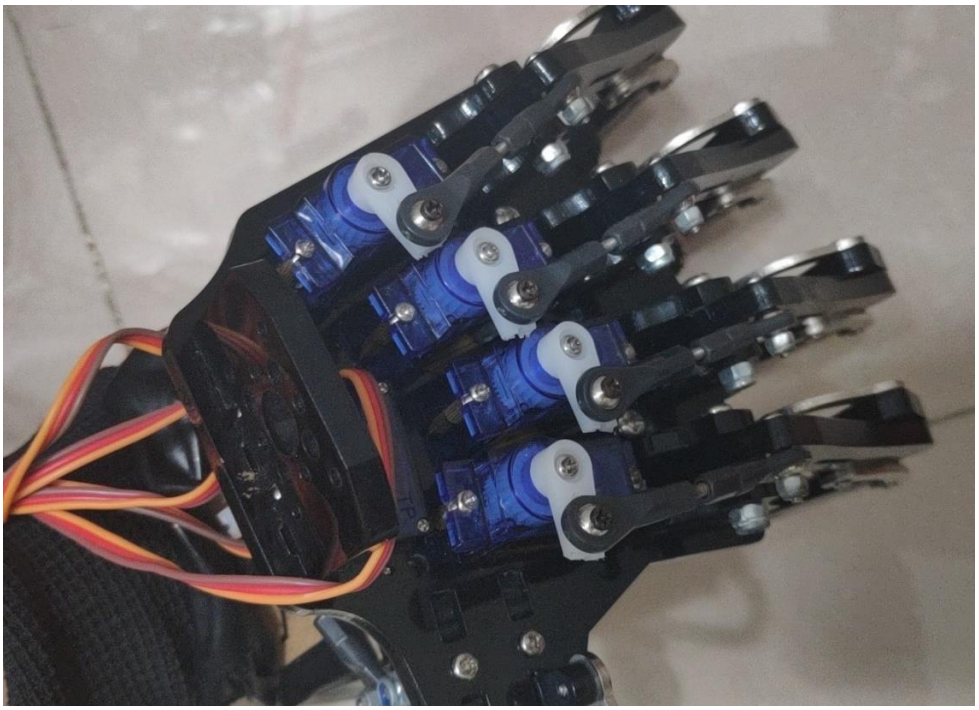
目录

1. 绪论

1.1 背景

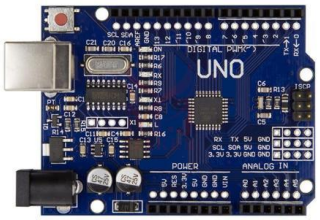
偏瘫（hemiplegia）又叫半身不遂，是指同一侧上下肢、面肌和舌肌下部的运动障碍，是急性脑血管病的常见症状。严重者常卧床不起，丧失生活能力。偏瘫的治疗时间长，需要持之以恒。除了药物等医学治疗外，更重要的是要进行康复训练。包括衣、食、住、行的日常生活基础动作训练。提高患者日常生活活动能力。目的是让患者逐渐适应个人生活、家庭生活、社会生活的种种需要。通过对关节活动度训练、肌肉牵伸训练和肌力训练，强化肌肉记忆，有利于康复。为了帮助偏瘫患者，希望能在可行的成本前提下制作辅助手部肌肉训练的机械手套。

