# 目　录

# 第 1 章　参赛队伍和方案

## 1.1　epsilonGoal 队简介

### 1.1.1　队名介绍

”ε”，epsilon 喻指微小但具有无限可能的事物。epslionGoal，寓意我们在未知的、浩瀚的、充满着无限可能的机器人世界里探索。我们有着明确的目标，我们要将那个远大的目标分成很多微小的目标，一个一个地实现。我们相信只要仰望星空，脚踏实地，我们最终的目标就会实现！

### 1.1.2　成员分工

**表 1.1　组员组成及分工**

| 姓名 | 学号 | 分工 |
| --- | --- | --- |
| | | 队长，负责整体规划，团队建设和视觉模块 |
| | | 负责机械设计与建模 |
| | | 负责电控模块设计，机器人理论计算 |
| | | 负责电控模块设计，比赛任务系统设计 |
| | | 负责机械设计与建模 |

## 1.2　机器人功能与结构概述

本次比赛内容为机器人投篮，要求机器人能实现自动巡线、识别并抓取篮球、越障、投篮等多种任务，并在尽可能短的时间内得到更多的分数，具体而言包括以下内容：

- 自动巡线，实现机器人的移动功能
- 越障，并保证机器人结构和功能实现的稳定性
- 准确识别篮球
- 自动抓取并储存识别的篮球
- 完成投篮

本组经过对比赛任务的解读和商定，基本确定了机器人机械结构设计的基本规范和大体布局，初步建模并修改、调整后，得到基本模型如下：

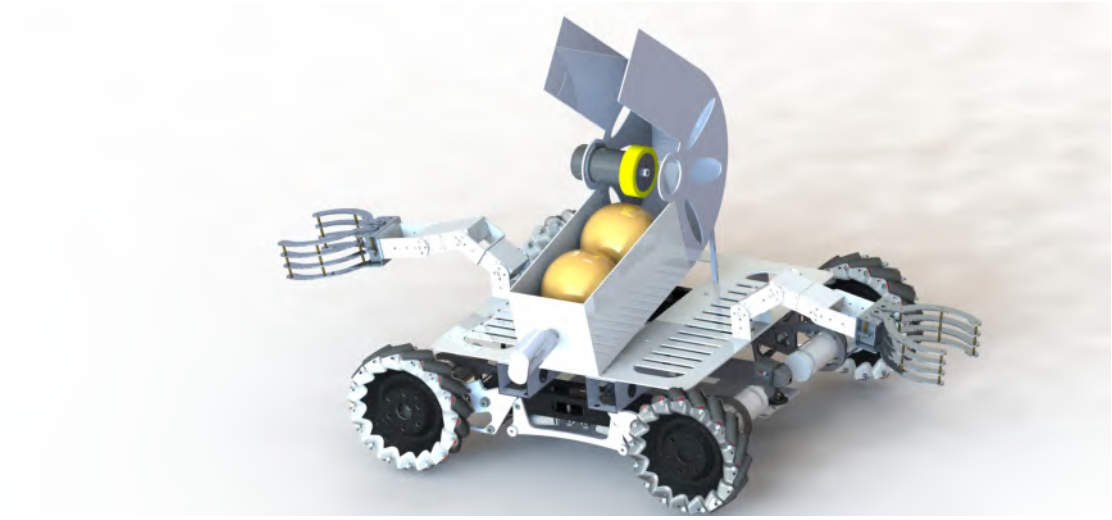


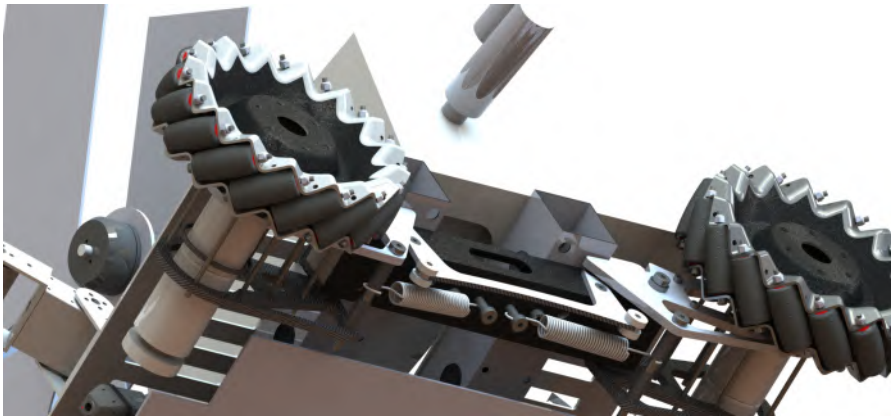
**图 1.1　基本模型**

- 越障设计: 考虑到需要进行越障功能，我们决定同向放置麦克纳姆轮，保持机器人越障的稳定性实现，同时在采用大梁固定加独立悬挂结构，极大的减少了越障过程对机械结构和传感器的影响。
- 对称设计: 机器人的底盘部分关于平行于长中轴对称，投篮部分关于平行于宽的中轴对称，同时在选材、设计、安装时注意保持整体结构重心居中、四轮着地。
- 双向机械臂抓取: 左右两个机械臂设计，平衡了机器人重心的同时，便于不通过转向从两侧实现抓取，节省了时间，减小了失误的可能性。
- 推进式储物箱和摩擦轮发射结构: 设计能合理容纳两个篮球的储物箱储存收集的篮球，在稳定并投篮时运行摩擦轮，并通过推杆将小球与摩擦轮接触，实现快速发射投篮。
- 巡迹设计: 结构简明，实现方便，成本低廉，反应灵敏，便于近距离路面情况的检测，抗干扰能力强，不会因为周围环境的差别而产生不同的结果。

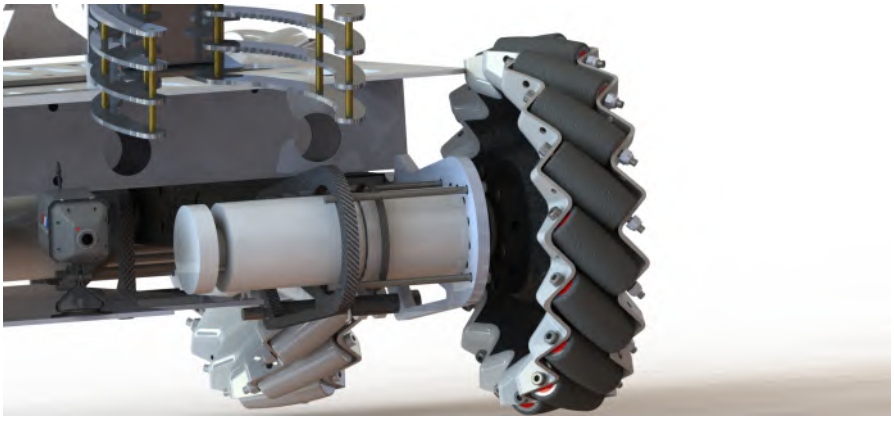
# 第 2 章　主要机械部分设计与功能

## 2.1　底盘结构与麦克纳姆轮

本次设计中，充分考虑到越障和巡线功能的需要，利用中央大梁链接左右两侧麦克纳姆轮的固定结构，并通过如图所示实现独立悬挂结构，保证机器人在运行过程中的可靠性与越障的稳定性。

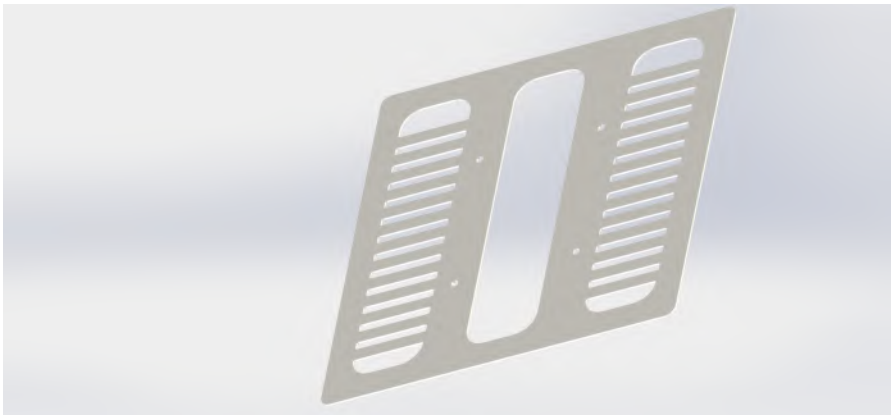


麦克纳姆轮采用 12.7cm 直径尺寸，结合所选电机，通过里外对称的两个亚克力板固定，并连接到轮轴的主梁，并保持上下单维度的自由运动空间，上有梁和亚克力板的阻挡关系限制，下有弹簧结构连接，保持悬挂结构的灵活和稳定。



两层梁稳定结构提供了较大存放空间，其下悬挂一层下底板，用于安排、固定各类传感器，并利于梳理、隐藏设计线路，保证整洁美观与高效。

上层梁上直接固定上底板，进一步加强了整体结构的稳定性同时，为安放机械臂、储存与摩擦轮发射台结构提供了平台，通过合理镂空钻孔与设计，减轻机器人重量的同时提供线路整理的渠道。



## 2.2　机械臂结构

机械臂结构整体对称分布在小车左右两侧（见小车总装渲染图），用来抓取存放在两侧的小球，根据计算和演示，机械臂抓取时所处位置高度与小球存放高度相符。

每个机械臂由三个舵机和一个机械爪结构，机械爪用来抓取小球，舵机控制机械臂二维运动，将小球放在储物箱中，根据调整机械臂位置，使放入小球位置在储物箱靠近摩擦轮位置，第一个小球沿着斜面自然下落，为第二个小球腾出位置。

单个机械臂示意图如下：

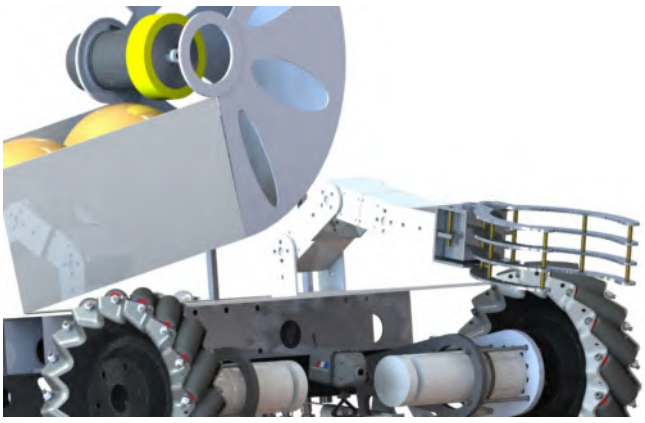


**图 2.1　机械臂示意图**

机械爪结构，通过舵机连接驱动机械抓开合，向斜下方倾斜抓取小球，因为小球表面非刚体，机械抓抓取后，通过舵机控制，使得小球表面产生微小形变，防止脱落。

若经费充足，机械臂套装可以直接购买，若经费紧张，也可购买散装零件，

自己动手制作。

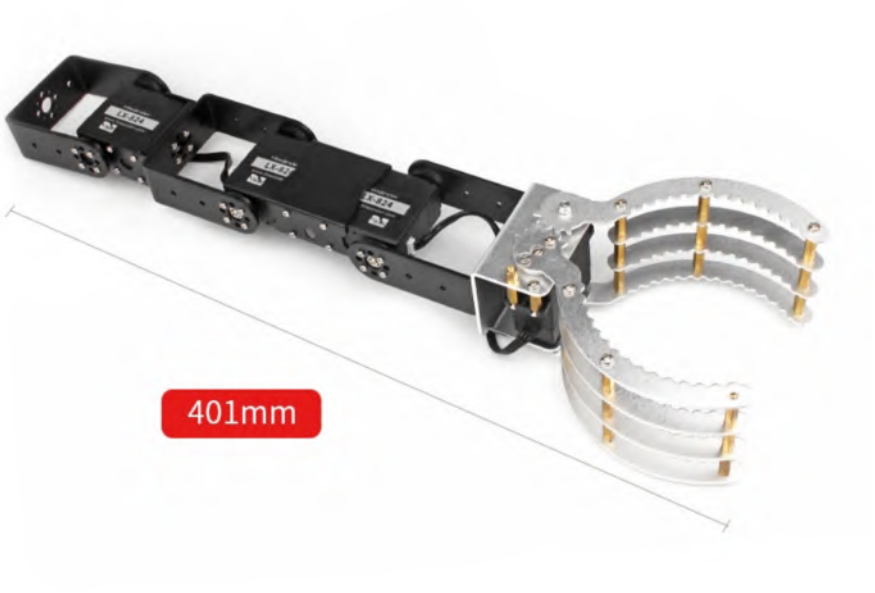



**图 2.2　机械臂实物图**

## 2.3　储物箱结构

储物箱由两部分构成，分别是 220mm 长，100mm 宽的箱体和推杆结构组成，箱体整体与车座平面呈现约 15 度左右夹角（具体角度由成型后根据发射情况调试至最佳角度），斜放结构使得小球自然下落至箱底，实现运输过程，需要发射时，由推杆将小球推至与摩擦轮接触发射。

**图 2.3　储物箱示意图**

**图 2.4　储物箱体图**

## 2.4　发射台结构

如图是小车发射台装置，整个发射台由储物箱结构和导轨结构装配而成，储物箱结构主要实现了小球的存储功能，具体原理和实现上文已有解释，此处不多赘述，这里着重介绍导轨结构。

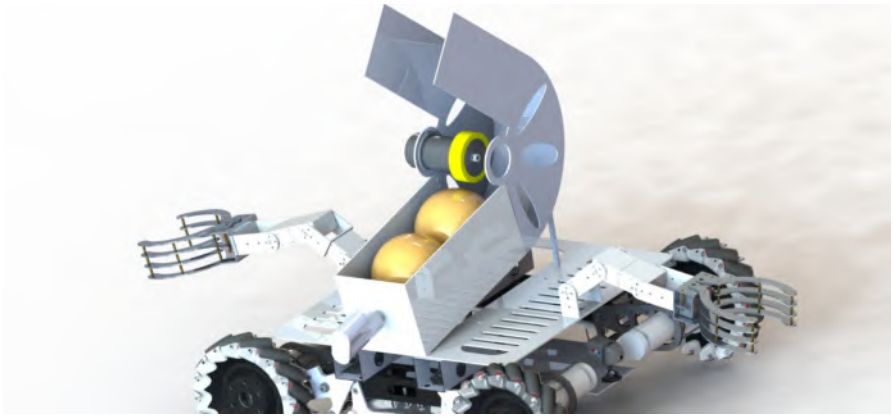


**图 2.5　发射台示意图**

导轨结构实现小球的发射功能，储物箱推杆将小球上推，与摩擦轮接触之后，摩擦轮通过摩擦给小球加速，小球沿着导轨发射出去。

导轨结构如下：

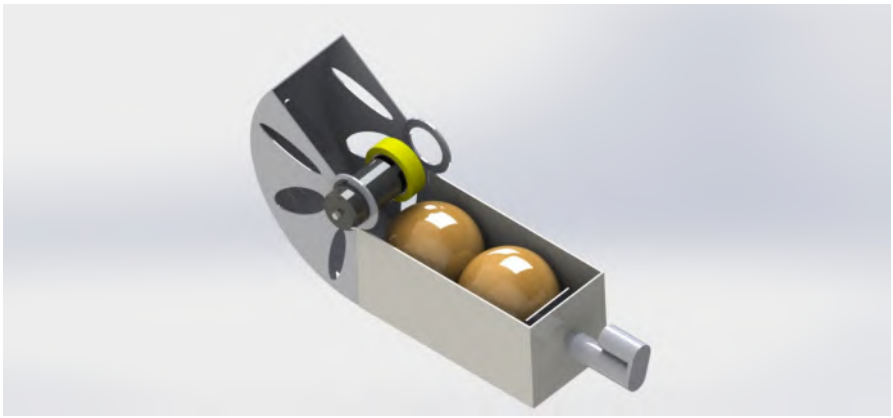


**图 2.6　导轨及储物箱示意图**

我们注意到，对比图2.6，图2.5 发射台示意图在导轨侧面又伸出来两条挡板。该挡板的作用是，在小球发射时保证小球沿着导轨方向出射，避免了因摩擦轮或小球形变造成的水平方向的位移。

