Robomasters2017

全国大学生机器人大赛

技术报告

哈爾濱工業大學

学　　校：哈尔滨工业大学（深圳）

队伍名称：**南工骁鹰**

1 绪论

RoboMasters2015 全国大学生机器人大赛是由共青团中央学校部和全国学联秘书处联合主办、大疆创新科技有限公司承办的竞赛，是被誉为“战斗力爆表” 的激战类机器人竞赛。其设计内容涵盖了机械、汽车、电子、电气、自动控制、计算机、传感技术、能源等多个学科的的知识领域，其目的宗旨在为大学生提供颠覆传统的创新实践平台，将教学理论与科技实践紧密结合，让高校学生在这个平台体验淋漓尽致的技术对抗，在团队中展现人格魅力、感受合作的力量，探索科技与人之间的默契。

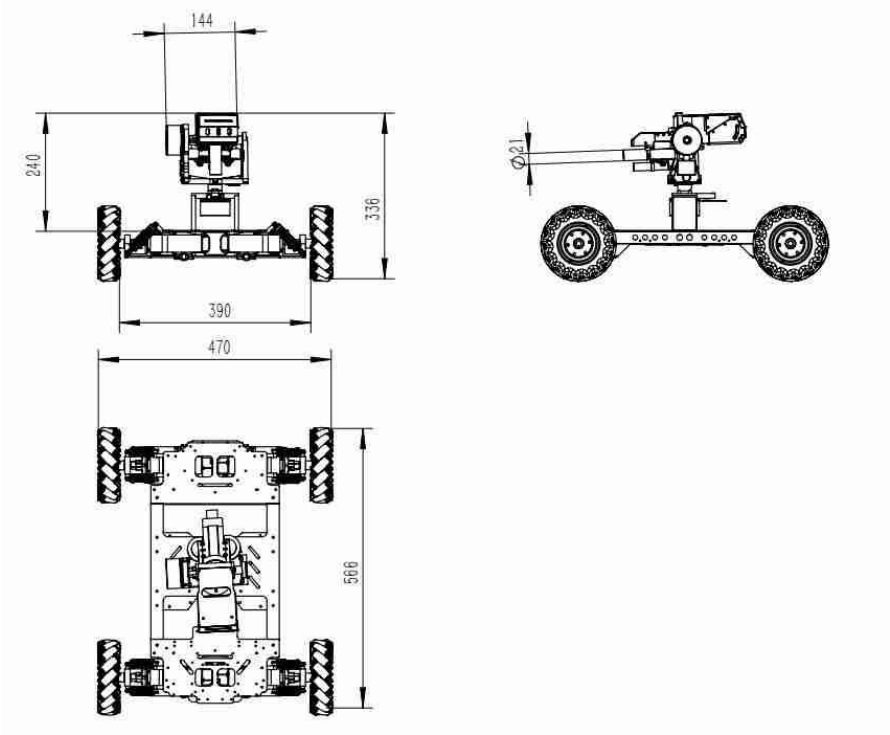
RoboMasters 大赛强调对抗和协调配合，要求战车在高速运动中具备极强的稳定性和抗 击打能力。因此，战队设计战车的整体思路为提高车辆刚度、强度，调整车辆重心，并保证 各机构间的协调。战车基于官方提供的丰富开源资料设计，并结合战队的一些丰富的机构、控制的创新，提升战车的可控性和稳定性。

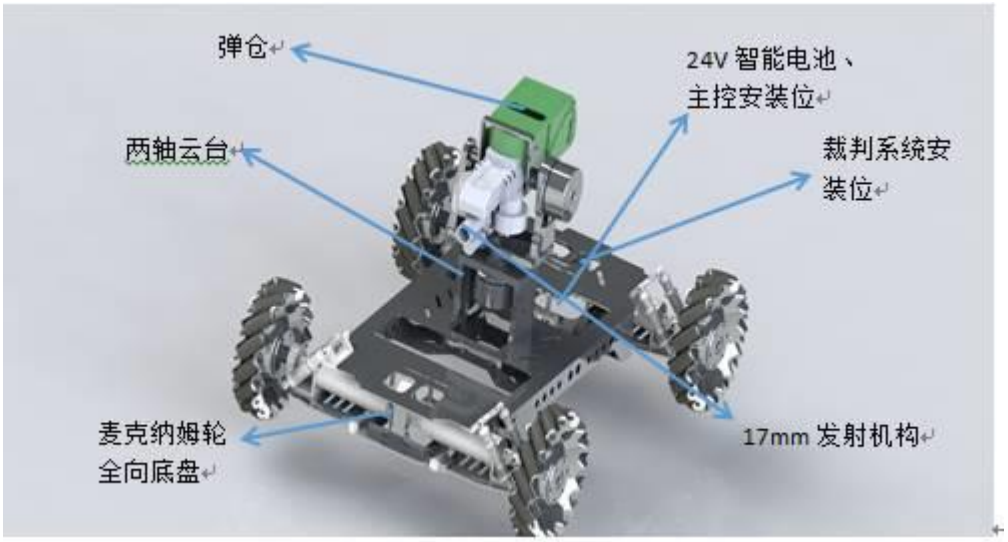
2 机器人制作进度规划表

机器人类型	数量	目前资金消耗	预计资金消耗	方案设计	采购/机加工	组装	调试/改进
步兵机器人	3	3 万	3 万	2016.11	2017.1	2017.1	2017.2
基地机器人	1	1 万	1.5 万	2017.2	2017.3	2017.3	2017.3—2017.4
英雄机器人	1	1 万	2 万	2017.2	2017.3	2017.4	2017.4
工程机器人	1	0.5 万	1 万	2017.2	2017.3	2017.4	2017.4

3 机构设计说明

3.1 步兵车设计



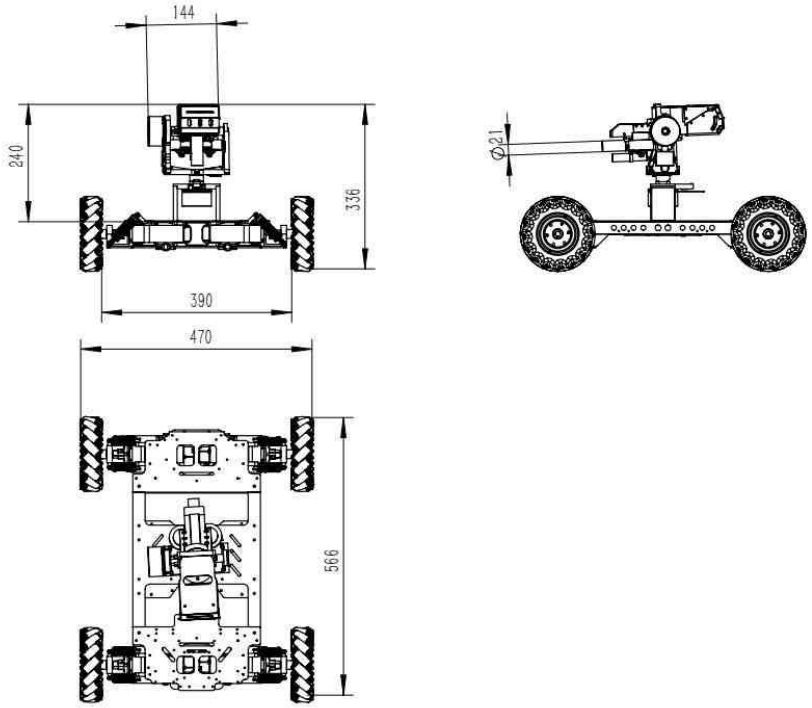


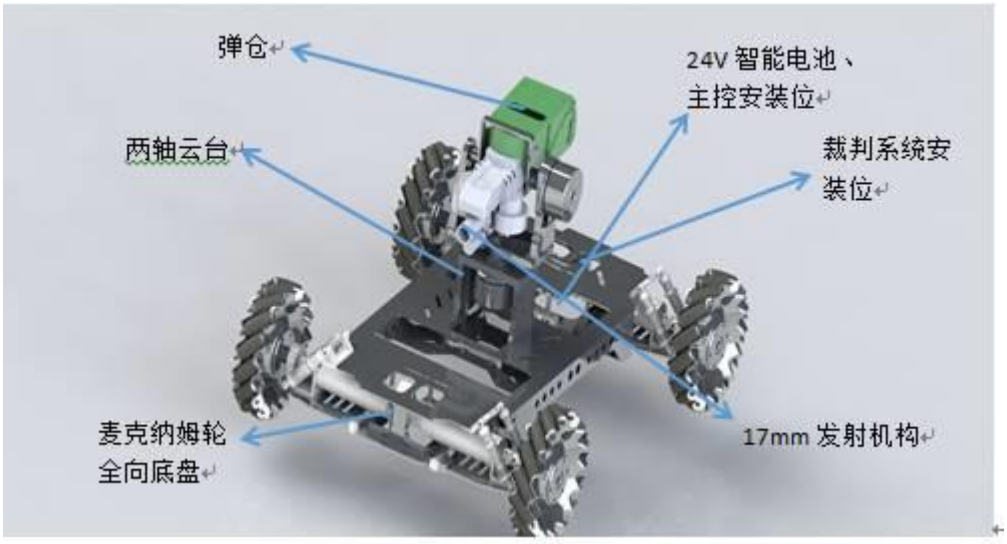
实例说明：底盘采用二代战车的底盘，主要框架采用铝合金，部分受力和连接结构采用碳纤维材料，底盘底部三块受力板采用强度更大的碳纤维板，马达和车轮连接处的活动板也采用碳纤维板，同时连接上部弹簧缓震，使得车辆具备较强的活动性和抗撞击性，底盘的其余部分采用的铝合金材料，保证了车辆底盘的硬度，降低车辆的重心，提高稳定性。麦克纳姆轮四轮独立驱动全向移动方案，云台为两轴，摩擦轮式 17mm 弹丸发射机构，电池放在底盘中后部，主控放在云台下方。采用底盘跟随云台的控制方案；采用以 PID 为基础的电机驱动控制方案，我们使用 stm32F405R 单片机作为主控单元，单片机通过 can1 通信来控制 6623pitch 电机和 6623yawn 电机的电调驱动，进而来驱动云台电机的运行。云台 PID 控制我们采用的是位置 PID 控制算法。摩擦轮电机配套使用的电调采用方波驱动。采用九轴陀螺仪可以反馈机器人的姿态。电机驱动板与其他电路板进行通信采用 CAN 总线通讯方式。在找到目标之后调用 openCV 中的 CamShift 算法对目标进行跟踪。

实物图：

3.2 英雄车设计

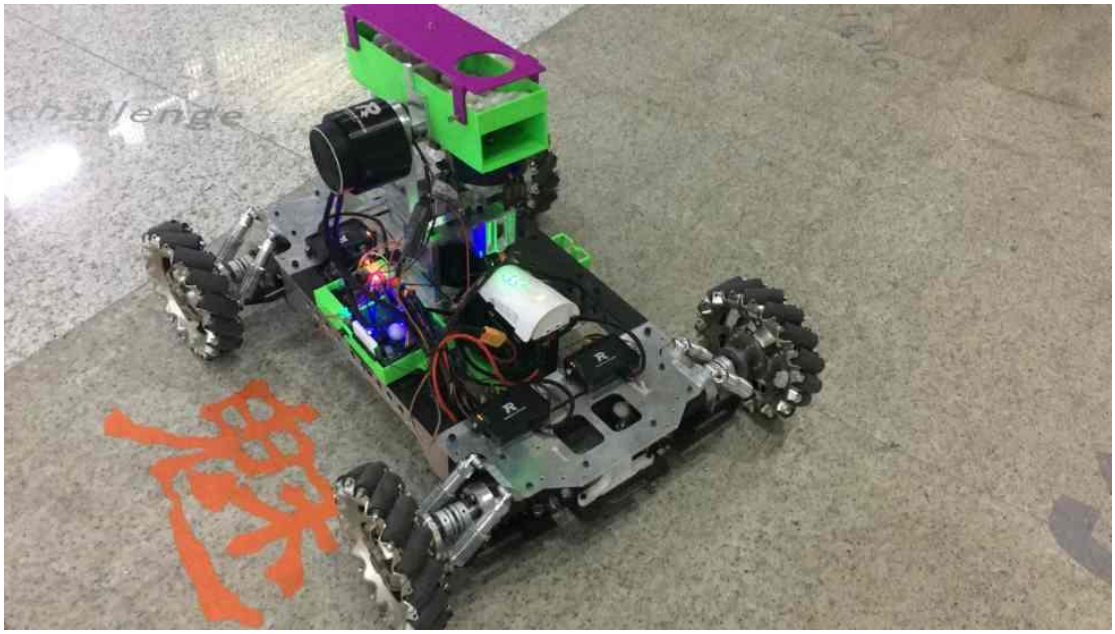
方案一、超级步兵



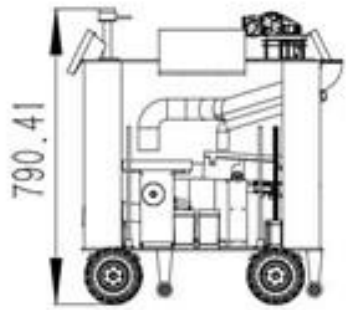


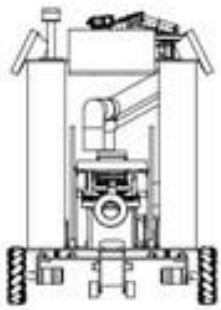
底盘采用二代战车的底盘，主要框架采用铝合金，部分受力和连接结构采用碳纤维材料麦克纳姆轮四轮独立驱动全向移动方案，云台为两轴，摩擦轮式 17mm 弹丸发射机构，电池放在底盘中后部，主控放在云台下方。采用底盘跟随云台的控制方案；采用以 PID 为基础的电机驱动控制方案，我们使用 stm32F405R 单片机作为主控单元，单片机通过 can1 通信来控制 6623pitch 电机和 6623yawn 电机的电调驱动，进而来驱动云台电机的运行。云台 PID 控制我们采用的是位置 PID 控制算法。摩擦轮电机配套使用的电调采用方波驱动。采用九轴陀螺仪可以反馈机器人的姿态。电机驱动板与其他电路板进行通信采用 CAN 总线通讯方式。在找到目标之后调用 openCV 中的 CamShift 算法对目标进行跟踪。

