

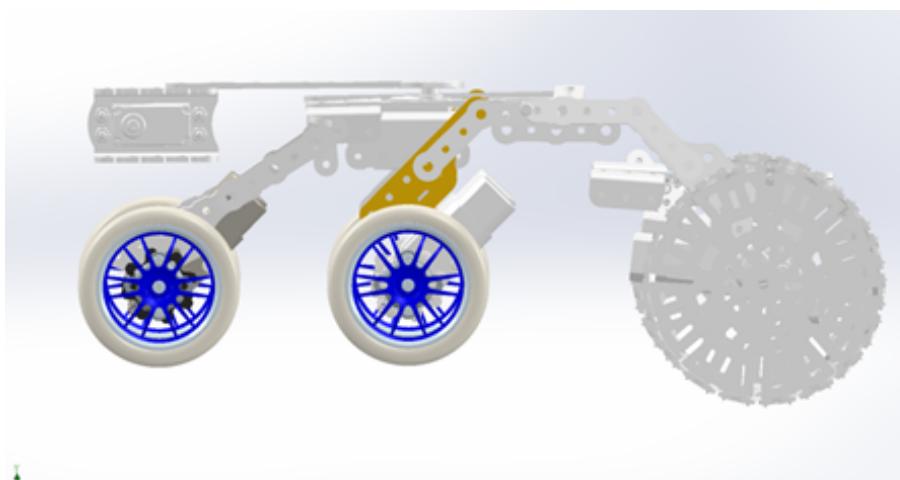
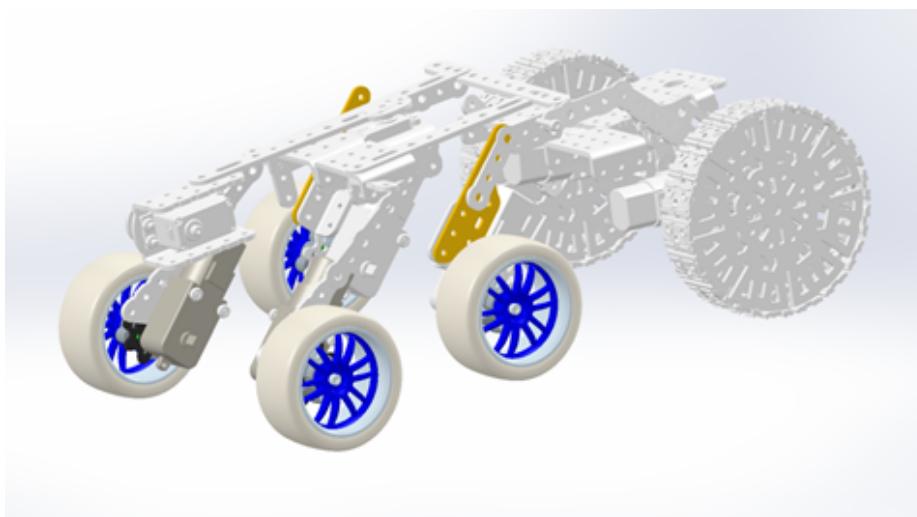
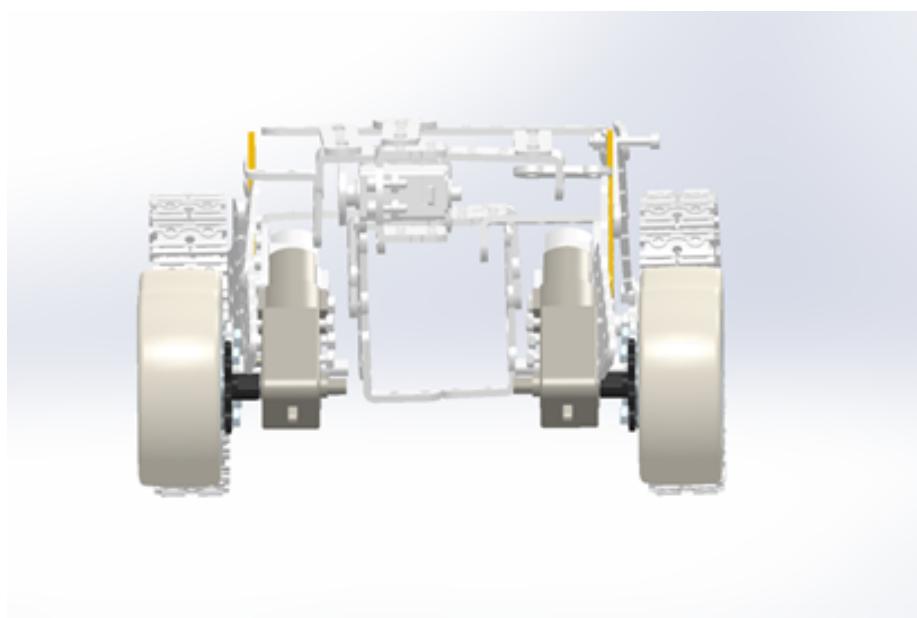
第 1 章 作品名称与团队简介.....	错误!未定义书签。
第 2 章 作品简介	2
第 3 章 结构方案设计	4
3.1 底盘车架设计	4
3.2 循迹设计	7
3.3 弹珠运送与倾倒	8
3.4 颜色识别	9
第 4 章 控制方案设计	9
4.1 BASRA 主控板	9
4.2 BIGFISH 扩展板.....	11
4.3 循迹转弯	11
4.4 颜色识别	14
4.5 物品倾倒	17
4.6 直流电机模块.....	17
4.7 任务结束策略.....	17
第 5 章 创新设计说明	17
5.1 颜色识别优化.....	17
5.2 六轮悬挂式	18
第 6 章 设计过程	18
6.1 结构迭代.....	18
6.2 程序迭代	19
第 7 章 团队评价	错误!未定义书签。
7.1 队员评价	错误!未定义书签。
7.2 指导老师评价.....	错误!未定义书签。

