

浙江大学实验报告

专业：信息工程

姓名：_____

学号：_____

日期：_____

地点：_____

课程名称：智能移动系统设计 指导老师：叶险峰 成绩：_____

实验名称：基于 Arduino 的智能小车的设计与制作 实验类型：_____ 同组学生姓名：_____

一、实验目的

1. 掌握并熟练 arduino 编程
2. 进一步熟练电路设计和实现能力
3. 使用亚博智能小车套件完成一辆避障寻迹小车的制作
4. 自主设计并制作基于 Arduino 的多功能智能小车

二、亚博智能小车套件小车制作

1. 小车组装与基础功能实现

1.1 组装

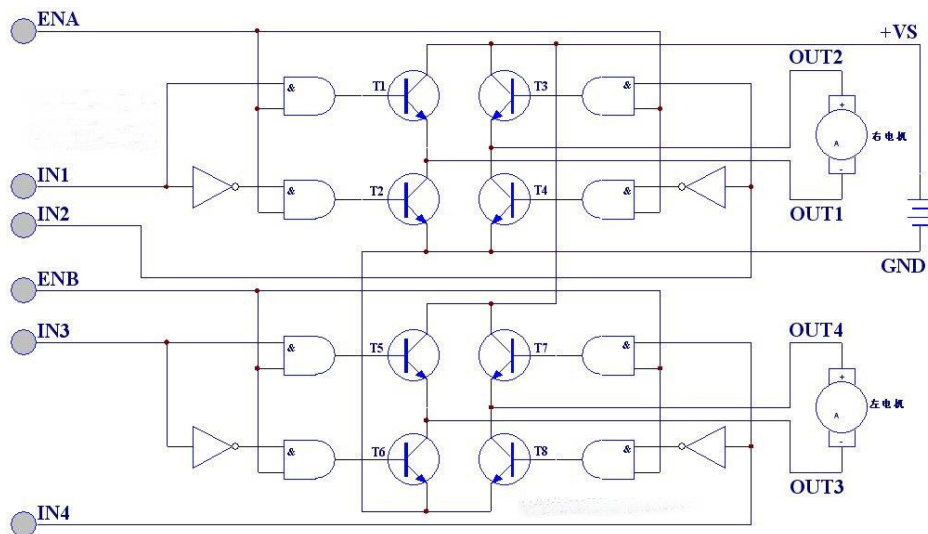
按照视频教程分别装好电机, 车轮, 使用平头螺丝将电池盒安装在底盘下方. 再使用铜柱分别将 arduino Uno R3 开发板, 面包板安装在底盘上方.

1.2 基础功能

1.2.1 小车运动

- 原理: 通过 arduino 控制 L293D 电机驱动模块, 驱动电机转动速度与方向

■ L293D 控制原理:



L293D 芯片内部使用了双 H 桥路的电路结构. 首先 ENA 和 ENB 通过与门控制 IN1, IN2, IN3, IN4 信号输入使能. 每个输入 IN 信号控制两个 npn 三极管的通断, 且由于非门. 两个三极管不同时导通.

以 IN1 控制 OUT1 为例: IN1 为高电平时, T1 饱和导通, T2 截止, +VS 电压将加在 T2 三极管的集电极, 则 OUT1 被置为高电压供电; 反之, IN2 为低电平时, T1 截止, T2 饱和导通, +VS 电压加在 T1 的集电极而 T1 的发射极与 GND 近似等电, 则 OUT1 为不供电

而 OUT1~OUT4 分别对应两个电机的供电电压. 当电机两端一边接地一边高电压 VS+ 供电时, 有电流流经电机带动转子转动. 调换两端电压高低, 则转子转

动方向也相反. 注意 OUT1, OUT2 与 OUT3, OUT4 不可同时置高电平否则会烧坏电机.

■ Arduino 控制原理与 PWM 波

分别使用 D8, D9, D10, D11 口对电机驱动模块 IN1, IN2, IN3, IN4 输入, 分别控制左轮后退, 左轮前进, 右轮前进, 右轮后退. 注意同一车轮的前进, 后退信号不能同时保持高电平, 因此每次输出时均要对统一车轮的两个口写入.

代码即为多次 `digitalWrite()`, 如果需要调速即可使用 `analogWrite()`.

值得注意的是, Arduino 没有真正意义上的模拟输出, 当使用 `analogWrite()` 时, 实际上是输出 PWM (Pulse width modulation wave, 脉宽调制) 波, 由于冲量相等效果相同的原理 (冲量: 脉冲的幅值对时间积分), 不同占空比的 PWM 波可等效于不同值的模拟量.

ArduinoUnoR3 只有部分 digital 输出口支持 PWM 波, D8 就不支持 PWM 波输出, 所以理论上对于 D8 口 (此处为控制左轮后退), `<255` 的 `analogWrite` 值都是输出低电平, 仅当 `analogWrite` 值为 255 时有高电平输出. 因此后退时必定只有一轮可以调速.

● 代码实现

多次 `digitalWrite()` 或 `analogWrite()`, 略.

1.2.2 按键启动和蜂鸣器报警

● 原理:

■ 当按键按下时, 按钮两端将会由开路变成短路, 因此可以让按钮一端接高电平, 在另一端监测电平高低判断按钮是否按下;

■ 按键消颤动与蜂鸣器按键提示:

当按钮从 开始按下 到 完全导通, 以及从 导通 到 完全松开的过程中, 由于手指压力波动, 按钮不稳定, 按钮的输出将会有波动.

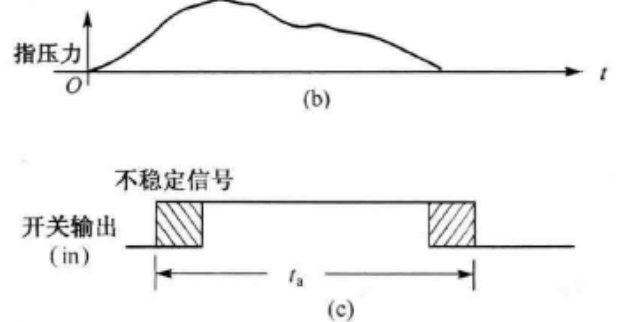
不稳定信号段的时长一般为几毫秒, 而按键按下一次的总时长 t_a 一般小于 100ms, 因此在首次检测到按键按下时, 等待 10ms, 若按钮按下此

时应已经进入稳定信号阶段, 再做一次判断, 可确保按钮是被稳定按下的.

此时打开蜂鸣器提示按钮按下, 直到检测到第一次开关断开时关闭蜂鸣器, 可获得时长合适、声音清晰且只出现一次的按键提示音.

● 代码实现: `keyscan()` 函数, 在 `loop()` 函数的最开头. 若没有按下按键, 将不会跳出 `keyscan()` 函数.

```
void keyscan()
{
    int val;
    val=digitalRead(key); //读取数字7 口电平值赋给 val
    while(!digitalRead(key)) //当按键没被按下时, 一直循环
    {
```



```
val=digitalRead(key);//此句可省略，可让循环跑空
}
while(digitalRead(key))//当按键被按下时
{
    delay(10); //延时10ms
    val=digitalRead(key);//读取数字7 口电平值赋给 val
    if(val==HIGH) //第二次判断按键是否被按下
    {
        digitalWrite(beep,HIGH); //蜂鸣器响
        while(digitalRead(key)); //判断按键是否被松开
        digitalWrite(beep,LOW); //蜂鸣器停止
    }
    else
        digitalWrite(beep,LOW);//蜂鸣器停止
}
}
```

