手机蓝牙控制小车

姓名：陈昊喆　　**学号**：3130000368

项目介绍：

利用 arduino 和蓝牙模块与手机建立通信，从而通过 android 手机控制小车。

1．小车组成

1. 小车底盘 X 1；
2. 直流减速电机 X 2；
3. 轮子 X 2；
4. 万向轮 X 1；
5. 4 节 5 号电池盒 X 1；
6. Arduino Uno 板子 X 1；
7. HC-06 蓝牙模块 X 1；
8. L298N 电机驱动模块 X 1；
9. 导线和紧固件若干；

2. 控制方式

设计可通过 4 种方式进行控制：

1. 字符命令：
   通过蓝牙串口向 arduino 发送特定字符，从而实现对小车的控制，主要用于调试和基础功能测试。
2. 按键命令：
   在 android 手机上设置特定按键，触摸按键发送相应的指令，从而控制小车，简化了字符控制方式，可以简便地控制小车并实现简单功能。
3. 滑动条对两电机单独控制：
   在 android 手机上设置两个独立的滑动条，根据滑动条的位置对两个电机进行单独控制，比较灵活，可以实现各种动作，但是操作难度较大，不易进行准确控制。
4. 通过手机姿态控制：
   利用 android 手机自带的重力感应获取当前手机姿态，从而确定小车的行进方式，通过蓝牙控制小车。操作比较方便灵活，灵敏度比较高，更容易准确快捷地对小车进行调整。

模块介绍：

1. HC-06 蓝牙模块

蓝牙参数特点

1.蓝牙核心模块使用 HC-06 从模块，引出接口包括 VCC,GND,TXD,RXD,预留 LED 状态输出脚，单片机可通过该脚状态判断蓝牙是否已经连接

2.led 指示蓝牙连接状态，闪烁表示没有蓝牙连接，常亮表示蓝牙已连接并打开了端口

3.输入电压 3.6~6V，未配对时电流约 30mA，配对后约 10mA，输入电压禁止超过 7V！

4.可以直接连接各种单片机（51，AVR，PIC，ARM，MSP430 等），5V 单片机也可直接连接

5.在未建立蓝牙连接时支持通过 AT 指令设置波特率、名称、配对密码，设置的参数掉电保存。蓝牙连接以后自动切换到透传模式

6.体积　3.57cm*1.52cm

7.该蓝牙为从机，从机能与各种带蓝牙功能的电脑、蓝牙主机、大部分带蓝牙的手机、Android、PDA、PSP 等智能终端配对，从机之间不能配对。

8.蓝牙 Class 2 功率级别

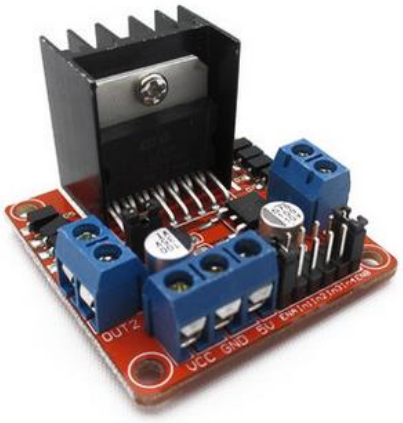
Arduino 与蓝牙模块连接方法

VCC：接 Arduino 的 5V。
GND：接 Arduino 的 GND。

TXD：发送端，一般表示为自己的发送端，接 Arduino 的 RX。
RXD：接收端，一般表示为自己的接收端，接 Arduino 的 TX。
正常通信时候本身的 TXD 永远接设备的 RXD！正常通信时 RXD 接其他设备的 TXD。

连接后正常使用串口功能即可，注意烧录程序时断开 TX 和 RX 的连接。

2. L28N 电机控制模块



一、产品参数：

1、驱动芯片：L298N 双 H 桥直流电机驱动芯片

2、驱动部分端子供电范围 Vs：＋5V～＋35V ； 如需要板内取电，则供电范围 Vs：+7V～+35V

3.驱动部分峰值电流 Io：2A

4.逻辑部分端子供电范围 Vss：＋5V～＋7V（可板内取电＋5V）

5.逻辑部分工作电流范围:0～36mA

6.控制信号输入电压范围：

低电平：－0.3V≤Vin≤1.5V

高电平：2.3V≤Vin≤Vss

7.使能信号输入电压范围：

低电平：－0.3≤Vin≤1.5V（控制信号无效）

高电平：2.3V≤Vin≤Vss（控制信号有效）

8.最大功耗：20W（温度 T＝75℃时）

9.存储温度：－25℃～＋130℃

10.驱动板尺寸:55mm*49mm*33mm(带固定铜柱和散热片高度)