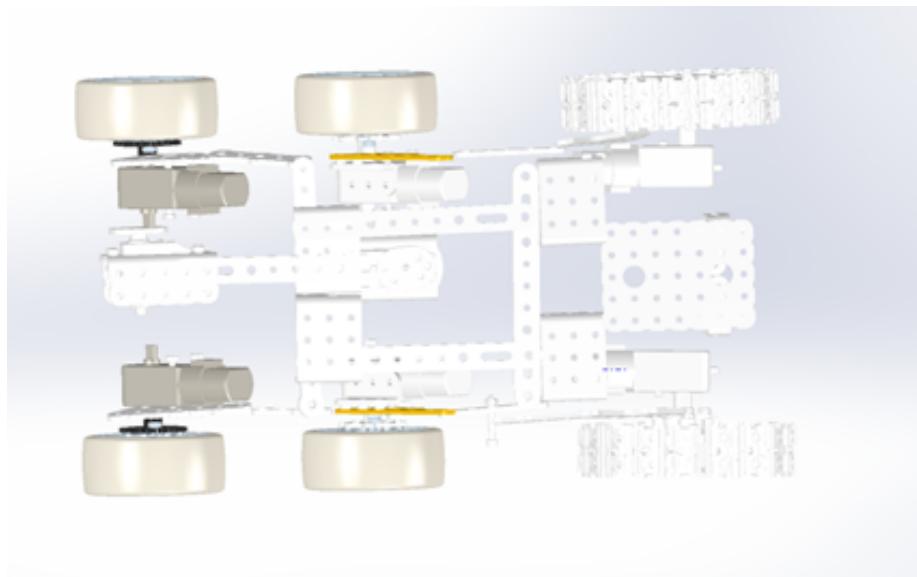
第1章 作品简介

根据比赛相关要求，本团队在设计小车时综合考虑场地构造、障碍形态等因素，经过数次试跑、改装，遵循车体轻盈、受重均匀、越障顺畅的原则，确定了小车的六轮悬挂式结构，通过 IIC 颜色传感器进行颜色识别，通过一组舵机来控制载盘的移动与倾倒。

为保证顺利跑完全程，所设计的小车需首先保证巡线准确，因此安装三个灰度传感器于车体前下端距地面约 2.00cm 处，灰度传感器采用三路循迹方案，通过 ADC 转换来接受传感器信号，通过灰度传感器来修正小车循线、转向、停车等功能，控制小车的精准走向。小车整体采用悬挂式六轮驱动设计，保证小车轻盈灵活的同时也保证其在越障过程中的稳定。考虑到小车越障时的顺畅性，将前四轮设计为一体悬挂式结构使其在上下台阶时拥有更高的灵活自主性，并且能最大化减少车体在越障过程中产生的颠簸。车体前两轮采用大轮缠绕履带，后四轮均采用 1: 10 模型轮胎。车体前半部分安装 IIC 颜色识别模块，转动模块上的定位器与镜头限位器调整检测距离到色卡安放位置进行精准颜色识别，颜色准确识别完毕后进而执行倾倒任务。

由于载盘固定安装后在车辆经过障碍物或转弯时极易出现不稳定情况，所以经研究，我们利用精巧的车体结构最大化减小安装载盘平台的倾斜角度：我们根据单节台阶的长度来调整小车的车轮间距，使小车上安放载盘的平台的倾斜角度始终保持在较小范围内，同时控制小车移速减慢以尽量保证弹珠始终保持在载盘内。对于最后的倾倒部分，利用一组 180 度的伺服电机（舵机）驱动摇臂进行旋转倾倒。当颜色识别模块识别完毕，其中 1 号舵机会将 2 号舵机舵机与载盘旋转至料盒上方，2 号舵机进行倾倒动作，载盘将在舵机的控制下在对应位置进行旋转，进而将玻璃弹珠准确倾倒入对应的颜色料盒内。整车 3D 结构图如所示：





