Android 嵌入式纲要

---------柯浩然

一、Android运用

1.button的使用 (按钮)

button文件的布局

在配置文件xml中 使用Android的ui布局控件 构建 button按钮

```
  <Button
        android:id="@+id/btnOne"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:background="@drawable/btn_bg1"
        android:text="按钮"/>


    <Button
        android:id="@+id/btnTwo"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:text="按钮不可用"/>
```

#####

Button的可用属性

Button是TextView的子类，所以TextView上很多属性也可以应用到Button 上

id：为TextView设置一个组件id，根据id，我们可以在Java代码中通过findViewById()的方法获取到该对象，然后进行相关属性的设置，又或者使用RelativeLayout时，参考组件用的也是id！ layout_width：组件的宽度，一般写：**wrap_content**或者**match_parent(fill_parent)**，前者是控件显示的内容多大，控件就多大，而后者会填满该控件所在的父容器；当然也可以设置成特定的大小，比如我这里为了显示效果，设置成了200dp。 layout_height：组件的宽度，内容同上。 gravity：设置控件中内容的对齐方向，TextView中是文字，ImageView中是图片等等。 text：设置显示的文本内容，一般我们是把字符串写到string.xml文件中，然后通过@String/xxx取得对应的字符串内容的，这里为了方便我直接就写到""里，不建议这样写！！！ textColor：设置字体颜色，同上，通过colors.xml资源来引用，别直接这样写！ textStyle：设置字体风格，三个可选值：**normal**(无效果)，**bold**(加粗)，**italic**(斜体) textSize：字体大小，单位一般是用sp！ background：控件的背景颜色，可以理解为填充整个控件的颜色，可以是图片哦！

---

button事件的监听

```
public class MainActivity extends Activity {
```

```
2
3       private Button btnOne,btnTwo;
4
5       @Override
6       protected void onCreate(Bundle savedInstanceState) {
7           super.onCreate(savedInstanceState);
8           setContentView(R.layout.activity_main);
9           btnOne = (Button) findViewById(R.id.btnOne);
10          btnTwo = (Button) findViewById(R.id.btnTwo);
11          btnTwo.setOnClickListener(new OnClickListener() {
12              @Override
13              public void onClick(View v) {
14                  if(btnTwo.getText().toString().equals("按钮不可用")){
15                      btnOne.setEnabled(false);
16                      btnTwo.setText("按钮可用");
17                  }else{
18                      btnOne.setEnabled(true);
19                      btnTwo.setText("按钮不可用");
20                  }
21              }
22          });
23      }
24
25  }
```

在布局中构建好控件后只会在屏幕中出现这个控件但并未有这个控件的相关功能，如要使用控件的相关功能，就会用到java编程了，也就是安卓控制编写。一般每一个页面布局的xml文件会对应一个相关的activity类，这个类生成时会自动继承Activity 而我们就是在这个类中编写安卓按键的功能使用。

```
1       btnOne = (Button) findViewById(R.id.btnOne);
2           btnTwo = (Button) findViewById(R.id.btnTwo);
```

抓取布局中的按钮对象，获取到对象后才能对改按钮进行操作。

```
1   btnTwo.setOnClickListener(new OnClickListener() {
2               @Override
3               public void onClick(View v) {
4                   if(btnTwo.getText().toString().equals("按钮不可用")){
5                       btnOne.setEnabled(false);
6                       btnTwo.setText("按钮可用");
7                   }else{
8                       btnOne.setEnabled(true);
9                       btnTwo.setText("按钮不可用");
10                  }
11              }
12          });
```

给按钮设置点击监听实践 click就是点击后的要进行的方法 在嵌入式中这个按钮就是进行向底层传输数据的操作项。当然设置事件监听还有其他方法，就可以去查查文档，但大致都是一样都需要写一个事件方法作为点击时要触发的事件。

2.EditText的使用 (文本框)

EditText的文件布局

在配置文件xml中 使用Android的ui布局控件 构建 EditText文本框

```
<EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="phone" />
```

EditText的可用属性

EditText因为也继承了TextView的原因 除了拥有TextView的属性之外 都还拥有这其他的属性

```
android:hint="默认提示文本"              //设置提示话语
android:textColorHint="#95A1AA"          //设置提示话语字体颜色
android:selectAllOnFocus="true"          //设置文本框聚焦后自动选择全部文字
android:singleLine="true"              //设置文本框是否允许跳行
android:textScaleX="1.5"                 //设置字与字的水平间隔
android:textScaleY="1.5"                 //设置字与字的垂直间隔
....
```

EditText的可用方法

`void`	`extendSelection(int index)` **方便** `extendSelection(Spannable, int)`。
`CharSequence`	`getAccessibilityClassName()` 返回此对象的类名以用于辅助功能。
`boolean`	`getFreezesText()` 返回此文本视图是否将其整个文本内容包含在冻结的冰柱中。
`Editable`	`getText()` 返回TextView正在显示的文本。
`void`	`selectAll()` 方便 `selectAll(Spannable)`。
`void`	`setEllipsize(TextUtils.TruncateAt ellipsis)` 导致文本中长于视图宽度的单词被椭圆化而不是在中间断开。
`void`	`setSelection(int start, int stop)` 方便 `setSelection(Spannable, int, int)`。
`void`	`setSelection(int index)` 方便 `setSelection(Spannable, int)`。
`void`	`setText(CharSequence text, TextView.BufferType type)` 设置此TextView要显示的文本（请参阅 `setText(CharSequence)` 参考资料），并设置它是否存储在可设置样式/ spannable的缓冲区中以及是否可编辑。

3.Spinner的使用 (下拉框)

Adapter的使用

在使用Spinner时我们必须要构建一个adapter来进行对Spinner的数据渲染 给一个基本的自定义的adapter 只能参考 不一定能适用。

```
public class MyAdapter extends BaseAdapter {

    private Context mContext;
    private LinkedList<Data> mData;

    public MyAdapter() {
    }

    public MyAdapter(LinkedList<Data> mData, Context mContext) {
        this.mData = mData;
        this.mContext = mContext;
    }

    @Override
    public int getCount() {
        return mData.size();
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder = null;
        if (convertView == null) {
            convertView = LayoutInflater.from(mContext).inflate(R.layout.item_list,
parent, false);
            holder = new ViewHolder();
            holder.img_icon = (ImageView) convertView.findViewById(R.id.img_icon);
            holder.txt_content = (TextView)
convertView.findViewById(R.id.txt_content);
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }
        holder.img_icon.setImageResource(mData.get(position).getImgId());
        holder.txt_content.setText(mData.get(position).getContent());
        return convertView;
    }

    //添加一个元素
```

```
    public void add(Data data) {
        if (mData == null) {
            mData = new LinkedList<>();
        }
        mData.add(data);
        notifyDataSetChanged();
    }

    //往特定位置，添加一个元素
    public void add(int position,Data data){
        if (mData == null) {
            mData = new LinkedList<>();
        }
        mData.add(position, data);
        notifyDataSetChanged();
    }

    public void remove(Data data) {
        if(mData != null) {
            mData.remove(data);
        }
        notifyDataSetChanged();
    }

    public void remove(int position) {
        if(mData != null) {
            mData.remove(position);
        }
        notifyDataSetChanged();
    }

    public void clear() {
        if(mData != null) {
            mData.clear();
        }
        notifyDataSetChanged();
    }

    private class ViewHolder {
        ImageView img_icon;
        TextView txt_content;
    }

}
```

这个adapter是继承了baseAdapter 进行的自定义的adapter

如果看不懂可以去先看看Android对adapter的概念和理解

http://www.runoob.com/w3cnote/android-tutorial-adapter.html 这里面就有具体的adapter 和后续baseAdapter的教程和使用问题。

效果展示

如图这就是要做出来的样子大概的，或许数据还不用这么多。

Spinner的布局

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="选择你的拿手英雄~"
        android:textColor="#F5684A"
        android:textSize="18sp" />

    <Spinner
        android:id="@+id/spin_two"
        android:layout_width="wrap_content"
        android:layout_height="64dp" />
```

在xml文件布置方面很简单只需要一个Spinner标签就可以了难点就是讲数据加载进去，当然如果会adapter之后 其实发现也没什么难点 只是重复的输入数据，当然我们还有最重要的配置xml的一步。因为是要将数据一条一条的展示所以我们还有配置一个行间的布局xml文件，如果没有这个文件就整个就算有数据也不行，so：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <ImageView
        android:id="@+id/img_icon"
        android:layout_width="48dp"
        android:layout_height="48dp"
        android:src="@drawable/aaa" />
    <TextView
        android:id="@+id/txt_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="15dp"
        android:text="德玛西亚"
        android:textSize="16sp" />
</LinearLayout>
```

这就是我们新创建的行间样式的布局了，其实也是很简单的，只是在创建了一个页面布局而已。所以总的来说Spinner的布局相当于就是一个容器页面，将无数个自动生成的子页面放在一起然后展示出来(个人理解),这些子页面就是公用的一个布局文件，就像是javaEE前端的js渲染 。