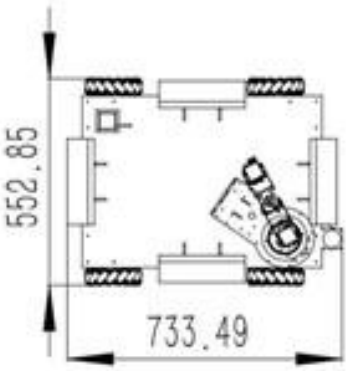
实物图：

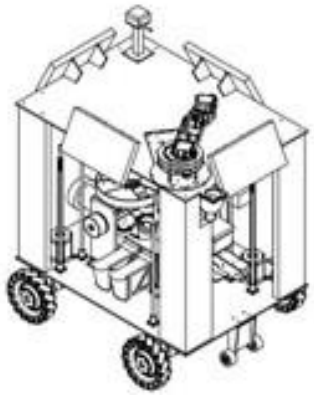


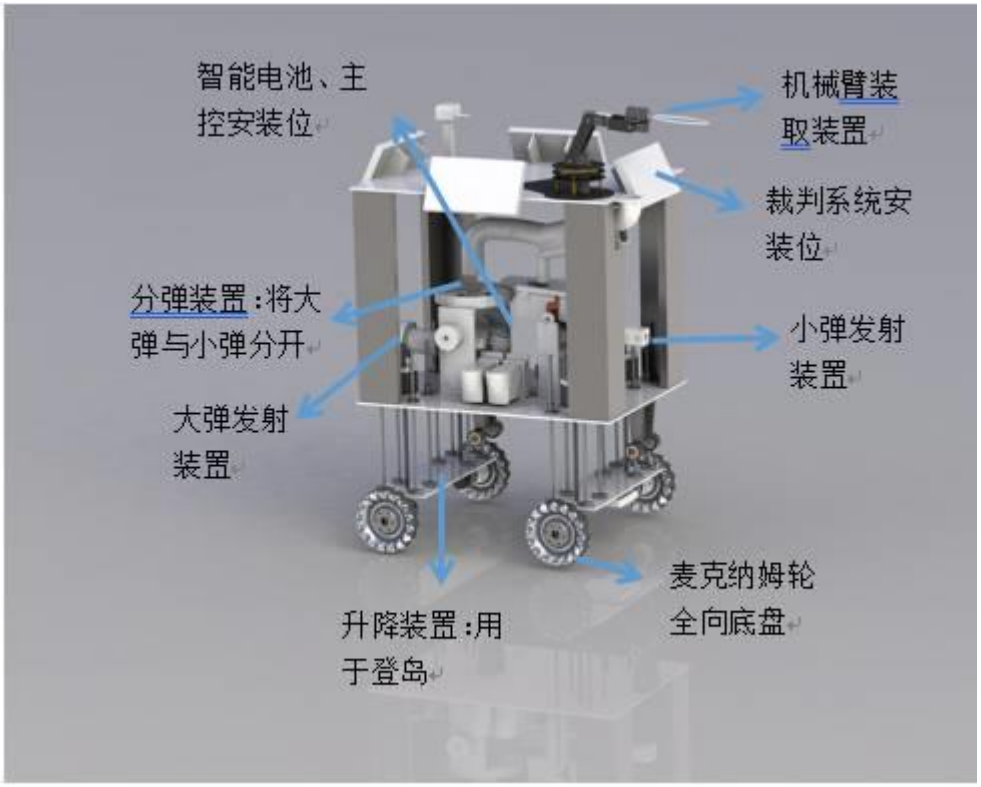
后期方案二、上岛英雄车设想


790.41



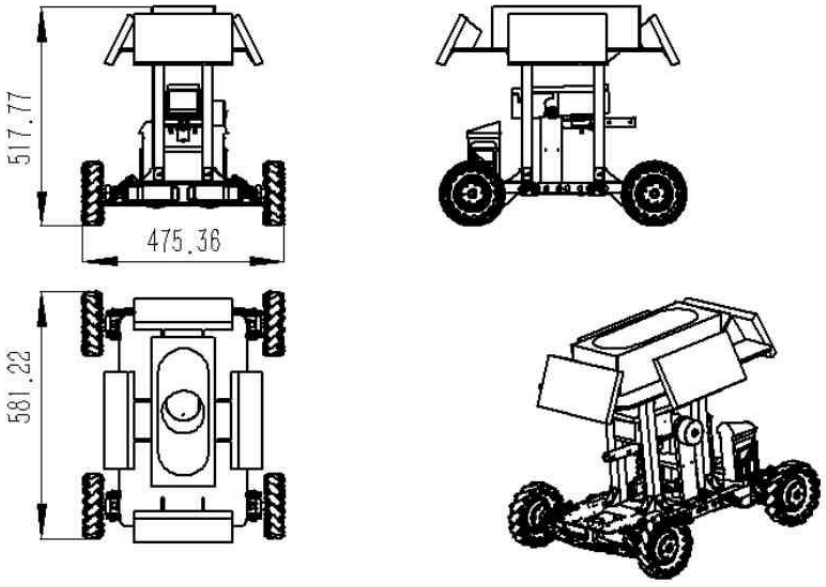

552.85
733.49

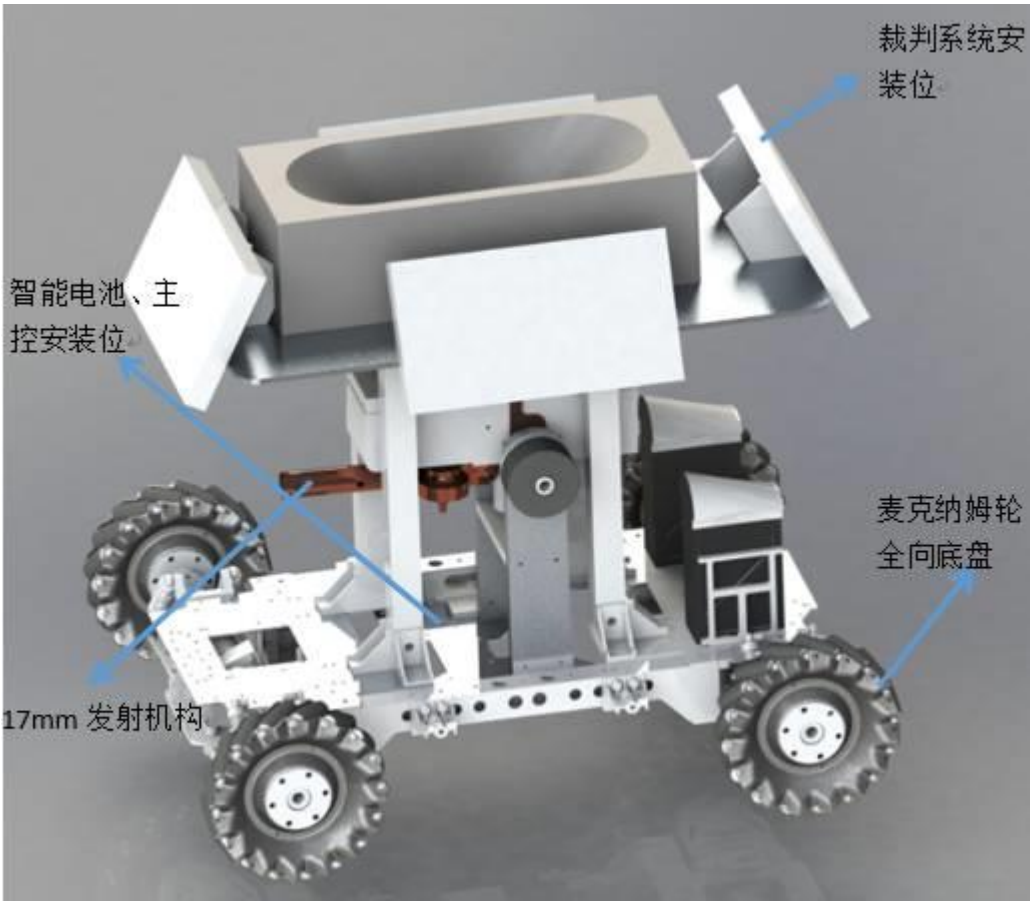



智能电池、主控安装位
机械臂装取装置
裁判系统安装位
分弹装置：将大弹与小弹分开
小弹发射装置
大弹发射装置
麦克纳姆轮全向底盘
升降装置：用于登岛

实例说明：底盘和上板采用的铝合金材料。麦克纳姆轮四轮独立驱动全向移动方案，两云台均为两轴，摩擦轮式弹丸大弹、小弹发射机构，电池放在底盘中部，主控放在云台下方，上板采用的机械臂的抓取方式取高尔夫球大弹。采用底盘跟随云台的控制方案；采用以 PID 为基础的电机驱动控制方案，我们使用 stm32F405R 单片机作为主控单元，单片机通过 can1 通信来控制 6623pitch 电机和 6623yawn 电机的电调驱动，进而来驱动云台电机的运行。云台 PID 控制我们采用的是位置 PID 控制算法。摩擦轮电机配套使用的电调采用方波驱动。采用九轴陀螺仪可以反馈机器人的姿态。电机驱动板与其他电路板进行通信采用 CAN 总线通讯方式。在找到目标之后调用 openCV 中的 CamShift 算法对目标进行跟踪。

3.3 基地机器人设计

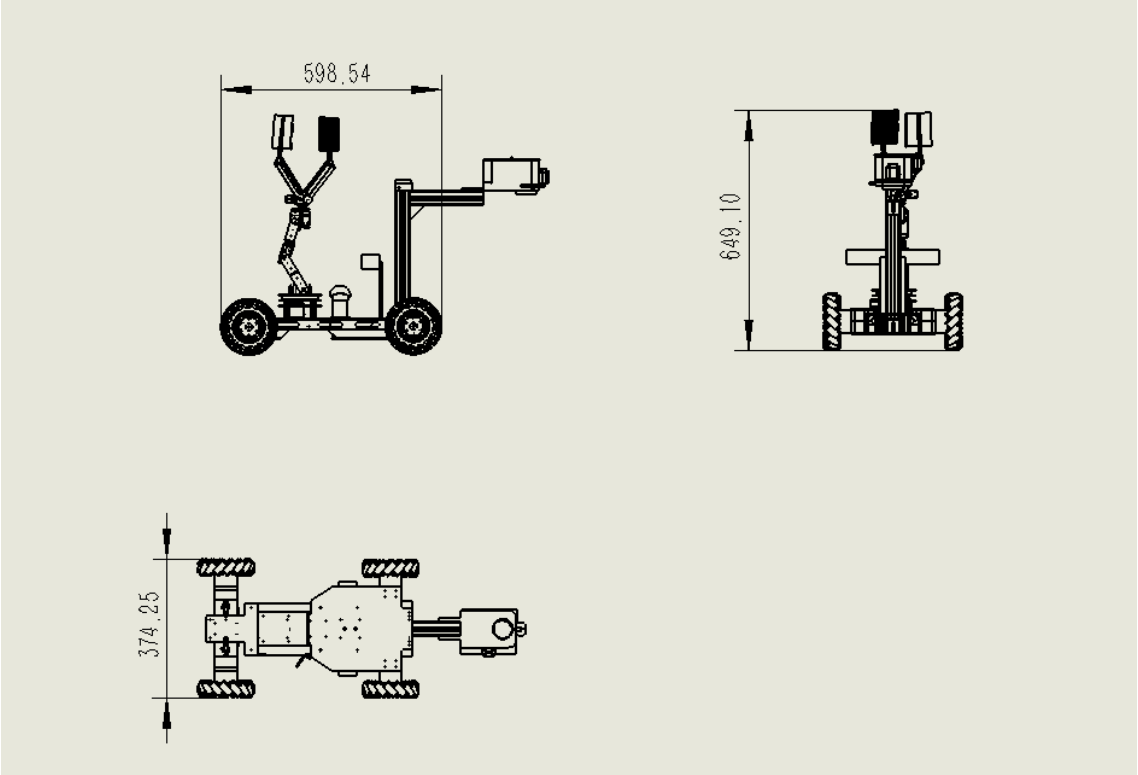


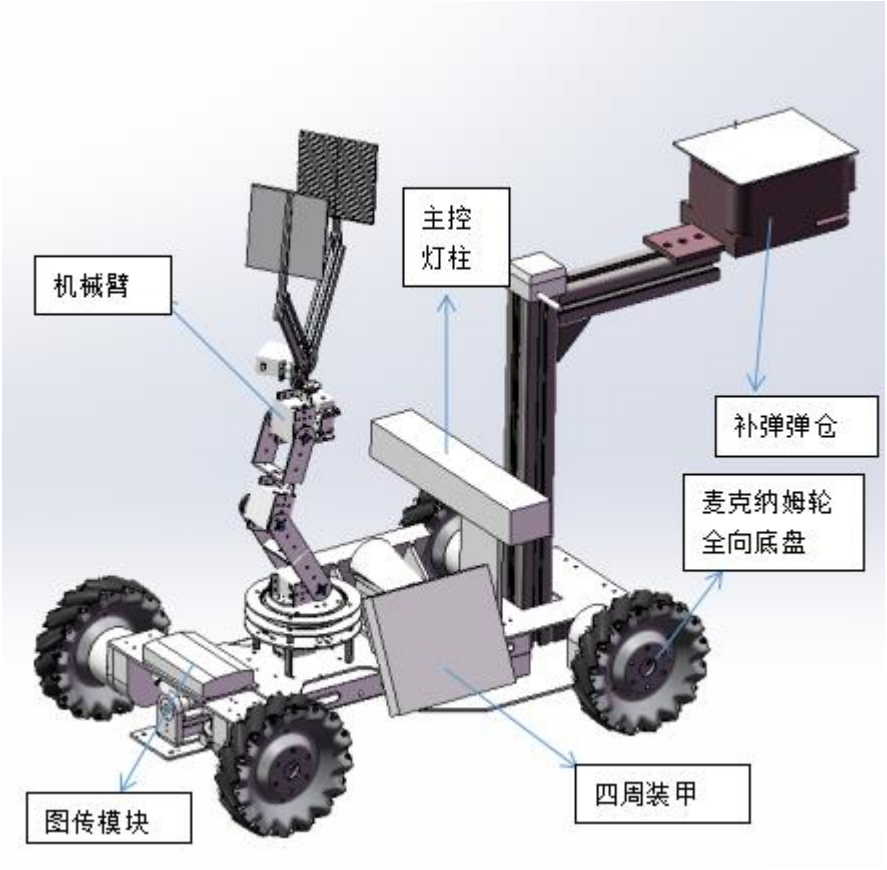


实例说明：底盘和上板采用的铝合金材料。麦克纳姆轮四轮独立驱动全向移动方案，两云台均为两轴，摩擦轮式弹丸大弹、小弹发射机构，电池放在底盘中部，主控放在云台下方，采用以 PID 为基础的电机驱动控制方案，我们使用 stm32F405R 单片机作为主控单元，单片机通过 can1 通信来控制 6623pitch 电机和 6623yawn 电机的电调驱动，进而来驱动云台电机的运行。**采用 HC-SR04 超声波模块来进行距离的测量。**