## 2.5　机械结构可行性检验

**图 2.7　投篮示意图**

发射台两端的挡板保证小球出射时水平方向不发生位移，出射角度可以通过导轨角度和曲率调整。

**图 2.8　越障示意图**

麦克纳姆轮朝向与发射面垂直, 发射台及导轨所在平面始终与篮筐垂直，全程没有车身的旋转，过减速带时，麦克纳姆轮垂直与减速带。

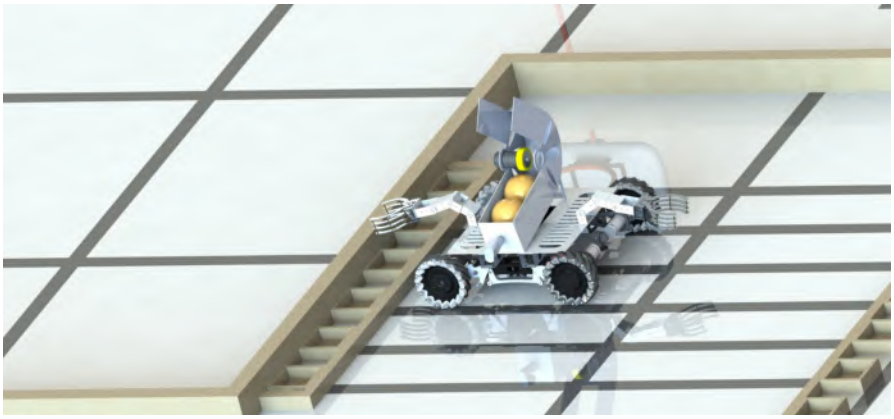


**图 2.9　取球示意图**

抓取装置为对称的两个，这样机器人就可以在车身方向不变的情况下，可在两侧进行抓取。

抓取装置的位置：抓取装置安置在第二个球的位置，就是靠近摩擦轮的位置，在抓取一颗球时，放入储存盒中，由于盒子本身倾斜，球会自然向下滑动，到达储物箱底，第一个球的位置自然空出来，机械臂可以放置下一个球。

摄像头位置的确定：摄像头位于机械臂下方，位置比较偏低，因为球盒位置也很低。两侧均有摄像头，两侧同时识别。

## 2.6　越障与循迹的协同

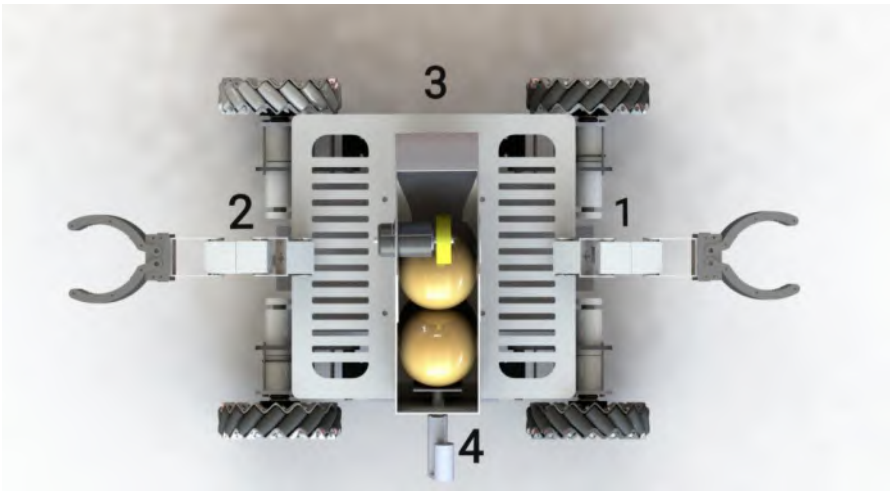
以上就是机械结构的设计，下面这部分内容是为了解决越障与红外对管巡线之间的矛盾。

矛盾：我队选择通过越障的方式进入投篮区投篮，据举办方数据显示，障碍物高度约为 1.1cm，而红外对管识别的有效距离约为 1.5cm，所以根据对实际情况的猜测，机器人在越障的过程中，红外对管可能会与障碍物相撞。为了保证红外对管识别精确，又要实现机器人越障时红外对管与障碍物不相撞，我队商讨了很多，最终选择如下方案：（由于此方案是对实际情况的预测，故而作为备用方案写入计划书，到时根据实际小车运动情况来决定是否采用此方案）


我们将购买两个上图所示的巡线模块——七路灰度巡线模块。其余仍是普通的红外对管集成板。

选择它是因为其探测距离（探头底部离地距离）为 10-50mm，在 10-25mm 区间效果良好。

按照我队的越障方案，麦克纳姆轮垂直与障碍物过障。故而采用如下策略：




第一点：将红外对管模块安装至前后两侧，即 1，2 位置，如图。将红外对管与麦克纳姆轮绑定，这样，当麦克纳姆轮被障碍物抬起时，红外对管也被抬起，这样，保证了红外 1，2 位置的红外对管与障碍物不相撞。

第二点：我们将上面的七路灰度巡线模块安装至车身两侧，即 3，4 位置，如图。在该位置上，由于该巡线模块推荐距离为 10 25mm，与障碍物有一些距离，车在越障过程中，尽管车身高度可能在某些时刻会下降，该巡线模块也不会与障碍物相撞。

# 第 3 章　主要电路部分设计与分析

## 3.1　主要电路框图

以下是我组机器人主要电路框图。

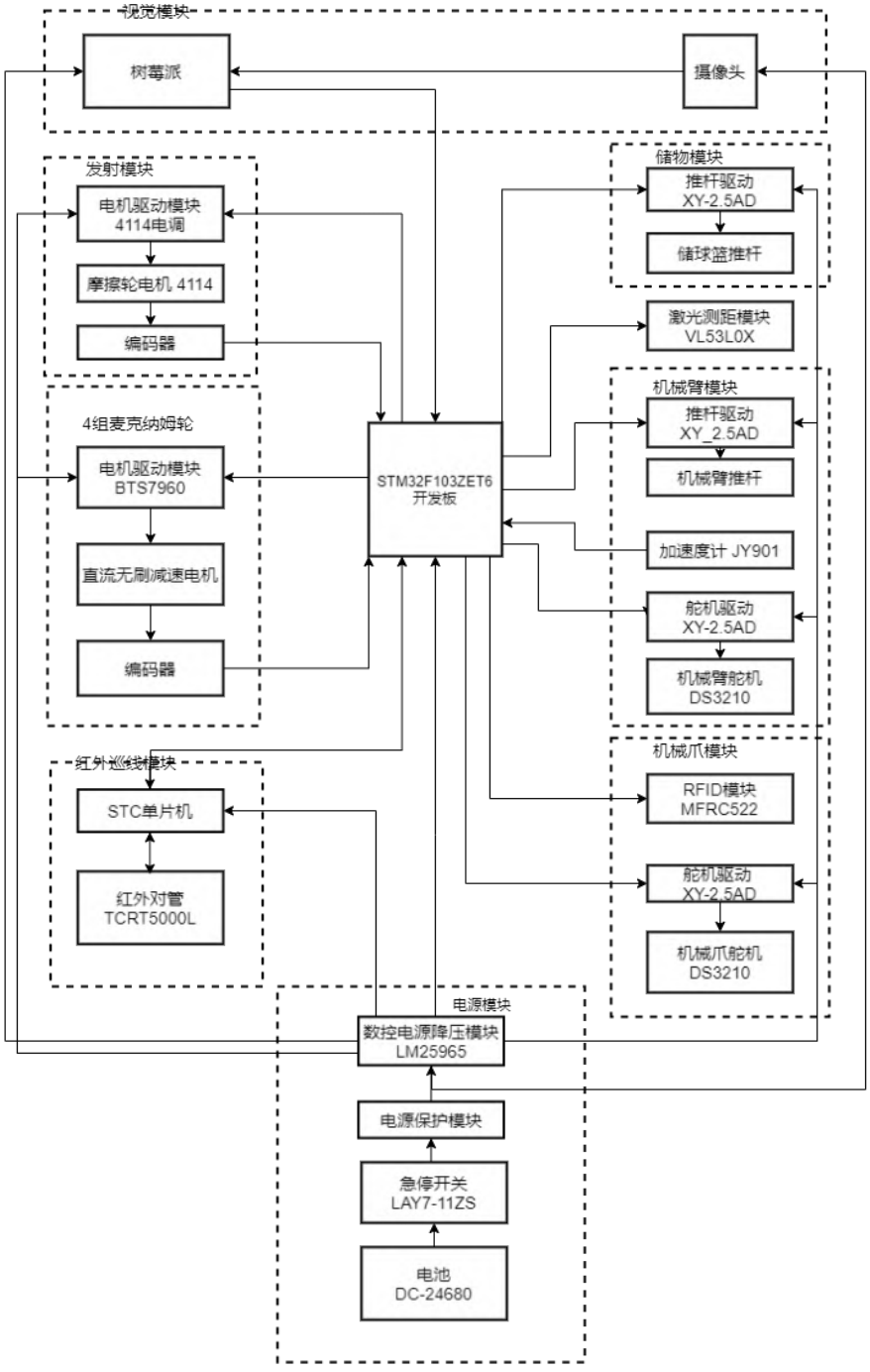



**图 3.1　主要电路框图**

## 3.2　下位机主控

STM32 是由意法半导体基于 ARM Cortex-M 研制和生产的一系列 32 位单片机。STM32F103 器件采用 Cortex-M3 内核，CPU 最高速度达 72 MHz。该产品系列具有 16KB ~ 1MB Flash、多种控制外设、USB 全速接口和 CAN。考虑到本届赛事的实际需要，经过综合考虑，我组选择型号为 STM32F103ZET6 的微控制器，作为机器人的主控下位机。

我组采用 Telesky 店铺发行的开发板，具体情况如下。

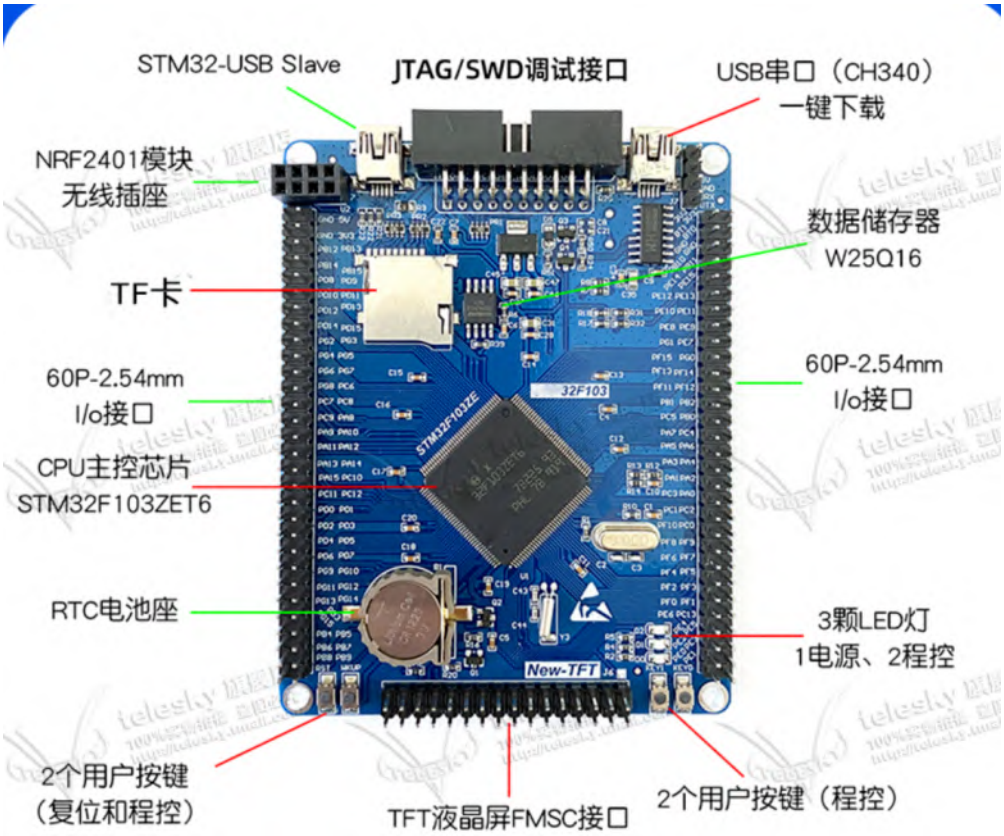



**图 3.2　开发板基本情况**

该型号具备 STM32F103ZET6 型号芯片，其具有如下特点：

| 功能 | 特点 |
| --- | --- |
| 内核 | ARM 32 位的 Cortex™-M3 CPU<br>最高 72MHz 工作频率<br>单周期乘法和硬件除法 |
| 存储器 | 从 256K 至 512K 字节的闪存程序存储器<br>高达 64K 字节的 SRAM<br>带 4 个片选的静态存储器控制器 |
| 时钟、复位和电源管理 | 2.0～3.6 伏供电和 I/O 引脚<br>上电/断电复位 (POR/PDR)、可编程电压监测器 (PVD)<br>4～16MHz 晶体振荡器<br>内嵌经出厂调校的 8MHz 的 RC 振荡器<br>内嵌带校准的 40kHz 的 RC 振荡器<br>带校准功能的 32kHz RTC 振荡器 |
| 低功耗 | 睡眠、停机和待机模式<br>VBAT 为 RTC 和后备寄存器供电 |
| ADC | 3 个 12 位模数转换器，1μs 转换时间 (多达 21 个输入通道)<br>转换范围：0 至 3.6V<br>三倍采样和保持功能 |
| DAC | 2 通道 12 位 D/A 转换器 |
| DMA | 支持的外设：定时器、ADC、DAC、SDIO、<br>I2S、SPI、I2C 和 USART |
| 调试模式 | 串行单线调试 (SWD) 和 JTAG 接口<br>Cortex-M3 内嵌跟踪模块 (ETM) |
| 快速 I/O 端口 | 所有 I/O 口可以映像到 16 个外部中断<br>几乎所有端口均可容忍 5V 信号 |
| 11 个定时器 | 4 个 16 位定时器<br>每个定时器有 4 个用于输入捕获/输出比较/PWM<br>或脉冲计数的通道和增量编码器输入<br>2 个 16 位带死区控制和紧急刹车，<br>用于电机控制的 PWM 高级控制定时器<br>2 个看门狗定时器 (独立的和窗口型的)<br>系统时间定时器：24 位自减型计数器<br>2 个 16 位基本定时器用于驱动 DAC |
| 13 个通信接口 | 2 个 I2C 接口 (支持 SMBus/PMBus)<br>5 个 USART 接口<br>3 个 SPI 接口 (18M 位/秒)，2 个可复用为 I2S 接口 |

该芯片原理图如下。

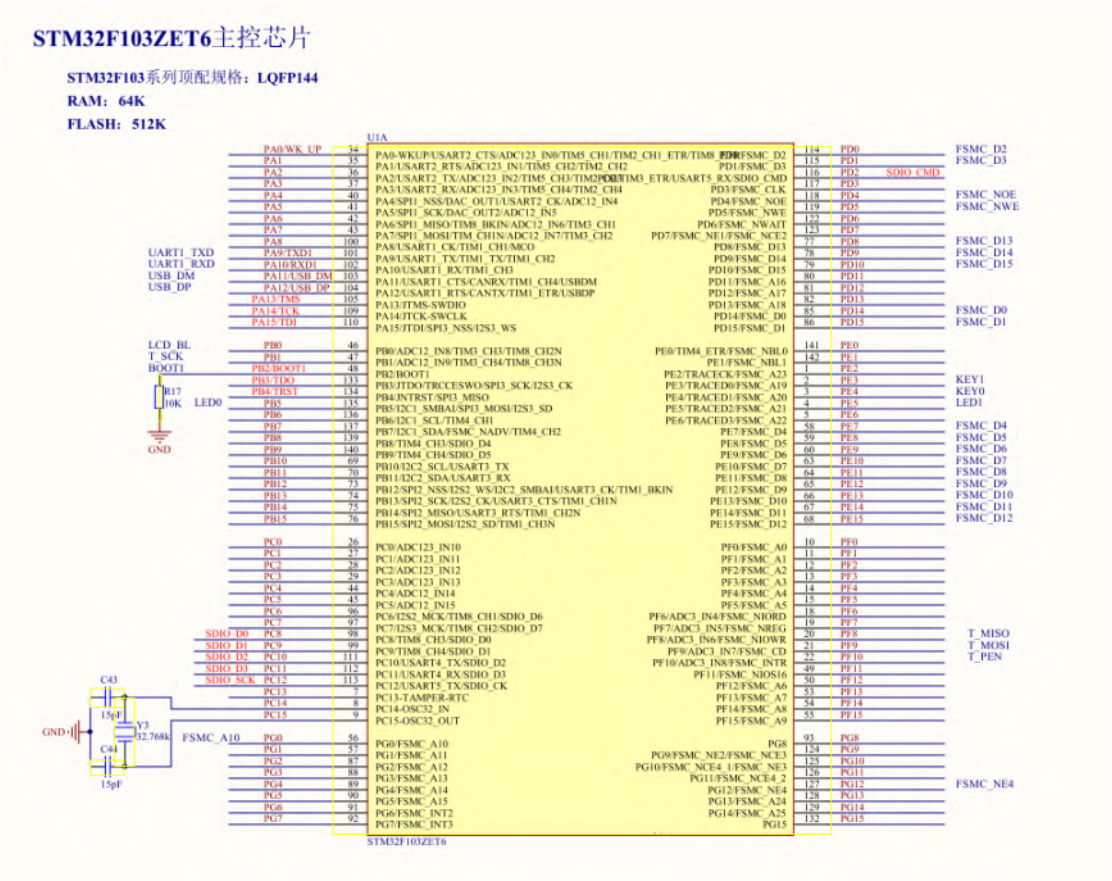



**图 3.3　芯片原理图**

## 3.3　系统供电

本机器人主要供电线路如下。

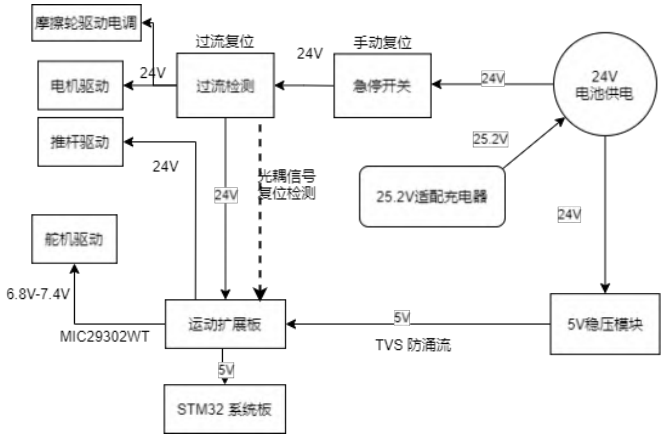



**图 3.4　供电系统**

其中具有的特点包括：

- 单电源供电，LDO 稳压降压满足需要。
- 板载低压输出和电源滤波。
- TVS 防涌流设计。
- 光耦检测，驱动电路掉电时控制电路自动复位。
- 配备急停开关，及时停止机器人进入不符合预期的操作模式