1.2.3 LCD 显示功能

● 原理

1602LCD 采用标准 16 脚接口，可显示 16*2 个字符，字符尺寸为 2.95×4.35mm. 其各引脚的功能为：

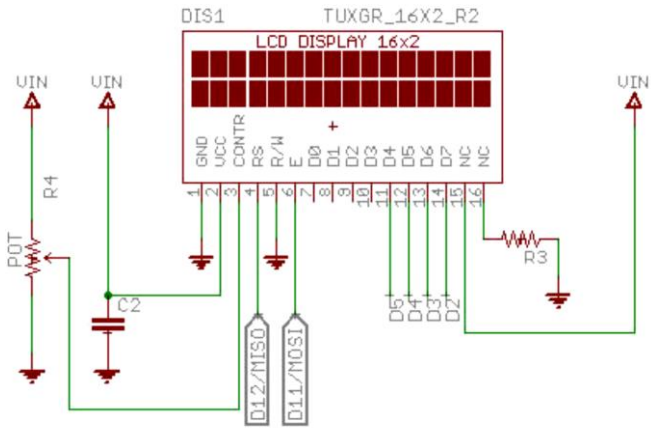
第 1 脚：VSS 为地电源
第 2 脚：VDD 接 5V 正电源
第 3 脚：VO 为液晶显示器对比度调整端，接正电源时对比度最弱，接地电源时对比度最高，对比度过高时会产生“鬼影”，使用时可以通过一个 10K 的电位器调整对比度
第 4 脚：RS 为寄存器选择，高电平时选择数据寄存器、低电平时选择指令寄存器。
第 5 脚：R/W 为读写信号线，高电平时进行读操作，低电平时进行写操作。当 RS 和 RW 共同为低电平时可以写入指令或者显示地址，当 RS 为低电平 RW 为高电平时可以读信号，当 RS 为高电平 RW 为低电平时可以写入数据。
第 6 脚：E 端为使能端，当 E 端由高电平跳变成低电平时，液晶模块执行命令。
第 7~14 脚：D0~D7 为 8 位双向数据线。
第 15 脚：背光源正极
第 16 脚：背光源负极

● 硬件实现

我们采用四位连接模式，接线如右图所示：

其中，可变电阻直接用 2 个 2kΩ 电阻串联代替。

C2 为 104 电容，即 0.1 μF.



- 代码实现

首先应包含<LiquidCrystal.h>库

4 位连接模式的接口声明为: LiquidCrystal(rs, enable, d4, d5, d6, d7))

这里采用 D13, D12, D6, D5, D4, D3 作为对应, 即

LiquidCrystal lcd(13, 12, 6, 5, 4, 3);

使用 LCD 显示距离代码略, 较简单且例程中已经包含。

使用 LCD 显示模式代码如下:

```
void Display_mode(int Mode)
{
    lcd.home();
    if(!Mode)
    {
        lcd.print("Free Control");
        lcd.setCursor(6,2);    //把光标定位在第2行, 第6列
        lcd.print("Mode");
    }
    else
    {
        lcd.print("Tracking");
        lcd.setCursor(6,2);    //把光标定位在第2行, 第6列
        lcd.print("Mode");
    }
}
```

2. 综合功能实现（每个功能均有对应视频展示）

2.1 超声波测距

- 原理

声速 $344\text{m/s} = 344\text{ }\mu\text{m}/344\text{ }\mu\text{s}$;

则发射一道超声波, X 秒后被前方 Y 米处的遮挡反射回来, 有:

Y 米 = (X 秒 \times 344) / 2

则 X 秒 = (2 \times Y 米) / 344

则 X 秒 = 0.0058 \times Y 米

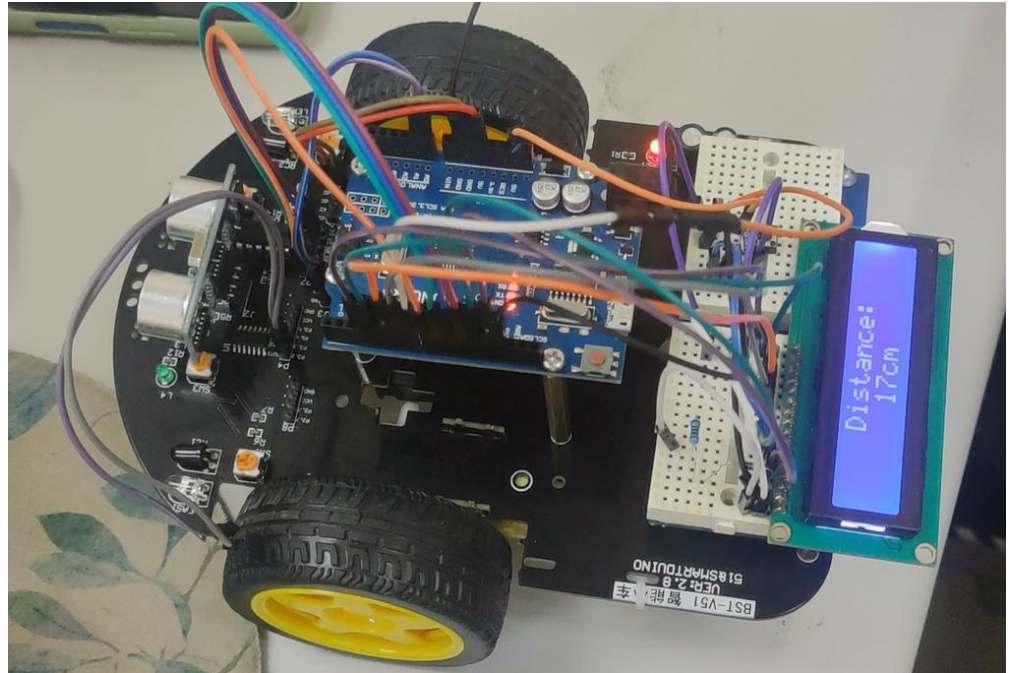
则 Y 厘米 = X 微秒 / 58

- 代码实现

```
float Distance_test()    // 量出前方距离
{
    digitalWrite(Trig, LOW);    // 给触发脚低电平 2 $\mu\text{s}$ 
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);    // 给触发脚高电平 10 $\mu\text{s}$ , 这里至少是 10 $\mu\text{s}$ 
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);    // 持续给触发脚低电
```

```
float Fdistance = pulseIn(Echo, HIGH); // 读取高电平时间(单位:
微秒)
Fdistance= Fdistance/58;
return Fdistance;
}
```

- 实现效果（详见视频）
成功通过 LCD 显示了前方距离



2.2 无舵机超声波避障（详见视频）

当检测到前方有障碍时，向右转一定角度。

2.3 带舵机超声波避障（详见视频）

当检测到正前方有障碍，舵机控制超声波传感器检测左右测空旷距离，选择往空旷距离更大的一侧转弯。

2.4 红外避障（由于底板可调电阻损坏，效果不好，详见视频）

通过底板上的红外传感器监测前方距离。红外传感器只能判断有反射/无反射，读取红外传感器信号为 `digitalRead (Sensor)`，若接收到的值为 `HIGH`，则表明无反射，若接收到的值为 `LOW`，则表明有反射。某侧有反射时，说明那侧有障碍，则向反方向转向。

由于可调电阻损坏，有一侧朝前的红外传感器始终保持无信号，则始终向一个方向转弯。

注：依靠红外或者超声波的跟随，只需要在代码中将避障的逻辑改变成向障碍物方向运动，但与障碍物保持一定距离。与避障功能的实现无本质区别，因此略去。

2.5 红外遥控（详见视频）

- 原理

红外遥控器将会以 38KHz 的频率发送有限长红外脉冲串。当 VS1838 红外接收器接收到脉冲串时，会输出低电平；而它未接收到脉冲串时，输出高电平。因此红外接收器就能输出高低电平序列表示编码。

通过<IRremote.h>库，解析红外遥控器的红外信号。

- 代码实现

```
#include <IRremote.h> // 包含红外库
int RECV_PIN = A5; // 端口声明
int on=0;
IRrecv irrecv(RECV_PIN);
decode_results results; // 结构声明

void dump(decode_results *results)
{
    int count = results->rawlen;
    if (results->decode_type == UNKNOWN)
    {
        //Serial.println("Could not decode message");
        brake(1);
    }
}

-----调-用-方-法-----

void Loop()
{
    .....省略.....

    if (irrecv.decode(&results)) // 调用库函数：解码
    {
        flag_change=1;

        // If it's been at least 1/4 second since the last
        // IR received, toggle the relay
        if (millis() - last > 250) // 确定接收到信号
        {
            on = !on; // 标志位置反
            digitalWrite(13, on ? HIGH : LOW); // 板子上接收到信号
            闪烁一下 led
            dump(&results); // 解码红外信号
        }
    }

    .....省略.....
}
```


而后调用 `results.value` 即可获得红外指令。

2.6 黑线循迹（详见视频）

使用底盘底部的红外传感器，黑色吸收红外线能力更强，因此在同等距离下，通过适当调节红外传感器的灵敏度，可以使得其恰能接受到白色地砖的红外反射而不能接受到黑色胶布区域的红外反射。