实物图

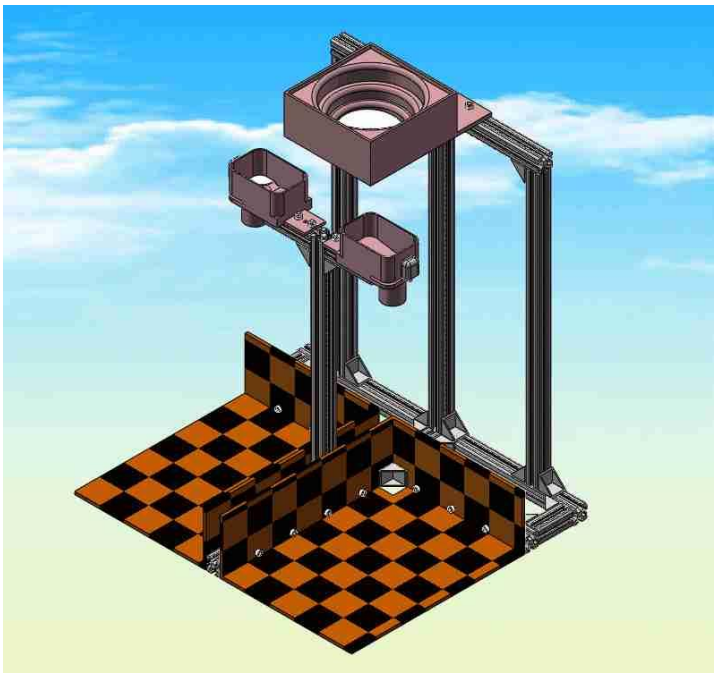
3.4 工程机器人设计





底盘和上板采用的铝合金材料。麦克纳姆轮四轮独立驱动全向移动方案，电池放在底盘中部，主控放在云台下方，弹仓外围加弹，机械臂抓取正方形障碍块，并套在机关立柱上。采用以 PID 为基础的电机驱动控制方案，我们使用 stm32F405R 单片机作为主控单元，单片机通过 can1 通信来控制 6623pitch 电机和 6623yawn 电机的电调驱动，进而来驱动云台电机的运行。

3.5 补给站设计



加弹原理：由上部大漏斗进弹，由分弹电机将弹药随机分入两边下部的小漏斗，进而完成对步兵车或英雄车的加弹。底部的四块挡板可以保证无论战车从什么角度冲入加弹装置，都可由挡板固定位置，完成加弹，进而提高加弹的效率。

补给站的设计是我们队伍机械方面的一大创新，为了满足多辆车能够同时加弹，我们做了一个分弹装置（即上方最大的一个仓），来把子弹分到左右两边的补弹仓里。每一个弹仓都有自己对应的电机，例如，对于直接加弹的两个弹仓，我们用的是普通的舵机，当车子靠近补给站的加弹口时，通过超声波模块的感应，能把电信号发给舵机，操纵舵机，带动 3D 打印圆盘的开合，，实现补弹的功能。

另外，我们的补给站采用厚木板作为底面，这样可以确保补给站的稳固性，当小车冲入补给站时，不会由于补给站摩擦轮过小而被撞离原来位置。

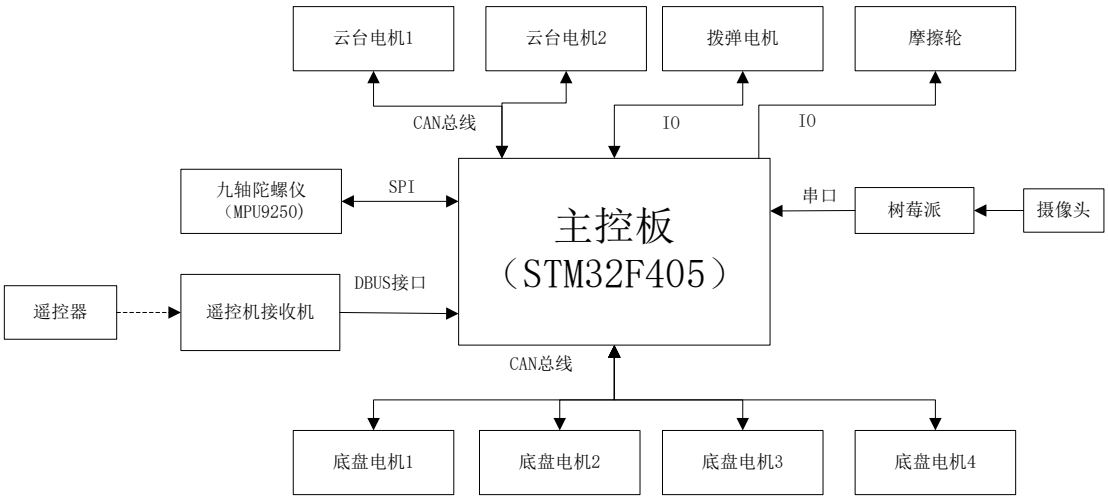
使用材料：三个漏斗使用 3D 打印，支撑架使用铝合金。

实物图：

4 控制设计说明

4.1 系统整体硬件结构

整个步兵机器人的整体硬件结构如下图 1 所示，包含底盘电机四个、云台电机两个、拨弹电机和摩擦轮以及陀螺仪、遥控器接收机和树莓派这些外部器件主要通过 CAN 总线、DBUS 接口、串口、SPI、或者 IO 口与主控板（STM32F405）连接在一起。



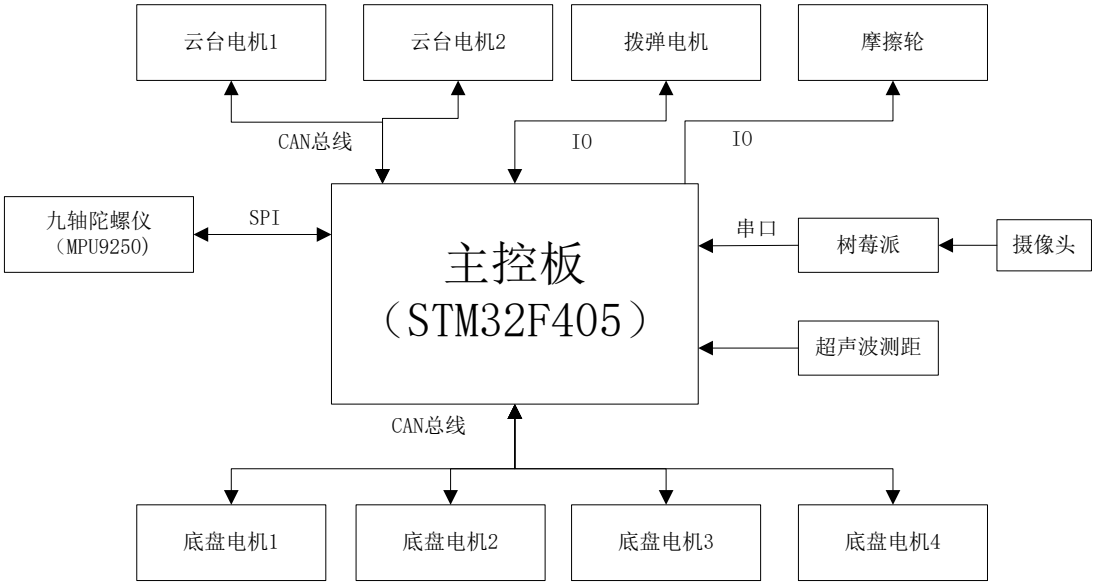
图：步兵机器人整体框图

主控板（STM32F405）作为整个机器人的“大脑”与所有的外部器件连接在一起，操作手通过遥控器向步兵机器人发出命令，主控板（STM32F405）通过遥控器接收机接受遥控器发来的命令，并控制想应的器件做出操作手发出的命令。例如通过底盘电机控制机器人的前进，后退，通过云台电机控制炮台的旋转，通过拨弹电机供弹，摩擦轮打弹等。

主控板（STM32F405）通过定时器工作在编码器模式驱动直流减速电机，主控板（STM32F405）向电机驱动 L298N 发送 PWM 波形驱动电机旋转带动拨弹盘转动，进而控制弹丸漏进炮管的速度不草果比赛规定（1 秒 5 发）。

主控板（STM32F405）可以通过 IO 口输入电机编码器输出的反馈信号，得知电机的转速，再通过调节输出 PWM 波的占空比来控制直流电机的转速恒定。

二、整个基地机器人的整体硬件结构如下图 2 所示，基地机器人基于步兵机器人进行设计，并根据比赛规则，取出来遥控部分，加装了超声测距模块。



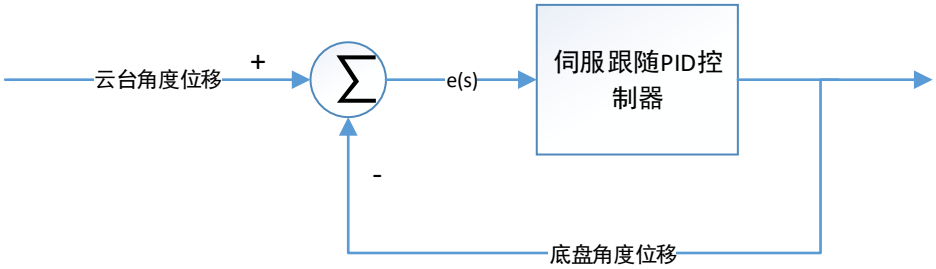
图：基地机器人的整体硬件结构

4.2 底盘控制

4.2.1 底盘跟随控制

底盘跟随的目的是使底盘跟随云台 yaw 轴的旋转而旋转，使底盘与炮口同向，也可以使摄像头随时与炮口同向。

闭环框图如下：

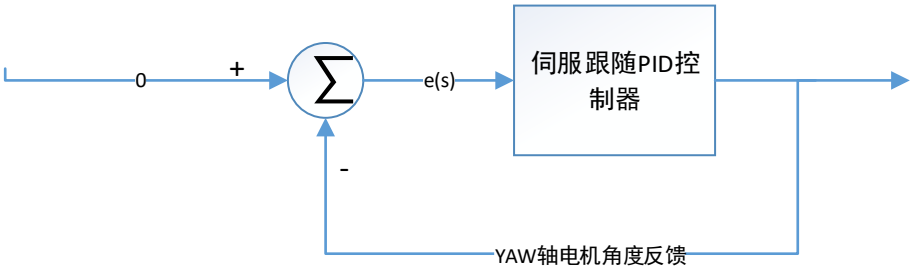


图：底盘跟随闭环框图

其中，云台的转动位移作为底盘旋转控制的输入，底盘的转动位移作为闭环反馈

由于：Yaw 轴电机角度 = 云台角度位移 – 底盘角度位移

因此闭环框图可改为如下形式：



图：底盘跟随闭环框图（修改）

由于底盘的转动会导致云台的绝对位置发生变化，而底盘的跟随只是要改变底盘的绝对位置，因此为达到底盘跟随的目的，云台不是静止的，而是与底盘相反的转动，因此云台的输入是底盘旋转角度的相反数。

4.2.2PID 控制算法

为实现底盘电机的快速响应，准确灵敏快速地达到预先设定的转速，有效地实现 P 轴、Y 轴和 R 轴的灵活移动，采用以 PID 为基础的电机驱动控制方案，能够很好地实现这个目标。

PID 简介

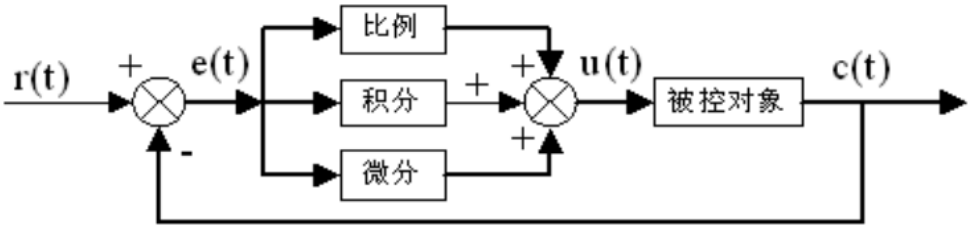
PID 是比例、积分、微分的简称。P 就是比例，就是输入偏差乘以一个系数；I 就是积分，就是对输入偏差进行积分运算；D 就是微分，对输入偏差进行微分运算。