Activity的编写

说完了布局，那么就是相对来说主类Activity的编写，用来实现选择效果，不然光有布局是没有任何反应。

```
public class SevenActivity extends Activity implements OnItemSelectedListener{
  private Spinner spin_one;
    private Spinner spin_two;
    private Context mContext;
    //判断是否为刚进去时触发onItemSelected的标志
```

```
    private boolean one_selected = false;
    private boolean two_selected = false;
    private LinkedList<Hero> mData = null;
    private MyAdapter myAdadpter = null;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_seven);
    mContext = SevenActivity.this;
        mData = new LinkedList<Hero>();
  spin_one = (Spinner) findViewById(R.id.spin_one);
    spin_two = (Spinner) findViewById(R.id.spin_two);

    mData.add(new Hero(R.drawable.aaa,"迅捷斥候：提莫（Teemo）"));
    mData.add(new Hero(R.drawable.aaa,"诺克萨斯之手：德莱厄斯（Darius）"));
    mData.add(new Hero(R.drawable.aaa,"无极剑圣：易（Yi）"));
    mData.add(new Hero(R.drawable.aaa,"德莱厄斯：德莱文（Draven）"));
    mData.add(new Hero(R.drawable.aaa,"德邦总管：赵信（XinZhao）"));
    mData.add(new Hero(R.drawable.aaa,"狂战士：奥拉夫（Olaf）"));

    myAdadpter = new MyAdapter(mData,mContext);
    spin_two.setAdapter(myAdadpter);
    spin_one.setOnItemSelectedListener(this);
    spin_two.setOnItemSelectedListener(this);
}

@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
{
  switch (parent.getId()){
        case R.id.spin_one:
            if(one_selected){
                Toast.makeText(mContext,"您的分段是~：" +
parent.getItemAtPosition(position).toString(),
                        Toast.LENGTH_SHORT).show();
            }else one_selected = true;
            break;
        case R.id.spin_two:
            if(two_selected){
                TextView txt_name = (TextView) view.findViewById(R.id.txt_name);
                Toast.makeText(mContext,"您选择的英雄是~：" +
txt_name.getText().toString(),
                        Toast.LENGTH_SHORT).show();
            }else two_selected = true;
        break;
  }
}

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        // TODO Auto-generated method stub

    }
```

```
56 }
```

这就是相对于来说整个主Activity的内容了我们再来剖析一下

```
1  private Spinner spin_one;
2  private Spinner spin_two;
3  private Context mContext;
4  private LinkedList<Hero> mData = null;
5  private MyAdapter myAdadpter = null;
6
7  mContext = SevenActivity.this;
8  mData = new LinkedList<Hero>();
9  spin_one = (Spinner) findViewById(R.id.spin_one);
10 spin_two = (Spinner) findViewById(R.id.spin_two);
```

这就不用说了 这是在实例化对象 将所需要的对象获取出来 其中要注意 我在实例化 LinkedList中 Hero是我自己写的实体类 因为我在下拉框中运用到了图片 正常文字 所以我用了一个Hero类来装

```
1  mData.add(new Hero(R.drawable.aaa,"迅捷斥候：提莫 (Teemo) "));
2  mData.add(new Hero(R.drawable.aaa,"诺克萨斯之手：德莱厄斯 (Darius) "));
3  mData.add(new Hero(R.drawable.aaa,"无极剑圣：易 (Yi) "));
4  mData.add(new Hero(R.drawable.aaa,"德莱厄斯：德莱文 (Draven) "));
5  mData.add(new Hero(R.drawable.aaa,"德邦总管：赵信 (XinZhao) "));
6  mData.add(new Hero(R.drawable.aaa,"狂战士：奥拉夫 (Olaf) "));
```

这就是我们将所想要添加的数据添加进我们写好的集合中

```
1
2  spin_two.setAdapter(myAdadpter);
3
```

这是最简单的一步也是最重要的一步，给我们的spinner对象设置一个Adapter设置完成就完成了整个数据展示，到了这部其实我们的下拉框就可以用了，但是因为可能我们要做一些选择了下拉框内容时做点其他的想法，就有了新的监听方法

```
1   spin_two.setOnItemSelectedListener(this);
2
3  public class SevenActivity extends Activity implements OnItemSelectedListener{
```

这就是我们给Spinner对象设置选项的点击监听事件 但是设置这个的前提是我们要给我们的Activity继承一个接口**OnItemSelectedListener**不然这样是会报错的。继承了接口后我们会继承一些方法

```
1  @Override
2  public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
   {
3    switch (parent.getId()){
4          case R.id.spin_one:
5              if(one_selected){
6                  Toast.makeText(mContext,"您的分段是~: " +
   parent.getItemAtPosition(position).toString(),
```

```
                        Toast.LENGTH_SHORT).show();
            }else one_selected = true;
            break;
        case R.id.spin_two:
            if(two_selected){
                TextView txt_name = (TextView) view.findViewById(R.id.txt_name);
                Toast.makeText(mContext,"您选择的英雄是~: " +
txt_name.getText().toString(),
                        Toast.LENGTH_SHORT).show();
            }else two_selected = true;
        break;
  }
}
@Override
public void onNothingSelected(AdapterView<?> parent) {
    // TODO Auto-generated method stub

}
```

这个两个方法就是继承接口后出现的，第二个方法我们可以不用管，第一个方法就是我们主要的在点击选项中是会触发的点击事件。在这个事件中，有两个点击 因为我写了两个下拉框 所以点击有两个。但是分开来看

`parent`	`AdapterView`：**发生单击的AdapterView。**
`view`	`View`：AdapterView中被单击的视图（这将是适配器提供的视图）
`position`	`int`：适配器中视图的位置。
`id`	`long`：已单击的项的行ID。

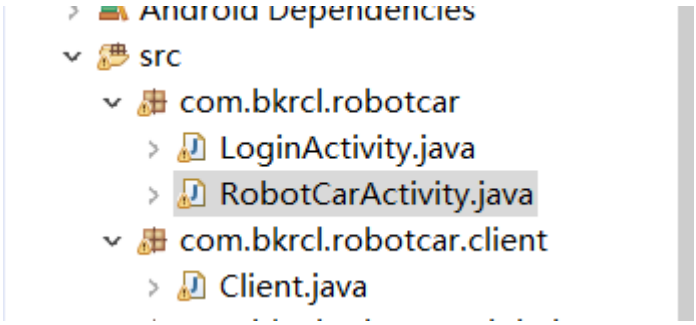
这个方法中参数的意义。这样spinner就很明显了。

二、嵌入式项目理解

1.地位认知

在整个嵌入式比赛中，因为我们嵌入式比赛不止我们Android的这一方面，还涉及底层的C语言方面，还有焊板子物理层方面，当然焊板子的物理层方面和我Android的关系几乎不大，主要的还是底层的C语言方面的。所以整个嵌入式比赛我们不是唯一的，我们要先认清自己在比赛中的位置，该做什么，该怎么做好。说到位置，其实也就是地位，从一开始我们Android的地位是首要的，最关键的就是我们Android这一块，不论是图形图像的处理还是小车移动的处理。或者说是对小车的操控。如果Android崩了，那整个项目也就GG，整个比赛也就结束可。但是在去年，我们整个由于技术更新和进行了明确分工，已经将Android操控小车的一切，以Android为主体的情况转变为了现在以底层C语言为主，有底层来调用上层的Android的逻辑结构。现在控制小车的一切操作都交给了底层来操作，而现在Android负责的主要就是摄像头的控制，对图像进行处理，和数据传递的功能。所以在现在我们应该要明确的认识自己在比赛里的任务，更好的完成它！

2.代码解析



我们在比赛中操作的类主要就是三个类，这三个类主要的就是对其他包内的方法的调用。

LoginActivity解析

```
public class LoginActivity extends Activity implements OnClickListener {
  private Button bt_login;
  private RadioButton rb_wifi, rb_serialPort;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    init();

  }

  private void init() {
    bt_login = (Button) findViewById(R.id.bt_login);
    rb_wifi = (RadioButton) findViewById(R.id.rb_wifi);
    rb_serialPort = (RadioButton) findViewById(R.id.rb_serialPort);

    bt_login.setOnClickListener(this);
  }

  @Override
  public void onClick(View v) {
    switch (v.getId()) {
    case R.id.bt_login:
      Intent intent = new Intent();
      if (rb_wifi.isChecked()) {
        intent.putExtra("communicationMode", "wifi");
      } else if (rb_serialPort.isChecked()) {
        intent.putExtra("communicationMode", "serialPort");
      }

      intent.setClass(this, RobotCarActivity.class);
      startActivity(intent);
      break;

    default:
      break;
    }
  }
```

```
}
```

这是LoginActivity类，也就是进入程序时的第一份activity，这个activity就是用来选择是否是有线连接或者是无线连接的。主要是应用intent这个对象进行activity之间的跳转。如果要涉及跳转的问题。

http://www.runoob.com/w3cnote/android-tutorial-intent-base.html

这个网站上对intent有了个清楚的描述。

RobotCarActivity解析

然后就是RobotCarActivity类，这个类由于内容太多就不全部展示出来了，但是还是讲一些主要：

```
// 创建时
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_robot_car);
    communicationMode = getIntent().getStringExtra("communicationMode");
    if (communicationMode.equals("wifi")) {
      isSerialPort = false;
    } else if (communicationMode.equals("serialPort")) {
      isSerialPort = true;
    }
    // 控件初始化
    init();
    // 实例化client类
    client = new Client();
    client.TYPE = 0xAA;
    // 搜索摄像头图片工具
    cameraCommandUtil = new CameraCommandUtil();
    // 摄像头初始化
    Camer_Init();
    // // 注册广播接收器
    // IntentFilter intentFilter = new IntentFilter();
    // intentFilter.addAction(A_S);
    // registerReceiver(myBroadcastReceiver, intentFilter);
    if (isSerialPort) {// 串口
      mHandler.sendEmptyMessageDelayed(MESSAGE_REFRESH,
          REFRESH_TIMEOUT_MILLIS); // 启动usb的识别和获取
    } else {// WiFi
      // 开启后台服务搜索摄像头
      // search();
      // 得到服务器的IP地址
      wifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
      dhcpInfo = wifiManager.getDhcpInfo();
      IPCar = Formatter.formatIpAddress(dhcpInfo.gateway);
      Log.e("小车IP值: ", IPCar);
    }
    // 搜索摄像头
    search();
    // 导入OpenCv库
    OpenCvLoadUtil.loadOpenCV(RobotCarActivity.this);
    // 同wifi服务器连接
    socketThread.start();
```

```
    // 自动运行线程
    client.autoThread.start();

  };
```

