

方案设计书

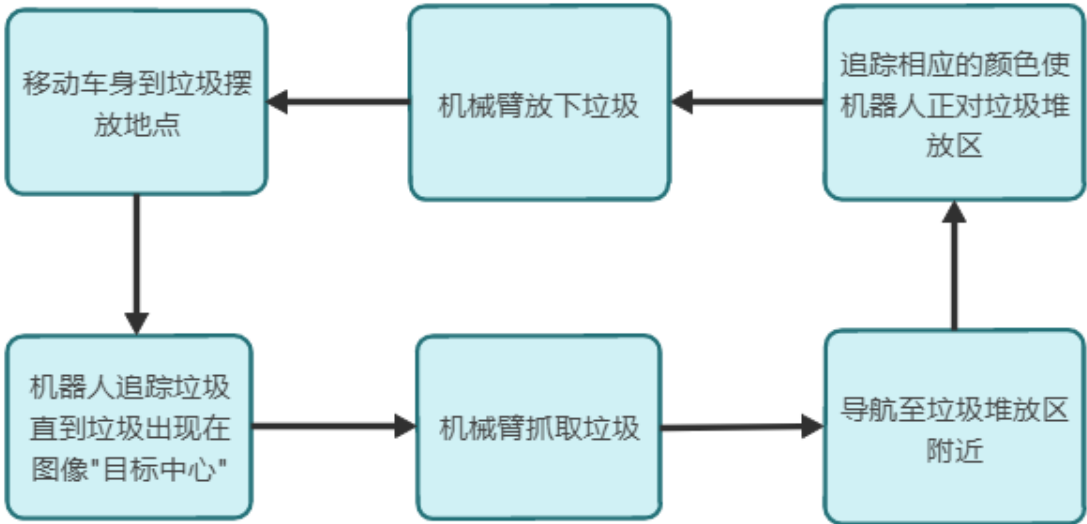
题目：光电智能垃圾分拣车

队名：67373upup

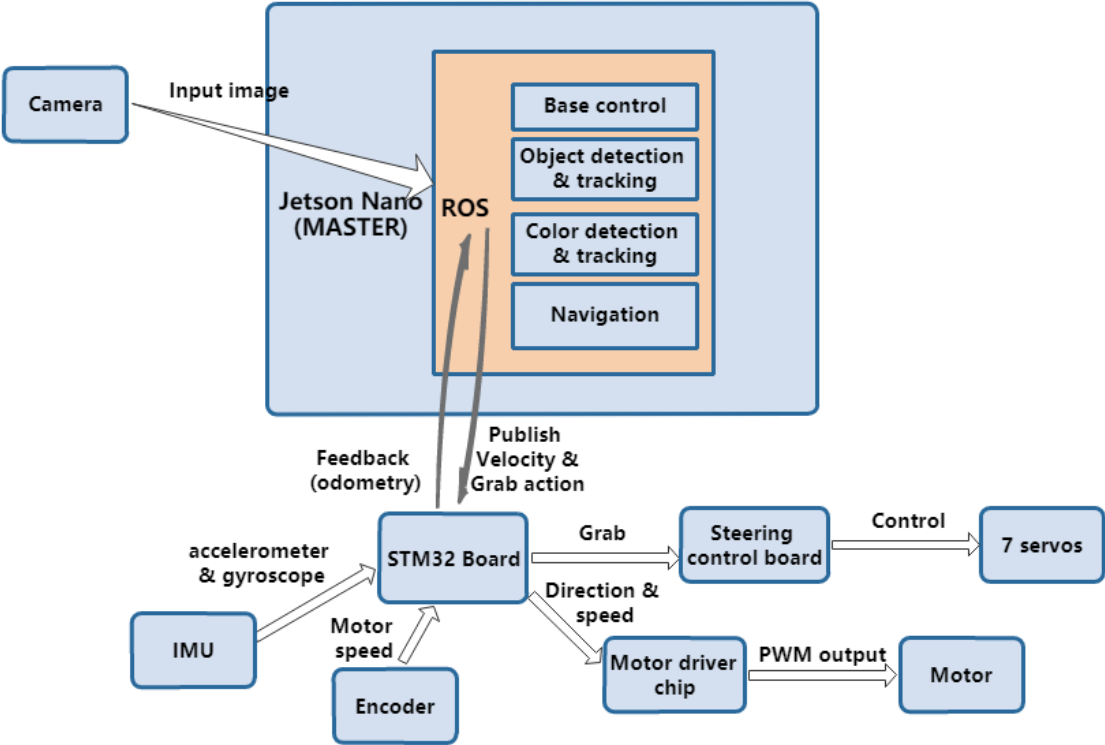
推荐理由：
1、自制扫把、簸箕垃圾拾拣机构；
2、基于 ROS 平台的垃圾分拣智能控制；
3、基于 PID 控制的机器人自主导航；
4、基于深度学习的高效的垃圾识别算法；
5、基于 OPENCV 的颜色识别算法。

队长姓名			学校		学号		专业班级	
E-mail						手机		
合 作 者	姓名	学号	专业、班级		所在学校		联系方式	
指 导 教 师	姓名	学校	所在院系		联系电话/手机		E-mail	

流程图：

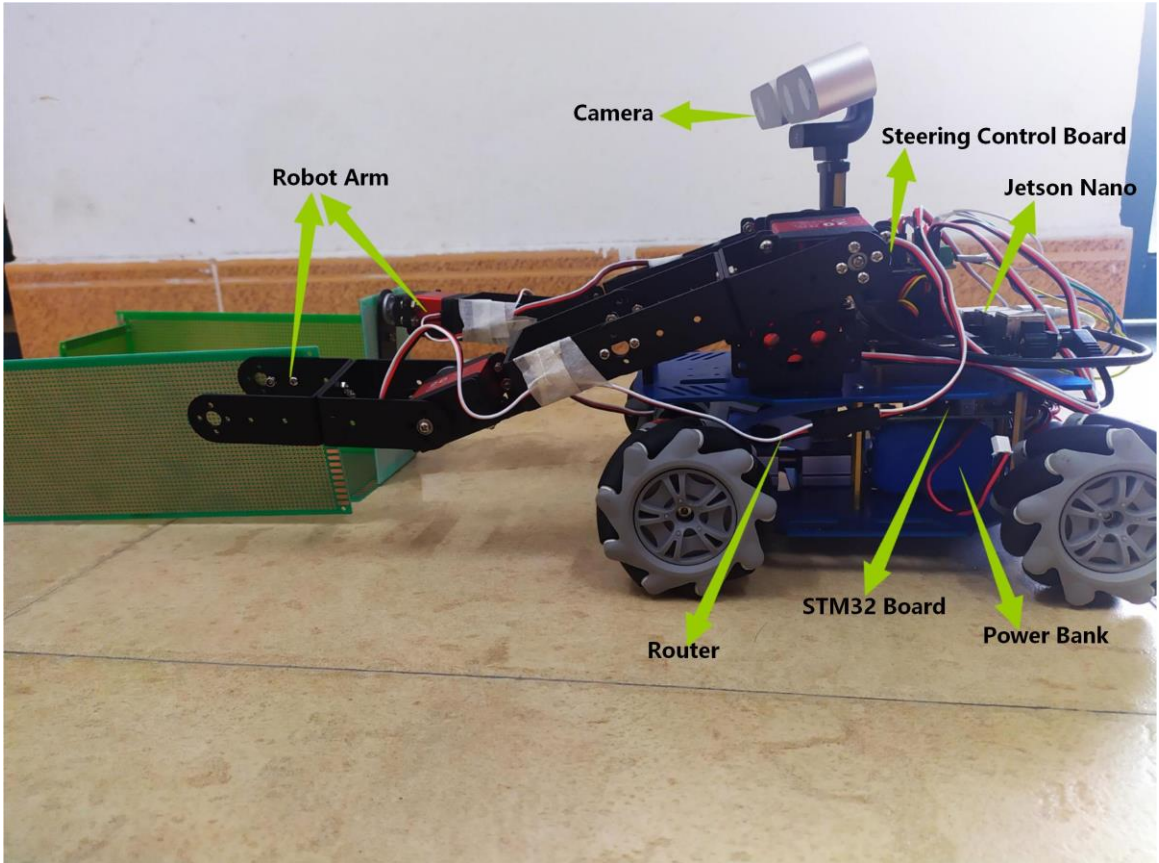


工作框架：



作
品
策
划
方
案

机器人结构：



硬件			
模块名称	实物图	提供商	功能
Jetson Nano B01		淘宝-makebit旗舰店	处理数据、主控制中心
STM32F405RGT6 芯片底层控制板		淘宝-幻宇机器人	底层中心控制板

作品策划方案	16 路舵机控制板		淘宝-深圳维特智能科技有限公司	接收下位机指令，集中控制机械臂上的多个舵机以完成抓取、放下动作																		
	Astra 乐视摄像头		淘宝-体感中国	实时获取每帧图像，用于目标检测																		
	TP-LINK 无线路由器		淘 宝 -TpLink 官方旗舰店	连接 Jetson Nano 和 PC，用于程序的拷贝、调试																		
	12.6V ，6000mAH 锂电池	<div><table><tr><td>型号:</td><td>HRX-1206</td></tr><tr><td>电压:</td><td>12V</td></tr><tr><td>容量:</td><td>6000mAh</td></tr><tr><td>电芯:</td><td>6节 18650 芯</td></tr><tr><td>尺寸:</td><td>70*54*37mm</td></tr><tr><td>重量:</td><td>约267g</td></tr><tr><td>保护功能:</td><td>过充、过放、短路等</td></tr><tr><td>可持续电流:</td><td>8A</td></tr><tr><td>过流保护值:</td><td>16A</td></tr></table></div>	型号:	HRX-1206	电压:	12V	容量:	6000mAh	电芯:	6节 18650 芯	尺寸:	70*54*37mm	重量:	约267g	保护功能:	过充、过放、短路等	可持续电流:	8A	过流保护值:	16A	淘宝-鸿锐翔电子有限公司	为所有组块供电
	型号:	HRX-1206																				
电压:	12V																					
容量:	6000mAh																					
电芯:	6节 18650 芯																					
尺寸:	70*54*37mm																					
重量:	约267g																					
保护功能:	过充、过放、短路等																					
可持续电流:	8A																					
过流保护值:	16A																					
麦克纳姆轮小车底盘		淘宝-幻宇机器人	做为运动平台载体																			

