创新设计报告

课　程　名　称：　电子与计算机系统工程实训

设计项目名称：基于体感和 Android 控制的智能小车

专　业　班　级：_____

姓　　名：_____

学　　号：_____

指　导　教　师：_____

完　成　时　间：　2024　年_____月_____日

摘要

随着智慧农业的发展，为了提高农业生产效率，减少人力成本，我们设计并实现了一款基于体感和 Android 控制的智能小车。主要应用场景在大棚或农场等大面积的人造环境中，用来检测诸如光照、温度等影响植株生长的核心因素，便于人们及时发现问题调整。

该小车以一块 STC15F2K61S2 单片机为核心，配备超声波传感器、HC-06 蓝牙模块、L9110S 两路电机驱动板、TT 电机、热敏电阻、光敏电阻、蜂鸣器等设备，使其能够在复杂环境下按照控制运动，并执行如植保巡检、环境监测等任务。通过三轴加速度传感器，小车能够识别手势并据此控制移动。同时通过 Android 应用实现远程控制，用户可以实时监控环境温度和光照条件，遇到不符合条件的环境会在地图上进行标注，并在必要时进行干预。本项目旨在通过自动化工具减轻农民负担，提高农业生产的智能化水平。

目前我们已经实现了基本功能，小车前进、转弯、变速丝滑；蓝牙控制灵敏，软件交互性好，使用方便。但是受限于红外通信不稳定，手势识别不太灵敏；此外，电机动力较差，当小车自身重量增多之后速度明显下降。

目录

1 绪论

1.1 选题背景

近些年来，国家大力推广智慧农业，推动农业领域朝着全面数字化、网络化和智能化的转型，旨在解决传统农业生产中存在的生产效率低、人力成本高的情况。在生产过程中，植保巡检、检测植株生长环境状态和数据收集分析是必不可少的环节。在传统农业中，这类需要频繁重复进行的任务极其耗费人力物力。

为了把人从简单重复的任务中解脱出来，我们研究制作了一辆以 STC15F2K61S2 单片机为核心的智能小车。小车安装有超声波传感器、HC-06 蓝牙模块、L9110S 两路电机驱动板、TT 电机等设备，使得小车具备高机动性和灵活性。可以通过 Android 和姿势控制小车的方向和速度、利用超声波传感器和二轮差速实现避障，小车因此可以在复杂的农田环境中自由行驶，避免碰撞；同时通过热敏电阻和光敏电阻检测植株生长环境的情况；将收集的数据汇总上传服务器，用户可以通过手机应用界面查看。节省生产过程中的人力成本。

1.2 任务与功能

① 利用三轴加速度传感器，识别当前手势，并根据手势控制小车运动。例如，抬手控制小车后退，按下按钮增大小车速度。

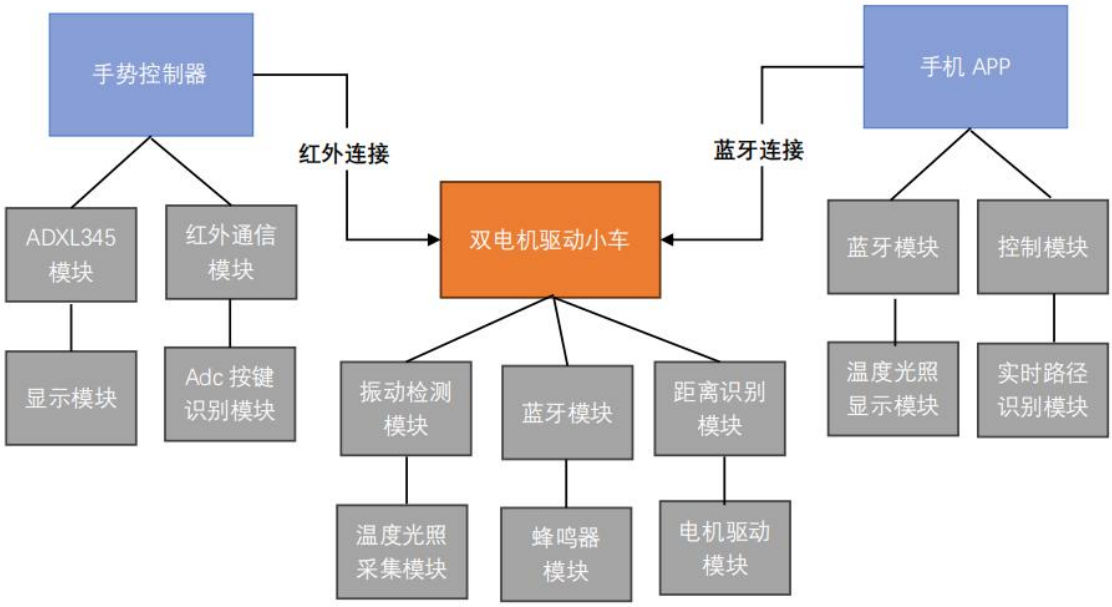
② 在 Android 手机上开发一款应用，可以与小车上的 HC-06 蓝牙模块匹配连接，向小车发送运动方向和速度指令，可鸣笛。同时手机会接收到小车发来的当前环境的温度和光照值，并显示小车在当前农场中的位置。若是当前环境不符合要求，就会在地图上进行标注。

③ 小车运动分为两种模式：BT 蓝牙模式和 IR 红外模式。BT 蓝牙模式下，小车和 Android 手机连接，发送当前环境的温度和光照，并接收速度方向指令。IR 红外模式下仅仅接受速度方向指令。此外，小车在遇到撞击时会停机；前方物体距离小车太近会鸣笛；按下按钮改变小车的运动模式和速度；数码管显示当前模式、方向和运动时间。

1.3 本文结构

引言部分简要介绍了研究背景、目的以及研究的重要性和意义。第二章系统总体设计与分工详细描述了整个系统的架构设计，包括各个模块的功能划分及其相互之间的协作关系。确保项目的有序进行。第三章硬件电路原理深入探讨了系统硬件部分的设计理念和工作原理，涵盖了所有必要的硬件组件和技术规格，为读者提供了硬件层面的全面理解。在第四章软件设计与实现中，我们将注意力转向软件方面，详细介绍软件架构的设计思路以及关键算法的实现细节。此外，还会讨论软件开发过程中遇到的主要挑战及其解决方案。第五章则聚焦于系统的实际测试情况，包括测试环境的搭建、所采用的测试方法以及从测试数据中得出的结果分析。这部分不仅验证了系统功能的正确性，还对其性能进行了评估。最后一章是对全文内容的归纳总结，并对未来的研究方向或可能的应用扩展提出了建议。此章节也反思了项目过程中的经验和教训，以供后续研究参考。

2 系统总体设计与分工



2.1 双电机驱动小车

双电机驱动小车车身是两块透明亚克力板，中间有些许小孔方便铜柱支撑；左右两个 TT 电机各自驱动一个橡胶轮，前后分别一个万向轮保持平衡；控制核心是一块 STC-B 学习板，EXT 口接着一个 HC-06 蓝牙模块、SM 口接着 L9110S 电机驱动板、RS485 连接另一块 STC-B 板并附带一个超声波模块；除此之外，一个 5000mA 电池通过 USB-A 拓展口向两个 STC-B 板供电。

电机驱动要将 P4 设置为推挽模式，控制 P4.1、P4.2、P4.3 和 P4.4 口的输出，每两个通过 L9110S 驱动板驱动一个 TT 马达，可以控制正转反转；蜂鸣器模块用来警报，当超声波传感器检测到有动物或人离得十分近时，鸣笛提醒。也可以主动控制鸣笛;温度光照采集模块会根据热敏电阻和光敏电阻的值每隔 1s 发送一次当前环境的温度和光照值，若是不符合要求会在手机上标记；振动检测模块适用于小车来不及刹车撞上墙这种情况，通过撞击产生的振动识别，并将停止小车轮子转动，防止造成二次伤害；距离识别是通过超声波传感器，当前方物体距离小车在 15 里面以内就会报警；蓝牙模块用来与 Android 手机通信，接收其发送来的方向和速度信息，并将温度和光照信息传递过去。

这其中的重难点有很多：首先是硬件选择与购买，要考虑到农场的现实因素，选择小巧、灵活的车型，同时传感器要考虑是否能和 STC-B 板子的扩展口兼容，扩展口是否充足；其次是车身搭建，考虑到经济原因，我们选择的都是价格低廉、性价比高的产品，质量难免不佳。在搭建小车时，要保证四个轮子水平，两两对称。车上的板子和模块应该尽量居中，保持车身平衡。接着是代码编写。近九成的单片机代码都在小车的板子上，要保证代码模块化，写好变量名和注释，注意彼此之间的逻辑关系。最后是 BUG 调试，keil 的编译器纠错功能很差，我们必须保持专注，确保逻辑正确。同时不能掉入认知陷阱——“只谈代码、不谈硬件”，要检查每一个硬件功能是否完好。