11.驱动板重量：33g

12.其他扩展：控制方向指示灯、逻辑部分板内取电接口。

使用时电池盒正极接 VCC 口，负极接 GND。In1,2,3,4 四个引脚分别控制 out1,2,3,4 个供电口的电压输出。例如向 In1 引脚通高电频，就能从 out1 供电口得到高电压。从而控制这 4 个引脚就能控制 2 个电机的正转与反转。

两个势能引脚 ENA 和 ENB 用于向两个电机发送转动信号，最初已用跳线帽连接至板内的 5V，去掉跳线帽，将 ENA，ENB 接至 Arduino 通过 PWM 就能控制电机转速。

注意观察参数信息，发现如果需要板内取电，则供电电压需要在 7V 以上，现在使用的 4 节 5 号电池电压为 6V，则不能通过板内取电的方式向 Arduino 供电，需要直接将电池的正负极并接在 Arduino 的 Vin 和 GND 上。（经测试如果使用板内取电的方式向 Arduino 供电，则从 Arduino 取电的蓝牙模块不能正常工作）

In1	In2	电机 A	In3	In4	电机 B
0	1	正转	0	1	反转
1	0	反转	1	0	正转
0	0	停止	0	0	停止
1	1	停止	1	1	停止

3. 直流减速电机：

减速直流电机/减速电机

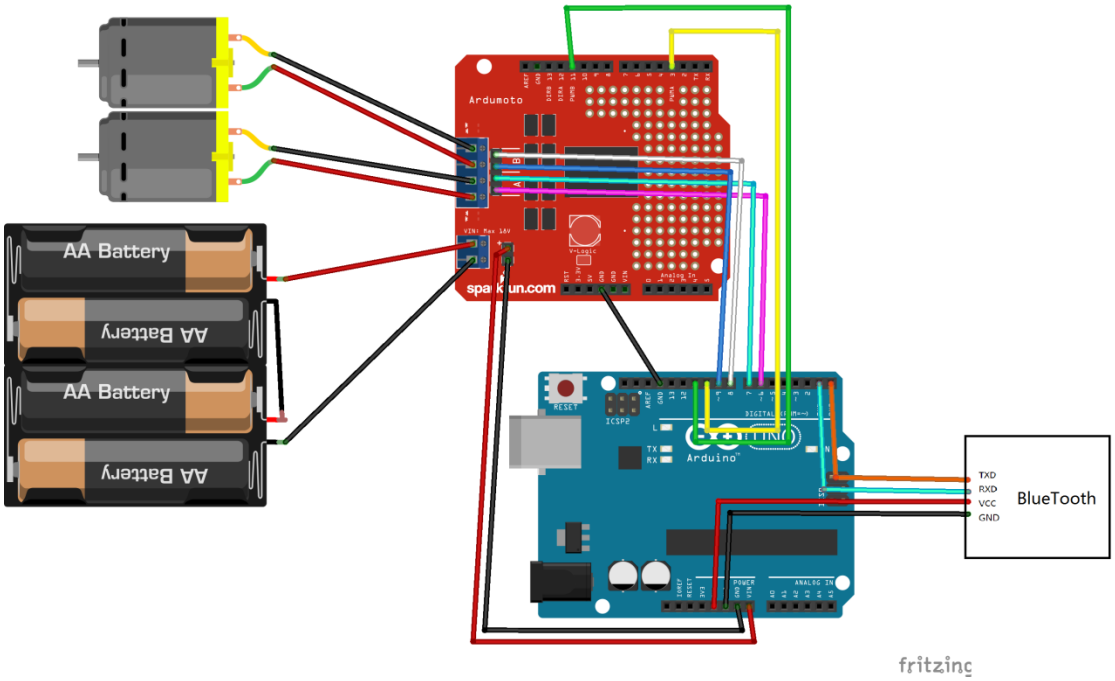
电机参数：

工作电压	DC 3V	DC 5V	DC 6V
工作电流	100ma	100ma	120ma
减速箱减速比	48:1		
空载（加轮子）	100 转/分	190 转/分	240 转/分
轮胎直径	6.6cm		
空载速度	20 米/分	39 米/分	48 米/分
重量	50g		
外形尺寸	70mm*22mm*18mm		
噪声	<65db		