### 3.3.1　电源

考虑到本机器人将具备大量电机同时运转，多组传感器模块共同运作，不同控制器交替调控，为确保各模块的正常运转，我组采用迪普威科技出品的 DC-24680 的动力锂电池组，其具备 24V 输出电压，6800mAh 的容量，可以满足我组需要。


**图 3.5　电池组实物图**

### 3.3.2　电源稳压模块

众所周知，控制模块和各传感器对电压要求较高，不能直接使用电池供电。我组采用 Telesky 店铺定制的 DC-DC 5A 可调数控降压数显电源模块，其具有如下特点。

- 4-38V 输入电压，1.25-36V 连续可调输出
- 输出电流最高 5A，可以 4.5A 连续使用
- 转换效率高，具有过热保护
- 实时显示输入输出电压，可自校准

**图 3.6　电源稳压模块**

后期，我组将使用自制控制板板载电源稳压模块完成这一功能，如下

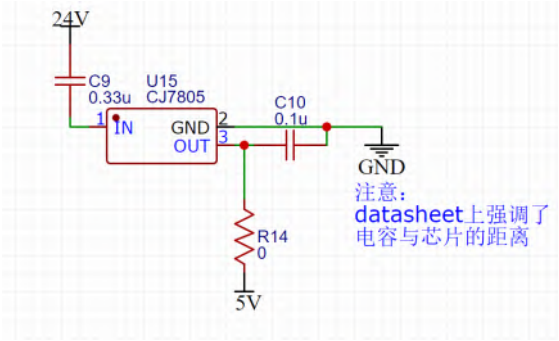



**图 3.7　24V 转 5V**

其中使用了芯片 CJ7805，参数和实物图如下：

| 参数 | 内容 |
|---|---|
| 压降 | 2V @ 1A(typ) |
| 最大输出电流 | 1.5A |
| 最大输入电压 | 35V |
| 输出类型 | 固定 |
| 输出电压 | 5V |

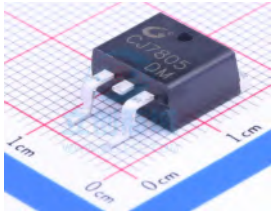


**图 3.8　CJ7805**

考虑到实际情况下芯片工作电流较大，可能产生较大热量散失，造成元件损坏，我组也设计了使用开关电路的芯片 LM2596R 作为后备转换模块，电路图如下所示。

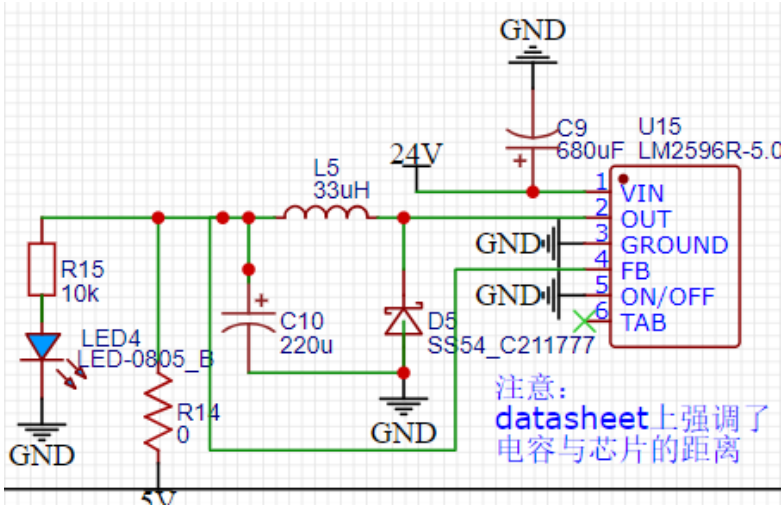



**图 3.9　LM2596R 配套电路**

可以通过去除和焊接 0 欧电阻来控制是否启用该电路。

基于同样的原理，我们也设计了适用于舵机的 24V 转 6V 稳压电路，使用的芯片为 XL4003，原理图如下。

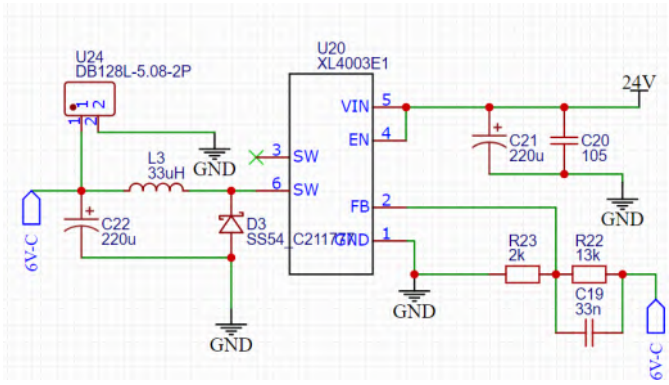



**图 3.10　XL4003 配套电路**

### 3.3.3　电路过流保护模块

为保证单片机控制板和电机不会发生协同过流，控制电路和驱动电路的供电应当是从电源引出的不同分支。然而，仅仅这样做仍不能保证全电路的安全稳定，极端情况下，我们需要适当的电路保护模块，保证触发熔断效应，减少不必要的设备报废。

我组使用型号为 YYI-4 的电流检测模块。由于电路中经过大量电流，为确保有效性与可靠性，我组不使用自制 PCB 承担如此任务。

该模块具有如下特点。

- 检测电压 7-30V，电流 0-30A
- 检测精度 100mA，待机电流 20mA
- 实时电流显示

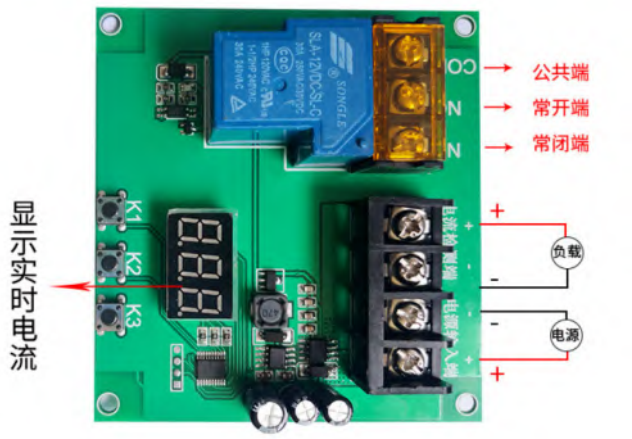



**图 3.11　电路过流保护模块**

## 3.4　系统驱动和电机

我组使用电源供电为 24V，为各电机与舵机提供充足的动力。

### 3.4.1　主要动力电机

我组使用 CHP-36GP-555 ABHL 永磁行星编码减速电机，该电机动力充足，编码精准，可以良好地完成预期任务。

具体特点如下。

- 6-24V 宽电压输入
- AB 双相编码器 17 线，信号电压 3.3V 或 5.0V
- 20.0mm 单向出轴，8mmD 字型轴双滚珠轴承定位结构
- XH2.54-6Pin 端子连接头

其中具备的霍尔编码器的参数如下。

| 类型 | AB 双相增量式磁性霍尔编码器 |
|---|---|
| 线速 | 基础脉冲 17 PPRx 齿轮减速比 |
| 供电电压 | DC 3.3V 或 5.0V |
| 基本功能 | 自带上拉整形电阻，单片机直连 |
| 接口类型 | XH2.54-6PIN |
| 输出信号类型 | 方波 AB 相 |
| 响应频率 | 100kHz |
| 基础脉冲数 | 17PPR |
| 磁环触发极数 | 34 级（17 对极） |

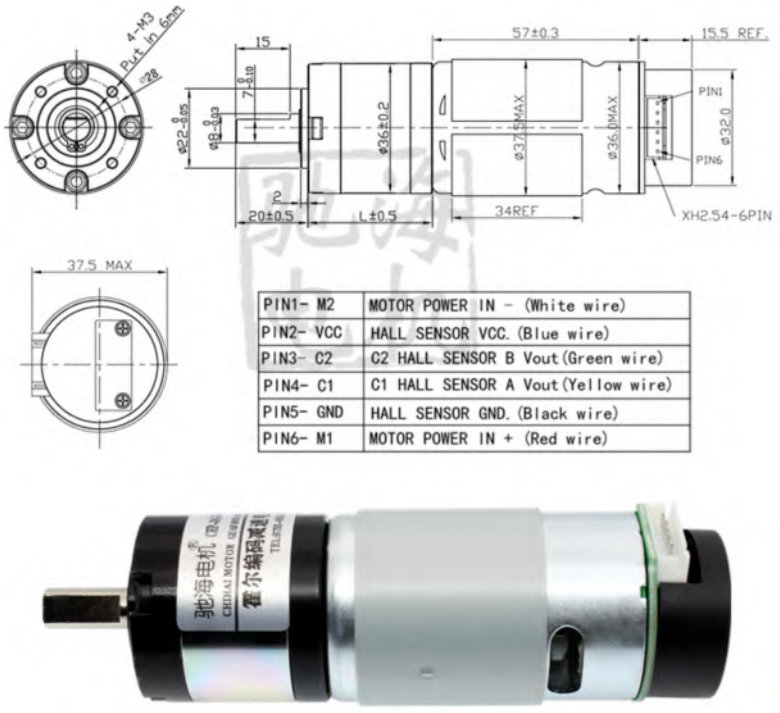






**图 3.12　电机实物图**

### 3.4.2　主要舵机

考虑到实际中机器人机械臂需要高速转动，为保证其转动的正确性，我组采用了型号为 DS3120 的舵机，其具有如下特性。

- 堵转大电流防烧防水
- 金属齿轮不扫齿，虚位小
- 中间金属散热良好
- 大扭矩动力足
- 数字舵机，控制精度高，线性度好


**图 3.13　舵机实物图**

### 3.4.3　储球筐推杆

篮球离开储球筐进入摩擦轮轨道，需要推杆给予一定的动力和位移。我组采用常见的电动推杆完成这一功能。其具有如下特性。

- 限位开关，确保不会空转
- 自带机锁，可以在任意位置停止
- 兼容 24V 电压输入
- 推杆速度恒定，推力达到预期


**图 3.14　推杆实物图**

### 3.4.4　摩擦轮电机和电调

我组采用摩擦轮方式实现投球功能，为此选择了高转速的 MT4114 电机。该电机具有适配的摩擦轮组件和相应电调。

MT4114 的具体参数和示意图如下。

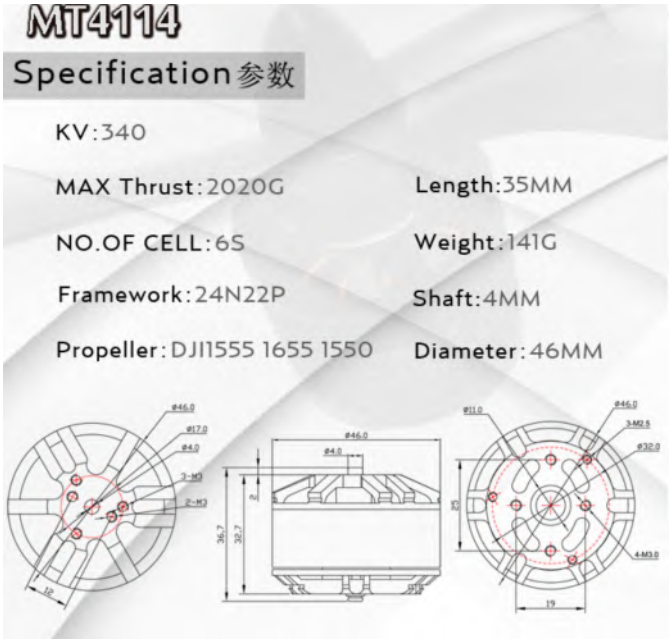



**图 3.15　MT4114**

根据相关数据手册，其配套电调具有 20A 和 60A 两者输出类型。考虑到本

届赛事的实际需要，我们只需要不超过 4A 的电流即可达到预期目的。

其配备 20A 电调参数如下。

| 参数 | 内容 |
| --- | --- |
| 输出能力 | 持续电流 20A，短时电流 25A（不少于 10 秒） |
| 电源输入 | 12V 电压系统 |
| BEC 输出 | 5V @ 2A（线性稳压模式） |
| 尺寸 | 长宽高 42mm × 25mm × 8mm |
| 重量 | 19g，含散热片 |


**图 3.16　MT4114 配套电调**

考虑到航模电机开环控制下转速的不稳定性，我组也准备了另一款电机以备使用。M3508，RoboMaster 适用的直流无刷减速电机，具体参数如下。


| | | |
| --- | --- | --- |
| M3508 减速电机套装参数 | 额定电压 | 24V |
| | 空载转速 | 482rpm |
| | 持续最大扭矩 | 2.8N · m |
| | 2.8Nm下最大转速 | 469rpm |
| | 使用环境温度 | 0~50℃ |
| M3508 直流无刷减速电机参数 | 重量 | 365g |
| | 外径 | 42mm |
| | 总长度 | 98.4mm |
| | 输出轴 | D型带螺纹孔 |
| | 输出轴直径 | 10mm |

**图 3.17　M3508 具体参数**

配套电调参数如下。

| C620 无刷电机调速器参数 | | |
|---|---|---|
| | 额定电压 | 24V |
| | 重量 | 35g |
| | 尺寸（长宽高，不含线） | 49.4*25.8*11.5mm |
| | 带线总长度 | 359mm |
| | 信号类型 | CAN指令、PWM |
| | 最大持续电流 | 20A |

**图 3.18　C620 无刷电机调速器**

以及如下的一款电机，均可以作为备选使用。


**图 3.19　57BL75S10-230TF8 实物图**

| 型号 | 57BL75S10-230TF8 |
|---|---|
| 电压 | 24VDC |
| 转速 | 3000rpm |
| 转矩 | 0.32Nm |
| 电流 | 5.9A |
| 功率 | 100W |
| 峰值转矩 | 0.66Nm |
| 力矩常数 | 0.06N.M/A |
| 反电势常数 | 4.27V/kRPM |
| 转子惯量 | 11.9kg.mm2 |
| 线电阻 | 0.42±10% ohms@25℃ |
| 线电感 | 1.12±20% mH@1kHz |
| 极对数 | 2 |