当黑线处于两红外传感器中间时即为正确方向，此时两传感器输出都为 LOW。而当方向偏离，红外传感器到了黑线上方，则会输出 HIGH。则哪边输出 HIGH, 就往哪边转向。通过合理设置转向和前进的时间和速度，可以实现始终保持黑线处于两红外传感器之间。

代码实现为例程稍加修改。例程中对应的黑线较宽，当两传感器输出 HIGH 时是正确方向。因此需要将两传感器的判断逻辑取反，即改成两传感器输出 LOW 时为正确方向。

2.7 遥控&循迹——模式切换（详见视频）

我们使用了红外遥控器的多余按键，来控制模式的切换。由此先进行自由控制，在不需重新烧录代码的情况下，可以切换到循迹或者舵机超声波避障。

其中 `flag` 变量是控制模式的变量，通过红外遥控器更改。我们使用了 `<TaskScheduler.h>` 库来每隔一定时间就尝试接受并解析红外线信号。关键代码如下：

```
long tracking = 0x00ff42bd; //按键7 控制主动循迹
long avoidance = 0x00ff52ad; //案件9 控制避障
long remote_control = 0x00ffa857; //三角按键 红外控制

int flag=0;

void flagUpdate()
{
    .....
    解析红外信号且修改 flag 的值
    .....
}

Task FlagTask(10,TASK_FOREVER,&flagUpdate);
Scheduler Sch;

/*
    首先需要声明一个任务日程表叫做 Sch，然后声明一个任务 FlagTask
    这个任务的内容是每 10ms，调用一次 flagUpdate 函数，即解析红外信号并修改
    flag 变量的值。
    在之后的 setup 函数中，需要对任务日程表 Sch 初始化，再将 FlagTask 函数加入
    日程表 Sch，最后对 FlagTask 任务使能。
*/
void setup()
{
    .....
    省略
```

```

        .....

    Sch.init();                //初始化
    Sch.addTask(FlagTask);    //添加任务
    FlagTask.enable();        //任务使能
}

void Loop()
{
    .....省略.....
    Sch.execute();

    switch(flag)
    {
        case 0://自由控制
            .....
            例程代码:根据红外信号的解析结果,用 switch-case 控制调用运动函数
            .....
        case 1://循迹
            .....
            例程代码:循迹
            .....
        case 2://避障
            .....
            例程代码:避障
            .....
    }
    .....省略.....
}

```

三、第二辆智能小车

3.1 车辆设计

3.1.1 明确功能需求

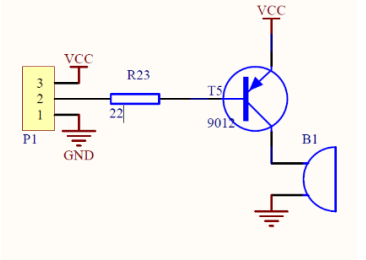
- 1) 音乐播放器功能:
 - a) 能够播放/暂停音乐,能够存储多首音乐并切换音乐。
 - b) 音乐在暂停后能够继续播放
- 2) 左右转向灯与倒车灯功能
 - a) 转向和倒车时自动开启
 - b) 可通过手机 app 直接控制
- 3) 蓝牙控制功能: 所有功能的控制信号通过蓝牙发送。
- 4) 状态测量与显示功能:

- 1、小车实时速度测量与显示
- 2、小车所在环境温度测量与显示
- 3、小车所在环境光强测量与显示
- 5) 左右轮转速（analogWrite 值）手动控制
- 6) 按键小车运动控制
- 7) 万向运动控制（拖动小球，通过小球坐标更新小车左右轮转速）

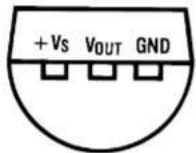
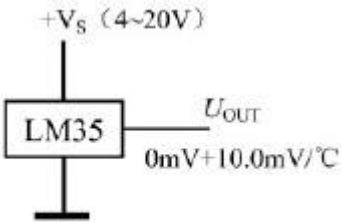
3.1.2 各功能模块设计思路

Arduino 接口规划：

3 个 digital 6,12,13	LED（左转向，右转向，夜晚开前灯）
1 个 digital 7	蜂鸣器
4 个 digital 8,9,10,11	马达
2 个 digital 2,3	（槽形光电开关*2）
2 个 digital 4,5	蓝牙
1 个 analog A1	光强感应
1 个 analog A2	LM35 温度传感

功能	硬件实现原理	arduino 实现原理
无源蜂鸣器 音乐播放	<p>一个驱动电路，带 PNP 三极管 S9012 和 1k Ω 电阻。</p> <p>当三极管基极为低电平时，三极管导通。通过三极管驱动可以使得蜂鸣器有较大的电流驱动。值得注意的是，在不播放音乐时，应该将蜂鸣器的基极设置为高电平，以使 PNP 三极管截止。</p>  <p>上图中的电阻应该是 1k Ω S9012 封装，平面向自己，从左到右 EBC</p> <p>无源蜂鸣器原理图与封装，从立创商城下载</p>	通过 tone 和 noTone 函数指定蜂鸣器以某一频率发声。

蓝牙控制	HC-06（ZS-040）蓝牙模块	通过<SoftwareSerial.h>库中的read, print, available 等方法								
左右转向灯与倒车灯的自动与手动控制	通过 3 个 Digital 接口控制 3 个 LED 灯	在运动函数中调用灯光控制函数，且通过蓝牙接收开关灯信号								
测速	使用槽型光耦与 20 格码盘。槽中无遮挡，接收管导通，输出高电平；有遮挡，输出低电平。 需要使用 arduino 的 2,3 口，这样才能使用中断函数。	使用中断函数（让 CPU 中断手中的事去立刻执行某个紧急的命令）								
		外部中断（被外部信号触发） 用来累计被测速轮遮挡的次数								
		<div>attachInterrupt(interrupt, function, mode)</div> <div>interrupt:触发源 function:触发后要执行的函数 mode:触发模式</div> <table><tr><td>RISING</td><td>上升沿触发</td></tr><tr><td>FALLING</td><td>下降沿触发</td></tr><tr><td>LOW</td><td>低电平触发</td></tr><tr><td>CHANGE</td><td>变化时触发</td></tr></table>	RISING	上升沿触发	FALLING	下降沿触发	LOW	低电平触发	CHANGE	变化时触发
		RISING	上升沿触发							
FALLING	下降沿触发									
LOW	低电平触发									
CHANGE	变化时触发									
<div>功能：当 interrupt 口的信号发生了 mode 所说明的变化时，立刻执行 function.</div>										
		内部中断（被内部计时器触发）								
		<div>MsTimer2 库</div>								
		<div>MsTimer2::set(time, function) 从计时开始起，每 time 时间执行一次 function</div>								
		<div>MsTimer2::start()开始计时</div>								
		<div>MsTimer2::end()结束计时</div>								
温度测量	LM35 温度传感器，具有很高的工作精度和较宽的线性工作范围，工作电压范围为	<div>Vin = analogRead(LM35) * 5.0 / 1023; //计算出 A0 的电压，单位为 V temperature = Vin * 1000.0 / 10.0;</div>								

	<p>4~30V。其精度为 0.5° C，输出电压与摄氏温度成线性比例关系，非线性温度误差</p> <p>低于±0.25° C，线性温度系数为+10mV/° C，无需外部校准或微调。输出电压与温度的表达式为：</p> $U_{out} = 10mV/^{\circ}C \times t/^{\circ}C$ <p>传感器采用单电源模式供电。单电源供电模式在 25° C 下电流约为 50 μ A，温度测量范围</p> <p>0° C~100° C，其输出电压范围为 0~1V。封装示意图如下。</p> <div><p>TO-92 Plastic Package</p><p>BOTTOM VIEW DS005516-2</p><p>Order Number LM35CZ, LM35CAZ or LM35DZ</p><p>(a) 单电源模式</p></div>	<p>//将 A0 电压要转换成 mV，根据 LM35 转换系数 10mV/° C，除以 10，得出温度</p> <p>即首先 Arduino 的模拟端口实质上是取值范围从 0~1023，共有 1024 个离散取值的端口，则单位电压为参考电压比上 1023。端口读取的数据是将端口电压比上单位电压的相对值。例如 A0 端口输入 2.5V，相当约于 512 个单位电压，则 analogRead(A0) 的结果应为 512，乘上单位电压 5.0/1023 才能得到电压绝对值 2.5V。</p> <p>而 LM35 的转换系数是 10mV 对应 1 摄氏度，因此将电压绝对值转换为 mV 再除以 10 即为温度值。</p>
光强测量	使用光敏模块	使用 analogRead 再进行一定的数据处理。
	Arduino 实现原理	
状态显示功能	小车端通过蓝牙发送字符串并在 App 处理	
按键控制功能	手机通过发送字符串并在小车 arduino 内处理，更新速度值。 Arduino 内有 String 类型的字符串对象，可以字符串进行便捷的处	

