基于 Kinect 和 Arduino 的手势遥控机器人设计

胡靖逸

中国矿业大学孙越崎学院 14 级本科生

摘 要：Kinect 和 Arduino 都是近年来新兴的开发平台，在各自的平台上都有相当的开发者进行项目开发，但是两平台交叉的却寥寥无几。本文介绍了一个手势遥控机器人的上述平台交叉项目，电脑使用 C#编程，操纵 Kinect 识别、捕捉手势命令，通过 WIFI 发送给的使用 Arduino 板驱动的机器人，机器人根据命令移动，抓取、放下物品或在自动控制模式下自主行动。

关键词：Kinect、Arduino、WIFI、机器人

Design of Gesture Remote Control Robot Based on Kinect and Arduino

Hu Jingyi

Grade 14 Undergraduate, Sun Yueqi Honors College

Abstract: Kinect and Arduino are emerging in recent years, the development of the platform, in their respective platforms have considerable development projects to develop, but the two platforms are very few cross. This paper describes a gesture remote robot for the above platform cross project, the computer uses C# programming, controls Kinect identification, capture gesture commands, uses WIFI to sending orders to the robot driven by Arduino , robot based on the command to move, grab, put down the object or in the automatic control mode of autonomous action.

Key word: Kinect; Arduino; WIFI; robot

一、引言

随着各种开发平台对底层的封装日渐成熟，完成机器人开发对底层技术掌握要求越来越低，使得开发人员可以集中精力处理控制、模块沟通上。Kinect 是微软在 2009 年 6 月 2 日的 E3 大展上，正式公布的 XBOX360 体感周边外设。它是一种 3D 体感摄影机，同时它导入了即时动态捕捉、影像辨识、麦克风输入、语音辨识、社群互动等功能；在开发上可以使用微软官方的 SDK——Kinect for Windows（支持 C# 与.NET Framework 4.0）。Arduino 是一款便捷灵活、方便上手的开源电子原型平台，包含硬件（各种型号的 Arduino 板）和软件（Arduino IDE)。它的硬件包含一个以 Atmel AVR 单片机为核心的开发板和其他各种 I/O 板。Arduino 语言基于 wiring 语言开发，是对 AVRGCC 库的二次封装，不需要太多的单片机基础、编程基础，简单学习后，即可进行项目开发。Arduino 的硬件原理图、电路图、IDE 软件及核心库文件都是开源的，在开源协议范围内里可以任意修改原始设计及相应代码。本文是在此背景下通过 Kinect 进行手势识别，在上位机生成控制信号，通过 WIFI 通信来控制 Arudino 驱动机器人的设计。

二、上位机程序设计

1.手势识别

Microsoft Kinect SDK 并没有包含手势识别引擎。手势识别相对来说可以简单也可以很复杂，这取决与要识别的手势。有三种基本的方法可以用来识别手势：基于算法，基于神经网络和基于手势样本库。每一种方法都有其优缺点。基于算法的手势识别相对简单容易实现，基于神经网络和手

势样本库则有些复杂。鉴于在本项目中实现的功能，选择使用基于算法的手势识别。接下来以挥动(wave)手势为例简述识别方法。

Xbox 中的挥手动作定义为：从胳膊开始到肘部弯曲。用户以胳膊肘为焦点来回移动前臂，移动平面和肩部在一个平面上，并且胳膊和地面保持平行,在手势的中部（如下图 1 所示），前臂垂直于后臂和地面。