**图 3.20　57BL75S10-230TF8 参数表**

### 3.4.5　电机驱动

考虑到电机运作时额定电流较大，我组使用基于 BTS7960 芯片的电机驱动模块，其具有如下特点。

- 43A 最大电流驱动
- 双 H 桥双向驱动
- 单片机 5V 隔离防止烧毁
- 宽电压输入 5.5V-27V

**图 3.21　BTS7960 模块实物图**

### 3.4.6　舵机驱动

舵机正常工作电流相较电机来说较小，可以集合入自制 PCB 模块。但是在机器人制作前期，我们仍使用已有的模块，以保证前期工作的安全与稳定。

我组使用基于 XY-2.5AD 芯片的驱动模块，具体特点如下。

- 低导通内阻 MOS 开关管，发热小，效率高
- 双路 2.5A，带有过热保护
- 体积小，质量轻，0.1uA 待机电流
- 双路 H 桥电机驱动

**图 3.22　XY-2.5AD 模块实物图**

### 3.4.7　推杆驱动

推杆正常工作电流取决于具体工作负荷，在制作前期，我们将使用以 L298N 为核心的电机驱动。我组选择的电机驱动具有如下特点。


**图 3.23　L298N 模块实物图和参数**

由于推杆工作负荷较小，实际电流不大，我们将在后期实现 PCB 控制板载驱动，来完成这一功能。

### 3.4.8　CAN 隔离电路模块

由于摩擦轮电调要求 CAN 通信，而 CAN 通信隔离电路要求较高，我组采用已有的模块 TD301DCAN 实现这一功能。实物图如下。


**图 3.24　CAN 隔离通信模块**

## 3.5　系统环境感知

### 3.5.1　姿态感知模块

为辅助循线和机器人姿态调整，我组使用以 MPU9250 芯片为核心的模块 GY-9250 进行姿态感知和方位解算。

模块的主要参数和实物图如下。

| 参数 | 内容 |
|---|---|
| 模块型号 | GY-9250 |
| 使用芯片 | MPU-9250 |
| 供电电源 | 3-5V 内部低压差稳压 |
| 通信方式 | 标准 $I^2C$ / SPI |
| 陀螺仪范围 | ±250　500　1000　2000/s |
| 加速度计范围 | ±2　4　8　16g |
| 磁场范围 | ±4800uT |
| 引脚间距 | 100mil/2.54mm |
| 模块尺寸 | 25mm*15mm |

**表 3.1　GY-9250 参数列表**


**图 3.25　GY-9250 实物图**

具体姿态解算将在下一章节叙述，此处不再展开。

### 3.5.2　红外对管黑度检测

为实现循线功能，机器人需要有效识别地面路线的能力。考虑到地盘的尺寸和限度，我组采用光电对管方法进行地面黑线识别。

光电对管，TCRT5000L，是一种反射光学传感器。对于黑色物体，其具有良好的识别能力。

具体参数和实物图如下。

| 参数 | 内容 |
|---|---|
| 模块型号 | TCRT5000L |
| 供电电源 | 3.3V-5V |
| 感应距离 | 15mm |
| 正向 DC 电流 | 60mA |
| 输出类型 | 光电晶体管 |
| 安装类型 | 通孔 |

**表 3.2　TCRT5000L 参数列表**

**图 3.26　TCRT5000L 实物图**

为了实现模块化轨迹识别，基于上届设计模块，我组重新实现了个性化的红外对管集成板，特点如下。

- $I^2C$/SPI 通信协议
- 八路红外对管集成。
- 8051 单片机电压数据采集。

原理图如下。

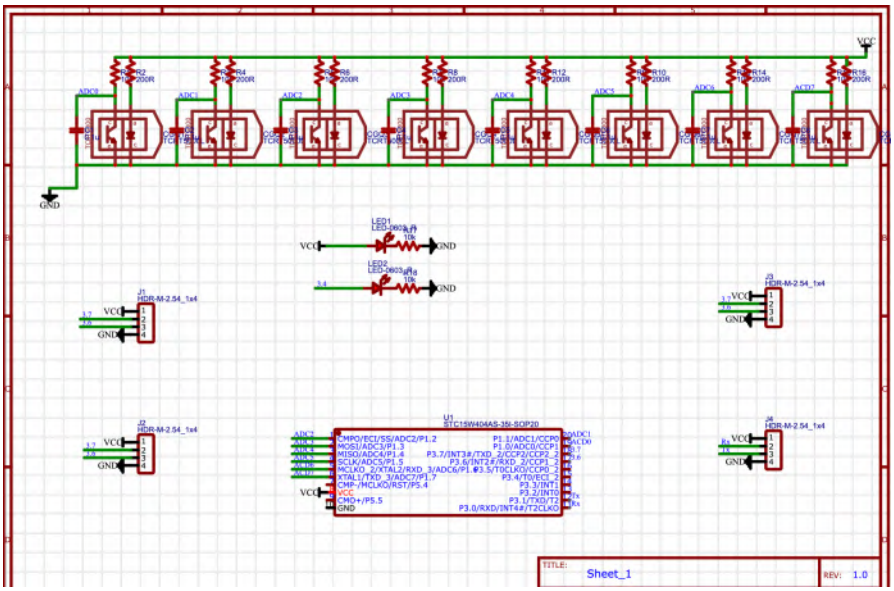



**图 3.27　红外对管电路板原理图**

### 3.5.3　STC15W404AS 单片机

为实现循线功能，机器人需要多组红外对管完成轨迹识别。由于红外对管的工作原理，机器人需要多路 ADC 读取电路电压。于是，基于上届设计模块，我组采用单片机 STC15W404AS 作为 ADC 模块，其具有如下特点。

| 参数 | 内容 |
|---|---|
| 工作电压 | 2.5V-5.5V |
| Flash | 4k bytes |
| SRAM | 512 bytes |
| UART | 1 个 |
| SPI | 1 个 |
| 10bit ADC | 8 通道 |
| PWM/DAC | 3 通道 |

**表 3.3　STC 单片机参数列表**


**图 3.28　STC15W404AS 单片机实物图，SOP20 封装**

基于上届设计模块，我组采用 SOP20 封装。集成板内含有串口烧录端口，可

以直接烧录所需程序。

### 3.5.4　激光测距模块

为实现靠墙测距，控制方位与检测距离等功能，我组采用模块 VL53L0X V2 实现这一功能。

其具体参数和实物图如下。

| 参数 | 内容 |
| --- | --- |
| 通信方式 | $I^2C$ |
| 测量距离 | 3cm-200cm |
| 测量速度 | 20ms(max) |
| 测量精度 | ±3% |
| 工作电压 | DC 3.3V/5V |
| 工作电流 | 12mA - 20mA |
| 工作温度 | -20°C ~ 70°C |

**表 3.4　VL53L0X 参数列表**

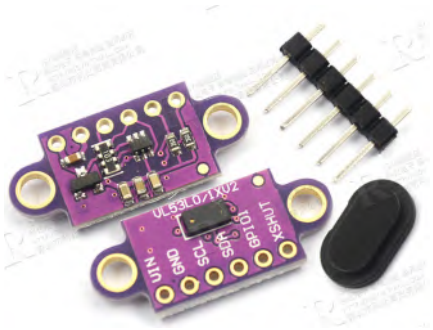



**图 3.29　VL53L0X V2 模块实物图**

### 3.5.5　RFID 读卡模块

本届比赛中具有利用 RFID 读取球体信息的环节。我组采用常见的的 RFID 模块，即 PN5180。使用与否取决于具体调试进度。

具体特点和实物图如下。

- 支持多种协议，ISO/IEC18092.14443 A/B,FeliCa, 15693。
- 增加 DPS 功能，增加接受的稳定度。
- 支持卡模拟和 P2P 模式。
- 使用串行外设接口 SPI。

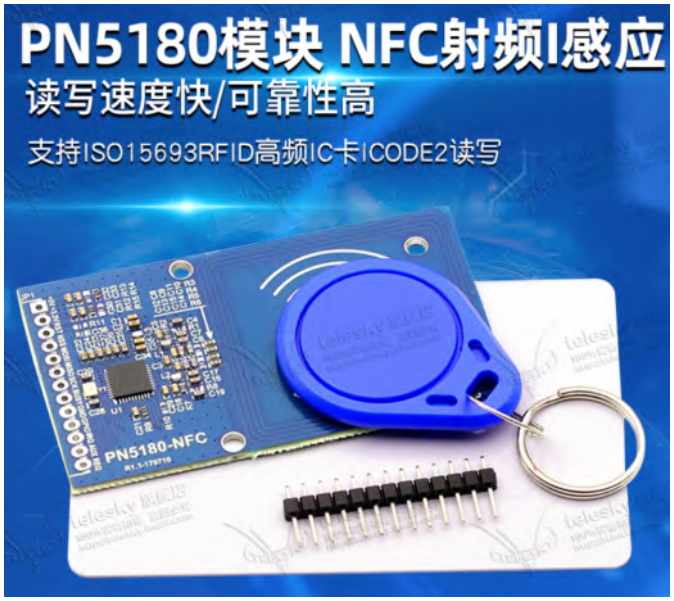



**图 3.30　RFID 读卡模块**

### 3.5.6　NRF24L01 无线通信模块 (仅调试)

注意：本模块仅在调试时使用。正式比赛时将被去除。

为便于调试阶段机器人的控制，参考中国科学技术大学电子设计实践 I 课程内容，我组采用 NRF24L01 作为调试阶段的遥控方式。

NRF24L01 具有如下参数。

**表 3.5　RFID 模块参数列表**

| 参数 | 内容 |
|---|---|
| 工作电压 | 1.9~ 3.6V |
| 工作速率 | 2Mbps |
| 多频点 | 2.4GHz-2.5GHz，多频段可供使用 |
| 功耗 | 多种模式不同功耗，正确配置功耗较低 |
| 特点 | 小型，低应用成本，便于开发 |

**表 3.6　NRF24L01 具体特点**

该模块实物图如下。

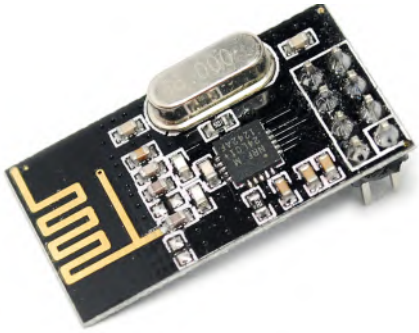


**图 3.31　NRF24L01 无线通信模块**

## 3.6　系统控制

### 3.6.1　自绘主控制板

由于本机器人具有大量模块，杂乱的引线并不利于各部分的调试和运作。为此，我组设计如下运动控制板，其上实现了多种功能。

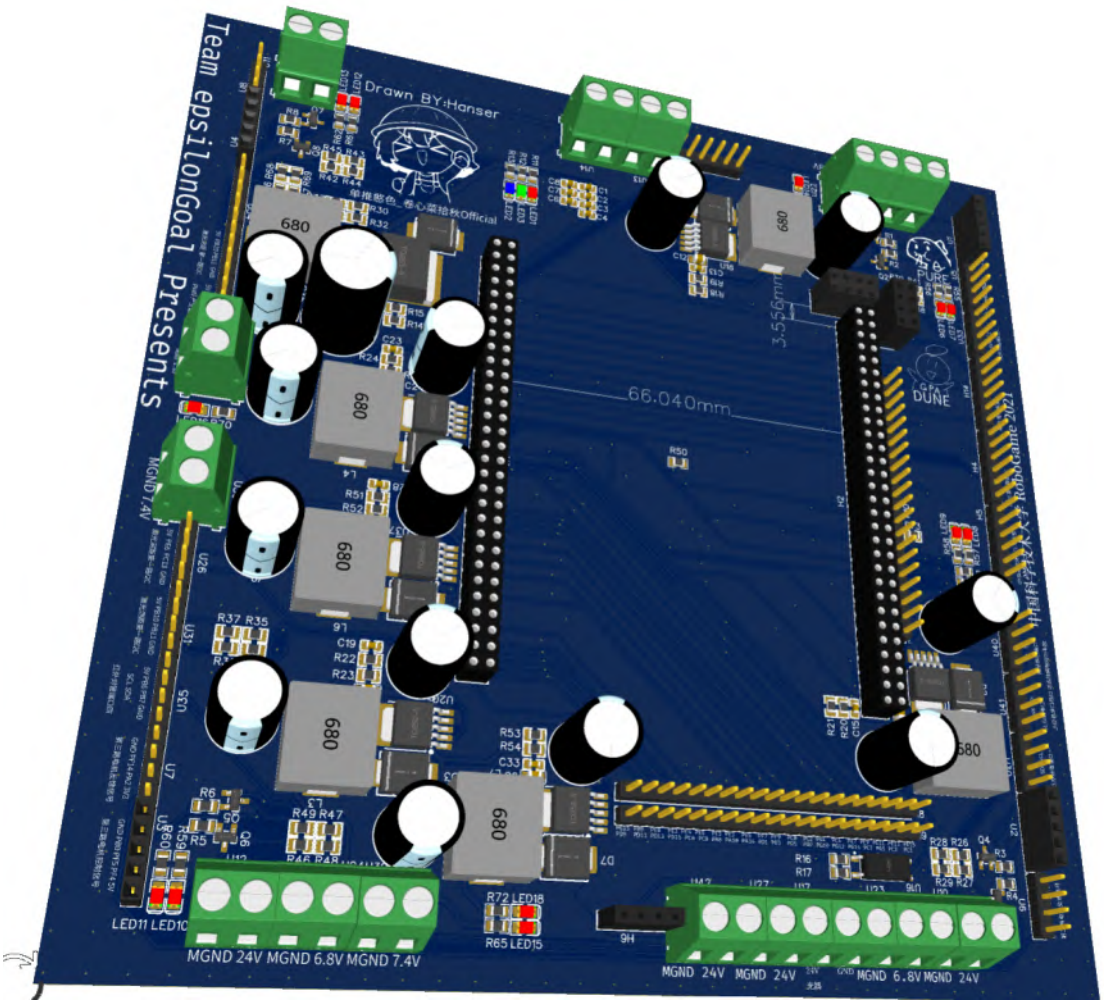



**图 3.32　运动控制板 PCB**

- DC-DC 稳压模块将 24V 转为 6.8V 和 7.4V 实现舵机驱动。
- 四路地盘麦克纳姆轮控制信号。
- 两路机械臂，四组舵机控制信号。
- 两路机械爪二组舵机控制信号。
- 红外循线模块接受信号。
- 串口调试信息输出端口。
- NRF24L01 调试控制信号输入端口。
- 开发板引脚嵌入扩展排母。
- 各路控制信号显示指示灯。
- 激光测距接受信号。
- RFID 模块读取信息接受信号。
- 上位机全双工通信端口。
- 双电压电源输入与滤波。

- 外联 CAN 隔离电路模块。
- 引脚引出。

该控制面板实现了对我组选择的 STM32F103ZET6 开发板的拓展，使之能够充分满足我组机器人实现各种功能的需要。

由于 24V 电压的驱动控制具有一定技术难题，我组只使用控制面板驱动电流相对较小的舵机和推杆，而使用成熟模块驱动电流较大的底盘电机。

注意：其中的所有无线通信模块都将在比赛时去除。

### 3.6.2　舵机 PWM 模块

因此在此次机器人比赛中，我们会使用较多的舵机。每个电机控制信号需要一定频率的 PWM 波。为了合理地生成控制信号，避免达到主控芯片的性能极限，我组使用 PWM 生成模块，参数和实物图如下。

| 特点 |
| --- |
| 3.3V 5V 兼容 |
| $I^2C$ 通信 |
| 1.6kHz 可调频 PWM 输出 |
| 可配置推挽和开漏输出 |
| 反向极性保护 |
| 输出端配置电阻器保护电路 |

