### 硬件部分

* Arduino Mega2560 单片机开发板



图1：ARDUINO MEGA2560

小车的控制部件采用Arduino Mega2560单片机平台，控制端手机发送的控制信号由它接收，并以此控制车载电机和舵机，进而使得小车按照手机所发出的控制指令确定运动方向。

* HC-06主从一体蓝牙模块

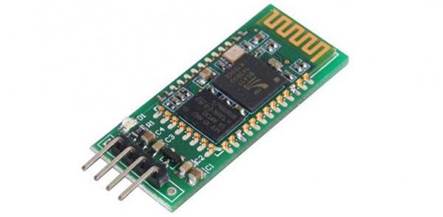


图2：HC-06蓝牙模块

HC-06主从一体蓝牙模块将主模块与从模块相结合，既能接收配对请求，也能发送配对请求。本模块所采用的是低成本、低功耗CSR芯片，支持蓝牙v2.0协议标准，当终端处在空旷环境下时，它可实现10M内的通讯，目前的应用量相当大。

我们组的小车中，我们用HC-06与控制端手机配对，以此接收手机传递的字符串，并通过单片机的RX、TX端口与单片机进行通信。

* Futaba S3003舵机



图3：FUTABA S3003 舵机

舵机，也称为伺服电机，主要由直流电机、减速齿轮组、传感器和控制电路组成，可根据接收到的PWM信号调整转过的角度。

我们小组的小车利用舵机来控制小车前进的方向，使用的最大角度范围约60°。

* L9110两路电机驱动

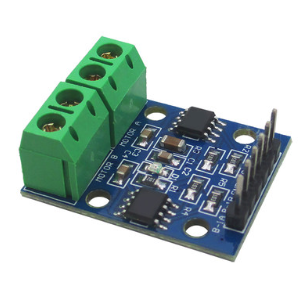


图4：L9110两路电机驱动

该电机驱动模块这双L9110S的电机驱动，供电电压介于2.5V~12V之间，最大工作电流为0.8A，可同时驱动2个直流电机。

在实际应用中，单片机端子能提供的最大电流其实比较有限，这会限制小车的运动速度，我们小组使用这个电机驱动模块，也是出于这个原因，控制信号由单片机提供，而小车的电源则直接由电池供给。

* Android 6.0手机两台



图5： Android 6.0手机

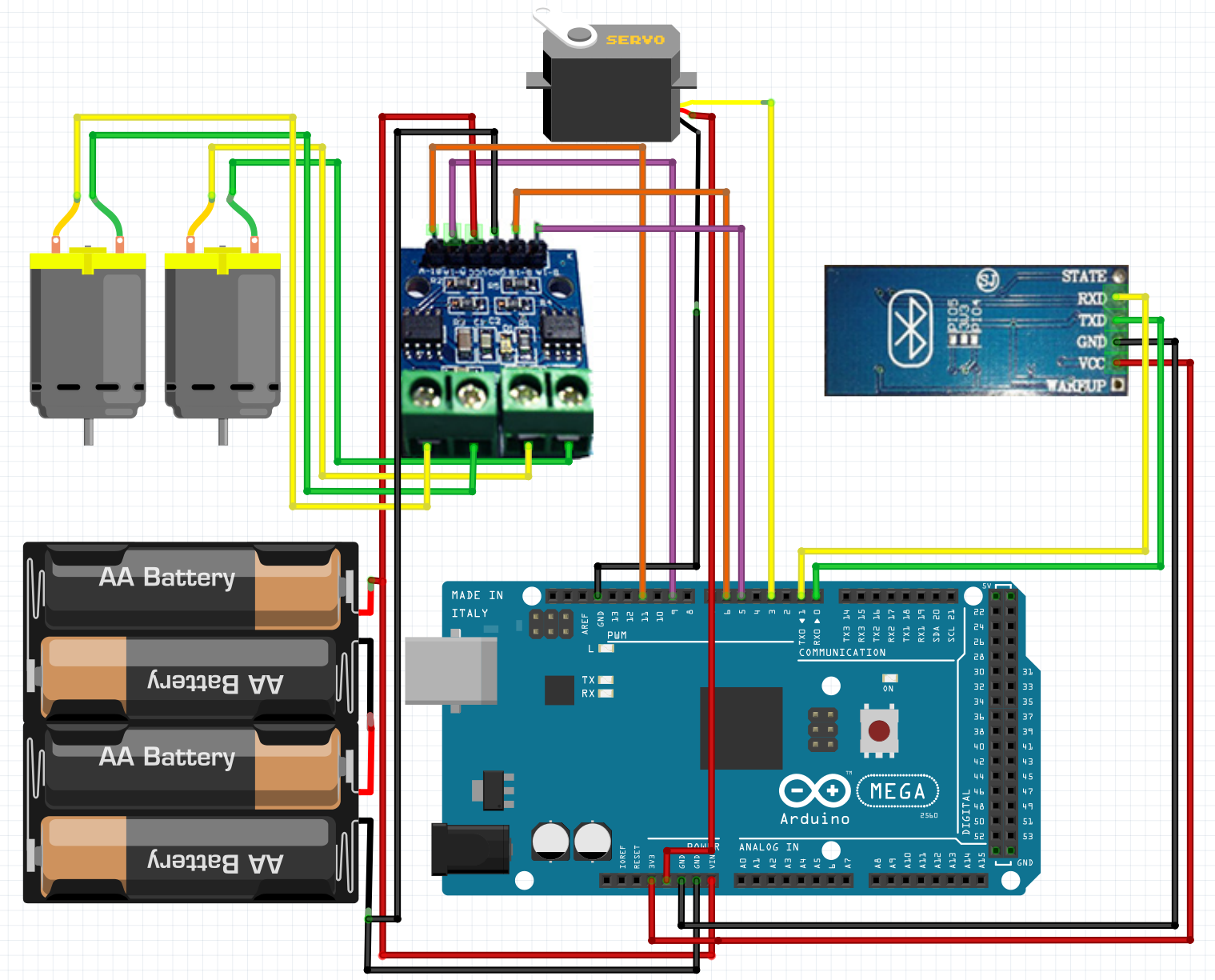
Android 6.0系统的两台手机，分别为小车端和控制端。小车端手机固定在小车上，控制端手机用于手持操作。

小车端手机和控制端手机通过WiFi互相传输视频信息。控制端手机可对小车端手机在运动过程中实时拍摄的视频做出分析，通过蓝牙向单片机平台发送控制信号，决定小车的运动方向。

* 整车组合

我们小车端电路系统连线如上图所示，通过在不同端口置高电位或者低电位，设置不同的PWM来控制电机、舵机，进而控制小车的运动。

因为移动电源占据的空间和重量都相对过大，用它为Arduino系统供电比较麻烦，所以我们小组采用自制电源进行供电。此外由于电源本身会导致Arduino对电机供电不足的问题，我们又另外使用了电机驱动板，使电机可获得充足的供电。