正通电就正传，反通电就反转，通过 PWM 控制输入电压从而控制电机转速。

项目实现：

1. 接线：



其中使用 4 节 5 号电池作为电源，接上电源控制模块的正负极，同时并联 Arduino 的 Vin 和 GND 向 Arduino 供电，两电机接分别接在 A,B 两个供电输出上，Arduino 上的 6,7,8,9 号引脚接电源控制模块对应两电机的 4 个接口，10 号引脚接 PWM A 用于控制 A 电机的转速，11 号引脚接 PWM B 用于控制 B 电机的转速。Arduino 的 5V 接蓝牙模块的 VCC，GND 对接，向蓝牙模块供电。蓝牙模块的 TX 接 Arduino 的 RX，RX 接 Arduino 的

TX，实现 arduino 和蓝牙模块间的串口通信。

2. Arduino 端

配置好蓝牙模块后只需要正常使用串口通信就行了。两个电机的正转反转也只需通过对特定引脚设定高低电频即可。而两个电机的转速只要通过 PWM 调整对应使能信号的引脚输出即可。相对比较简单。

```
//work.ino
#define abs(a) (((a) > 0)?(a):(-a))
int out[5] = {-1,6,7,8,9};//对应 L298N 上的 In1,2,3,4 号引脚，控制 OUT1,2,3,4 供电
int Left = 10,Right = 11;//对应左右电机的使能信号，利用 PWM 控制转速
void LMove(int u) {//控制左边的电机
    switch (u) {
        case -1://反转
        digitalWrite(out[1], HIGH);
        digitalWrite(out[2], LOW);
        break;
        case 1://正转
        digitalWrite(out[1], LOW);
        digitalWrite(out[2], HIGH);
        break;
        default://停止
        digitalWrite(out[1], LOW);
        digitalWrite(out[2], LOW);
    }
}
void RMove(int u) {//控制右边的电机
    switch (u) {
        case -1://反转
        digitalWrite(out[4], HIGH);
        digitalWrite(out[3], LOW);
        break;
        case 1://正转
        digitalWrite(out[4], LOW);
        digitalWrite(out[3], HIGH);
        break;
        default://停止
        digitalWrite(out[4], LOW);
        digitalWrite(out[3], LOW);
    }
}
void setup() {
    Serial.begin(9600);//打开串口
    analogWrite(Left, 255);//默认使能最高
    analogWrite(Right, 255);
```

```
    for (int i = 1;i <= 4;i++) {
        pinMode(out[i], OUTPUT);
    }
}
void loop() {
    int v;
    while (Serial.available()) {
        char ch = Serial.read();
        Serial.write(ch);//向 Android 手机反馈信息，反馈的就是得到的字符
        switch(ch) {
            case 'w'://前进
                   RMove(1);
                   LMove(1);
               break;
            case 's'://后退
                   RMove(-1);
                   LMove(-1);
               break;
            case 'a'://左转
                   LMove(0);
                   RMove(1);
               break;
            case 'd'://右转
                   LMove(1);
                   RMove(0);
               break;
            case 'p'://停止
                   LMove(0);
                   RMove(0);
               break;
            case 'l'://改变左边电机的使能信号，但不改变电机的控制信号
            case 'L'://改变左边电机的使能信号，同时改变电机的控制信号
               v = Serial.parseInt();
               if (ch == 'L') {
                   if (v > 80) LMove(1);
                   else if (v < -80) LMove(-1);
                   else LMove(0);
               }
               analogWrite(Left, abs(v));;
               break;
            case 'r'://改变右边电机的使能信号，但不改变电机的控制信号
            case 'R'://改变右边电机的使能信号，同时改变电机的控制信号
               v = Serial.parseInt();
               if (ch == 'R') {
```

```
                if (v > 80) RMove(1);
                else if (v  < -80) RMove(-1);
                else RMove(0);
            }
            analogWrite(Right, abs(v));
            break;
        }
    }
}
```