1.2 成品图



2. 硬件材料列举

2.1 Arduino UNO：Arduino 主板，共 3 块，分别用于控制操作界面、机械骨骼和体感手套。



2.2 SG90 舵机：机械骨骼驱动装置，5 个，分别用于驱动手部五根手指。

2.3 LCD1602 屏幕：1 个，用于构建操作界面。

2.4 8*8 小键盘：1 个，用于作为交互控制的媒介。

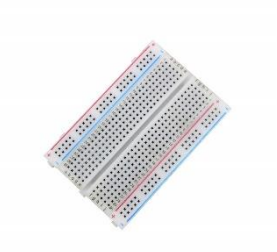
2.5 Hc-05：蓝牙，2 个。Arduino 间进行无线通信。

2.6 电位器：调节 LCD1602 屏幕。

2.7 光敏电阻传感器：2 个，用于制作体感手套，感应手部做出的动作。

2.8 杜邦线：若干条，用于连接电路。

2.9 面包板：1 块，用于连接电路。



2.10 Arduino 供电套装：1 个，用于作为 Arduino UNO 的电源。

2.11 Arduino Power Shield 锂电池扩展板太阳能充电供电 5V 电源：1 个，用于作为 Arduino UNO 的电源。

2.12 QS Safety PU2006e 劳保手套：1 双，左手用于制作连接机械骨骼的机械手套，右手用于制作体感手套。

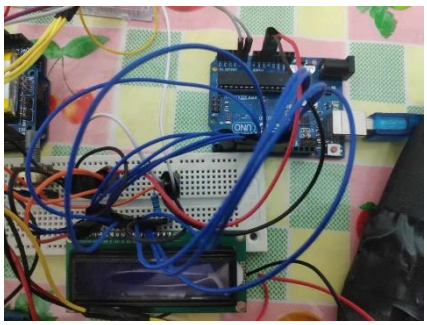
2.13　简易机械骨骼：1 个。用于作为机械手套的主体部分。

3. 总体介绍

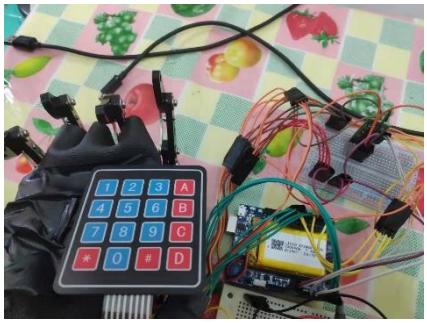
3.1　构成介绍

Arduino 极简机械手套由三部分组成：

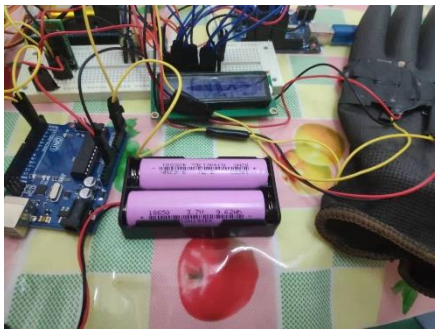
3.1.1 操作交互部分



3.1.2 机械骨骼部分



3.1.3 极简体感手套部分



3.2 各部分设计思路简介

3.2.1 操作交互部分

利用 LCD1602 屏幕进行显示，通过 8*8 小键盘实现控制，进行模式选择。

3.2.2 机械骨骼部分

利用五个舵机带动机械骨骼的手指进行运动。

3.2.3 极简体感手套部分

利用光敏电阻判断手部动作。

3.2.4 各部分连接方式

操作交互部分与机械骨骼部分采取有线软串口进行通信，机械骨骼部分和极简体感手套部分则通过 Hc-05 蓝牙进行无线通信。

3.3 功能介绍

接通电源后，LCD1602 进入交互模式。蓝牙成功配对后，根据提示进入主菜单。此时有两种模式可供选择：

3.3.1 训练模式

此模式共有 9 种训练方式：小键盘数字 1-7 对应机械骨骼驱动手指完成数字 1-7 的数数动作，小键盘数字 8 对应机械骨骼驱动手指完成握紧成拳头和松开为手掌的连贯动作，小键盘数字 9 对应机械骨骼驱动手指单独完成握紧成拳头的动作。(由于数字 5 用手指表示其实就是与松开为手掌的单独动作相对应)。