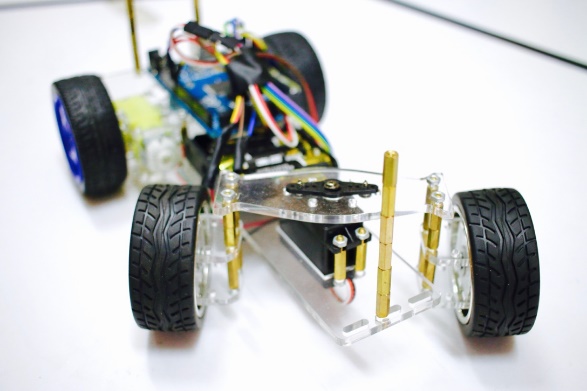
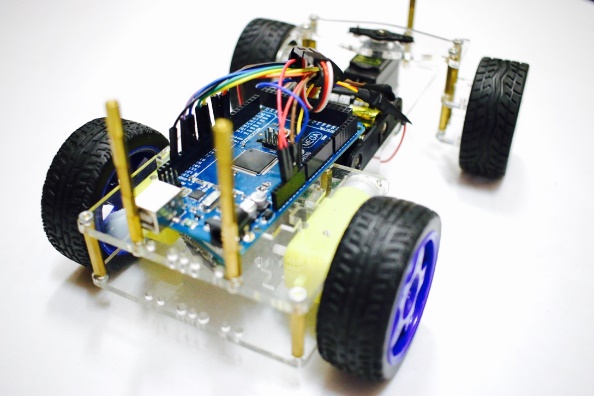
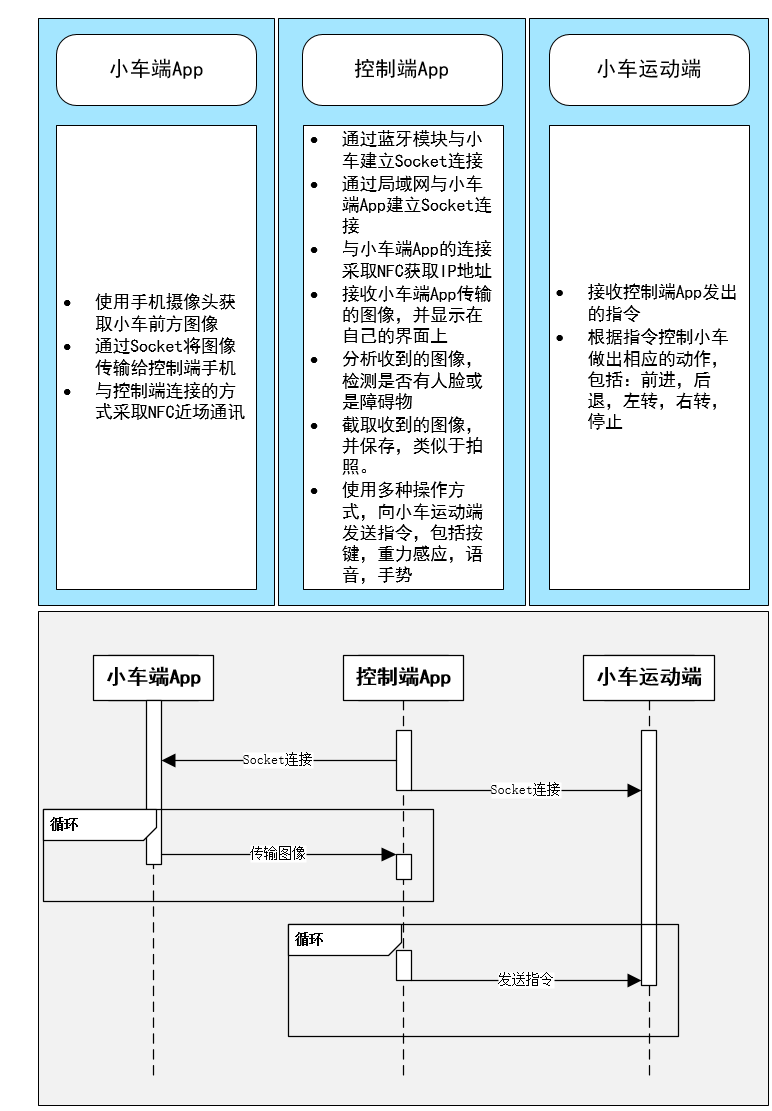


图6：小车整体布局

1. **整体架构及概述：**



### 单片机部分

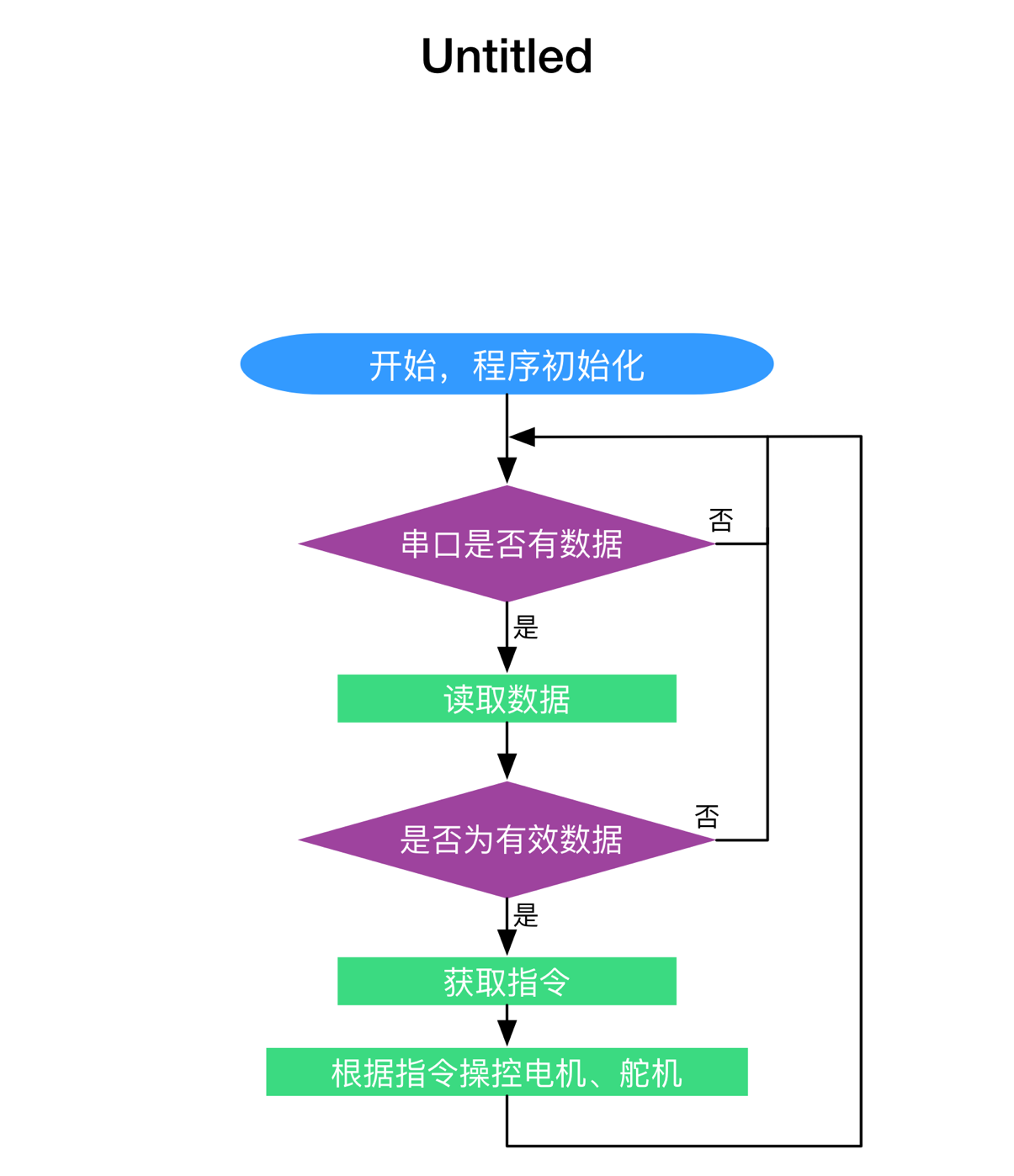


图6：小车程序流程图

* 控制逻辑设计

我们使用Arduino Mega 2560 v3进行单片机开发，主要思路是电脑通过蓝牙传送一个字符给单片机，单片机接收不同字符，进而控制电机与舵机状态。

主要有以下几种指令，括号内为对应的蓝牙需要接受的字符：

电机控制：

前进（W）：电机正转

后退（S）：电机反转

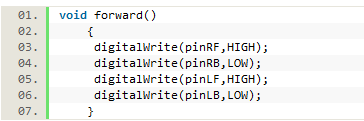
停止（Q）：电机停止

舵机控制：

左转（A）：舵机左转10°，最大可偏离正前方向-50°

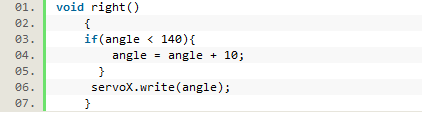
右转（D）：舵机右转10°，最大可偏离正前方向50°

对于电机的启动与方向，我们直接操纵单片机引脚输出的高低电平来进行的控制。以下是前进操作的代码。



对于舵机的控制，我们使用了一个控制舵机的库——“servo.h”。先给舵机实例化一个对象servoX，并用servoX.attach(int I/O)来绑定，通过函数servoX.write(int angle)来控制舵机的角度。根据Arduino开发手册，当angle==90时，舵机方向为正前方。基于我们之前的设计，有效的偏转角度区间为[40,140]