作品策划方案

20kg 大扭力舵机 (7个)		淘宝-星呗机器人官方商城	做为两个机械臂运动的关节
扫把、簸箕		我们自己	自制的扫把、簸箕用于扫取垃圾

软件/工具/库	
模块名称	描述
Ubuntu18.04	做为 ROS 的操作系统
Robot Operating System(ROS)	开发机器人项目的平台
OpenCV	图像处理
Rviz	将话题和视频流可视化
Yolov3	训练目标检测模型的框架
Keil	STM32 代码编写

总览概述:

我们的项目可以分解为四个方面，底层控制(Base Control)，目标检测和追踪(Object detection & tracking)，颜色检测和追踪(Color detection & tracking)，自主导航(Navigation)。

在底层控制(Base Control)中，我们从四个层面来阐述各环节的联动控制。首先是各模块的 12V, 5V, 3.3V 电力供应；然后是下位机向上位机实时发送传感器数据、上位机向下位机发送运动和控制命令；最后我们阐述用 Ros 中的 TF 包将各传感器的数据在不同参考系间进行转换。

在目标检测和追踪(Object detection & tracking)中，我们从两个层面来阐述垃圾识别分类的原理和方法。目标检测：我们在实际羽毛球场地拍摄垃圾图像并自主贴标签制作训练集训练模型，并用训练出来的模型检测视野内存在的目标垃圾；目标追踪：对视野内的某一目标垃圾进行追踪。

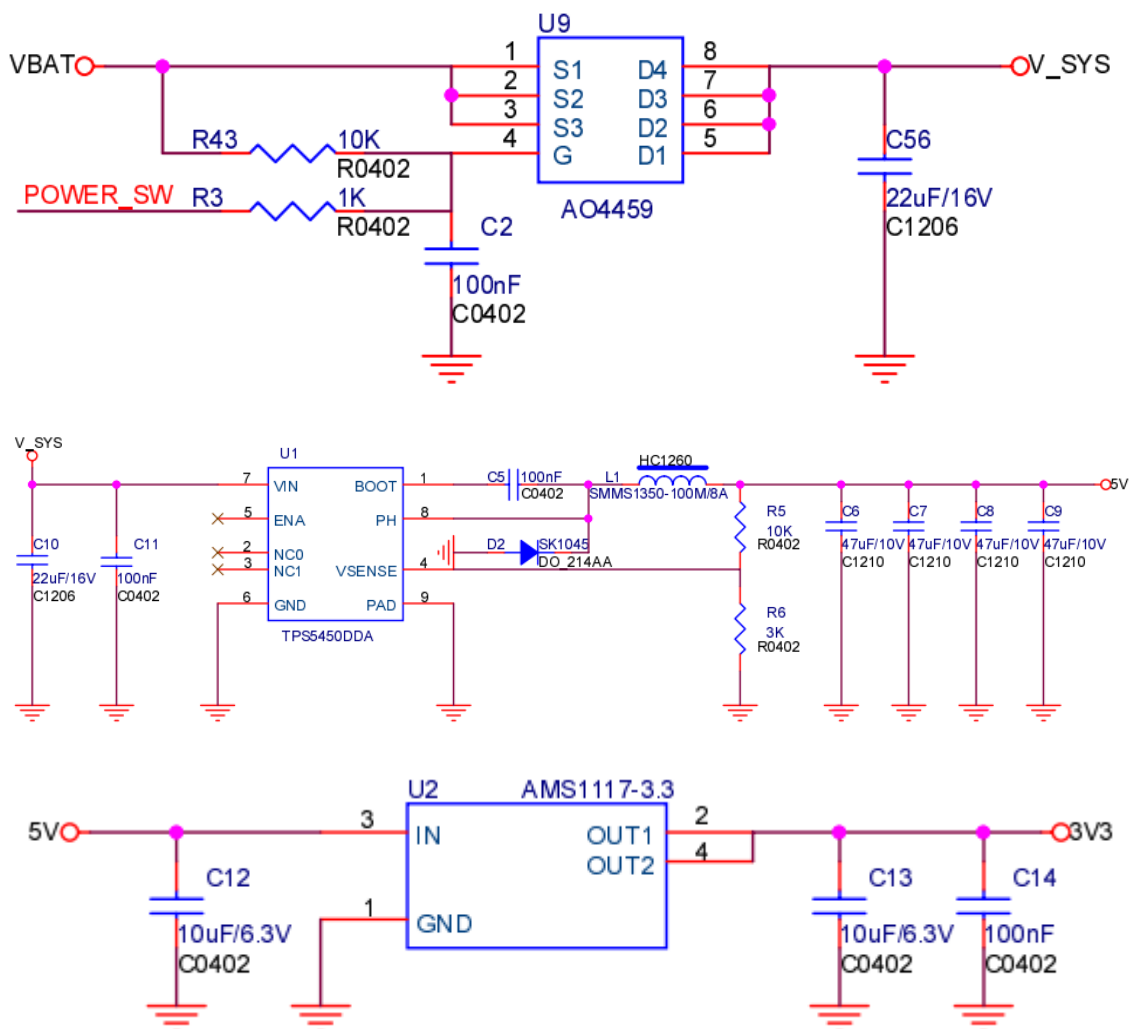
在颜色检测和追踪(Color detection & tracking)中，我们从两个层面来阐述如何识别垃圾堆放区域。颜色检测：用 OpenCV 来进行单颜色检测进而识别相应的垃圾堆放区域。颜色追踪：对垃圾堆放区域进行追踪。

在自主导航(Navigation)中，利用 PID 控制实现机器人自主从场地中一点移动到另一点。

1. 底层控制(Base control)

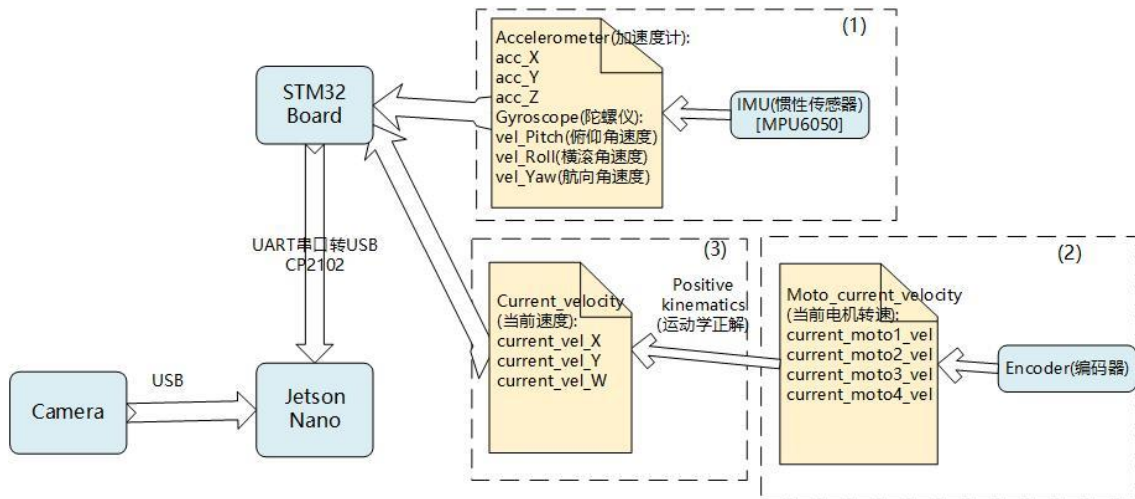
1.1 电池模块

底层电路逻辑图：



我们用的是两并三串 VBAT=12.6V，6000mAh 的 18650 锂电池组，通过 POWER_SW 开关和 AO4459MOS 管的配合来控制电源的通断，并用输出的 V_SYS 给 12V 的直流有刷电机供电。用 TPS5450 稳压降压芯片将 V_SYS 输入电压转换成 5V 的输出电压，来为 Jetson Nano 和 7 个 20kg 大扭力舵机供电。因为 AMS1117-3.3V 正向低压降稳压器内部集成过热保护和限流电路，能提高电路的稳定性，因此我们使用它来将 5V 电压转换成 3.3V 电压，来为 STM32F4 主控芯片、TB6612FNG 电机驱动模块、MPU6050 惯性传感器、舵机控制板等供电。

