

第七届

全国大学生集成电路创新创业大赛

报告类型: 设计报告

参赛杯赛: Robei 杯 (A 组, 仅限本科生)

作品名称: 基于 RobeiEDA 工具的可重构智能仓储系统

队伍编号: CICC1235

团队名称: 我看你说的都队

基于 Robei EDA 工具的可重构智能仓储系统

Reconfigurable intelligent warehousing system based on Robei EDA software

团队名称：我看你说的都队

团队编号：CICC1235

团队成员：朱锦添，余韶伟，宗少云

摘要：随着进出货物流量和速度需求的日益增长，普通老旧的仓储物流模式已经无法满足当前高度信息化便捷化社会的发展要求，特别是中小型的仓储设施。然而，由于场地和资金的限制，这些仓储场所进行设施升级时往往遭遇困难。本团队设计了一款基于 Robei EDA 工具的智能仓储系统，以视觉识别，机械臂抓取，麦克纳姆轮小车驱动为基础，多种交互方式，实现了多场景适应、便捷移植、对环境需求低且成本低的智能化仓储设施。经过系统验证，本系统可以达成预期要求，随着时间的推移，我们将继续完善系统，并进一步提高其功能和性能，以满足不断发展的仓储需求。

Abstracts: With the increasing demands for volume and speed of goods handling, the conventional and outdated warehouse logistics models are no longer able to meet the requirements of today's highly information-driven and

convenient society, especially for small and medium-sized warehouse facilities. However, these warehouse places often encounter difficulties in facility upgrades due to limitations in space and funding. Our team has designed an intelligent warehousing system based on Robei EDA tools, incorporating visual recognition, robotic arm gripping, and Mecanum wheel-driven small vehicles. This system offers multiple interaction methods and achieves adaptability to various scenarios, easy portability, low environmental requirements, and cost-effectiveness. Through systematic validation, the system has met the expected requirements. As time goes on, we will continue to refine the system and further enhance its functionality and performance to meet the ever-evolving warehousing demands.

关键词：Robei EDA 、FPGA、机器人、智能仓储

Keywords: Robei EDA、FPGA、Robot、Intelligent Storage System



一、作品简介	7
1.1 现有仓储设施状况分析	7
1.2 作品设计理念与创新场景	7
1.3 作品创新技术	8
二、架构设计	9
三、控制电路	15
3.1 麦克纳姆轮驱动模块	15
3.1.1 总线指令集解析	17
3.2 舵机控制模块	18
3.3 蓝牙控制模块	20
四、传感系统	21
4.1 IMU 惯性传感模块	21
4.1.1 UART 通信协议	22
4.2 超声波传感器	24
4.3 OV5640 摄像头模块	25
五、机械结构	25
5.1 六自由度机械臂	25
5.2 ov5640 摄像头升降臂	29
5.3 麦克纳姆轮小车底盘	30
六、算法系统	33
6.1 视觉识别算法	33
6.1.1 RGB 格式与 YCbCr 格式换算算法	33
6.1.2 二值化处理	38
6.1.3 边缘检测	39
6.1.4 物块坐标给出	41
6.1.5 LCD 显示屏驱动模块	44
6.2 机械臂逆运动学算法	49
6.2.1 CORDIC 算法	49
6.2.2 机械臂平滑	52

6.3 麦克纳姆轮驱动算法.....	54
6.3.1 麦克纳姆轮逆运动学算法.....	54
6.3.2 Imu 数据解析模块.....	56
6.3.3 速度数值转字符串	61
七、系统整合与调试.....	62
7.1 imu 数据解析仿真	62
7.2 cordic 的代码解释与仿真	64
7.3 视觉仿真	73
八、未来发展与提升.....	76
8.1 IMU 姿态融合空间定位算法	76
8.2 语音控制温湿度传感器	76
九、总结（Conclusion）	77
十、参考资料（References）	78

一、作品简介

1.1 现有仓储设施状况分析

在智能化社会的大背景下，智能化应用的发展已经成为了产业升级和科技振兴的重要方向。而在现有的中小型仓储设施的应用场景如快递收发驿站，医院药房，超市等，普遍还存在着需要大量人力物力进行货品分类，分配，分发的的工作。这些场景的统一特征就是处于物流交互的末端，从前端单一的商家对商家，仓库对仓库变为了末端的，复杂的直接面向用户。面对多时段，多货源，多目标的复杂任务，加上随时面临的以人为基础的各种不确定性因素，导致中小型仓储设施的整体工作效率不足以满足当前快节奏社会的要求。

近年来，伴随着人工智能、传感器等技术的不断发展，市场上早就出现了类似问题的解决方案。这些解决方案通常都是通过购置大型的自动化分拣设备，通过人工对货品进行分类，分区存储，最后才能按照需要的出货组合进行快速的拣取，从本质角度看仍然是一个更大一号的饮料贩卖机。

并且由于昂贵的价格，巨大的空间占用，单一的功能，这些设备难以适应资金不充裕，空间小，且货品需求多种多样的中小型仓储市场，所以市场急需一种成本低，易推广的智能仓储设施。

结合我国对于产业升级，智能化生产生活的要求不断提高的大背景下，智慧型，更为人性化的仓储设施更能完好的解决市场需求。

1.2 作品设计理念与创新场景

我们设计的这款智能仓储系统以一台车为主体，只需要在原有的仓储场景下进行使用，无需更新整套环境设备，环境要求低。一般的仓储设施都是以货架为基础，我们设计的产品只需要对货架位置和高度信息进行建模后即可移植使用整套原设施。