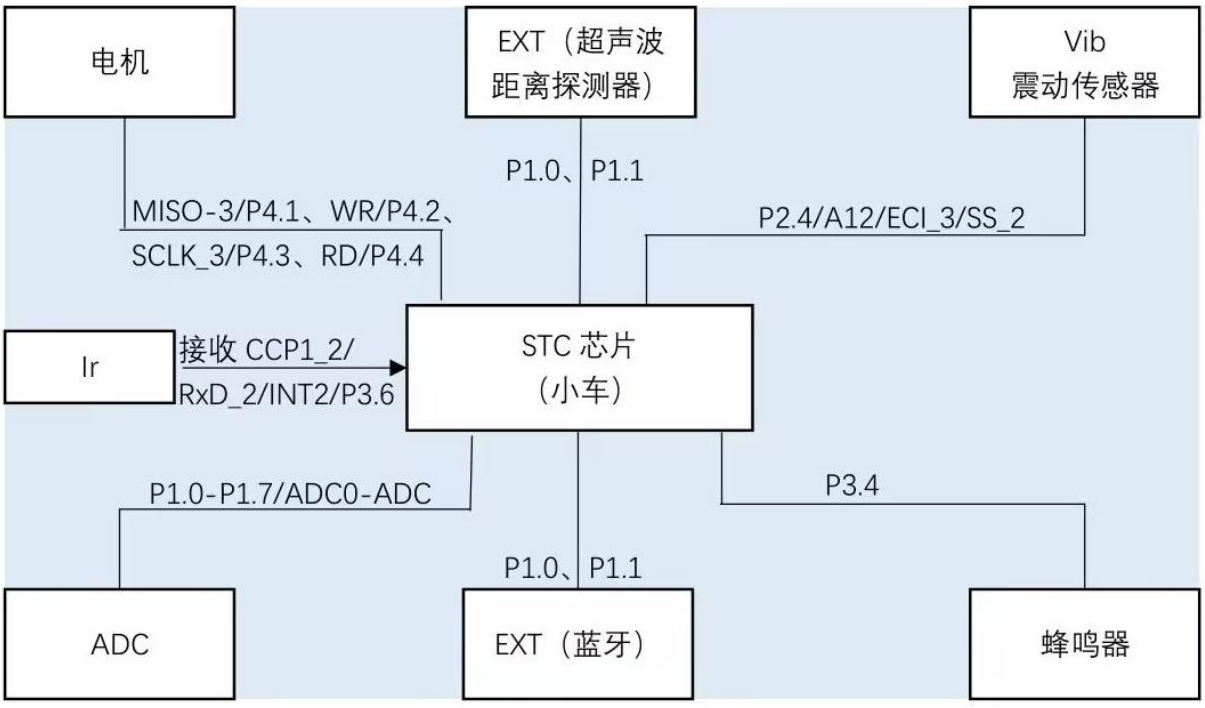
2.2 蓝牙控制 APP

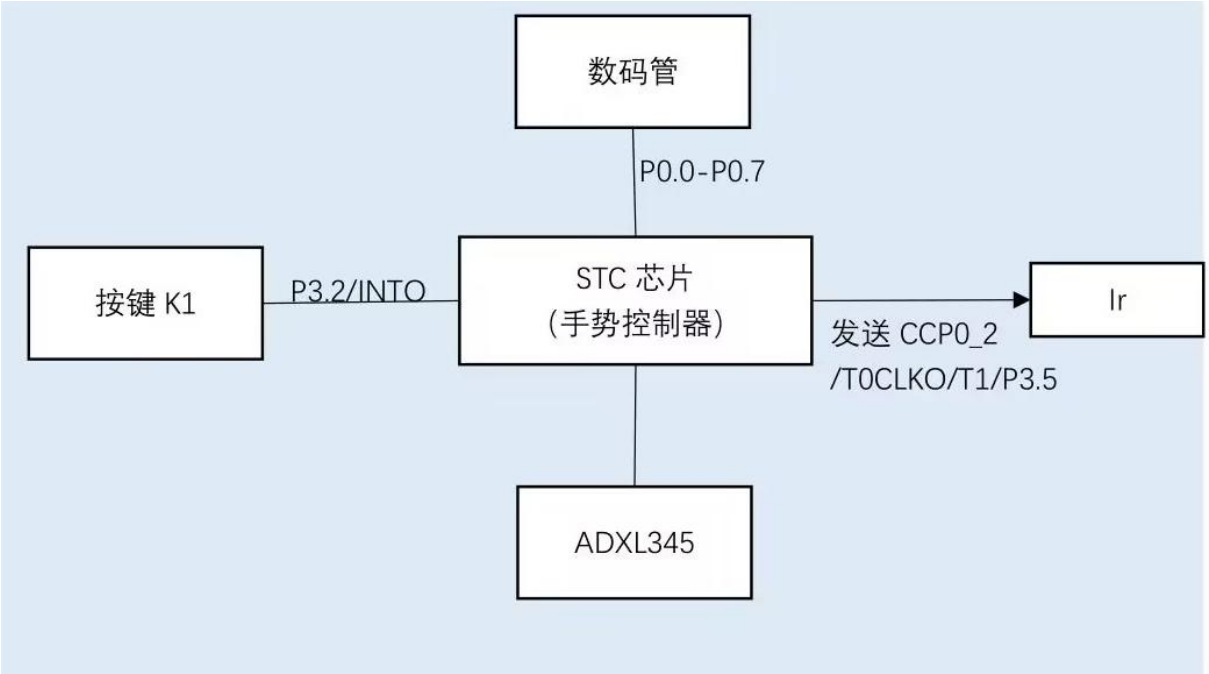
各模块与要实现的具体功能：
（1） 蓝牙模块：通过自购的 HC06 蓝牙模块实现手机与小车之间的连接，实现数据互传；实现在手机上点击“蓝牙选择”按钮后弹出可用蓝牙列表以连接特定蓝牙，点击“断开”按钮后断开当前连接的蓝牙；
（2） 温度光照显示模块：实现蓝牙接收小车传递的温度光照信息，显示在屏幕上；
（3） 控制模块：实现鸣笛、前进、后退、左转、右转、停车、速度控制（通过滑动条实现）功能；
（4） 实时路径识别模块：实现对小车当前位置的显示与已走过路径的记录

2.3 手势识别控制器

各模块与要实现的具体功能：
（1） ADXL345 模块：
通过借用带 ADXL345 模块的学习板，实时检测 xyz 三轴方向加速度，通过三角函数计算偏移角度，并根据角度偏移来实现手势识别，识别到手部动作后会产生相应指令。
（2） 导航按键模块：
通过上下按键调整车速，按中间键可进行急停，产生事件后会对应产生相应指令。
（3） 红外通信模块
指令通过红外通信模块发送给小车

3 硬件电路原理





3.1 振动传感器

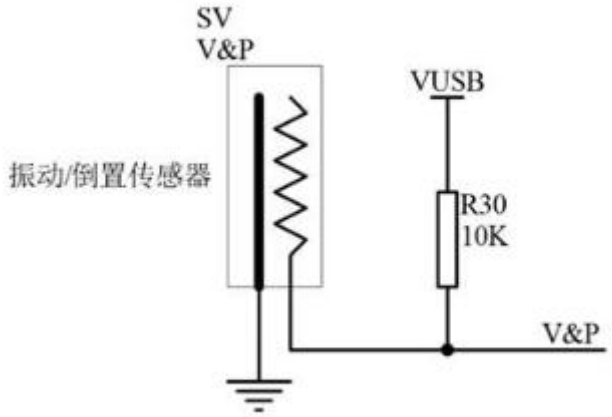


图 1 无源蜂鸣器电路原理图

本实验板中使用的振动传感器是一种简单的器件，管内有一跟固定的导线，在这根导线的周围有另一根较细的导线以螺旋状环绕它。可以想象为一个弹簧旁边有一跟导线。在不振动时，两根导线不会相碰，一旦振动发生，两根导线就会短接。所以我们只需判断导线是否短接了，就可以知道振动是否发生。