万向控制功能	理。			
手动控制左右轮转速	<p>String 有便捷的类型转换功能：</p> <pre>String stringOne = "Hello String"; // using a constant String String stringOne = String('a'); // converting a constant char into a String String stringTwo = String("This is a string"); // converting a constant string into a String object String stringOne = String(13); // using a constant integer String stringOne = String(analogRead(0), DEC); // using an int and a base String stringOne = String(5.698, 3); // using a float and the decimal places</pre> <p>有较多字符串处理函数，如：</p> <table><tr><td><p>描述</p><p>在字符串中选择特定的字符，并返回位置的功能函数(正向)。可在一个很长的字符串中查找这个特别的关键字</p><pre>string1.indexOf(val); string1.indexOf(val,from);</pre><p>参数</p><p>string1：原本的字符串</p><p>val：要找的关键字，可以是char或 string 字符串。</p><p>from：选择性参数，你可以特别指定从那个位置开始寻找这个关键字。</p><p>返回值</p><p>成功返回位置，失败返回-1</p></td><td><p>substring()</p><p>描述</p><p>用来截取字符串中间的某一位置。另外是决定那里结束，</p><pre>string1.substring(from); //返回 from 之后的 string1.substring(from,to); //返回 from 和 to 之间的</pre><p>参数</p><p>string1：原本的字符串</p><p>from：要截取的字符串从第几个字开始。</p><p>to：要截取后面的第几个字</p><p>返回值</p><p>字符串</p></td><td><p>toInt()</p><p>描述</p><p>字符串转成int</p><pre>string1.toInt();</pre><p>参数</p><p>string1：字符串，如"123"</p></td></tr></table>	<p>描述</p> <p>在字符串中选择特定的字符，并返回位置的功能函数(正向)。可在一个很长的字符串中查找这个特别的关键字</p> <pre>string1.indexOf(val); string1.indexOf(val,from);</pre> <p>参数</p> <p>string1：原本的字符串</p> <p>val：要找的关键字，可以是char或 string 字符串。</p> <p>from：选择性参数，你可以特别指定从那个位置开始寻找这个关键字。</p> <p>返回值</p> <p>成功返回位置，失败返回-1</p>	<p>substring()</p> <p>描述</p> <p>用来截取字符串中间的某一位置。另外是决定那里结束，</p> <pre>string1.substring(from); //返回 from 之后的 string1.substring(from,to); //返回 from 和 to 之间的</pre> <p>参数</p> <p>string1：原本的字符串</p> <p>from：要截取的字符串从第几个字开始。</p> <p>to：要截取后面的第几个字</p> <p>返回值</p> <p>字符串</p>	<p>toInt()</p> <p>描述</p> <p>字符串转成int</p> <pre>string1.toInt();</pre> <p>参数</p> <p>string1：字符串，如"123"</p>
<p>描述</p> <p>在字符串中选择特定的字符，并返回位置的功能函数(正向)。可在一个很长的字符串中查找这个特别的关键字</p> <pre>string1.indexOf(val); string1.indexOf(val,from);</pre> <p>参数</p> <p>string1：原本的字符串</p> <p>val：要找的关键字，可以是char或 string 字符串。</p> <p>from：选择性参数，你可以特别指定从那个位置开始寻找这个关键字。</p> <p>返回值</p> <p>成功返回位置，失败返回-1</p>	<p>substring()</p> <p>描述</p> <p>用来截取字符串中间的某一位置。另外是决定那里结束，</p> <pre>string1.substring(from); //返回 from 之后的 string1.substring(from,to); //返回 from 和 to 之间的</pre> <p>参数</p> <p>string1：原本的字符串</p> <p>from：要截取的字符串从第几个字开始。</p> <p>to：要截取后面的第几个字</p> <p>返回值</p> <p>字符串</p>	<p>toInt()</p> <p>描述</p> <p>字符串转成int</p> <pre>string1.toInt();</pre> <p>参数</p> <p>string1：字符串，如"123"</p>		

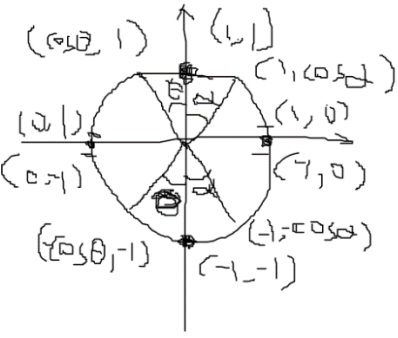
	返回值 整数, 如 123
--	------------------

3.1.3 手机 app 设计思路

总体上是将每种控制信号设置为一个关键字符，当按下按键时调用蓝牙客户端发送该字符。当 arduino 接受到该字符即执行对应的功能。

对于涉及到速度更改的，在关键字符后面再用两个整数拼接成一个长字符串，然后调用蓝牙模块发送该字符串。

以*作为结尾字符,表示指令发送完毕.

音乐播放	三个按键，按键事件分别为发送： “b*” 播放； “z*” 暂停； “n*” 下一首歌	
状态显示功能	接受 arduino 端的字符串，并使用列表显示框显示	
手动开关灯功能	三个按键，分别发送 “j*” “k*” “l*” 表示左、中、右车灯的开关	
按键控制功能	7 个按键，分别为前、后、左、右、停、与左掉头，右掉头为，发送的字符串分别为 “w*” “x*” “a*” “d*” “s*” “q*” “e*”	
万向控制功能	<div>发送字符串格式为：</div> <div>“c_【左轮 analogWrite 值】!【右轮 analogWrite 值】*”</div> <div>即字符 ‘c’ 和两个整数，以一个下划线与一个感叹号分隔，最后以星号结尾 所拼接成的字符串。</div>	<div>使用画布和图像精灵功能，获取图像精灵在画布上的坐标，计算对应的速度。每次监测到拖动图像精灵就发送一次命令字符串。</div> <div>计算到原点的距离，计算径向连线与 y 轴的夹角，再根据象限决定正负，可以对左右轮速度值分别乘上一对系数（如下图括号括起来的一对数），以此分配左右轮速度。让小车运动方向与图像精灵拖动方向相同。</div> 
手动写入左右轮转速		使用文本输入框功能，输入文本后，当按下更新速度按钮，发送命令字符串

3.2 功能实现

3.2.1 无源蜂鸣器音乐播放

蜂鸣器定义与乐谱存储:

```
#define none 0
#define L1 262
#define L2 294
#define L3 330
#define L4 349 //低
#define L5 392
#define L6 440
#define L7 494

#define M1 523
#define M2 587
#define M3 659
#define M4 698 //中
#define M5 784
#define M6 880
#define M7 988

#define H1 1046
#define H2 1109
#define H3 1318
#define H4 1397 //高
#define H5 1568
#define H6 1760
#define H7 1976

#define NOTE_C0 0
#define NOTE_CL1 131
#define NOTE_CL2 147
#define NOTE_CL3 165
#define NOTE_CL4 175
#define NOTE_CL5 196
#define NOTE_CL6 221
#define NOTE_CL7 248

#define NOTE_C1 262
#define NOTE_C2 294
#define NOTE_C3 330
#define NOTE_C4 350
#define NOTE_C5 393
#define NOTE_C6 441
#define NOTE_C7 495
```

```

#define NOTE_CH1 525
#define NOTE_CH2 589
#define NOTE_CH3 661
#define NOTE_CH4 700
#define NOTE_CH5 786
#define NOTE_CH6 882
#define NOTE_CH7 990

int
table[22]={none,L1,L2,L3,L4,L5,L6,L7,M1,M2,M3,M4,M5,M6,M7,H1,H2,H3,H4
,H5,H6,H7};
int
table2[22]={NOTE_C0,NOTE_CL1,NOTE_CL2,NOTE_CL3,NOTE_CL4,NOTE_CL5,NOTE
_CL6,
NOTE_CL7,NOTE_C1,NOTE_C2,NOTE_C3,NOTE_C4,NOTE_C5,NOTE_C6,NOTE_C7,NOTE
_CH1,NOTE_CH2,
NOTE_CH3,NOTE_CH4,NOTE_CH5,NOTE_CH6,NOTE_CH7};
//该部分是定义是把每个音符和频率值对应起来，由于两首乐曲使用两个不同的调，
因此需要两个频率对照表

//
int note=0;
int lenth1;
int lenth2;
/*
 * music1 是《蜜雪冰城》，music2 是《遇见》，music3 是《最伟大的作品》
 * 经过测试，动态内存占用率>85%时会导致运行问题，且乐曲存放将消耗大量动态内
存，
 * 因此 music2 被注释掉以确保足够的空间
 */
//频谱'
char
music1[]="589:<<=<:9889::989098:<<=<:9889::9980;;<==<<:89089:<<=<:988
9::9988";
//char
music2[]="000<::<99:99880876787889::00<::<99:998808767877889800<=>;:9
8><::<99";
char
music3[]="0:=AA@B0:==@@?A0:=??>@0???@>0:=AA@@E?EA@A?0?>?A@=0:??>=0:::A@
B0:==@@?A0:=";
//节拍