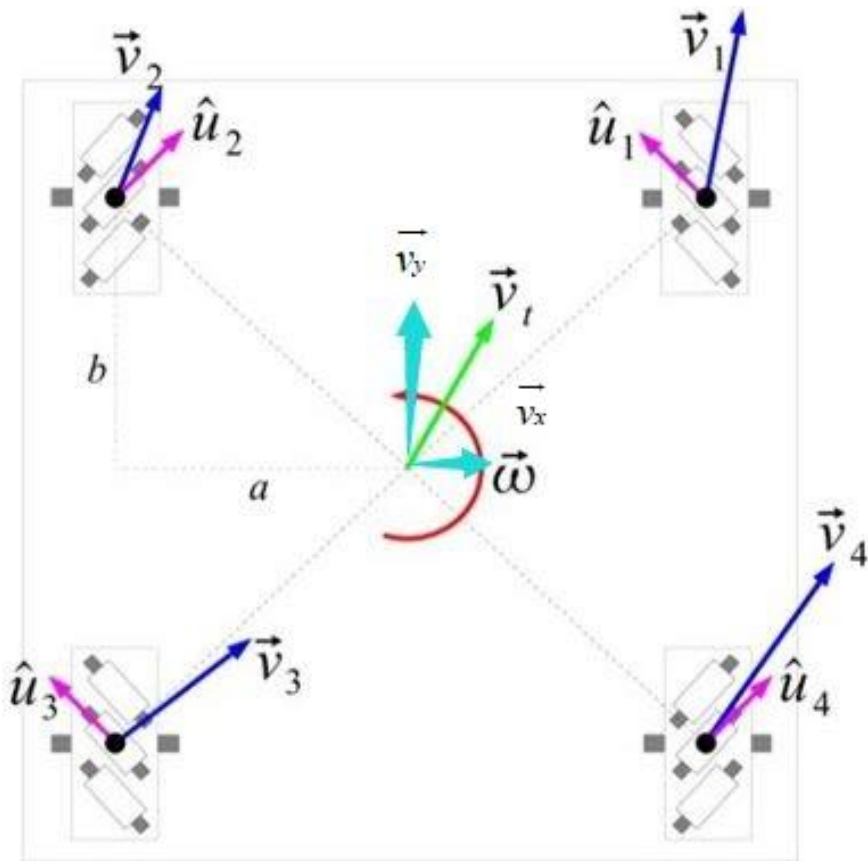
1.2 下位机向上位机传送数据 数据传送逻辑图：



下位机 STM32 实时获取 IMU 传来的姿态数据和编码器传来的里程计数据，并以 20Hz 的频率通过 UART 串口转 USB 的 CP2102 模块向上位机 Jetson Nano 传送；USB 摄像头以 30FPS 的帧率向上位机传送视野内图像信息。

- (1) 我们通过 IIC 总线读取 IMU（惯性传感器）的三轴角速度（ $vel_Pitch, vel_Roll, vel_Yaw$ ）以及三轴加速度（ acc_X, acc_Y, acc_Z ）。由于我们的车载体在二维平面内运动，不涉及到俯仰和横滚，所以我们只关心 acc_Z, vel_Yaw 即可。由于陀螺仪输出的角速度是瞬时量，角速度在姿态平衡上是不能直接使用，需要角速度与时间积分计算角度，得到的角度变化量与初始角度相加，才能得到目标角度，其中 dt 越小输出角度越精确，但陀螺仪的原理决定了它的测量基准是自身，并没有系统外的绝对参照物，再加上 dt 是不可能无限小，所以积分的累积误差会随着时间流逝迅速增加，最终导致输出角度与实际不符，所以对于加速度计和陀螺仪的动静态的不同，我们在 Ros 层采用了扩展卡尔曼滤波 (EKF) 利用不同的权值矩阵协方差进行姿态融合。
- (2) 我们使用非接触式增量编码器获得 dt (50ms) 时间内编码器输出的脉冲数 ($pulses$)，再由已知的轮子每圈脉冲数 ($cycle_Pulses$) 和轮子直径 ($diameter$) 通过公式 $moto_vel = (\pi * diameter * (pulses / cycle_Pulses)) / dt$ 计算获得车载体的四个轮子的运动速度。
- (3) 我们采用的是麦克纳姆轮的车载体，所以由其运动学正解公式可以知道 v_x, v_y, w （ v_x 是 x 方向分速度， v_y 是 y 方向分速度， w 是逆时针角速度， v_1, v_2, v_3, v_4 分别是车载体的四个轮子的运动速度）。

麦克纳姆轮速度分解图：



作
品
策
划
方
案

运动学正解公式：

$$v_x = (v_4 - v_3 + v_2 - v_1) / 4$$

$$v_y = (v_4 + v_3 + v_2 + v_1) / 4$$

$$w = (v_4 - v_3 - v_2 + v_1) / 4(a + b)$$

运动学逆解公式：

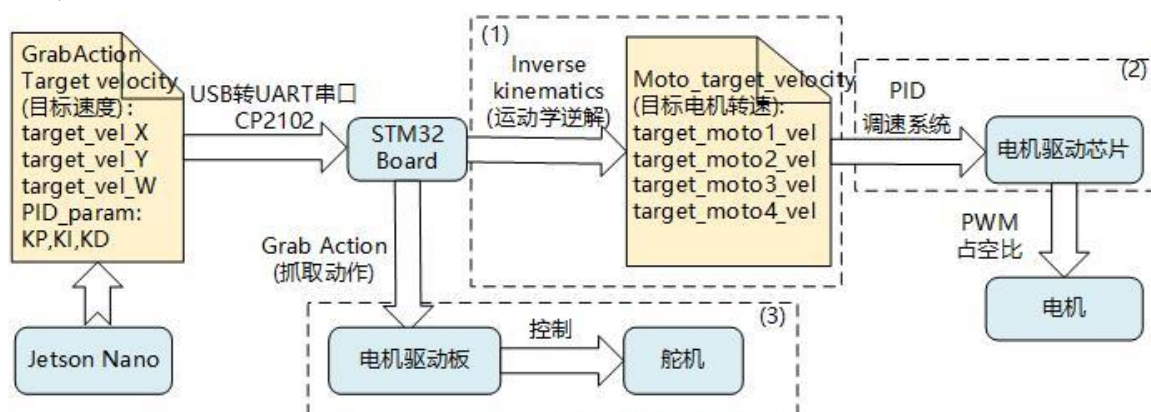
$$v_1 = v_y - v_x + w^*(a + b) / 2$$

$$v_2 = v_y + v_x - w^*(a + b) / 2$$

$$v_3 = v_y - v_x - w^*(a + b) / 2$$

$$v_4 = v_y + v_x + w^*(a + b) / 2$$

1.3 上位机向下位机发送命令 命令发送逻辑图：

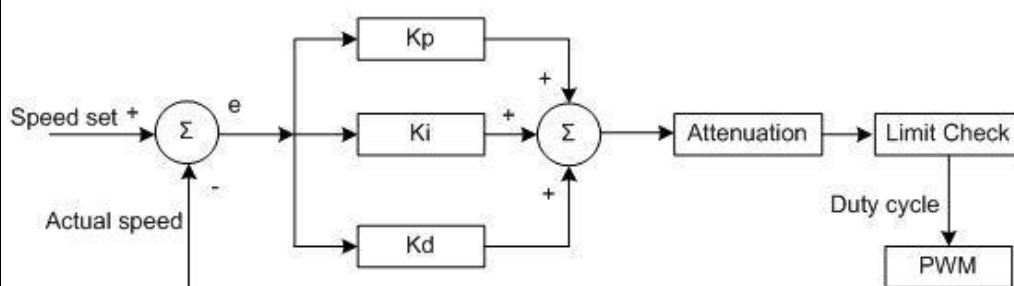


上位机 Jetson Nano 向下位机发送目标速度、PID 参数和抓取动作命令，下位机将其视为中断进行处理。

(1) 上位机发送下来的 v_x, v_y, w 经麦克纳姆轮运动学逆解公式解算后得到目标转速

$$v_1, v_2, v_3, v_4。$$

PID 控制逻辑图：



PID 主要代码：

```
//更新当前的一阶以进行当前比例积分微分计算
LL_Error = L_Error;
L_Error = Error;
Error = target_moto_vel - current_moto_vel;

P_Error = Error;    //比例项
I_Error = Error - L_Error;    //积分项
D_Error = Error - 2*L_Error + LL_Error;    //微分项

//计算比例积分微分，输出占空比
add = KP*P_Error + KI*I_Error + KD*D_Error
Output_PWN +=add;
```

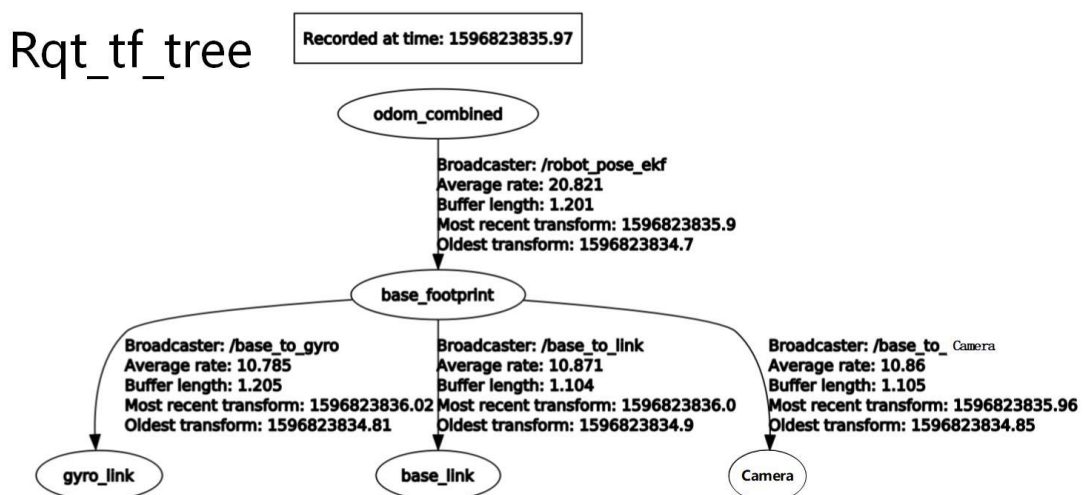
(2) 因为我们通过编码器知道电机当前的转速，经(1)解算后可以知道电机的目标转速，所以我们可以使用 PID 调节转速，从而使机器人的控制效果更及时。

(3) 我们采用自制的两个机械臂，左臂扫把(3 个舵机控制)，右臂簸箕(4 个舵机控制)协同合作将垃圾扫进簸箕中，并统一用舵机控制板控制这 7 路舵机。

1.4 传感器数据融合

机器人在工作过程中会接收到多个传感器传来的数据，但这些数据都是相对传感器坐标系而言的，所以我们需要一种手段来同时追踪多个参考系，并且可以将点、向量等数据的坐标在任意两个参考系中完成坐标转换。恰巧 ROS 的 TF (Transform) 功能包就提供了不同参考系之间变换的功能，我们只需告诉 TF 树这些参考系之间的变换公式，TF 就会自动管理我们所需要的参考系变换。