1）比例控制是一种最简单的控制方式。其控制器的输出与输入误差信号成比例关系。当仅有比例控制时系统输出存在稳态误差。增大比例系数使系统反应灵敏，调节速度加快，并且可以减小稳态误差。但是比例系数过大会使超调量增大，振荡次数增加，调节时间加长，动态性能变坏，比例系数太大甚至会使闭环系统不稳定。

2）积分控制中，控制器的输出与输入误差信号的积分成正比关系。对一个自动控制系统，如果在进入稳态后存在稳态误差，则称这个控制系统是有稳态误差的或简称有差系统。为了消除稳态误差，在控制器中必须引入“积分项”。积分项对误差取决于时间的积分，随着时间的增加，积分项会增大。这样，即便误差很小，积分项也会随着时间的增加而加大，它推动控制器的输出增大使稳态误差进一步减小，直到等于零。如果积分作用太强（即积分时间太小），其累积的作用会使系统输出的动态性能变差，超调量增大，甚至使系统不稳定。积分作用太弱（即积分时间太大），则消除稳态误差的速度太慢，积分时间的值应取得适中。

3）微分控制中，控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。自动控制系统在克服误差的调节过程中可能会出现振荡甚至失稳。其原因是由于存在有较大惯性组件（环节）或有滞后组件，具有抑制误差的作用，其变化总是落后于误差的变化。解决的办法是使抑制误差的作用的变化“超前”，即在误差接近零时，抑制误差的作用就应该是零。“微分项”能预测误差变化的趋势，提前使抑制误差的控制作用等于零。适当的微分控制作用可以使超调量减小，增加系统的稳定性。微分时间与微分作用的强弱成正比，微分时间越大，微分作用越强。如果微分时间太大，在误差快速变化时，响应曲线上可能会出现“毛刺”。微分控制的缺点是对干扰噪声敏感，使系统抑制干扰的能力降低。为此可在微分部分增加惯性滤波环节。

PID 控制原理：



图：PID 控制器结构

PID 参数调整方法：

在整定 PID 控制器参数时，可以根据控制器的参数与系统动态性能和稳态性能之间的定性关系，用实验的方法来调节控制器的参数。有经验的调试人员一般可以较快地得到较为满意的调试结果。在调试中最重要的问题是在系统性能不能令人满意时，知道应该调节哪一个参数，该参数应该增大还是减小。

为了减少需要整定的参数，首先可以采用 PI 控制器。为了保证系统的安全，在调试开始时应设置比较保守的参数，例如比例系数不要太大，积分时间不要太小，以避免出现系统不稳定或超调量过大的异常情况。给出一个阶跃给定信号，根据被控量的输出波形可以获得系统性能的信息，例如超调量和调节时间。应根据 PID 参数与系统性能的关系，反复调节 PID 的参数。

如果阶跃响应的超调量太大，经过多次振荡才能稳定或者根本不稳定，应减小比例系数、增大积分时间。如果阶跃响应没有超调量，但是被控量上升过于缓慢，过渡过程时间太长，应按相反的方向调整参数。

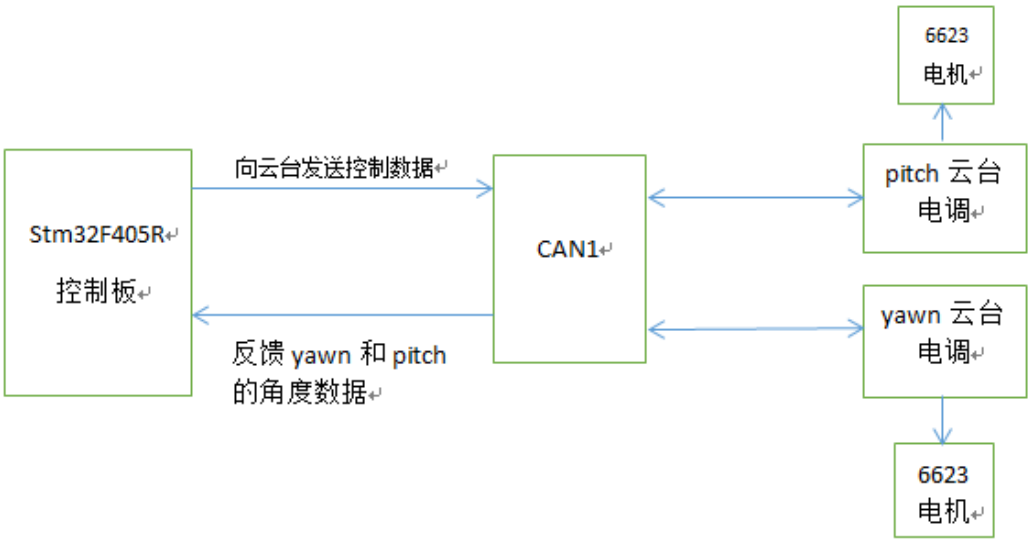
如果消除误差的速度较慢，可以适当减小积分时间，增强积分作用。

反复调节比例系数和积分时间，如果超调量仍然较大，可以加入微分控制，微分时间从 0 逐渐增大，反复调节控制器的比例、积分和微分部分的参数。

总之，PID 参数的调试是一个综合的、各参数互相影响的过程，实际调试过程中的多次尝试是非常重要的，也是必须的。

4.3 云台控制

系统框图



图：云台系统框图

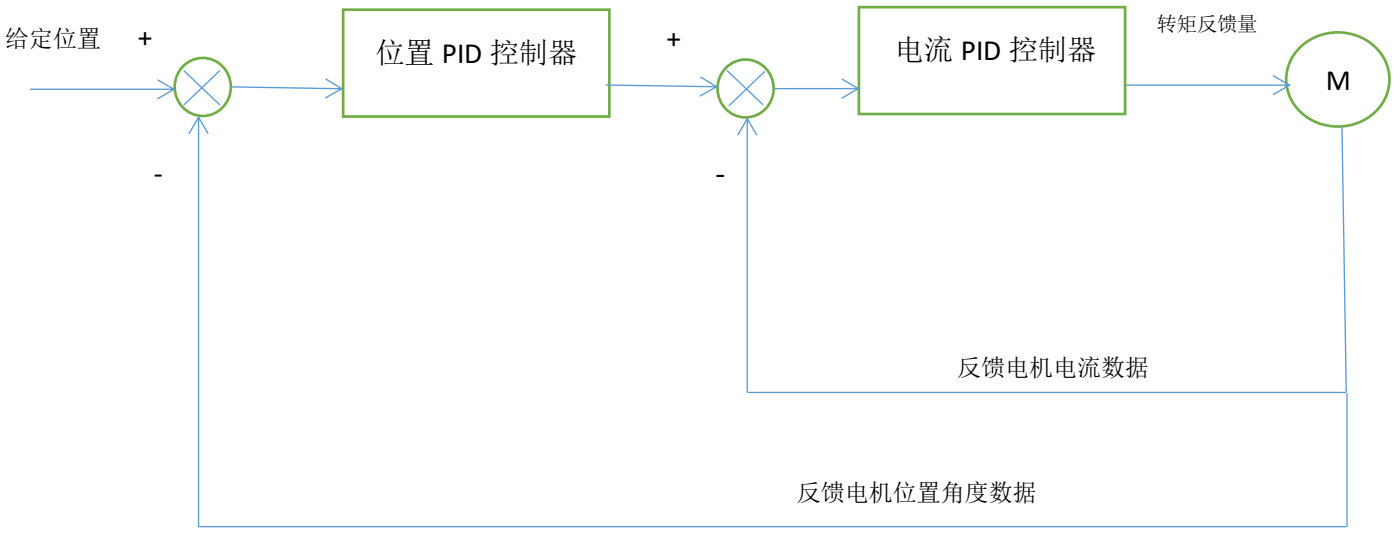
我们使用 stm32F405R 单片机作为主控单元，单片机通过 can1 通信来控制 6623pitch 电机和 6623yawn 电机的电调驱动，进而来驱动云台电机的运行。并且同时，云台 yawn 和 pitch 的电调同样通过 can1 通信，把当前的云台角度数据传送给单片机，来进行对云台位置环 PID 的闭环处理。

由图 1.1，我们可以看出，云台控制板通过 CAN 通信协议与 RM6623 驱动板进行通信，控制信息以一个 CAN 消息帧的形式传输。其中，由云台控制板到 RM6623 驱动板的 CAN 帧内容为三轴驱动电流大小，但此处实际只用到了两轴电流大小的数据。由 RM6623 驱动板到云台控制板的 CAN 帧内容包括三轴实时驱动电流大小以及三轴当前绝对角度值，同理，这里我们只会用到其中两轴（pitch 和 yawn）的相关数据。通过得到的这两个参数（位置角度和转矩电流）的大小，我们可以对云台进行位置环和电流环的双闭环控制。

算法控制

1 云台 PID 控制结构框图

云台位置和电流环双闭环控制系统结构框图

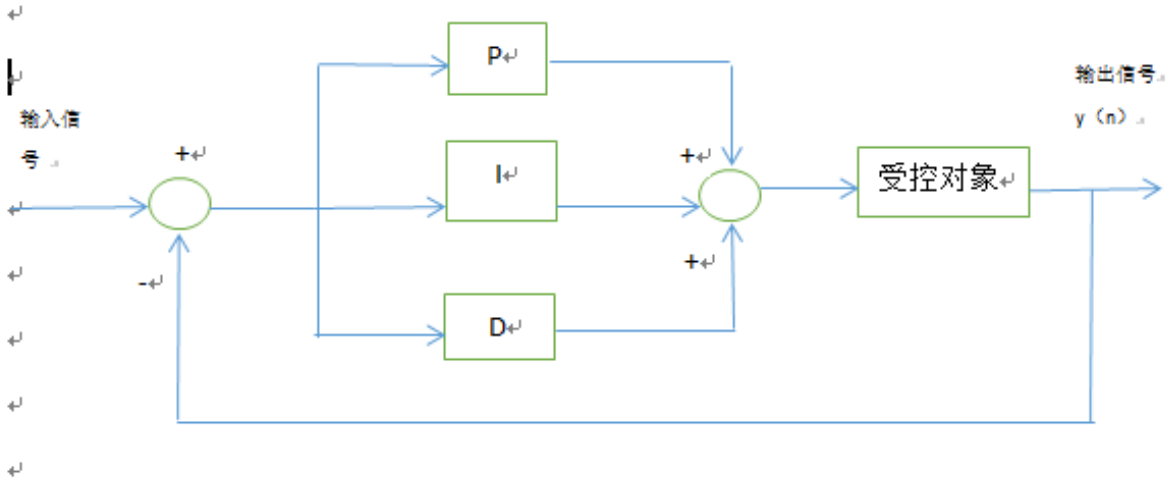


图：云台位置和电流环双闭环 PID 结构框图

由图 1.2 可以看出，我们通过给定云台的给定位置与电机的反馈位置角度数据做差进行 PID 控制，然后用它们的差作为电流环的给定与电机的反馈电流数据做差进行 PID 控制，然后输出数据作用到电机上对电机进行闭环控制。

2 位置 PID 控制算法

云台 PID 控制我们采用的是位置 PID 控制算法。



图：位置 PID 控制算法

PID 调节器是一种线性调节器，控制偏差定义为 e(n) = r(n) – y(n)，并对该控制偏差作比例，积分，微分等运算，并将其各步运算结果作线性运算作为控制量输出，用于控制受控对象。其中 PID 算法根据实现方式分为模拟式与数字式 PID 算法。对于本系统，采用数字式 PID 控制算法。其输出变量的离散表达式为：

$$u(n) = k_p\left(e(n) + \frac{T}{T_I}\sum_{j=0}^{k} e(j) + \frac{T_D}{T}(e(n) - e(n-1))\right)$$

