

工科创 3C——31 小组

手机（平板）控制小车

1、项目概览

小组成员信息

组长：李珉超：515030910361

组员：崔家铭：515030910359

组员：王晨奕：515030910362

组员：强志文：515030910367

分工情况

组长：李珉超：软件端的实践与调试，以及小车造型设计

组员：崔家铭：硬件部分的连接与调试，以及最后的网页的制作

组员：王晨奕：软件端的页面设计、实现以及整个软件的测试

组员：强志文：硬件部分的连接与调试，以及报告的撰写

工作照片

李珉超



王晨奕



崔家铭



强志文



作品静态清晰照片



2、项目介绍

整体完成情况

我们小组实现了通过安卓手机对小车发出指令，并且指令能够正确传输与识别，同时小车能很好地执行不同的指令。除此之外，我们小组还实现了包括重力感应，语音识别，视频互传，手势识别在内的拓展功能。

整体结构说明

手机 A (手持端) 与放在小车上的手机 B (小车端)建立连接，小车与小车蓝牙模块建立连接. 手机向小车器发送指令，小车将受到的指令传送给蓝牙，蓝牙通过串口通信将对应的指令传给单片机，单片机对接收到的指令进行解析，并控制小车做出相应的动作。

两部手机通过 wifi 建立连接，并连接小车的蓝牙。 小车器手机的 Android 软件监听不同类

别的信息，如语音和重力感应等，并将其转化为控制小车的指令。控制端程序将指令信号发送给小车上的蓝牙。单片机接收来自蓝牙串口的指令信号，根据指令调整舵机的高低电平及占空比，进而控制车轮的方向和速度，实现前进、后退、停止、转弯等动作。

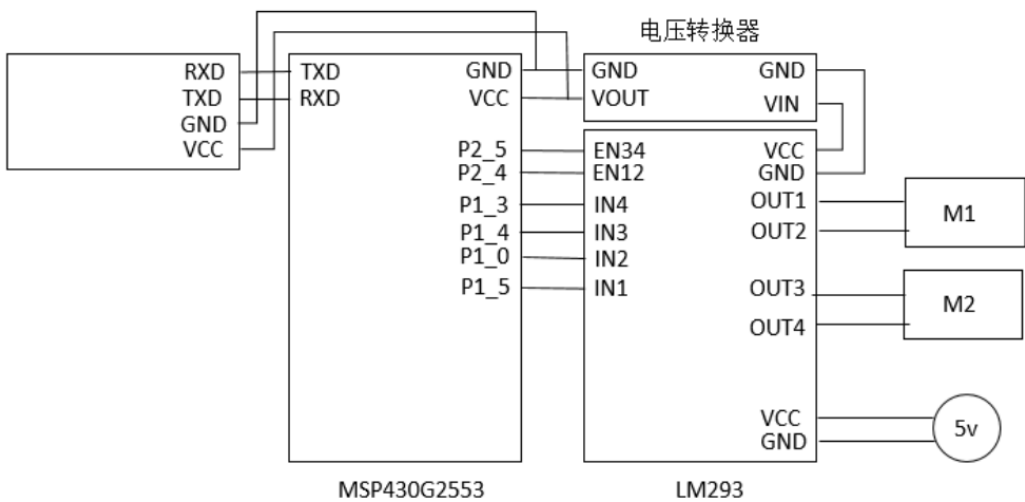
3、硬件部分介绍

小车型装

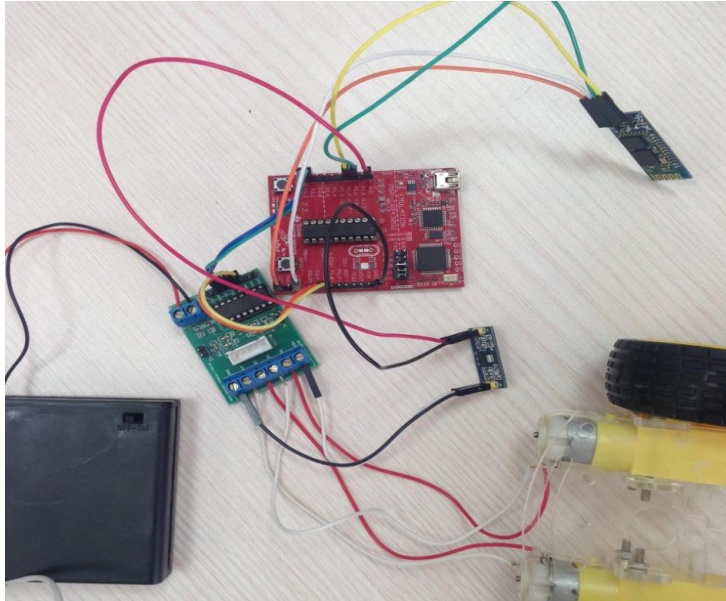
下图为电机驱动模块，L9110 模块可控制两组直流电机 1，2，电机接在右侧接口控制端-连单片机板：GND 与单片机的 GND 相连 VCCmotor 切记连电池输出，而不要连到单片机的 VCC. IA1-2, IB1-2 连单片机 IO 脚 Px.x，IA1，IB1 控制一组电机，IA2，IB2 控制另一组电机如需控制速度，可将同组中一个控制脚接入 PWM 输出脚（P2.1，P2.2，P2.4，P2.5，P1.6，P2.6）。