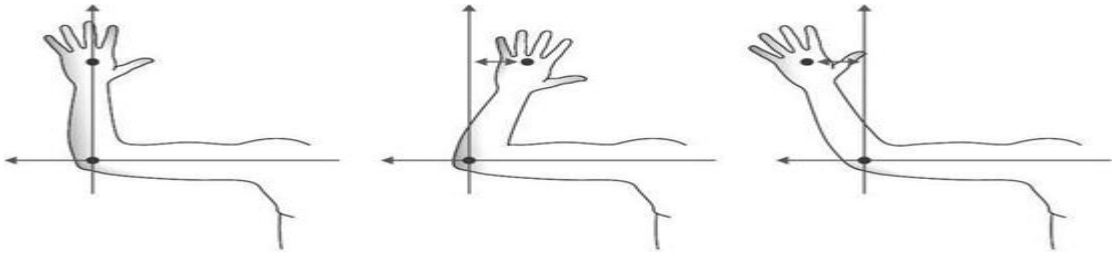


图 1:展示挥手运动状态

从图中可以观察得出一些规律，第一个规律就是，手和手腕都是在肘部和肩部之上的，这也是大多是挥手动作的特征。这也是我们识别挥手这一手势的第一个标准。

第一幅图展示了挥手这一姿势的中间位置，前臂和后臂垂直。如果用户手臂改变了这种关系，前臂在垂直线左边或者右边，我们则认为这是该手势的一个片段。对于挥手这一姿势，每一个姿势片段必须来回重复多次，否则就不是一个完整的手势。这一运动规律就是我们的第二个准则：当某一手势是挥手时，手或者手腕，必须在中间姿势的左右来回重复特定的次数。使用这两点通过观察得到的规律，我们可以通过算法建立算法准则，来识别挥动手势了。

算法通过计算手离开中间姿势区域的次数。中间区域是一个以胳膊肘为原点并给予一定阈值的区域。算法也需要用户在一定的时间段内完成这个手势，否则识别就会失败。这里定义的挥动手势识别算法只是一个单独的算法，不包含在一个多层的手势识别系统内。算法维护自身的状态，并在识别完成时以事件形式告知用户识别结果。挥动识别监视多个用户以及两双手的挥动手势。识别算法计算新产生的每一帧骨骼数据，因此必须记录这些识别的状态。[1]

2.socket 通信

在本方案中，通过 TCP 协议进行网络通信，使用 C#提供的应用编程接口（API）来实现 TCP/IP 协议软件的具体细节。具体则是调用 C#官方定义的 System.Net.Sockets 类，通过 C/S 架构（客户机/服务器结构)来实现将手势识别获取的数据发送到下位机(机器人)。其中上位机(电脑与 Kinect）为服务器，下位机（机器人）为客户端。以下为上位机编程。

在全局变量中，创建一个 Socket 对象（即请求系统为进程创建一个套接字）。当套接字被创建后，它的端口号和和 IP 地址都是空的，使用 Bind()函数将设置好的端口号和本地 IP 填写到已创建的套接字中。

在调用 Bind()函数后，程序还需调用 Listen()函数将套接字设置为被动方式（启动监听），以便随时接受下位机的服务请求（要求接受手势产生的命令）。为方便将来拓展上位机与下位机为一对多的工作模式，这里设计服务器的工作方式为并发方式，具体实现是将监听客户端连接的 ListenClientConnect（）函数启用线程执行，编程如下：

```
Thread myThread = new Thread(ListenClientConnect);
```
[2]

实际使用时，客户机和服务器都在 TCP 系统上使用 send 系统调用数据传送，使用 recv 系统调用数据接收。（C#中，使用 Send()函数和 Receive()函数）首先，下位机（机器人）使用 send 发送

请求，上位机使用 recv 接收请求，之后上位机使用 send 发送最新一次系统采集的手势识别产生的命令，下位机使用 recv 接收命令。之后下位机开始工作。[3]（下位机使用的 TCP/IP 的 API 在 5.4 中有介绍）

三、机器人硬件设计

机器人的硬件设计分为四部分：机械臂、小车部分、WIFI 模块、传感器模块、Arduino-UNO R3 主控板。

1.机械臂

机械手臂是机械人技术领域中得到最广泛实际应用的自动化机械装置，能够接受指令，精确地定位到三维（或二维）空间上的某一点进行作业。这里使用的机械臂使用 4 个舵机 P0090 及其他拼装材料组成，能达到 4 个自由度。如图 2 所示。