3.3.2 游戏模式

为增加趣味性，在训练模式的基础上添加游戏模式，通过右手极简体感手套与机器玩猜拳游戏。游戏开始时，机器与人类同时做出动作，由 LCD 屏幕显示得分。采取 5 局制，比完为止，LCD 屏幕会将最终输赢显示。此模式里又包括普通竞技模式和作弊模式。普通竞技模式中机器随机做出“锤子”“剪子”“布”的动作，而作弊模式中机器会根据人类的动作有针对性的做出“锤子”“剪子”或“布”的动作。

4. 机械结构说明

4.1 机械结构设计

对于机械骨骼驱动部分的机械结构，曾进行了如下多种设想：

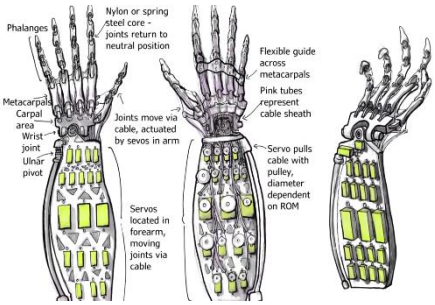
4.1.1 一舵机拉动

如图所示，最初设计中打算采用五根手指通过绳子统一连接在同一个舵机上。当舵机旋转时，拉动五个手指同时运动，进行握紧成拳头，松开成手掌的动作。

优点：简单、方便

缺点：完成动作单一

4.1.2 多舵机驱动



如图所示，此图为 YouTube 网站上一个关于仿生机械手视频中展示的设计图，可以看到：由于手部关节繁多，要做出完美的仿生机械手起码需要十几个舵机进行驱动。

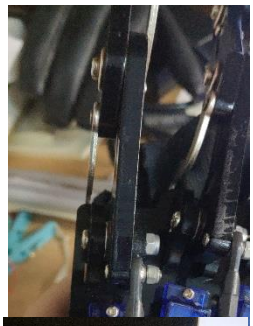
优点：精细，完善

缺点：无论是设计还是制作，短时间内是几乎不可能完成的。

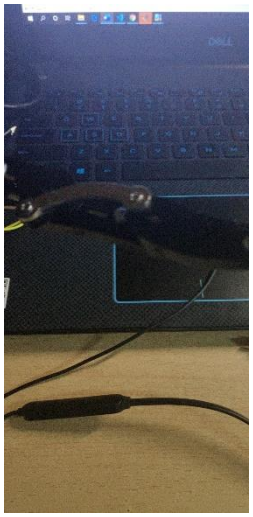
4.1.3 五舵机驱动

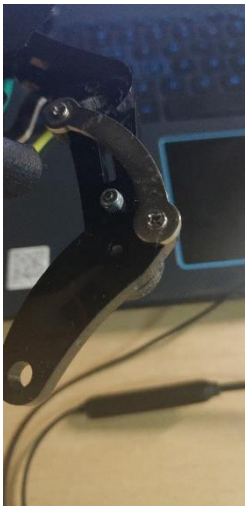
如图所示，此图采用五个舵机分别作为五个手指的驱动，以此实现对五根手指的分别控制，弥补了第一种方案只能共同运动的短板。每根手指上利用传动装置的连接使舵机只需带动一个关节运动即可传导至同根手指的其他关节，虽在灵活性上较第二种方案略差，但解决了第二种方案过于复杂的问题。因此，确定此为最终方案。