TF 树：



Rqt_tf_tree 树结构说明：

(1) 我们知道摄像头、编码器、姿态传感器都是固定在机器人上的，他们和机器人的位置是永远固定的，不会随机器人运动而变化，所以我们使用 TF 提供的静态发布固定 TF 转换功能来管理他们。将编码器数据正解后得到的里程计参考系 (base_link)，姿态传感器参考系 (gyro_link)，摄像头参考系 (camera) 和机器人中心参考系 (base_footprint) 建立连系。

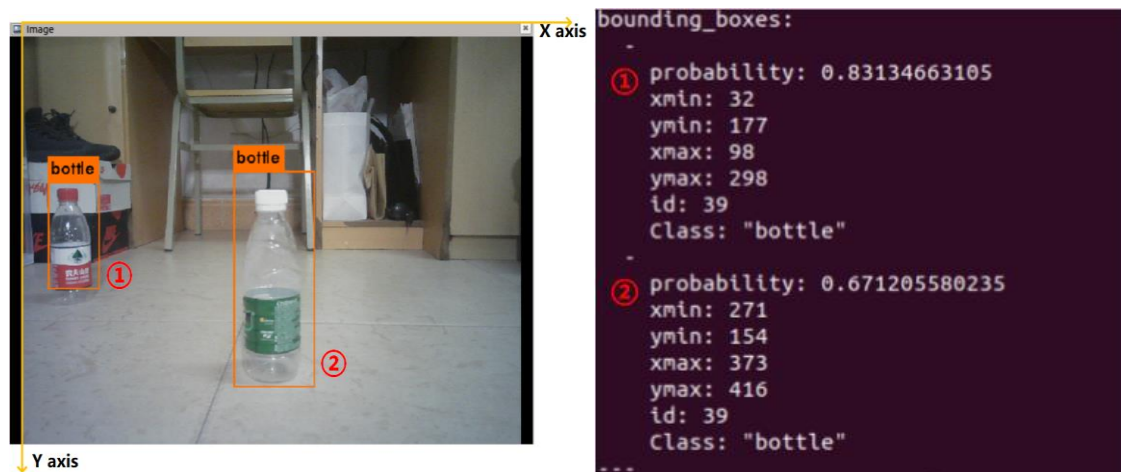
(2) odom_combined->base_footprint 之间的 tf 转换是由 /robot_pose_ekf 提供的，也就是说机器人从上电时刻开始推演的轨迹是由 EKF 融合后提供的。

2. 目标检测和追踪(Object detection & tracking)

2.1 目标检测(Object detection)

2.1.1 主体思路

摄像头目标检测图像及命令行结果显示：



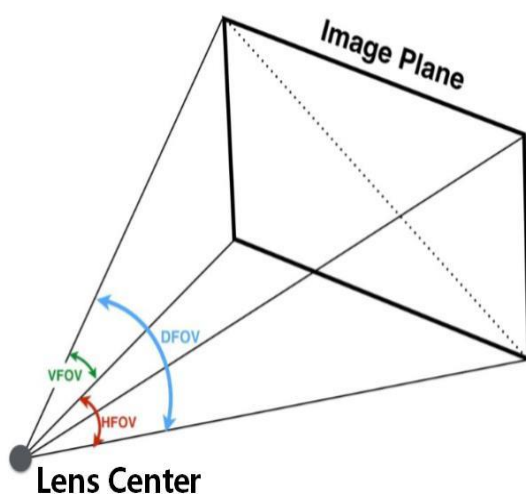
作品策划方案

我们用 USB 接口摄像头实时获取机器人正前方每帧图像信息，并将图像喂进我们事先用 YOLOv3 网络训练好的目标检测模型中，进而推理出帧图像中垃圾的位置。模型反馈给我们的信息有 *probability, xmin, ymin, xmax, ymax, id, class*。*probability* 代表推理出的结果的置信度，我们根据经验设置阈值 *threshold*，凡是 *probability > threshold* 的推理，我们便相信该推理是准确的，该处确实存在垃圾。

xmin, ymin, xmax, ymax 代表图像中垃圾位置的矩形框信息。*id, class* 则提供给了我们相应垃圾的种类信息。

2.1.2 摄像头

摄像头视场角示意图：



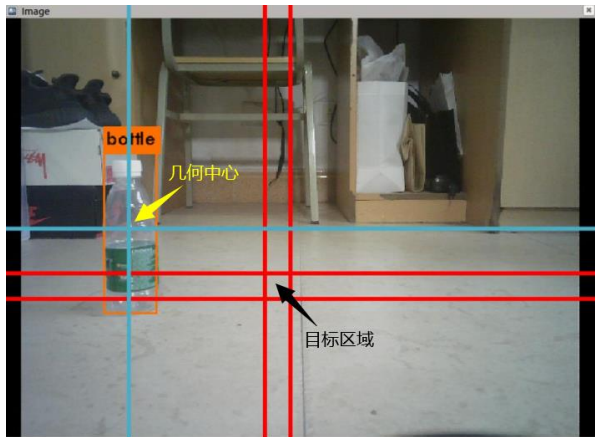
详细说明：

1. 我们使用的是 Astra 乐视广角摄像头，水平视场角(HFOV)是 66.1° ，垂直视场角(VFOV)是 40.2° ，视场角较大，易于检测到更大视野范围内的目标；图像畸变较小，图像处理更容易、准确。

2. 我们设置的彩色图像分辨率是 640×480 的，帧率在 Jetson Nano 上有 30FPS，目标检测的频率能达到 5FPS，对于我们速度发布频率为 20Hz 的机器人来说，二者频率是可以匹配的。

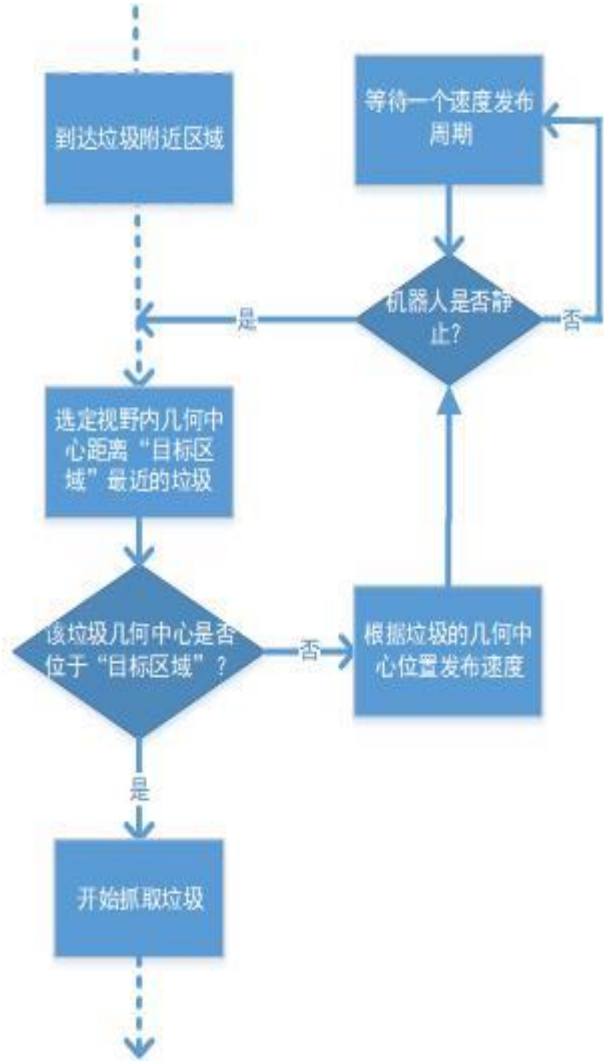
2.2 目标追踪(Object tracking)

2.2.1 主体思路



Object tracking 的任务是识别垃圾的几何中心在视频帧中的位置，并通过轮子的移动将垃圾的几何中心移动到视频帧的特定小区域内（我们将该特定小区域称之为“目标中心”，）。一旦将该垃圾的几何中心位于“目标中心”内，机器人与特定垃圾之间的相对位置就是绝对固定的了，这时便可发布命令让机械臂进行预先设计的动作组以抓住该垃圾。

2.2.2 流程图



流程图说明：

1. 目标检测提供给了我们垃圾在图像中的具体位置信息，我们根据式子 $centerx = (xmin + xmax) / 2$ 可以算出 $centery = (ymin + ymax) / 2$ 垃圾的几何中心。

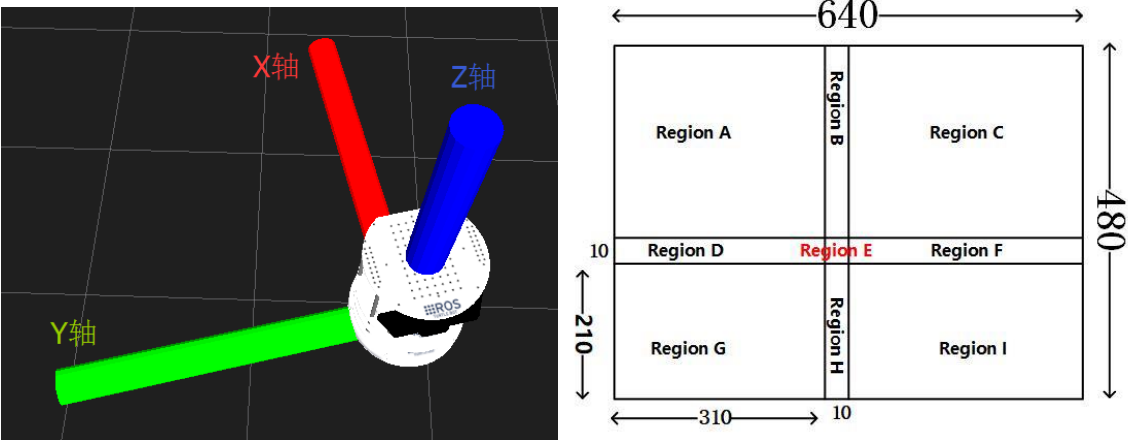
2. 摄像头的视野内可能有多个目标，我们选取几何中心距离“目标区域”中心最近的垃圾进行追踪，因为这样的垃圾在实际中距离机器人的物理距离也是最近的。