3.2 EXT 口和串口通信

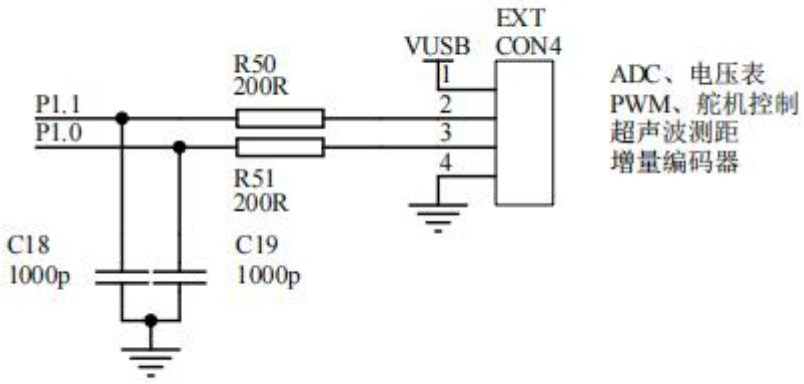
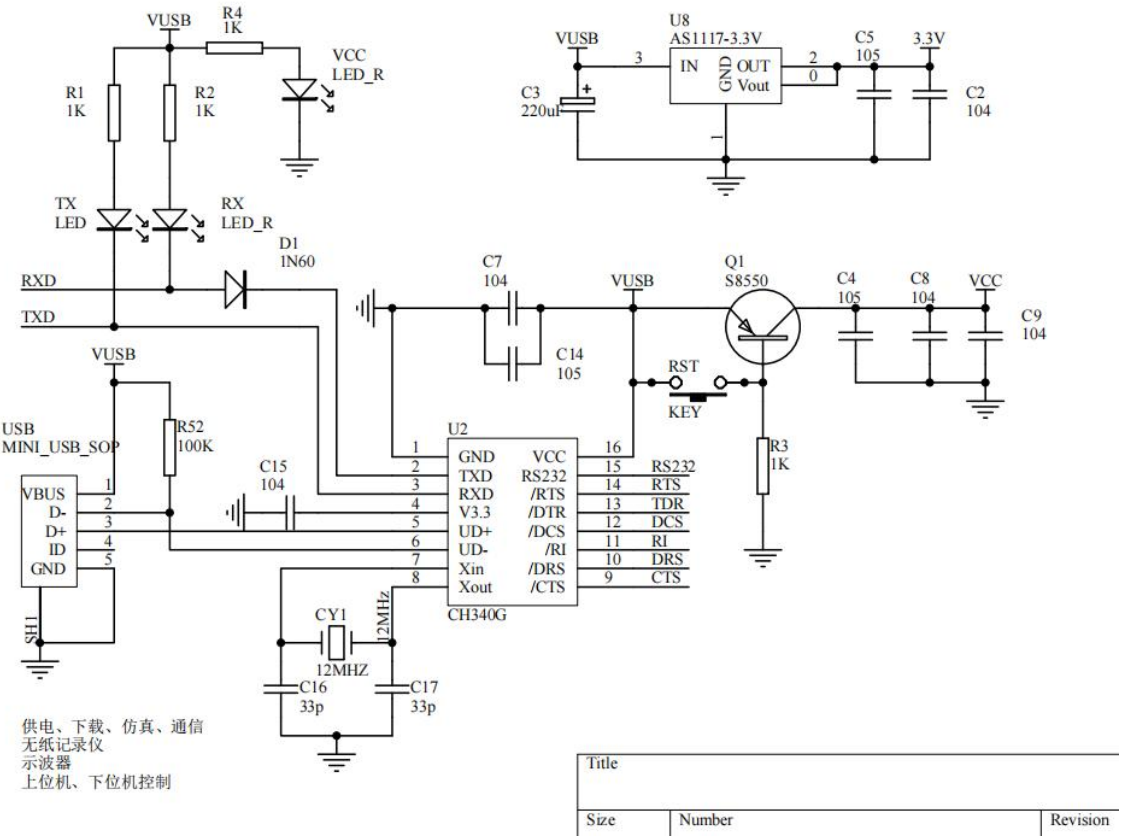
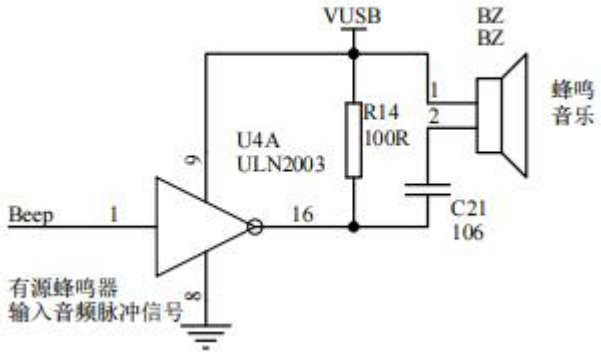


图 2 EXT 口电路原理图



单片机集成了 USB 转串口模块，对应使用 RXD 线接收数据，用 TXD 发送数据。每个串口由 2 个数据缓冲器（相互独立 1 收 1 发）、一个移位寄存器（一字节数据一位一位发送出去）、一个串行控制器和一个波特率发生器（这个比较重要，结合相关的定时器）组成。对应发送、接收数据完成（RI、TI 硬件置 1）都会触发串口中断，但是无法确定是哪个触发的，所以在串口中断中我们要判断是接收数据产生的中断还是发送数据产生的中断，对于发送数据产生的中断，我们要软件将 TI 清 0，并将数据就绪标志清 0，允许下一字节数据发送，发送数据函数中通过 while 循环，等待发送数据准备就绪，完了将就绪的数据复制给 SBUF；对于接收数据产生的中断，我们要软件将 RI 清 0，并从 SBUF 中读取数据。

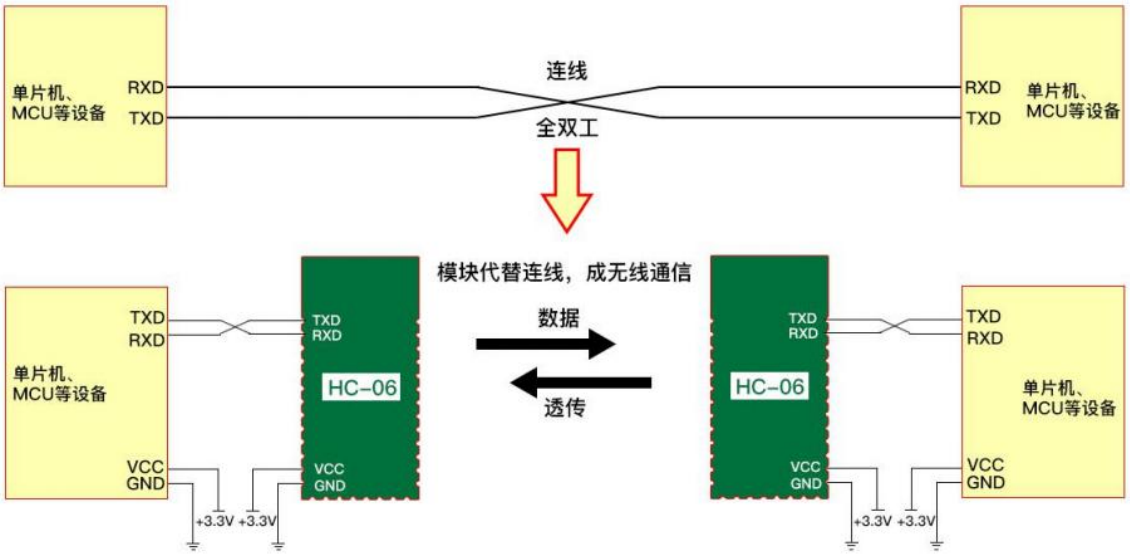
3.3 无源蜂鸣器



蜂鸣器分为有源蜂鸣器和无源蜂鸣器，这里的源特指振荡源；有源蜂鸣器直接加电就可以响起，无源蜂鸣器需要我们给提供振荡源。理想的振荡源为一定频率的方波。

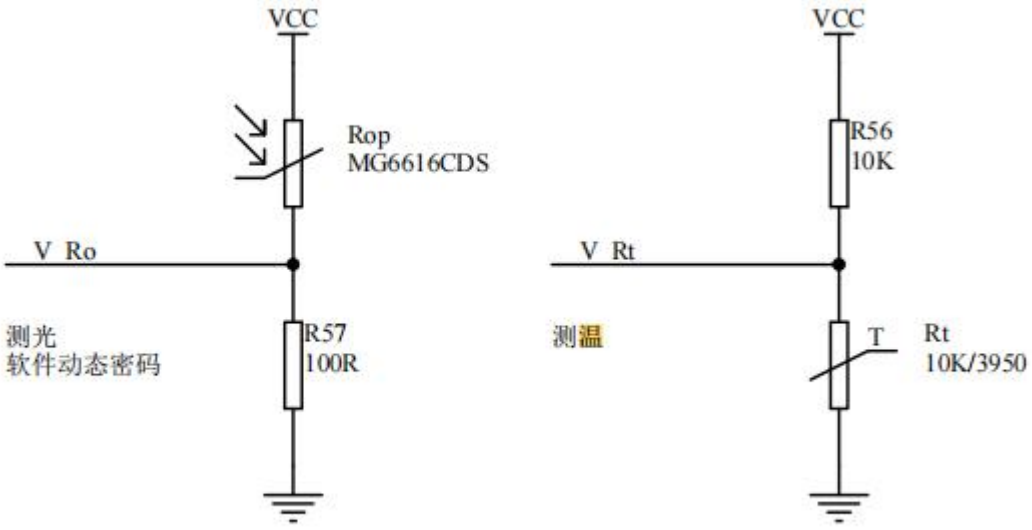
无源蜂鸣器只需改变 Beep 端口的电平,产生一个周期性的方波即可使蜂鸣器发生声音，不同的频率发出的声音不同。其中，ULN2003 是一个功放，用于放大电流。电阻 R14 和电容 C21 是用来保护电路的。若人为将 Beep 端口的电平一直置为高电平，在没有保护电路的情况下，容易烧毁电路，但即使有保护电路也应该注意不要将 Beep 端口长时间置于高电平，这对器件也是有一定损害的。当我们给无源蜂鸣器一个周期性的方波即可让其振动发声.输入方波放频率发生改变，蜂鸣器振动的频率就会发生改变，从而发出不同的音调。