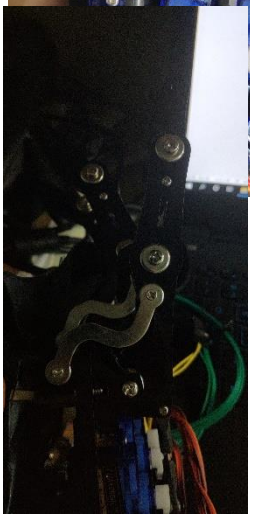
4.2 传动装置解析

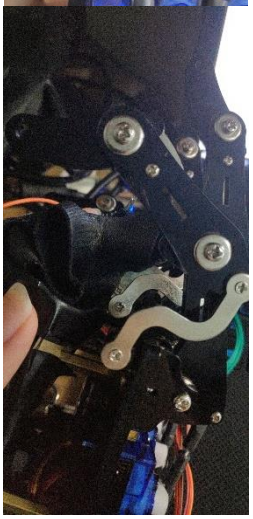


如图所示：舵机转动时，带动传动杆前进。而由于手指根部是固定在转轴上的，它将无法进行前后运动。所以只能随传动杆的前进而旋转弯曲。在手指根部的传动杆，则在根部旋转时，也随之会旋转，进而推动手指中部前进，由于手指中部是固定在转轴上的，它将无法随传动杆的