这是这个类的调动整个安卓的核心，初始化控件，初始化摄像头，连接opencv的分析库，并开始用sockt对象进行数据的传调。

```
communicationMode = getIntent().getStringExtra("communicationMode");
    if (communicationMode.equals("wifi")) {
      isSerialPort = false;
    } else if (communicationMode.equals("serialPort")) {
      isSerialPort = true;
    }
```

这个判断就是获取从第一个activity里传过来的值进行对比，来决定与小车的联系到底是有线连接还是无线连接。

```
if (isSerialPort) {// 串口
      mHandler.sendEmptyMessageDelayed(MESSAGE_REFRESH,
          REFRESH_TIMEOUT_MILLIS); // 启动usb的识别和获取
    } else {// WiFi
      // 开启后台服务搜索摄像头
      // search();
      // 得到服务器的IP地址
      wifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
      dhcpInfo = wifiManager.getDhcpInfo();
      IPCar = Formatter.formatIpAddress(dhcpInfo.gateway);
      Log.e("小车IP值: ", IPCar);
    }
```

这个判定就是根据前面判断有线无线连接进行分别连接小车，如果是有线连接，就调用串口。

```
    // 搜索摄像头
    search();
    // 导入OpenCv库
    OpenCvLoadUtil.loadOpenCV(RobotCarActivity.this);
    // 同wifi服务器连接
    socketThread.start();
    // 自动运行线程
    client.autoThread.start();
```

最后的这一部分，看着简单，但是算是整个程序最重要的起点，如果没有他们，我们Android方面就GG，这上面的功能其实注释已经描述的很仔细了。其中我们最关心得就是client.autoThread.start();这一段代码，这个就是让我们写好的程序可以自动的进行运行。

再解释完了RobotCarActivity主要的oncreate方法后，其余的也就不再细说，但是总的来说在这个类里面剩下的方法基本上就是对上面控件所会触发的事件进行监听，还有就是对事件的初始化。其他的就是剩下的各种识别方法的编写就在这里面，然而调用我们则是运用线程，在另一个Client类里的设置调用 ，为的就应该是防止线程出现问题和主类的拥挤还有就是代码的可看性吧。这些方法我们在讲到调用时再回来细说。

Client解析

Clientl类中，我们讲究的就是只负责调用，数据处理神什么，方法内该怎么写我们都不负责，Client类，换个思维来说这是个中转的调度站，算是大脑，但我们只负责发送执行方法的指令和将要执行方法需要的变量发送给该方法就ok，其他的我们一概不。大体就是这样。

```
protected void auto() {
    // 接受RFID上传的32位10进制数据或16位16进制数据124为包头，125为包尾
    if (isEffective_RFID_DATA == false) {// 如果没收到RFID数据
      getRFID_Data();
    }

    // 判断状态值是否为底层发送的命令
    ReceiveCommands();

    switch (mark) {
    // 80识别颜色 81数码管显示 82识别交通标志 83识别车牌 84立体显示 85打开报警器 86识别二维码 87TFT显示
    // 摄像头1正前，2左转，3右转
    case 1:
      // *****************2018年5月27日国赛任务
      action();
      state_camera(2);
      setMark(-7);

      break;// ===================================

    case 2:
      setMark(-7);
      break;// ===================================

    case 3:
      setMark(-7);
      break;
    //
**********************************************************************************************************************
    // **********************以下为底层可调用任务，ReceiveCommands()控制mrak值
**************************************************************************
    //
**********************************************************************************************************************
    case 80:// 识别颜色

      // identifyPlateNumber();
      // state_camera(1);
      // TFTDisplayImageUpDownRandom(2);
      // yanchi(2000);// 避免拍照不准确，等待两秒，确保摄像头图像是目标图像
      // state_camera(2);
      identifyColorShape();
      // state_camera(1);
      // sendQRMsg(MyGlobal.TK_K07 + "");
      // LEDtwoShow_6chars(MyGlobal.TK_K06);
      TFTDisplayPlateNumber(MyGlobal.CarID);
      // sendCarID(MyGlobal.Shape.subSequence(0, 6) + ""
```

```
      // + MyGlobal.Shape.subSequence(8, 10));
      action();
      // state_camera(3);
      // setMark(-7);
      break;// ====================================
    case 81:// LED数码管第二排显示6位字符
      // MyGlobal.TK_K06图形个数
      LEDtwoShow_6chars(MyGlobal.TK_K06);
      action();
      break;// ====================================
    case 82:// 识别交通标志
      state_camera(1);
      identifyTraffic();
      // state_camera(1);
      // RobotCarActivity.voiceController(MyGlobal.M06);
      action();
      // state_camera(1);
      break;// ====================================
    case 83:// 识别车牌
      // state_camera(2);
      identifyPlateNumber();
      // state_camera(1);
      action();
      // state_camera(3);
      break;// ====================================
    case 84:// 立体显示
      // for (int i = 0; i < 3; i++) {
      // // A矩形B圆形C三角形E五角星(使用车牌通信协议显示)AaBbCcDdEe
      // // getRbyteM07(10);
      // if (!MyGlobal.Shape.equals("")) {
      // threeDisplayShowPlateNumber(MyGlobal.Shape);
      // } else {
      // threeDisplayShowPlateNumber(MyGlobal.TK_M06_Shape);
      // }
      // delay(500);
      // }
      action();
      break;// ====================================
    case 85:// 打开报警器
      policeController();
      action();
      break;// ====================================
    case 86:// 识别二维码
      // state_camera(3);
      sendQrCodeMsg(0x20, 1);
      if (MyGlobal.TK_M01.equals("")) {
        RobotCarActivity.cameraCommandUtil.postHttp(
            RobotCarActivity.IPCamera, 0, 1);// 抬头
      }
      if (MyGlobal.TK_M01.equals("")) {
        RobotCarActivity.cameraCommandUtil.postHttp(
            RobotCarActivity.IPCamera, 2, 1);// 低头
      }
```

```
      // state_camera(1);
      // 换位算法
      decode(MyGlobal.TK_M01);
      action();
      state_camera(1);
      break;// ====================================
    case 87:// TFT显示
      TFTDisplayPlateNumber(MyGlobal.TK_M06_Shape);
      delay(5000);
      // RFID卡坐标显示
      getRbyteM07(10);
      TFTDisplayPlateNumber("RFID" + MyGlobal.RFID_POSITION);
      break;
    default:
      break;// ====================================
    }

  }
```

这一段代码就是我们在比赛中我们和底层的交互的核心。因为在整个代码中我们有一个死循环的方法，一直在运作着，一直接收者底层所返回的一个数组集合。而我们就是用这个数组集合中的一个make变量的值来对其判断底层来要求我们来进行什么操作。才有了后续的操作。

80识别颜色 81数码管显示 82识别交通标志 83识别车牌 84立体显示 85打开报警器 86识别二维码 87TFT显示 这些数字就是我们和底层的约定的协议，当我们获取到的mrak值是符合这几个数字时我们会自动跳入改判定中来执行我们的方法。

80 识别颜色

```
case 80:// 识别颜色

    identifyColorShape();//调用识别颜色

    TFTDisplayPlateNumber(MyGlobal.CarID);//发送给底层数据

    action();

    break
```

我们来解析下去年国赛的代码，这个是在国赛中所约定的调用识别颜色图形图像的方法。这里面有一共有三行我们一行一行解析，第一行：

```
  identifyColorShape(); //调用识别颜色
```

这个就直接是调用方法不用解释了吧调用这个方法和我们看看

```
/**
 * 识别图形图像形状颜色
 */
private void identifyColorShape() {
  msgShow("识别颜色");
  // 状态80
  carControlHandler.sendEmptyMessage(0x30);//发送handler消息
  MarkResetWait();
}
```

在这个方法中其实会发送一个Handler的消息来进行传递，而传递的标志就是0x30 这个是自己定义的但是要和整个项目相匹配嘛，其实可以是字母数字都行，这样写就比较规范。然后在发送消息发送后会看到有执行了一行代码

```
/**
   * 让client线程任务暂停，等待RobotCarActivity中的线程任务完成。
   * */
  private void MarkResetWait() {
    markTemp = mark;
    mark = -8;
    MAJOR = 0;
    msgShow("-----开始等待-----");
    while (mark != markTemp) {
      // 线程A等待线程B完成动作
      if (MAJOR != 0) {
        send();
        MAJOR = 0;
      }
    }
    // msgShow("-----等待结束-----");
  }
```

这个方法注释很清楚，就是用来让线程停住，放置你调用的任务线程还没有做完，就继续执行其他的后续方法。因为在Android中，线程的先后执行顺序是不可控的，你无法掌控线程的完成否。所以就需要一个这样的方法来停住我们的client线程任务，等待着前面调用的任务线程来使用。

在回来看看前面handler发送的数据参数，这个参数发送后我们会在RonotCarActivity里接收到这个消息：

```
case 0x30:// 形状颜色识别，并发送解析
    MyGlobal.TK_M05_CarID = "";
    // 第一次车牌
    client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
    // IdentifyPlateNumber();
    // if (!MyGlobal.TK_M05_CarID.equals("")) {
    // client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
    // // client.voiceController("车牌识别成功");
    // }
    // 图像识别
    IdentifyColorShape();
    if (!MyGlobal.Shape.equals("")) {
        MyGlobal.TK_M06_Shape = MyGlobal.Shape;
      }
    // 第二次车牌
```

```
    // if (!MyGlobal.TK_M05_CarID.equals("")) {
    // client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
    // IdentifyPlateNumber();
    // // client.voiceController("车牌识别成功");
    // }
    // if (!MyGlobal.CarID.equals("")) {
    // client.sendCarID(MyGlobal.CarID);
    // } else {
    // client.sendCarID(MyGlobal.TK_M05_CarID);
    // }
    client.sendRFIDcode();
    client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
    client.MarkGo();
    break;// ===================================
```