以下是右转操作的代码



* 初始化设置

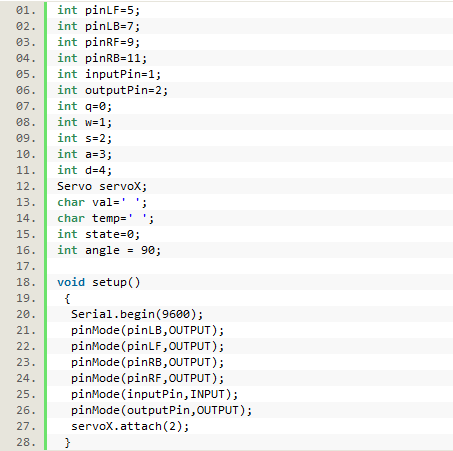
程序初始化的工作为

管脚定义：包括蓝牙、舵机、电机

舵机方向初始化

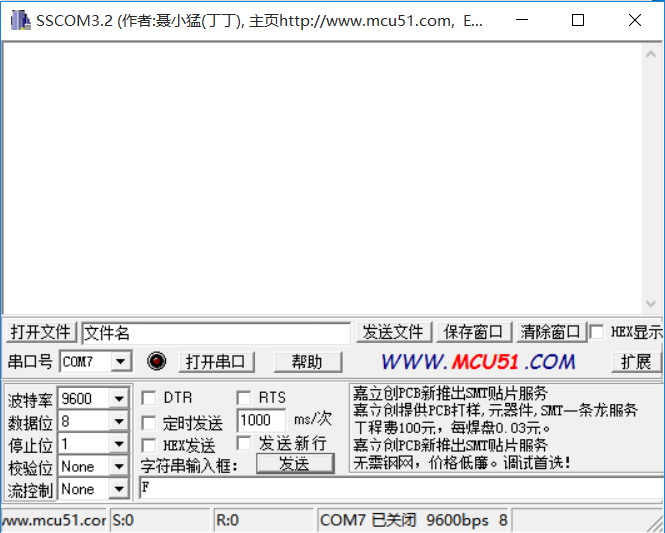
电机初始化

串口与指令字符初始化



* 程序调试

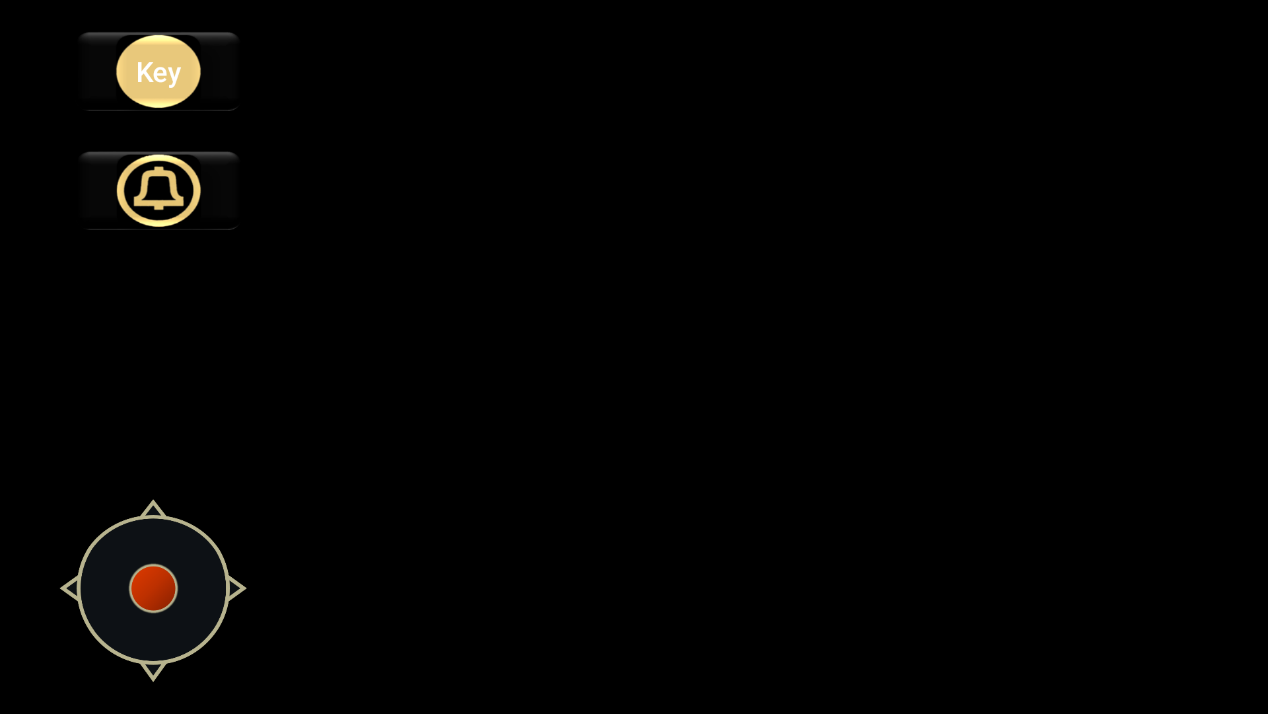
　　在跟主机端一起调试之前，我们使用SSCOM3.2串口调试工具来进行调试。在用蓝牙跟hc-06建立了连接以后，选择对应的串口，发送指定的字符串，看下车的运行情况。