第2章 结构方案设计

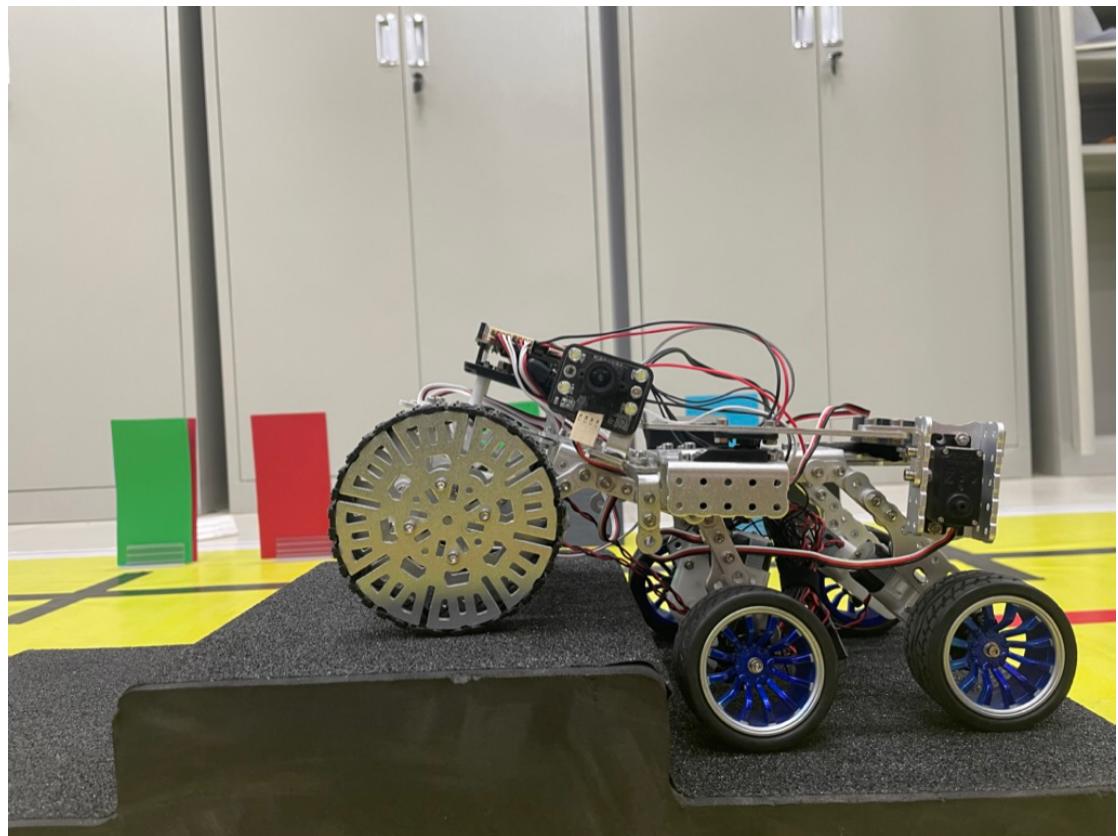
2.1 底盘车架设计

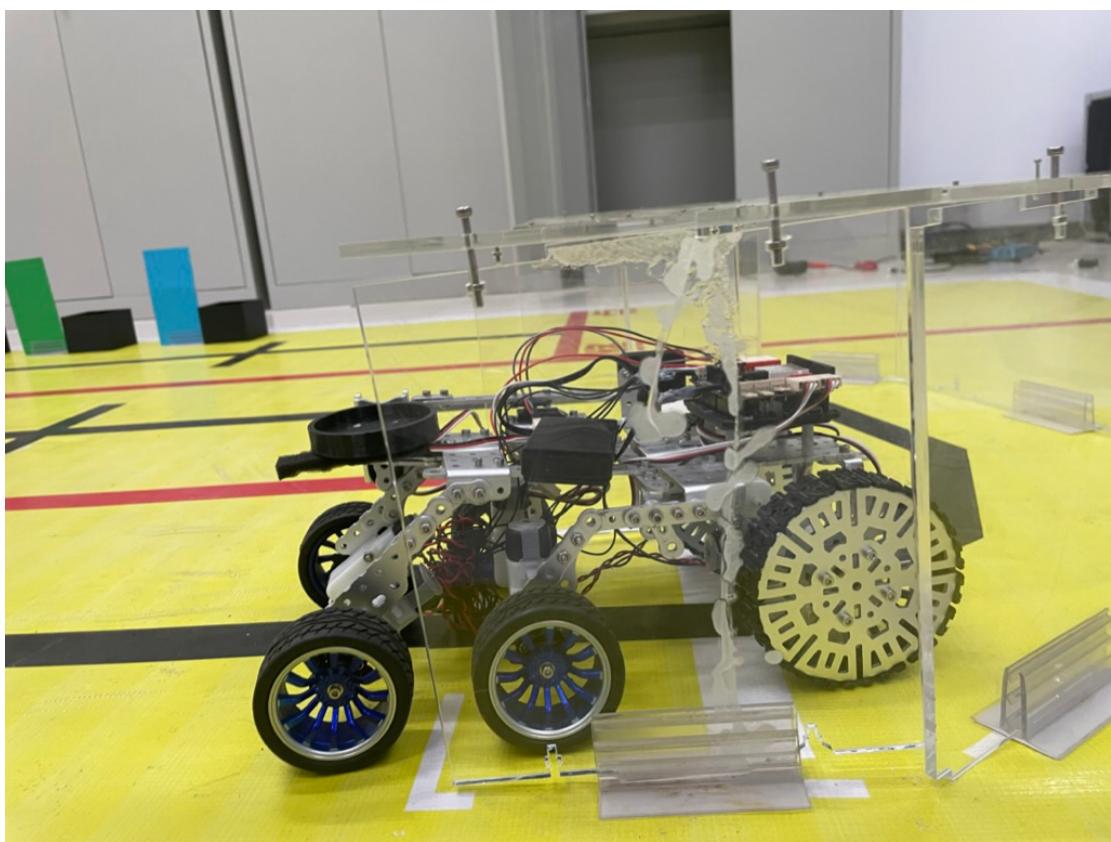
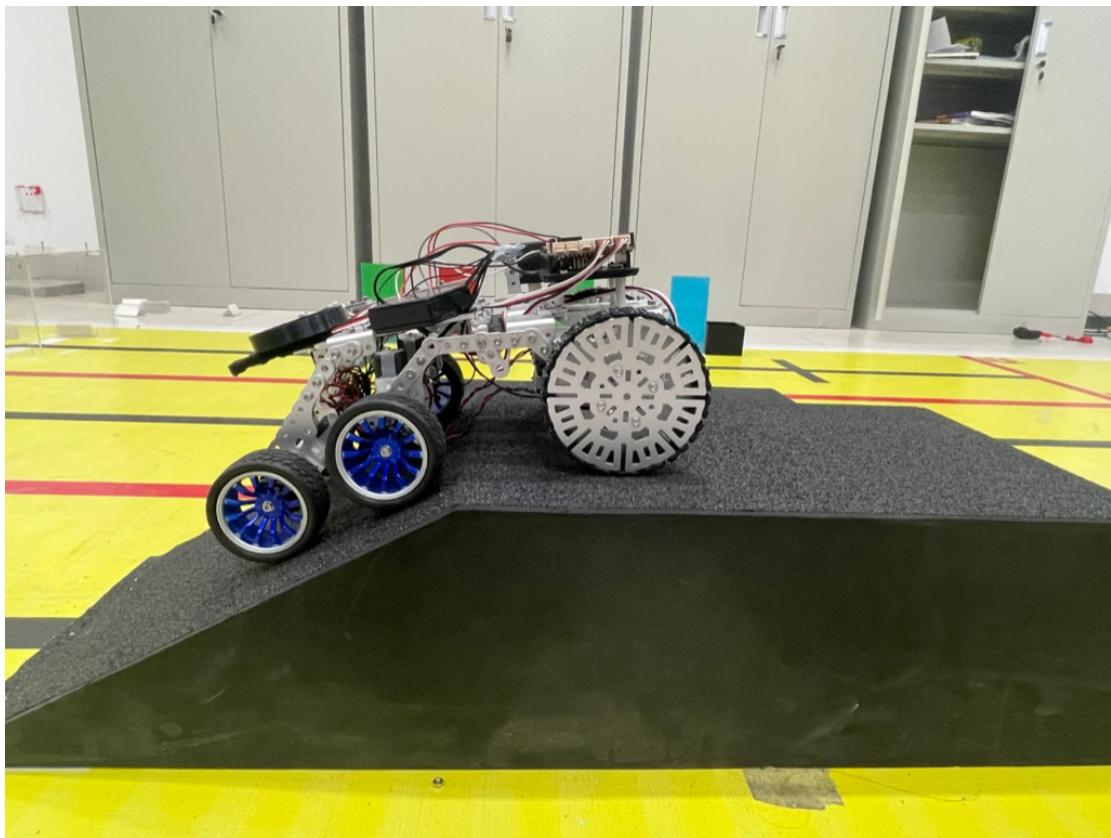
根据赛项规则的介绍，对小车的整体长宽都有着极其严格的限制，如窄桥和台阶的宽度只有 25cm，管道的内壁只有 26cm，这意味着小车的轮间距最大不能超过 25cm，且管道内的高度为 20cm，也就代表着小车整体最高高度不能高于 20cm，并且小车的高度过高也会造成车体重心上移导致小车上坡时发生倾覆状况。

针对全地形车在越过障碍时容易出现因灰度传感器的安装位置距离地面过近对车辆前进产生阻碍的情况或因与地面距离不合造成循迹功能发生问题而偏离路线或者转弯受阻等问题，我们采用了前轮履带覆盖大轮，连接轮毂单元，扩大轮子直径的方案，该方案在增强爬坡和上台阶的能力的同时提高了移动速度。

针对增强车辆的通过性问题，我们采取了悬挂式六轮结构方案，我们根据多次实验评估发现，如果轮间距在 25cm 的尺寸左右，那对小车的上下坡姿态的要求会变得极为苛刻，因为一旦车体出现偏移，宽度过大的小车必然会出现轮胎掉出障碍物的情况，导致小车发生倾覆。而在其中的管道障碍中更是明确的限制了小车的宽度不能超过管道的内宽，不仅于此，我们根据多次实验发现，管道内壁为 26cm，而一旦小车的轮间距宽度超过 24cm 且车身长度长于 40cm 时，小车在