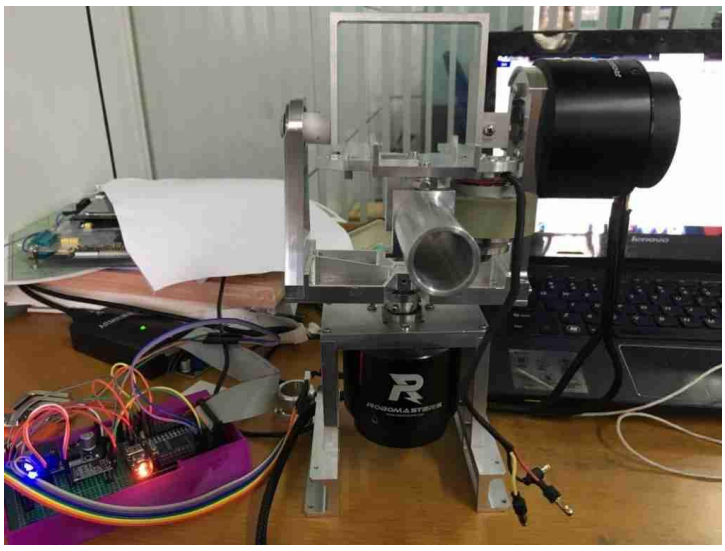
上式可以进一步简化为：

$$u(n) = k_p e(n) + k_i\sum_{j=0}^{k} e(j) + k_d(e(n) - e(n-1))$$

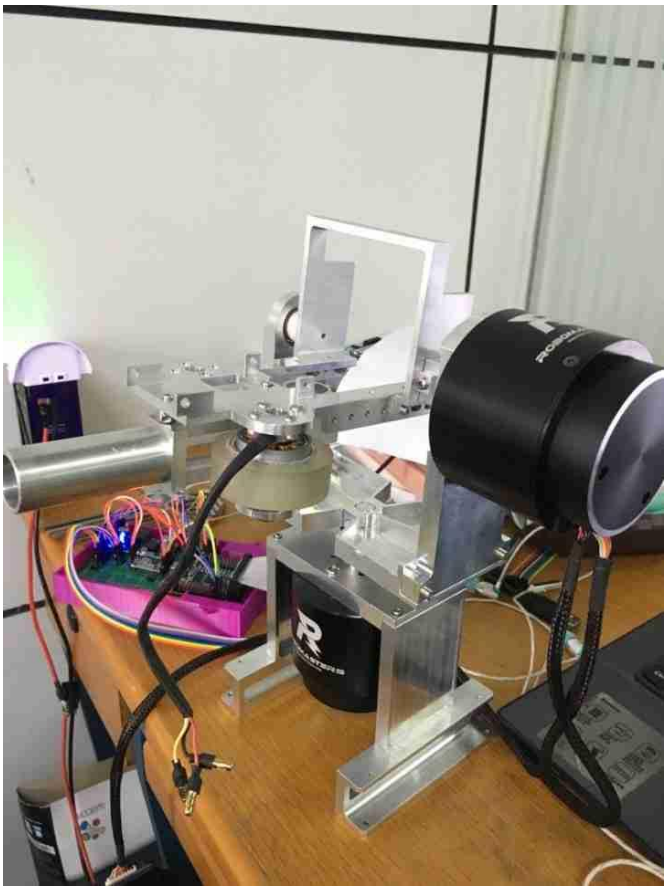
其中，k_p, k_i, k_d 分别称为比例常数，积分常数与微分常数，T, T_I, T_D 分别为系统采样时间，积分区间时间，微分区间时间。实际利用位置式 PID 控制受控系统时，常采用经验值法或试凑法来分别调整 k_p, k_i, k_d 的大小，调整或改善系统控制性能。

PID 运算结果 u(n)直接控制执行机构，在本系统中，该值对应着驱动两轴云台无刷电机的电流值大小，其缺点在于当前采样时刻的输出与过去的各个状态有关，计算时要对各个时刻的 e(n)进行累加，运算量大，且由于控制器的输出 u(n)对应的是直接驱动无刷电机的电流大小，因而若计算出现异常，u(n)的大幅度变化会引起云台运动的极其不稳定。其优点在于适用性广，易于实现且对于位置式控制方式的系统，系统适用性总体较强,控制效果较好。

云台控制问题与解决方法



图：云台实物图（正面）



图：云台实物图（侧面）

（1）双闭环 PID 控制的参数太多很难调试

为了解决这个问题，我们根据实际的调试发现，单单通过位置环单闭环控制就能够很好的实现对云台的控制，而且大大减少了 PID 参数调试的难度。

（2）在云台开启后，发现会出现‘吱吱’的声音

这种声音，明显表明电机工作不正常，存在磨损情况。即，电机不能在一个位置稳定下来，即在平衡位置存在振荡问题。另一方面，源自机械装置固定不紧密，部分连接处螺丝随云台抖动而松动，与金属杆碰撞发出的声音。

为了解决这个问题，我们一方面对机械连接固定部分进行了加固，另一方面对 PID 的参数又进行了大量的调试，最终解决了这个问题。

（3）云台上半部分往一边倾斜，并且云台的炮筒过于靠前

云台上半部分往一边倾斜是由于一侧安装了 pitch 云台电机，一侧什么也没有也没有安装，导致了云台的左右中心偏移，往一边倾斜；另一方面是由于上下两个云台部分连接面积过小，导致了云台不能牢固的接触，使得云台倾斜。

考虑到车身重量的限制，我们不能进行作用配重的策略，为此我们改进云台的连接结构，把上下两个部分的接触面积扩大，并且通过多个螺丝进行固定，大大提高了云台的接触稳定性。另一方面云台炮筒重心前移，我们在后面加上少量的配重来平衡，因为要考虑到炮筒后面弹仓等重量的影响。

4.4 摩擦轮与拨弹电机控制

摩擦轮电机选用的是 RM_2312 电机，与之配套使用的电调是 RM_ 420 LITE 电调，该电调采用方波驱动，功能简化，小巧轻便，内置高效、轻快响应算法，该电调的规格如下：

最大允许电压	17.4V
最大允许电流	20A
最大允许峰值电流*（3 秒）	30A
PWM 输入信号电平	3.3V/5V 兼容
兼容信号频率	30-450Hz
电池	3S-4S LiPo
重量（不含线）	12.5g
重量（含线）	27g

*25 度通风良好情况测得数据

发弹部分由两个 RM_2312 电机和两个 RM_ 420 LITE 电调组成，两个电机的转子上装上两个橡胶圈，工作时，两个电机向相反方向旋转，当子弹进入 两个电机中间时，在橡胶圈的作用下将子弹挤出。由于是 PWM 信号输入，所以控制摩擦轮的程序较为简单，主要工作是后期地不断调试，控制子弹的发出速度不超过规定值。

下图是摩擦轮的实物照片：

图：摩擦轮的实物照片

拨弹电机用来控制给炮台供弹的速度，按照比赛规定不得超过 1 秒 5 发。这里拨弹电机选择日本 Namiki 空心杯减速电机型 号：22CL-3501PG。

图：拨弹电机

参数如下：

堵转扭矩：1.6Nm （16Kg • cm）

连续扭矩：0.5Nm （5Kg • cm）

电　压：12VDC

直　径：22mm

轴　长：19mm（带有 90 度双切口）

轴直径：4mm

长　度：67mm（含编码器、减速箱，但不含输出轴）

堵转电流：1.8A

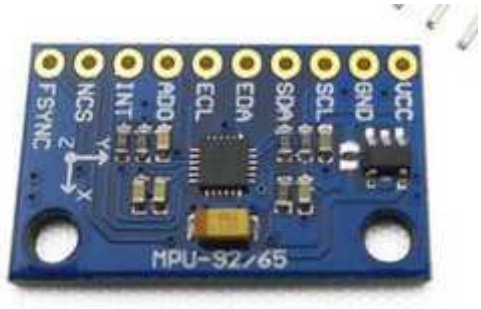
减速比：80：1（金属行星减速）

输出转速：120 转/分钟（输入电压直流 12V）

编码器：2 脉冲每圈

4.5 九轴陀螺仪

陀螺仪选择 MPU9250 ，这个芯片集成有 3 轴加速度，3 轴陀螺仪和 3 轴磁力计以及数字运动处理器（DMP），可以检测出机器人的行驶姿态，把数据通过 SPI 反馈个主控板，来保持机器人姿态稳定。



图：九轴芯片

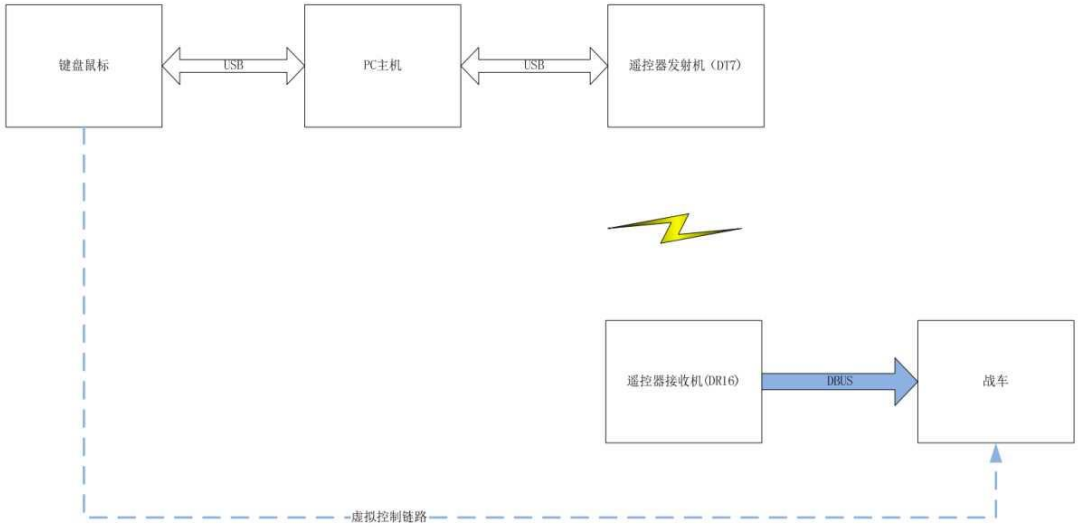
MPU9250 是一个 QFN 封装的复合芯片（MCM），它由 2 部分组成。一组是 3 轴速度还有 3 轴陀螺仪，另一组则是 AKM 公司的 AK8963 3 轴磁力计。所以，MPU9250 是一款 9 轴运动跟踪装置，它在小小的 3X3X1mm 的封装中融合了 3 轴加速度，3 轴陀螺仪以及数字运动处理器（DMP）并且兼容 MPU6515。其完美的 I2C 方案，可直接输出 9 轴的全部数据。一体化的设计，运动性的融合，时钟校准功能，让开发者避开了繁琐复杂的芯片选择和外设成本，保证最佳的性能。本芯片也为兼容其它传感器开放了辅助 I2C 接口，比如连接压力传感器。MPU9250 的具有三个 16 位加速度 AD 输出，三个 16 位陀螺仪 AD 输出，三个 6 位磁力计 AD 输出。精密的慢速和快速运动跟踪，提供给客户全量程的可编程陀螺仪参数选择（ ±250，±500，±1000，and±2000 °/秒（dps） ），可编程的加速度参数选择 ±2g，±4g，±8g，±16g，以及最大磁力计可达到±4800uT。其他业界领先的功能还有可编程的数字滤波器，40-85℃时带高精度的 1%的时钟漂移，嵌入了温度传感器，并且带有可编程中断。该装置提供 I2C 和 SPI 的接口，2.4-3.6V 的供电电压，还有单独的数字 IO 口，支持 1.71V 到 VDD。通信采用 400KHz 的 I2C 和 1MHz 的 SPI，若需要更快的速度，可以用 SPI 在 20MHz 的模式下直接读取传感器和中断寄存器。采用 CMOS-MEMS 的制作平台，让传感器以低成本的高性能集成在一个 3x3x1mm 的芯片内，并且能承受住 10，000g 的震动冲击。

这个九轴芯片可以反馈机器人的姿态，主控板（STM32F405）可以根据这些数据知道机器人的行驶姿态，并根据这些数据控制机器人做出相应的操作保证机器人姿态稳定。

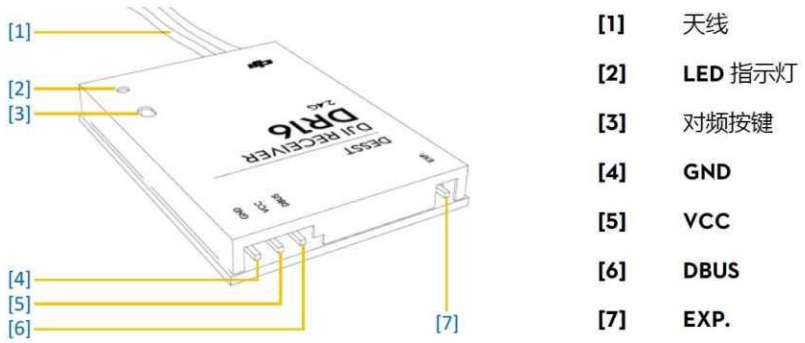
4.6 通信方式

4.6.1 无线通信

遥控器接收机输出信号为标准 DBUS 协议数据，



图：控制链路



图：遥控器接收机

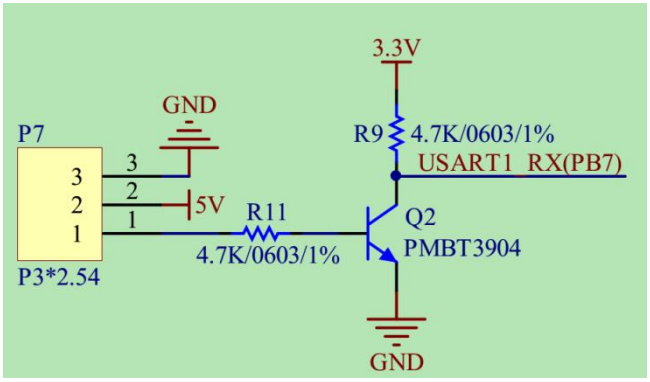
其中 4 是地线，5 是 VCC 用于为接收机供电，电压范围为 4-8.4V，6 是 DBUS 数据输出线。

当接收机和发射机建立连接后，接收机会每隔 7ms 通过 DBUS 发送一帧数据（18 字节），DBUS 的通信参数如下：

表 1 DBUS 通信参数

DBUS 参数	**数值**
波特率	100kpbs
单元数据长度	8
奇偶校验位	偶校验
结束位	1
流控	无

DBUS 信号控制电平符合 TTL，但是和普通 UART 信号是相反的，所以需要在 MCU 端增加三极管取反电路，MCU 才能正常识别出 UART 信号。



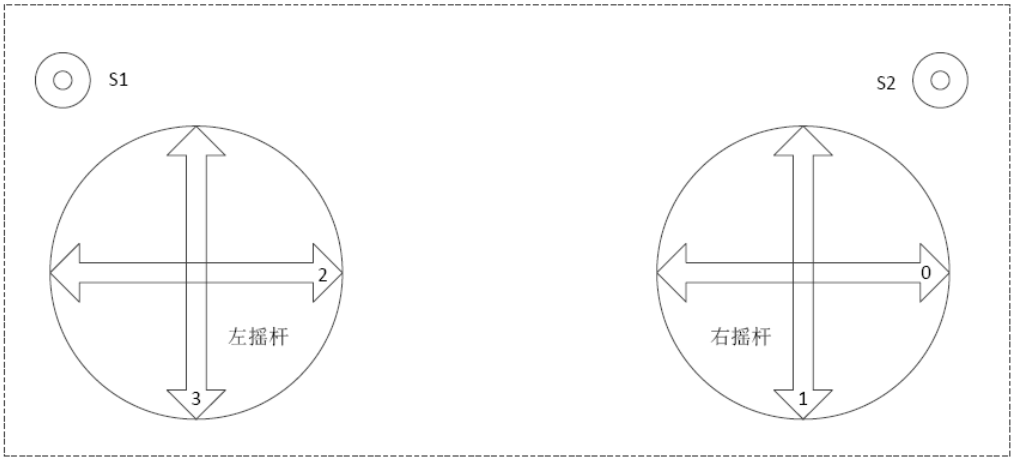
图：三极管取反电路

控制帧的结构如下所示：

表 2 帧结构

域	**通道 0**	**通道 1**	**通道 2**	**通道 3**	**S1**	**S2**
偏移	0	11	22	33	44	46
长度 (bit)	11	11	11	11	2	2
符号位	无	无	无	无	无	无
范围	最大值 1684	最大值 1684	最大值 1684	最大值 1684	最大值 3	最大值 3