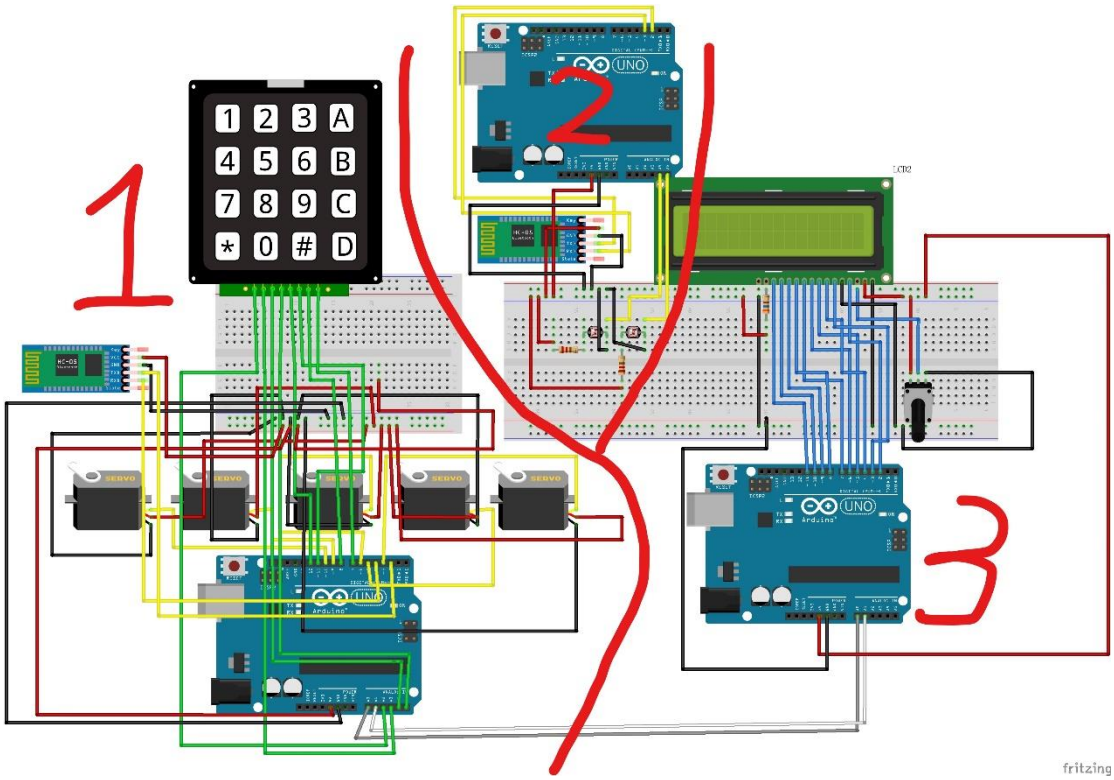


前进而前进，被迫发生旋转。同理，手指中部也存在着传动杆，在中部旋转时，传动杆转动，试图推动手指尖前进，同样因为旋转轴的存在而使得其发生旋转。

通过传动杆和旋转轴的作用，实现一个舵机便能控制手指同时完成手指屈伸过程中各个关节需要完成的动作。

5. 电路分析

各元器件之间的连接均采用并联方式。使用 Fritzing 画图工具绘制电路如下：



5.1 1 号区域具体连线说明

VCC 线 →+5V；
GND 线 →GND；
拇指舵机引脚 →3 号引脚；
食指舵机引脚 →5 号引脚；
中指舵机引脚 →6 号引脚；
无名指舵机引脚 →9 号引脚；
尾指舵机引脚 →10 号引脚；
Hc-05 蓝牙 RXD →4 号引脚；
Hc-05 蓝牙 TXD →2 号引脚；
8*8 小键盘连接引脚(从左至右)：
Col1 →A2 引脚；
Col2 →A3 引脚；
Col3 →A4 引脚；
Col4 →A5 引脚；

Row1 →7 号引脚；
Row2 →8 号引脚；
Row3 →11 号引脚；
Row4 →12 号引脚；

5.2 2 号区域具体连线说明

VCC 线 →+5V；
GND 线 →GND；
光敏电阻传感器 →A4 引脚、A5 引脚；
(由于未找到对应 Fritzing 图片，使用一个光敏电阻连接 10kΩ电阻代替)
Hc-05 蓝牙 RXD →3 号引脚；
Hc-05 蓝牙 TXD →2 号引脚；

5.3 3 号区域具体连线说明

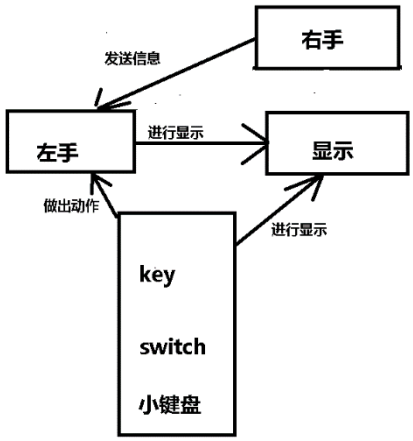
VCC 线 →+5V；
GND 线 →GND；
LCD1602 屏幕：
VSS →GND；
VDD →leg2(电位器)；
V0 →wiper(电位器)；
RS →2 号引脚；
R/W →GND；
Enable →3 号引脚；
DB0 →4 号引脚；
DB1 →5 号引脚；
DB2 →6 号引脚；
DB3 →7 号引脚；
DB4 →8 号引脚；
DB5 →9 号引脚；
DB6 →10 号引脚；
DB7 →11 号引脚；
LED+ →+5V；
LED- →GND；
电位器
leg1 →GND；

5.4 1 号和 3 号区域具体连线说明

A0 →A0 引脚；
A1 →A1 引脚；

6. 程序代码分析

6.1 简要分析



整个程序包括 4 个文件(左手、右手、显示屏、蓝牙)：总代码大约共 608 行。

左手，右手，显示屏三个文件之间关系设想如下图：

程序运行时，通过显示界面提示对 8*8 小键盘进行操作。由于引脚不足，8*8 小键盘连接在控制左手机械骨骼的 Arduino UNO 板上，两部分之间利用有限软串口进行数据传输通信。因此，按下 8*8 小键盘上的按钮后，读取对应字符传递至显示部分对 LCD1602 屏幕界面进行更新，同时实现对左手机械骨骼的控制。使用右手时，通过蓝牙传输数据至左手。

6.2 详细分析

6.2.1 左手代码

左手代码需要实现以下功能：

1、蓝牙通信

蓝牙配对详见 6.2.4 调试代码。

包括头文件：#include<SoftwareSerial.h>

定义蓝牙接口：Pin2 为 RX，接 HC05 的 TXD；Pin4 为 TX，接 HC05 的 RXD：SoftwareSerial BSlave(2, 4);

定义函数：read(); 进行读取数据，核心代码如下：

```
void read()
{
    if (BSlave.available()) {
    data = BSlave.read();
    }
    run = 1;
}
```