3. 我们根据垃圾的几何中心相对目标区域的位置将视频帧分为 9 个部分，每个部分对应于要发布的速度。

4. 上位机向下位机发布速度命令的频率是 20Hz，即一个速度发布周期 50ms，当机器人获得并执行一次速度命令后，我们需等待其完成该次运动后（即机器人静止）再开启下一轮判断，否则机器人会陷入无限“徘徊震荡”当中。

2.2.3 运动追踪

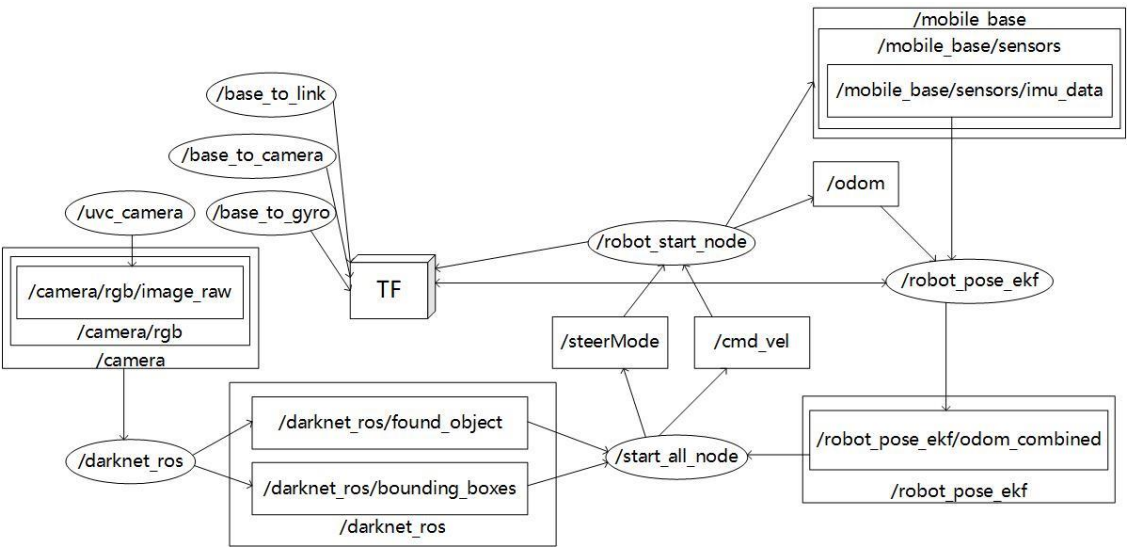
我们将目标垃圾的几何中心在图像中位置分为九个区域，每个区域对应于一个相应要发布的速度，我们的目标是使垃圾的几何中心位于“目标区域”（Region E）。



注：ROS 中所用的右手坐标系

视频帧区域	机器人运动	X 方向线速度 m/s	Y 方向线速度 m/s	Z 方向角速度
Region A	向前走、向左走	$V_x=0.01$	$V_y=0.01$	$V_z=0$
Region B	向前走	$V_x=0.01$	$V_y=0$	$V_z=0$
Region C	向前走、向右走	$V_x=0.01$	$V_y=-0.01$	$V_z=0$
Region D	向左走	$V_x=0$	$V_y=0.01$	$V_z=0$
Region E	定位成功	$V_x=0$	$V_y=0$	$V_z=0$
Region F	向右走	$V_x=0$	$V_y=-0.01$	$V_z=0$
Region G	向后走、向左走	$V_x=-0.01$	$V_y=0.01$	$V_z=0$
Region H	向后走	$V_x=-0.01$	$V_y=0$	$V_z=0$
Region I	向后走、向右走	$V_x=-0.01$	$V_y=-0.01$	$V_z=0$

2.2.4 Ros 节点话题关系图



节点名称(Node name):/uvc_camera			
话题(Topic)	方向	消息类型(Message type)	功能
/camera/rgb /image_raw	Publisher	Sensor_msgs/Image	得到摄像头图像

节点名称(Node name):/darknet_ros			
话题(Topic)	方向	消息类型(Message type)	功能
/camera/rgb/ image_raw	Subscriber	Sensor_msgs/Image	获取摄像头图像
/darknet_ros/ found_object	Publisher	Darknet_ros_msgs/ ObjectCount	输出视野内目标个数
/darknet_ros/ bounding_boxes	Publisher	Darknet_ros_msgs/ BoundingBoxes	输出视野内所有目标的位置、类别

节点名称(Node name):/start_all_node			
话题(Topic)	方向	消息类型(Message type)	功能
/darknet_ros/ found_object	Subscriber	Darknet_ros_msgs/ ObjectCount	获得视野内目标个数
/darknet_ros/ bounding_boxes	Subscriber	Darknet_ros_msgs/ BoundingBoxes	获得视野内所有目标的位置、类别
/robot_pose_ekf/ odom_combined	Subscriber	geometry_msgs/ PoseWithCovarianceStamped	获取机器人具体位置
/steerMode	Publisher	Std_msgs/Char	发布机械手动作命令
/cmd_vel	Publisher	geometry_msgs/Twist	发布运动命令

节点名称(Node name):/robot_pose_ekf			
话题(Topic)	方向	消息类型(Message type)	功能
/odom	Subscriber	nav_msgs/Odometry	获得编码器解算后的里程计信息
/mobile_base/ sensors/imu_data	Subscriber	sensor_msgs/Imu	获得 IMU 信息
/robot_pose_ekf/ odom_combined	Publisher	geometry_msgs/ PoseWithCovarianceStamped	发布传感器融合后的里程计信息

robot_pose_ekf 节点用于评估机器人的 3D 位姿，基于来自不同来源的位姿测量信息。它使用带有 6D(3D position and 3D orientation)模型信息的扩展卡尔曼滤波器来整合来自轮式里程计的消息 odom 和 IMU 传感器的消息 imu_data,用松耦合方式融合不同传感器信息实现位姿计。

3. 颜色检测和追踪 (Color detection & tracking)

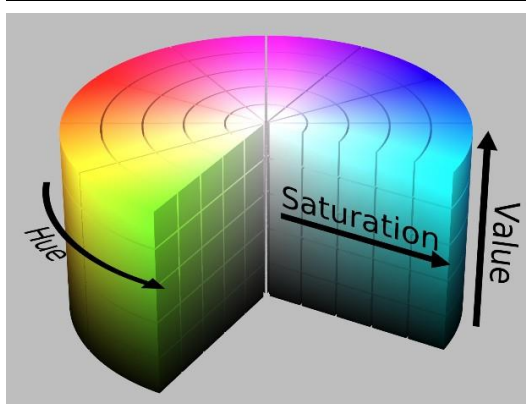
3.1 颜色检测 (Color detection)

3.1.1 主体思路

由于可回收、厨余、有害、其他垃圾堆放区域分别用蓝、绿、红、黑灰颜色进行标定，所以我们用 OpenCV 来进行单颜色检测，进而识别不同颜色的垃圾堆放区。每一帧的图像默认由 RGB 存储，因此我们通过五个步骤进行颜色检测：

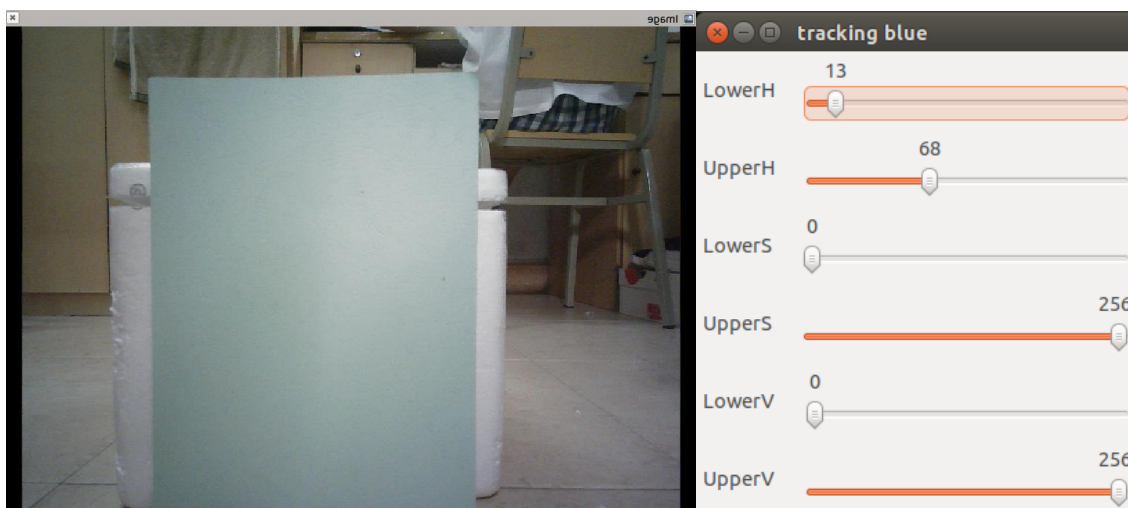
- 读入帧图像，将其转换到 HSV 空间
- 设定 HSV 边界来检测特定颜色并产生掩膜 mask
- 应用形态学腐蚀操、高斯滤波作来消除 mask 上的噪声
- 检测颜色区域并获得其集中区域