下图为硬件部分连接示意图，系统采用 1.2V*4 可充电镍氢电池提供 4.8V 直流电压。



下图是小车的连接示意图。



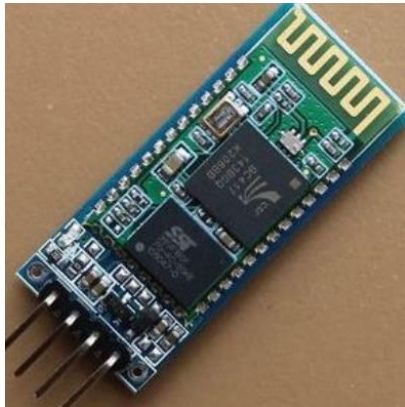
我们将小车的万向轮固定，这样可以达到更好的转弯效果。

蓝牙通信

TXD: 发送端，一般表示为自己的发送端，正常通信必须接另一个设备的 **RXD**。

RXD: 接收端，一般表示为自己的接收端，正常通信必须接另一个设备的 **TXD**。

在本实验中，HC-06 蓝牙模块接收来自安卓端的指令，传递给 MSP430 launchpad，从而实现手机对智能车的无线遥控。



蓝牙供电后：手机上可搜到 HC-06 ， 05 之类的蓝牙设备，配对（密码为 1234 ）后可以通过 Appinventor 的蓝牙模块控制单片机，单片机采用串口接收数据或发送数据。

硬件编程

这部分同工科创 2B 大致类似，我们使用的是 Energia 进行编程，因为 Energia 程序更接近 C 语言，容易调试。需要注意的是在烧程序时要插好相应接口的跳线帽，否则程序可能会导入失败。同时在左转右转是要测试不同的参数，达到较好的转弯效果。




LaunchPad with MSP430G2553

Revision 1.5

Flash 16 KB
Serial Hardware

Pin	Label	Function
1	+3.3V	
2	RED_LED	A0 P1.0
3		RXD A1 P1.1
4		TXD A2 P1.2
5	PUSH2	A3 P1.3
6		A4 P1.4
7		SCK (B0) A5 P1.5
8		CS (B0) P2.0
9		P2.1
10		P2.2

Rei Vilo, 2012-2013
embeddedcomputing.weebly.com
version 1.3 2102-09-09



Hardware
Pin number

PC
Serial UART
SPI

analogRead()
digitalRead() and digitalWrite()
digitalRead(), digitalWrite()
and analogWrite()

Pin	Label	Function
20		GROUND
19	P2.6	XIN
18	P2.7	XOUT
17		TEST
16		RESET
15	P1.7	A7 SDA MOSI (B0)
14	P1.6	A6 SCL MISO (B0) GREEN_LED
13	P2.5	
12	P2.4	
11	P2.3	

4、软件部分介绍

我们的开发平台为 windows 下的 android studio。我们开发了两款 app，用户使用其中的一个 app"AE86"来操纵手机，同时"AE86"的界面上会出现由另一个 app"driver"回传的实时画面。在操作时 app"driver"被固定在小车上。app"AE86"通过蓝牙和小车上的蓝牙模块进行通信，关于小车的蓝牙模块的介绍，请参考硬件部分。

"AE86":

用户可以通过"AE86"用四种方式通过蓝牙控制小车，分别为按键控制，语音控制，重力控制以及手势控制。在"AE86"这个 app 上，我们设计的布局为，通过四个按键来切换用户的 4 种控制方式，每一种方式都有自己的布局。另外，我们还在布局上显示了由"driver"传来的实时画面，所以一共有 5 个布局。

【ae86_initialize】

然后通过重写 activity 的 onCreate 函数，进行"AE86"的初始化。初始化内容包括对应于 4 种控制方式的 4 个按键的加载，蓝牙的启动以及做好接收回传视频的准备工作。


```

//布局切换按键
private Button keyControl_button;
private Button voiceControl_button;
private Button gravityControl_button;
private Button gestureControl_button;

//5 个可选布局
private KeyControlFragment keyControlFragment;
private VoiceControlFragment voiceControlFragment;
private GravityControlFragment gravityControlFragment;
private GestureControlFragment gestureControlFragment;
private CameraFragment cameraFragment;

//蓝牙设备与串口
private BluetoothDevice mDevice = null;
private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"); |