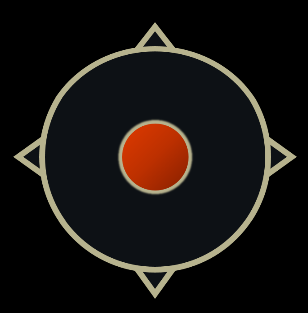
　　经测试，发送指令后小车能按指令运动，则测试成功。

1. **UI Design：**

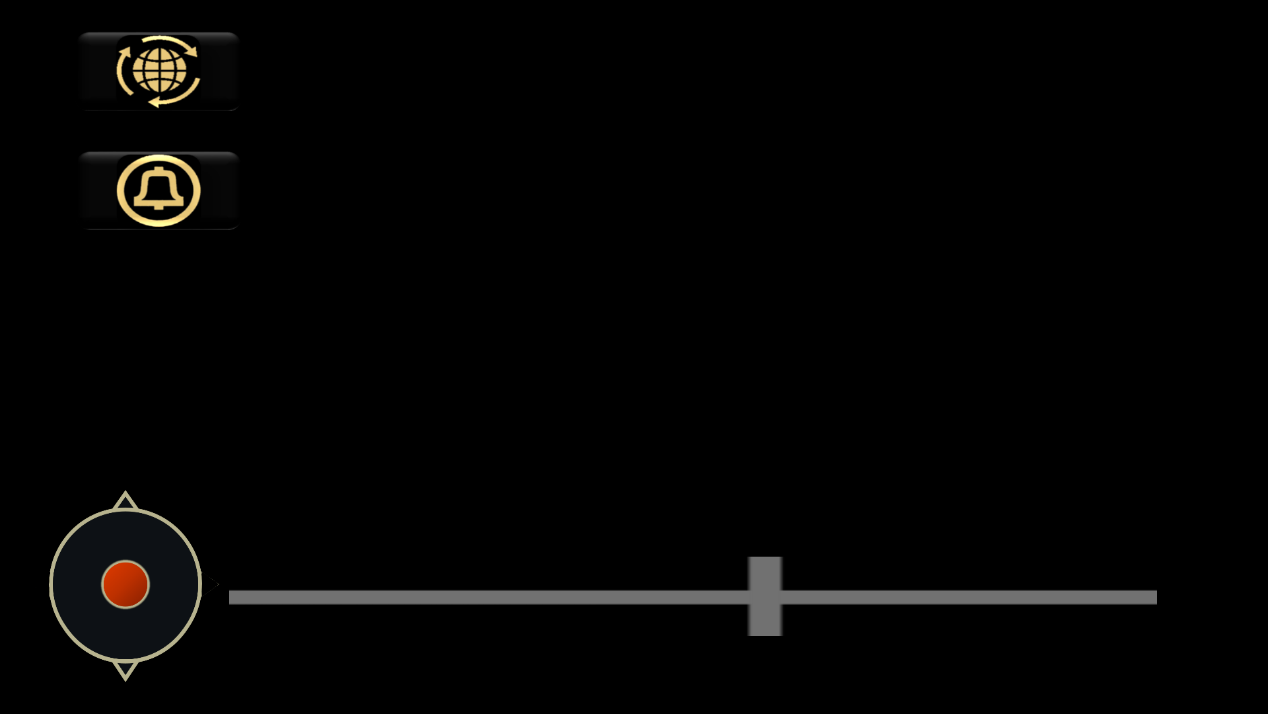
* UI以黑，金，红三个颜色为主色调，彰显强大的功能和稳定的性能。



* 方向键与拍照键五键合一，化零为整，方便使用，增强交互。

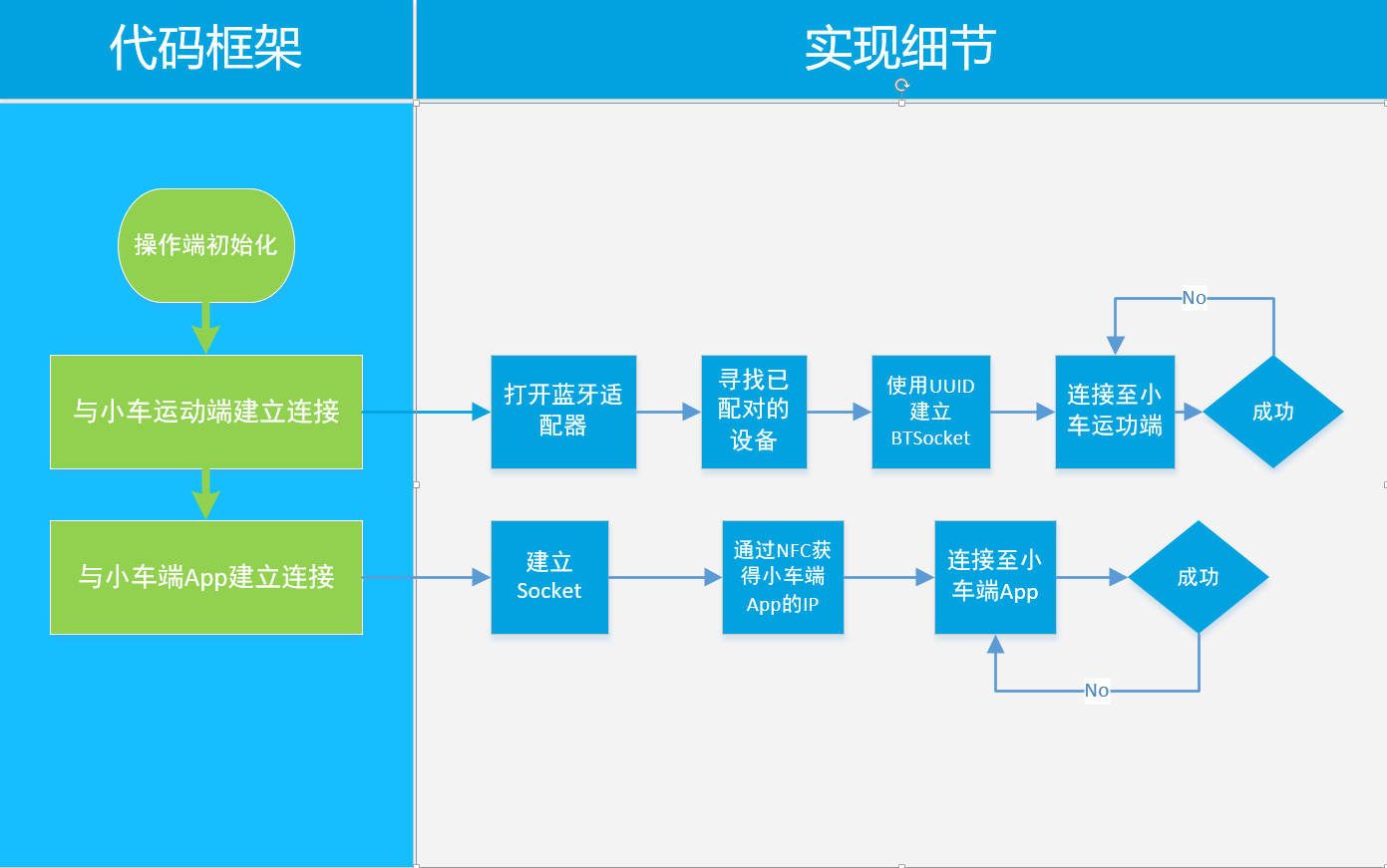


* 模式更换等辅助操作按钮与基本方向操作键分开放置，减少误操作。
* 语音识别，重力感应等可视化传感器数据，方便操作及矫正。





1. **操作端初始化：**



与小车运动端建立连接：

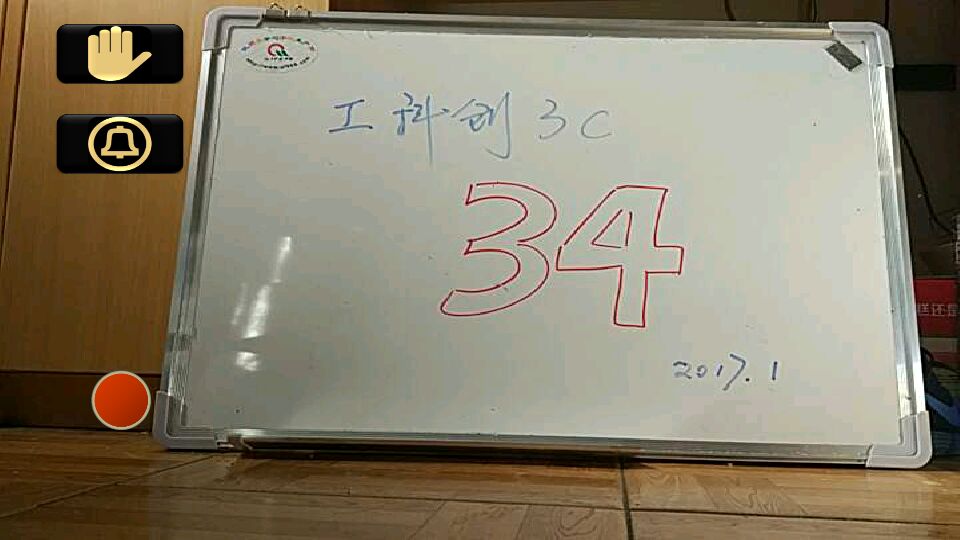
|  |
| --- |
| **new** Thread**(new** Runnable**(){**  public void run**()** **{**  **while(true)** **{**  //掉线重连  **try** **{**  **if(**sendData**(**"-"**))** **{**  Thread**.**sleep**(**1000**);**  **continue;**  **}**  //连接  mmSocket**.**connect**();**  //定义接收与发送流  mmOutputStream **=** mmSocket**.**getOutputStream**();**  mmInputStream **=** mmSocket**.**getInputStream**();**  pht**=new** BufferedReader**(new** InputStreamReader**(**mmInputStream**));**  **}** **catch** **(**Exception e**)** **{**  **}**  **}**  **}**  **}).**start**();** |

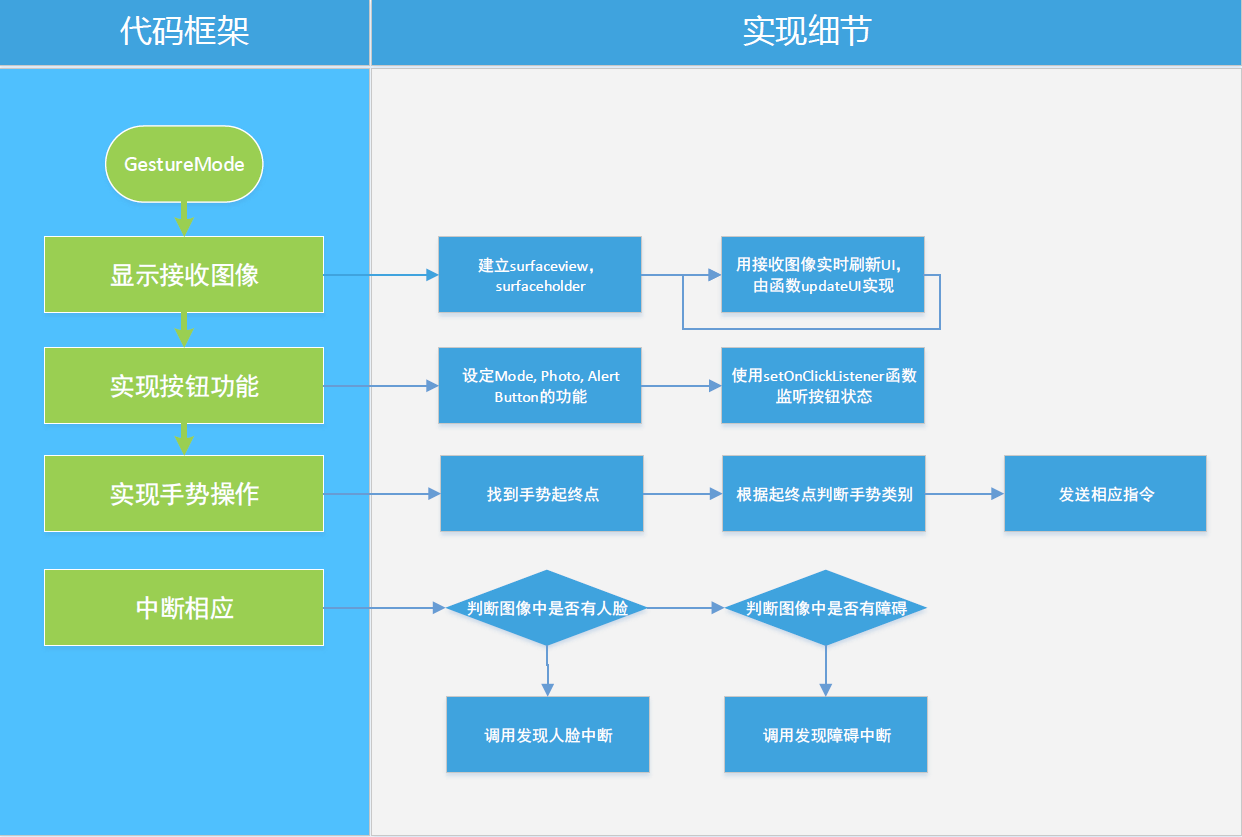
与小车端App建立连接：

|  |
| --- |
| server **=** **new** SocketServer**();**  server**.**start**();**  sp**=new** SoundPool**(**10**,** AudioManager**.**STREAM\_SYSTEM**,**5**);**  **new** Thread**(new** Runnable**(){**  public void run**()** **{**  **while(true)** **{**  **try** **{**  Thread**.**sleep**(**500**);**  sendData**(**"C"**);**  **if(**Integer**.**valueOf**(**pht**.**readLine**())<**20**)** **{**  server**.**add**(**0**);**  server**.**mDataListener**.**conv**(**1**);**  Thread**.**sleep**(**5000**);**  **}**  **}** **catch** **(**Exception e**)** **{**  **}**  **}**  **}**  **}).**start**();** |