通过管道的过程中进行转向时，小车的轮胎与车身就极其容易与内壁发生碰撞摩擦从而导致无法正确地进行转向，甚至发生整个车体卡在管道内部的情况。针对转弯时弯道中存在的限制，我们综合小车的爬坡能力和过弯速度合理设计了车架的长宽高尺寸。如图所示，我们的车辆对障碍物有良好的通过性：

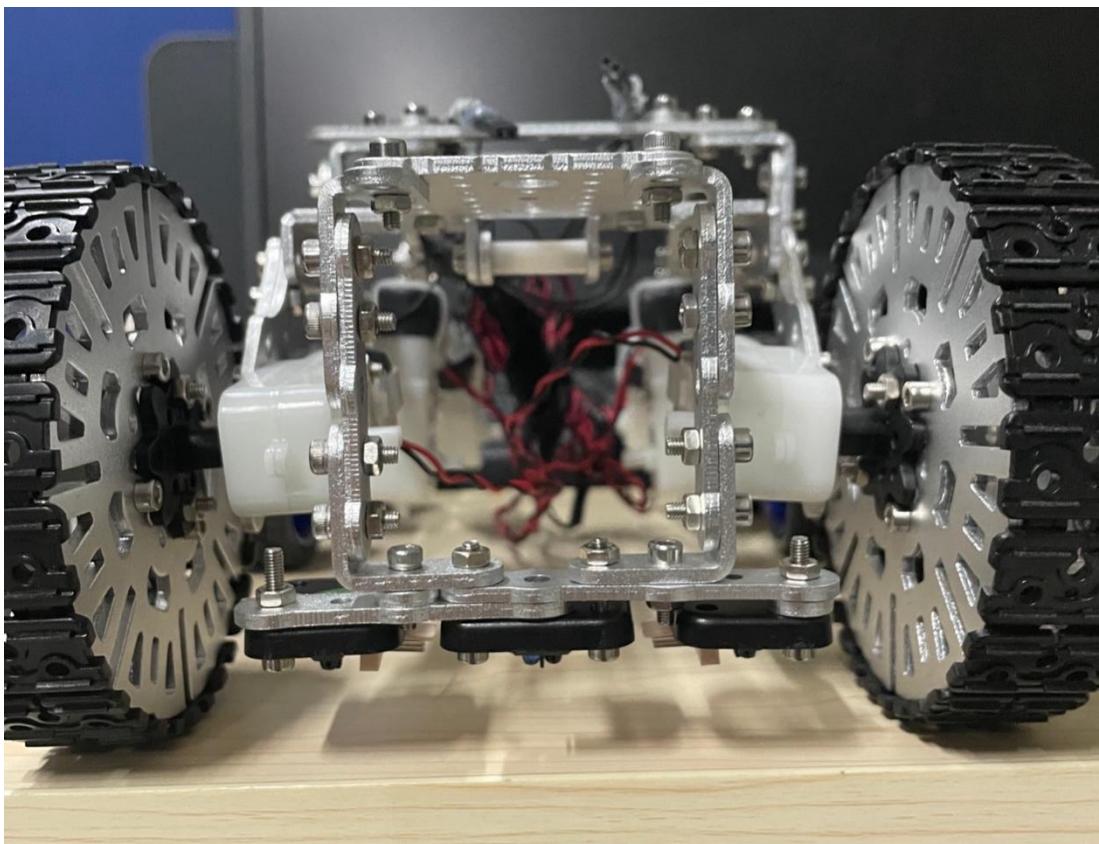




2.2 循迹设计

在我们前期的实验阶段中，灰度传感器的位置在前轮处不断调整，这一结构位置往往在经过管道、窄桥与台阶时出现不适应现象。灰度传感器的安装位置过于靠后会导致小车在经过管道时无法及时调整姿态而碰壁，而安装位置过于靠前则会因小车前轮高度有限导致小车在越障时与障碍物相碰卡住，前轮无法继续前进。灰度传感器的高度位置也同样影响小车的循迹功能，具体体现在小车转弯处，若高度不适宜小车将会偏离赛道黑线，或者在越障时与障碍物相撞卡住。

根据这一现象，我们在设计结构时着重考虑灰度传感器的安装位置，因此传感器的安装位置不能过于靠后，至多也不能超过小车前两轮，应处于前两轮中心轴线偏前处，而其最佳安装高度应位于距地面 0.7cm~2cm 处，安装位置如图所示。