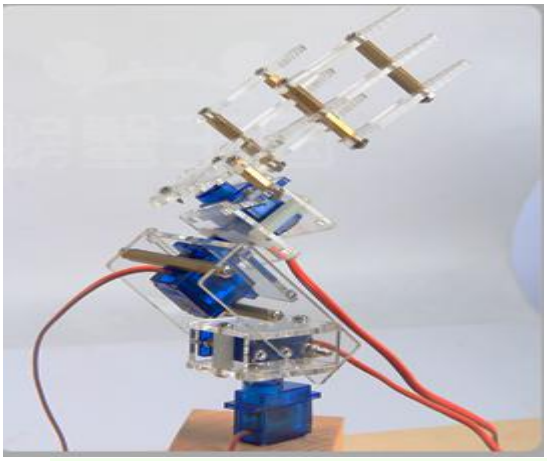


图 2:机械臂展示

图 3:减速电机展示

2.小车部分

小车部分包括小车底盘，4 个车轮及其减速电机，2 个 L298N 电机驱动模块及其供电电池盒。其中为了减小负重，小车底盘及车轮骨架均使用高强度的塑料板制作。

在齿轮马达的齿轮传动中，齿轮比是由齿轮中心与接触点之间的距离来决定的。如在含有两个齿轮的设备中，如果一个齿轮的直径是另外一个齿轮的两倍，那么齿轮比就是 2：1.在现代的齿轮减速电机中的齿轮均使用一种称为渐开线的特殊齿形，这是因为这种齿形有一个十分重要的特性，它们可以在两个齿轮间维持一个恒速比。

在斜齿轮减速电机上，轮齿的切口方向与齿轮表面成一定的角度。当斜齿轮系统上的两个轮齿啮合时，接触点将从轮齿的一端开始，并随着齿轮的旋转逐渐移到另一端，直至两个轮齿完全的啮合。如图 3 所示。

而减速电机正是通过这种啮合的方式，斜齿轮在运转时要比正齿轮平稳，噪音也比较小，所以，几乎所有的汽车变速齿轮箱都是使用斜齿轮。由于斜齿轮减速电机上的轮齿呈现一定的角度，因此当它们啮合时，齿轮将会受到一定的压力，使用斜齿轮的设备都装有轴承来承受这种压力。斜齿轮还有一个非常有趣的特点，那就是如果齿轮轮齿的角度适当，便可以将它们装载相互垂直的轴上，从而使啮合角度变为 90 度。[4]

L298N 是 ST 公司的产品，比较常见的是 15 脚 Multiwatt 封装的 L298N，内部同样包含 4 通道

逻辑驱动电路。可以方便的驱动两个直流电机，或一个两相步进电机。

L298N 芯片可以驱动两个二相电机，也可以驱动一个四相电机，输出电压最高可达 50V，可以直接通过电源来调节输出电压；可以直接用单片机的 IO 口提供信号；而且电路简单，使用比较方便。并且 L298N 具有防止过温功能和较高的噪声抑制比，故十分适用于本项目小车部分中。

L298N 可接受标准 TTL 逻辑电平信号 VSS，VSS 可接 4．5～7 V 电压。4 脚 VS 接电源电压，VS 别单独引出以便接入电流采样电阻，形成电流传感信号。L298 可驱动 2 个电动机，OUT1，OUT2 和 OUT3，OUT4 之间可分别接电动机，本实验装置我们选用驱动一台电动机。5，7，10，12 脚接输入控制电压范围 VIH 为＋2．5～46 V。输出电流可达 2 A，可驱动电感性负载。1 脚和 15 脚下管的发射极分电平，控制电机的正反转。EnA，EnB 接控制使能端，控制电机的停转。表 1 是 L298N 功能逻辑图。[5]

图 4:L298N 驱动模块展示

E_{nA}	In1	In2	运转状态
0	X	X	停止
1	1	0	正转
1	0	1	反转
1	1	1	刹停
1	0	0	停止

表 1：L298N 功能模块

3.WIFI 模块

鉴于成本及将来的可推广性，这里使用乐鑫信息科技有限公司的 ESP8266 进行 WIFI 通信。ESP8266 拥有高性能无线 SOC，是一个完整且自成体系的 WIFI 网络解决方案，能够独立运行，也可以作为 slave 搭载于其他 Host 运行。价格也相对低廉。

ESP8266 支持 softAP 模式，station 模式，softAP+station 共存模式三种，利用 ESP8266 可以实现十分灵活的组网方式和网络拓扑。

注：

SoftAP：即无线接入点，是一个无线网络的中心节点，通常使用的无线路由器就是一个无线接入点。

Station：即无线终端，是一个无线网络的终端。[6]

4.红外传感器模块

该红外传感器模块对环境光线适应能力强，其具有一对红外线发射与接收管，发射管发射出一定频率的红外线，当检测方向遇到障碍物（反射面）时，红外线反射回来被接收管接收，经过比较器电路处理之后，绿色指示灯会亮起，同时信号输出接口输出数字信号（一个低电平信号），可通过电位器旋钮调节检测距离，有效距离范围 2～30cm，工作电压为 3.3V-5V。该传感器的探测距离