	中间值 1024 最小值 364	中间值 1024 最小值 364	中间值 1024 最小值 364	中间值 1024 最小值 364	最小值 1	最小值 1
功能	无符号类型 遥控器通道 0 控制信息	无符号类型 遥控器通道 1 控制信息	无符号类型 遥控器通道 2 控制信息	无符号类型 遥控器通道 3 控制信息	遥控器发射机 S1 开关位置 1：上 2：下 3：中	遥控器发射机 S2 开关位置 1：上 2：下 3：中



遥控器的通道和控制开关如下所示：

图：遥控通道

（此图引用自大疆遥控器说明）

鼠标信息如下表所示：

表 3 鼠标信息

域	鼠标 X 轴	鼠标 Y 轴	鼠标 Z 轴	鼠标左键	鼠标右键
偏移	48	64	80	86	94
长度	16	16	16	8	8
符号位	有	有	有	无	无
范围	最大值 32767 最小值-32767	最大值 32767 最小值-32767	最大值 32767 最小值-32767	最大值 1 最小值 0	最大值 1 最小值 0

	静止值 0	静止值 0	静止值 0		
功能	鼠标在 X 轴的移动速度 负值标识往左移动 正值标识往右移动	鼠标在 Y 轴的移动速度 负值标识往左移动 正值标识往右移动	鼠标在 Z 轴的移动速度 负值标识往左移动 正值标识往右移动	鼠标左键是否按下 0 没按下 1 按下	鼠标右键是否按下 0 没按下 1 按下

键盘信息如下表所示：

表 4 键盘信息

域	**按键**	
偏移	102	
长度	16	
符号位	无	
范围	位值标识	
功能	Bit0-------W 键 Bit1-------S 键 Bit2-------A 键 Bit3-------D 键 Bit4-------Q 键 Bit5-------E 键 Bit6-------Shift 键 Bit7-------Ctrl 键	Bit8-------R 键 Bit9-------F 键 Bit10-------G 键 Bit11-------Z 键 Bit12-------X 键 Bit13-------C 键 Bit14-------V 键 Bit15-------B 键

通过解析数据包，便可以获得来自遥控器或是键盘鼠标的控制信息。

图:无线遥控数据接收测试线路连接图

无线遥控数据接收测试结果如下图所示：

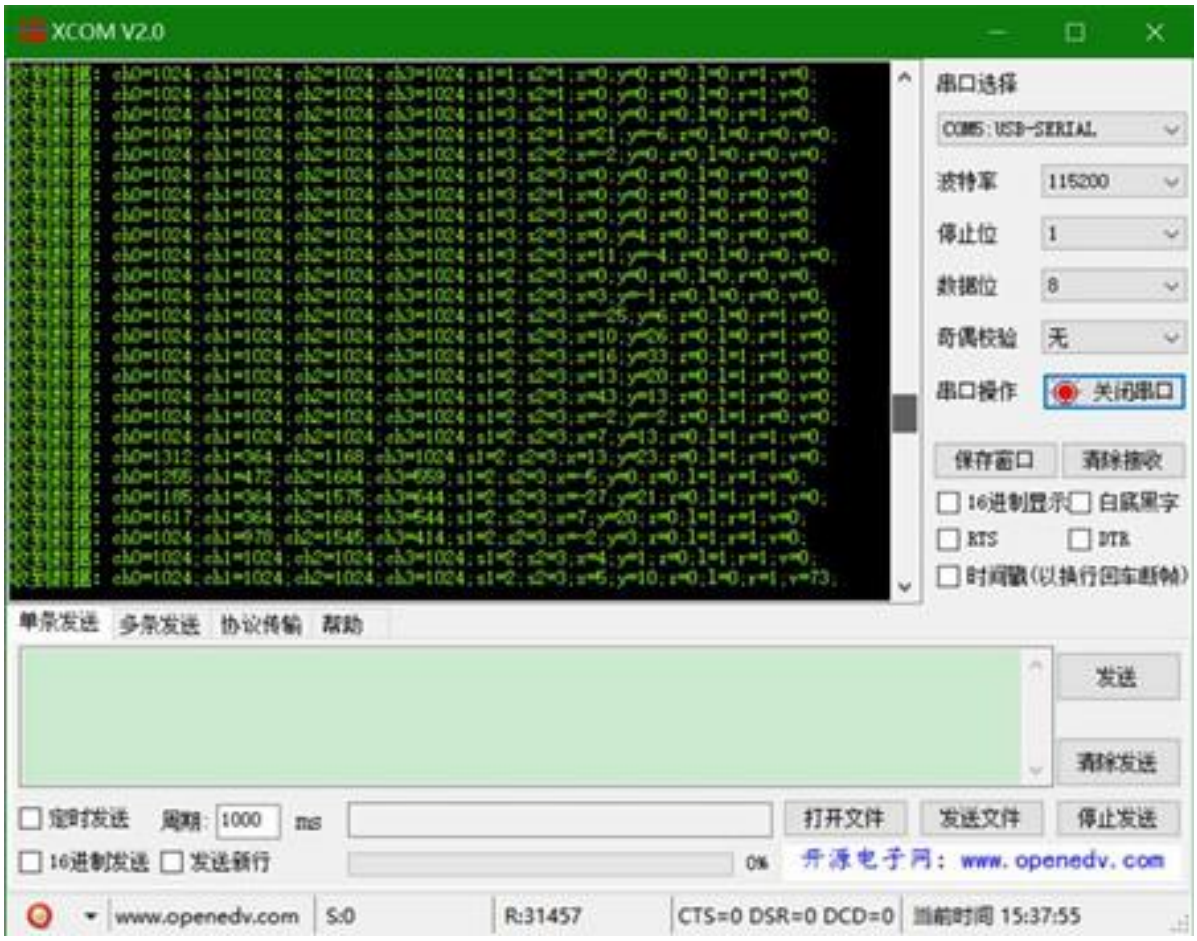


图:无线遥控数据串口输出结果

4.6.2 WiFi 模块

为了方便调试，使用 wifi 模块用于战车状态数据的传输，手机连接上 wifi 后，通过相应的 APP 便可以获取超声波测取的数据信息，小车当前的速度，转向等信息。同时，还可以通过手机向战车发送控制信息，这极大地方便了调试，在测子弹速度时，需要一遍一遍的改变摩擦轮的速度，如果改一次就烧写一次程序，效率极低。有了 wifi 模块就可以通过手机向战车发送数据改变摩擦轮的转速，从而改变子弹的速度。

我们使用的 wifi 模块是 ESP8266，ESP8266 是一个完整且自成体系的 Wi-Fi 网络解决方案，能够搭载软件应用，或通过另一个应用处理器卸载所有 Wi-Fi 网络功能。其具有以下特征：

1、802.11 b/g/n

2、Wi-Fi Direct（P2P）、soft-AP

3、内置 TCP/IP 协议栈

4、内置 TR 开关、balum、LNA、功率放大器和匹配网络

5、内置 PLL、稳压器和电源管理组件

6、802.11b 模式下+19.5dBm 的输出功率

7、内置温度传感器

8、支持天线分集

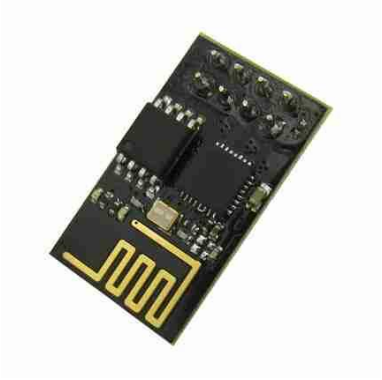
9、断电泄露电流小于 10uA

10、内置低功率 32 位 CPU；可以兼作应用处理器

11、SDIO 2.0、SPI、UART

12、2ms 之内唤醒、连接并传递数据包

13、待机状态消耗功率小于 1.0mW（DTIM3）



图：wifi 模块 ESP8266

该 wifi 模块与 STM32 之间通信采用的是 USART，通过串口向 wifi 模块发送 AT 指令即可控制该模块，编程的重点是对接收数据的提取，因为该模块返回的信息并不仅仅是数据，还有一些其他的附带信息。AT 指令有很多，我们仅仅用到了其中的一部分，具体罗列如下：

指令	说明
AT+RST	重启模块
ATE0	关闭回显
AT+CWMODE	选择 WIFI 应用模式
AT+CIPSERVER	配置 TCP 服务器
AT+CIPSEND	发送数据
AT+CIPMUX	启动多连接
AT+CIOBAUD	设置串口配置

为了能够拥有一个适合我们自己的手机调试软件，我们使用 Android Studio 编写了一个适合自己的 APP 软件，部分界面如下图所示：

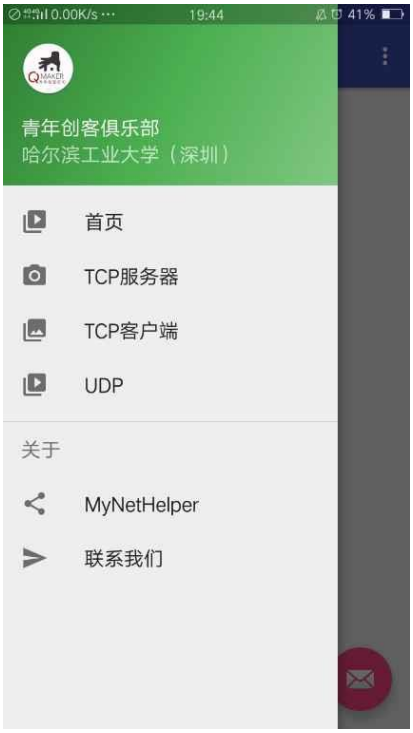
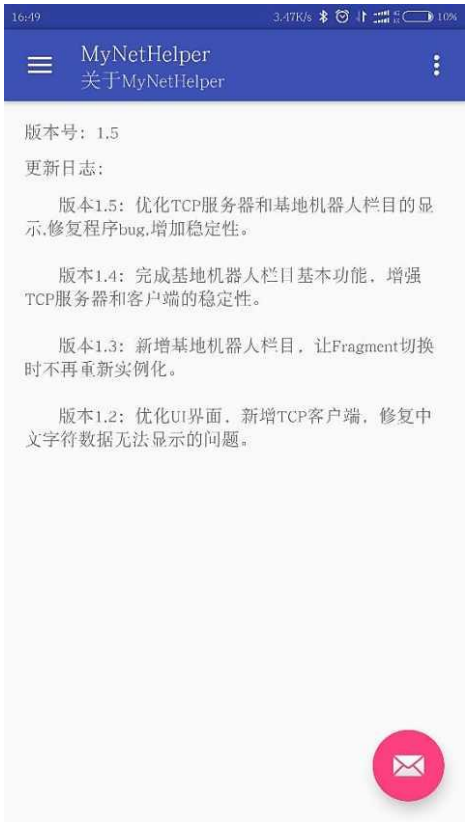
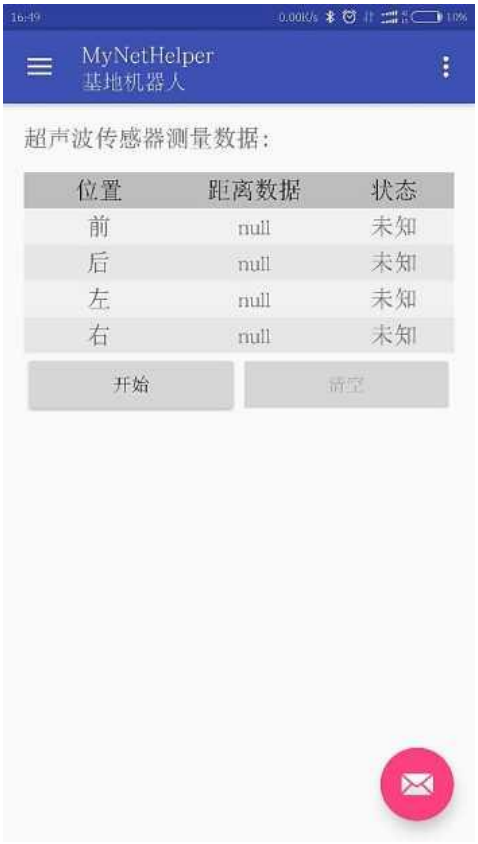


图 1


16:49
3.47K/s
10%
MyNetHelper
关于MyNetHelper
版本号：1.5
更新日志：
版本1.5：优化TCP服务器和基地机器人栏目的显示,修复程序bug,增加稳定性。
版本1.4：完成基地机器人栏目基本功能，增强TCP服务器和客户端的稳定性。
版本1.3：新增基地机器人栏目，让Fragment切换时不再重新实例化。
版本1.2：优化UI界面，新增TCP客户端，修复中文字符数据无法显示的问题。