可以看到，读取数据的代码非常简单。但是在处理这一模块的时候，有很大的困难：读取的数据会出现不稳定的情况。具体出现过这么几个问题：数据连缀(仅需接收一个字符，接收结果变为这个字符连续出现的长短不一字符串)、数据不正确(出现非传输数据)。经分析发现，问题发生在后续键盘控制方面的设计导致其必须满足在一次程序中不断地在某一段代码中进行循环，并同时能够利用蓝牙读取数据。首先想到的解决方法是：在这一段代码内部插入 read();函数。这种方法虽然解决了以上两个问题，却又带来了新的问题：数据更新缓慢，无法在短时间内更新数据。根据这些问题，设想：在程序执行过程中，在极短时间内中断程序，读取数据，继续运行程序。因此，决定使用<MsTimer2.h>中的中断函数，核心代码如下：

```
MsTimer2::set(1,read);
        if(!run)
            MsTimer2::start();
        else
```

```
                {
                    MsTimer2::stop();
            }
```

利用这种方式，使得读取蓝牙数据更加准确，成功解决了以上的几个问题。

2、8*8 小键盘控制

包括头文件：#include<Keypad.h>

Keypad keypad = Keypad(makeKeymap(keysboard), rows_pin, cols_pin, rows_num, cols_num);//创建键盘

核心变量：key，利用 getkey();函数读取小键盘所按下的键。

定义函数：check(); check2(); backA(); backB();

check();函数：对应游戏模式的竞技模式。利用 random 函数产生 1-3 之间的随机数，并指定对应左手机械骨骼的动作。将蓝牙接收的数据与随机数进行比较，产生赢、和、输的结果，发送至显示部分进行计分。

核心代码如下：

```
    void check()
    {
            read();
            int compare = random(1,4);
                switch(compare)
                {
                    case 1:cloth();delay(1000);break;
                    case 2:scissors();delay(1000);break;
                    case 3:stone();delay(1000);break;
                }
                if(data-'0'==compare)
                {
                    key = '#';
                    i--;
                }
     else
  if((data=='1'&&compare==3)||(data=='2'&&compare==1)||(data=='3'&&
  compare==2))
                    key = 'C';
    else
if((compare==1&&data=='3')||(compare==2&&data=='1')||(compare==3&
&data=='2'))
                    key = 'D';
}
```

check2();函数：对应游戏模式的作弊模式。直接读取蓝牙接收的数据，产生必胜的对应数据，发送给显示部分并指定左手机械骨骼做出动作。

核心代码如下：

```
    void check2()
    {
            read();
                if(data=='1')
```

```
                {
                    scissors();
                    delay(1000);
                    key = '*';
                }
                else if(data=='2')
                {
                    stone();
                    delay(1000);
                    key = '*';
                }
                else if(data=='3')
                {
                    cloth();
                    delay(1000);
                    key = '*';
                }
        }
```

backA();函数：对应游戏模式的竞技模式，由于游戏采用五局打满制，所以每次执行代码时利用变量 i 进行计数，超过 5 次后结束对一段代码的循环。核心代码如下：

```
    void backA()
    {
        if(i<=5)
        {
            key = 'A';
            count = 3;
            i++;
        }
        else
        {
            key = '0';
            count = 0;
        }
    }
```

backB();函数：与 backA();函数同理

3、软串口通信

包括头文件：#include<SoftwareSerial.h>

定义软串口：PinA0 为 RX，接另一块 Arduino 的 TX；PinA1 为 TX，接另一块 Arduino 的 RX：SoftwareSerial Connect(14,15);

定义函数：send(); 此函数意在发送核心数值 key 到另一块 Arduino UNO。考虑到下面定义的接收函数会将接收到的字符转化为整型变量，所以利用 switch 函数将小键盘按钮映射到整型数字，再通过 itoa()函数转换为字符串进行发送。代码如下：

```
    void send()
    {
        if(send_data)
```

```
            {
                itoa(send_data,result,10);
                Connect.write(result);
                delay(100);
            }
        }
```