可以通过电位器调节、具有干扰小、便于装配、使用方便等特点，可以广泛应用于机器人避障、避障小车、流水线计数及黑白线循迹等众多场合。

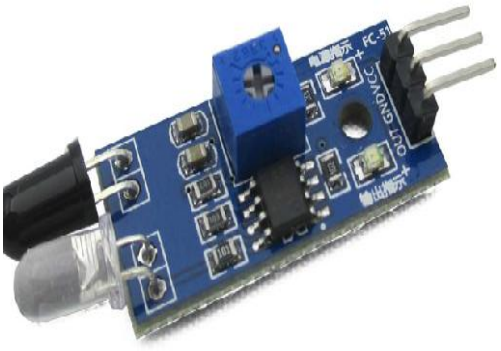


图 5：红外传感器模块实物图

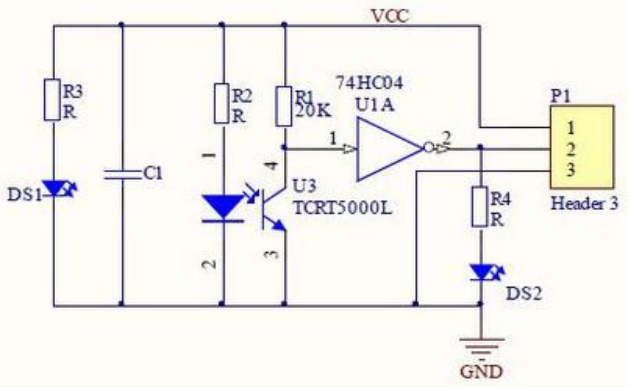


图 6：红外传感器模块原理图

5.Arduino-UNO R3 主控板

ArduinoUNO 是 ArduinoUSB 接口系列的最新版本，作为 Arduino 平台的参考标准模板。UNO 的处理器核心是 ATmega328，同时具有 14 路数字输入/输出口（其中 6 路可作为 PWM 输出），6 路模拟输入，一个 16MHz 晶体振荡器，一个 USB 口，一个电源插座，一个 ICSP header 和一个复位按钮。UNO 已经发布到第三版，与前两版相比有以下新的特点：

(1):在 AREF 处增加了两个管脚 SDA 和 SCL，支持 I2C 接口；增加 IOREF 和一个预留管脚，将来扩展板将能兼容 5V 和 3.3V 核心板

(2)改进了复位电路设计

(3)USB 接口芯片由 ATmega16U2 替代了 ATmega8U2[7]

四、Arduino 控制原理及程序设计

控制程序分为电机驱动模块、机械臂模块、传感器及自主避障模块、WIFI 通信模块及流程控制模块。

1.电机驱动

在本方案中，使用的两个电机驱动模块由 6V 电池盒供电，动力较小，故 ENA,ENB 两端直接接高电平，让小车全速运行。由 Arduino 控制 IN1-IN4，让信号线并联接在两个电机驱动模块上，再将 OUT1-OUT4 同时分别接在同侧的两个轮子上，实现一个信号控制两个轮子。程序样式如下：

```
 void Turn_left() //左转
{
  digitalWrite(In1, HIGH);
  digitalWrite(In2, LOW);
  digitalWrite(In3, LOW);
  digitalWrite(In4, HIGH);
}
```

此部分还包括包含函数 Turn_right()（右转），Run_forward()（前进），Run_behind()（后退）使用 Arduino 只对电机驱动模块的控制位 IN1-IN4 进行干涉，可以减少控制量，便于完成作品，在后期对产品调整时可将 ENA,ENB 接上以达到使用 PWM 调控小车速度的目的。

而且，在下一次调用移动函数前，小车将保持之前的状态，保证了运动的连续性。如一直前进，检测到左前方有障碍物之后向右拐弯，当检测不到时小测又切换成前进移动。这个在后面传感器及

自主避障模块会详细说明。

2. 机械臂

这里实际运用了卢光跃，吴涛[9]的理论，对使用 9，10 引脚控制舵机进而操纵机械臂。

```
/*****************机械臂部分初始化*****************/
  myservo1.attach(9);//确定引脚
  myservo2.attach(10);

  myservo1.write(90);//初始化舵机位置
  myservo2.write(20);
```