```

【ae86_onCreate】

我们通过定义函数 `initViews` 来加载按键，下图为按键控制布局加载的代码展示，另外 3 种方法的实现效果类似。首先读取并加载 xml 定义的布局，然后再绑定按键事件。即，实例化一个 `keyControlFragment` 类，然后用 `FragmentManager` 类替换掉 `content` 的 `layout`。

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // 设置三个按钮的布局
    initViews();
    // 开启蓝牙
    startBluetooth();
    // 开启视频
    initCameraPreview();
}

```

【ae86_initViews】

我们将蓝牙的连接封装在了函数 `startBluetooth` 中，具体实现方法参考了 android 官方的 api 文档，url 地址为 <http://android.xsoftlab.net/guide/topics/connectivity/bluetooth.html> 此外我们将蓝牙的传送借口进行封装成类 `Sender`，用于蓝牙信息的传输。

```

// 设置4个按钮的布局
private void initViews() {
    // 点击按钮，展开布局
    // 方向键方式
    keyControl_button = (Button) findViewById(R.id.keyControl_button);
    keyControl_button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // 用KeyControlFragment类来布置一个layout
            // 再用以下方法把content的FrameLayout替换为keyControlFragment布置的layout
            FragmentManager fragmentManager = getFragmentManager();
            FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

            keyControlFragment = new KeyControlFragment();
            fragmentTransaction.replace(R.id.content, keyControlFragment);
            fragmentTransaction.commit();
        }
    });
}

```

【ae86_startBluetooth】

最后，我们用 `initViews` 中类似的方法，加载了视频传输的布局。

```
// 开启蓝牙
public void startBluetooth() {
    // 获取蓝牙适配器
    BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if (mBluetoothAdapter == null) {
        Toast.makeText(this, "不支持蓝牙", Toast.LENGTH_LONG).show();
        DisplayText("不支持蓝牙");
        return;
    }
    if (!mBluetoothAdapter.isEnabled()) {
        Intent mIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(mIntent, 1);
        DisplayText("蓝牙未打开，请重试");
        return;
    }
    mBluetoothAdapter.startDiscovery();
    // 获取配对设备
    Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
    for (BluetoothDevice d : pairedDevices) {
        if (d.getName().equals("HC-06")) {
            mDevice = d;
            DisplayToast("已和蓝牙模块配对！");
            break;
        }
    }
    try {
        Sender.btSocket = mDevice.createRfcommSocketToServiceRecord(MY_UUID);
    } catch (IOException e) {
        DisplayToast("套接字创建失败！");
    }
    mBluetoothAdapter.cancelDiscovery();
    try {
        Sender.btSocket.connect();
        DisplayToast("连接成功建立，数据连接打开！");
        DisplayText("蓝牙连接状态良好");
    } catch (IOException e) {
        try {
            DisplayToast("连接没有建立");
            DisplayText("蓝牙未连接，请重试");
            Sender.btSocket.close();
        } catch (IOException e2) {
            DisplayToast("连接没有建立，无法关闭套接字！");
        }
    }
}
```

【ae86_initCamera】

最后定义四种方式的蓝牙接口，至此 `mainActivity` 的任务完成。

```
private void initCameraPreview(){
    FragmentManager fragmentManager = getFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

    //if(cameraFragment!=null) cameraFragment.stop();
    cameraFragment = new CameraFragment();
    fragmentTransaction.replace(R.id.camera,cameraFragment);
    fragmentTransaction.commit();
    DisplayToast("布局完成");
}
```

【ae86_port】

在 `keyControlFragment` 中，通过定义 5 个方向键，并定义按键事件为发送相应的字符给单片机，实现小车的前进，停止，后退，左转，右转。

```

// 方向控制接口
public void onButtonClicked_control(String message){
    bluetoothSender(message);
}
// 重力控制接口
public void on_gravity(String message) {
    bluetoothSender(message);
}
// 语音控制接口
public void on_voice(String message) {
    bluetoothSender(message);
}
// 手势控制接口
public void on_gesture(String message){
    bluetoothSender(message);
}

// 蓝牙发送器接口
public void bluetoothSender(String command){
    try {
        byte[] buffer = command.getBytes();
        Sender.getOutputStream().write(buffer);
    } catch (Exception e) {
    }
}
}

```

【key】

最后定义接口类 OnButtonClickedListener_Control，并重写 onAttach，通过实例化 activity 为 OnButtonClickedListener_Control 对象，完成与 mainActivity 的关联。