```


[illegible]

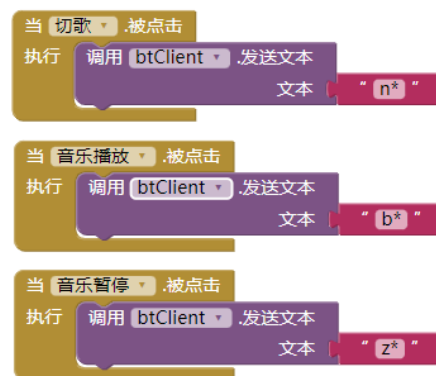
由于乐曲存放使用全局变量，为了节省有限的空间，我们使用字符串存放乐曲的长度和音符频率，并在播放函数中进行处理以转换成正确的字符串。音乐播放部分的处理如下：

每次 loop 只播放一个音符, note 是记录歌曲的第几个音符。music[note]就是当前所要播放的音符所对应的字符 char, 字符 char 的 ascii 码值减去 48, 即为对应音符频率表 table 中的下标。

而 `music_delay[note]` 是当前音符所对应的长度（以 16 分音符为单位长度），也以字符变量存储，转换为整数需要减去 ‘0’ 字符的 `ascii` 码再加 1。

```
void playMusic()
{
    switch(whichMusic)
    {
        case 0:
            tone(Buzzer, table[music1[note]-48]); //驱动蜂鸣器
            delay((music_delay1[note]-'0'+1) * 100); //节拍实现
            noTone(Buzzer); //停止蜂鸣器
            break;
        case 1:
            tone(Buzzer, table2[music3[note]-48]); //驱动蜂鸣器
            delay((music_delay3[note]-'0'+1) * 135); //节拍实现
            noTone(Buzzer); //停止蜂鸣器
            break;
    }
    note=note+1;
    if(note==lenth1) note=0;
}
```

app 端音乐播放的三个按钮与其逻辑如下：



3.2.2 蓝牙控制

arduino 接收蓝牙指令的函数如下，使用 Task 使该函数每 10ms 执行一次且永远执行
任务日程所使用的<TaskScheduler.h>库的使用方法已经在【2.7 遥控&循迹——模式切
换】中说明

BT.read() 实际上每次只能接受一个字符，因此使用 command 字符串，当未接收到指令结
尾标志 '*' 时，继续接收指令并将新的字符加在 command 字符串的末位，如果接收到 '*'，
则调用 Parse() 函数对 command 字符串解析。

```
#include <SoftwareSerial.h>
#include <TaskScheduler.h>
SoftwareSerial BT(4,5);

char c;
String command; //接收的指令

void UpdateState()
{
    while(BT.available())
    {
        c=BT.read();
        if(c=='*')
            Parse();
        else
            command=command+c;
    }
}

Task Control(10,TASK_FOREVER,&UpdateState);
Scheduler Sch;

void setup()
{
    Sch.init();
    Sch.addTask(Control);
    Control.enable();
}
```

```
}
```

Parse 函数如下，如果接受到音乐播放相关的指令，就改变 isMusicPlay 标志和 whichMusi 变量。如果接受到 'c' 开头的字符串，则调用 change_speed() 函数改变速度。

```
void Parse()
{
    //Serial.println(command);
    //Serial.println(command[0]);
    switch(command[0])
    {
        case 'b':
            isMusicPlay=1;
            note=0;
            break;
        case 'z':
            isMusicPlay=0;
            note=0;
            break;
        case 'n':
            whichMusic=(whichMusic<1)?(whichMusic+1):0;
            break;
        case 'c':
            change_speed();
            command="";
            break;
        default:
            break;
    }
    return;
}
```

change_speed() 函数如下，后面的【手动设置左右轮转速】和【万向控制模式】功能，在 arduino 端都基于此函数。

其中使用了 String 对象的 substring 方法与 indexOf 方法来拆分字符串。字符串中的分隔字符下划线和感叹号此时作为标志，便于查找和拆分。

counterflag 和 counterlflag 的作用是在运动函数中是否启用计数器。因为小车左右轮后退时不能正常使用 analogWrite 控制电机转速（与使用的是否是 pwm 口有关，也与 MsTime2.h 库存在的情况下，arduino 内置计时器使用有优先级有关）。因此使用计数器间断性使某个轮子停转来制造两个轮子的差速。

```
void change_speed()
{
    static bool continuous_0=false;
    String Left =
    command.substring(command.indexOf('_')+1,command.indexOf('!'));
    //获取子字符串
```

```
String Right = command.substring(command.indexOf('!')+1);  
int ltemp=Left.toInt();  
int rtemp=Right.toInt();
```

```
if(ltemp==0&&rtemp==0)  
{  
    if(continuous_0==true) return;  
    else  
    {  
        continuous_0=true;  
        counter=0;  
    }  
}  
else  
{  
    if(continuous_0==true)  
        continuous_0=false;  
}
```

//continuous_0 是监测如果是反复接受到设置速度为 0，则避免反复执行节省时间。

```
if(ltemp>=0){  
    left_speed_back=0;  
    left_speed_forward = ltemp;    counterflag=false;  
    counterlflag=false;  
}else{  
    left_speed_forward=0;  
    left_speed_back = -1*ltemp;  
    counterlflag=true;  
}
```

```
if(rtemp>=0){  
    right_speed_back=0;  
    right_speed_forward = rtemp;  
    counterflag=false;  
    counterlflag=false;  
}else{  
    counterflag=true;  
    right_speed_forward=0;  
    right_speed_back = -1*rtemp;  
}
```

```
if(counterlflag&&counterflag)  
{
```

```

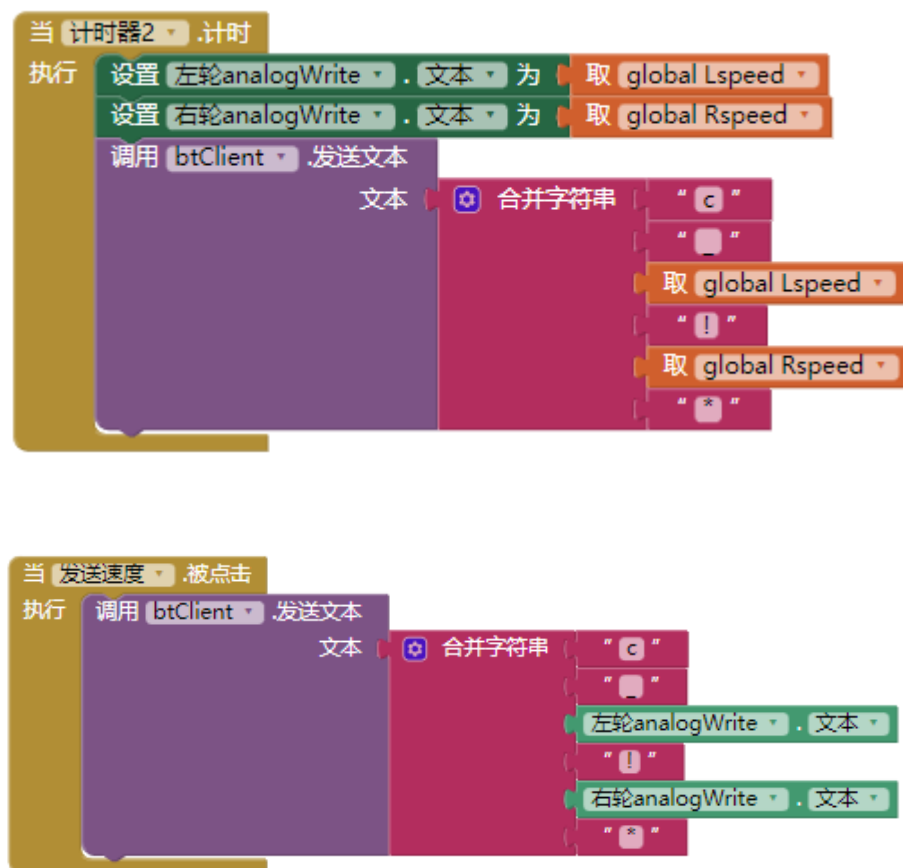
        counterlflag=(right_speed_back<left_speed_back)?false:true;
        counterflag=(right_speed_back>left_speed_back)?false:true;
    }//counterlflag 和 counterflag 分别控制左右轮后退的计数器是否启用

    forward(1);//调用运动函数，forward 函数的速度参数可以调整
    return;
}