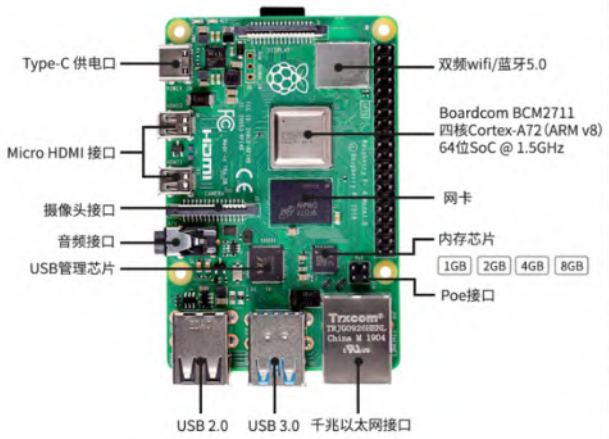
**表 3.7　PWM 模块特点**

**图 3.33　PWM 模块实物图**

## 3.7　上位机与图像采集模块

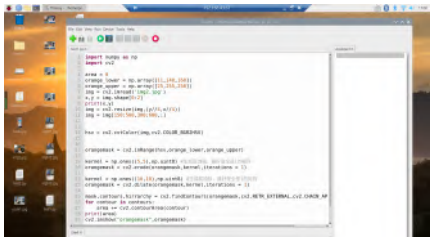
### 3.7.1　树莓派选型与简介

树莓派 (Raspberry Pi) 是一款为学习计算机编程教育而设计的，基于 Linux 系统的微型电脑。相对于面向硬件的 STM32 单片机，树莓派具有更加贴近实际电脑的强大算力，因此在此次机器人比赛中，我们将它作为上位机，进行图像处理操作，并且与 STM32 进行通信，指导机器人的的运动和操作。我们选择树莓派 4b，其实物图和具体参数如下。




| 型 号 | PI3 B | PI3 B+ | PI4 B |
|---|---|---|---|
| 处理器 | 64位1.2GHz四核 | | 64位1.5GHz四核 |
| 运行内存 | 1GB | | 1GB.2GB.4GB.8GB<br>可根据型号选择 |
| 无线WiFi | 802.11n 无线 2.4 GHz | 802.11n 无线 2.4GHz / 5GHz 双频WiFi | |
| 蓝 牙 | 蓝牙4.1 BLE | 蓝牙4.2 BLE | 蓝牙5.0 BLE |
| 以太网网口 | 100Mbps | 300Mbps | 千兆以太网 |
| USB 口 | 4个 USB 2.0 端口 | | 2个 USB 3.0 端口<br>2个 USB 2.0 端口 |
| GPIO 口 | 40个GPIO引脚 | | |
| 视频音频接口 | 1个全尺寸HDMI端口<br>MIPI DSI显示端口<br>MIPI CSI摄像头端口<br>立体声输出和复合视频端口 | | 2个视频和声音<br>micro HDMI端口,<br>最高支持4Kp60,<br>MIPI DSI显示端口<br>MIPI CSI摄像机端口<br>立体声音频和复合视频端口 |
| 多媒体支持 | H.264，MPEG-4解码：1080p30<br>H.264编码：1080p30<br>OpenGL ES： 1.1,2.0graphics | | H.265：4Kp60解码<br>H.264：1080p60解码，1080p30编码<br>OpenGL ES：3.0图形 |
| SD卡支持 | Micro SD卡接口 | | |
| 供电方式 | Micro USB | | USB type C |
| POE | 无 | POE(需额外加模块) | |
| 输入功率 | 5V 2.5A | | 5V 3A |
| 分辨率支持 | 1080分辨率 | | 高达4K分辨率支持<br>双显示屏 |
| 工作环境 | 0-50°C | | |

**图 3.34　树莓派**



树莓派配备使用 USB 摄像头，拍摄清晰度为 720P 高清，内置 CMOS 芯片，图像默认分辨率为 1280×970。该摄像头将拍摄到的图像传给树莓派进行处理，从而使系统获得图像信息。

该摄像头实物图如下。

**图 3.35　USB 摄像头**

# 第 4 章　上位机图像识别主要方法和处理逻辑

由于上位机树莓派的处理相对独立，我们将其独立成章进行简介。

## 4.1　主要介绍

上位机树莓派的主要图像识别工具是基于 Python 的 OpenCV。OpenCV 是一个开源的计算机视觉库，里面里面较为齐全地封装了关于图像处理的库函数，支持 C++ 与 Python 编程。由于 Python 具有 NumPy，开源的数值计算扩展，且 Python 编程相对 C++ 入门相对简易，在本次机器人制作中，我组使用 OpenCV+Python 实现这一功能。

## 4.2　图像识别原理简介

OpenCV 中，一个正常的彩色图片由一个三维矩阵组成，每个维度分别为图片的高度、宽度、通道数。

OpenCV 提供两种色彩空间，即 RGB/BGR 色彩空间和 HSV 色彩空间。一般导入的图片是 RGB 空间，而在 OpenCV 提供的算法中多使用 HSV 空间。RGB 色彩空间包括红色 R、绿色 G 和蓝色 B 三个通道，HSV 色彩空间包括色调 H、饱和度 S 和亮度 V 三个通道。

本次机器人视觉算法中，要求使用算法将正常图片进行二值化处理，生成二值图像，即像素只为 0 或者 255 的单通道图像。这样做的目的是将除目标颜色外的颜色与目标颜色分开，提取掩膜。

为了降低噪声对图像识别的影响，本次算法中将使用腐蚀和膨胀两个方法，平滑二值图像的边缘，提高图像识别的精确度。

由于比赛时会有环境光的影响，图片可能会有一定的颜色偏差，为了使颜色识别更准确，计划使用白平衡算法。白平衡算法有很多，包括均值白平衡算法、完美反射算法、灰度世界算法、基于图像分析的偏色检测及颜色校正方法、动态阈值算法等等。本次机器人视觉算法中计划使用均值白平衡算法。

## 4.3　识别方案

考虑到本届比赛 RFID 协议使用复杂，我们将在比赛中采用两种方案进行篮球识别：颜色识别和条形码识别。考虑到对机器人速度的影响，我组不直接采用条形码识别，而是用之验证取球是否正确。即利用取球的短时间识别条形码，如

果证明不是篮球，识别有误，则进入错误处理阶段，放弃先前的预识别结论。

考虑到颜色识别的诸多优点，计划优先使用颜色识别，条形码识别作为备选方案。以下为我组两种方案的具体设计。

### 4.3.1　单树莓派双摄像头

本方案使用一个树莓派同时打开两个摄像头，在对摄像头编号后，进行两侧图像的获取。通过处理二值图片，计算轮廓的面积来确定图像中球体属性，如果轮廓面积大于一个设定值 A，说明摄像头识别到橙色篮球。如果摄像头未识别到篮球，则轮廓面积会很小。经过实验，可以确定设定值 A，容错率很大。

但是，树莓派中 USB 串口编号问题很难解决，配置环境难度很大。而且识别速度会很慢。

当然可以一个摄像头用树莓派自带摄像头，另一个用 USB 摄像头，应该可以解决此问题。由于此计划的可行性，我们将这种方案作为尝试方案之一。

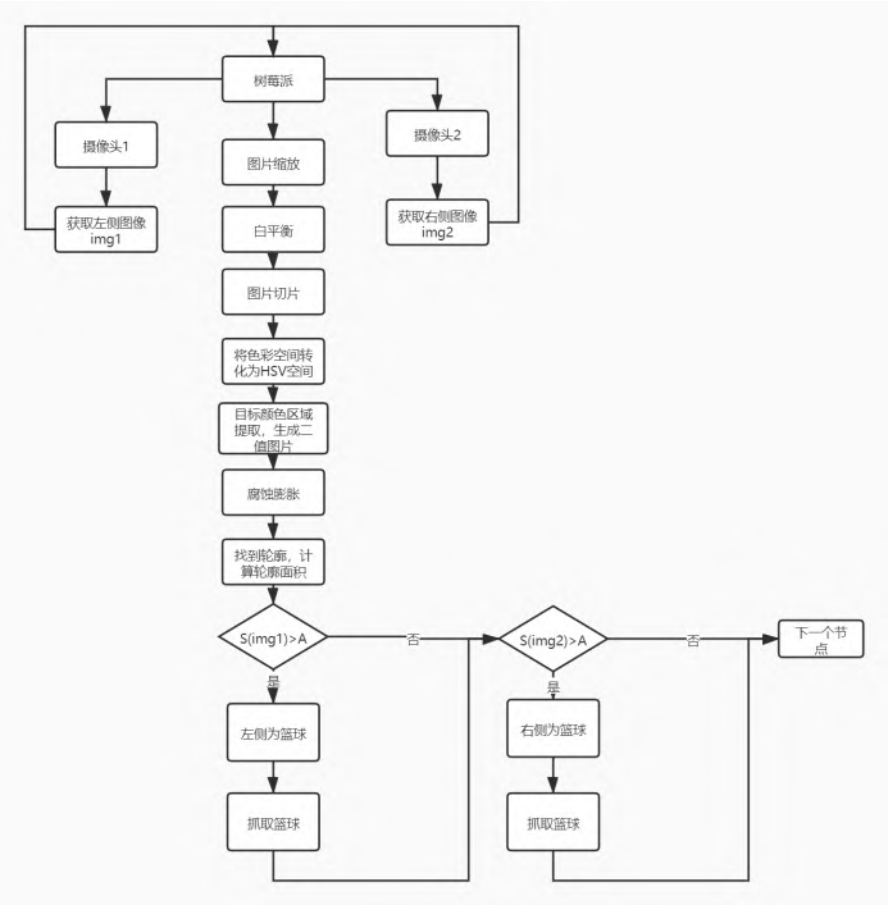



**图 4.1　该方案主要流程**

### 4.3.2　单树莓派单摄像头转动识别

我们用舵机来控制摄像头的转动，将其放置在机器人的中轴线上。假设先识别右边一侧的篮球，摄像头转向右侧，当该侧篮球识别完，则开始识别另一侧的篮球，此时，摄像头转向另一侧。不过这种安装方式需要改变取球策略，即单边取球。

该方案具有的优点是，可以近距离识别，且一个树莓派控制一个摄像头易于实现。缺点是，对电控的难度要求很高，即取球方案设计会相对复杂。

### 4.3.3　单树莓派单摄像头镜面识别

这种方案对机械结构有一些要求，设计如下图。

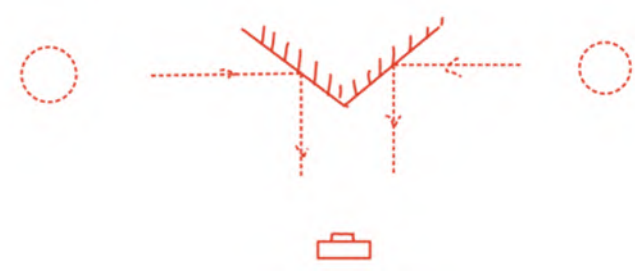


**图 4.2　镜面识别机械结构**

机器人地盘处安装一个由两个面拼接而成的反射镜，夹角 90°。有一个摄像头正对着反射面的中轴线。虚线圆圈部分为机器人两边场地上放置的球。当电控控制车到达一定位置（此位置的确定可在调试过程中尝试），使摄像头能通过反射镜看到两个小球的像。摄像机拍摄，则将两个球的像放置在一张图片中，将图片传到树莓派中处理。处理时，如果二值图片中轮廓面积为两倍的预估面积，那么说明两侧均为篮球，若面积大于一个预估面积，而小于两倍的预估面积，则为一个球，这时只要找到该轮廓中线的横坐标，就很容易确定是左侧还是右侧的球。

该方案具有的优点是，没有方案 2 那样复杂的环境配置，且处理一张图片，对程序的简单易行有很大帮助，很方便的判断两侧球的颜色情况。

缺点是，由于镜子在中心处，距离球稍远，经过计算，视角较小，鲁棒性较差。

当然，如果仅是图像太小，可以在摄像头与镜子之间装一个放大镜，或对图片切片放大均可。具体实现流程如下。

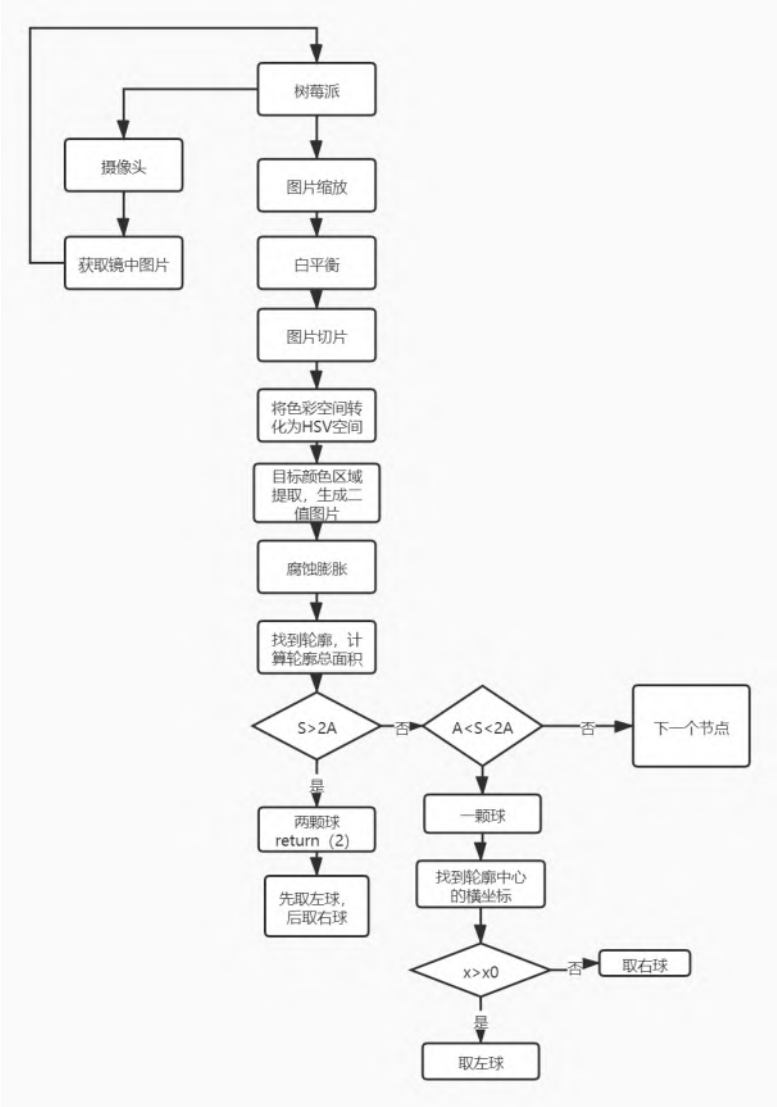



**图 4.3　镜面识别实现流程**

### 4.3.4　近距离扫码

为了防止在之后尝试过程中 1，2，3 方案有很大误差，特别准备了近距离扫描的方案。该方案很耗费时间，因为条形码比较小，要靠近些才能拍得清楚。

## 4.4　识别效果测试

### 4.4.1　颜色识别

我们使用如下的测试代码，进行颜色识别的调试。

```
import numpy as np
import cv2

area = 0
orange_lower = np. array([11, 140, 160])
orange_upper = np. array([25, 255, 250])
img = cv2.imread('img2.jpg')  # 图片读入
x, y = img.shape[0:2]  # 获取图片大小
print(x, y)

img = cv2.resize(img, (y//4, x//4))
img = img[150:500, 300:600, :]  # 图片的切片

hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)  # 将图片转化为HSV空间

orangemask = cv2.inRange(hsv, orange_lower, orange_upper)
# 二值化，提取掩膜
kernel = np.ones((5, 5), np.uint8)  # 腐蚀
orangemask = cv2.erode(orangemask, kernel, iterations=1)

kernel.np.ones((10, 10).np.uint8)  # 膨胀
orangemask = cv2.dilate(orangemask, kernel, iterations=1)

mask, contours, hirrarchy = cv2.findContours(
    orangemask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
#找到轮廓
for contour in contours:
    area += cv2.contourArea(contour)
    # 计算所有轮廓所包区域的总面积
print(area)

cv2.imshow("orangemask".orangemask)  # 显示二值图片
cv2.waitKey(0)
cv2.destroyAllWindows()
```