4、手部运动控制

包括头文件：#include<Servo.h>

定义五个舵机对应五根手指：

```
Servo thumb;//拇指
Servo index;//食指
Servo middle;//中指
Servo ring;//无名指
Servo little;//尾指
```

定义函数：

```
one();//数字 1
two();//数字 2
three();//数字 3
four();//数字 4
five();//数字 5
six();//数字 6
seven();//数字 7
fist();//拳头
hand();//手掌
stone();//石头
scissors();//剪子
cloth();//布
```

手部运动控制是较为简单的舵机控制，就不一一赘述。

5、多层次控制设计

为满足设想中的多种模式内嵌套多种模式，实现多层次控制设计，定义层数变量 count。层数构建如下：

第零层：主菜单

第一层：训练模式、游戏模式

第二层：训练模式下各种动作，游戏模式下竞技模式与作弊模式。

第三层：竞技模式与作弊模式下的判断。

第四层：竞技模式与作弊模式下的发送数据

为实现判断能够连续执行 5 次，利用 goto 函数不断循环 switch 函数，核心代码如下：

```
    else if(count==3)
        switch(key)
        {
            case 'A':send_data = 10;count++;check();goto star;break;
            case 'B':send_data = 11;count++;check2();goto star;break;
            case '0':send_data = 14;count=1;hand();break;
```

```
        }
    else if(count==4)
        switch(key)
        {
            case '*':send_data = 12;send();backB();goto star;break;
            case '#':send_data = 16;send();backA();goto star;break;
            case 'C':send_data = 12;send();backA();goto star;break;
            case 'D':send_data = 13;send();backA();goto star;break;
            case '0':send_data = 14;count=1;hand();break;
        }
```

由于通常是在每次程序结束后发送数据，所以在循环程序时要发送数据则 补上 send 函数

6.2.2 右手代码

右手需要完成以下功能：

1、控制两个光敏电阻

如图：两个光敏电阻判断右手做出的猜拳动作，两个遮光为石头，一个遮光为剪刀，两个都未遮光为布 定义两个变量，利用 analogRead()读取模拟串口的数据。

2、蓝牙发送数据

包括头文件：#include <SoftwareSerial.h>

Pin2 为 RX，接 HC05 的 TXD；Pin3 为 TX，接 HC05 的 RXD：SoftwareSerial BT(2, 3);

蓝牙发送数据代码与前面通过软串口发送数据的 send 函数原理相同，较为简单，不再赘述。

6.2.3 显示屏代码

显示屏代码需要实现的功能有：

1、接受软串口数据

包括头文件：#include <SoftwareSerial.h>

定义软串口：PinA1 为 RX，接另一块 Arduino 的 TX；PinA0 为 TX，接另一块 Arduino 的 RX： SoftwareSerial mySerial(15,14);

定义函数：read();核心代码如下：

```
void read()
{
    //读取数据
    while (mySerial.available() > 0)
    {
        comdata += char(mySerial.read());
        delay(2);
    }
    rdata = comdata.toInt();
    if (comdata.length() > 0)
        comdata = "";

}
```

定义一个 String 类型字符串，不断读取数据。然后通过 toInt()，转化为整型变量。

2、显示功能

包括头文件：#include <LiquidCrystal.h>

LiquidCrystal LCD (2,3,4,5,6,7,8,9,10,11);

利用 if 条件语句对显示功能进行区分。采用左手代码中所提到的多层次控制设计，定义整型变量 flo 实现多种模式嵌套多种模式

3、计数与判断输赢

根据软串口读取的数据，判断输赢。定义两个整型变量对两个玩家进行计数。当整型变量之和等于 5 时，判断两个变量的大小关系，从而判断输赢。

6.2.4 蓝牙调试代码

蓝牙需要建立连接才能进行通信。

需以下条件：

1、两台设备上设置相同的波特率

2、确保密码相同

3、找到地址

需进行以下步骤：

1、编译代码，核心代码如下：

```
void loop() {
  if (Serial.available()) {
    val = Serial.read();
    BT.print(val);
  }

  if (BT.available()) {
    val = BT.read();
    Serial.print(val);
  }
}
```