NFC近场通讯配对连接的部分将在后边NFC的部分详细讲述，这里略过。

1. **Gesture Mode：**





Mode button: 变换至照片浏览界面

|  |
| --- |
| final Button button **=** **(**Button**)** findViewById**(**R**.**id**.**button**);**  //监听按钮状态  button**.**setOnClickListener**(new** View**.**OnClickListener**()** **{**  public void onClick**(**View v**)** **{**  //设定按钮功能  Intent intent **=** **new** Intent**();**  intent**.**setClass**(**GestureMode**.this,** PhotoMode**.**class**);**  startActivity**(**intent**);**  **}**  **});** |

Photo button: 按下按钮后拍照

|  |
| --- |
| final ImageButton ibutton5 **=** **(**ImageButton**)** findViewById**(**R**.**id**.**button5**);**  //监听按钮状态  ibutton5**.**setOnClickListener**(new** View**.**OnClickListener**()** **{**  //设定按钮功能  public void onClick**(**View v**)** **{ ((**MyApplication**)**getApplication**()).**phts**.**addFirst**(((**MyApplication**)**getApplication**()).**mLastFrame**);**  Toast**.**makeText**(**getApplicationContext**(),** "Photo Taken!"**,**  Toast**.**LENGTH\_SHORT**).**show**();**  **}**  **});** |

Alert button: 发出警报

|  |
| --- |
| final Button gray\_radio **=** **(** Button**)** findViewById**(**R**.**id**.**button2**);**  //监听按钮状态  gray\_radio**.**setOnClickListener**(new** View**.**OnClickListener**()** **{**  //设定按钮功能  public void onClick**(**View v**)** **{**  Toast**.**makeText**(**getApplicationContext**(),** "Cops on their way"**,**  Toast**.**LENGTH\_SHORT**).**show**();**  **((**MyApplication**)**getApplication**()).**server**.**add**(**1**);**  **}**  **});** |

Gesture manipulations: 响应手势操作

* 输入手势的起终点位置
* 计算手势位移
* 发送响应指令

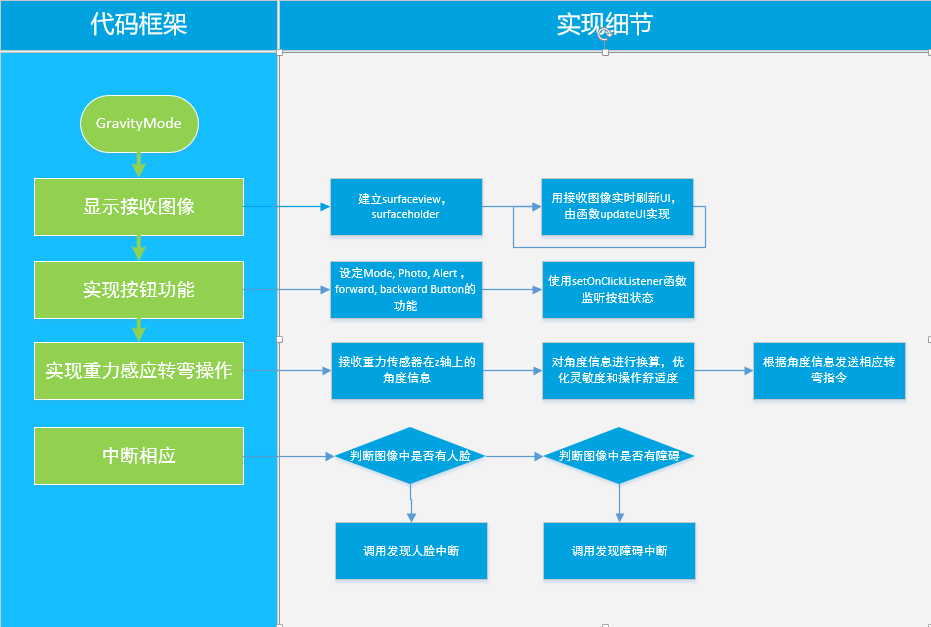
|  |
| --- |
| public boolean onFling**(**MotionEvent e1**,** MotionEvent e2**,** float velocityX**,** float velocityY**)** **{**  System**.**out**.**println**(**"zkw"**+(**e1**.**getX**()-**e2**.**getX**()));**  //左转  **if(**e1**.**getX**()-**e2**.**getX**()** **>** 200 **&&** Math**.**abs**(**velocityX**)** **>** 200**){**  sendData**(**"A"**);**  //右转  **}else** **if(**e2**.**getX**()** **-** e1**.**getX**()** **>** 200 **&&** Math**.**abs**(**velocityX**)** **>** 200**){**  sendData**(**"D"**);**  //前进  **}else** **if(**e1**.**getY**()-**e2**.**getY**()** **>** 200 **&&** Math**.**abs**(**velocityY**)** **>** 200**){**  sendData**(**"W"**);**  //后退  **}else** **if(**e2**.**getY**()-**e1**.**getY**()** **>** 200 **&&** Math**.**abs**(**velocityY**)** **>** 200**){**  sendData**(**"S"**);**  //停止  **}else{**  sendData**(**"Q"**);**  **}**  **return** **false;**  **}** |

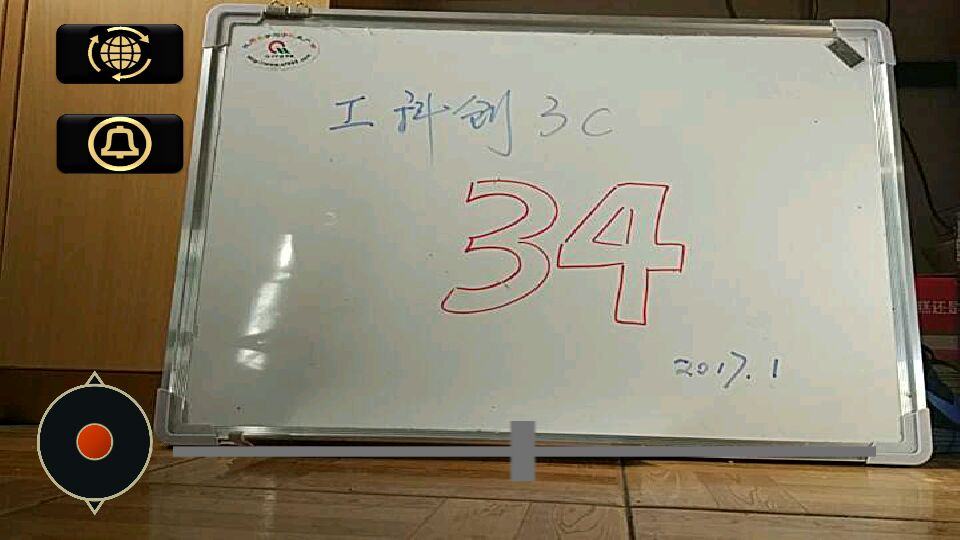
响应发现人脸，发现障碍的中断：

* 获取事件编号
* 调用相应中断相应程序

|  |
| --- |
| myHandler **=** **new** Handler**()** **{**  public void handleMessage**(**Message msg**)** **{**  int t**=**msg**.**what**;**  **if(**t**==**13**){**  //发现人脸的handler  Toast**.**makeText**(**getApplicationContext**(),** "Find a human."**,**  Toast**.**LENGTH\_SHORT**).**show**();**  **}**  **if(**t**==**1**){**  //发现障碍的handler  Toast**.**makeText**(**getApplicationContext**(),** "Find something."**,**  Toast**.**LENGTH\_SHORT**).**show**();**  **}**  **}**  **};** |