```

在 app 端，用蓝牙下发控制速度指令的逻辑如下：

（计时器 2 为 53ms 执行一次，仅在万向控制时启用）



3.2.3 左右转向灯与倒车灯

在运动函数中加上开关灯函数即可，下面以停车函数为例

```

void brake(int time) //刹车，停车
{
    L_led_close();//左灯关
    R_led_close();//右灯关
    B_led_open();//尾灯开
    digitalWrite(Right_motor_go,LOW);
    digitalWrite(Right_motor_back,LOW);
    digitalWrite(Left_motor_go,LOW);
}

```

```

digitalWrite(Left_motor_back,LOW);
delay(time * 100); //执行时间，可以调整
}

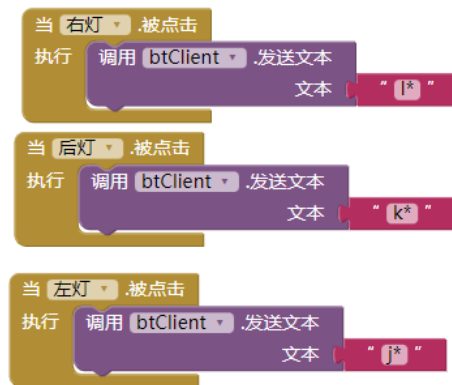
void L_led_open(){
    digitalWrite(l_led,HIGH);}
void R_led_open(){
    digitalWrite(r_led,HIGH);}
void B_led_open(){
    digitalWrite(forward_led,HIGH);}
void L_led_close(){
    digitalWrite(l_led,LOW);}
void R_led_close(){
    digitalWrite(r_led,LOW);}
void B_led_close(){
    digitalWrite(forward_led,LOW);}

```

app 中的三个按钮如下



逻辑与音乐播放类似



3.2.4 测速

中断函数的原理与使用方法已经在设计思路中介绍, 下面给出代码.

```

#include <MsTimer2.h>
void left_motor() { //触发函数
    motor1++;
}

void right_motor() { //触发函数
    motor2++;
}

```

```

void flash()//计算速度的触发函数
{
    speed1=motor1;//speed 是 1s 内码盘遮挡次数
    speed2=motor2;
    motor1=0;    //重新定义 motor1 的值
    motor2=0;    //重新定义 motor1 的值
}

void setup()
{
    attachInterrupt(0,left_motor, RISING); //D2 对应左轮
    attachInterrupt(1,right_motor,RISING); //D3 对应右轮
    MsTimer2::set(1000, flash); // 中断设置函数，每 500ms 进入一次中断
    MsTimer2::start(); //开始计时
}

```

3.2.5 温度测量

LM35 代码如下

```

float temp=analogRead(temp_sensor)*500.0/1024.0;

```

3.2.6 光强测量

光敏电阻的输出有负值，且光强越强，输出值越低。因此做了粗糙的数据处理以变成光强数据。

```

light = analogRead(light_sensor);

if(510-light < 0) light = 0;
else light = 510-light;

```

3.2.7 状态显示功能

状态显示功能中读取温度值、光强值、速度值，并拼接成以下划线分隔的字符串。

```

void UploadState()
{
    float temp=analogRead(temp_sensor)*500.0/1024.0;

    int speed_av,len,light;
    len=State.length();
    speed_av = (speed1+speed2)/2; //左右轮平均速度
    light = analogRead(light_sensor);

    if(510-light < 0) light = 0;
    else light = 510-light;
}

```



```

String string1 = String(speed_av);
String string2 = String(temp);
String string3 = String(light);
State = String(string1+'_'+string2+'_'+string3);

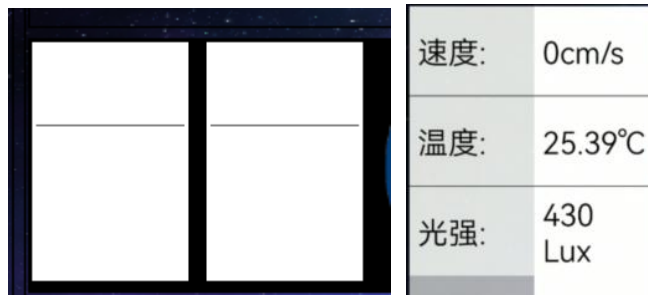
BT.println(State);
}

Task Upload(100,TASK_FOREVER,&UploadState);
Scheduler Sch;

void setup()
{
  Sch.init();
  Sch.addTask(Upload);
  Upload.enable();
}

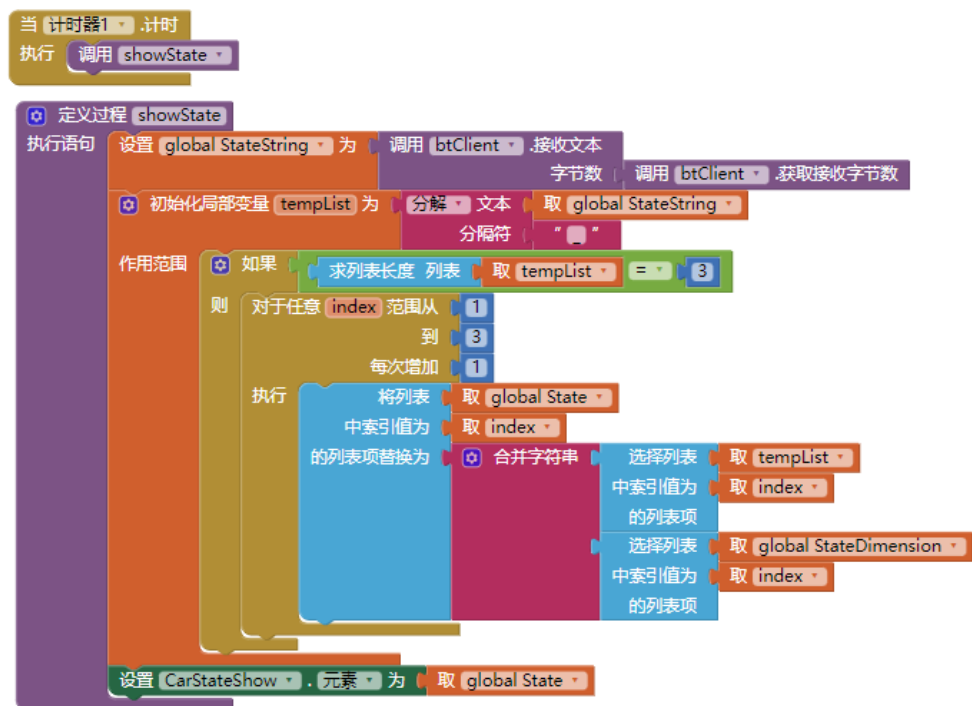
```