```

public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View keyControlLayout = inflater.inflate(R.layout.keycontrolfragment, container, false);

    forward=(Button) keyControlLayout.findViewById(R.id.forward);
    forward.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v) {
            msg = "f";
            onButtonClickedListener_control.onButtonClicked_control(msg);
        }
    });
    back=(Button) keyControlLayout.findViewById(R.id.back);
    back.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v) {
            msg = "b";
            onButtonClickedListener_control.onButtonClicked_control(msg);
        }
    });
    stop=(Button) keyControlLayout.findViewById(R.id.stop);
    stop.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v) {
            msg = "s";
            onButtonClickedListener_control.onButtonClicked_control(msg);
        }
    });
    left=(Button) keyControlLayout.findViewById(R.id.left);
    left.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v) {
            msg = "l";
            onButtonClickedListener_control.onButtonClicked_control(msg);
        }
    });
    right=(Button) keyControlLayout.findViewById(R.id.right);
    right.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v) {
            msg = "r";
            onButtonClickedListener_control.onButtonClicked_control(msg);
        }
    });

    return keyControlLayout;
}

```


【key_interface】

在语音控制的部分，我们使用了讯飞的 android API 包，并且参考了讯飞的使用文档。参考 url:<http://blog.csdn.net/zhoudumushui/article/details/45293827>

在 VoiceControlFragment 中首先用类似的方法完成布局和讯飞服务的初始化，定义按键为语音控制的参数初始化，包括引擎和码率。

```
// 发送器接口
public interface OnButtonClickedListener_Control {
    public void onButtonClicked_control(String ButtonText);
}

private OnButtonClickedListener_Control onButtonClickedListener_control;

@Override
public void onAttach(Activity activity){
    super.onAttach(activity);

    try {
        onButtonClickedListener_control = (OnButtonClickedListener_Control) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString() + "must implement OnButtonClickedListener");
    }
}
```

【voice_init】

然后再定义讯飞的监听器，重写其中的函数，简单地通过匹配识别出来的字符串，来发送相应的字符到蓝牙。

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View voiceLayout = inflater.inflate(R.layout.voicefragment,
        container, false);

    final Button voice_button_start = (Button) voiceLayout.findViewById(R.id.record_button);

    SpeechUtility.createUtility(getActivity(), "appid=5a13d557");
    mToast=Toast.makeText(getActivity(),"",Toast.LENGTH_LONG);
    txt_voice = (TextView)voiceLayout.findViewById(R.id.txt_voice);
    mIat= com.iflytek.cloud.SpeechRecognizer.createRecognizer(getActivity(), mInitListener);

    voice_button_start.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mIat.setParameter(SpeechConstant.DOMAIN,engine);
            mIat.setParameter(SpeechConstant.SAMPLE_RATE, rate);
            mIat.startListening(mRecognizerListener);
        }
    });

    return voiceLayout;
}
```

【voice_listen】

最后类似地，通过定义接口类，完成与 MainActivity 的关联。

```

private com.iflytek.cloud.RecognizerListener mRecognizerListener = new com.iflytek.cloud.RecognizerListener() {
    @Override
    public void onBeginOfSpeech() {
        showTip("开始说话");
    }

    @Override
    public void onEndOfSpeech() {
        showTip("结束说话");
    }

    @Override
    public void onResult(com.iflytek.cloud.RecognizerResult recognizerResult, boolean b) {
        showTip("???");
        String text=JsonParser.parseIatResult(recognizerResult.getResultString());
        txt_voice.setText(text);
        showTip(text);
        String stringA = "前进";
        String stringB = "后退";
        String stringC = "停止";
        String stringD = "左转";
        String stringE = "右转";
        if (text.equals(stringA)){
            msg = "f";
            onlistener_voice.on_voice(msg);
        }
        if (text.equals(stringB)){
            msg = "b";
            onlistener_voice.on_voice(msg);
        }
        if (text.equals(stringC)){
            msg = "s";
            onlistener_voice.on_voice(msg);
        }
        if (text.equals(stringD)){
            msg = "l";
            onlistener_voice.on_voice(msg);
        }
        if (text.equals(stringE)){
            msg = "r";
            onlistener_voice.on_voice(msg);
        }
    }
}

```

【voice_interface】

在 GravityControlFragment 中，首先完成布局的初始化，然后获得加速度传感器的权限，并定义 listener。通过获取手机的倾斜角度，通过 5 个条件分类，在 listener 中发送相应字符到蓝牙。

```

public interface Onlistener_Voice {
    public void on_voice(String ButtonText);
}

private Onlistener_Voice onlistener_voice;

private void showTip(String str){
    if (!TextUtils.isEmpty(str)){
        mToast.setText(str);
        mToast.show();
    }
}

@Override
public void onAttach(Activity activity){
    super.onAttach(activity);
    try {
        onlistener_voice = (Onlistener_Voice) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString() + "must implement OnButtonClickedListener");
    }
}