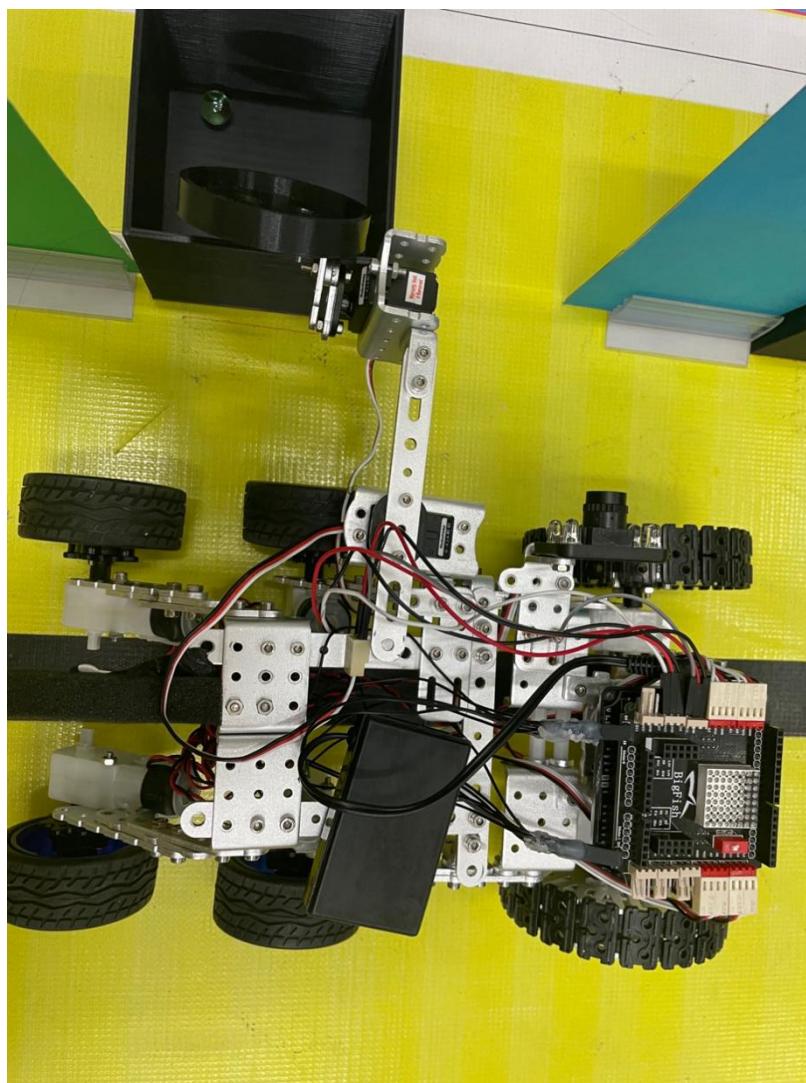


本设计采用三灰度差速循迹策略，每个灰度传感器包含一对红外对管，利用红外光在不同色值材料上的反射率不同进行对不同颜色的数值读取，并作为灰度传感器的输出。我们采用了同时通过动态调整两侧轮胎转速来进行差速转弯与循迹过程，使得小车在管道转弯时具有良好的适应性，且能够保证小车在上坡或越障时保持两侧轮胎位置的一致。

2.3 弹珠运送与倾倒

由于赛道中存在斜坡与台阶障碍物，在通过障碍物时车辆会发生大幅摆动，出现颠簸的情况，而弹珠在小车运动时受力不均出现相互碰撞且存放弹珠的载盘周围无外物阻挡，因此在没有稳定措施的情况下弹珠极易脱离载盘。我们决定利用精巧的车体结构最大化减小上下台阶时载盘安装平台的倾斜程度，根据单节台阶的长度来调整小车的车轮间距，使小车上安放载盘的平台的倾斜角度始终保持在较小范围内，同时控制小车移速减慢以尽量保证弹珠始终保持在载盘内。

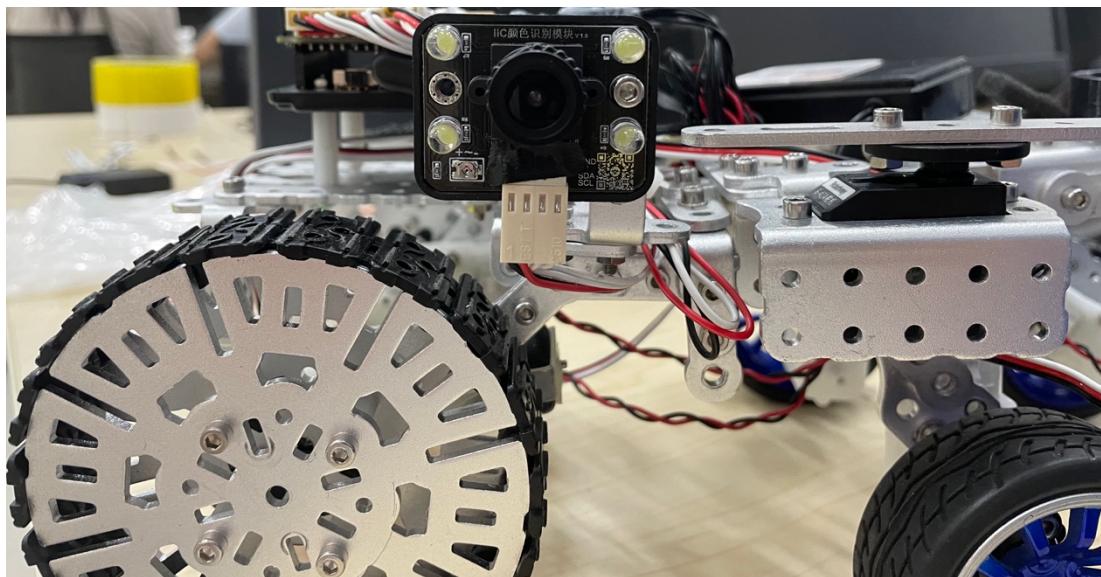
在识别到色卡颜色后，因为我们舵机安装在后半部分，所以我们在识别到颜色后控制车辆向前行驶一段距离后停车，再利用舵机驱动摇臂进行倾倒，摇臂通过舵机的转动来实现摆动，根据舵机转动角度的不同，可以控制摇臂转至任意角度。通过舵机将载盘旋转至料盒上方，另一个舵机进行载盘的旋转，将玻璃弹珠准确投掷在料盒内。



2.4 颜色识别

传统的颜色识别效率较低，识别一次时间长，误差较大，对任务完成的效率产生了很大的影响，为此我们利用灰度传感器 TCS34725，其通过 IIC 总线接口输出数字色彩信号，通过集成 RGB 颜色滤光片与光电二极管，实现对环境光的 RGB 三原色成分检测，并通过 IIC 接口输出数字量化的颜色数据，为环境光颜色的检测与识别提供基础。