app 使用定时器+蓝牙客户端周期性接受文本，再调用列表显示框显示：
两个列表显示框如下左图，而手机上效果为没有中间的缝隙，如下右图



逻辑如下：

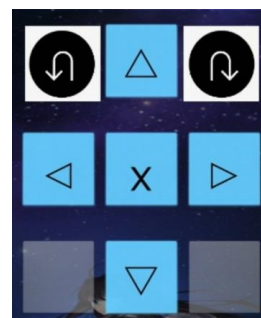




3.2.8 按键控制功能

如右图。

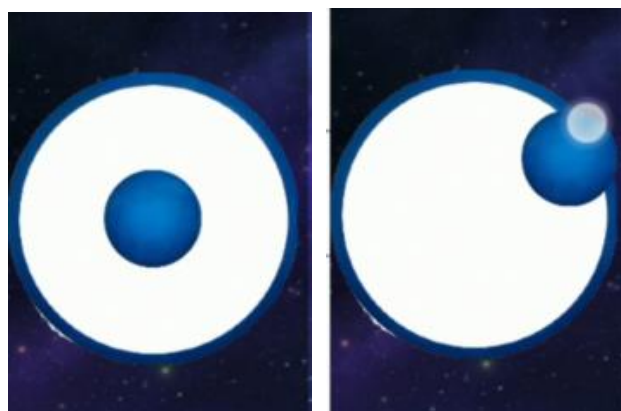
其逻辑，以向左掉头的逻辑为例：



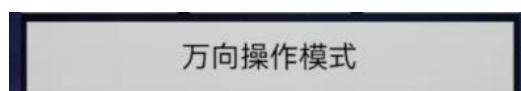
3.2.9 万向控制功能

默认情况，图像精灵处于画布中心，表示速度为 0，静止，如左图。

拖动时的效果如右图。



使用万向操作模式可切换按键操作与万向操作。



在万象操作模式下，下方的文本框会分别显示下方的左轮 analogWrite 值，右轮 analogWrite 值，以及计算过程。

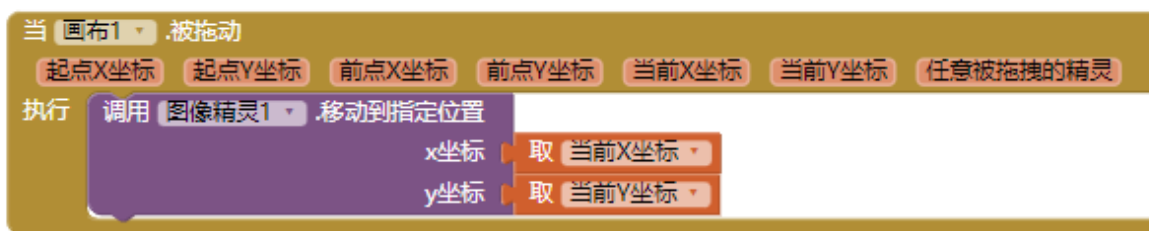
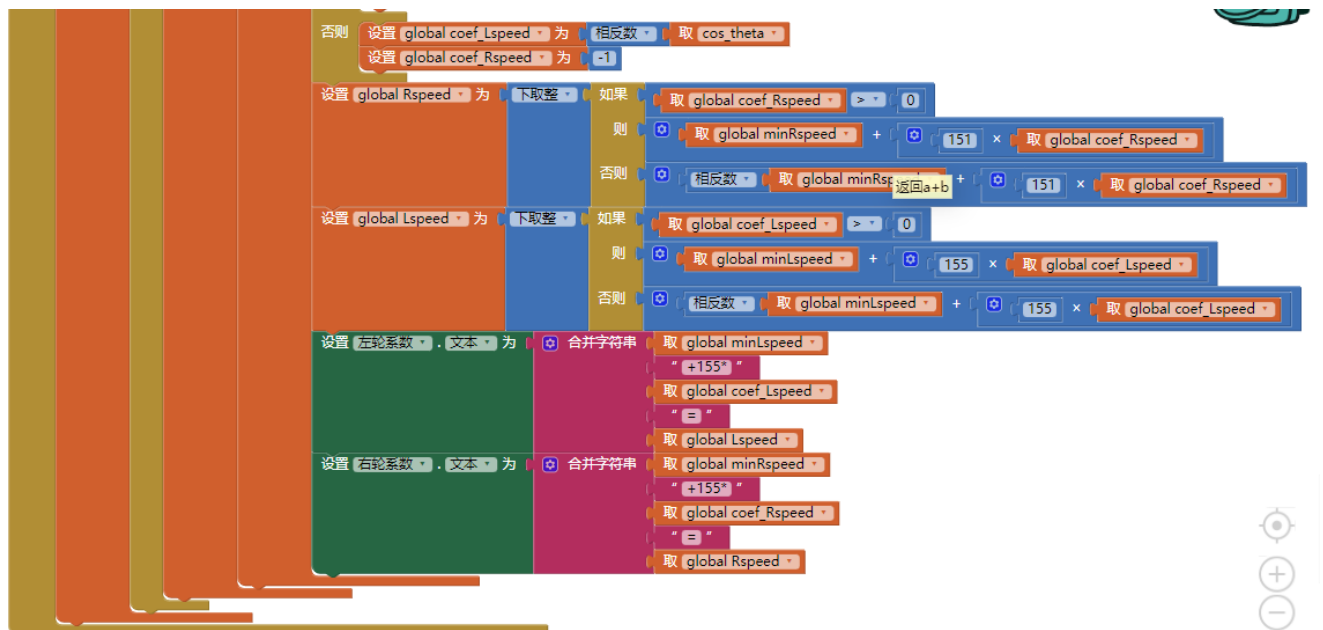
如右图，此时图像精灵拖动方向为右前方，则小车的行驶方向应该为右前方。