我们根据场地距离，利用镜面 45º 反射，得到不同角度下的三张反射图像。经调试，可以识别到约为 2000 的总面积。这证明方案 3 具有一定程度的可行性。

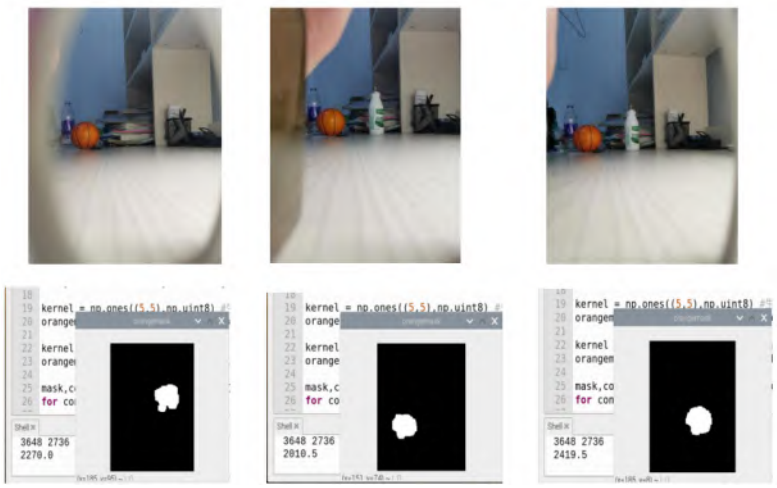



**图 4.4　镜面识别测试**

### 4.4.2　条形码识别

我们使用如下的测试代码，进行条形码识别的调试。

```
import time
import cv2
import numpy as np
import pyzbar.pyzbar as zbar
if __name__ == '__main__':
    font = cv2.FONT_HERSHEY_SIMPLEX
    img = cv2.imread('code.jpg')
    img = cv2.resize(img, (400, 300))
    img_ROI_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    barcodes = zbar.decode(img_ROI_gray)
    # 程序的核心，扫描条形码。注意最好传入灰度图像。
    for barcode in barcodes:
        barcodeData = barcode.data.decode("utf-8")
        barcodeType = barcode.type
        text = "{} ({})".format(barcodeData, barcodeType)
        cv2.putText(img, text, (20, 100), font, 1, (0, 255, 0), 4)
        print("[INFO] Found {} QRcode/barcode: {}".format(
                                                    barcodeType,
                                                    barcodeData))
    cv2.imshow('image', img)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

得到如下的测试结果。可见测试结果十分准确，精度很高。

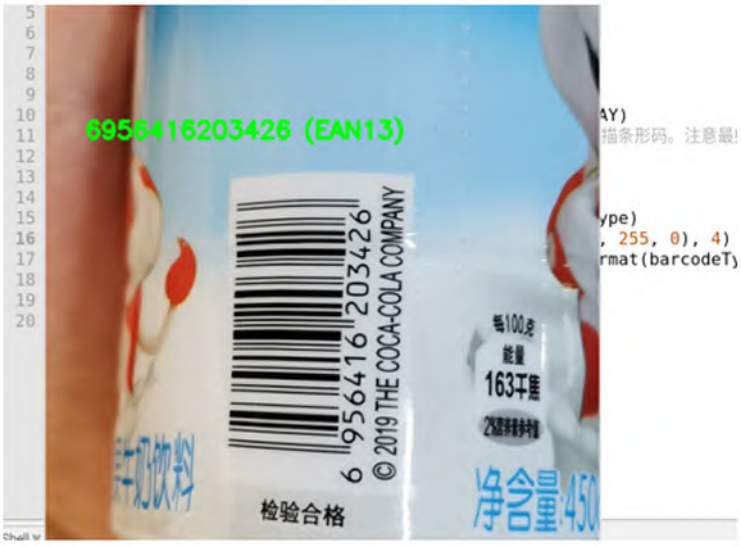



**图 4.5　条形码识别测试**

# 第 5 章　主要程序控制部分设计、原理与实现

## 5.1　STM32HAL 库简要介绍

考虑到本项目需要开启 STM32 大量外设，同时配置大量引脚，为了提高项目程序的有序性，减少不必要的错误使用，我组采用了 ST 公司开发的 HAL 开发库进行项目程序的整体设计。

HAL，全称 Hardware Abstraction Layer，是 ST 公司为 STM32 推出的抽象层嵌入式软件，可以更好的确保跨 STM32 产品的最大可移植性。

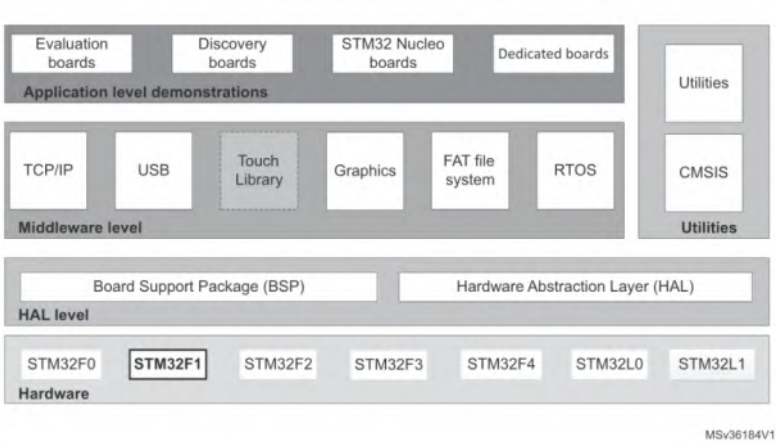



**图 5.1　HAL 库的抽象层结构**

ST 公司为 HAL 库开发了提供了全新的工具链，其中对我组项目开发具有重要意义的是 STM32CubeMX。该软件提供了简单、方便、可视化且直观的方式，完成 MCU 选型、引脚配置、时钟树初始化、外设配置等一系列工作，省去了大量时间和精力手动操作这一过程。

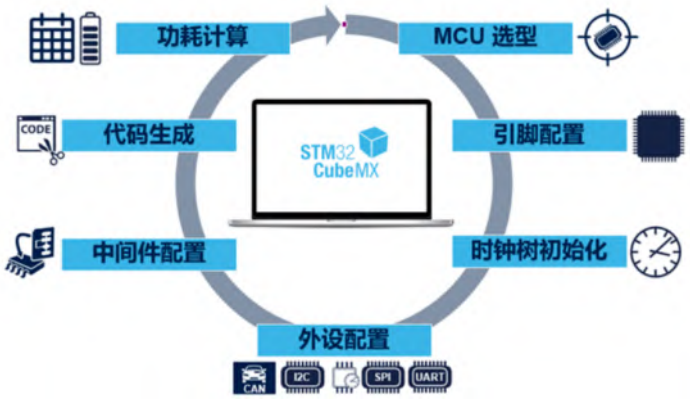



**图 5.2　STM32CubeMX 基本功能**

与标准外设库不同的是，HAL 库采取了面向对象的操作思想。外设的初始配置和实例化，可以具象为一个句柄。HAL 库提供了相应的`HAL_XXX_YYY(XXX_HandleTypeDef *handle, ...)`函数，在进行外设操作时无需具体到芯片特性或寄存器，便于提高可移植性。

同时，HAL 库将 MCU Specific Package（即 MSP）抽象出来，可以在不同 STM32 设备中允许，而无需修改除 MSP 函数以外的代码。

同时，相对于标准外设库的 Handler 函数，HAL 库提供的接口函数为 Callback 函数，进一步简化了中断的处理过程，便于用户应用层的开发。

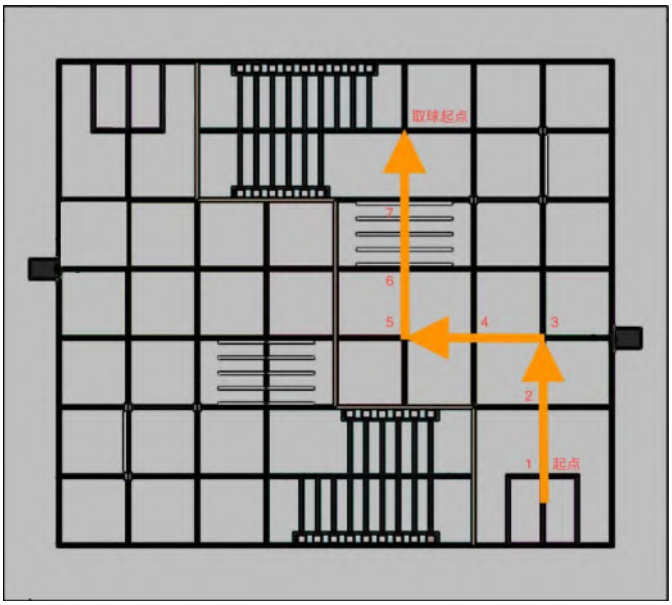
## 5.2　主要运动流程与规划

### 5.2.1　比赛流程

比赛开始后，机器人沿指定路线经过障碍到达取球区域，在取球区域取球结束后经过障碍到达投篮区域（3 分区），投篮结束原路返回取球区域，重复进行 4 次。

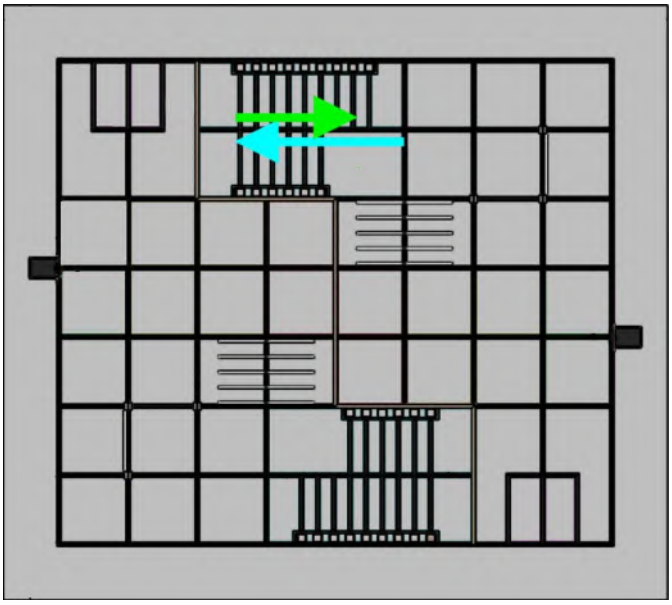
### 5.2.2　路线详解

#### 1. 循线过障至取球起点

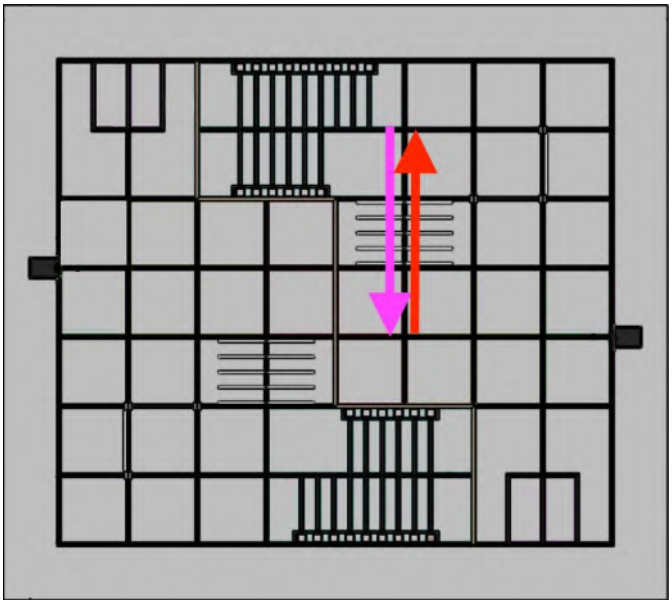



比赛开始，机器人从起点开始直行，在第 3，5 个节点改变运动方向（对麦克纳姆轮进行控制，机器人本身并不旋转）

2. 完成第一次取球

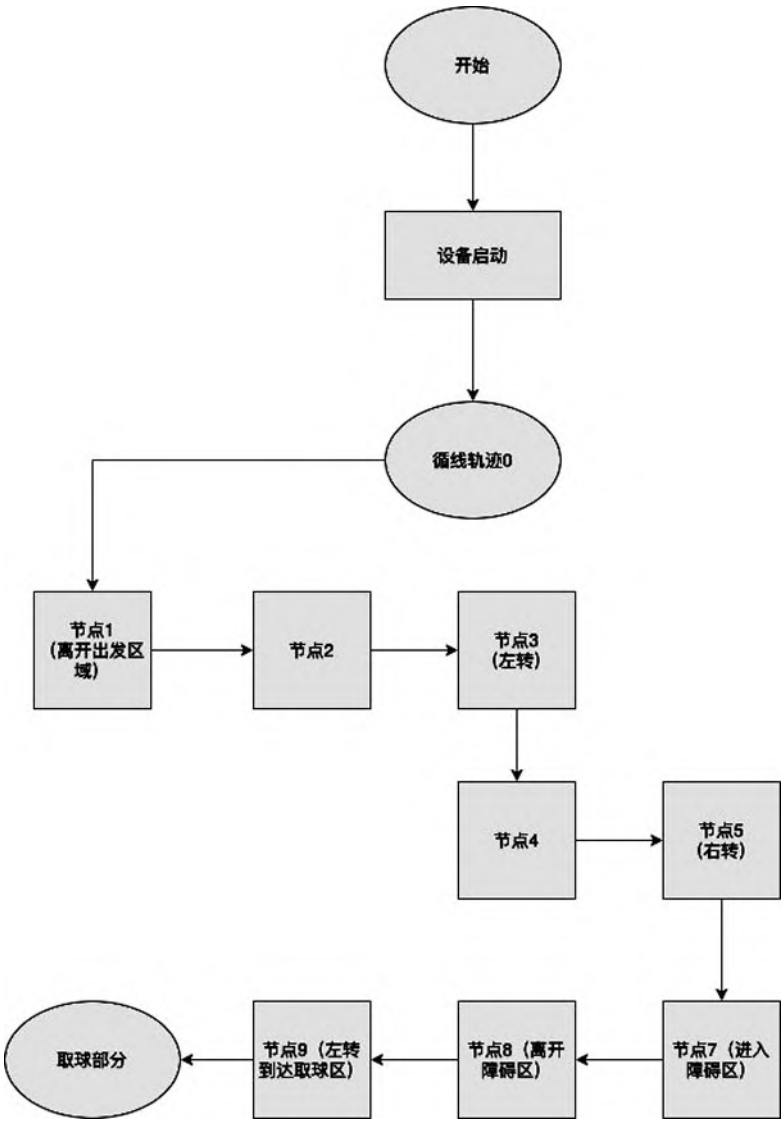


3. 循线过障，完成投篮，进入下一个循环

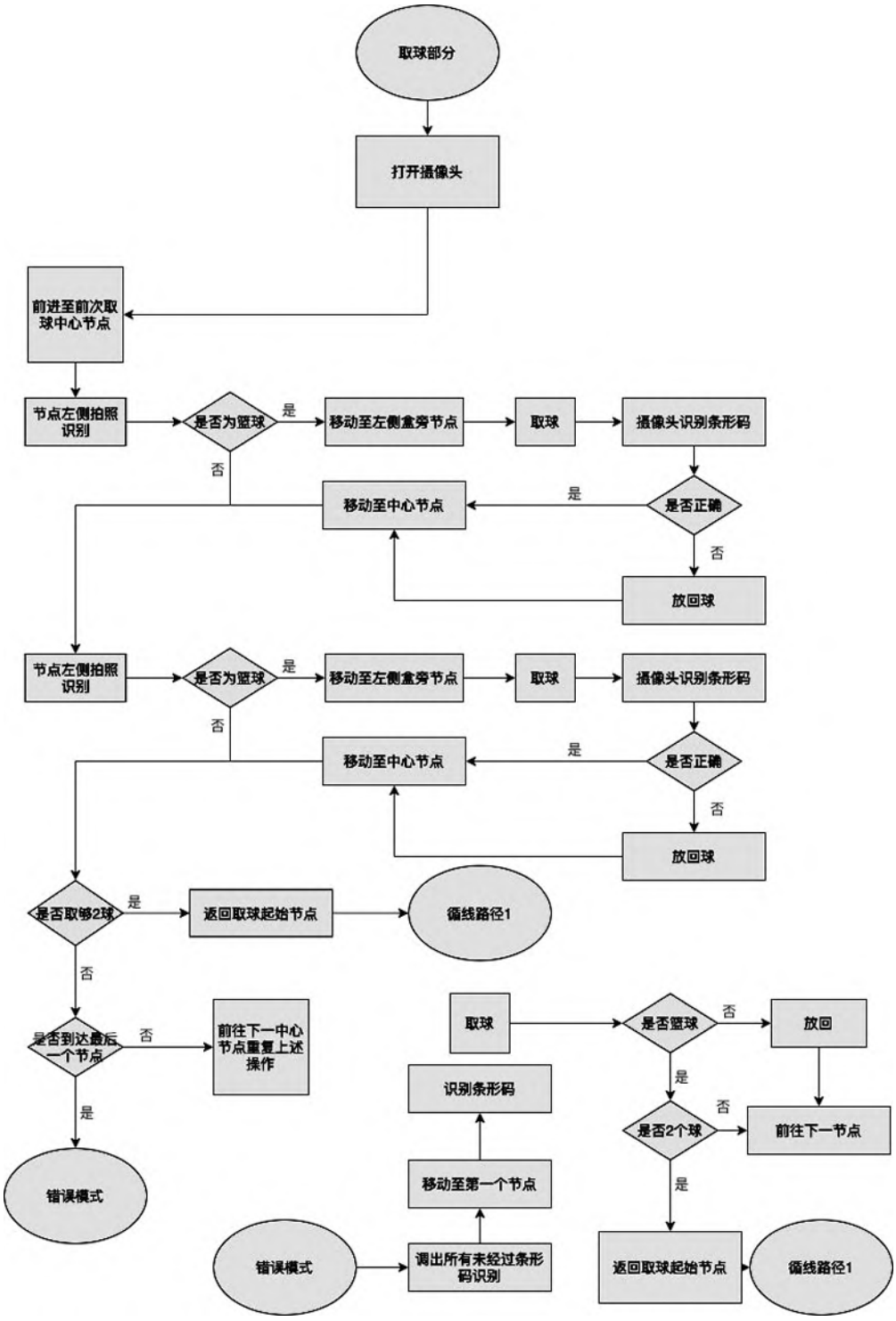


### 5.2.3　总流程

1. 循线过障至取球起点



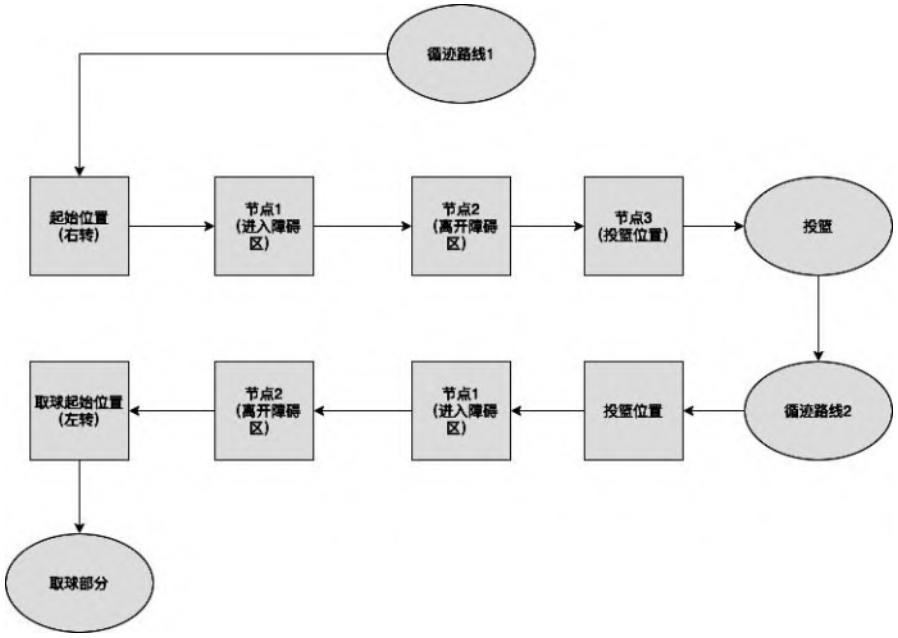
2. 进入取球部分，逐个节点检测并取球




3. 异常处理

进入“错误模式”，机器人在这个模式中不使用颜色识别，仅通过识别条形码在未扫描条形码的位置完成的位置开始逐个检测，成功取得两个球后便返回起始位置，进行循线，到达投篮场地。

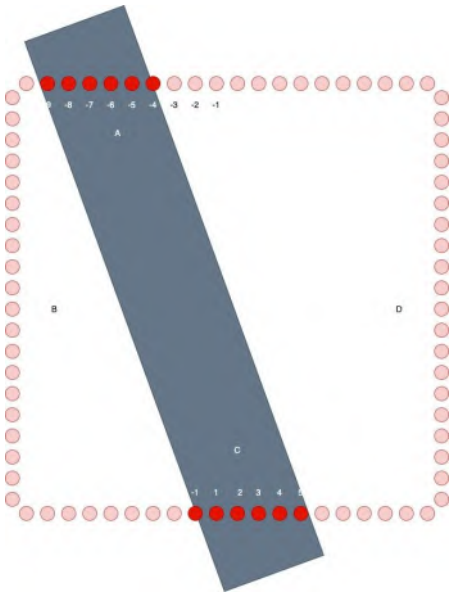
4. 循线往返投篮与取球场地




## 5.3　系统姿态感知与判断

### 5.3.1　红外对管与姿态感知模块

红外对管可以接收不同强度的红外光，光强越强，输出电流越高。这样可以检测出黑线。我们选择是 STC15 单片机的八路 ADC 对红外对管的输出电平进行检测（替代运放）。在底盘上我们每隔 1cm 放置 1 组红外对管，根据其输出可以判断黑线的相对位置和方向，从而经过数学运算得到机器人实际位置。



如图，黑色部分为地面黑线，红外对管在四侧分布，我们为它们命名为ABCD。四侧在红外对管识别到黑线时，返回其所在位置的值，最后我们将每侧的返回值算平均，作为这一侧的输出值。在如图情况下，$A = (-9-8-7-6-5-4)/6 = -6.5$, $C = (-1+1+2+3+4+5)/6 = 2.5$。通过这两个值我们可以计算出机器人方向和位置的偏移量。设位置的偏移量是 $E_1$, 方向（角度）的偏移量是 $E_2$。则 $E_1 = (A+C)/2 = -2$, 将其返回给控制横向位置的 PID 程序中调整机器人的位置；而 $E_2 = (A-C)/2 = -4.5$, 将其返回给控制角度的 PID 程序中调整机器人的行进方向。

同时姿态感知模块也可以得到机器人的转动角度。姿态感知模块返回 3 组数据，分别是：

1. 加速度 3 轴数据
2. 陀螺仪 3 轴数据
3. 磁力计 3 轴数据

由于机器人并不涉及攀爬等运动，故并不用接收 Z 轴和加速度的数据。由于磁力计容易受到场地等原因的干扰，我们优先考虑陀螺仪返回的数据，通过 X 和 Y 方向的返回值求反正切, 即可计算出角度的绝对大小，与开始行驶时的角度作差即可得到相对的角度，便于修正红外对管返回的角度值。

我们根据姿态感知模块返回参数使用 PID 算法进行转动角度控制，并利用红外对管对位置控制和及转动角度的验证，最终使机器人位置落在姿态感知模块和红外对管的误差范围之内。

## 5.4　电机运作驱动与检测

### 5.4.1　PID 控制原理和方法

**PID 控制器**，即比例-积分-微分控制器，是由比例单元（Proportional）、积分单元（Intergral）和微分单元（Derivative）组成的控制器。PID 控制器主要适用于基本上呈现线性关系，且动态特性不随时间变化的系统。在具体的使用中，可以通过调整三个单元的增益 $K$，即 $K_p, K_i, K_d$ 来调定器特性。我们有如下的控制理论公式。

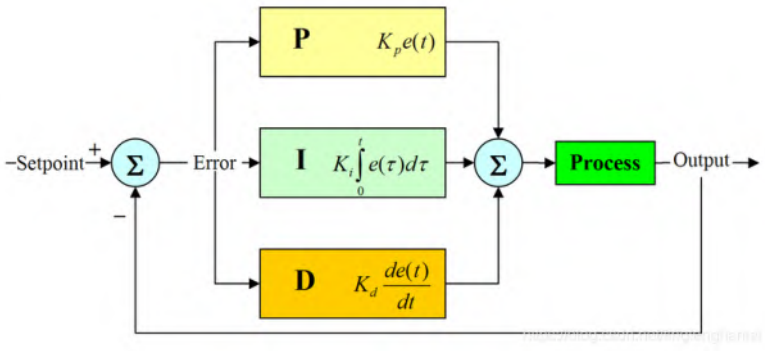



**图 5.3　PID 控制理论**

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t)\mathrm{d}t + T_d\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right)$$
$$= K_p e(t) + K_i\int_0^t e(t)\mathrm{d}t + K_d\frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

其中参数如下。

$K_p$：比例增益，是调适参数

$K_i$：积分增益，也是调适参数

$K_d$：微分增益，也是调适参数

$e$：误差 = 设定值（SP）- 回授值（PV）

$t$：目前时间

$\tau$：积分变数，数值从 0 到目前时间 t

### 1. 调参方法

在机器人上，考虑到地面的材质变化，我们采用人工调整的办法实现上述参数调定。

由于 4 个麦克纳姆轮可能存在承重不同的问题，我们对 4 个轮分别加入独立的 PID 算法控制。具体方法为先将 $K_i$ 及 $K_d$ 设为零，增加 $K_p$ 一直到回路输出震荡为止。之后再将 $K_p$ 设定为 “$\frac{1}{4}$ 振幅衰减” ① 增益的一半，然后增加 $K_i$ 直到一定时间后的稳态误差可被修正为止。若 $K_i$ 造成不稳定，最后若有需要，可以增加 $K_d$，但是应确保在负载变动后回路可以够快的回到其设定值。

① 使系统第二次过冲量是第一次的 $\frac{1}{4}$

### 2. 参数影响

| 调整方式 | 上升时间 | 超调量 | 安定时间 | 稳态误差 | 稳定性 |
|---|---|---|---|---|---|
| $\uparrow K_p$ | 减少 ↓ | 增加 ↑ | 小幅增加 ↗ | 减少 ↓ | 变差 ↓ |
| $\uparrow K_i$ | 小幅减少 ↘ | 增加 ↑ | 增加 ↑ | 大幅减少 ↓↓ | 变差 ↓ |
| $\uparrow K_d$ | 小幅减少 ↘ | 减少 ↓ | 减少 ↓ | 变动不大 → | 变好 ↑ |

### 3. 伪代码

PID 控制流程较为简单，只需要根据如下伪代码，编写适合的程序即可。