Step 1: 读入帧图像，将其转换到 HSV 空间



摄像头读入的每一帧图像是在 RGB 空间的，由于在自然环境下获取的图像容易受自然光照、遮挡和阴影的影响，即对亮度敏感，而 RGB 空间的三个分量都与亮度密切相关，即只要亮度改变，三个分量都会随之相应地改变，没有一种更直观的方式来表达。因此我们将其转换到 HSV 空间：Hue（色调、色相），Saturation（饱和度、色彩纯净度），Value（明度），方便进行颜色的对比。

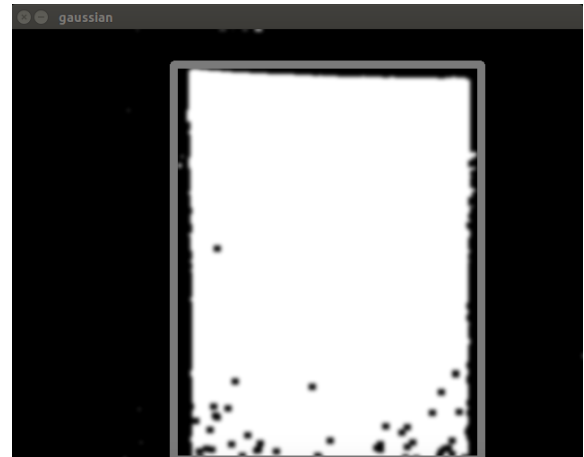
Step 2: 设定 HSV 边界来检测特定颜色并产生掩膜 mask





我们需要设置目标 HSV 值，然后设置一个合适的范围来消除其他值。为了实现这一点，我们创建了六轨条，相应地控制 HSV 的下限值和上限值。程序将从轨迹栏读取值并使用它来生成掩膜。cv::inRange() 函数只返回一个二进制掩码，其中白色像素（255）表示落在上限和下限范围内的像素，而黑色像素（0）则表示在范围之外的像素。

Step 3: 应用形态学腐蚀操作来消除 mask 上的噪声



进行颜色分割之前，我们需要用形态学变换中的腐蚀操作、高斯滤波来消除噪声。腐蚀操作是对白色部分（高亮部分）而言的，而非黑色部分。腐蚀就是原图中的高亮部分被腐蚀，“领域被蚕食”，效果图拥有比原图更小的高亮区域。高斯滤波就是将图像平滑化处理，或者说图像模糊处理，用一个模板（或称卷积、掩模）扫描图像中的每一个像素，用模板确定的邻域内像素的加权平均灰度值去替代模板中心像素点的值，以达到消除噪声的目的。

Step 4: 检测颜色区域并获得其集中区域

OpenCV 提供了在掩膜 mask(cnt) 中查找白色轮廓的功能。由于垃圾堆放区呈矩形。cv::boundingRect(cnt) 返回掩膜 mask 中的白色区域的矩形位置。由此，我们可以使用 cv::rectangle() 将其绘制出来以进行可视化。

3.2 颜色追踪(Color tracking)

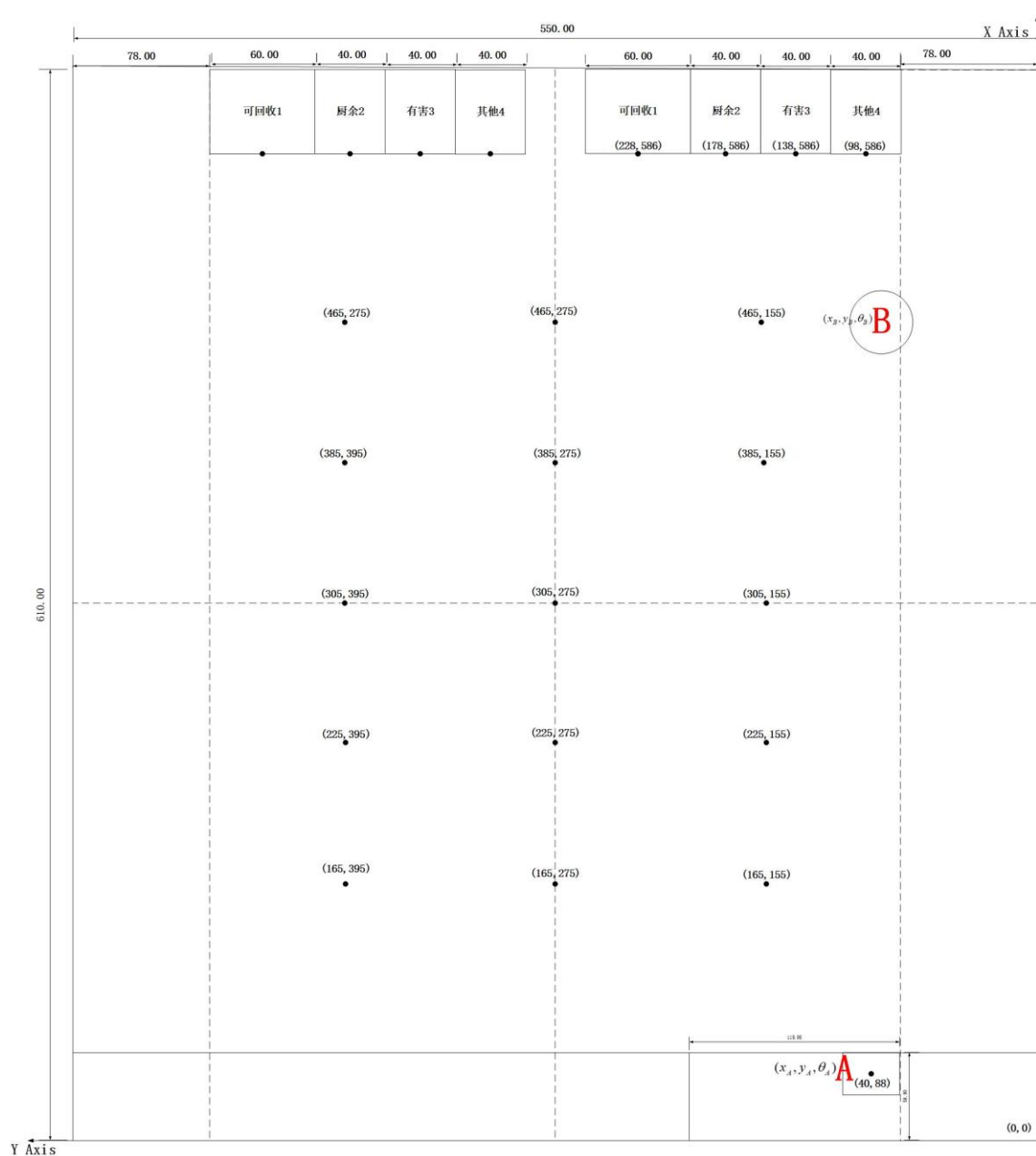
与目标追踪(Object tracking)类似，不断移动车身使相应垃圾堆放区的单色板“几何中心”位于目标区域中。

4. 自主导航 (Navigation)

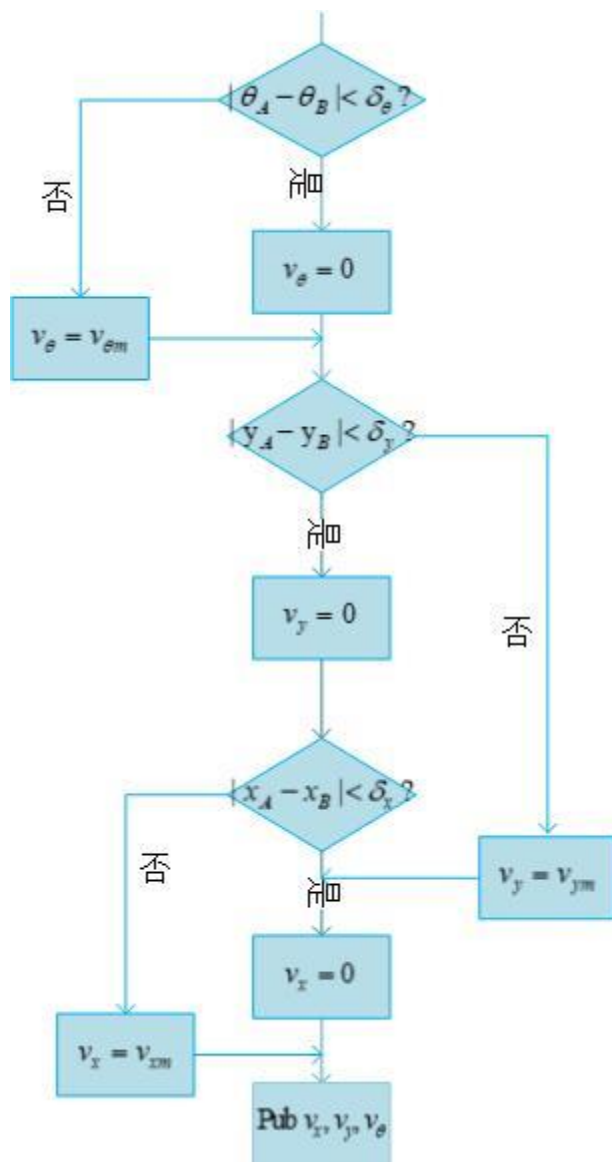
4.1 主体思路

机器人未抓取垃圾时需要从它的位置运动到垃圾摆放点附近，抓到垃圾后需要运动到相应的垃圾堆放区，在运动过程中要避免机器人出界。地图中的垃圾摆放点、垃圾堆放区、场地等二维坐标均是清楚已知的。而 ROS 层的 /robot_pose_ekf/odom_combined 话题给出了较为精确的实时机器人位置 (x, y, θ) 。

因此我们可以将机器人所有自主导航的任务统一归类为从 A 点 (x_A, y_A, θ_A) 移动到 B 点 (x_B, y_B, θ_B) 。STM32 以 20Hz 的频率向 ROS 层提供位置 (x, y, θ) ，速度 (v_x, v_y, v_θ) ，因此我们每隔 50ms 进行一次判断以规划机器人下一步的运动。