1. **Gravity Mode：**





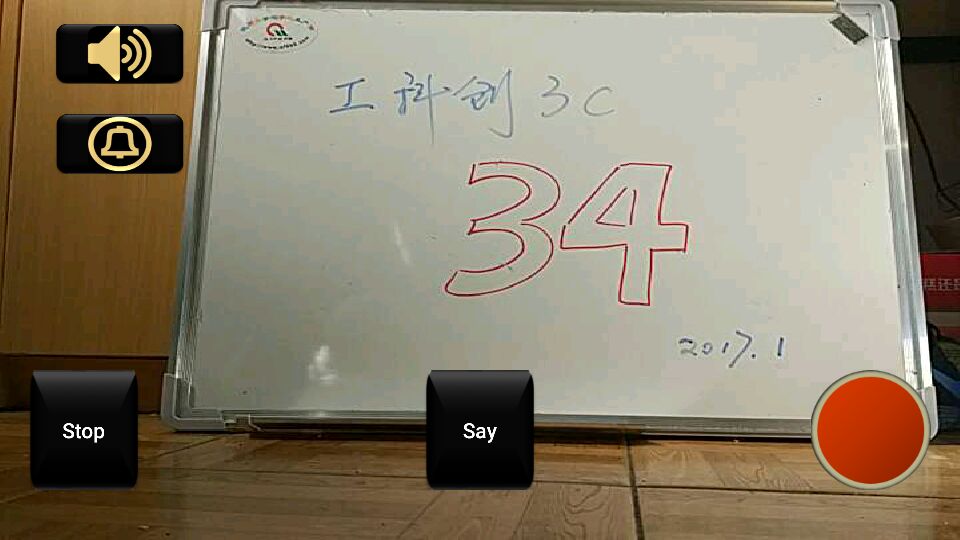
大部分的实现方式与Gesture 相同，这里不做赘述，仅介绍一下，重力传感器相关实现方式。

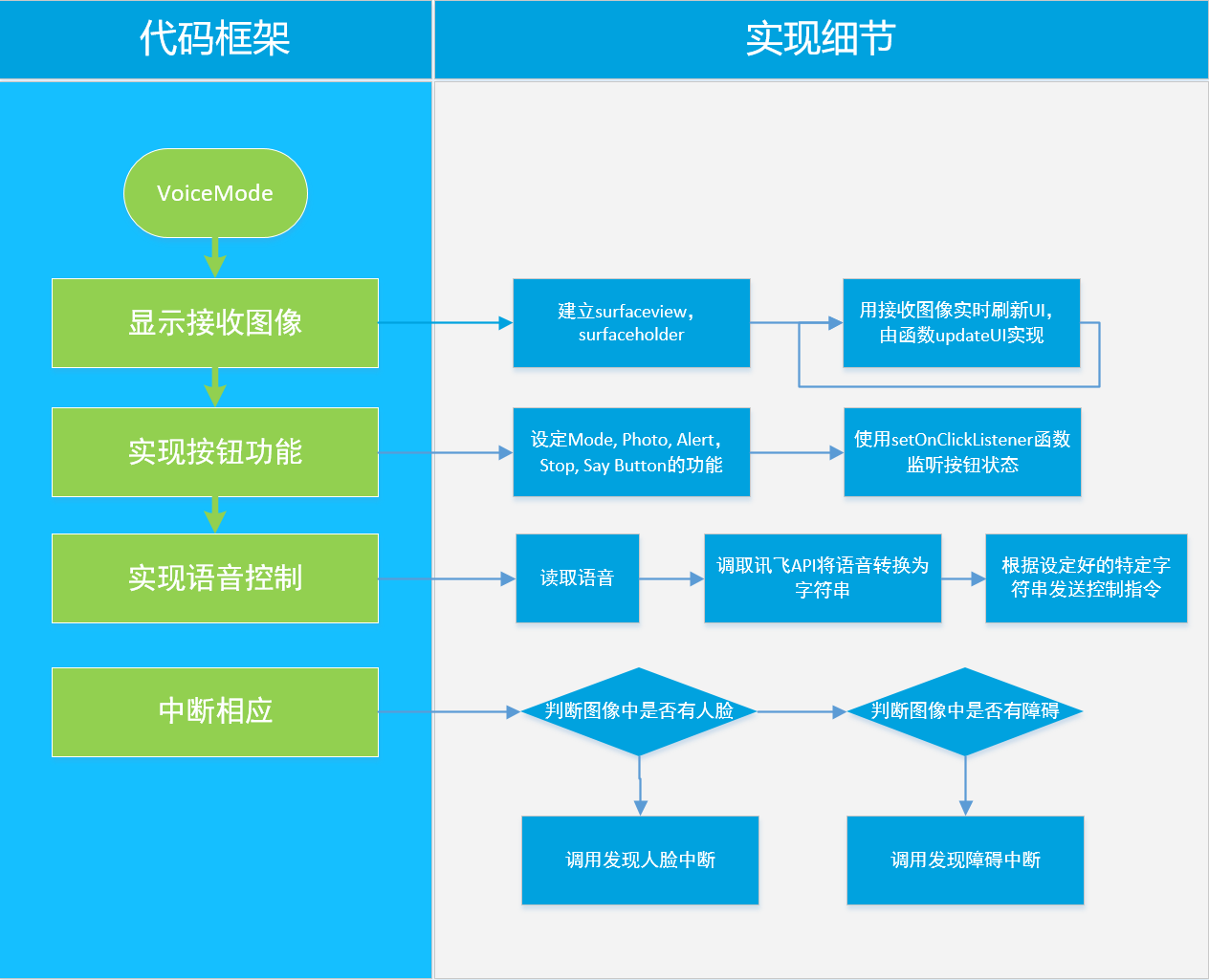
重力传感器实现转弯：

* 得到三个轴的角度信息
* 对y轴方向的角度信息进行一些数值处理，优化操作灵敏度
* 根据角度信息发送相应控制指令

|  |
| --- |
| public void onSensorChanged**(**SensorEvent event**)** **{**  **if(**on**==**0**)**  **return;**  **if** **(**event**.**sensor **==** **null)** **{**  **return;**  **}**  **if** **(**event**.**sensor**.**getType**()** **==** Sensor**.**TYPE\_ACCELEROMETER**)** **{**  //得到三个轴的角度信息  int x **=** **(**int**)** event**.**values**[**0**];**  int y **=** **(**int**)** event**.**values**[**1**];**  int z **=** **(**int**)** event**.**values**[**2**];**  **if(**y**>**5**)**y**=**5**;**  **if(**y**<-**5**)**y**=-**5**;**  MyImageView ps **=** **(**MyImageView**)** findViewById**(**R**.**id**.**imageView2**);**  //做一些换算  int sign**=**1**;**  **if(**y**<**0**)**sign**=-**1**;**  y**=**y**\***sign**;**  double t**=**y**;**  t**/=**5**;**  t**=**Math**.**pow**(**t**,**1.0**/**1**);**  t**=**t**\***sign**\***100**;**  ps**.**offset**=**t**;**  **if(**t**>=**40**){**  **if(**statu**!=**1**)**  sendData**(**"D"**);**  statu**=**1**;**  //发送相应的转弯信息  **}else** **if(**t**<=-**40**){**  **if(**statu**!=-**1**)**  sendData**(**"A"**);**  statu**=-**1**;**  **}else{**  **if(**statu**!=**0**)** **{**  sendData**(**"Q"**);**  statu **=** 0**;**  **}**  **}**  ps**.**invalidate**();**  **}**  **}** |

1. **Voice mode：**





从语音中分析字符串：

* 显示讯飞语音听写功能的UI
* 调用讯飞语音的API进行语音转换，获得说话人说的字符串

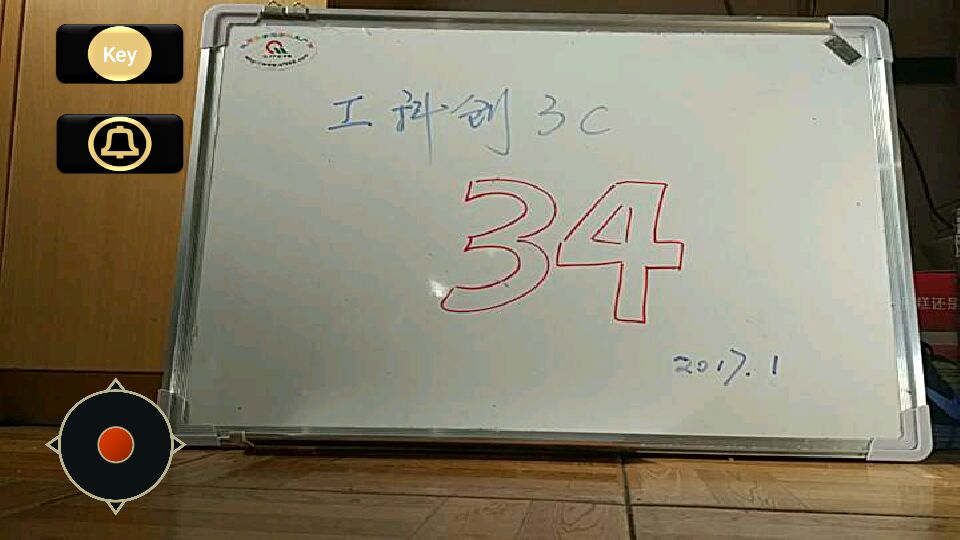
|  |
| --- |
| public void GetWordFromVoice**()**  **{**  boolean isShowDialog **=** mSharedPreferences**.**getBoolean**(**"iat\_show"**,true);**  **if** **(**isShowDialog**)** **{**  showIatDialog**();**  **}** **else** **{**  **if(null** **==** iatRecognizer**)** **{**  iatRecognizer**=**SpeechRecognizer**.**createRecognizer**(this);**  **}**  **if(**iatRecognizer**.**isListening**())** **{**  iatRecognizer**.**stopListening**();**  **}** **else** **{**  **}**  **}**  **}** |