在初始化中，将机械臂的动作固定为：机械头摇上，机械手手钳张开。

目前在实际使用中只使用机械臂其中两个自由度，即机械手手钳合与机械手 Y 轴转动，下面以“抓取”动作为例分析。

```
void hand_catch()
{
  int angle=180;//为变量设置一个值，使程序能够正常编译

    for(angle=90;angle<175;angle++)//将头“摇下”，delay 控制速度
  {
    myservo1.write(angle);
    delay(15);
  }

  for(angle=20;angle<80;angle++)//夹子合拢，抓取物品
  {
    myservo2.write(angle);
    delay(15);
  }

  for(angle=175;angle>90;angle--)//将头“摇回”，delay 控制速度
  {
    myservo1.write(angle);
    delay(15);
  }
}
```

除抓取外，还有函数 hand_throw()（放下），这两个动作相互对应，使机械臂能够完成抓取、放置物品的动作。

3. 传感器及自主避障模块

在本方案中，设计了函数 Auto_mode()进行自主避障，在主程序里使用为循环调用 Auto_mode()，使运动状态不断更新。使用两个红外传感器模块，固定在小车车头。方向分别为一左一右，分别与 Arduino 的 4,13 引脚连接。实现运动的基本方式为，在 Auto_mode()内部根据情况调用运动函数，再延时一定时间让动作得到执行，程序结束。

在判断前方是否有障碍物时，若传感器模块均输出高电平（未检测到障碍物），则可认为前方无障碍物，调用函数 Run_forward()使小车前进；若检测到只有侧有障碍物，则调用函数 Turn_right()

或 Turn_left()是小车向另一方向进行一定的转向，当判定无障碍物时，即可继续前进；当左右都有障碍物时，则调用函数 Run_behind()后退一段，再调用 Turn_left()向左转一些来寻找出路。

```
void Auto_mode()//自主避障模式
{
    Frared_right = digitalRead(4);//接收右侧红外传感器数字信号
    Frared_left = digitalRead(13);;//接收左侧红外传感器数字信号
          if(Frared_right==0) //若右前方有障碍
        {
            if( Frared_left)//判断左前方有无障碍
              {
                Turn_left();//若右侧有障碍，左侧无，则左转
                delay(200);
              }
              else
              {
                 Run_behind();//若左右都有障碍，则后退并左转找路
                 delay(1000);
                 Turn_left();
                 delay(1000);
               }
        }
         if(Frared_right) //若右前方无障碍
        {
           if( Frared_left) //同时左前方无障碍
              {
                Run_foward();//前方无障碍，则前进
                delay(200);
              }
              else
              {
                 Turn_right();//否则，说明左前方有障碍，应右转
                 delay(200);
               }
         }
}
```

4.WIFI 通信

本方案根据 ITEAD 创易工作室[10]开发的 wifi 库（uartWIFI.h）改编进行通信。这里将 ESP8266 设置为 Server 模式（无线接入点）来使用，初始化时开启多连接模式，同时设置好要使用的端口号（这里为 8080），之后即可使用。接入点和名称也可修改，但为便于操作，注释掉了这一块。

```
/*****************WIFI 部分初始化*****************/
  _cell.begin(115200);  //设置波特率
  DebugSerial.begin(debugBaudRate);
  //_cell.println("AT+CWSAP=SSID,password,1,3");
```

```
  //delay(200);
   _cell.println("AT+CIPMUX=1");//开启 Server 服务
  delay(200);
  _cell.println("AT+CIPSERVER=1,8080");//开启端口
  delay(200);
```