这里面主要也是用了三行代码第一个：

```
MyGlobal.TK_M05_CarID = "";
```

这是变量置空不用说了 因为这个变量是我们定义的变量类MyGlobal类里面的静态全局的变量，在程序的其他地方可能给他赋过值，但是为了代码的稳定性，就在用这个方法前先置空一次。

```
client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
```

这行代码就是向小车底层发送个消息让它去切换TFT显示器的页面，先说说TFT为什么要翻页吧，因为在TFT显示器中，他显示的不只是图形图像还有着车牌，而且车牌和图像图像一般就是一张车牌一张图形图像。所以就需要进行翻页判断。

```
public void TFTDisplayImageUpDownRandom(int index) {
    msgShow("TFT翻页");
    if (index == 1) {// 向上
      // MAJOR = (short) 0x52;
      // FIRST = (byte) 1;
      // SECOND = (byte) 0;
      // THRID = (byte) 0;
      // send();
      TFT_LCD(0x10, 0x01, 0x00, 0x00);
    } else if (index == 2) {// 向下

      // MAJOR = (short) 0x52;
      // FIRST = (byte) 2;
      // SECOND = (byte) 0;
      // THRID = (byte) 0;
      // send();
      TFT_LCD(0x10, 0x02, 0x00, 0x00);

    } else if (index == 3) {// 自动

      TFT_LCD(0x10, 0x03, 0x00, 0x00);

    }
```

```
    delay(4000);
  }
```

这里就是对TFT显示器通过小车发送数据，其中发送数据的参数，这个可能就和底层或者当场考试的信息协议有关系，所以在比赛时这个就可能会有所变化并不是唯一。

```
public void TFT_LCD(int MAIN, int KIND, int COMMAD, int DEPUTY) {
    byte type = (byte) TYPE;
    TYPE = (short) 0x0B;
    MAJOR = (short) MAIN;
    FIRST = (byte) KIND;
    SECOND = (byte) COMMAD;
    THRID = (byte) DEPUTY;
    System.out.println(1);
    send();
    System.out.println(2);
    TYPE = type;
  }
```

这个方法就是将前面要发送的数据传递过来进行字节转换，转换成小车底层能够识别的16进制代码，然后通过send()方法发送给车。

```
public void send() {
    // 避免连续发送send（500毫秒内连续发送会导致接收不到正确的指令）
    if (sendTime != 0)
      while (System.currentTimeMillis() - sendTime < 500) {
        // delay(50);
        // System.out.println(System.currentTimeMillis() - sendTime);
      }
    sendTime = System.currentTimeMillis();
    try {
      CHECKSUM = (short) ((MAJOR + FIRST + SECOND + THRID) % 256);
      // 发送数据字节数组
      byte[] sbyte = { 0x55, (byte) TYPE, (byte) MAJOR, (byte) FIRST,
          (byte) SECOND, (byte) THRID, (byte) CHECKSUM, (byte) 0xBB };
      System.out.println(Arrays.toString(sbyte));
      if (RobotCarActivity.isSerialPort) {
                //// 是否串口通信，串口通信为true,wifi通信为false
        //public static boolean isSerialPort = false;

        if (RobotCarActivity.sPort != null) {//usb的链接处理
          RobotCarActivity.sPort.write(sbyte, 5000);
        }
      } else if (socket != null && !socket.isClosed()) {
        bOutputStream.write(sbyte, 0, sbyte.length);
        bOutputStream.flush();
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
```

这个就是真正的发送方法，在这方法中我们先做了一个避免重发的可能性处理，因为这是在Android的线程中，重发对的可能性会有，但是接收到重发话我们的代码逻辑可能会炸，所以做了一个这样的处理，保证代码能够正常运行。然后就是主要功能数据的发送，这了可能代码能看懂但不知道为什么这么写，只能说这应该 就是小车的协议了，要进行一层封装，在对连接模式是有线还是无线连接进行处理。具体的变量意义我也不能全部粘贴完全，如果要看可以在eclipse中按住Ctrl 在点击那个变量会跳转到他的定义里，哪里应该 有注释，就能理解下。

```
client.sendRFIDcode();
client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
client.MarkGo();
```

这里又会触发一个方法，因该就是根据题目所获取的变量来发送小车入库时的车库

```
/**
    * 发送主车所入的车库
    */
   public void sendRFIDcode() {
     int s = MyGlobal.TK_K01, t = Integer.valueOf(MyGlobal.TK_M06_Shape
         .substring(7, 8));
     if (s > 15) {
       s = s % 15;
     }
     if (t > 3) {
       t = t % 3;
     }
     // A矩形B圆形C三角形D菱形E五角星
     MAJOR = 0x97;
     FIRST = 0;
     SECOND = (short) s;// 红色图形的数量，超过15对15取余
     THRID = (short) t;// 菱形的数量，超过3对3取余
     send();
     System.out.println("s:" + s + "t:" + t);
   }
```

从变量中取出所需要的值然后设定，这就是一个简单逻辑处理，然后将值发给小车让小车进行处理或者储存数据。之后数据发送后我们要有选择性的继续翻页，要问为什么，在比赛中可能图像图形识别实在同一个显示器上识别，所以要保证逻辑正确，最好回到最初的车牌页面或者图形图像页面。保证代码成功效率。

```
/**
    * 让mark值恢复markTemp,MarkResetWait跳出死循环
    * */
   public void MarkGo() {
     mark = markTemp;
   }
```

在最后我们还调用了一个方法，应该还记得在调用前面图形图像识别的时候，有一个MarkResetWait()方法吧，它让当前的线程暂停，跳转类去执行其他类的线程，我们其他类的线程执行完了已经回来了，理所应当就要让当前暂停的线程又重新活过来。

可能会在像现在疑惑到现在为什么发送个消息要调用这么多方法，要转这么多次，那是因为代码的复用性和代码的健壮性，这样代码可以公用而不是一个方法就要一个，减少了代码的臃肿。

```
// 图像识别
IdentifyColorShape();
```

在执行完翻页以后，叫执行图像识别了：

```
private void IdentifyColorShape() {
    // 保存形状颜色识别的信息
    // 颜色形状识别代码块
    // String uuid = UUID.randomUUID().toString();
    // FileUtil.savePhoto(bitmap, uuid + ".png");
    // 保存一次图片
    client.delay(1000);
    FileUtil.savePhoto(bitmap, MyGlobal.SHAPE_COLOR_IMAGE_NAME);
    String rString = "";
    long startTime = 0, endTime = 0;
    int nC = 1;
    try {
      // 直接转为mat类型传入到opencv识别图片，不再重复保存，每次保存耗时400毫秒
      Mat mat = new Mat();
      // List<Contour> contours;
      StatisticUtil sut = new StatisticUtil();
      // 识别1次耗时1.1秒
      for (int i = 0; i < nC; i++) {
        Utils.bitmapToMat(bitmap, mat);
        if (null == opcv) {// 初始化
          opcv = new OpenCvShapeColor();
          startTime = System.currentTimeMillis();
          contours_robot = opcv.run(mat, RobotCarActivity.this);
          endTime = System.currentTimeMillis() - startTime;
          // voiceController("初始化耗时" + endTime);
          System.out.println("oepncv初始化总耗时：" + endTime);
          // client.voiceController("图像识别初始化");
          return;
        } else {
          contours_robot = opcv.run(mat, RobotCarActivity.this);
        }
        sut.statistics(contours_robot, client);
        while (sut.hashMap.size() < 6) {// 如果识别到的图形个数小于6，翻页再次识别
          client.TFTDisplayImageUpDownRandom(2);
          Utils.bitmapToMat(bitmap, mat);
          contours_robot = opcv.run(mat, RobotCarActivity.this);
          sut.statistics(contours_robot, client);
        }
        client.voiceController(sut.hashMap.size() + "");
        // client.voiceController("图像识别成功");
      }
      if (nC > 1) {
        sut.dataComparison();
      }
      rString = sut.rString;
      // sut.saveResult(uuid);
      FileUtil.saveToSDCard(MyGlobal.SHAPE_COLOR_TXT_NAME,
```

```
            MyGlobal.SHAPE_COLOR_TXT_RESULT);
        sut.outputAllResults();
        MyGlobal.TK_K01 = sut.query("红色");
        MyGlobal.TK_M06_Shape = "A" + sut.query("矩形");
        MyGlobal.TK_M06_Shape = MyGlobal.TK_M06_Shape + "B"
            + sut.query("圆形");
        MyGlobal.TK_M06_Shape = MyGlobal.TK_M06_Shape + "C"
            + sut.query("三角形");
        MyGlobal.TK_M06_Shape = MyGlobal.TK_M06_Shape + "D"
            + sut.query("菱形");
        MyGlobal.TK_M06_Shape = MyGlobal.TK_M06_Shape + "E"
            + sut.query("五角星");
        // client.LEDtwoShow_6chars(MyGlobal.TK_M06_Shape);
      } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
      // 这是查询方法，会返回指定目标的int数量，语音播报所有图形写在查询方法里面了
      // voiceController(MyGlobal.M08);
      msgShow(MyGlobal.TK_K06 + "////" + rString + "////耗时: " + endTime);
    }
```

我们还是来解析方法，以来就掉了了一个client.delay(1000);

```
// 沉睡
  public void delay(int time) {
    try {
      Thread.sleep(time);
    } catch (InterruptedException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }
```

这个方法就是用来进行延迟等待，因为在进行图形处理是要等带图像完成，而代码执行是很快的，如果不延迟很可能就会将图像图像拍成模糊的，识别率就大大降低。

```
FileUtil.savePhoto(bitmap, MyGlobal.SHAPE_COLOR_IMAGE_NAME);
String rString = "";
long startTime = 0, endTime = 0;
int nC = 1;
```