```
previous_error := 0  // 上一次偏差
integral := 0    // 积分和

// 循环
// 采样周期为 dt
loop:
  // setpoint 设定值
  // measured_value 反馈值
     error := setpoint - measured_value // 计算得到偏差
     integral := integral + error × dt // 计算得到积分累加和
     derivative := (error - previous_error) / dt // 计算得到微分
     output := Kp × error + Ki × integral + Kd × derivative
     // 计算得到 PID 输出
     previous_error := error
     // 保存当前偏差为下一次采样时所需要的历史偏差
     wait(dt) // 等待下一次采用
     goto loop
```

## 5.4.2　编码器读取实际速度

编码器是一种将角位移或者角速度转换成一连串电数字脉冲的旋转式传感器，我们可以通过编码器测量到底位移或者速度信息。

霍尔编码器是一种通过磁电转换将输出轴上的机械几何位移量转换成脉冲或数字量的传感器。霍尔编码器是由霍尔码盘和霍尔元件组成。霍尔码盘是在一定直径的圆板上等分地布置有不同的磁极。霍尔码盘与电动机同轴，电动机旋转时，霍尔元件检测输出若干脉冲信号，为判断转向，一般输出两组存在一定相位差的方波信号。

电机使用 AB 相增量式霍尔编码器，可以由 AB 双相确定电机的正、反向转动。A 相超前 B 相 1/4 个周期则是正向，落后 1/4 个周期则为反向，对需要改变转动方向的麦克纳姆轮是十分有用的。

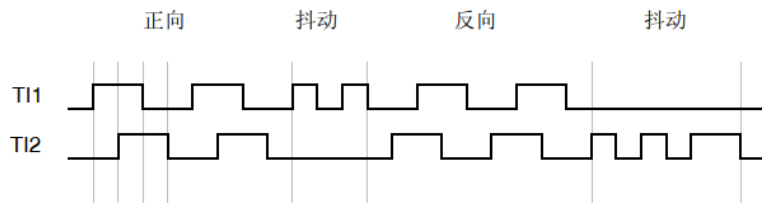



**图 5.4　AB 相增量式霍尔编码器示意图**

TODO：图片解算逻辑

## 5.5　麦克纳姆轮的运动解算

机器人的运动状态，可以通过三个量来进行描述 $v_x, v_y, \omega_z$，分别表示 X 轴方向上的平动，Y 轴方向的平动，以及围绕 Z 轴旋转[①]。

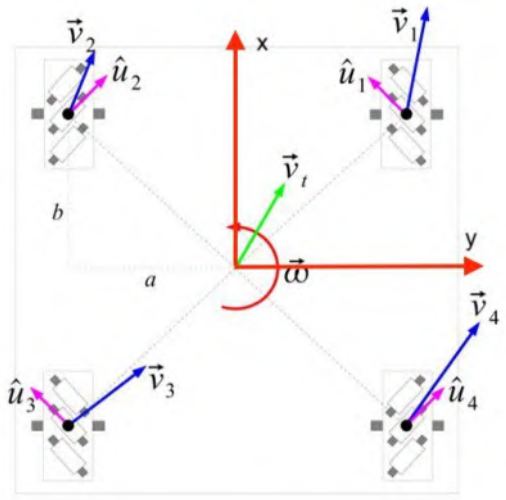



**图 5.5　麦克纳姆轮组坐标系示意图**

### 5.5.1　x 轴方向平动

机器人纵向向前运动时，只需要四个电机以轮毂轴心为圆心，向 X 轴正方向旋转麦克纳姆轮。向后运动时，只需要四个电机以轮毂轴心为圆心，向 X 轴负方向旋转麦克纳姆轮。此时，底盘期望速度只有 X 轴方向的速度。设 $\vec{V}$ 是小车的状态量，有如下公式。

$$\vec{V} = [v_x, 0, 0]$$

[①]逆时针方向为正

可以得到此时的四个麦克纳姆轮的期望转速，其中 $k_x$ 为系数，需根据实际情况调定。

$$\begin{cases} v_1 = v_x \times k_x \\ v_2 = v_x \times k_x \\ v_3 = v_x \times k_x \\ v_4 = v_x \times k_x \end{cases}$$

### 5.5.2　y 轴方向平动

机器人需要横向沿着 Y 轴正方向运动时，1、3 号轮沿着轮毂轴向着 X 轴负方向旋转，2、4 号轮沿着轮毂轴向着 X 轴正方向旋转；小车需要沿着 Y 轴负方向运动时，1、3 号轮和 2、4 号轮的旋转方向相反。

此时，底盘期望速度只有 Y 轴方向的速度，小车的状态向量为

$$\vec{V} = [0, v_y, 0]$$

可以得到此时的四个麦克纳姆轮的期望转速，其中 $k_y$ 为系数，需根据实际情况调定。

$$\begin{cases} v_1 = -v_y \times k_y \\ v_2 = v_y \times k_y \\ v_3 = -v_y \times k_y \\ v_4 = v_y \times k_y \end{cases}$$

### 5.5.3　z 轴方向转动

麦克纳姆轮也方便了机器人绕 Z 轴转动。注意，X，Y，Z 三轴构成笛卡尔坐标系。机器人逆时针旋转要求 1、2 轮向 X 轴正方向旋转，3、4 轮向 X 轴负方向旋转；顺时针旋转要求，1、2 轮和 3、4 轮的旋转方向相反。

机器人此时的状态向量为

$$\vec{V} = [0, 0, \omega_v]$$

四个轮子的期望转速为

$$\begin{cases} v_1 = -\omega_v \times (a + b) \\ v_2 = -\omega_v \times (a + b) \\ v_3 = \omega_v \times (a + b) \\ v_4 = \omega_v \times (a + b) \end{cases}$$

其中，a 为任意一个麦克纳姆轮到 X 轴的距离，b 为任意一个麦克纳姆轮到 Y 轴的距离。

### 5.5.4　速度合成

简单将上述内容相加合成，有状态向量

$$\vec{V} = [v_x, v_y, \omega_v]$$

对应麦克纳姆轮转速为

$$\begin{cases} v_1 = v_x \times k_x - v_y \times k_y - \omega_v \times (a+b) \\ v_2 = v_x \times k_x + v_y \times k_y - \omega_v \times (a+b) \\ v_3 = v_x \times k_x - v_y \times k_y + \omega_v \times (a+b) \\ v_4 = v_x \times k_x + v_y \times k_y + \omega_v \times (a+b) \end{cases}$$

仅做好运动模型的分解是不足以达成运动目的的。实际控制过程中，需要电机快速的响应到期望的转速，这样就要求对每个电机加上单独的 PID 控制。

## 5.6　异常控制处理和反馈

由于机器人在比赛过程中可能会出现突发情况，可能是场地原因，也可能是代码出现未发现的问题。部分错误我们可以通过进入错误模式进行调整，但若出现不可以自动调整解决的问题，我们为防止出现进一步的事故，应当让机器人立刻停止。

### 5.6.1　识别异常

由于使用摄像头对各个球体进行颜色识别会被环境因素干扰（光源，阴影），在调整完成软件参数后仍有可能会出现错误识别。如果检测到最后一组节点未检测到 8 个篮球，在这时就应该逐一对通过识别条形码进行判断。

### 5.6.2　行进异常

由于循线部分可能存在工作异常，导致“脱轨”。我们设计若红外对管范围内未出现黑线，机器人立刻停止工作。

## 5.7　行进中停靠控制

机器人的运动大致分类为启动、运行和停靠阶段。前两部分相对简易，只需控制电机驱动，并利用姿态感知模块利用负反馈调整行进方向即可。

对于停靠阶段，由于本届比赛的特殊性，为达成赛事要求，我们对停靠精度有不同的要求，具体任务如下。

1. 普通停止

机器人需要在改变运动方向时，需要在黑线交点处停止，但并不需要机器人准确地停到交点的位置。我们控制机器人在接近目标交点时进行匀减速运动，并在检测到目标交点时开始改变运动方式，利用 PID 算法可以平滑改变机器人的运动方式。

2. 准确停止

我们需要机器人在特定的黑线交点准确停靠，并且在投篮时需要位置和方向完全确定。我们使用红外对管，可以搭配使用姿态感知模块在相当准确的范围内固定机器人的角度、位置。

3. 激光测距对准

由于在取球时不可以撞上盒子，也就是说无法在目标位置左右移动调整位置。我们使用激光测距模块测量机器人与盒子的距离，在接近盒子的位置降低速度，逐渐接近盒子并停止在固定的距离，同时固定的距离也方便调整机械臂的姿态。

## 5.8　机械臂及推杆控制

由于机械臂和推杆使用舵机为开环控制，故不需要写 PID 算法。机械臂有三个部位需要控制，实现机械臂的翻转，机械爪的旋转以及机械爪的开合。

1. 抓取

机器人在取球盒旁停稳后，机械臂从内向外翻转。在接近取球盒时机械爪向下翻转，保持机械爪张开，在其余两个舵机运动完成并固定后，控制机械爪舵机使机械爪收紧，抓起篮球。

2. 放置

在机器人停止过程中，取球侧机械臂由外向内翻转，同时机械爪向内翻转，放置篮球至储球篮上方，控制机械爪张开，使篮球落入储球篮中。

## 5.9　系统遥控和调试信息输出

### 5.9.1　NRF24L01 遥控方法和逻辑

注意：本模块仅在调试时使用。正式比赛时将被去除。

NRF24L01 无线模块可以在机器人未完成时帮助我们调试代码。

一个基本的 NRF 模块的原理图如下。

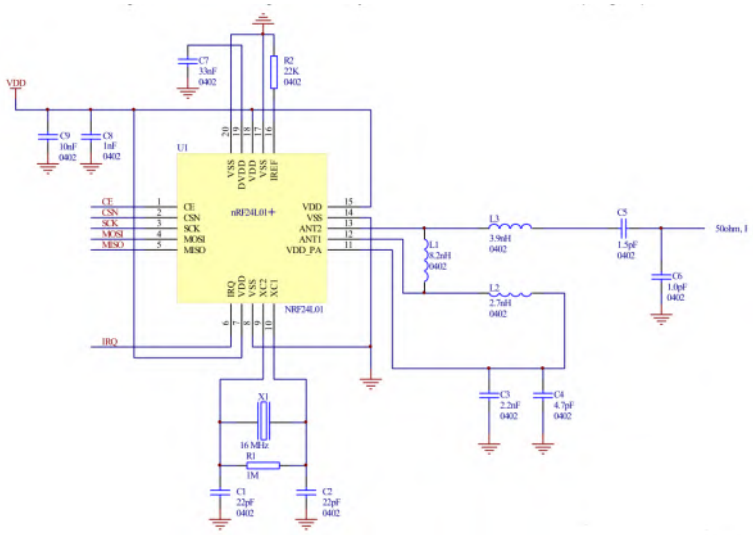



**图 5.6　NRF24L01 无线模块原理图**

NRF24L01 无线模块有如下工作模式。

| 模式 | PWR_UP | PRIM_RX | CE | FIFO寄存器状态 |
|---|---|---|---|---|
| 接收模式 | 1 | 1 | 1 | - |
| 发射模式 | 1 | 0 | 1 | 数据在TX FIFO 寄存器中 |
| 发射模式 | 1 | 0 | 1→0 | 停留在发送模式，直至数据发送完 |
| 待机模式2 | 1 | 0 | 1 | TX FIFO 为空 |
| 待机模式1 | 1 | - | 0 | 无数据传输 |
| 掉电 | 0 | - | - | |

**图 5.7　NRF24L01 无线模块原理图**

NRF24L01 无线模块可以进行如下的操作流程。

发送流程为：把地址和要发送的数据按时序送入 NRF24L01；配置寄存器，使之进入发送模式；MCU 将 CE 置为高，激发 Enhanced ShockBurstTM；发射完成，NRF24L01 进入接受应答状态。

接收流程为：配置接收地址和要接收的数据包大小；配置寄存器，使之进入接收模式，把 CE 置高；130us 后,NRF24L01 进入监视状态，等待数据包的到来；当接收到正确的数据包，NRF24L01 自动移去字头、地址和 CRC 校验位；NRF24L01 通过把 STATUS 寄存器的 RX_DR 置位通知 MUC; MCU 把数据从 FIFO 读出；

我们通过 NRF24L01 模块发送如下数据结构进行调试时的控制。