我们将颜色传感器放置于车体前半部分左侧，通过多次实验来确定其合适位置以便正确识别颜色。

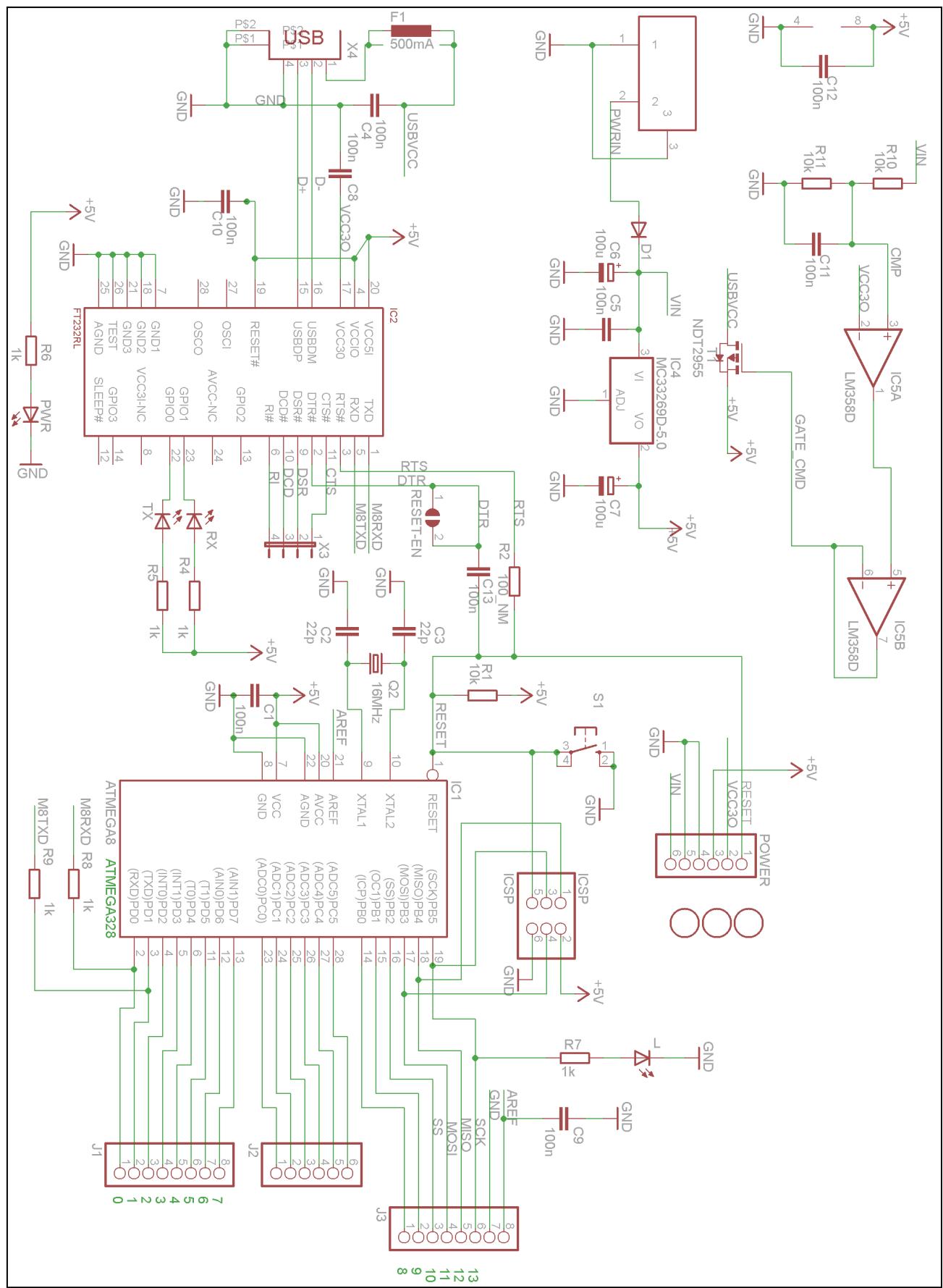


第3章 控制方案设计

3.1 Basra 主控板

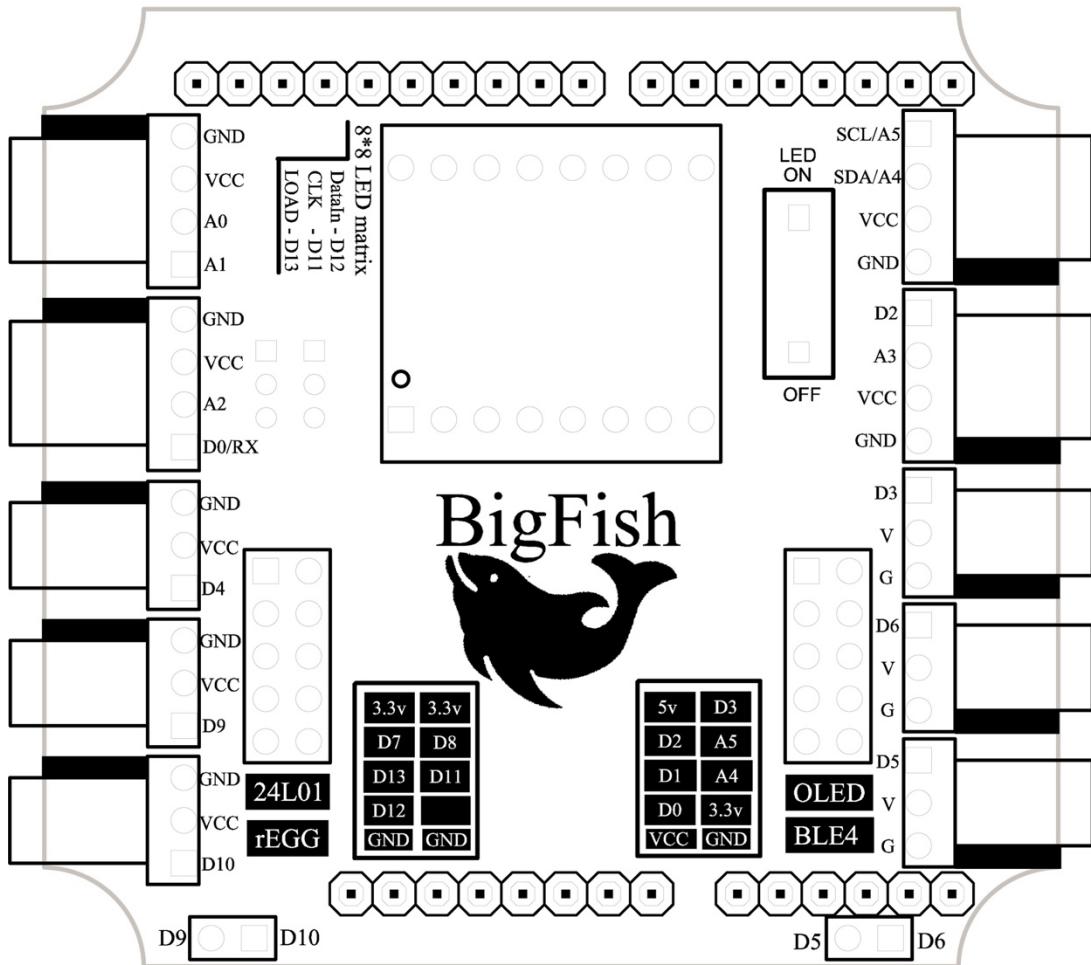
本次项目使用的是 Basra 主控板开发板。Basra 主控板是基于 ATmega328 的单片机开发板。ATmega328 包括了片上 32KB Flash，其中 0.5KB 用于 Bootloader。同时还有 2KB SRAM 和 1KB EEPROM。同时 Basra 主控板开发板是当前主流的开发板，其拥有所有 Arduino 系列单片机的功能，Basra 主控板供给 ATmega328 芯片的电压是 5.0 V，理论上可以工作在 20 MHz 以内任何时钟频率上。板载 USB

驱动芯片及自动复位电路，烧录程序时无需手动复位。Basra 主控板原理图见图



3.2 Bigfish 扩展板

如果仅靠主控板的引脚已能完成大部分的控制需求，但在电机驱动上需要额外的电机驱动模块，而探索者套件中的 Bigfish 扩展板集成了电机驱动的模块，不仅节省了安装额外驱动模块的空间，还提供了更高的稳定性，确保了小车的电机输出持续稳定。并且可以使用 4pin 传感器线进行颜色传感器与灰度传感器的连接，方便可靠。