这一段就是保存拍照成功的图片和定义好初始的变量。

```
/**
   * 保存图片
   *
   * @param b
   *            图片资源
   * @param strFileName
   *            图片名称
```

```
   * @throws IOException
   */
  public static void savePhoto(Bitmap b, String strFileName) {
    try {
      File file = new File(Environment.getExternalStorageDirectory(),
          strFileName);
      if (!file.isDirectory()) {
        file.createNewFile();
      }
      FileOutputStream fos = new FileOutputStream(file);
      if (fos != null) {
        b.compress(Bitmap.CompressFormat.PNG, 80, fos);
        fos.flush();
        fos.close();
      }
      file.canExecute();
    } catch (FileNotFoundException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }
```

这就是用于保存图片的方法，怎么保存的就不用我解释了吧，这个就是Java的图片保存文件的方法。有兴趣的就可以去查一查。这个方法我们就不用修改，只用知道这个是怎么保存保存在哪里。关于保存在哪，名字是什么就是关于传进啦的两个参数，第一个就是图片本身的bitmap对象，和一个string字符串，这个字符串就是我们保存的图片的名字，如果有兴趣查询就可以跟踪传进来的变量名，在前面说到的变量类MyGlobal类里面找到。

```
// 直接转为mat类型传入到opencv识别图片，不再重复保存，每次保存耗时400毫秒
      Mat mat = new Mat();
      // List<Contour> contours;
      StatisticUtil sut = new StatisticUtil();
      // 识别1次耗时1.1秒
      for (int i = 0; i < nC; i++) {
        Utils.bitmapToMat(bitmap, mat);
        if (null == opcv) {// 初始化
          opcv = new OpenCvShapeColor();
          startTime = System.currentTimeMillis();
          contours_robot = opcv.run(mat, RobotCarActivity.this);
          endTime = System.currentTimeMillis() - startTime;
          // voiceController("初始化耗时" + endTime);
          System.out.println("oepncv初始化总耗时: " + endTime);
          // client.voiceController("图像识别初始化");
          return;
        } else {
          contours_robot = opcv.run(mat, RobotCarActivity.this);
        }
        sut.statistics(contours_robot, client);
        while (sut.hashMap.size() < 6) {// 如果识别到的图形个数小于6，翻页再次识别
          client.TFTDisplayImageUpDownRandom(2);
```

```
            Utils.bitmapToMat(bitmap, mat);
            contours_robot = opcv.run(mat, RobotCarActivity.this);
            sut.statistics(contours_robot, client);
          }
          client.voiceController(sut.hashMap.size() + "");
          // client.voiceController("图像识别成功");
        }
        if (nC > 1) {
          sut.dataComparison();
        }
        rString = sut.rString;
```

这 一段代码就是对这个图片的分析处理，调用了opencv库的方法。最终获取到一个字符串，在字符串里面就是所分析的颜色和图形图像。具体如何使用opencv，那就要去学习一下了，毕竟opencv算是一种新的语言。

```
rString = sut.rString;
// sut.saveResult(uuid);
FileUtil.saveToSDCard(MyGlobal.SHAPE_COLOR_TXT_NAME,MyGlobal.SHAPE_COLOR_TXT_RESULT
);
sut.outputAllResults();
MyGlobal.TK_K01 = sut.query("红色");
MyGlobal.TK_M06_Shape = "A" + sut.query("矩形");
MyGlobal.TK_M06_Shape = MyGlobal.TK_M06_Shape + "B"+ sut.query("圆形");
MyGlobal.TK_M06_Shape = MyGlobal.TK_M06_Shape + "C"+ sut.query("三角形");
MyGlobal.TK_M06_Shape = MyGlobal.TK_M06_Shape + "D"+ sut.query("菱形");
MyGlobal.TK_M06_Shape = MyGlobal.TK_M06_Shape + "E"+ sut.query("五角星");
```

在这一些代码中就是将图形图像中的所有的形状个数或者颜色个数通过sut.query（）获取出来然后存储到全局变量里面，具体什么变量还是要去变量类MyGlobal类中寻找。这一任务的难点算是整个安卓负责的难点最大的两个，因为图形图像处理收到的外界因素太多，会导致分析错误等等问题，还可能会有调转摄像头还问完成就开始拍照等代码的顺序性问题，所以在做处理的时候，我们要细心点，笨一点的方法就是我多等一会在执行，但是这样的几率还是存在，或者就是代码优化，但是代码优化有涉及许多，所以这也是考虑的问题。总体来说图形图像识别算是个重大的难点。

81 LED显示

```
case 81:// LED数码管第二排显示6位字符
  // MyGlobal.TK_K06图形个数

  LEDtwoShow_6chars(MyGlobal.TK_K06);

  action();
  break;
```

如注解所描述的，这个是就是在比赛中要将摄像头拍照然后分析出来的图形图像按照题目规定的展示出来，代码中MyGlobal.TK_K06就是图形图像识别出来后将结果存储的变量名。然后通过LEDtwoShow_6chars()方法将这个变量传输给小车；

```
// LED第二排显示图形数量AaBbCc
  public void LEDtwoShow_6chars(String str) {
    String showInfo = str;
    int one = Integer.parseInt(showInfo.substring(0, 2), 16);
    int two = Integer.parseInt(showInfo.substring(2, 4), 16);
    int three = Integer.parseInt(showInfo.substring(4, 6), 16);
    digital(2, one, two, three);
  }
```

这个方法就是将传进来的字符串分割改变类型 然后在调用方法发送出去。

```
/**
   * 数码管显示数据LED显示
   *
   * @param i
   * @param one
   * @param two
   * @param three
   */
  public void digital(int i, int one, int two, int three) {// 数码管
    byte type = (byte) TYPE;
    TYPE = 0x04;
    if (i == 1) {// 数据写入第一排数码管
      MAJOR = 0x01;
      FIRST = (byte) one;
      SECOND = (byte) two;
      THRID = (byte) three;
    } else if (i == 2) {// 数据写入第二排数码管
      MAJOR = 0x02;
      FIRST = (byte) one;
      SECOND = (byte) two;
      THRID = (byte) three;
    }
    send();
    TYPE = type;
  }
```

这就是最后的发送步骤，该方法中有4个变量，除了第一个变量外，其他的都是底层所接收的展示的数值，而第一份变量则是关系到LED显示器的第几排显示的问题。

```
action(); //表示任务完成完毕，告知底层继续执行任务。
```

最后一行就是我用来解释的 //表示任务完成完毕，告知底层继续执行任务。这个任务的难点就是在于所获取到的变量是否有值。而是否获取到值那么就和其他任务有关系了，所以在这种情况我们也会做一些预防措施，放置因为没有值所以Android崩溃，那就是我们会在变量设置默认的值，或者在执行方法时，进行判定等等，方法多样就看看自己喜好，唯一的准则就是将程序顺利的走下去，官方点就是保持程序的健壮性。

82 识别交通标志

```
case 82:// 识别交通标志
  state_camera(1);
  identifyTraffic();
  // state_camera(1);
  // RobotCarActivity.voiceController(MyGlobal.M06);
  action();
  // state_camera(1);
  break;// ===================================
```

现在也是一大难点的识别交通标志所在了，看到代码中有一个state_camera(1);

```
/**
   * 调整摄像头
   */
  public void state_camera(int n) {
    msgShow("调整摄像头CALL" + n);
    switch (n) {
    case 1:// 向前复位
      RobotCarActivity.state_camera = 8;
      break;
    case 2:
      RobotCarActivity.state_camera = 9;
      break;
    case 3:
      RobotCarActivity.state_camera = 10;
      break;

    default:
      break;
    }
    delay(3000);
  }
```

正如代码里注释所说这个方法就是我们调用摄像头进行转头的方法，在比赛中因为各种环境因素影响，或许小车转向不方便容易出现意外问题，这时调整摄像头就比调整小车的位置就好多了，所以在比赛中，我们就会预设的在运行前将摄像头转的角度设好位置，然后在开始设定。当然能设定的也最多只能是3个位置，这还包括了复位的设置，所以额外的设置就是只有2个了。

然后继续摄像头调整后又会执行：

```
identifyTraffic();
```

这个方法就是我们这个任务的主体了：

```
/**
   * 识别交通标志物
   */
  private void identifyTraffic() {
    msgShow("识别交通标志物");
    // 状态82
    carControlHandler.sendEmptyMessage(0x50);
    MarkResetWait();
  }
```

还是和前面识别颜色的方式一样用handler传递消息启动任务线程，然后又暂停当前线程。

```
case 0x50:// 识别交通灯，并发送解析
  IdentifyTrafficSignal();

  client.MarkGo();
  break;// ====================================
```

在跳转到这个线程后就会执行当前要做的方法IdentifyTrafficSignal();

```
/**
   * 2018/5/17 王渔 红黄绿灯识别 准确率90%以上
   *
   * @throws Exception
   */
  int IdentifyTrafficSignal_cunt = 0;//识别交通等的分路

  public void IdentifyTrafficSignal() {
    IdentifyTrafficSignal_cunt = 0;
    System.out.println("识别交通");
    client.trafficLightStop();
    client.delay(1500);
    MyGlobal.M06 = "空";
    Mat mat = new Mat();
    Utils.bitmapToMat(bitmap, mat);
    Imgproc.cvtColor(mat, mat, Imgproc.COLOR_BGRA2RGB);
    try {
      MyGlobal.M06 = OpenCvTrafficLight.run(mat);
    } catch (Exception e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    // client.voiceController(MyGlobal.M06);
    if (MyGlobal.M06.equals("空") && IdentifyTrafficSignal_cunt < 2) {
      IdentifyTrafficSignal_cunt++;
      // client.voiceController(MyGlobal.M06);
      IdentifyTrafficSignal();
    } else {
      client.trafficLightMsg(MyGlobal.M06);
      // client.voiceController(MyGlobal.M06);
    }
  }
```