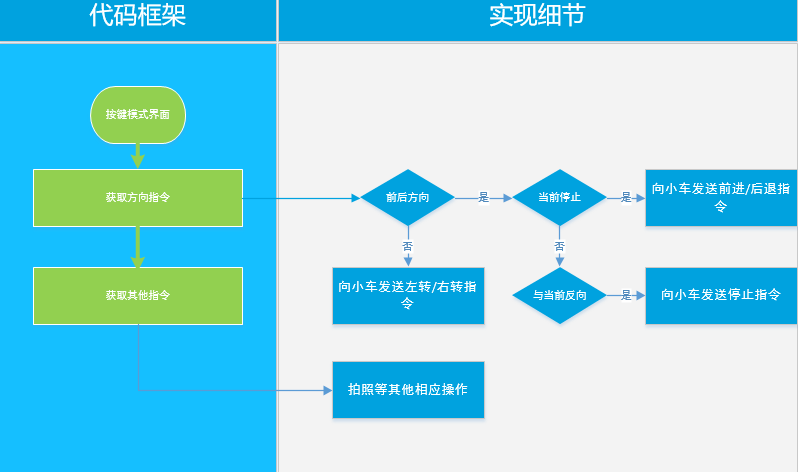
根据字符串产生指令：

* 输入得到的语音内容的字符串
* 根据字符串中包含的关键词发送指令

|  |
| --- |
| public void dos**(**String a**){**  **if(**a**.**contains**(**"左"**))**  sendData**(**"A"**);**  **else** **if(**a**.**contains**(**"右"**))**  sendData**(**"D"**);**  **else** **if(**a**.**contains**(**"前"**))**  sendData**(**"W"**);**  **else** **if(**a**.**contains**(**"后"**))**  sendData**(**"S"**);**  **}** |

**按键控制**

****



按键功能通过Android内部类ImageButtom 来获取UI界面的按钮是否被用户触发。这里以前进与后退操作为例

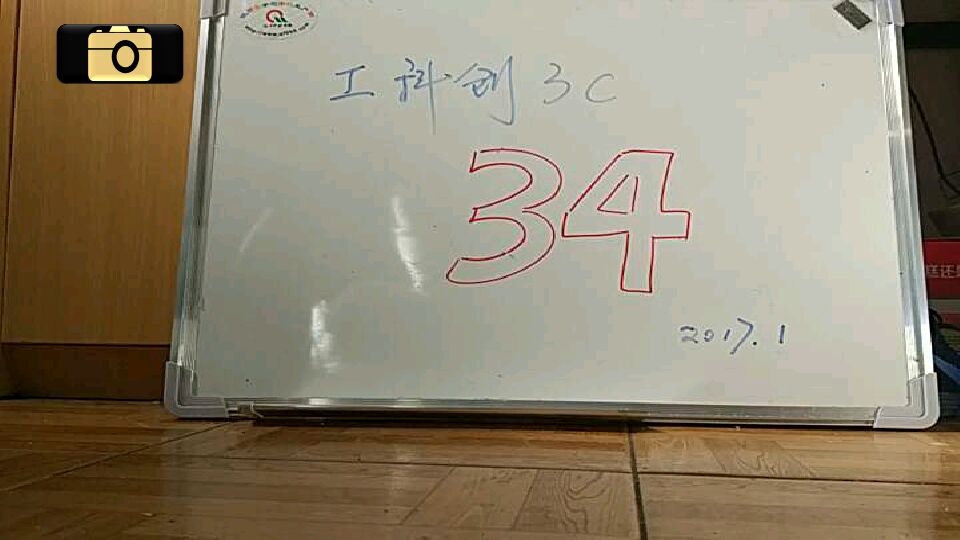
|  |
| --- |
| final ImageButton ibutton **=** **(**ImageButton**)** findViewById**(**R**.**id**.**button6**);**  ibutton**.**setOnTouchListener**(new** View**.**OnTouchListener**(){**  @Override  public boolean onTouch**(**View v**,** MotionEvent e**){**  **switch(**e**.**getAction**()){**  **case** MotionEvent**.**ACTION\_DOWN**:**  sendData**(**"W"**);**  **return** **true;**  **case** MotionEvent**.**ACTION\_UP**:**  sendData**(**"Q"**);**  **return** **true;**  **}**  **return** **false;**  **}**  **});**  final ImageButton ibutton2 **=** **(**ImageButton**)** findViewById**(**R**.**id**.**button3**);**  ibutton2**.**setOnTouchListener**(new** View**.**OnTouchListener**(){**  @Override  public boolean onTouch**(**View v**,** MotionEvent e**){**  **switch(**e**.**getAction**()){**  **case** MotionEvent**.**ACTION\_DOWN**:**  sendData**(**"S"**);**  **return** **true;**  **case** MotionEvent**.**ACTION\_UP**:**  sendData**(**"Q"**);**  **return** **true;**  **}**  **return** **false;**  **}**  **});** |

发送信息的sendData实现方法如下：

|  |
| --- |
| boolean sendData**(**String m**){**  **try{**  String msg **=** m**;**  **((**MyApplication**)**getApplication**()).**mmOutputStream**.**write**(**msg**.**getBytes**());**  System**.**out**.**println**(**"Data Sent!"**);**  **return** **true;**  **}catch** **(**Exception e**){**  **return** **false;**  **}**  **}** |

其中mmOutputStream是Android内部类OutputStream的一个实例化对象。

**拍照与查看**



照片的拍摄也是使用了Android内部类ImageButtom，按下UI界面上的对应按钮后，充当eye的手机传送来当前画面。我们使用了内部类Bitmap来作为图片的保存方式，并使用了LinkedList建立链表来作为Bitmap的容器使用，将这个图片链表实例化

|  |
| --- |
| public LinkedList**<**Bitmap**>** phts**;** |

之后每次拍摄进行视频流当前图像的发送并将成功信息返回至界面，代码如下

|  |
| --- |
| final ImageButton ibutton5 **=** **(**ImageButton**)** findViewById**(**R**.**id**.**button5**);**  ibutton5**.**setOnClickListener**(new** View**.**OnClickListener**()** **{**  public void onClick**(**View v**)** **{**  **((**MyApplication**)**getApplication**()).**phts**.**addFirst**(((**MyApplication**)**getApplication**()).**mLastFrame**);**  Toast**.**makeText**(**getApplicationContext**(),** "Photo Taken!"**,**  Toast**.**LENGTH\_SHORT**).**show**();**  **}**  **});** |

照片的查看实现比较简单，和手势操作类似，使用了MotionEvent类来获取用户在UI界面上的手势以切换当前照片。不同点在于判定条件，具体见以下代码：

|  |
| --- |
| @Override  public boolean onFling**(**MotionEvent e1**,** MotionEvent e2**,** float velocityX**,** float velocityY**)** **{**  System**.**out**.**println**(**"zkw"**+(**e1**.**getX**()-**e2**.**getX**()));**  **if(**e1**.**getX**()-**e2**.**getX**()** **>** 20 **&&** Math**.**abs**(**velocityX**)** **>** 10**){**  System**.**out**.**println**(**""**+(**e1**.**getX**()-**e2**.**getX**())+**","**+**velocityX**);**  **if(**it**!=null&&**it**.**hasPrevious**())**  paint**(**it**.**previous**());**  **}else** **if(**e2**.**getX**()** **-** e1**.**getX**()** **>** 20 **&&** Math**.**abs**(**velocityX**)** **>** 10**){**  **if(**it**!=null&&**it**.**hasNext**()){**  paint**(**it**.**next**());**  **}**  **}else** **if(**e1**.**getY**()-**e2**.**getY**()** **>** 20 **&&** Math**.**abs**(**velocityY**)** **>** 10**){**  **}else** **if(**e2**.**getY**()-**e1**.**getY**()** **>** 20 **&&** Math**.**abs**(**velocityY**)** **>** 10**){**  **}**  **return** **false;**  **}** |