3.4 EXT 口和 HC-06 蓝牙模块



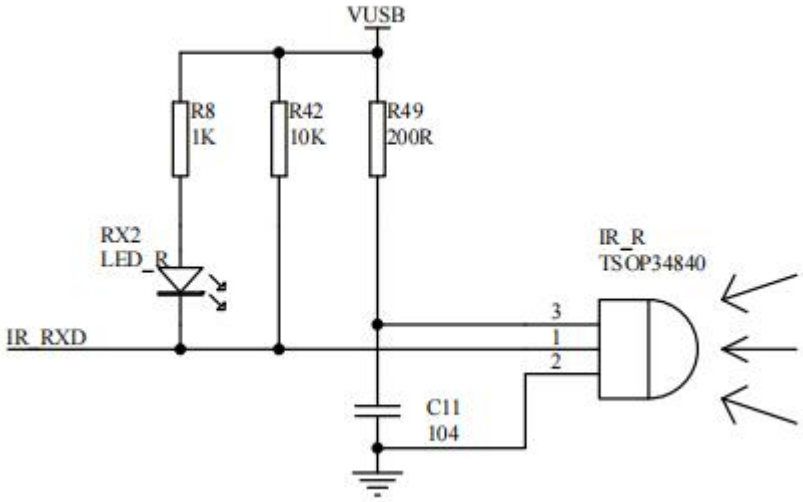
如上图所示，HC-06 模块用于代替全双工通信时的物理连线。左边的设备向模块发送串口数据，模块的 RXD 端 口收到串口数据后，自动将数据以无线电波的方式发送到空中。右边的模块能自动接收到，并从 TXD 还原最初左边设备所发的串口数据。从右到左也是一样的。

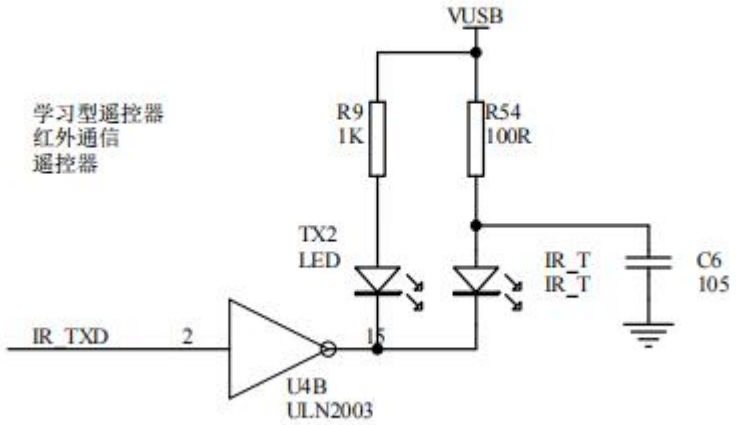
3.5 热敏电阻和光敏电阻



热敏电阻随温度呈线性变化，光敏电阻电流随光强线性变化。通过 AD 采集光敏电阻和热敏电阻的输出值，输出对应的 AD 值，光照值直接为 AD 值，而采集的温度 AD 值，首先把 10 位转换成 8 位 AD 值，然后再通过查找对应的表来获取温度。

3.6 红外传感器



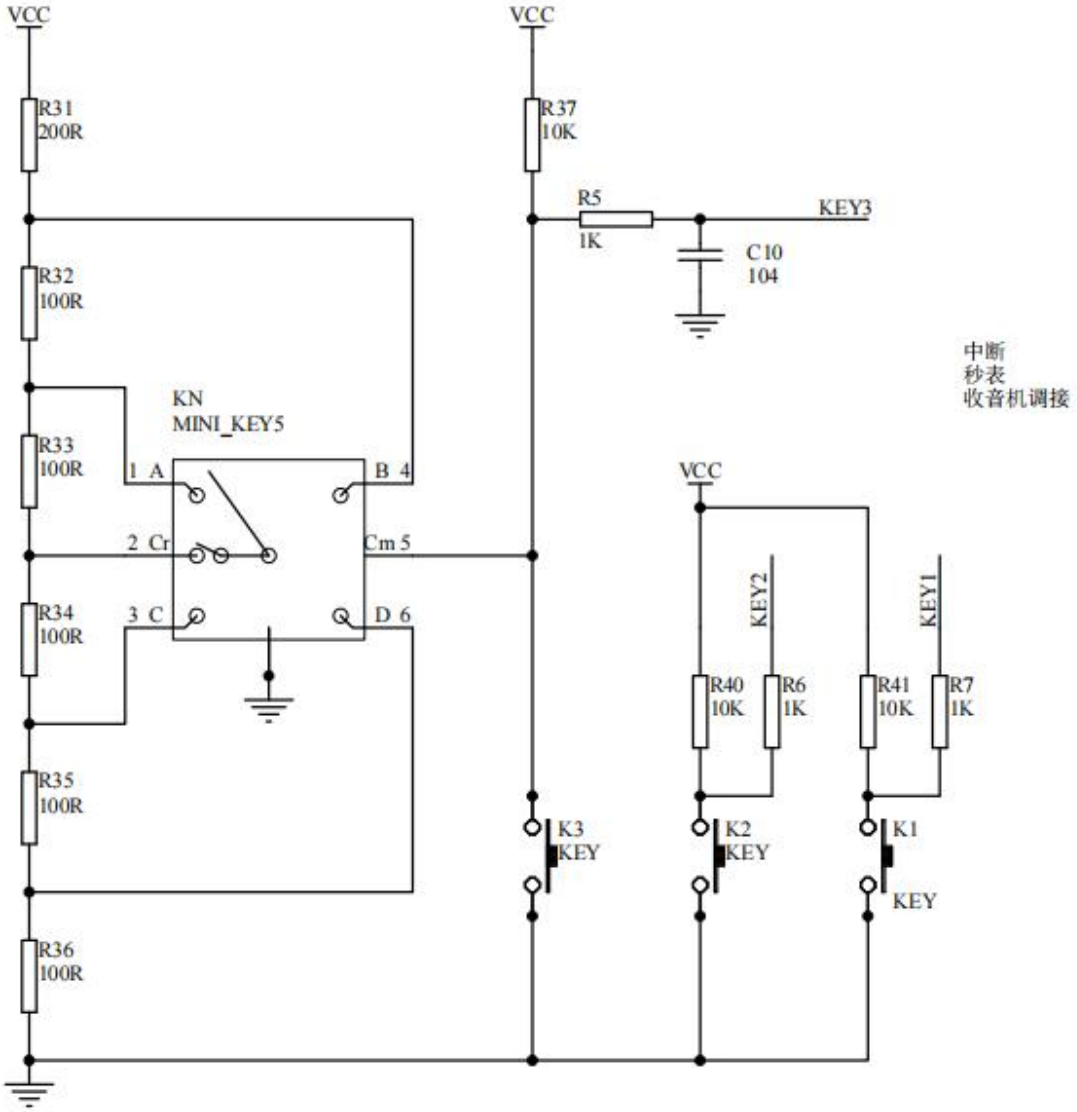


红外接收头 IR_R 用于接收红外发光二极管 IR_T 发出的红外信号，从而达到一个通信的目的。但在自然环境中并非只有红外发光二极管能发出红外线，自然光、日光灯灯光等光线中都含有红外线，故红外接收头需要对红外信号进行区分，把无关信号过滤。

因此，红外接收头被设计为只能接受一定频率范围内的红外线脉冲。例如，当红外发光二极管每隔 13us 发出一次红外线脉冲，发光时间也为 13us，即发出了一个 38kHz 的红外脉冲信号，而这个信号的频率恰好在接收头的接收范围内，接收头就会接收此红外信号并把这个 38kHz 的红外信号方波转换成电信号。而自然环境中的红外干扰信号不在接收头的接收频率内，接收头不会接收。