3.3 循迹转弯

灰度既可以当作数字量传感器使用，也可以当作模拟量传感器使用。灰度检测的有效距离在 0.7cm~3cm 之间，我们使用 ADC 转换来实现循迹功能，我们读取灰度传感器的模拟值，通过多次实验我们确定了灰度值<240 时将此时状态标记为识别到黑线，其余状态为黄色地面。

我们采用的三灰度循线逻辑如下：

A2	A3	A0		
0	0	0	0x00	表示三个传感器都没有触发
0	0	1	0x01	表示小车右边一个传感器触发
0	1	0	0x02	表示小车中间传感器触发
0	1	1	0x03	表示右边两个传感器都触发
1	0	0	0x04	表示小车左边一个传感器触发
1	0	1	0x05	
1	1	0	0x06	表示左边两个传感器都触发
1	1	1	0x07	表示小车三个传感器都触发

我们通过三个传感器的灰度值来控制电机差速循迹，代码逻辑如下：

```

void Automatic_Tracking() //小车前方两个灰度传感器用来寻迹（即小车巡
线）。
{
    int Get_Data = 0;
    Get_Data = Detection_tracking();
    switch(Get_Data)
    {
        case 0x00:
            Car_Move(Forward,Forward_Left_Speed,Forward_Right_Speed);break;//forward
        case 0x01:
            Car_Move(Right,Right_Left_Speed,Right_Right_Speed );break;//RIGHT
        case 0x02:
            Car_Move(Forward,Forward_Left_Speed,Forward_Right_Speed);break;//forward
        case 0x03:
            Car_Move(Right,Right_Left_Speed,Right_Right_Speed );break;//right
        case 0x04:
            Car_Move(Left,Left_Left_Speed,Left_Right_Speed );break;//left
        case 0x06:
            Car_Move(Left,Left_Left_Speed,Left_Right_Speed );break;//LEFT
    }
}

```

```

        case 0x07: //Gray_Three++;
    }

    Car_Move(Forward,Forward_Left_Speed,Forward_Right_Speed);break;//forward
    default: break;
}
}

```

同时我们实现了在每个障碍物前进行识别十字路口的功能，但 0x07 状态时，三个传感器都触发，实现对十字路口的识别，识别到后我们会开始越障与颜色识别的部分。

第一个十字路口时进行加速，在爬至坡顶时我们进行停车操作，然后缓慢下坡，防止弹珠掉落，代码逻辑如下：

```

void road_one()
{
    Car_Move(Forward,170,200);delay(2200);
    Car_Move(Forward,105,135);delay(1000);
    Car_Move(Stop,Car_speed_stop,Car_speed_stop);delay(500);
    Car_Move(Forward,30,60);delay(700);
    Car_Move(Stop,Car_speed_stop,Car_speed_stop);delay(500);
    Car_move_state_delaytime(Tracking_automatic,5500);
}

```

第二个十字路口时，我们先加速上台阶，然后在下台阶时采取分段加速停车的操作，代码如下：

```

void road_two()
{
    delay(1000);
    Car_Move(Forward,130,150);delay(2320);
    Car_Move(Stop,Car_speed_stop,Car_speed_stop);delay(1000);
    Car_Move(Forward,50,80);delay(600);
}

```

第三个十字路口时，我们采取停车动作，进行色卡的识别，代码逻辑如下

```
void road_three()
{
    Tracking_Automatic_Tracking(370);
    Car_Move(Stop,Car_speed_stop,Car_speed_stop);delay(500);
}
```

3.4 颜色识别

本模块是基于 TCS34725 彩色光数字转换器为芯片设计的颜色识别传感器，传感器提供红色、绿色、蓝色(RGB)和清晰光感应值的数字输出。集成红外阻挡

滤光片，可最大限度地减少入射光的红外光谱成分，并可精确地进行颜色测量。具有高灵敏度、宽动态范围，且配置红外阻隔滤波器。最小化 IR 和 UV 光谱分量效应，以产生准确的颜色测量。带有环境光强检测和可屏蔽中断。通过 I2C 接口通信。

本模块增加 M12 镜头，有效增加检测距离。并设置有 4 个高亮 LED 灯进行补光，可通过红外检测头自主控制 LED 灯的亮灭。通过内置的定位器旋钮调节检测距离（最大为常量，最小为常闭）来控制 LED 灯的感应距离。

我们先将色卡的颜色进行存储，分别与后三个色卡进行比对来确定正确颜色，识别目标色卡与标记色卡的代码逻辑如下：

```
void return_color_ballon()
{
    int numbers_count = 0;
    int color_judge[12]={0,0,0,0,0,0,0,0,0,0,0,0};
    int red_summer,green_summer,blue_summer;
    Serial.println("-----Start-----");
    unsigned long time_now = millis();
    while( (millis() - time_now )< 2000)
    {
        numbers_count++;
        uint16_t clear, red, green, blue;
        tcs.getRGBO(&red, &green, &blue, &clear);
        if(red>=blue){color_judge[0]      =      color_judge[0]      +1;}
        else{color_judge[1] = color_judge[1] +1;}
        if(red>=green){color_judge[2]      =      color_judge[2]      +1;}
        else{color_judge[3] = color_judge[3] +1;}

        if(blue>=red){color_judge[4]      =      color_judge[4]      +1;}
        else{color_judge[5] = color_judge[5] +1;}
        if(blue>=green){color_judge[6]      =      color_judge[6]      +1;}
        else{color_judge[7] = color_judge[7] +1;}
```

```

        if(green>=red){color_judge[8]      =      color_judge[8]      +1;}
else{color_judge[9] = color_judge[9] +1;}

        if(green>=blue){color_judge[10]     =      color_judge[10]     +1;}
else{color_judge[11] = color_judge[11] +1;}

}

```

```

void return_color_card()
{
    int numbers_count = 0;
    int color_judge[12]={0,0,0,0,0,0,0,0,0,0,0,0};
    int red_summer,green_summer,blue_summer;
    Serial.println("-----Start-----");
    unsigned long time_now = millis();
    while( (millis() - time_now ) < 2000)
    {
        numbers_count++;
        uint16_t clear, red, green, blue;
        tcs.getRGBO(&red, &green, &blue, &clear);
        if(red>=blue){color_judge[0]      =      color_judge[0]      +1;}
else{color_judge[1] = color_judge[1] +1;}

        if(red>=green){color_judge[2]     =      color_judge[2]     +1;}
else{color_judge[3] = color_judge[3] +1;}


        if(blue>=red){color_judge[4]      =      color_judge[4]      +1;}
else{color_judge[5] = color_judge[5] +1;}

        if(blue>=green){color_judge[6]     =      color_judge[6]     +1;}
else{color_judge[7] = color_judge[7] +1;}


        if(green>=red){color_judge[8]      =      color_judge[8]      +1;}
else{color_judge[9] = color_judge[9] +1;}

        if(green>=blue){color_judge[10]     =      color_judge[10]     +1;}
else{color_judge[11] = color_judge[11] +1;}}