3. Android 端

Android 主要需要实现对蓝牙模块的配对，监视数据传输，同时实现 4 种控制方式。

1. 蓝牙

由于现有设备已确定，所以采用固定 MAC 地址连接的方式，省去了搜索可配对蓝牙设备的环节。在确定了蓝牙模块的 MAC 地址之后直接在 Android 内部代码里内置好相应信息即可。

```
private static final UUID myUUID =
    UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
private static String address = "98:D3:31:B3:7E:B1";
```

这里的蓝牙通信使用的是 RFCOMM 协议，也就是一个蓝牙串口的传输协议，波特率为 9600，8 位数据。首先要在 AndroidMainFest.xml 里设置使用蓝牙的协议。

```
<uses-permission
    android:name="android.permission.BLUETOOTH" />
<uses-permission
    android:name="android.permission.BLUETOOTH_ADMIN" />
```

首先要获取一个对蓝牙操作的蓝牙适配器，BluetoothAdapter。

```
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```

之后就能通过 BluetoothAdapter 利用最初设定的 MAC 地址获取蓝牙设备。

```
BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
```

对获取的设备创建嵌套字服务。

```
try {
            btSocket =  device.createRfcommSocketToServiceRecord(myUUID);
        } catch (IOException e) {
            e.printStackTrace();
        }
```

连接设备：

```
try {
          btSocket.connect();
          Log.d("Connect", "BT connection established, data transfer link open.");
       } catch (IOException e) {
          try {
             btSocket.close();
             e.printStackTrace();
          } catch (IOException ee) {
             ee.printStackTrace();
          }
        }
```

控制过程中与蓝牙之间的工作主要有两个，发送和接收。发送用于向蓝牙模块发送信息指令，接收用于获取蓝牙模块返回的反馈信息。

发送：

从 Socket 中获取输出流，向输出流中写入字节就能以字节流的方式向蓝牙模块发送消息。每次发送前检查是否已连接，如果连接断开了则重新连接。

```
   public void Send(String st) {
      if (!btSocket.isConnected()) {
         try {
            Log.d("Connect","Not Connected");
            btSocket.connect();
            Log.d("Connect","Connect established");
         } catch (IOException e) {
            e.printStackTrace();
            Log.e("Connect","Send Failed");
            Toast.makeText(this, "Not Connected", Toast.LENGTH_SHORT);
            return;
         }
      }
      try {
         outStream = btSocket.getOutputStream();
      } catch (IOException e) {
         Log.e("Connect","Output stream creation failed");
         e.printStackTrace();
      }
      byte[] buffer = st.getBytes();
      try {
         outStream.write(buffer);
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
```

接收：

接收需要创建新的监听线程，从 Socket 获取输入流，从输入流中得到字节数组，转换为字符串发送回程序用于处理。

```
private class ReadThread extends Thread {
    public void run() {
        byte[] buffer = new byte[1024];
        int bytes = 0;
        InputStream myIn = null;

        try {
            myIn = btSocket.getInputStream();
        } catch (IOException e) {
            e.printStackTrace();
        }

        while (true) {
            try {
                if ((bytes = myIn.read(buffer)) > 0) {//得到字节流
                    char data[] = new char[bytes];
                    for (int i = 0;i < bytes;i++)
                        data[i] = (char)buffer[i];
                    String s = new String(data);//得到字符串
                    Message msg = new Message();//用于传递消息
                    msg.obj = s;
                    msg.what = 0;
                    listenHandler.sendMessage(msg);
                }
            } catch (IOException e) {
                try {
                    myIn.close();
                } catch (IOException ee) {
                    ee.printStackTrace();
                }
                break;
            }
        }
    }
}
```

对于得到的反馈消息，这里将它显示在屏幕中。