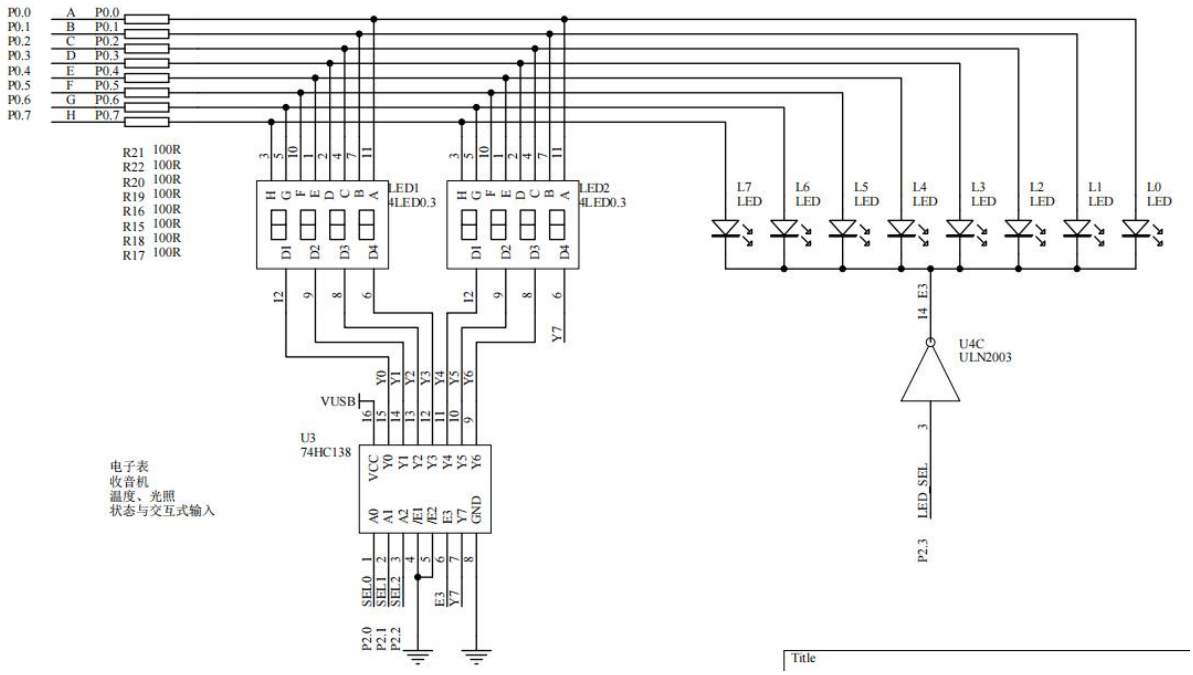
在我们的日常生活中，红外收发十分常用。电视机和空调的遥控的使用就是一个常见的事例。遥控器和电器上的接收器也是按类似的原理进行信号收发的，目的就是为了排除环境中的红外线的干扰。

3.6 按键模块



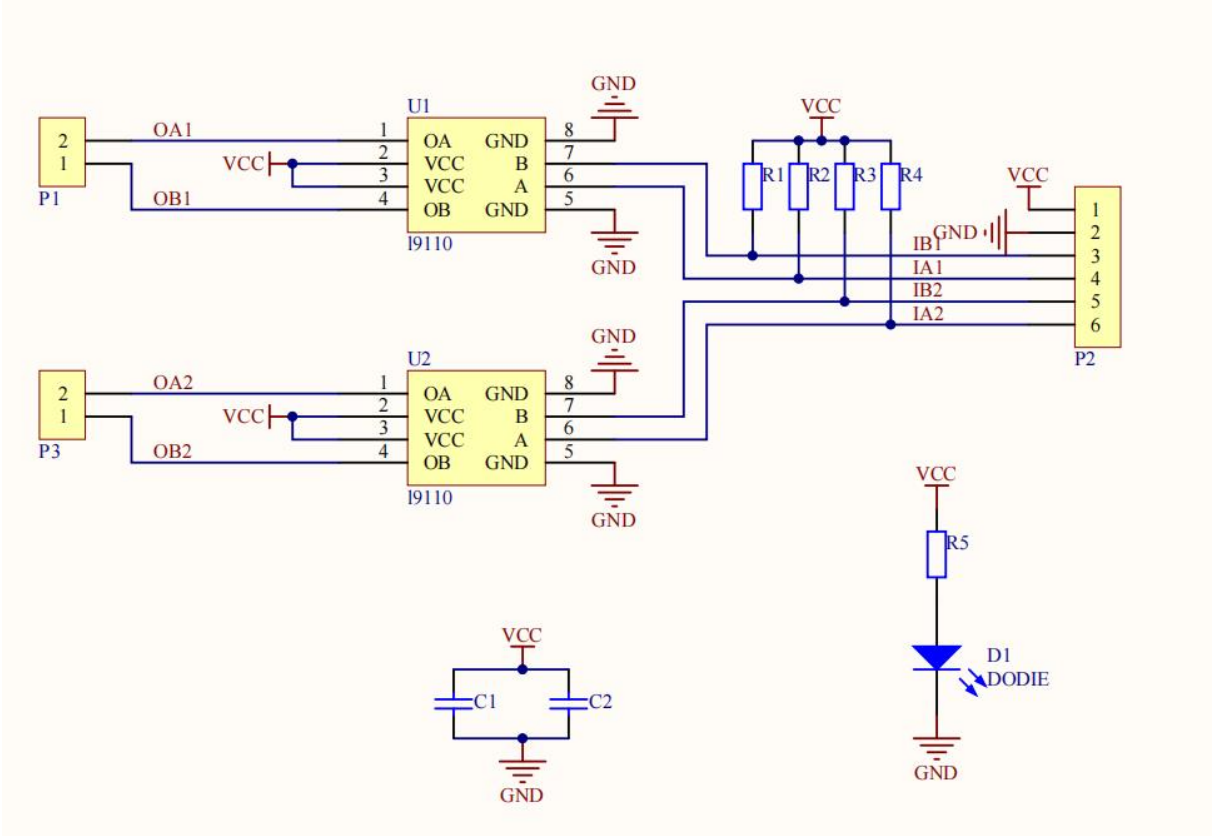
当按键被按下的时候，电路导通接地，I/O 口为低电平；当按键未被按下时，电路断开，I/O 口保持高电平。

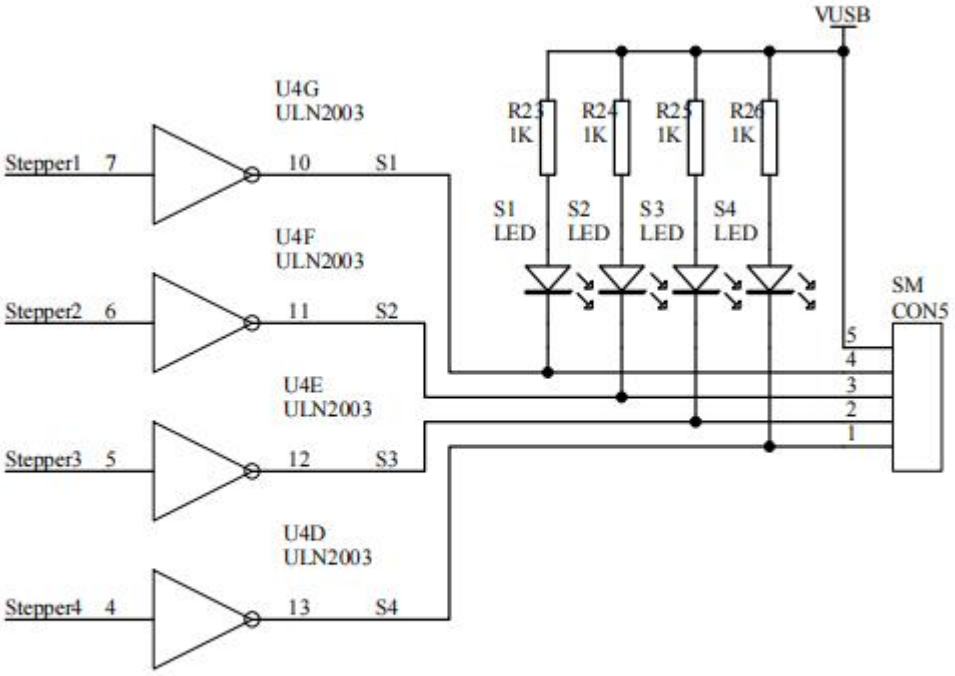
3.7 数码管显示



P0 口的 8 位输出分别控制 1 个 LED 数码管的 7 段和一个小数点；而 P2.3 经反相器 U4C 控制 74HC138 的使能信号 E3，结合 P2.0、P2.1、P2.2 这 3 个位选控制信号确定 8 个 LED 数码管中的哪个被点亮；电阻 R15～R22 为限流电阻。当段选为高、使能信号有效时，对应的 LED 管将会发光。通过以一定频率扫描位选信号，修改段选信号进行数码管点亮一段时间，从而给人视觉上几个数码管几乎同时显示的效果。