我们截取了原库函数的一部分，封装为函数 receive_message(char *buf)（接收数据），send_message(byte id, String str)（发送数据），和 closeMux(void)及 closeMux(byte id)（关闭链接），这些即可满足本设计方案的需要。另，设计了函数 receiveorder()用来接收指令，在使用时，主程序通过调用 receiveorder()来不断检测是否有新的命令。

```
int receiveorder()//接收指令
{
int result=10;//指令的默认值，表示维持现状
char buf[100];
String text="";
int iLen = receive_message(buf);//检测是否有上位机发送数据
if(iLen>0) {
  text=buf;
  DebugSerial.println( text);
  DebugSerial.println( text.length());
  if(text.endsWith("stop")) result=0;          //指令“停止”
  if(text.endsWith("forward")) result=1;       //指令“前进”
  if(text.endsWith("behind")) result=2;        //指令“后退”
  if(text.endsWith("left")) result=3;          //指令“左转”
  if(text.endsWith("right")) result=4;         //指令“右转”
  if(text.endsWith("catch")) result=5;         //指令“抓取”
  if(text.endsWith("throw")) result=6;         //指令“放下”
  if(text.endsWith("fullauto")) result=7;      //指令“切换到模式 1”（全自动模式）
  if(text.endsWith("halfauto")) result=8;      //指令“切换到模式 2”（半自动模式）
  if(text.endsWith("noauto")) result=9;        //指令“切换到模式 3”（手动模式）
  if(result<10)
    send_message(chlID,"OK");//指令已接收，反馈给上位机和使用者
  else
    send_message(chlID,"ERROR");//指令无法识别，反馈给上位机和使用者
  return result;//输出命令，以此作为信号实际控制下位机
  }
}
```

5. 流程控制模块

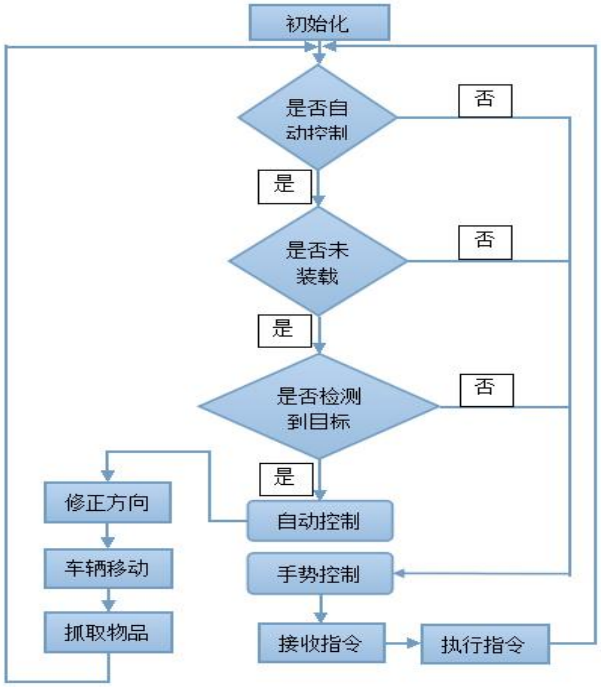
首先，在小车初始化时设置整型变量 mode_change 用来选择工作模式（初始值为 2，默认为半自动模式）。工作模式具体见下表：

mode_change 的值	1	2	3
工作模式	全自动模式	半自动模式	手动模式
特点	自己执行避障程序	在执行避障程序的同	完全由命令进行控制

		时，接受命令控制，且命令优先级高	
可执行指令	模式切换，停止	模式切换，停止 前进，后退，左/右转	模式切换，停止 前进，后退，左/右转 （机械臂）抓取，放下

```
void loop()
{
  if(mode_change==1){
    order_control1();
     //相关控制语句
}

     if(mode_change==2){
     order_control2();
     //相关控制语句
    }
```




```
     if(mode_change==3){
     order_control3();
    }
}
```

在程序运行过程中，首先判定是哪种工作模式，进入相应工作模式后，进入主控函数 order_controlx()(x=1,2,3)，在其中调用函数 receiveorder()判定指令，再根据指令控制下位机的动作，动作执行后程序结束一次迭代，进入下一次迭代，重复上述过程。