```

【grav_init】

最后定义接口类，并注意在 `fragment` 的生命周期结束时，释放我们对加速传感器的申请。

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                          Bundle savedInstanceState) {
    View gravityLayout = inflater.inflate(R.layout.gravityfragment,
        container, false);

    txt_gravity = (TextView) gravityLayout.findViewById(R.id.txt_gravity_y);
    sensorManager = (SensorManager) getActivity().getSystemService(Context.SENSOR_SERVICE);
    gravitySensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    gravityListener = new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            x = (int) event.values[SensorManager.DATA_X];
            y = (int) event.values[SensorManager.DATA_Y];

            if (x < 5 && x > -5 && y > -5 && y < 5) {
                txt_gravity.setText("停止");
                msg = "s";
                onListener_gravity.on_gravity(msg);
            }
            if (x < -5 && y > -5 && y < 5) {
                txt_gravity.setText("右转");
                msg = "r";
                onListener_gravity.on_gravity(msg);
            }
            if (x > 5 && y > -5 && y < 5) {
                txt_gravity.setText("左转");
                msg = "l";
                onListener_gravity.on_gravity(msg);
            }
            if (x < 5 && x > -5 && y < -5) {
                txt_gravity.setText("前进");
                msg = "f";
                onListener_gravity.on_gravity(msg);
            }
            if (x < 5 && x > -5 && y > 5) {
                txt_gravity.setText("后退");
                msg = "b";
            }
        }
    };
}
```

【grav_interface】

在 `GestureControlFragment` 中，首先完成布局的初始化，然后在 `listener` 中，通过两个 `MotionEvent` 对应我们触碰到屏幕和从屏幕上移开两个事件。我们记录接触屏幕和离开屏幕的两次点的相对位置，定义 5 个条件对应 5 个小车的行为。

```
public interface OnListener_Gravity {
    public void on_gravity(String ButtonText);
}

private OnListener_Gravity onListener_gravity;

@Override
public void onAttach(Activity activity){
    super.onAttach(activity);

    try {
        onListener_gravity = (OnListener_Gravity) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString() + "must implement OnButtonClickedListener");
    }
}

@Override
public void onPause(){
    super.onPause();
    sensorManager.unregisterListener(gravityListener);
}
```

【gesture_init】

接口定义

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState){
    View view = inflater.inflate(R.layout.gesturefragment, container, false);
    txt_gesture_show = (TextView)view.findViewById(R.id.txt_gesture_show);
    txt_gesture_show.setOnTouchListener(new View.OnTouchListener() {

        @SuppressWarnings("ClickableViewAccessibility")
        @Override
        public boolean onTouch(View v, MotionEvent e) {
            switch(e.getAction()){
                case MotionEvent.ACTION_DOWN:
                    x1 = (int)e.getX();
                    y1 = (int)e.getY();
                    break;
                case MotionEvent.ACTION_UP:
                    x2 = (int)e.getX();
                    y2 = (int)e.getY();
                    if(y1 - y2 > 100) {
                        txt_gesture_show.setText("前进");
                        msg = "f";
                        onListener_gesture.on_gesture(msg);
                    } else if(y2 - y1 > 100) {
                        txt_gesture_show.setText("后退");
                        msg = "b";
                        onListener_gesture.on_gesture(msg);
                    } else if(x1 - x2 > 100) {
                        txt_gesture_show.setText("左转");
                        msg = "l";
                        onListener_gesture.on_gesture(msg);
                    } else if(x2 - x1 > 100) {
                        txt_gesture_show.setText("右转");
                        msg = "r";
                        onListener_gesture.on_gesture(msg);
                    } else if(((x1-x2)<10||x2-x1<10) && ((y1-y2)<10||y2-y1<10)){
                        txt_gesture_show.setText("停止");
                        msg = "s";
                        onListener_gesture.on_gesture(msg);
                    }
                    break;
            }
            return true;
        }
    });
    return view;
}
```

【gesture_interface】

CameraFragment 中，先初始化了布局，然后新建了句柄 handler 用来显示图片，并在最后新建了实例化图片的线程类 receiveThread 类为 receivethread。

```
public interface OnListener_Gesture {
    public void on_gesture(String ButtonText);
}

private OnListener_Gesture onListener_gesture;

@Override
public void onAttach(Activity activity){
    super.onAttach(activity);

    try {
        onListener_gesture = (OnListener_Gesture) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString() + "must implement OnButtonClickedListener");
    }
}
```