3.8 电机驱动

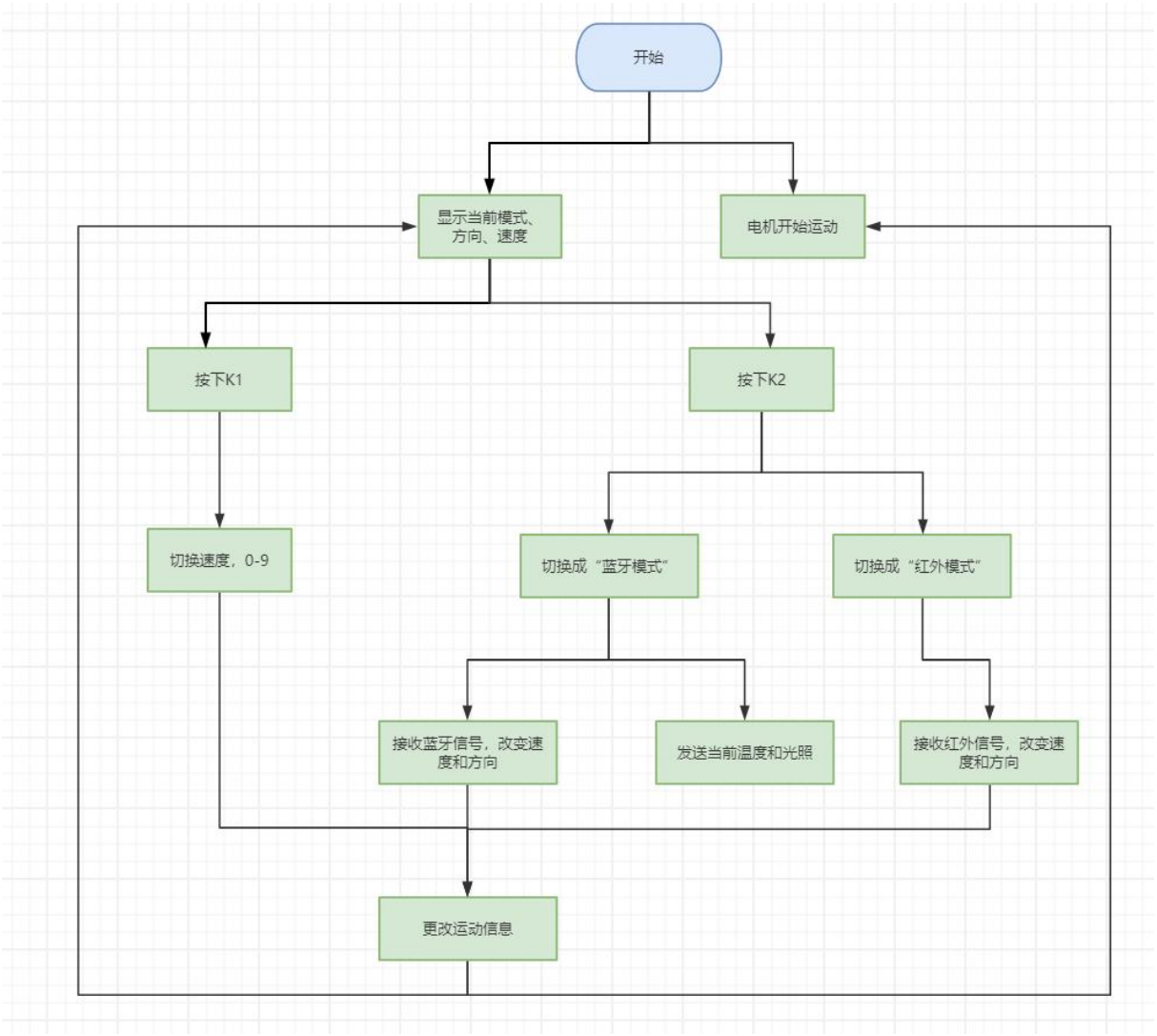




步进电机
指针电子钟

通过 P4.1、P4.2、P4.3 和 P4.4 控制 SM 口的四个引脚电平高低，截止通过 L9110S 的 H 桥电路控制两个 TT 电机。

4 软件设计与实现


开始
显示当前模式、
方向、速度
电机开始运动
按下K1
按下K2
切换速度，0-9
切换成“蓝牙模式”
切换成“红外模式”
接收蓝牙信号，改变速
度和方向
发送当前温度和光照
接收红外信号，改变速
度和方向
更改运动信息

motor.c

```
26  void init_motor()
27  {
28
29      P4M0 = 0x3f;
30      P4M1 = 0x00;    // 推挽模式
31
32      SetEventCallBack(enumEventSyslmS, pwm);
33
34
35  }
36
37  void go_forward()
38  {
39
40      P41 = 0;
41      P42 = 1;    //左轮子向前
42
43      P43 = 0;
44      P44 = 1;    //右轮子向前
45  }
46
47  void go_back()
48  {
49
50      P41 = 1;
51      P42 = 0;
52
53      P43 = 1;
54      P44 = 0;
55
56  }
57
58  void go_left()
59  {
60      P41 = 0;
61      P42 = 0;
62
63      P43 = 0;
64      P44 = 1;
65  }
66
67  void go_right()
68  {
69      P41 = 0;
70      P42 = 1;
71
72      P43 = 0;
73      P44 = 0;
74  }
75
```

```
76  void stop()
77  {
78      P41 = 0;
79      P42 = 0;
80
81
82      P43 = 0;
83      P44 = 0;
84  }
85
86
87  unsigned char  Bound = 10, Compare = 1; //pwm
88  unsigned char  Counter = 0;
89  enum direct my_direct = forward;
90
91  void pwm()
92  {
93      Counter++;
94      Counter %= Bound;
95
96      LedPrint(Counter);
97      if(Counter < Compare)
98      {
99          switch (my_direct)
100         {
101             case forward:
102                 go_forward();
103                 break;
104             case back:
105                 go_back();
106                 break;
107             case left:
108                 go_left();
109                 break;
110             case right:
111                 go_right();
112                 break;
113         }
114     }
115     else stop();
116 }
117
118
119 void set_compare(int cmp)
120 {
121     Compare = cmp;
122 }
123
124 void set_mydirect(enum direct new_direct)
125 {
126     my_direct = new_direct;
127 }
128
```

在 init_motor()中，将设置 enumEventSys1mS 事件的回调函数为 pwm()；go_back()，go_right()等函数是控制两个电机的开关，以实现同速或差速。

```
87  unsigned char  Bound = 10, Compare = 1; //pwm
88  unsigned char  Counter = 0;
89  enum direct my_direct = forward;
90
91  void pwm()
92  {
93      Counter++;
94      Counter %= Bound;
95
96      LedPrint(Counter);
97      if(Counter < Compare)
98      {
99          switch (my_direct)
100         {
101             case forward:
102                 go_forward();
103                 break;
104             case back:
105                 go_back();
106                 break;
107             case left:
108                 go_left();
109                 break;
110             case right:
111                 go_right();
112                 break;
113         }
114     }
115     else stop();
116 }
```

PWM（Pulse Width Modulation，脉冲宽度调制）是一种电子电路中常用的信号编码方法，用于模拟信号的数字传输。它通过改变数字信号的脉冲宽度来编码模拟信号的幅度。PWM 信号通常是一个固定频率的方波，其高电平持续的时间（即脉冲宽度）决定了所传输的模拟值的大小。

在 PWM 技术中，信号的占空比（即高电平时间与整个周期时间的比例）是可变的。这个占空比可以用来代表不同的电压水平或者数据值。例如，在一个 50%占空比的 PWM 信号中，高电平时间和低电平时间相等，这通常对应于供电电压的一半。如果占空比增加到 75%，那么输出的有效电压将接近于供电电压。

这里 Compare 的值高，占空比大；Compare 的值低，占空比小。取值范围为 1-10，所以速度档位有 10 个。

dis.c

```c
unsigned char sign = 15;

// unsigned char a;
// unsigned char b;
// unsigned char c;
// unsigned char d;

void print_info()
{
    switch (my_direct)
    {
        case forward:
            sign = 15;
            break;
        case back:
            sign = 11;
            break;
        case left:
            sign = 17;
            break;
        case right:
            sign = 10;
            break;
    }

    if(my_model == bluetooth) Seg7Print(11, 17, 24, 24, sign, 24, 24, Compare - 1);
    else if(my_model == ir) Seg7Print(1, 10, 24, 24, sign, 24, 24, Compare - 1);
    else Seg7Print(10, 5, 24, 24, sign, 24, 24, Compare - 1);


}

void init_dis()
{
    DisplayerInit();
    SetDisplayerArea(0, 7);

    SetEventCallBack(enumEventSys100mS, print_info);
}
```

