity.java //维护模式

│ MainActivity.java //主页面

│

└─res

├─drawable

│ chu.jpg

│ chuyu.png

│ hai.jpg

│ ic\_launcher\_background.xml

│ ke.png

│ kehui.jpg

│ qi.jpg

│ qita.png

│ youhai.png

│

├─raw

│ chinese.mp3 //中文语音

│ english.mp3 //英文语音

四、 项目总结

安卓上位机和MCU通信App开发，它不仅仅是一款手机上的App，而更深入的应用到了系统硬件，通过CON3的UART pin脚，需要签名获取权限，第一次体验了apk打包，了解到调用底层硬件权限需要平台的签名包或者用安卓源文件编译包，学习并编写了apk签名、adb覆盖安装apk并自动打开app的脚本，简化了开发流程，实现了安卓上位机和电脑COM口进行UART串口通信（收/发）。

对Java语言有了更深入的了解，面向对象的语言虽然有时令人费解，但是理解了他的内涵，明白了它的用法后又会觉得很生动形象，很贴近生活，甚至于在生活中我们也有实例化，也有各种抽象“器”实现特定功能，在各种研究的同时，自身的工程能力、搜索信息的能力、对AS IDE的了解也进一步加深，对前端也产生了更浓厚的兴趣。

希望下次还能有机会进行安卓开发，进行底层硬件通信。虽然这次走了很多弯路，但是无论是这次积累的经验，还是磨练的探索精神，相信在以后的学习中之后一定能够发挥他们的作用，让下一次挑战更成功、更有收获。