【camera_init】

receiveThread 类中，定义并使用 8888 作为端口号。然后用 while(true)通过 InputStream 实例，不断从 buffer 中读取当前 buffer 中的全部内容，然后放到 data 中。然后新建一个 setImageThread 类的实例来处理 data 中存放的图像信息。

【camera_receive】

setImageThread 类中，将 data 中的数据还原成图像。

```
class receiveThread extends Thread{
    private int length = 0;
    private int num = 0;
    private byte[] buffer = new byte[2048];
    private byte[] data = new byte[204800];
    @Override
    public void run(){
        //showTip("receiveThread initialize...");
        try{
            if(serverSocket==null)serverSocket = new ServerSocket(PORT);
            //不断接受从PORT中发来的数据
            while(true){
                Socket socket = serverSocket.accept();
                try{
                    InputStream input = socket.getInputStream();
                    num = 0;
                    do{
                        length = input.read(buffer);
                        if(length >= 0){
                            //将buffer中从0开始length长度的数据读入以num位起始的数据中
                            System.arraycopy(buffer,0,data,num,length);
                            num += length;//下一次开始的位置
                        }
                    }while(length >= 0);
                    new setImageThread(data,num).start();
                    input.close();
                }
                catch(Exception e){
                    e.printStackTrace();
                }finally {
                    socket.close();
                }
            }
        }catch(IOException e){
            e.printStackTrace();
        }finally {
            try{
                serverSocket.close();
            }catch (Exception e){}
        }
    }
}
```

【camera_setImage】

"driver":

将应用设置为常亮，无标题和全屏，这样就不会因为应用长时间没有用户交互而被系统后台 kill 掉。另外定义了两个按键，用来维护一个 boolean 变量 flag，用来控制视频传输与否。

```
class setImageThread extends Thread{
    private byte[]data;
    private int num;
    public setImageThread(byte[] data, int num){
        this.data = data;
        this.num = num;
    }
    @Override
    public void run(){
        //showTip("setImageThreadinitialize...");
        bitmap = BitmapFactory.decodeByteArray(data, 0, num);
        Message msg=new Message();
        handler.sendMessage(msg);
    }
}
```

【driver_init】

增加菜单栏，其中包括"系统设置"和"退出"两个选项。对于前者会加载 setting 的

xml，而对于后者则会让系统 kill 掉自己。

【driver_surface】

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    // TODO Auto-generated method stub
    try{
        Toast.makeText(MainActivity.this, "create try ok", Toast.LENGTH_LONG).show();
        if(camera != null){
            camera.setPreviewDisplay(surfaceHolder);
            camera.startPreview();
        }
    }catch(IOException e){
        e.printStackTrace();
        Toast.makeText(MainActivity.this, "失败surface", Toast.LENGTH_LONG).show();
    }
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width,
                           int height) {
    // TODO Auto-generated method stub
    if (camera == null) return;
    camera.stopPreview();
    camera.setPreviewCallback(this);
    camera.setDisplayOrientation(90); //设置横屏录制
    Camera.Parameters parameters = camera.getParameters();
    parameters.setPreviewSize(640, 480);
    camera.setParameters(parameters);
    camera.startPreview();
}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    // TODO Auto-generated method stub
    if(camera != null){
        camera.setPreviewCallback(null);
        camera.stopPreview();
        camera.release();
        camera = null;
    }
}
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //常亮
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
        WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    //无标题
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    //全屏
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FORCE_NOT_FULLSCREEN);
    setContentView(R.layout.activity_main);

    Transmit = (Button) findViewById(R.id.Transmit);
    Stop = (Button) findViewById(R.id.Stop);
    surfaceView = (SurfaceView) findViewById(R.id.surfaceView1);
    //点击发送视频
    Transmit.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            flag = true;
        }
    });
    //点击停止发送
    Stop.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            flag = false;
        }
    });
}
```

【driver_option】

如果 flag 不为 0，那么每次计数器到设置的刷新率 videoRate 时就将图片发送出去。将拍摄得到的 YUV 以指定 videoRate，(videoRate 越高，最后的图像质量越好)转换为 jpg 格式后放到 ByteArrayOutputStream 的一个实例 outputStream 中，然后再通过 sendThread(outputStream,name,IP,serverPort)发送到"AE86"上。

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    super.onOptionsItemSelected(item); // 获取菜单
    switch(item.getItemId()) // 菜单序号
    {
        case 0: // 系统设置
        {
            Intent intent = new Intent(this, SettingActivity.class);
            startActivity(intent);
        }
        break;
        case 1: // 退出
        {
            android.os.Process.killProcess(android.os.Process.myPid());
        }
        break;
    }
    Toast.makeText(MainActivity.this, "Option ok!", Toast.LENGTH_LONG).show();
    return true;
}
```