```
Handler listenHandler = new Handler() {
    public void handleMessage(Message msg) {
        if (msg.what == 0) {
            String s = msg.obj.toString();
            textView.setText(s);
        }
    }
};
```

在 Android 上总共实现 4 中控制方式，字符指令，按键，拖动条，重力感应。

字符指令：

从输入框获取字符，触摸发送按键时发送指令。

```
    editText= (EditText)findViewById(R.id.Send_text);
    String msg = editText.getText().toString();
    Send(msg);
```

按键：

触摸相应按键时发送特定字符指令。

```
public void ClickForward(View view){
    Send("w");
}
public void ClickBack(View view) {
    Send("s");
}
public void ClickLeft(View view) {
    Send("a");
}
public void ClickRight(View view) {
    Send("d");
}
public void ClickPause(View view) {
    Send("p");
}
public void ClickMin(View view) {
    Send("l0r0");
}
public void ClickMax(View view) {
    Send("l255r255");
}
```

拖动条：

设定初始位置为拖动条中间，也就是电机停止的位置，向前滑动电机正传，向后滑动电机反转。

```
        rightBar = (SeekBar)findViewById(R.id.rightbar);
        rightBar.setProgress(50);
        rightBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

            @Override
            public void onStopTrackingTouch(SeekBar arg0) {
            }

            @Override
            public void onStartTrackingTouch(SeekBar arg0) {
            }

            @Override
            public void onProgressChanged(SeekBar arg0, int arg1, boolean arg2) {
                int tmp = arg1 - 50;
                tmp = (int)((double)tmp * 5.1);
                rightText.setText(String.valueOf(tmp));
                Send("R" + String.valueOf(tmp));
            }
        });
```

两个电机分别控制，左电机与此相同。

重力感应：

重力感应需要先注册相应的传感器服务，同时主程序需要拓展相应接口。

这里我们选定的是 SensorManager.SENSOR_DELAY_GAME 级别的灵敏度，也就是 20ms。

```
public class MainActivity extends Activity implements SensorEventListener {
```