```

```
else{color_judge[11] = color_judge[11] +1;}  
}
```

3.5 物品倾倒

本设计通过控制一组 180 度舵机进行倾倒过程，首先一个 180 度伺服电机将摇臂上所承载的载盘旋转移动至料盒上方，然后通过控制另一个 180 舵机进行旋转，将载盘进行旋转完成倾倒物品任务。

我们使用了 ServoTimer2 定时器舵机库来避免与颜色传感器的冲突，通过 map 映射脉冲来控制舵机的运动。

3.6 直流电机模块

我们所使用的 TT 电机为探索者套件中自带的减速 TT 电机模块，由于同侧的驱动轮的运动状态始终是一样的。因此同侧的 3 个直流电机通过自制的一拖三电机线上，共用 1 个直流电机接口，两个直流电机接口的针脚号分别为 (D5,D6) 以及 (D9,D10) 。

3.7 任务结束策略

在弹珠投放任务结束后，小车将继续进行巡线任务，当三个灰度传感器传回十字路口的数据后，通过延时来控制最后的停车。至此程序执行完毕。

第4章 创新设计说明

4.1 颜色识别优化

传统的颜色识别单次识别时间较长、效率较低，对任务完成的整体效率没有实现有效的帮助，为此我们改进的颜色识别方案——多次颜色识别进行累加。提高颜色脉冲的采样次数。降低单次采集数据的时间在保证精准的识别颜色的同时

又将识别速度提高了一个等级，让完成任务的时间大大缩短。

4.2 六轮悬挂式

在试跑过程中，前后轮一体式的结构虽然更加稳定与轻便，但在上台阶时其简单的结构极具局限性，在载重较大的情况下仅靠前轮驱动力无法登上台阶，很难顺利通过障碍。于是将前后轮一体式结构优化改进为六轮悬挂式结构，灵活的悬挂式结构在上台阶时的优势体现的尤为明显，前轮接触到台阶的瞬间紧接能靠活动的悬挂结构爬上台阶，其灵活性足以保证小车的越障能力并且将动力持续均匀输出，并且将前两轮之间加入连接结构使小车在保持六轮均衡抓地的同时，两边轮能够在同一时间登上障碍以保持两侧车轮的同步性，进一步为车体的平稳性提供了有力保障，小车得以平稳而顺利地越过障碍。

第5章 设计过程

5.1 结构迭代

第一代结构中我们的小车设计的是前置四轮一体式悬挂。但在我门经过研究实验后发现，履带大轮的摩擦力远远不及硅胶小轮，且因主控板电机输出口只有两个，无法精准的控制每一个电机的转速，只能分别控制两侧各三个轮子的转速，因此在转速一致时，大轮的线速度会因较大的车轮直径远大于后面四小轮的线速度使得整个小车的拖速现象非常严重，于是我们进行了第一次结构迭代：将其更新为六硅胶小轮。虽然六硅胶小轮的速度的控制得以平衡稳定，然因其直径太小且台阶的高度较高，小车无法顺利上阶，往往在越障时卡住。因此我们又对小车的结构进行了更新迭代：将前轮缠绕上履带片。或许因车体载重不均或各轮运行不够同步，我们在运行小车时又发现两侧独立悬挂的车轮结构在进行爬坡和越障时都极易出现因前两轮不稳而翻车倾覆的现象，我们在利用悬挂结构时刚开始使用限位结构件限制了悬挂的抬起角度，限位结构件本身强度并不高，而悬挂的抬起角度一旦受到限制会导致悬挂起到的作用被进一步降低，因此六轮悬挂方案被我们暂时搁置。经过对前几轮迭代方案的总结与改善，我们在讨论与实验过后得出了一体式四轮结构的方案，此方案在初阶段越障巡线时的表现都比前几代更

加顺畅，且车体结构也比较精简轻便，而当我们在车体上安装上云台载盘等结构后，车体重量明显增大，与此同时我们也开始着重考虑减小玻璃弹珠于载盘中掉落出来的可能性，然而一体式四轮在越障过程中几乎只是单纯依靠高速度冲上台阶，小车整体结构起到的作用微乎其微，除此以外，小车时常因车体过重在下坡时出现不稳翻车情况，在上台阶时因缺乏灵活性而无法顺利通过，因此我们不得不放弃一体式四轮结构方案，综合考虑小车需体现的多个功能模块，着重研究车体的稳定性以及灵活性。

最后，我们又将小车结构改为六轮悬挂式结构，但此次结构与初代悬挂结构并不相同，我们结合迭代过程中安装车体的经验，车体结构比起以往更加牢固精简，并且综合考虑到弹珠载盘的稳定性，我们取消了安装陀螺仪控制云台的方案，转而通过调整小车前后车轮间距使其与单节台阶的长度相适应，因此小车上安放载盘的平台的倾斜角度可以始终保持在较小范围内，在此基础上我们同时通过控制小车移速减慢以尽量保证弹珠始终保持在载盘内。

在灰度传感器的安装上，我们也经历了多次思考与实验。灰度传感器的安装位置如果过于靠后，在经过管道时，小车的姿态会因无法及时调整而导致车头撞壁。而若其安装位置过于靠前，在越障过程中则会导致灰度传感器卡在台阶处，小车轮胎无法接触台阶，小车无法顺利完成越障过程。因此灰度传感器的安装位置必须于小车前两轮的中心轴线偏前处。另外，灰度传感器的高度确定也同等重要。起初我们忽略了灰度传感器的高度问题，在进行灰度循迹工作时小车总是跑偏或者在转弯时无法发挥作用，后来确定灰度传感器的高度不能过高或过低，最适宜距离为距地面 0.7cm~2cm 之间，否则会引起识别误差影响小车完整执行巡线任务。

5.2 程序迭代

在最开始的程序设计中，灰度传感器采用的是简单的数字量读取判断，后来发现由于环境光变化的原因数字量判断非常不准确且不可控，如在中午和下午两个不同的时间段分别运行时传回的数字量完全不同，甚至在之后使用模拟量的过程中，白天和黑夜的灰度值也相差非常之大。所以我们采用了 ADC 转换的方式读取传感器的值。

在颜色识别中，我们通过多次实验确定了最终 RGB 色值的识别参数，达到准确的识别。