这个方法内容就算是重点中的重点，因为在我们上次的Android中，主要管理的就是摄像头的处理，然后就是对图像的处理，最后才是数据传递。

```
/**
    * 交通灯停止
    *
    * @param hex
    */
   public void trafficLightStop() {
     MAJOR = 0xC1;
     FIRST = 0;
     SECOND = 0;
     THRID = 0;
     send();
     msgShow("发送指令");
   }
```

一开始，在比赛中交通灯是一致变换不定的，我们要识别就要现将它停止下来进行再进行拍照识别。所以我们要将停止交通灯的通讯协议发送给小车底层，另外因为通讯的问题在发送信息是我们要延迟时间，让小车反应过来。

```
  MyGlobal.M06 = "空";
  Mat mat = new Mat();
  Utils.bitmapToMat(bitmap, mat);
  Imgproc.cvtColor(mat, mat, Imgproc.COLOR_BGRA2RGB);
  try {
    MyGlobal.M06 = OpenCvTrafficLight.run(mat);
```

这一段代码我们就是在用opencv对拍照出来的照片进行处理了。

```
public static boolean println = true;

  public static long startTime = 0, endTime = 0;
  public static List<Contour> contours;
  public static String result = "空";

  public static String run(Mat org) throws Exception {
    result = "空";
    Mat orgImg = org;
    Mat orgImg1 = orgImg;
    Mat orgGrayImg = orgImg;
    // orgGrayImg=Proc.PengZhang(orgGrayImg,1);
    orgGrayImg = Proc.FuShi(orgGrayImg, 1);
    // orgGrayImg = Proc.getWhiteBalance(orgGrayImg);
    orgGrayImg = Proc.medianBlur(orgGrayImg, 1);
    orgGrayImg = Proc.gray(orgGrayImg);
    Mat circles = new Mat();
    int minDist = orgGrayImg.rows() / 10;
    // System.out.println("minDist=" + minDist);
    // Imgcodecs.imwrite("orgImg.jpg", orgImg);
    // Imgcodecs.imwrite("orgGrayImg.jpg", orgGrayImg);
```

```
    Imgproc.HoughCircles(orgGrayImg, circles, Imgproc.CV_HOUGH_GRADIENT, 1,
        minDist, 120, 30, 10, 50);
    for (int i = 0; i < circles.cols(); i++) {
      Point center = new Point(circles.get(0, i)[0], circles.get(0, i)[1]);
      int radius = (int) circles.get(0, i)[2];
      Scalar color = new Scalar(0, 67, 255);
      Mat circle = new Mat(radius * 2, radius * 2, CvType.CV_8UC3);
      for (int j = 0; j < circle.rows(); j++) {
        for (int j2 = 0; j2 < circle.cols(); j2++) {
          int x, y;
          x = (int) center.y - radius + j;
          y = (int) center.x - radius + j2;
          double[] temp = orgImg1.get(x, y);
          circle.put(j, j2, temp);
        }
      }
      Proc.filterColor(circle);
      Imgcodecs.imwrite(Environment.getExternalStorageDirectory() + "/"
          + i + "circle.jpg", circle);
      List<Mat> mv = new ArrayList<Mat>();
      Core.split(circle, mv);
      double b, g, r;
      b = Core.mean(mv.get(0)).val[0];
      g = Core.mean(mv.get(1)).val[0];
      r = Core.mean(mv.get(2)).val[0];
      double mean = (b + g + r) / 3;
      if (Math.abs(mean - b) < 10 && Math.abs(mean - r) < 10
          && Math.abs(mean - g) < 10) {
        // System.out.println("白");
        // result = "白色";
      } else if (b > r && b > g) {
        // System.out.println("蓝");
        // result = "蓝色";
      } else if (r > g && r > b && Math.abs(r - g) > 30) {
//        System.out.println("红");
        if (result.equals("空"))
          result = "红色";
      } else if (Math.abs(r - g) < 20 && (r + g) > 100) {
//        System.out.println("黄");
        if (result.equals("空"))
          result = "黄色";
      } else if (g > r) {
//        System.out.println("绿");
        if (result.equals("空"))
          result = "绿色";
      }
      if (!result.equals("红色") && !result.equals("黄色")
          && !result.equals("绿色")) {
      }
    }
//    System.out.println("最终: " + result);
    return result;

```

```
  }

}
```

这一大段应该就是怎么将拍出的交通灯用颜色的方式传送回来了，其中到底是如何具体处理的图片方式那就要在对opencv有了足够的了解后在说了。

```
if (MyGlobal.MO6.equals("空") && IdentifyTrafficSignal_cunt < 2) {
      IdentifyTrafficSignal_cunt++;
      // client.voiceController(MyGlobal.MO6);
      IdentifyTrafficSignal();
    } else {
      client.trafficLightMsg(MyGlobal.MO6);
      // client.voiceController(MyGlobal.MO6);
    }
```

这一部分代码就是在将图片处理之后，对图片的结果进行分析或者说是多次确定，如果是空或其他的都或者分析次数不够，那么久将再次执行本方法再次处理一次图片，这样才能使多次分析，有次数来解决精度的问题。然后就将拍摄的图片处理后的细信息就发送小车。

```
/**
    * 交通灯识别结果 1红2绿3黄
    *
    * @param hex
    */
  public void trafficLightMsg(String str) {
    int n = -1;
    switch (str) {
    case "红色":
      n = 1;
      break;
    case "绿色":
      n = 2;
      break;
    case "黄色":
      n = 3;
      break;

    default:
      break;
    }
    MAJOR = 0xC2;
    FIRST = (short) n;
    SECOND = 0;
    THRID = 0;
    send();
    msgShow("发送指令");
  }

```

这就是将最后的颜色结果进行转换成数字类型发送个小车底层。

83 识别车牌

这就是在识别颜色中提到过一句的车牌识别。

```
case 83:// 识别车牌
      // state_camera(2);
      identifyPlateNumber();
      // state_camera(1);
      action();
      // state_camera(3);
      break;// ====================================
```

当底层将识别车牌的序号发送给我们之后，我们上层Android就会自动判定到现在的方法中。

```
identifyPlateNumber
```

这就是像前面识别颜色类似的，调用方法传递消息。

```
/**
   * 识别车牌
   */
  private void identifyPlateNumber() {
    msgShow("识别车牌");
    // 状态83
    carControlHandler.sendEmptyMessage(0x35);
    MarkResetWait();
  }
```

handler传递消息调用对应的线程方式。并且暂停当前的线程。

```
case 0x35:// 车牌识别，并发送解析
        if (MyGlobal.TK_M06_Shape.equals("")) {
          // voiceController("车牌识别");
          client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
          // MyGlobal.TK_M06 = "";
          // 第一次识别车牌
          IdentifyPlateNumber();
          // 如果识别失败
          if (MyGlobal.TK_M06_Shape.equals("")) {
            // voiceController("识别失败，翻页");
            // 翻页
            client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
            // client.delay(10000);
            // 第二次识别车牌
            IdentifyPlateNumber();
          }
          // 如果第一次识别成功则翻页
          // TFTDisplayImageUpDownRandom(2);// TFT下翻页
          // 如果播报识别结果
          if (MyGlobal.TK_M06_Shape.equals("")) {
            // client.voiceController("车牌识别失败");
```

```
            // MyGlobal.TK_M06 = "A123B4";
          } else {
            // voiceController("识别完成");
            client.sendCarID(MyGlobal.TK_M06_Shape);
            client.voiceController("车牌为" + MyGlobal.TK_M06_Shape);
            FileUtil.savePhoto(bitmap,
                MyGlobal.PLATE_NUMBER_IMAGE_NAME);
          }
          if (!MyGlobal.TK_M06_Shape.equals("")) {
            // client.voiceController("车牌识别成功");
          }
        }
        client.MarkGo();
        break;// ===================================
```

一样的来解析一下代码，从一开始我们接收到handler消息后会进入到这个方法，第一步：

```
if (MyGlobal.TK_M06_Shape.equals("")) {
          // voiceController("车牌识别");
          client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
          // MyGlobal.TK_M06 = "";
          // 第一次识别车牌
          IdentifyPlateNumber();
          // 如果识别失败
          if (MyGlobal.TK_M06_Shape.equals("")) {
            // voiceController("识别失败，翻页");
            // 翻页
            client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
            // client.delay(10000);
            // 第二次识别车牌
            IdentifyPlateNumber();
          }
```

我们进来方法后就会对已经固定写好存储车牌号的全局变量(这个变量最好不要自己写，用变量类里面的，有利于代码的维护和再利用)，所以一进来就判断变量里是否为空，当然我们一开始变量肯定是空的，所以肯定会进入这个方法，如果这个方法不进入那证明在哪里肯定调用了它。就排错吧。

成功进入这个方法后开始说的，因为为我们图形图像识别可能和车牌识别实在同一个显示器，而这个显示器都是有规律的一页车牌一页图形图像，所以先主动地翻页一次预防别人用过后没有重启显示器。所以会执行：

```
client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
```

翻页结束后就开始正式的执行识别车牌的方法：

```
IdentifyPlateNumber();//车牌识别
```

```
/**
   * 识别车牌
   */
  String plateNumber = "";