程序非常简单，只需让蓝牙与电脑能够实现串口通信即可。

2、进入 AT 模式

首先，将 Arduino 断电，然后按着蓝牙模块上的黑色按钮，同时让 Arduino 通电，直到蓝牙模块指示灯按照 2 秒一下的频率闪烁，表明蓝牙模块已经正确进入 AT 模式。

3、进行蓝牙基本参数设置

打开 Arduino IDE 的串口监视器，选择正确的端口，将输出格式设置为 Both: NL & CR ，波特率设置为 38400 ，可以看到串口监视器中显示 BT is ready! 的信息。然后，输入 AT ，如果一切正常，串口显示器会显示 OK。接下来，即可对蓝牙模块进行设置。

设置 hc-05 从模块: BSlave：

AT+ORGL(恢复出厂设置)

AT+NAME=BSlave(设置蓝牙名称)

AT+PSWD=3180100199(设置蓝牙匹配密码)

AT+ROLE =0(设置蓝牙为从模式)
AT+CMODE=1(设置蓝牙为任意设备连接模式)
AT+ADDR?(查询模块地址)
设置 hc-05 主模块：BMaster：
AT+ORGL(恢复出厂设置)
AT+NAME=BMaster(设置蓝牙名称)
AT+PSWD=3180100199(设置蓝牙匹配密码)
AT+ROLE =1(设置蓝牙为主模式)
AT+CMODE=1(设置蓝牙连接模式)
AT+ADDR?(查询模块地址)

4、连接蓝牙
BSlave：
设置串口波特率：AT+UART:38400,0,0
绑定 BMaster 的地址：AT+BIND=<BMaster 的地址，此处地址间隔用“,”而不是“:”>
BMaster
设置串口波特率：AT+UART:38400,0,0
设置查询模式：AT+INQM=1,5,30(按 RSSI 模式搜索，超过 5 个蓝牙设备响应则终止查询，设定超时为 48*1.28=61.44 秒)
开始查询：AT+INQ
试探连接：AT+PAIR=<BSlave 的地址>,<连接超时>
连接：AT+LINK=<BSlave 的地址>

5、完成连接
重启两个模块，可以看到两个模块会自动连接。(标志为 hc-05 灯：每 2 秒共同闪烁)

7. 总结和展望

7.1 测试方式与结果

测试分为依照模式分类进行。训练模式下对含有的不同模式进行切换，观察机械骨骼所做出的对应动作是否正确。游戏模式下戴上极简体感手套分别运行竞技模式和作弊模式，观察左手机械骨骼的动作与显示屏的计数是否正确。

测试结果：各模式运行正常，详见附件视频。

7.2 存在问题

现阶段作品还有待完善，存在着些许问题：

1、机械骨骼的灵活性还有待提高。
2、机械骨骼作为驱动的舵机马力不足，尚不能完全带动手指进行大幅度运动。
3、某些时候，在进行游戏模式时，蓝牙连接不稳定，导致左手机械骨骼的动作较右手延迟较长。

7.3 作品总结

极简机械手套采用五个舵机作为驱动装置，以蓝牙的无线连接和软串口的有线连接在 Arduino UNO 之间进行通信，配合 LCD1602 屏幕进行交互操

作，8*8 小键盘进行控制。完成了对多种训练模式与多种游戏模式的实现，使得在使用时既能有实际的帮助，又充满了趣味性。

7.4 未来展望

今后可以从以下几方面进行改进：

1、机械骨骼结构设计精细化。

2、驱动装置优化。

3、使用更加高效稳定的无线通信方式。

4、硬件外观整合美化。

5、软件功能实现多样化。