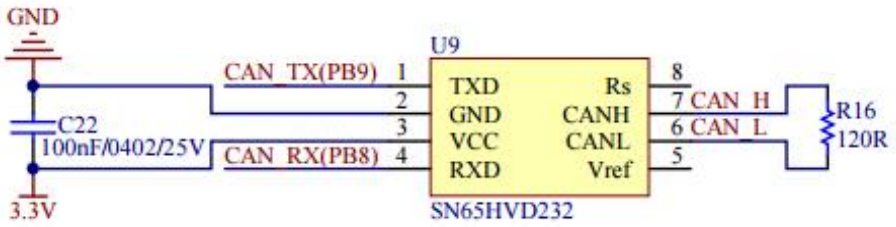
图二


16:49
0.00K/s
10%
MyNetHelper
基地机器人
超声波传感器测量数据：
位置
距离数据
状态
前
null
未知
后
null
未知
左
null
未知
右
null
未知
开始
清空

图三

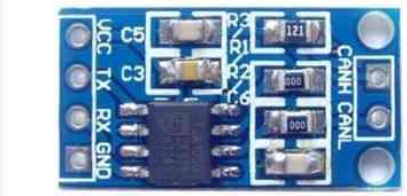
4.6.3 CAN 通信

电机驱动板与其他电路板进行通信可以采用 CAN 总线通讯方式，这是专业的汽车电子通讯方案，通讯误码率为 0%。



图：CAN 通信接口电路

CAN 通信需要一个驱动芯片，这个 CAN 总线驱动器提供了控制器与物理总线之间的接口，是影响系统通信性能的关键因素之一，常选用的是 TI 公司的 SN65HVD232 芯片或者是 TJA1050 芯片就可以很好地完成驱动功能。我们采用的是如图 8 所示的 TJA1050 CAN 控制器接口模块。



图：TJA1050 CAN 控制器接口模块

4.7 超声波测距

对于基地机器人来说，为了防止基地机器人走出边界，需要测出基地机器人与边界的距离，我们采用 HC-SR04 超声波模块来进行距离的测量，其主要参数为：

1、使用电压：DC5V

2、静态电流：小于 2mA

3、电平输出：高 5V

4、电平输出：底 0V

5、感应角度：不大于 15 度

6、探测距离：2cm-450cm

7、高精度：可达 0.3cm

超声波传感器的实物图如下所示，其中，VCC 接 5V 电源、trig 为控制端、echo 为接收端、GND 接 5V 电源的地。

图：超声波传感器

我们在基地机器人的四周放四个超声波传感器，通过检测四周距边界的距离来判断机器人接下来往那边走，为了使布线简化，我们进行了线路的合并和程序的优化，最后通过七根线来控制四个超声波传感器，其中四根电源正合并为一根，四根地合并为一根，四根出发超声波工作的 Trig 线合并为一根，四根 Echo 线不进行合并。

4.8 视觉跟踪算法

一：介绍

RoboMaster 比赛采用红蓝双方对抗模式，通过遥控机器人发射“弹丸”攻击对方机器人以获得胜利。所以需要通过识别红蓝两种颜色来对机器人进行目标识别和追踪，从而提高机器人进攻准确性。

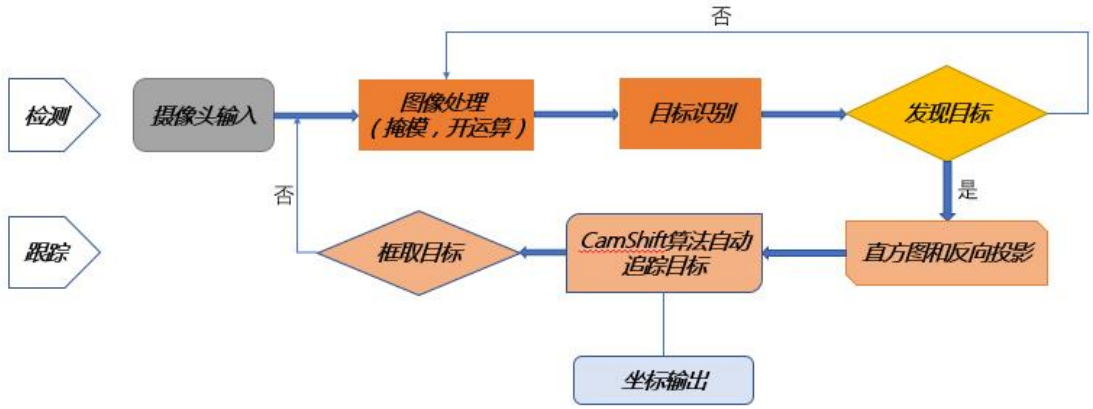
二：实现功能

所要实现的基本功能是在操作命令下达后，在较短的时间实现通过摄像头捕捉敌方机器人图像以达到发现目标并自动控制云台对目标进行瞄准并攻击。因此要用到图像识别功能。在树莓派 Linux 系统下通过使用 Python + openCV 编程，实现该过程。

三：图像识别

1. 理论基础

图像识别要实现两个功能，一是检测，为了在程序首次运行或跟踪失败的时候能够重新找回目标，二是跟踪，在视野内检测到了目标了通过算法实现自动跟踪。之后不再进行检测模块，直到跟踪失败后再进行检测。整个过程的流程控制图如图所示。



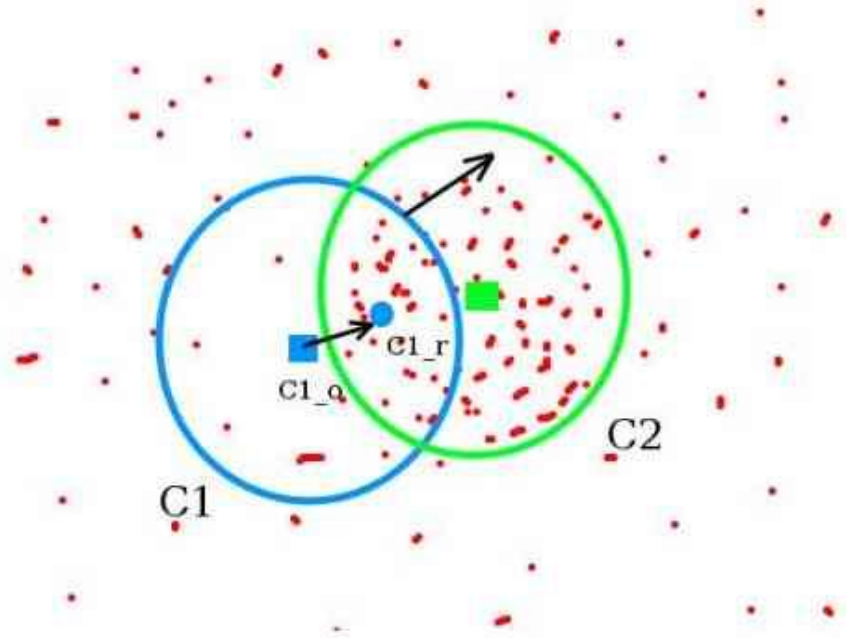
图：流程控制图

官方提供的开源资料中的图像识别方案是采用检测代替跟踪的方法，即完全使用检测的方法，将每一帧的数据作为独立的图像进行识别，它假设每一帧中目标的位置距离上一帧目标的位置最近，除此之外每一帧之间没有任何联系。这种方法的时间复杂度达到了 O（n2），而且在视野中出现较多目标颜色时处理速度较慢。因此我们采用了检测与跟踪相结合的方案，在找到目标之后调用 openCV 中的 CamShift 算法对目标进行跟踪。

Camshift 它是 MeanShift 算法的改进，称为连续自适应的 MeanShift 算法，CamShift 算法的全称是"Continuously Adaptive Mean-SHIFT"，它的基本思想是视频图像的所有帧作 MeanShift 运算，并将上一帧的结果（即 Search Window 的中心和大小）作为下一帧 MeanShift 算法的 Search Window 的初始值，如此迭代下去。Camshift 算法是一个半自动的颜色识别算法，它要求用户提供初始目标，目标丢失之后也不能再次找回。

考虑下面一幅图，图中的点的疏密代表概率，初始的时候我们有一个圆 C1，它的圆心位于 C1_o。首先，我们求得圆 C1 内的重心为 C1_r，然后将圆心移动到 C1_r，再重新计算重心。大多数时候圆心和重心之间是存在一定误差的，通过多

次迭代最终我们使圆心与重心重合（或者在给定的误差范围之内），最终得到圆



图：最终得到的圆

如 C2 所示。C2 就是局部最优解。

2. 实现过程

使用 Python 语言编写代码：

<1>通过 cap = cv2.VideoCapture(0)获取摄像头图像。

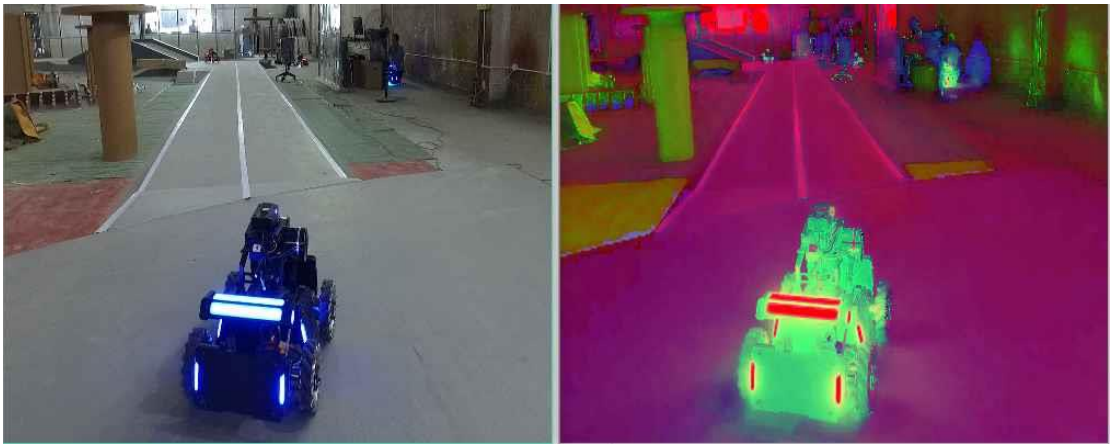
<2>定义初始化函数：

```
def __init__(self, image):
self.image = image
 self.kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2))
 self.kernel2 = cv2.getStructuringElement(cv2.MORPH_RECT, (15, 15))
 self.lower_blue = np.array([115, 100, 100])
 self.upper_blue = np.array([125, 255, 255])
```

cv2.getStructuringElement 是形态学操作中的结构化元素，通过该函数可以构建不同形状的核。

<3>RGB 颜色空间对光照变化较为敏感，为了减少此变化对跟踪效果的影响，首先将 RGB 空间转为 HSV 空间。HSV 即色相、饱和度、明度，色相是色彩的基本属性，就是我们平时常说的颜色，饱和度指色彩的纯度、明度指亮度。通过 cv2.cvtColor 函数将图像转为 HSV 色彩空间。如图所示：

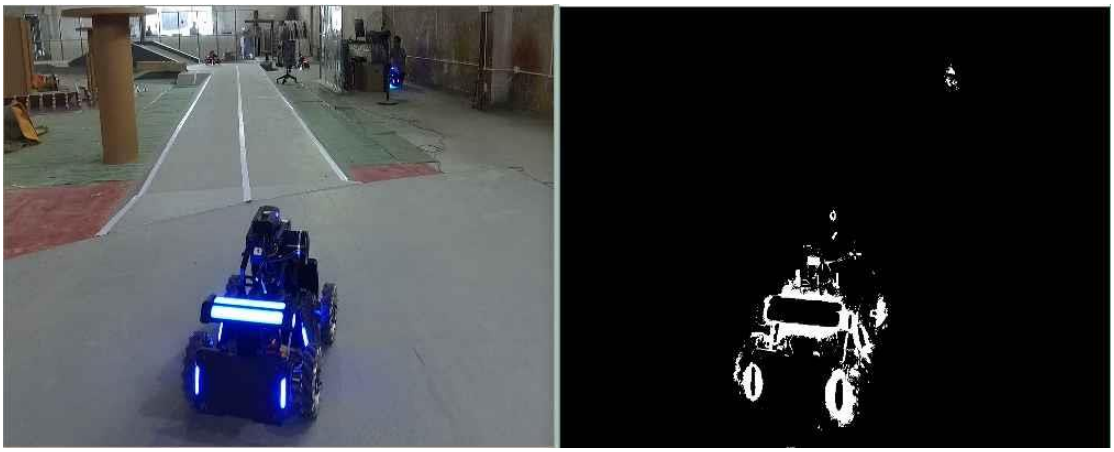
img_hsv = cv2.cvtColor(self.image, cv2.COLOR_BGR2HSV)



图：转化为 HSV 色彩空间

<4>通过色彩空间的转换，我们将一个三维的矩阵降维，仅在 H 分量上通过事先输入的色彩直方图对图像进行分割。根据初始化函数定义的颜色阈值构建掩模。

mask = cv2.inRange(img_hsv, self.lower_blue, self.upper_blue)



图：构建掩模

<5>使用构建好的掩模和 HSV 空间下的图片，构建直方图。颜色直方图反映的是图像的全局特征，它体现了图像的颜色分布。Cv2.calcHist 函数是对图像

进行反向投影，反向投影图是用输入图像的某一位置上像素值（多维或灰度）对应在直方图的一个 bin 上的值来代替该像素值，所以我们可以近似地将反向投影图视为目标的概率分布图。

```
hist = cv2.calcHist([img_hsv], [0], mask, [180], [0, 180])
```