五、流程控制

流程控制如下图 7，8 所示。

图 7:上位机（电脑与 Kinect)流程图　　图 8：下位机（机器人）流程图

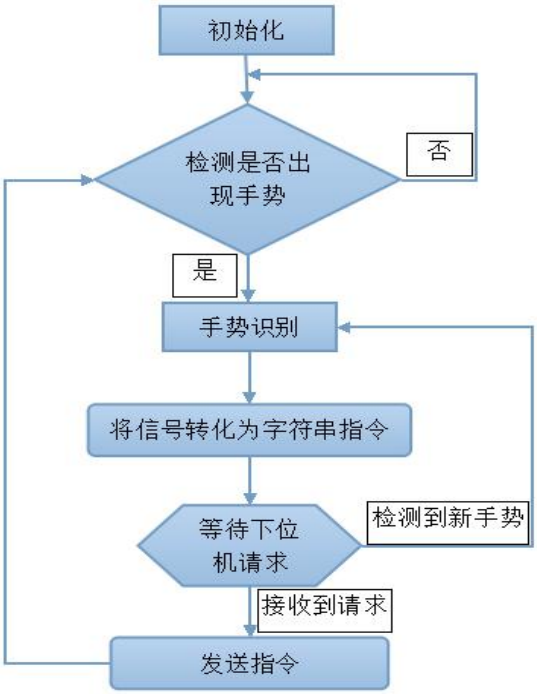
六、调试方案

在设计完成上位机与下位机的程序后，将程序分别在电脑，小车上独立运行，也将 ESP8266WIFI 模块连接在电脑上进行独立测试。在确认单独可以达到预定目标后。在小车上加装 ESP8266WIFI 模块，进行整体运行调试，中途因为接错插口烧坏了一个 ESP8266 模块，但是最终调试成功，证明了本设计方案的可行性。

下位机调试

一开始分别将电机驱动模块,机械臂模块在 Arduino 上测试,测试通之后,对函数 Auto_mode() 进行调试。在一开始调试 Auto_mode()时，发现小车在左侧有障碍物时仍然向左转，右侧有障碍物时则正常向左转。检查后发现，是在函数中存在逻辑判定错误，排除后可以正常避障。

之后在小车上加装 ESP8266WIFI 模块，使用手机 app《TCP 连接》测试是否能够通过无线命令控制下位机。在调整了波特率之后，Arduino 主控板成功通过串口与 ESP8266 进行通信，并能够实现通过手机直接控制下位机。WIFI 通讯方面，一开始是使用了 ITEAD 创易工作室开发的 wifi 库（uartWIFI.h），后发现命令传输速度较慢，最后将其类库进行修改以符合实际使用情况，能够实



现在手机发送指令 0.7 秒内下位机做出应答。

上位机调试

这次设计的虽然只是智能小车，但是前后却涉及到了 Kinect,Arduino 以及 WIFI 通信等不同模块，是对自己学习能力及综合运用能力的一次考验，也在这次的制作中学到了很多，为将来从事实际产品开发打下了良好的基础。

参考文献：

[1] yangecnu． [译]Kinect for Windows SDK 开发入门(十)：手势识别 上：基本概念 [EB/OL] ．
http://www.cnblogs.com/yangecnu/archive/2012/04/21/KinectSDK_GesturesDetection_part1_BasicConception.html，2012-04-21.
[2] Andrew_wx． C# Socket 简单例子（服务器与客户端通信） [EB/OL] ．
http://blog.csdn.net/andrew_wx/article/details/6629721，2011-07-24.
[3] 谢希仁． 计算机网络（第 5 版）［M］.北京：电子工业出版社，2007:279-280.
[4] 阿哲 wang． 减速马达工作原理 [EB/OL] ．
http://zhidao.baidu.com/link?url=ZiisH0SUymtxUk4IS6olFB6eeakWKl87zCCso9_hkdXnUZ5FvAGKdoB2pwsrR_JpKFyN3nWNweLEoQRKl1gVFq，2013-07-08.
[5] 远方． L298N 驱动模块的应用 [EB/OL] ．
http://blog.sina.com.cn/s/blog_80fb7ead0100sm30.html，2011-06-08.
[6] 喻菲． ESP8266 用户手册［M］.深圳：乐鑫信息科技 [M]，2014:6-8.
[7] Micbael Margolis（著），杨昆云（译）． Arduino 权威指南［M］.北京：人民邮电出版社，2015:1-4.
[8] tom． 我的红外传感器避障小车&&改进的过程 [EB/OL] ．
http://www.geek-workshop.com/thread-7311-1-1.html，2013-9-22.
[9] 卢光跃，吴涛． 基于 Arduino 控制的机械臂的运动与程序设计［J］． 机械制造，2014(3):52.
[10] ITEAD创易工作室． ESP8266 WIFI模块实现远程wifi控制 [EB/OL] ．
http://www.geek-workshop.com/thread-11266-1-1.html，2015-6-10.