【driver_flush】

sendThread 的实现步骤如下：先与指定 ip 的设备的指定端口进行套接字连接，然后根据套接字对象 socket 建立输出流 out。每一个线程都会努力将当前 buffer 中的所有数据输出到 out。注意此时 outputStream 也在不断接受图片数据，sendThread 会在某个 outputStream 中的数据为空时停止，否则将 outputStream 中的数据放到 buffer 中，准备下一次循环时，将 buffer 中的数据发送到指定 ip 和端口。

```
@Override
public void onPreviewFrame(byte[] data, Camera camera) {
    // TODO Auto-generated method stub
    Toast.makeText(MainActivity.this, "onPreviewFrame initialize...", Toast.LENGTH_LONG).show();
    if(!flag) return;
    // 视频刷新
    if(tmpRate < videoRate){
        tmpRate++;
        return;
    }
    tmpRate = 0;
    // Toast.makeText(MainActivity.this, "before try!", Toast.LENGTH_LONG).show();
    // 使用系统自带的类来发送图片
    Camera.Size size = camera.getParameters().getPreviewSize(); // 获取大小
    try { // 将YUV格式图像数据data转换成jpg
        if(data != null){
            // Toast.makeText(MainActivity.this, "try OKay!", Toast.LENGTH_LONG).show();
            YuvImage image = new YuvImage(data, ImageFormat.NV21, size.width, size.height, null);
            if(image != null){
                // 使用handler将图片发送出去
                ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
                // to jpg
                image.compressToJpeg(new Rect(0, 0, size.width, size.height), videoQuality, outputStream);
                outputStream.flush();
                Thread thread = new sendThread(outputStream, name, IP, serverPort);
                thread.start();
            }
        }
    }
} catch (Exception ex) {
    Log.e("Sys", "Error:" + ex.getMessage());
    Toast.makeText(MainActivity.this, "失败", Toast.LENGTH_LONG).show();
}
}
```

【driver_send1】

```
//发送图片线程
class sendThread extends Thread{
    private String name;
    private String ip;
    private int PORT;
    private int num = 0;
    private byte[] buffer = new byte[2048];
    private ByteArrayOutputStream output;
    public sendThread(ByteArrayOutputStream output,String name,String ip,int port){
        this.output = output;
        this.name = name;
        this.ip = ip;
        this.PORT = port;
        try{
            Toast.makeText(MainActivity.this, "sendThread initialize...", Toast.LENGTH_LONG).show();
            output.close();
        }catch(Exception e){
            e.printStackTrace();
            Toast.makeText(MainActivity.this, "失败", Toast.LENGTH_LONG).show();
        }
    }
}
```

【driver_send2】

最后通过重写 onStart 来设置 name, IP, videoRate, videoQuality 的 default 值, 当然可以通过 setting 中改写。另外在 driver 的 mainActivity 开始前申请了对于摄像机的权利, 对应地, 在生命周期结束前释放权利。

```
@Override
public void run(){
    Toast.makeText(MainActivity.this, "before thread's try!", Toast.LENGTH_LONG).show();
    try{
        Toast.makeText(MainActivity.this, "sendThread working...", Toast.LENGTH_LONG).show();
        Socket socket = new Socket(ip,PORT);
        OutputStream out = socket.getOutputStream();
        ByteArrayInputStream inputStream = new ByteArrayInputStream(output.toByteArray());
        num = inputStream.read(buffer);
        do{
            Toast.makeText(MainActivity.this, "success", Toast.LENGTH_LONG).show();
            out.write(buffer,0,num);
            num = inputStream.read(buffer);
        }while(num >= 0);
        out.flush();
        out.close();
        socket.close();
    }catch(UnknownHostException e){
        e.printStackTrace();
        Toast.makeText(MainActivity.this, "失败", Toast.LENGTH_LONG).show();
    }catch(IOException e){
        e.printStackTrace();
        Toast.makeText(MainActivity.this, "失败", Toast.LENGTH_LONG).show();
    }catch (Exception e){
        Toast.makeText(MainActivity.this, "失败", Toast.LENGTH_LONG).show();
    }
}
}
```