图：直方图

<6>膨胀和腐蚀运算是形态学图像处理的基础。膨胀是在二值图像中“加长”和“变粗”的操作，它使图像边界向外部扩张，可以用来填补物体中的空洞。腐蚀“收缩”和“细化”二值图像中的对象。它可以消除边界点，使图像边界向内部收缩，可以用来消除小而无意义的对象。先腐蚀后膨胀的过程称为开运算。用来消除小物体、在纤细点处分离物体、平滑较大物体的边界的同时并不明显改变其面积。

首先定义开运算：

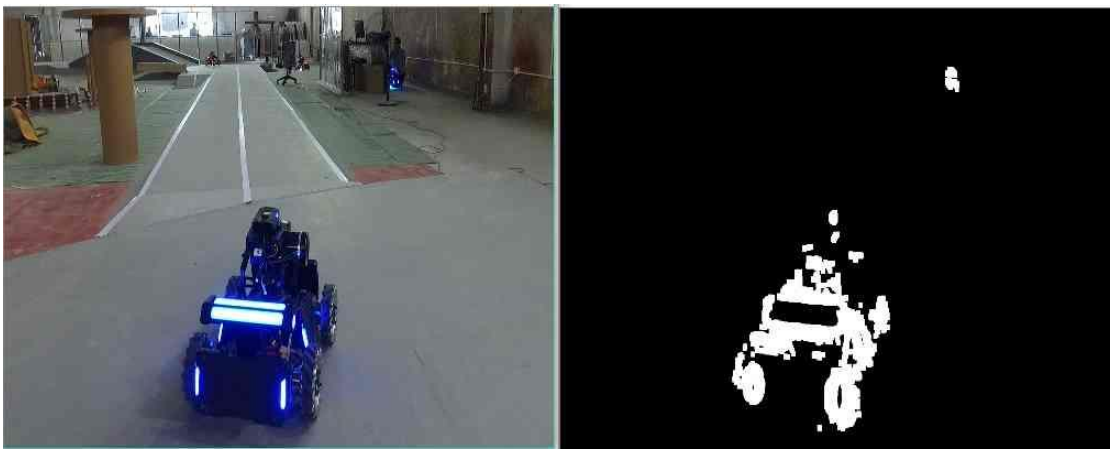
```
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, self.kernel)
```

进行腐蚀处理：

```
kernel3 = np.ones((1, 1), np.uint8)
mask = cv2.erode(mask, kernel3, iterations=1)
```

进行膨胀处理：

```
kernel4 = np.ones((3, 3), np.uint8)
mask = cv2.dilate(mask, kernel4, iterations=4)
```



图：膨胀和腐蚀运算

<7>通过 cv2.findContours 函数来查找敌方机器人蓝色发光位置的轮廓。当正式机器人时，机器人裁判系统上发光的灯分为左灯和右灯，所以需要定两个图框来框定两个灯，以图框的大小排序输出，图框分别定义为 c0 和 c1。

```
(cnts, _) = cv2.findContours(mask.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
if len(cnts) >= 2:
    c0 = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
    c1 = sorted(cnts, key=cv2.contourArea, reverse=True)[1]
```

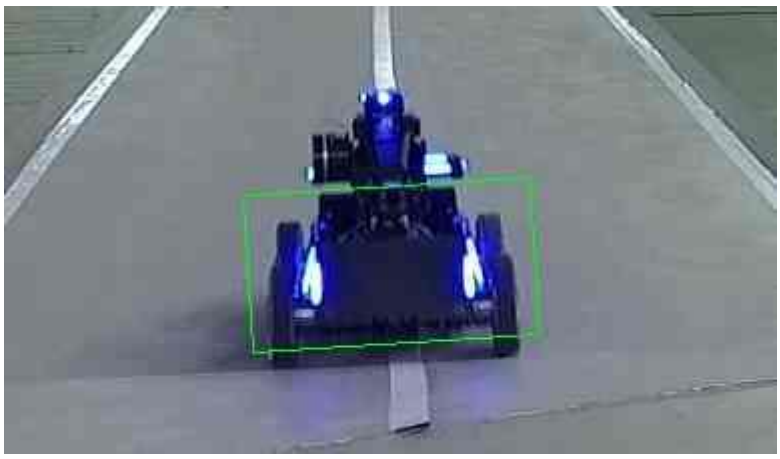
然后将图框的大小设定为包含图像的最小尺寸。

```
rect0 = cv2.minAreaRect(c0)
rect1 = cv2.minAreaRect(c1)
```

因为我方机器人的攻击目标是敌方机器人的裁判系统的面板，所以需要基于两个发光图框构建一整个图框将裁判系统面板框住。

```
if rect0[0][0] <= rect1[0][0]:
    x = int(rect0[0][0]) - int(rect0[1][0] / 2)
    x1 = int(rect1[0][0]) + int(rect1[1][0] / 2)
else:
    x = int(rect1[0][0]) - int(rect1[1][0] / 2)
    x1 = int(rect0[0][0]) + int(rect0[1][0] / 2)
if rect0[0][1] <= rect1[0][1]:
    y = int(rect0[0][1]) - int(rect0[1][1] / 2)
    y1 = int(rect1[0][1]) + int(rect1[1][1] / 2)
else:
    y = int(rect1[0][1]) - int(rect1[1][1] / 2)
    y1 = int(rect0[0][1]) + int(rect0[1][1] / 2)
w = x1 - x
h = y1 - y
tk_window = (x, y, w, h)
```

此时就可以在图像上识别到敌方机器人的发光面板。



图：识别敌方机器人发光面板

在主循环内通过判断图框的宽度是否为 0 和判断图框是否超出边界来构建识别后的图案，且对每一帧的判断实现了图像自动跟随的功能。

```
if is_going:
    mDetectImage = DetectImage(frame)
    track_window, roi_hist = mDetectImage.my_detect()
    if track_window[2] != 0:
        is_ok = True
    else:
        is_ok = False
if is_ok:
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    dst = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)
    ret, track_window = cv2.CamShift(dst, track_window, term_crit)
    x0, y0, w0, h0 = track_window
    if w0 == 0:
        is_ok = False
    else:
        is_ok = True
    if w0 * h0 / (960 * 540) >= 0.05 and (0 * h0 / (960 * 540) <= 0.6):
        is_going = False
    else:
        is_going = True
    pts = cv2.cv.BoxPoints(ret)
    pts = np.int0(pts)
    cv2.polylines(frame, [pts], True, [0, 255, 0], 1)
cv2.imshow("Capture", frame)
if cv2.waitKey(25) & 0xFF == ord('q'):
```

四：系统测试及调试

五：改进方案

<1>改进的 Camshift 算法：边缘方向直方图 ，是对目标物体中边缘点的边缘方向一种分布统计，可以被用来描述表达目标物体。它已经被证明具有快速运算和鲁棒的特点。

<2>kalman 滤波：Kalman 滤波可以对运动目标的位置和角度进行估计，估计值可以用于提高跟踪速度、计算新的边缘方向特征等。

5.创新点

5.1 弹仓补弹

由于开源文件给出小车的弹仓存在容易卡弹的问题，所以，我们队在经过一次又一次的修改后，最终设计出了具有我们队伍特色风格的补弹装置，具体结构如图所示，经过多次试验，我们设计出的补弹装置能很好的满足打弹的要求，不卡弹率在百分之九十以上，后期我们将进一步结合自适应控制来把卡弹问题彻彻底底的解决好：

5.2 避震装置

底盘采用二代战车的底盘，主要框架采用铝合金，部分受力和连接结构采用碳纤维材料，底盘底部三块受力板采用强度更大的碳纤维板，马达和车轮连接处的活动板也采用碳纤维板，同时连接上部弹簧缓震，使得车辆具备较强的活动性和抗撞击性，底盘的其余部分采用的铝合金材料，保证了车辆底盘的硬度，降低车辆的重心，提高稳定性。

5.3 自己设计的 app

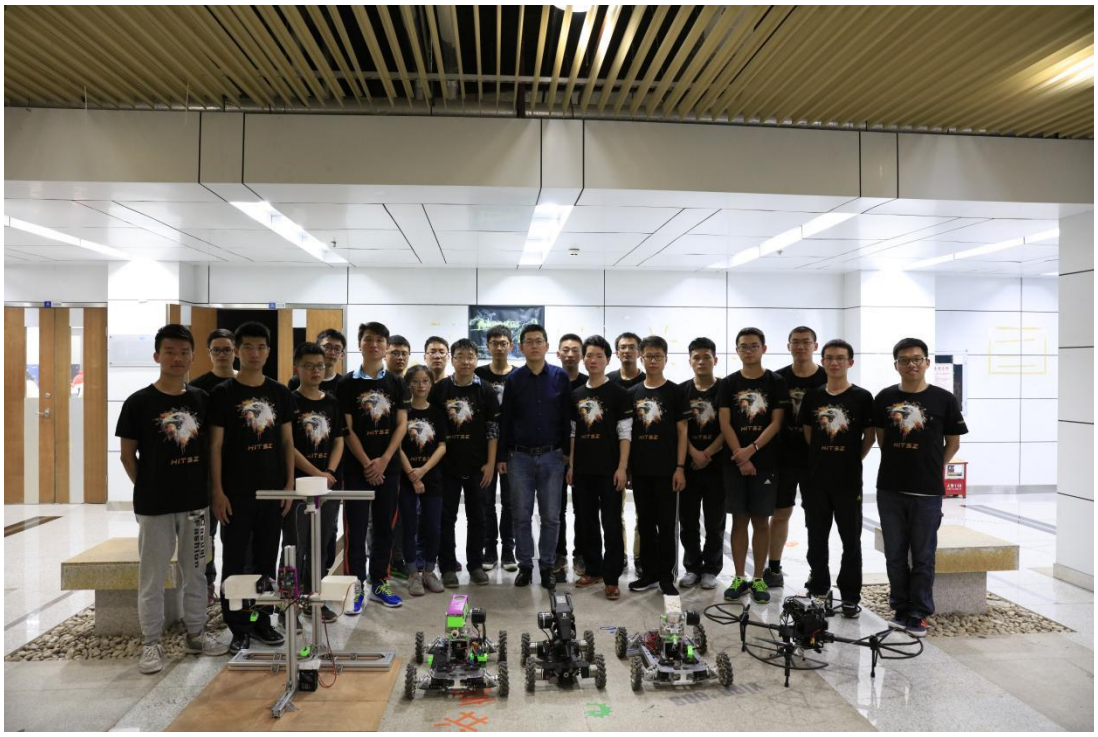
为了能够拥有一个适合我们自己的手机调试软件，我们使用 Android Studio 编写了一个适合自己的 APP 软件，部分界面如下图所示：

5.4 视觉跟踪算法

遇到的问题：捕捉目标以及准确的进行跟踪

解决方法：代码略

6.成员合照



7.视频

网址：http://v.youku.com/v_show/id_XMjY3NzkxNzc0NA==.html?qq-pf-to=pcqq.c2c

密码：0847

附录

源代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import cv2


class DetectImage(object):
    def __init__(self, image):
        self.image = image
        self.kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2))
        self.kernel2 = cv2.getStructuringElement(cv2.MORPH_RECT, (15, 15))
        self.lower_blue = np.array([115, 100, 100])
        self.upper_blue = np.array([125, 255, 255])

    def my_detect(self):
        tk_window = (0, 0, 0, 0)
        img_hsv = cv2.cvtColor(self.image, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(img_hsv, self.lower_blue, self.upper_blue)
hist = cv2.calcHist([img_hsv], [0], mask, [180], [0, 180])
cv2.normalize(hist, hist, 0, 255, cv2.NORM_MINMAX)
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, self.kernel)
p.ones((1, 1), np.uint8)
        mask = cv2.erode(mask, kernel3, iterations=1)

            kernel4 = np.ones((3, 3), np.uint8)
                (cnts, _) = cv2.findContours(mask.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        if len(cnts) >= 2:
            c0 = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
            c1 = sorted(cnts, key=cv2.contourArea, reverse=True)[1]
            rect0 = cv2.minAreaRect(c0)
            rect1 = cv2.minAreaRect(c1)
            if rect0[0][0] <= rect1[0][0]:
                x = int(rect0[0][0]) - int(rect0[1][0] / 2)
                x1 = int(rect1[0][0]) + int(rect1[1][0] / 2)
            else:
```

```
                x = int(rect1[0][0]) - int(rect1[1][0] / 2)
                x1 = int(rect0[0][0]) + int(rect0[1][0] / 2)
            if rect0[0][1] <= rect1[0][1]:
                y = int(rect0[0][1]) - int(rect0[1][1] / 2)
                y1 = int(rect1[0][1]) + int(rect1[1][1] / 2)
            else:
                y = int(rect1[0][1]) - int(rect1[1][1] / 2)
                y1 = int(rect0[0][1]) + int(rect0[1][1] / 2)
            w = x1 - x
            h = y1 - y
            tk_window = (x, y, w, h)

        return tk_window, hist



cap = cv2.VideoCapture(0)
term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)
is_ok = False
is_going = True
while True:
    ret, frame = cap.read()
    if ret:
            if is_going:
            mDetectImage = DetectImage(frame)
            track_window, roi_hist = mDetectImage.my_detect()
            if track_window[2] != 0:
                is_ok = True
            else:
                is_ok = False
        if is_ok:
            hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
            dst = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)
            ret, track_window = cv2.CamShift(dst, track_window, term_crit)
            x0, y0, w0, h0 = track_window
            if w0 == 0:
                is_ok = False
            else:
                is_ok = True
            if w0 * h0 / (960 * 540) >= 0.05 and (0 * h0 / (960 * 540) <= 0.6):
                is_going = False
            else:
                is_going = True
            pts = cv2.cv.BoxPoints(ret)
```

```
                pts = np.int0(pts)
                cv2.polylines(frame, [pts], True, [0, 255, 0], 1)
            cv2.imshow("Capture", frame)
            if cv2.waitKey(25) & 0xFF == ord('q'):
                break
        else:
            break
cv2.destroyAllWindows()
cap.release()
```