```

```
  private void IdentifyPlateNumber() {
    FileUtil.savePhoto(bitmap, MyGlobal.PLATE_NUMBER_IMAGE_NAME);
    try {
      plateNumber = "";
      plateNumber = new PlateNumberUtils().asyPlateNumber(
          MyGlobal.PLATE_NUMBER_IMAGE_NAME, RobotCarActivity.this);
      if (!plateNumber.equals("")) {
        MyGlobal.TK_M05_CarID = plateNumber;
        client.sendCarID(MyGlobal.TK_M05_CarID);
        // 保存车牌信息
        FileUtil.saveToSDCard(MyGlobal.PLATE_NUMBER_TXT_NAME,
            MyGlobal.TK_M05_CarID);
        FileUtil.savePhoto(bitmap, MyGlobal.PLATE_NUMBER_IMAGE_NAME);
      }
    } catch (Exception e) {

      e.printStackTrace();

    } finally {

      // client.MarkGo();
    }
  }
```

这一大段就是车牌识别的方式了，主体的还是将图片资源拿到一个分析类邻面进行数据处理最后会返回一个字符串在获取到字符串后进行保存然后将保存的字符串发送给小车。

```
client.sendCarID(MyGlobal.TK_M05_CarID);//发送给小车
```

```
/**
   * 发送车牌到底层
   */
  public void sendCarID(String str) {
    System.out.println(str);
    char[] chars = str.toCharArray();
    MAJOR = 0xA3;
    FIRST = (short) chars[0];
    SECOND = (short) chars[1];
    THRID = (short) chars[2];
    send();
    delay(1000);
    MAJOR = 0xA4;
    FIRST = (short) chars[3];
    SECOND = (short) chars[4];
    THRID = (short) chars[5];
    send();
    msgShow("发送车牌");
  }
```

发送之后就是保存了，将文件保存到移动端目录中。

```
// 保存车牌信息
    //保存到文件
    FileUtil.saveToSDCard(MyGlobal.PLATE_NUMBER_TXT_NAME,MyGlobal.TK_M05_CarID);
    //保存图片
    FileUtil.savePhoto(bitmap, MyGlobal.PLATE_NUMBER_IMAGE_NAME);
```

这样车牌就算是识别完成，但可能有其他因素原因导致车牌识别错误

```
if (MyGlobal.TK_M06_Shape.equals("")) {
        // voiceController("识别失败，翻页");
        // 翻页
        client.TFTDisplayImageUpDownRandom(2);// TFT下翻页
        // client.delay(10000);
        // 第二次识别车牌
        IdentifyPlateNumber();
      }
```

所以会又有一个判定，判定识别是否成功，如果没有成功就是翻页再次识别。

```
// 如果第一次识别成功则翻页
          // TFTDisplayImageUpDownRandom(2);// TFT下翻页
          // 如果播报识别结果
          if (MyGlobal.TK_M06_Shape.equals("")) {
            // client.voiceController("车牌识别失败");
            // MyGlobal.TK_M06 = "A123B4";
          } else {
            // voiceController("识别完成");
            client.sendCarID(MyGlobal.TK_M06_Shape);
            client.voiceController("车牌为" + MyGlobal.TK_M06_Shape);
            FileUtil.savePhoto(bitmap,
                MyGlobal.PLATE_NUMBER_IMAGE_NAME);
          }
          if (!MyGlobal.TK_M06_Shape.equals("")) {
            // client.voiceController("车牌识别成功");
          }
```

在车牌识别完成后，因为比赛可能要播报出识别的车牌，所以我们就要将其用语音 播报芯片或者音乐播报器播报出来，这就用到了又要发送一个协议给底层。其中我们在这播报钱也再次的判断储存车牌号的变量是否为空如果为空那么我们就将其忽略默认播报一个，如果不为空那么我们就正式播报。

```
/**
   * 发送车牌到底层
   */
  public void sendCarID(String str) {
    System.out.println(str);
    char[] chars = str.toCharArray();
    MAJOR = 0xA3;
    FIRST = (short) chars[0];
    SECOND = (short) chars[1];
    THRID = (short) chars[2];
    send();
```

```
    delay(1000);
    MAJOR = 0xA4;
    FIRST = (short) chars[3];
    SECOND = (short) chars[4];
    THRID = (short) chars[5];
    send();
    msgShow("发送车牌");
  }
```

```
/**
   *
   * 语音播报控制
   *
   * @param src
   *            源数据
   * @throws UnsupportedEncodingException
   */
  public void voiceController(String src) {
    int count = 0;
    try {

      byte[] sbyte = bytesend(src.getBytes("GBK"));
      System.out.println("sbyte.length" + sbyte.length);
      send_voice(sbyte);
    } catch (Exception e) {
      // TODO: handle exception
    }
    // yanchi(3000);
    while (rbyte[2] != 79 && count < 4000) {

      System.out.println("mByte[2]===" + rbyte[2]);
      count += 100;
      delay(100);
      // voiceController(src);
    }
    if (count > 4000) {
      voiceController(src);
      return;
    }
    stop();
  }
```

这就是将传进来的字符串参数进行格式化归类，然后转换成字节流，发送给底层小车。

```
public void send_voice(byte[] textbyte) {
  try {
    // 发送数据字节数组
    if (RobotCarActivity.isSerialPort) {
      if (RobotCarActivity.sPort != null) {

        RobotCarActivity.sPort.write(textbyte, 5000);
      }
```

```
      // SerialOutputStream.write(sbyte, 0, sbyte.length);
      // SerialOutputStream.flush();
    } else if (socket != null && !socket.isClosed()) {
      bOutputStream.write(textbyte, 0, textbyte.length);
      bOutputStream.flush();
    }
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

这就是将转换好的子节发送给小车，让其语音播报出来。

```
// 复位
  public void stop() {// 状态、码盘所有的清零，可以用于两个状态相同的动作之间
    MAJOR = 0x01;
    FIRST = 0x00;
    SECOND = 0x00;
    THRID = 0x00;
    send();
    delay(1000);
    msgShow("小车已复位");

  }
```

在播报结束后会执行一个方法就是复位，看着复位的介绍发现，会将小车的所有的 状态、码盘所有的清零，这样不会导致小车停止么，原因很简单，因为在使用语音播报器的时候，语音播报完成后会返回一个状态值如果，这个状态值不清空，那么我们程序就会卡死，那么久将其复位解决。

84 立体显示

```
case 84:// 立体显示
      // for (int i = 0; i < 3; i++) {
      // // A矩形B圆形C三角形E五角星(使用车牌通信协议显示)AaBbCcDdEe
      // // getRbyteM07(10);
      // if (!MyGlobal.Shape.equals("")) {
      // threeDisplayShowPlateNumber(MyGlobal.Shape);
      // } else {
      // threeDisplayShowPlateNumber(MyGlobal.TK_M06_Shape);
      // }
      // delay(500);
      // }
      action();
      break;// ===================================
```

到立体显示就很简单了，还是像往常一样获取底层返回的指令，是否是立体显示的指令，如果是就进入线程开始调用方法。

```
threeDisplayShowPlateNumber(MyGlobal.TK_M06_Shape);
```

```
/**
```

```
     *
     * 立体显示车牌+从车要停放的坐标位置
     *
     * @param plateNumberCoordinate
     */
    public void threeDisplayShowPlateNumber(String plateNumberCoordinate) {
      String str = plateNumberCoordinate;
      System.out.println("str:" + str);
      System.out.println("str.length():" + str.length());
      if (str.length() < 1) {
        System.out.println("字符串长度小于1");
        return;
      }
      if (str.length() < 8) {
        System.out.println("字符串长度小于8");
        return;
      }
      System.out.println("str.length()" + str.length());
      short[] li = StringToBytes(str);
      System.out.println("li.length" + li.length);
      data[0] = 0x20;
      data[1] = (short) (li[0]);
      data[2] = (short) (li[1]);
      data[3] = (short) (li[2]);
      data[4] = (short) (li[3]);
      infrared_stereo(data);
      data[0] = 0x10;
      data[1] = (short) (li[4]);
      data[2] = (short) (li[5]);
      data[3] = (short) (li[8]);
      data[4] = (short) (li[9]);
      infrared_stereo(data);
    }
```

这个方法主要的还是显示车牌信息或者车库位置，所以一般调用两次或者底层自己行动，因为在前面的方法中，我们已经将数据发送到了底层。现在这个方法就是将传进来的字符串进行类型转换，然后发送给小车进行执行任务。

85 打开报警器

```
case 85:// 打开报警器

      //应该还要讲获取的信息进行整合用算法解答出来在调用
      policeController();
      action();
      break;// ====================================
```

应该还要讲获取的信息进行整合用算法解答出来在调用policeController();这个就是比赛中最核心的地方了，因为涉及到了算法的事情，我们要写一个算法进行出力然后整合成6位16进制数进行处理，因为现在去年代码也没具体先写出来，我也不知道如何具体具体解释，不过大致的和java的调用获取对象调用方法类似。而我们就是要根据题目写一个解答的java(类似于ACM比赛或者蓝桥杯比赛的模式)，我们写完这个解答类后，将在比赛中获取到的信息(如:颜色个数、指定图形个数、二维码信息等等),将这些信息输入到解答类中后，得出一个结果，然后将结果传输给底层小车，用来打开报警器就是这样的。

86 识别二维码

```
case 86:// 识别二维码
      // state_camera(3);
      sendQrCodeMsg(0x20, 1);
      if (MyGlobal.TK_M01.equals("")) {
        RobotCarActivity.cameraCommandUtil.postHttp(
            RobotCarActivity.IPCamera, 0, 1);// 抬头
      }
      if (MyGlobal.TK_M01.equals("")) {
        RobotCarActivity.cameraCommandUtil.postHttp(
            RobotCarActivity.IPCamera, 2, 1);// 低头
      }
      // state_camera(1);
      // 换位算法
      decode(MyGlobal.TK_M01);
      action();
      state_camera(1);
      break;// ===================================
```

这次的还是和前面类似要用handler用来传递消息。