4.2 流程图



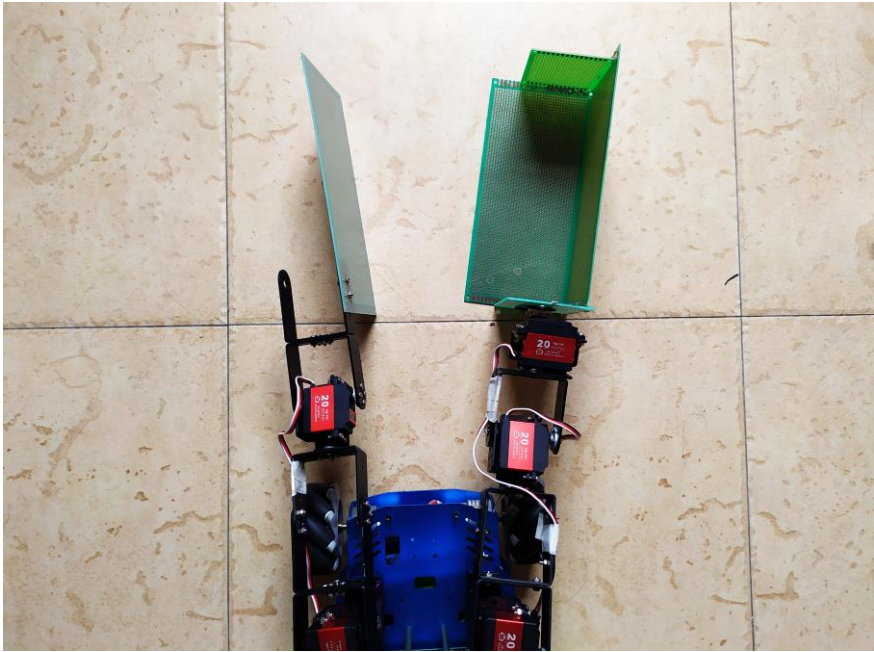
流程图说明：

1. 下位机 STM32 每隔 50ms 采样计算一次机器人在地图中的精确位置，上位机 ROS 根据实时当前位置和目标位置进行调整速度。
2. 机器人的位置无法完全精确，因此根据经验设置 x, y, θ 合理误差范围 δ 。
3. 机器人的朝向对于每次位置的改变影响极大，因此每次均进行方向 θ 的矫正。
4. 将机器人的运动分解为 x, y 两个方向的运动，机器人每次只完全 x 或 y 一个方向的运动。

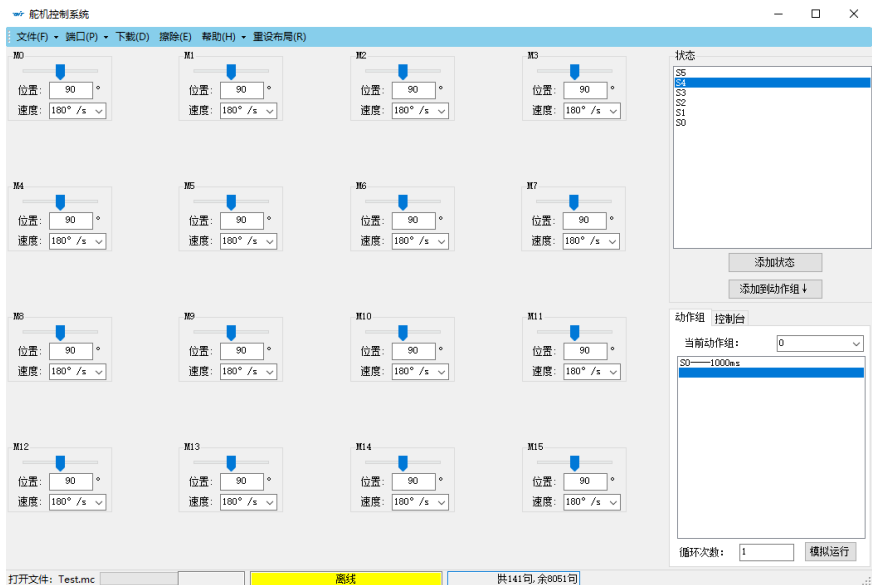
设计说明和效果展示：

在垃圾分类车这个项目中，我们人员分工是机械组，负责各硬件模块的电气通信和物理空间布局；视觉组，负责目标检测模型的训练和颜色识别；电控组，负责各模块软件的逻辑编写。从功能上，该项目最核心的三个部分是：①垃圾拾取与放下②垃圾检测与分类③行进与运动

① 垃圾拾取与放下
机械臂效果图：



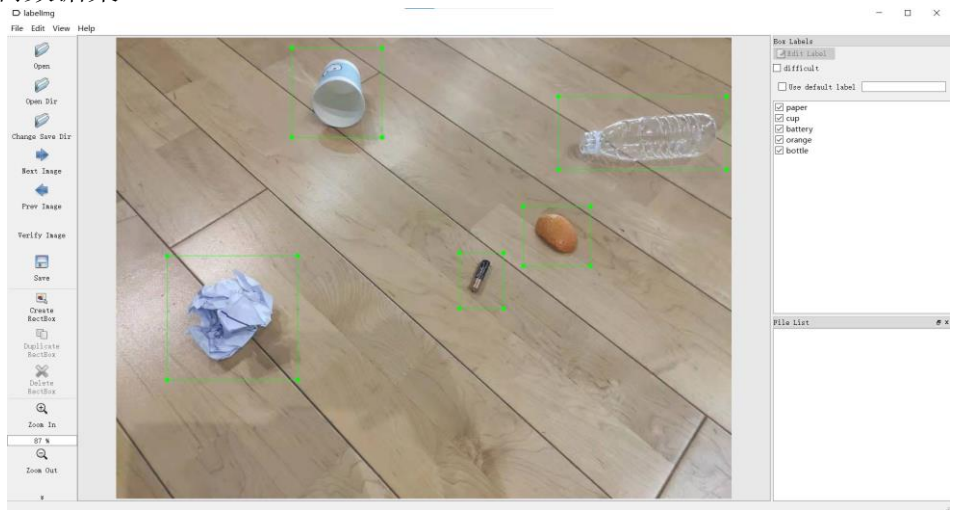
舵机控制软件界面：



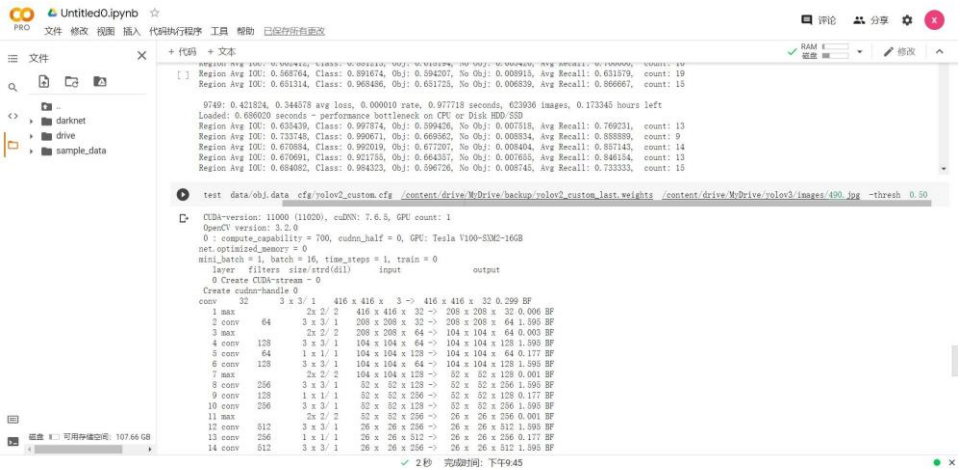
我们拾取、放下垃圾用的是“扫取”的方式，模拟人类扫垃圾，左机械臂由三个舵机组成“扫把”；右机械臂由四个舵机组成“簸箕”。总共七路舵机统一由舵机控制板控制协调工作。

舵机控制板可由舵机控制软件十分方便地写入命令。七个舵机的位置和速度保存成一个状态；若干个状态及其持续时间组成一个动作组。我们共预先保存了三个动作组，分别完成的是机械臂初始化、垃圾扫取、垃圾放下。而STM32底层板只需要简单地调用每个动作组即可。