```
        SensorManager manager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        if (manager.getSensorList(Sensor.TYPE_ACCELEROMETER).size() ==0) {
            Toast.makeText(getBaseContext(), "No Accelerometer installed",
Toast.LENGTH_SHORT).show();
        } else {
            Sensor accelerometer =
manager.getSensorList(Sensor.TYPE_ACCELEROMETER).get(0);
            if (!manager.registerListener(this, accelerometer,
SensorManager.SENSOR_DELAY_GAME)) {
                Toast.makeText(this, "Can't register listenter",
Toast.LENGTH_SHORT).show();
            }
        }
```

重写重力感应监听函数。监听函数能够从监听事件得到三个参数，对应重力感应服务就是 X,Y,Z 三个方向上的加速度，因为这里只考虑静止姿态，所以这三个参数其实是 X,Y,Z 三个方向上的重力加速度分量。所以知道其中两个就能确定第三个参数，这里只

取用其中两个。选取第一个参数 values[0]也就是 X 方向上的参数为方向参数，用于控制小车的前进方向。选取第二个参数 values[1]也就是 Y 方向上的参数为动力参数，用于控制小车两电机的转速。在最初动力参数的基础上，将方向参数所指向方向的电机的动力参数减去方向参数，通过调整就能控制前进方向。

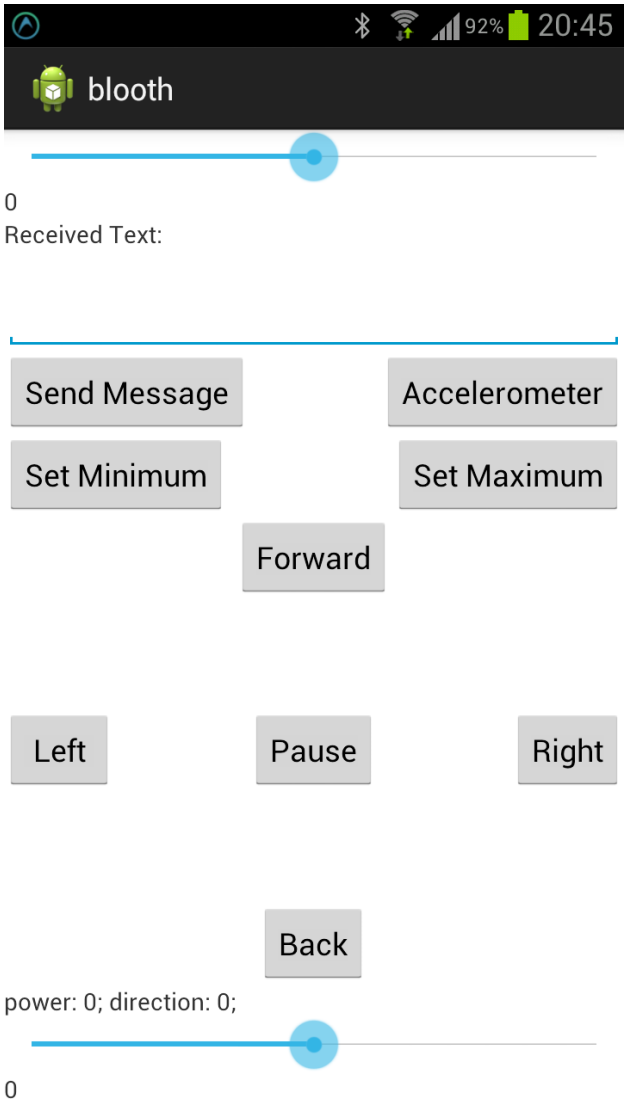
```
public void onSensorChanged(SensorEvent event) {
    if (Accelerometer_on) {
        int direction = (int) (event.values[0]/9.8 * 255.0);
        int power = (int)(event.values[1]/9.8 * 255.0);
        int Left,Right;
        Left = Right =  power;
        if (direction < -50) {
            Left += direction;
        } else if (direction > 50) {
            Right -= direction;
        }
        if (Left > 255) Left = 255;
        if (Right > 255) Right = 255;
        if (Left < -255) Left = -255;
        if (Right < -255) Right = -255;
        Left = -Left;
        Right = -Right;
        try {
            builder.setLength(0);
            builder.append('L');
            builder.append(Left);
            builder.append('R');
            builder.append(Right);
            byte[] buffer = builder.toString().getBytes();
            OutputStream output = btSocket.getOutputStream();
            output.write(buffer);
        } catch (IOException e) {
            e.printStackTrace();
        }
        HUD.setText(builder.toString());
    }
}

public void onAccuracyChanged(Sensor sensor, int accuracy) {
}
```

同时设置一个按键用于开关重力感应控制，否则其他控制方式无法正常工作。

```
public void ClickAccelerometer(View view) {
    Accelerometer_on = !Accelerometer_on;
    if (!Accelerometer_on) HUD.setText("Accelerometer off");
}
```

界面布局整体采用 LinearLayout，嵌套 RelativeLayout 和 GridLayout。




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <SeekBar
        android:id="@+id/leftbar"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"/>
    <TextView
        android:id="@+id/lefttext"
```

```
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:text="0"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Received Text:" />

<TextView
    android:id="@+id/Receive_text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<EditText
    android:id="@+id/Send_text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >

    <Button
        android:id="@+id/bt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:onClick="Send"
        android:text="Send Message" />

    <Button
        android:id="@+id/accelerometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:onClick="ClickAccelerometer"
        android:text="Accelerometer" />
</RelativeLayout>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" >

    <Button
        android:id="@+id/bt"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="Set Minimum"
        android:onClick="ClickMin"/>

    <Button
        android:id="@+id/accelerometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:text="Set Maximum"
        android:onClick="ClickMax" />
</RelativeLayout>

<GridLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="0.66"
    android:columnCount="1" >

    <Button
        android:id="@+id/button1"
        android:layout_column="0"
        android:layout_gravity="center_horizontal|top"
        android:layout_row="0"
        android:onClick="ClickForward"
        android:text="Forward" />

    <Button
        android:id="@+id/button3"
        android:layout_column="0"
        android:layout_gravity="left|center_vertical"
        android:layout_row="0"
        android:onClick="ClickLeft"
        android:text="Left" />

    <Button
        android:id="@+id/button4"
        android:layout_column="0"
        android:layout_gravity="right|center_vertical"
        android:layout_row="0"
        android:onClick="ClickRight"
        android:text="Right" />
```

```
        <Button
            android:id="@+id/button5"
            android:layout_column="0"
            android:layout_gravity="center"
            android:layout_row="0"
            android:onClick="ClickPause"
            android:text="Pause" />

        <Button
            android:id="@+id/button6"
            android:layout_column="0"
            android:layout_gravity="center_horizontal|bottom"
            android:layout_row="0"
            android:onClick="ClickBack"
            android:text="Back" />
    </GridLayout>

    <TextView
        android:id="@+id/HUD"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="power: 0; direction: 0;" />
    <SeekBar
        android:id="@+id/rightbar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/righttext"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="0"/>

</LinearLayout>
```