```
sendQrCodeMsg(0x20, 1);
```

但是这次方法给传递的方式不一样，传递了两个参数，第一个0x20这个参数就和前面的消息传递很相同了就是任务线程的消息标识，第二个1这个参数呢，这个参数是因为在比赛中我们需要识别几个二维码，而这个参数就是识别第几个二维码的标识了。

```
/**
   * 发送识别二维码的指令
   *
   * @param what
   *            识别二维码的指令
   * @param arg1
   *            识别第几张二维码指令 arg1=1则存在MyGlobal.M04中 arg1=2则存在MyGlobal.M012
中
   */
  private void sendQrCodeMsg(int what, int arg1) {

    msgShow("识别二维码" + arg1);
    Message msg = new Message();
    msg.what = what;
    msg.arg1 = arg1;

    carControlHandler.sendMessage(msg);
    MarkResetWait();
  }
```

这就是具体的将消息封装然后将二维码发送出去了。

```
case 0x20:// 扫描二维码，发送解析
    IdentifyQrCode(msg.arg1);

    break;// ====================================
```

这就是传递过来后，接收到消息的线程，然后开始调用识别方法。

```
/**
    * 2018/3/10 王渔 4秒内循环识别20次二维码，超时则跳出循环
    *
    * @param arg1
    *            识别的是第几个二维码
    */
   private void IdentifyQrCode(final int arg1) {
     MyGlobal.M12 = "";
     MyGlobal.M04 = "";
     MyGlobal.TK_M01 = "";
     Timer_flag = 0;
     result_qr = null;

     FileUtil.savePhoto(bitmap, MyGlobal.QRCODE_IMAGE_NAME);
     try {
       FileUtil.saveToSDCard(MyGlobal.QRCODE2_TXT_NAME, "空");
     } catch (IOException e1) {
       // TODO Auto-generated catch block
       e1.printStackTrace();
     }

     if (bitmap == null) {
       Log.i("error", "扫描二维码时未获取摄像头图像");
       return;
     }
     timer = new Timer();
     timer.schedule(new TimerTask() {
       @Override
       public void run() {
         // FileUtil.savePhoto(bitmap, Timer_flag
         // + MyGlobal.QRCODE2_IMAGE_NAME);
         msgShow("正在识别二维码" + Timer_flag);
         RGBLuminanceSource rSource = new RGBLuminanceSource(bitmap);
         try {
           result_qr = QRCodeUtil.resolveQRCode(rSource);

           if (result_qr != null) {// 当识别结果不为空
             if (arg1 == 1) {
               MyGlobal.M04 = result_qr;
               MyGlobal.TK_M01 = result_qr;
               FileUtil.savePhoto(bitmap,
                   MyGlobal.QRCODE_IMAGE_NAME);
               FileUtil.saveToSDCard(MyGlobal.QRCODE_TXT_NAME,
                   result_qr);
             } else if (arg1 == 2) {
```

```
                MyGlobal.M12 = result_qr;
                FileUtil.savePhoto(bitmap,
                    MyGlobal.QRCODE2_IMAGE_NAME);
                FileUtil.saveToSDCard(MyGlobal.QRCODE2_TXT_NAME,
                    result_qr);
              }
              msgShow("识别二维码成功");
              Timer_flag = 0;
              client.MarkGo();
              timer.cancel();
            }

          } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
          }
          // 在try catch里面Timer_flag++无效
          Timer_flag++;
          if (Timer_flag == 50 && result_qr == null) {
            Timer_flag = 0;
            msgShow("识别二维码超时");
            client.MarkGo();
            timer.cancel();
          }
        }// run()
      }, 0, 200);// schedule()
    };// IdentifyQrCode
```

这就是识别二维码方法的整体方法。

```

    MyGlobal.M12 = "";
    MyGlobal.M04 = "";
    MyGlobal.TK_M01 = "";
    Timer_flag = 0;
    result_qr = null;

```

这就是一开始对存储识别结果或者识别次数的标记等的变量的初始化。

```
FileUtil.savePhoto(bitmap, MyGlobal.QRCODE_IMAGE_NAME);
    try {
      FileUtil.saveToSDCard(MyGlobal.QRCODE2_TXT_NAME, "空");
    } catch (IOException e1) {
      // TODO Auto-generated catch block
      e1.printStackTrace();
    }

    if (bitmap == null) {
      Log.i("error", "扫描二维码时未获取摄像头图像");
      return;
    }
```

这一段代码就是图片的的判别或保存了，我们先执行的方式就是将突变进行保存，然后在保存一个文本为空的txt文件，在识别我们的图片是否获取成功。如果失败将跳出整个方法。

```
    timer = new Timer();
    timer.schedule(new TimerTask() {
      @Override
      public void run() {
        // FileUtil.savePhoto(bitmap, Timer_flag
        // + MyGlobal.QRCODE2_IMAGE_NAME);
        msgShow("正在识别二维码" + Timer_flag);
        RGBLuminanceSource rSource = new RGBLuminanceSource(bitmap);
        try {
          result_qr = QRCodeUtil.resolveQRCode(rSource);

          if (result_qr != null) {// 当识别结果不为空
            if (arg1 == 1) {
              MyGlobal.M04 = result_qr;
              MyGlobal.TK_M01 = result_qr;
              FileUtil.savePhoto(bitmap,
                  MyGlobal.QRCODE_IMAGE_NAME);
              FileUtil.saveToSDCard(MyGlobal.QRCODE_TXT_NAME,
                  result_qr);
            } else if (arg1 == 2) {
              MyGlobal.M12 = result_qr;
              FileUtil.savePhoto(bitmap,
                  MyGlobal.QRCODE2_IMAGE_NAME);
              FileUtil.saveToSDCard(MyGlobal.QRCODE2_TXT_NAME,
                  result_qr);
            }
            msgShow("识别二维码成功");
            Timer_flag = 0;
            client.MarkGo();
            timer.cancel();
          }

        } catch (Exception e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
        // 在try catch里面Timer_flag++无效
        Timer_flag++;
        if (Timer_flag == 50 && result_qr == null) {
          Timer_flag = 0;
          msgShow("识别二维码超时");
          client.MarkGo();
          timer.cancel();
        }
      }// run()
    }, 0, 200);// schedule()
```

这一段就是对二维码的正式识别，因为我们要保证二维码的正确性，我们都采用了多次识别的方式。所以在以这个目的为前提下，我们开启了Timer类的定时器，如果当循环次数和图像识别成功后就将取消定时器跳出循环所以在这方法中开始了。

```
RGBLuminanceSource rSource = new RGBLuminanceSource(bitmap);
```

这个方法就是二维码图片的处理，只获取二维码部分的图片。

```
result_qr = QRCodeUtil.resolveQRCode(rSource);
```

这个方法就是正式的解析二维码中的类容并将结果返回回来。

```
if (result_qr != null) {// 当识别结果不为空
            if (arg1 == 1) {
              MyGlobal.M04 = result_qr;
              MyGlobal.TK_M01 = result_qr;
              FileUtil.savePhoto(bitmap,
                  MyGlobal.QRCODE_IMAGE_NAME);
              FileUtil.saveToSDCard(MyGlobal.QRCODE_TXT_NAME,
                  result_qr);
            } else if (arg1 == 2) {
              MyGlobal.M12 = result_qr;
              FileUtil.savePhoto(bitmap,
                  MyGlobal.QRCODE2_IMAGE_NAME);
              FileUtil.saveToSDCard(MyGlobal.QRCODE2_TXT_NAME,
                  result_qr);
            }
            msgShow("识别二维码成功");
            Timer_flag = 0;
            client.MarkGo();
            timer.cancel();
          }
```

剩下的就是将判断结果是否为空，不为空，我们再根据我们传进来的是识别第几个二维码进行对应的文件保存结果和将结果保存在全局变量上等待着使用。

如果识别成功那么就会跳出这个定时器。继续执行方法，重新开始线程。

87 TFT显示器显示内容

```
case 87:// TFT显示
      TFTDisplayPlateNumber(MyGlobal.TK_M06_Shape);
      delay(5000);
      // RFID卡坐标显示
      getRbyteM07(10);
      TFTDisplayPlateNumber("RFID" + MyGlobal.RFID_POSITION);
      break;
```

最后这个方法就是将信息展示在TFT显示器上，比赛中，我们有许多显示的数据，具体能在什么设备上展示，那就要配合着实体进行设计了，不过展示数据的方式都是大致一样的。

```
/**
    *
    * TFT显示车牌
    *
    * @param plateNumber
    */
   public void TFTDisplayPlateNumber(String plateNumber) {

     char[] plateNumberChar = plateNumber.toCharArray();

     TFT_LCD(0x20, plateNumberChar[0], plateNumberChar[1],
         plateNumberChar[2]);
     delay(500);
     TFT_LCD(0x21, plateNumberChar[3], plateNumberChar[4],
         plateNumberChar[5]);

   }
```

这就是将传进来的数据格式化为char数组的形式，而且因为TFT只能显示3位数据，所以一般6位一个的数据就要分发两次。

```
public void TFT_LCD(int MAIN, int KIND, int COMMAD, int DEPUTY) {
    byte type = (byte) TYPE;
    TYPE = (short) 0x0B;
    MAJOR = (short) MAIN;
    FIRST = (byte) KIND;
    SECOND = (byte) COMMAD;
    THRID = (byte) DEPUTY;
    System.out.println(1);
    send();
    System.out.println(2);
    TYPE = type;
  }
```

这就是具体的将数据发送给底层的方式。

总结

目前为止在整个Android项目的介绍算是比较完整的，还有细微的可能就是如何对图片进行处理的，这个就要自己对opencv有点了解。现在如果再具体讲那个就又是很多内容了。有兴趣可以去了解一下，不过因为我们比赛要识别颜色图形图像，所以可能也会具体修改，因为要优化嘛，但那也是具体情况具体发挥。具体就要看临场能力了。