数码管上显示当前运动模式、方向和速度（0-9），每 100ms 一次。

btn.c

```c
void set_btn_info()
{
    if(GetKeyAct(enumKey2) == enumKeyPress)
    {
        my_model++;
        my_model %= 3;
    }
    else if(GetKeyAct(enumKey1) == enumKeyPress)
    {
        Compare++;
        if(Compare == 11)  Compare = 1;
    }

    switch (my_model)
    {
        case bluetooth:
            init_btooth();
            break;
        case ir:
            init_ir();
            break;
        case rs485:
            break;
    }

}


void init_btn()
{
    KeyInit();

    SetEventCallBack(enumEventKey, set_btn_info);
}
```

K1 用来调节速度大小，K2 用来调节运动模式。

buzzer.c

```c
void init_buzzer()
{
    BeepInit();
}

void make_noise()
{
    SetBeep(2400, 200);
}
```

将蜂鸣器类重新封装，每次鸣叫 2s。

vibration.c

```c
void init_vib()
{
    VibInit();
    SetEventCallBack(enumEventVib, make_noise_and_stop);
}

void make_noise_and_stop()
{
    Compare = 1;
    make_noise();

}
```

每次检测到振动发生，就将速度调节至最低档，并鸣叫。

light.c

```c
int code tempdata[] = {239, 197, 175, 160, 150, 142, 135, 129, 124, 120, 116, 113, 109, 107, 104, 101,
                       99, 97, 95, 93, 91, 90, 88, 86, 85, 84, 82, 81, 80, 78, 77, 76,
                       75, 74, 73, 72, 71, 70, 69, 68, 67, 67, 66, 65, 64, 63, 63, 62,
                       61, 61, 60, 59, 58, 58, 57, 57, 56, 55, 55, 54, 54, 53, 52, 52,
                       51, 51, 50, 50, 49, 49, 48, 48, 47, 47, 46, 46, 45, 45, 44, 44,
                       43, 43, 42, 42, 41, 41, 41, 40, 40, 39, 39, 38, 38, 38, 37, 37,
                       36, 36, 36, 35, 35, 34, 34, 34, 33, 33, 32, 32, 32, 31, 31, 31,
                       30, 30, 29, 29, 29, 28, 28, 28, 27, 27, 27, 26, 26, 26, 25, 25,
                       24, 24, 24, 23, 23, 23, 22, 22, 22, 21, 21, 21, 20, 20, 20, 19,
                       19, 19, 18, 18, 18, 17, 17, 16, 16, 16, 15, 15, 15, 14, 14, 14,
                       13, 13, 13, 12, 12, 12, 11, 11, 11, 10, 10, 9, 9, 9, 8, 8, 8, 7,
                       7, 7, 6, 6, 5, 5, 54, 4, 3, 3, 3, 2, 2, 1, 1, 1, 0, 0, -1, -1, -1,
                       -2, -2, -3, -3, -4, -4, -5, -5, -6, -6, -7, -7, -8, -8, -9, -9,
                       -10, -10, -11, -11, -12, -13, -13, -14, -14, -15, -16, -16, -17,
                       -18, -19, -19, -20, -21, -22, -23, -24, -25, -26, -27, -28, -29,
                       -30, -32, -33, -35, -36, -38, -40, -43, -46, -50, -55, -63, 361};

struct_ADC temp;
int sum = 0, countdown = 100;
unsigned char buffer_adc[2] = {0, 0}; //热 光

void calc_temp_and_light()
{
    temp = GetADC();
    buffer_adc[1] = temp.Rop;
    if (countdown)
    {
        countdown--;
        sum += temp.Rt >> 2;
    }
    else
    {
        sum = tempdata[sum / 100];
        buffer_adc[0] = sum;

        sum = 0;
        countdown = 100;
    }
}


void init_light()
{
    AdcInit(ADCexpEXT);
    SetEventCallBack(enumEventSys10mS, calc_temp_and_light);
}
```

ADC 中 Rop 是能直接获取的准确数据，Rt 则需要多次取平均值获得，这里采用查表的方法。每 10ms 回调一次，100 次回调得到一个准确值，相当于 1s 得到一次准确的信息。

btooth.c

```c
extern enum direct my_direct;
unsigned char buffer[1], *match;

void set_speed_and_direct()
{
    unsigned char inst = buffer[0];

    if (inst > 0x20) make_noise();
    else if(inst > 0x10) Compare = inst & 0x0f;
    else my_direct = inst;

}
void sent_temp_and_light()
{
    Uart2Print(buffer_adc, 2);
}
void init_btooth()
{
    Uart2Init(9600, Uart2UsedforEXT);
    SetUart2Rxd(buffer, 1, match, 0);

    SetEventCallBack(enumEventUart2Rxd, set_speed_and_direct);
    SetEventCallBack(enumEventSys1S, sent_temp_and_light);
}
```

这里蓝牙部分使用 Uart2EXT，接收缓冲区大小为 1 字节，根据高四位的值判断指令类型。高四位为 0x2 是鸣笛指令，0x1 是速度指令，0x0 是方向指令。同时每一秒向外发送温度和光照值，一共 2 字节。

ir_re.c

```c
unsigned char buffer_ir[1];


void set_speed_and_direct_ir()
{
    unsigned char inst = buffer_ir[0];

    if(inst > 0x10) Compare = inst & 0x0f;
    else my_direct = inst  & 0x07;

}
void init_ir()
{
    IrInit(NEC_R05d);
    SetIrRxd(buffer_ir,1);
    SetEventCallBack(enumEventIrRxd, set_speed_and_direct_ir);

}
```

红外模块只接受数据，调整速度大小和方向。

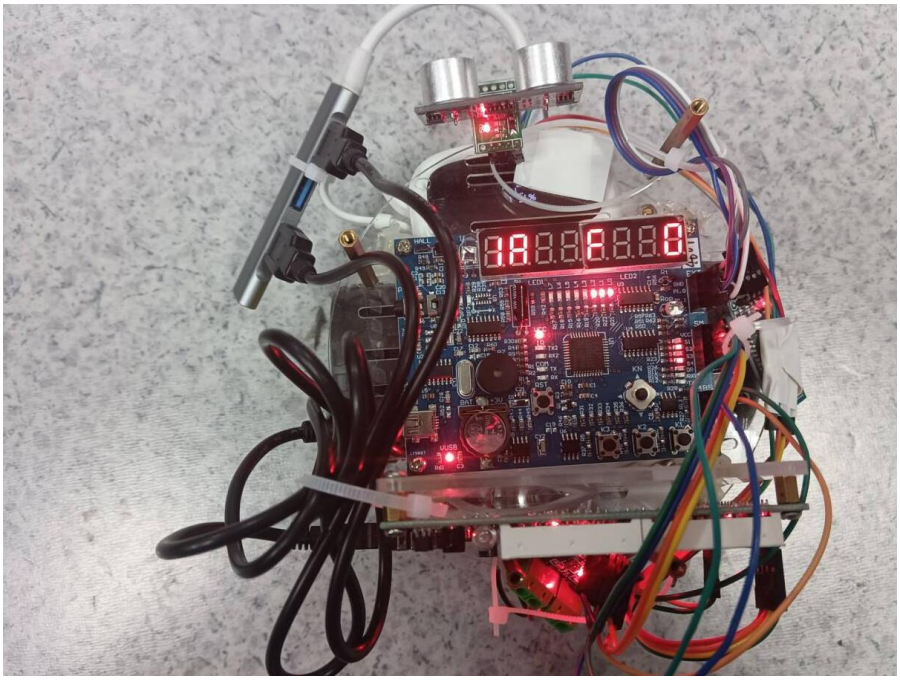
main.c

```c
code unsigned long SysClock = 11059200; // System clock frequency (Hz), used for timing calculations

enum model my_model = rs485;

void main()
{

    init_motor();
    init_dis();
    init_btn();
    init_buzzer();
    init_vib();
    init_light();


    MySTC_Init();
    while (1)
    {
        MySTC_OS();
    }

}
```