【driver_life】

```

@Override
protected void onStart() {
    // TODO Auto-generated method stub
    surfaceHolder = surfaceView.getHolder();
    surfaceHolder.addCallback(this);
    surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS); // 显示器类型
    // 读取配置文件
    SharedPreferences preParas = PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
    name = preParas.getString("name", "car");
    IP = preParas.getString("IP", "192.168.1.102");
    String tmp = preParas.getString("serverPort", "8888");
    serverPort = Integer.parseInt(tmp);
    tmp = preParas.getString("videoRate", "1");
    videoRate = Integer.parseInt(tmp);
    tmp = preParas.getString("videoQuality", "85");
    videoQuality = Integer.parseInt(tmp);
    super.onStart();
}

@Override
protected void onResume() {
    // TODO Auto-generated method stub
    super.onResume();
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
        != PackageManager.PERMISSION_GRANTED) {
        // 申请WRITE_EXTERNAL_STORAGE权限
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
            1);
    }
    try {
        camera = Camera.open();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@Override
protected void onPause() {
    // TODO Auto-generated method stub
    super.onPause();
    try {
        if (camera != null) {
            camera.setPreviewCallback(null);
            camera.stopPreview();
            camera.release();
            camera = null;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

5、系统演示录像（DV）或一组系统运行时照片



演示视频压缩版.mp4

6、系统测试情况

小车测试时一切正常，前进，后退，左转，右转指令正常执行，手势，重力感应，语音，视频互传功能表现良好。

7、系统的不足

自我评价

李珉超：在这个学期的工作中，我对于安卓应用开发的流程有了真实的感受。我们的开发平台是 win10 上的 android studio，开发使用的编程语言是 java。开发和调试的过程巩固了我对于 java 的认识和使用方法。在项目中收获颇丰的自然是在于对于 android 库的了解，通过对于官方 API 的阅读，我了解到每一个 app 的界面的加载流程，控件的生命周期、控件与软件行为之间的关联方法以及 移动端包括蓝牙，陀螺仪的控制。我也了解到 apk 开发过程中不同的文件的作用：权限申请、界面布局、主活动、用于动态加载 的活动等。另外，在语音识别部分的开发过程中，我们使用的是讯飞的 android 包。通过网上的博客，在不断的尝试后，我们知道了讯飞库的使用方法。本学期的项目着实涵盖了 android 开发过程中的诸多部分，相信本学期的经历会为我以后可能的 android 开发工作提供极大的指导。最后想感谢我的组员们，所有人都齐心协力，各司其职，为我们的项目画上了完美的句号。

崔家铭：在本学期的工程实践与科技创新III-C 中，我主要负责的是硬件部分的连接与调试，以及最后的网页的制作。

对于硬件部分，在参考了课程相关的资料后，我们基本能够将相关的器件连接上。但由于一些元件与参考资料中所提供的样例不一样，因此在这个问题上，我们消耗了一定的时间。特别是这一问题实际上仍能驱动小车，只是在左右转相关的方面上有问题，因此这实际上将问题复杂化了，而我们也确实花了很长时间才排除了软件方面存在问题的猜测。

实际上，这门课程从某种意义上，是一本实践性的课程。通过对相关知识的自学，我们完成了各自应当完成的部分，而从另一个角度上讲，这种依靠个人自学和团队协作相结合的学习方式，是即使到目前在很多课程中仍不多见的，最终，我们成功的完成了相应的测试，并取得了相应的成果，而在接下来的时间中，我更希望能够顺利的完成收尾工作，从而完满的结束这一课程的全部内容。

王晨奕：本次项目中，我负责的部分是软件端的页面设计、实现以及整个软件的简单测试。在界面实现的过程中，我通过 android studio 的可视化 UI 设计来最终实现了软件界面，编写代码来进行细节上的优化，从中对 android 端软件设计的文件结构和 java 语言特性有了初步的认识。在软件的测试环节，android studio 提供的手机模拟器在本次项目中仅能显示软件的界面，而对于代码和细节的优化，我使用了 monkey 做了简单的软件测试，以此对代码做了功能方面的细节修改，从而对软件与小车的接口部分有了更深的理解。

强志文：在本次项目中，我主要负责硬件端的连接与设计，以及最后报告的撰写。对于小车的连接，我主要参考了《小车电路设计参考科创》[1]一文中对于电机驱动模块和主体接线的介绍，在单片机编程中，我参考了工科创 2B 中的代码，使用相同的 IDE 和总体思路，最后在蓝牙连接过程中，因为我们的连线没有问题，所以这方面一切正常。但是由于我们最初的电机驱动模块的连接有问题，所

以长时间里小车无法正常的左转右转，最后我们排除了这个故障，这也让我们在以后的实践中更加注重细节。

通过这门课程的学习，我不仅对于硬件编程有了更深的体会，同时对于具体的线路连接问题，以及团队合作中如何协调有了更深的认识。在这里我想感谢老师一学期以来线上的解疑答惑，为我们指明了方向，还有其他的小组的热心帮助，我们小组成员之间的相互配合。

9、附录

[1] 《小车电路设计参考科创》

[2] 《2017-09-KC-3C》