```
typedef enum { CmdStageOne = 0xf0, CmdStageTwo = 0x0f } CmdStage;
typedef uint8_t CmdCount;
typedef struct {
    uint8_t ordinal;          // 要改变的底盘电机编号 和 PID相应参数编号，
                              // 值为0xXY := 0xX0 | 0x0Y，其中X为电机编号，
                              // Y 为1或2或3，对应P、I、D的系数
                              // X，Y任何一个数为零，认为不做修改
    double new_value;         // 新值
} PIDChangeParam;

struct {
    CmdStage Stage;                 //  命令类型
    CmdCount Count;                 //  命令编号，用于对齐和调试
    int16_t speed_x;                //  x轴行进方向
    int16_t speed_y;                //  y轴行进方向
    uint8_t left_wing[3];           //  左翼三舵机驱动
    uint8_t right_wing[3];          //  右翼三舵机驱动
    uint8_t wheel_speed;            //  摩擦轮速度
} Remote_CtrlSignalOne;

struct {
    CmdStage Stage;      //  命令类型
    CmdCount Count;      //  命令编号，用于对齐和调试
    PIDChangeParam ChangeCmd[2];          //  PID改变的命令
} Remote_CtrlSignalTwo;
```

### 5.9.2　调试信息输出方法

我们使用蓝牙串口模块实现调试信息的输出。在重定义`fputc()`函数和`fgetc()`为串口输出输入后，我们便可以使用标准库函数`printf()`进行信息的输出。这样，便可以在程序调试过程中实时显示各种参数。

同时，我们也可以使用 USART 接收中断，来接受调试信息任务，实现特性化的调试输出。

以下是使用 SPL 库实现的接收中断。注意到其中调用了`Debug_CommandHandler()`函数，用于处理调试命令。我们只需要将该片段修改为 HAL 库特性，再实现自己的`Debug_CommandHandler()`即可。

```
void DEBUG_USART_IRQHandler(void) {
    u8 tmp;
    if (USART_GetITStatus(DEBUG_USARTx, USART_IT_RXNE) != RESET) {
        USART_ClearITPendingBit(DEBUG_USARTx, USART_IT_RXNE);
        Debug_USART_BufferPush(tmp = USART_ReceiveData(DEBUG_USARTx));
        if (tmp == '\n') Debug_CommandHandler();
    }
    // https://blog.csdn.net/origin333/article/details/49992383
    // 检查 ORE 标志
    if (USART_GetFlagStatus(DEBUG_USARTx, USART_FLAG_ORE) == SET) {
        USART_ClearFlag(DEBUG_USARTx, USART_FLAG_ORE);
        USART_ReceiveData(DEBUG_USARTx);
    }
}
```

## 5.10　各路通信方法与逻辑

### 5.10.1　STM32 主要外设通信协议

#### 1. $I^2C$ 通信协议

循线模块与主单片机之间通信使用 $I^2C$。主器件用于启动总线传送数据，并产生时钟以开放传送的器件，此时任何被寻址的器件均被认为是从器件。在总线上主和从、发和收的关系不是恒定的，而取决于此时数据传送方向。如果主机要发送数据给从器件，则主机首先寻址从器件，然后主动发送数据至从器件，最后由主机终止数据传送；如果主机要接收从器件的数据，首先由主器件寻址从器件。然后主机接收从器件发送的数据，最后由主机终止接收过程。在这种情况下，主机负责产生定时时钟和终止数据传送。

#### 2. USART/UART 通信协议

USART 收发模块一般分为三大部分：时钟发生器、数据发送器和接收器。控制寄存器为所有的模块共享。时钟发生器由同步逻辑电路（在同步从模式下由外部时钟输入驱动）和波特率发生器组成。发送时钟引脚 XCK 仅用于同步发送模式下，发送器部分由一个单独的写入缓冲器（发送 UDR）、一个串行移位寄存器、校验位发生器和用于处理不同帧结构的控制逻辑电路构成。使用写入缓冲器，实现了连续发送多帧数据无延时的通信。接收器是 USART 模块最复杂的部分，最主要的是时钟和数据接收单元。数据接收单元用作异步数据的接收。除了接收单元，接收器还包括校验位校验器、控制逻辑、移位寄存器和两级接收缓冲器（接收 UDR）。接收器支持与发送器相同的帧结构，同时支持帧错误、数据溢出和校验错误的检测。

#### 3. SPI 通信协议

SPI 设备间的数据传输之所以又被称为数据交换，是因为 SPI 协议规定一个 SPI 设备不能在数据通信过程中仅仅只充当一个“发送者”或者“接收者”。在每个时钟周期内，SPI 设备都会发送并接收一个 bit 大小的数据，相当于该设备有一个 bit 大小的数据被交换了。一个从设备要想能够接收到主设备发过来的控制信号，必须在此之前能够被主设备进行访问。所以，主设备必须首先通过 SS/CS pin 对从设备进行片选，把想要访问的从设备选上。在数据传输的过程中，每次接收到的数据必须在下一次数据传输之前被采样。如果之前接收到的数据没有被读取，那么这些已经接收完成的数据将有可能会被丢弃，导致 SPI 物理模块最终失效。因此，在程序中一般都会在 SPI 传输完数据后，去读取 SPI 设备里的数据，即使这些数据在我们的程序里是无用的。

#### 4. CAN 通信协议

摩擦轮所使用的电调支持且推荐使用 CAN 协议。当 CAN 总线上的一个节点 (站) 发送数据时，它以报文形式广播给网络中所有节点。对每个节点来说，无论数据是否是发给自己的，都对其进行接收。每组报文开头的 11 位字符为标识符，定义了报文的优先级，这种报文格式称为面向内容的编址方案。在同一系统中标识符是唯一的，不可能有两个站发送具有相同标识符的报文。当几个站同时竞争总线读取时常用这种方法。

### 5.10.2　与 STC15 单片机通信逻辑

由于同时要与多路红外对管集合板进行通信，我们在 STC15 单片机和 STM32 单片机之间使用 I2C 通信。STC15 单片机通过 ADC 读取红外对管的电平，判断哪些红外对管探测到黑线，并将探测到黑线的红外对管编号转化为二进制数据传给主单片机。

### 5.10.3　电调通信逻辑

摩擦轮电调使用 CAN 进行通信。CAN 总线由 CAN_H 和 CAN_L 两根线构成，各个设备一起挂载在总线上。

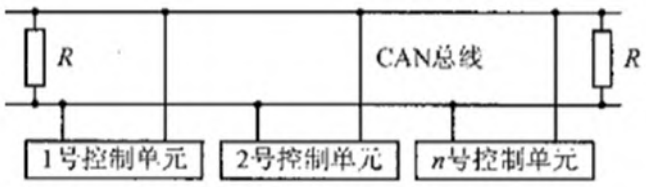



**图 5.8　CAN 总线示意图**

一个完整的数据帧由下图中的各个部分组成。

| 总线空闲 | 帧起始 | 仲裁场 | 控制场 | 数据场 | CRC 场 | 应答场 | 帧结尾 | 帧间隔 |
|---|---|---|---|---|---|---|---|---|

**图 5.9　数据帧示意图**

收发信号的数据结构如下。

标识符：0x200　　帧格式：DATA
帧类型：标准帧　　DLC：8 字节

| 数据域 | 内容 | 电调 ID |
|---|---|---|
| DATA[0] | 控制电流值高 8 位 | 1 |
| DATA[1] | 控制电流值低 8 位 | |
| DATA[2] | 控制电流值高 8 位 | 2 |
| DATA[3] | 控制电流值低 8 位 | |
| DATA[4] | 控制电流值高 8 位 | 3 |
| DATA[5] | 控制电流值低 8 位 | |
| DATA[6] | 控制电流值高 8 位 | 4 |
| DATA[7] | 控制电流值低 8 位 | |

**图 5.10　接收信号**

标识符：0x200 + 电调 ID
（如：ID 为 1，该标识符为 0x201）
帧类型：标准帧
帧格式：DATA
DLC：8 字节

| 数据域 | 内容 |
|---|---|
| DATA[0] | 转子机械角度高 8 位 |
| DATA[1] | 转子机械角度低 8 位 |
| DATA[2] | 转子转速高 8 位 |
| DATA[3] | 转子转速低 8 位 |
| DATA[4] | 实际转矩电流高 8 位 |
| DATA[5] | 实际转矩电流低 8 位 |
| DATA[6] | 电机温度 |
| DATA[7] | Null |

**图 5.11　发送信号**

为了达到控制电机的输出电流，从而控制电机转速的目的，需要发送数据给 1 号到 4 号电调。发送数据时，应当按照表中的内容，将发送的 CAN 数据帧的 ID 设置妥当，数据域中的 8Byte 数据按照电调 1 到 4 的高八位和第八位的顺序装填，帧格式和 DLC 也按照表中内容进行设置，最后进行数据的发送。

### 5.10.4　上位机下位机通信逻辑

上位机与下位机通信协议为 USART/UART 串口通信。在通信中起关键作用的是通信频率和通信数据结构。

小车到达取球区域之后，使用串口通信向树莓派发送指令，树莓派打开摄像头进行画面捕捉。在每一个中心节点停止后，单片机检测到位于节点处时，向树莓派发送捕捉画面的指令，树莓派操作摄像头拍照并识别是否有篮球，并通过 0 或 1 来快速返回结果给单片机。我们有两种方法，一种是按照我们计划好的顺序依次识别单侧的球体；另一种方法是利用摄像头较广的视野范围一并识别后面几组球体。为避免逻辑混乱，这几组数据会储存到树莓派中，接下来几次单片机仍然会向树莓派发送指令，树莓派再一次进行识别，综合多次识别结果增强可信度后，返回储存的结果。

条形码的识别同样是由单片机发出指令开始，到树莓派返回结果结束，过程相同。

# 第 6 章　预算及进度安排

## 6.1　预算

**表 6.1　预算**

| 类型 | 名称 | 数量 | 单价 | 总价 |
|---|---|---|---|---|
| 机械 | 机械爪套件 | 2 | 120 | 240 |
| | 麦克拉姆轮套装 | 1 | 628 | 628 |
| | 机器人底板 | 1 | 150 | 150 |
| | 30mm 摩擦轮 | 1 | 90 | 90 |
| | 碳纤维板 | 2 | 10 | 20 |
| | 机器人底部板 | 1 | 150 | 150 |
| | 弹性轴承座缓冲垫 | 4 | 4 | 16 |
| | 螺丝刀套件 | 1 | 50 | 50 |
| | 扳手套件 | 1 | 50 | 50 |
| | 螺丝，螺母，垫片，铜柱，T 型螺母等紧固件 | | 120 | 120 |
| | 轴承 623ZZ,624ZZ | 2 | 16 | 32 |
| | 金属支架 | 2 | 5 | 10 |
| 电路 | 摄像头 | 2 | 54 | 108 |
| | 逻辑分析仪 | 1 | 35 | 35 |
| | 电源 | 1 | 160 | 160 |
| | 电流检测模块 | 1 | 65 | 65 |
| | 激光测距模块 | 1 | 45.29 | 45.29 |
| | 降压模块 | 1 | 29.94 | 29.94 |
| | 电源模块 | 1 | 4.36 | 4.36 |
| | 单片机 | 1 | 178.05 | 178.05 |
| | 麦克纳姆轮电机 | 4 | 138 | 552 |
| | 姿态感知模块 | 1 | 51.4 | 5.4 |
| | 红外对管 | 1 | 3.048 | 0.8 |
| | STC15 单片机 | 12 | 6.12 | 73.44 |
| | 推杆 | 1 | 218 | 218 |
| | 机械臂舵机 | 6 | 72 | 432 |
| | 舵机驱动 | 4 | 30 | 120 |
| | 电机驱动 | 4 | 67 | 268 |
| | 电调和电机 | 1 | 476 | 476 |
| 合计 | | | | 4676.286 |

## 6.2　进度安排

**表 6.2　进度安排**

| 时间 | 进度 |
| --- | --- |
| 6 月 27 日——7 月 30 日 | 积极参加机器人培训，发射结构和地盘结构成型，测试发射结构可行性以及颜色识别的可行性。 |
| 7 月 31 日——8 月 8 日 | 完成机械结构整体的加工安装，测试。能够完成简单比赛动作：巡线，取球，投篮。 |
| 9 月 6 日——10 月 9 日 | 对比赛流程进行优化，完善程序代码。 |
| 10 月 10 日——10 月 15 日 | 完成全程任务，检查，调试，熟悉场地，准备比赛。 |