这里的it是Bitmap的一个迭代器，用于查找前后的图片



**NFC接受IP地址**

使用NFC功能需要引入以下Android类，这里先介绍brain端手机NFC接受IP地址的实现，eye端获取IP地址与输出在后面的部分分析。



NfcAdapter用于获取匹配的NFC设备。配对成功后，使用NdefMessage类来获取eye端手机的信息，在本程序中获取的是IP地址；在此过程中我们会利用NdefRecord类，将获取的字符串与Package Name比较，以防止Android应用记录的传递。

|  |
| --- |
| private void handleNfcIntent**(**Intent NfcIntent**)** **{**  //用于NFC设备识别配对  **if** **(**NfcAdapter**.**ACTION\_NDEF\_DISCOVERED**.**equals**(**NfcIntent**.**getAction**()))** **{**  Parcelable**[]** receivedArray **=**  NfcIntent**.**getParcelableArrayExtra**(**NfcAdapter**.**EXTRA\_NDEF\_MESSAGES**);**  //配对成功  **if** **(**receivedArray **!=** **null)** **{**  messagesReceivedArray**.**clear**();**  NdefMessage ndefMessage **=** **(**NdefMessage**)** receivedArray**[**0**];**  NdefRecord**[]** attachedRecords **=** ndefMessage**.**getRecords**();**  **for** **(**NdefRecord record **:** attachedRecords**)** **{**  String string **=** **new** String**(**record**.**getPayload**());**  //确保我们不会传递我们的AAR（Android应用程序记录）  **if** **(**string**.**equals**(**getPackageName**()))** **{**  **continue;**  **}**  messagesReceivedArray**.**add**(**string**);**  **}**  String receivedMessage **=** messagesReceivedArray**.**get**(**0**);**  System**.**out**.**println**(**receivedMessage**);**  EditText iptext **=** **(**EditText**)** findViewById**(**R**.**id**.**ip\_edit\_text**);**  iptext**.**setText**(**receivedMessage**);**  **}** **else** **{**  Toast**.**makeText**(this,** "Received Blank Parcel"**,** Toast**.**LENGTH\_LONG**).**show**();**  **}**  **}**  **}** |

完成获取后，我们将得到的IP字符串转换为EditText类型，由此可以将IP字符串显示到UI的输入文本框里等待用户确认。

因为目前许多的Android 6.0设备还没有NFC硬件，由此我们也对IP地址的传递提供了备选方案：UI界面的文本框输入。采用内部类EditText可以叫容易地实现。我们通过下面的方法来输出IP地址：

|  |
| --- |
| public static String getAddr**(){**  **if(**ipConfirmed **==** **false){**  **return** **null;**  **}**  **if(**editTextIpAddress **==** **null){**  **return** **null;**  **}**  **return** editTextIpAddress**.**getText**().**toString**();**  **}** |

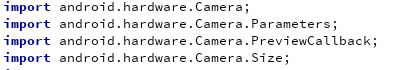
**EYE端实现**

Eye端主要实现两项功能：

1.IP地址的获取与传递

2.视频流传输

视频流传递实现主要是基于以下关于摄像机硬件的内部类：



此外，为了使多媒体流更加丰富，还使用了以下几个音频类



首先实例化出一个Camera对象



之后对Camera对象进行初始化，即对拍摄格式、拍摄尺寸、预览尺寸、FPS、聚焦模式等的设置，因为手机摄像机的设置在Android程序设计中十分常见，有较充足的范例，这里就不再细节分析了。以下为代码，

|  |
| --- |
| public CameraPreview**(**Context context**,** Camera camera**)** **{**  **super(**context**);**  mCamera **=** camera**;**  mHolder **=** getHolder**();**  mHolder**.**addCallback**(this);**  mHolder**.**setType**(**SurfaceHolder**.**SURFACE\_TYPE\_PUSH\_BUFFERS**);**  Parameters params **=** mCamera**.**getParameters**();**  params**.**setPreviewFormat**(**ImageFormat**.**NV21**);**  params**.**setFocusMode**(**Camera**.**Parameters**.**FOCUS\_MODE\_CONTINUOUS\_PICTURE**);**  params**.**setPictureSize**(**176**,**144**);**  params**.**setPreviewSize**(**720**,**480**);**  params**.**setPreviewFpsRange**(**15000**,** 15000**);**  mCamera**.**setParameters**(**params**);**  mPreviewSize **=** mCamera**.**getParameters**().**getPreviewSize**();**  int format **=** mCamera**.**getParameters**().**getPreviewFormat**();**  mFrameLength **=** mPreviewSize**.**width **\*** mPreviewSize**.**height **\*** ImageFormat**.**getBitsPerPixel**(**format**)** **/** 8**;**  sp**=new** SoundPool**(**10**,** AudioManager**.**STREAM\_SYSTEM**,**5**);**  sp**.**load**(**context**,**R**.**raw**.**aa**,**1**);**  sp2**=new** SoundPool**(**10**,** AudioManager**.**STREAM\_SYSTEM**,**5**);**  sp2**.**load**(**context**,**R**.**raw**.**eee**,**1**);**  sp3**=new** SoundPool**(**10**,** AudioManager**.**STREAM\_SYSTEM**,**5**);**  sp3**.**load**(**context**,**R**.**raw**.**asdasd**,**1**);**  **}** |

在eye中UI界面只需要展示小车前方图像即可，因此我们调用Camera.setPreviewDisplay，使用摄像机预览作为UI

|  |
| --- |
| public void surfaceCreated**(**SurfaceHolder holder**)** **{**  **try** **{**  mCamera**.**setPreviewDisplay**(**holder**);**  mCamera**.**startPreview**();**  **}** **catch** **(**IOException e**)** **{**  **}**  **}**  public void surfaceChanged**(**SurfaceHolder holder**,** int format**,** int w**,** int h**)** **{**  **if** **(**mHolder**.**getSurface**()** **==** **null){**  **return;**  **}**  **try** **{**  mCamera**.**stopPreview**();**  resetBuff**();**  **}** **catch** **(**Exception e**){**  **}**  **try** **{**  mCamera**.**setPreviewCallback**(**mPreviewCallback**);**  mCamera**.**setPreviewDisplay**(**mHolder**);**  mCamera**.**startPreview**();**  **}** **catch** **(**Exception e**){**  **}**  **}** |

当前获取的图像会存储在onPreviewFrame重载类之中，该函数里面有两个参数，第一个参数为byte[]，为摄像头当前图像数据，这里我们再次利用LinkedList作为容器来存储图像。这里的实例化对象mQueue可以看做视频流缓存

|  |
| --- |
| private LinkedList**<**byte**[]>** mQueue **=** **new** LinkedList**<**byte**[]>();**  @Override  public void onPreviewFrame**(**byte**[]** data**,** Camera camera**)** **{**  synchronized **(**mQueue**)** **{**  **if** **(**mQueue**.**size**()** **==** MAX\_BUFFER**)** **{**  mQueue**.**poll**();**  **}**  mQueue**.**add**(**data**);**  **}**  **}**  public byte**[]** getImageBuffer**()** **{**  synchronized **(**mQueue**)** **{**  **if** **(**mQueue**.**size**()** **>** 0**)** **{**  mLastFrame **=** mQueue**.**poll**();**  **}**  **}**  **return** mLastFrame**;**  **}**  private void resetBuff**()** **{**  synchronized **(**mQueue**)** **{**  mQueue**.**clear**();**  mLastFrame **=** **null;**  **}**  **}** |

同时我们还可以轻松地完成获取图像、清空目前视频图像缓存等操作

关于视频流的建立同样是基于socket，详情请参考实现架构部分

传输时我们使用YuvImage方法来将缓存中的数据转换为图像,再通过ByteArrayOutputStream来实现视频传输

|  |
| --- |
| YuvImage image **=** **new** YuvImage**(**mCameraPreview**.**getImageBuffer**(),** ImageFormat**.**NV21**,** mCameraPreview**.**getPreviewWidth**(),** mCameraPreview**.**getPreviewHeight**(),** **null);**  ByteArrayOutputStream myoutputstream **=** **new** ByteArrayOutputStream**();**  image**.**compressToJpeg**(new** Rect**(**0**,** 0**,** mCameraPreview**.**getPreviewWidth**(),** mCameraPreview**.**getPreviewHeight**()),** 60**,** myoutputstream**);**  myoutputstream**.**flush**();**  myoutputstream**.**close**();**  byte tmp**[]=**myoutputstream**.**toByteArray**();**  outputStream**.**write**(**intToBytes2**(**tmp**.**length**));**  outputStream**.**write**(**tmp**);**  System**.**out**.**println**(**"cxy: send"**+**tmp**.**length**+**":"**+**tmp**[**0**]+**tmp**[**500**]+**tmp**[**5000**]+**tmp**[**tmp**.**length**-**5000**]);**  outputStream**.**flush**();** |

1. **特色：**

* 多个终端连接简易方便，可扩展至所有安装有终端的设备之上：控制端App与小车端App采用NFC互联，快捷方便；控制端App与小车运动端采用蓝牙互联，普遍试用。
* 多模式操作，满足不同操作需求，包括语音，手势，按键及重力感应操作。
* 简约风格设计。在提高美观度的同时简化操作，提高人机交互效率。
* 各终端交互效率高。精简逻辑结构，使得小车运行，画面传输流畅，没有迟钝或是卡顿的现象。