项目总结

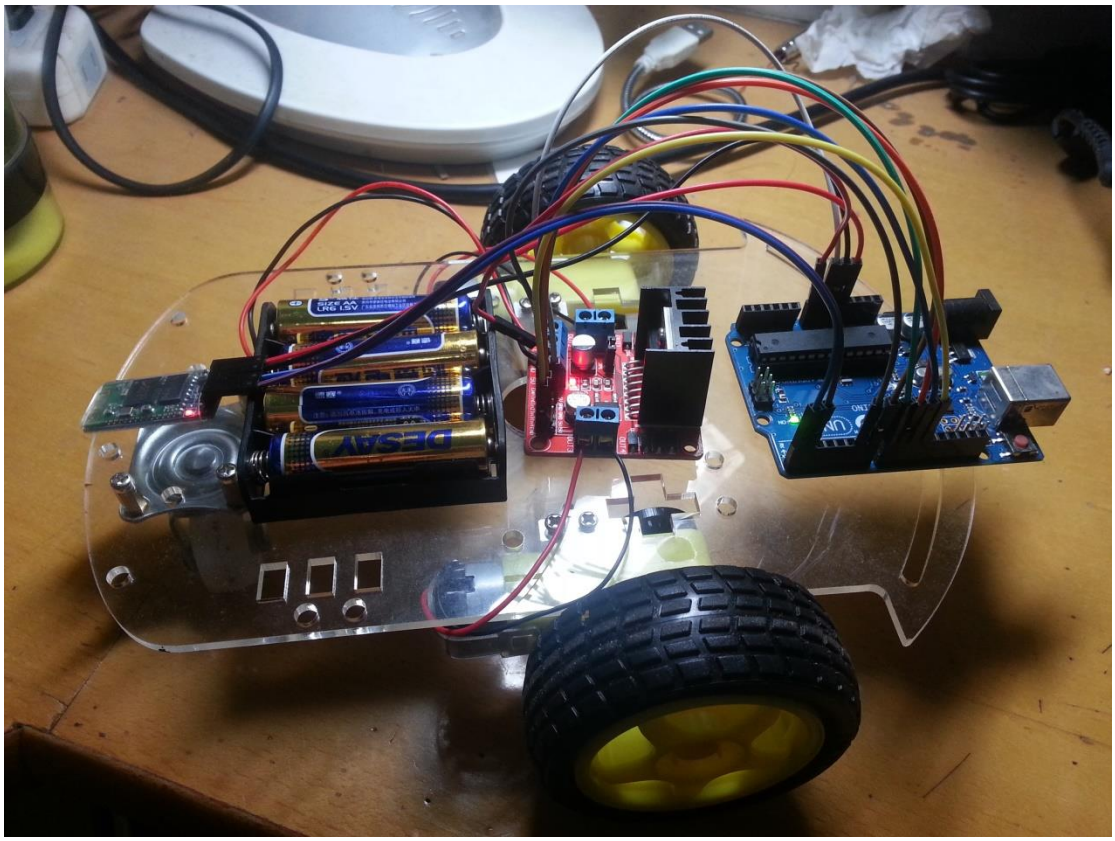
项目基本成功实现。成功完成了 Android 手机与蓝牙模块的对接，实现了全部 4 种控制方式，成功通过 L298N 电机控制模块控制两个相互独立的直流减速电机。