下方的四个文本框可看出，此时左轮为 255 前进，右轮为 193 前进，左轮前进速度更快，向右前方行驶，正确。

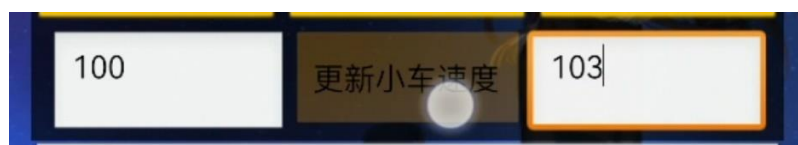


万象操作模式的逻辑如下：





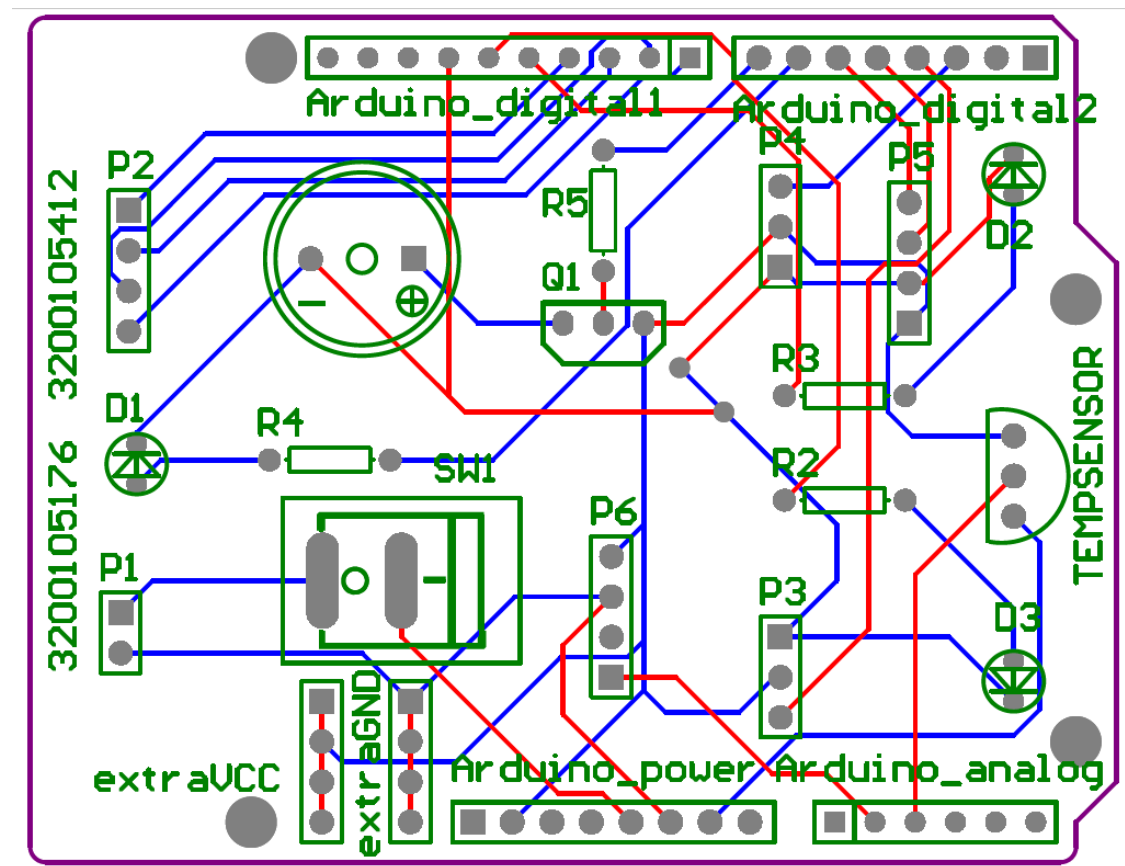
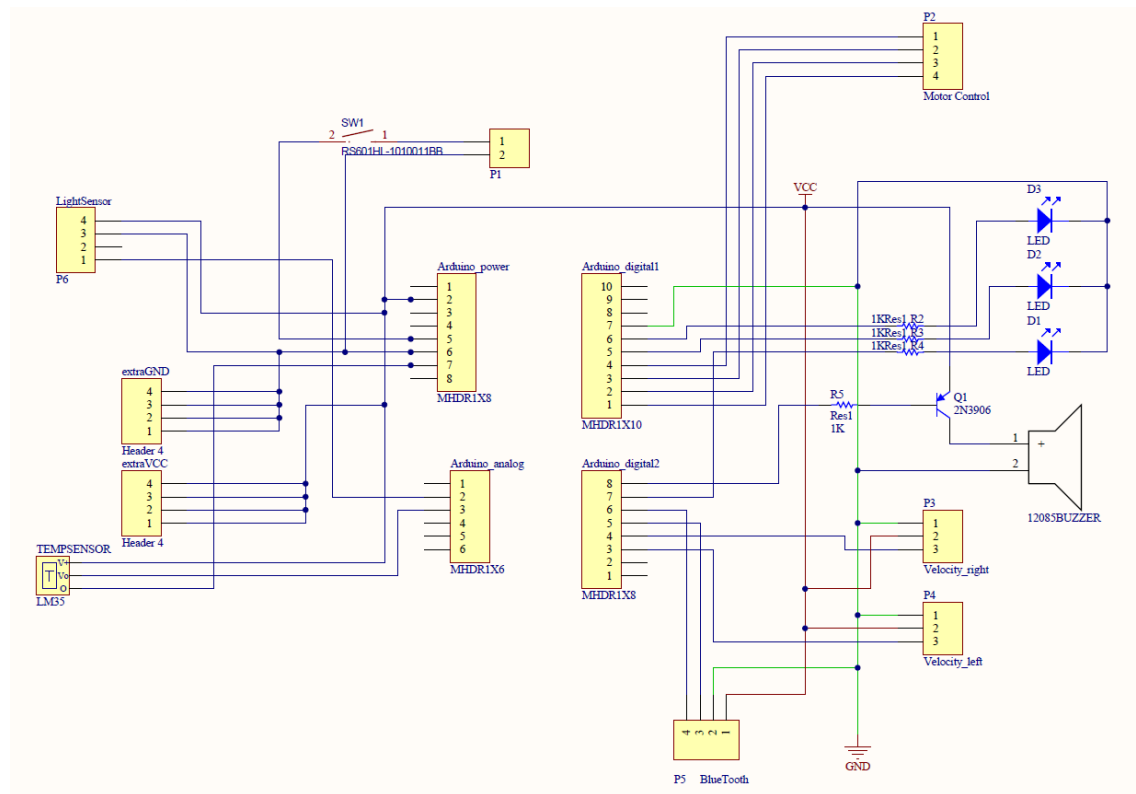
3.2.10 手动控制左右轮转速



逻辑略，即按下按钮时调用蓝牙客户端发送文本。

3.3 PCB 设计

PCB 原理图与布线如下



四、实验总结与心得体会

总结：

本次智能移动设计实验要求我们自主提出功能需求，再对应提出实现方案，较好地锻炼了我们的工程思维与动手能力，同时对电路设计、Arduino 程序开发等能力起到了融合贯通的锻炼作用。

在功能需求提出阶段，我们较好地考虑到了功能的可实现性，对于提出的功能均提前搜集了资料、购置了元器件、并在面包板上进行了验证。我们体会到在进行开发前进行方案预研是非常重要的，我们前期积极完善的资料收集和扩展知识学习极大地节省了后期实现的时间。

在 Arduino 程序设计方面，我们感受到我们的能力还有很大欠缺。对于 Arduino 等单片机的特性我们了解尚浅，比如本次实验中就涉及到了 `analogWrite()` 输出口的问题，以及 Arduino 内置时钟只能执行一个计时任务，不同的功能调用内置时钟有优先级，因此例如使用 `MsTimer2.h` 库时（调用了 `Timer2` 库），D3，D11 不能作为 PWM 引脚使用，因此导致右轮后退时无法进行 `analogWrite`，给我们带来了很大麻烦。

对于 app 设计，我体会到 android 应用开发非常复杂，AppInventor 2 提供的开发工具实现功能虽然方便但高度受限。要想掌握 App 的开发还需要更加深入的学习。

总的来说，我体会到完成综合工程能够增长许多知识。许多有价值的知识，都是在遇到问题-搜索解决方案后才了解到的。作为信息类的工科学子应该重视工程实践与理论结合，多做项目，才能完善自己的专业能力。

心得体会：

经过智能移动系统设计这一课程，我有如下感受：

首先，我们应该树立正确的学习态度，认清自己肩负的使命，认识到自己的学习成长与国家人民命运的关联。在当今时代，单片机与、电子电路技术得到了广泛应用，是集成电路技术、信息与通信技术的发展基石，但我们在电部分领域还与国外先进水平有一定差距。我们作为面临百年时代变局的一辈应敢于担当、勇于担当、能够担当起发展电子科技产业的重任。

其次，我们应有严谨务实的科学态度，要有追求真理、勇于探索的科学精神，和认真细致、精益求精的工程精神，做到孜孜不倦、敢于挑战、不惧怕失败。在简单的智能移动设计实验中我们就遇到了许多问题，不得不反复调试、检查、分析，在未来的工作或者科研中可想而知将会遇到更艰难的问题，更经常地遭遇挫折和失败。如不能对科学求真求实、对工程求稳求细、对失败越挫越勇，则断然不能在未来的事业中克服困难。

同时，失败也同时意味着经验的积累与学习的机会与方向。比如这次实验中，我们遇到了很多问题。比如由于 Arduino 动态内存只有 2kB，全局变量占用太多导致无法稳定运行，因此我们想到了将整数和浮点数的乐曲存储数组改变成字符串存储，而用音乐播放函数稍加处理的方式，并使用了 C 语言程序自动转换乐曲为字符串。遇到问题并解决问题的过程增强了我们的能力。

再者我们应具有责任感和使命感，有较强的团队合作精神，勇于担当。在智能移动设计实验中我和同伴密切合作，在操作时各自分工，在调试与寻找 bug 时彼此提出想法互相启发，使得项目事半功倍。我深切体会到团队合作精神是我们必须具有的优秀品质。

在学科专业知识方面，我也有一些更为具体的体会：

我们应该了解电子电路与其他科学技术的关系，为学习其他专业理论知识做好准备；对电路设计的知识内容及应用有较全面系统的认识。我们应灵活结合各专业的知识，既采用计算机编程的思维方式思考各个端口信号的处理与交互，又要时刻意识到基于电路的信号有其实际物理意义即电压，受到许多复杂电路环境的影响。

因此我们应在理解和掌握电子电路知识中的基本概念、基本原理的基础上，建立正确的逻辑思维方法，建立综合运用工程基础理论和技术手段解决问题的能力，提高自身设计和实施工程基础实验的能力。

我们还应该具备进行模拟电路和数字电路工程计算、功能分析和解决实际问题的能力；进而掌握科学的学习方法、建立独立获取知识的能力。对于工程学科最重要的不是掌握某一方法或技术，而是能够以问题为导向，明确自己需要的知识、技术、工具并能够自行学习。只有拥有这样的能力，才能持续地更新知识，实现突破，进而发展行业，贡献社会。