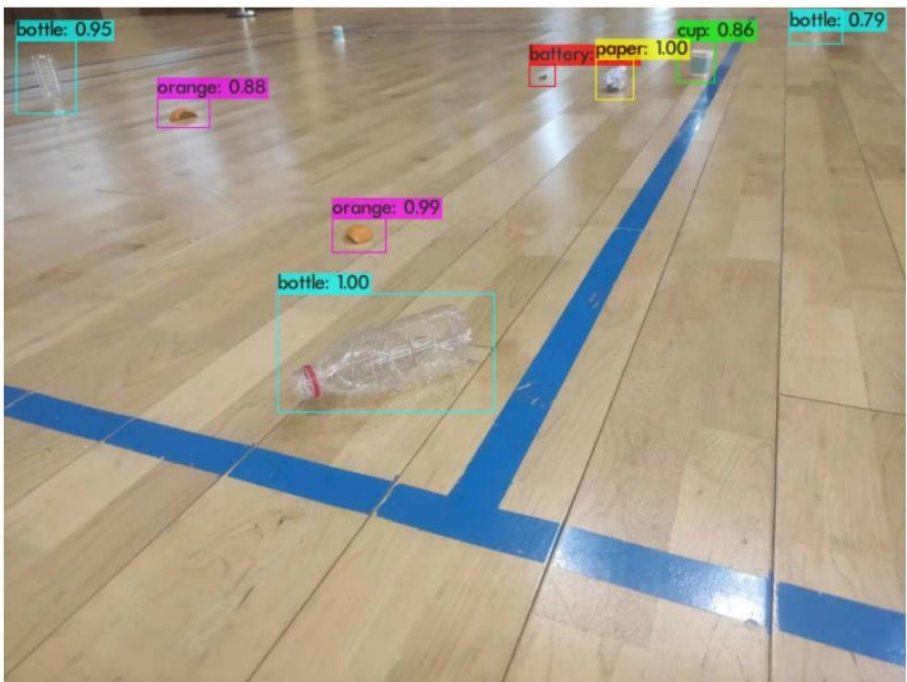
② 垃圾检测与分类
自制数据集：



谷歌Colab云实验室训练：



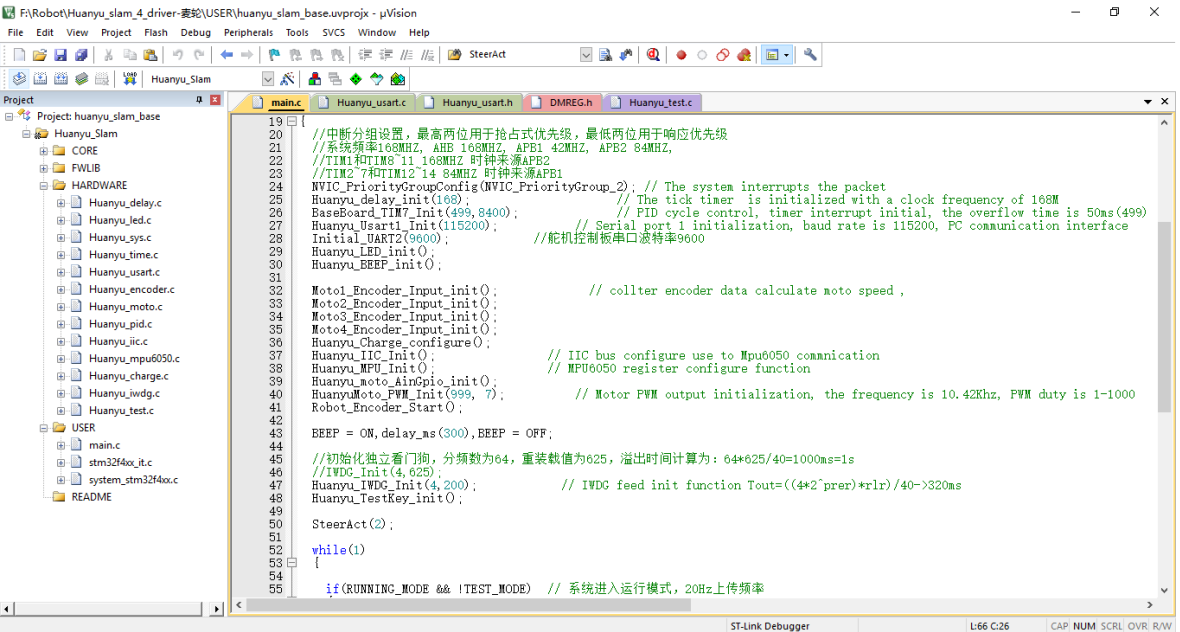
目标检测结果图：



我们在实际羽毛球馆拍摄规定的五种垃圾图像并自主贴标签共构建了包含492张图像、不同垃圾各种形态的数据集，并在谷歌Colab云实验室用小巧轻量级且功能强大的Yolo v3目标检测框架进行训练。在测试集中，检测准确率在90%以上；在实际羽毛球场地中随机拍摄的图像，垃圾检测准确率均能在70%以上，且检测速率在Jetson Nano上有5FPS，完全符合我们的应用需求。

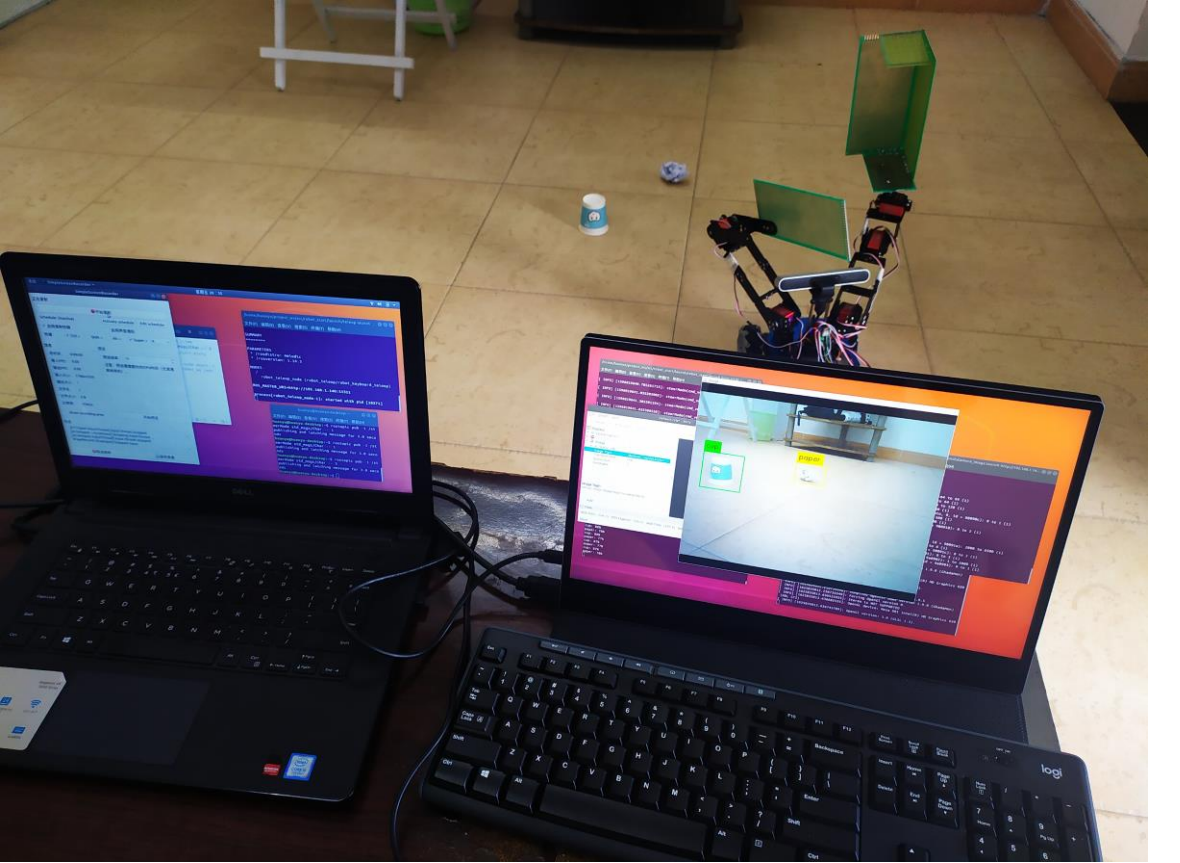
③ 行进与运动

Keil代码框架：



```
19 // 中断分组设置，最高两位用于抢占式优先级，最低两位用于响应优先级
20 // 系统频率168MHz, AHB 168MHz, APB1 42MHz, APB2 84MHz,
21 // TIM1和TIM8 11 168MHz 时钟来源APB2
22 // TIM2 7和TIM12 14 84MHz 时钟来源APB1
23
24 NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2); // The system interrupts the packet
25 Huanyu_delay_init(160); // The tick timer is initialized with a clock frequency of 168MHz
26 BaseBoard_TIM7_Init(499, 8400); // PID cycle control, timer interrupt initial, the overflow time is 50ms(499)
27 Huanyu_Uart1_Init(115200); // Serial port 1 initialization, baud rate is 115200, PC communication interface
28 Initial_UART2(9600); // 舵机控制板串口波特率9600
29 Huanyu_LED_init();
30 Huanyu_BEEP_init();
31
32 Moto1_Encoder_Input_init(); // collter encoder data calculate moto speed ,
33 Moto2_Encoder_Input_init();
34 Moto3_Encoder_Input_init();
35 Moto4_Encoder_Input_init();
36 Huanyu_Charge_configure();
37 Huanyu_IIC_Init(); // IIC bus configure use to Mpu6050 communication
38 Huanyu_MPU_Init(); // MPU6050 register configure function
39 Huanyu_moto_AinGpio_init();
40 HuanyuMoto_PWM_Init(999, 7); // Motor PWM output initialization, the frequency is 10.4Khz, PWM duty is 1-1000
41 Robot_Encoder_Start();
42
43 BEEP = ON, delay_ms(300), BEEP = OFF;
44
45 //初始化独立看门狗，分频数为64，重载值为625，溢出时间计算为：64*625/40=1000ms=1s
46 //IWDG_Init(4, 625)
47 Huanyu_IWDG_Init(4, 200); // IWDG feed init function Tout=((4*2^prer)*rldr)/40->320ms
48 Huanyu_TestKey_init();
49
50 SteerAct(2);
51
52 while(1)
53 {
54     if(RUNNING_MODE && !TEST_MODE) // 系统进入运行模式，20Hz上传频率
```

代码编写调试程序：



STM32板子集成有IMU位姿传感器、CP2102串口芯片、编码器电机芯片等，能完成与上位机Jetson Nano的通信，并向电机、舵机控制板发送命令，是底层核心，我们用Keil5烧写代码完成机器人的行进与运动。

作品创新点

1. 我们在实际羽毛球场地拍摄垃圾图像构建了数据集，并用轻巧强大的Yolov3训练模型，模型的运算速度快，泛化能力强，识别分类准确率高。
2. 我们的机械臂由“扫把”、“簸箕”组成，无论何种大小形状的垃圾，拾取成功率高，几乎不会掉垃圾。
3. 我们的车型选用的麦克纳姆轮，能直走、横走和转向，机器人的动作更加多样。