关于蓝牙连接：

以固定 Mac 地址的方式实现了与蓝牙模块的连接。此方法只能控制特定的蓝牙模块，对于其他的蓝牙模块不适用。当对应固定蓝牙模块时这种方法比较方便，但当更换蓝牙

模块时需重新更改 App 源码。如需要利用此 App 控制多个设备时可加入蓝牙设备的搜索功能，但加入搜索功能会消耗多余资源，且每次搜索设备也会引起不便。

问题：当手机资源占用过多时，连接蓝牙设备的进程会有延迟，导致程序启动后蓝牙没有正常连接，需重新启动程序。



关于四种控制方式：

成功实现了 4 种控制方式。

字符指令：能够准确地对小车下达指令，尤其对于两单机的 PWM 控制可以精确的控制使能信号的强度，从而控制电机转速，但是由于需要手动输入字符，控制不够方便，不能及时针对情况做出反应。

按键控制：按键能够方便快捷地对小车下达指令，但由于按键数量有限，只能实现基础的方向控制，当需要对两电机进行精确控制时，可以结合字符指令完成相应动作。

滑动条：滑动条成功实现了对两电机的分别控制，但是由于滑动条的精度有限，对两电机的转速控制虽然比较方便，也能实现基本的转速区分，但是不能够精确到特定数字。且当长时间使用滑动条时，因为程序和 Android 设备本身的延迟卡顿，有时会暂时失去对于两滑动条的控制。

手机姿态:手机姿态控制很容易针对当前情况对小车迅速下达相应指令，但是重力感应传感器会消耗大量资源。且手机姿态不易保持，对小车的精确控制取决于操作者针对小车行驶情况做出的调整。对操作者要求较高。

关于电机驱动模块控制两直流减速电机：

通过 L298N 电机驱动模块实现了对于两个电机的转向和转速的控制。成功实现了正转，反转，同时通过控制使能信号对两电机的转速能够做出调整和区分。

问题：但是由于两电机之间的区别、电机驱动模块本身向两电机输入电压的不定、包括万向轮对方向的影响在内的小车本身的结构问题，容易导致对于两电机的控制达不到预期的效果，精确下达的指令不能够被精确执行。最直观地表现为，调整两电机使能信号的 PWM 相同，前进指令发出后小车的前进方向有偏差，并非直线行驶，前进轨迹达不到预期中的效果。

考虑可以添加测速模块，实时测量两电机的转速，通过 PID 控制进行实时调整，从而使得指令能够被精确执行。在现有设备下尝试使用红外二极发射管和接收管，与直流减速电机配套的遮光码盘组成简易的测速装置，当电机带动码盘转动时，接收管可通过码盘上等间隔的空隙接收到红外发射管发出的红外信号，即可得知小车的转动过的角度，但是在现有设备下，简易的红外发射管和接收管得到的信号不准确，红发射管发出的信号散射过大，导致接收管经常能得到多余的错误红外信号。所以此方案因为设备问题失败，条件允许时可以再做尝试。

关于蓝牙连接和电源供电的稳定性：

从成品的图片中可以看到，向整个系统供电的是一个 4 节 5 号电池的电池盒，且电池盒裸露，电池也缺少固定。在测试的过程中发现，每当小车正面撞倒障碍物时，蓝牙模块的指示灯经常会变到闪烁的状态，显示蓝牙连接断开了。经过多次测试发现，是电池固定不稳，每当撞倒障碍物时，电池盒中用于保持电池连接的弹簧和电池会由于惯性的问题短暂断开，在短暂断开的过程中蓝牙设备断电，虽然能够迅速恢复供电，但是与 Android 控制器的蓝牙连接已经断开，需要重新连接。

考虑可以通过更换为整块的锂电池，从硬件上解决此问题。（未做）

也可以在 Android 段的 App 中加入检测，每次发送消息时检测是否连接，如果发现连接断开则重新连接，但是由于 Android 设备的延迟和连接过程的反锁，效果不佳。（已做）

项目至此结束，整体上实现了预期的效果，达到了基本目标。在某些方面仍存在一些不足，已经设想了解决办法并做了一些尝试，可以在今后继续补足。