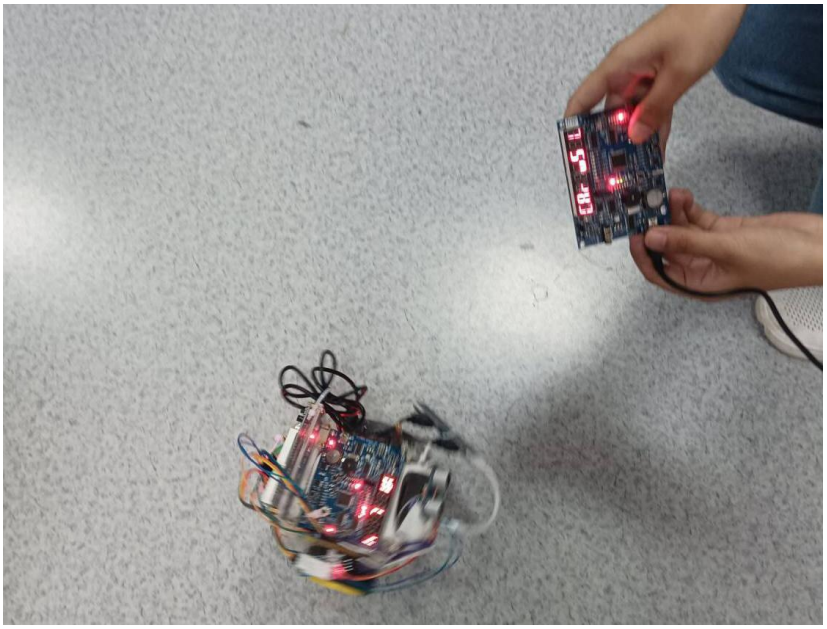
5　系统测试与分析

硬件环境：STC-B 学习板 ，15F2K61S2 芯片

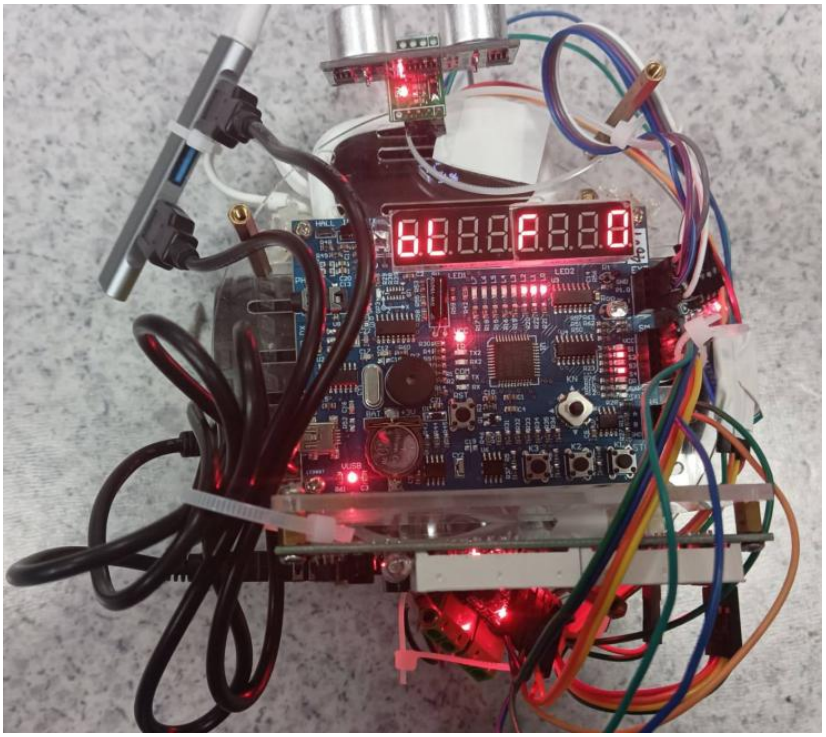
软件环境：Keil C，Android-12



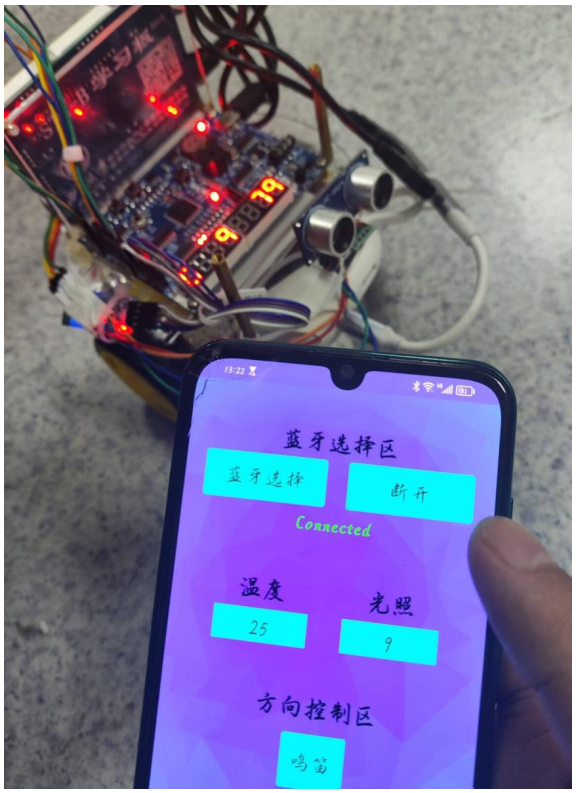
数码管上显示 “IR”，表示是红外模式。



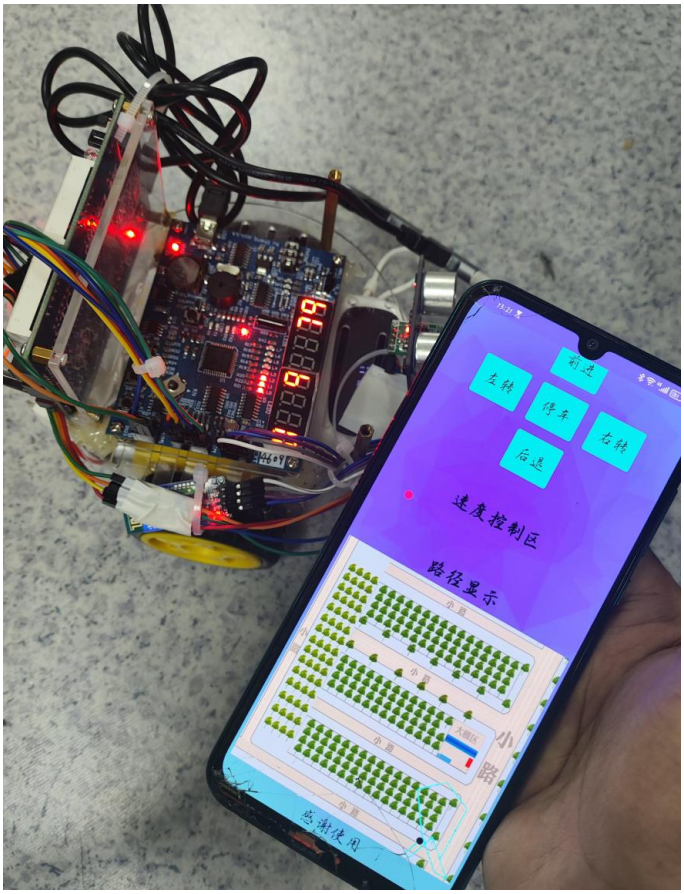
根据手势控制小车改变方向



数码管显示“bL”，进入蓝牙模式。



蓝牙已经连接上，手机上显示当前温度和光照



手机上显示小车走过的路径

6 总结与展望

本项目成功地开发了一款基于体感和 Android 控制的智能小车，实现了通过手势控制小车移动、通过 Android 应用进行远程操作并实时接收环境数据等功能。

小车通过超声波传感器和二轮差速技术实现了避障，提高了在复杂环境中的适应能力；同时，热敏电阻和光敏电阻用于检测环境条件，并通过手机端展示给用户。此外，小车还能在遇到障碍时自动停止，并通过蜂鸣器发出警告。手机端能控制小车的速度和方向，并显示小车位置，若是当前环境不符合要求就标记，便于后续查看。此外，还可以通过识别手势发送红外信息控制小车速度和方向。

项目实施过程中，我们克服了硬件选择、车身组装、代码编写及调试等一系列挑战，达到了预期目标。

然而，项目仍存在一些不足之处，比如小车的稳定性有待进一步提高，可以提高车身成本，使用更加坚固的材料；环境感知的准确性也有待增强，传感器获得的数据并不精准，算法优化层面也有待改善，在未来的工作中，我们可以考虑引入更先进的传感器技术来提升小车的感知能力，如使用更高精度的 GPS 定位系统来改善小车的位置跟踪功能，使用摄像头模块增强采集和识别功能；另外，可以探索机器学习算法的应用，以增强小车对复杂环境的理解和应对能力。同时，优化软件架构和代码质量也将是未来改进的重点。总之，通过不断的技术革新和功能完善，这款智能小车有望成为农业自动化领域的有力工具。