以下是我们系统适用的应用场景：

快递驿站的快件存取，快递入站操作只需要快递员扫描订单号码即可根据快

件的大小来安排对应的柜子，并将柜子信息与快件面单信息进行绑定，发送信息提醒客户进行取件；客户取件时，只需要识别身份信息后，系统即可将所有的与客户身份信息绑定的快件取出并放到取物台上，客户无需等待和寻找即可拿到自己的快件。解决了当前绝大多数快递驿站仍然需要快递员人工投放，用户自主寻找的问题，将快递驿站的空间利用效率极大的提高，避免了因人的失误导致的风险以及拥挤，极大提高了快递驿站的服务效率。

医院药房的快速出药和药品入库，药品入库操作从以往的需要人工摆放到货架上，且需要人工自主计算药品存入数量，而在本系统中只需要给出不同药品对应的不同位置，经过扫描后即可将药品送入指定位置，完全不需要人工进行存放。而药品的出药更为简单，医生开具处方之后，药品组合信息自动进入上位机系统，系统自动分配小车到达对应药品的位置进行抓取药品。统一放置到取药窗口。解决了医务人员因人为因素导致的药品错放，错取，实时准确的获取仓库内药品数量，提高了医院药房的服务效率。

超市货架的摆放，超市每天需要大量的人工进行货品的摆放，从仓库到货架的人工摆放效率低，使用本系统可以节省人工，只需要根据仓库内的货品信息与对应目标货架进行收录，本系统即可完成对货物的从仓库到货架的摆放，并能实时计算仓库内的货品数量进行补货提醒，以及结合收银台给予的商品解算信息实时计算对应商品货架上的冗余情况，对可能缺货的货架进行提前补货。

本系统具有极强的可移植性以及便捷性，便于中小型仓储场景的转型升级，以下以快递驿站为例进行整体技术介绍。

1.3 作品创新技术

Ov5640 摄像头与机械臂融合使用

在本系统中使用 ov5640 摄像头对物块进行坐标识别，给予机械臂物块的坐标，机械臂使用逆运动学进行对应坐标的物块抓取，提升了整体系统的鲁棒性。可以实现在摄像头识别范围内任意位置坐标的机械臂抓取。具有高精度简单高效

的特点。

麦克纳姆轮逆运动学算法

采用自研的麦克纳姆轮逆运动学算法，能够结合 imu 传感器的位置反馈进行实时动作重新规划，本团队运用若贝 EDA 软件在 FPGA 平台上对所有算法与逻辑控制进行硬件电路实现，在功耗和运算速度上表现优秀。

二、架构设计

本作品所设计的智能仓储系统的结构框架如图 4 所示，采用一体化多功能，便捷推广移植的设计思路，在一块 zynq7000 FPGA 开发板上实现整体系统的功能模块以及其他硬件外设的接口模块。

在顶层模块设计中我们根据系统的主要功能需要将系统分为视觉识别，机械臂运动，麦克纳姆轮驱动三大模块，各个模块下属有各自的功能分区。如图 2.6 所示，视觉识别模块中又分为摄像头模块驱动，屏幕显示模块驱动，超声波模块驱动等。如图 2.5 所示，机械臂模块中又分为逆运动学模块，cordic 函数计算模块等。如图 2.7 所示，麦克纳姆轮驱动模块中又分为 imu 数据解析模块，串口数据接收模块，蓝牙串口接收模块等。

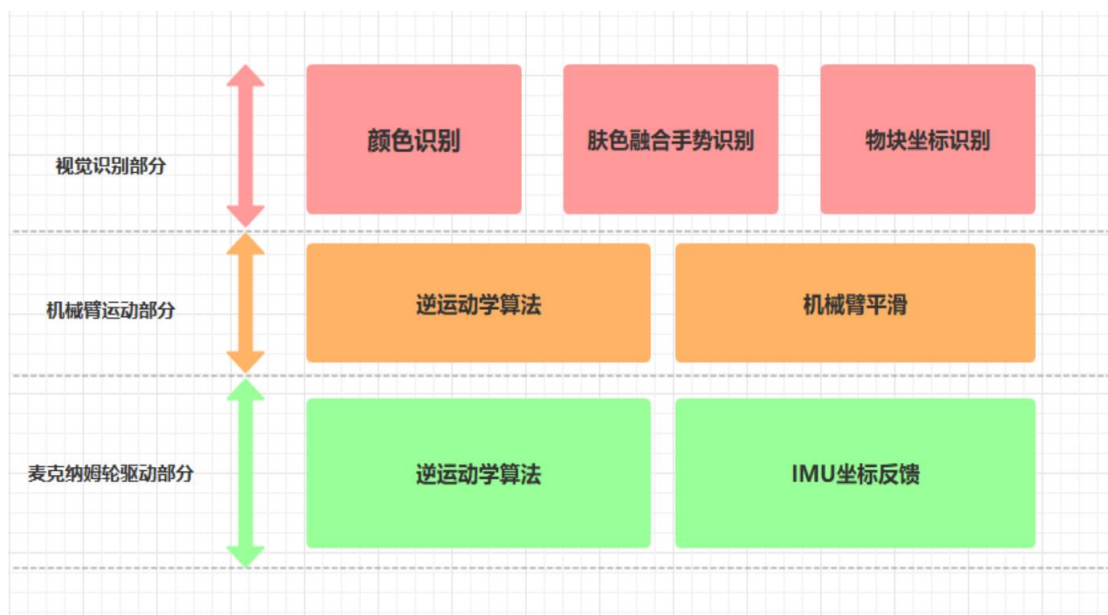


图 2.1 顶层架构图

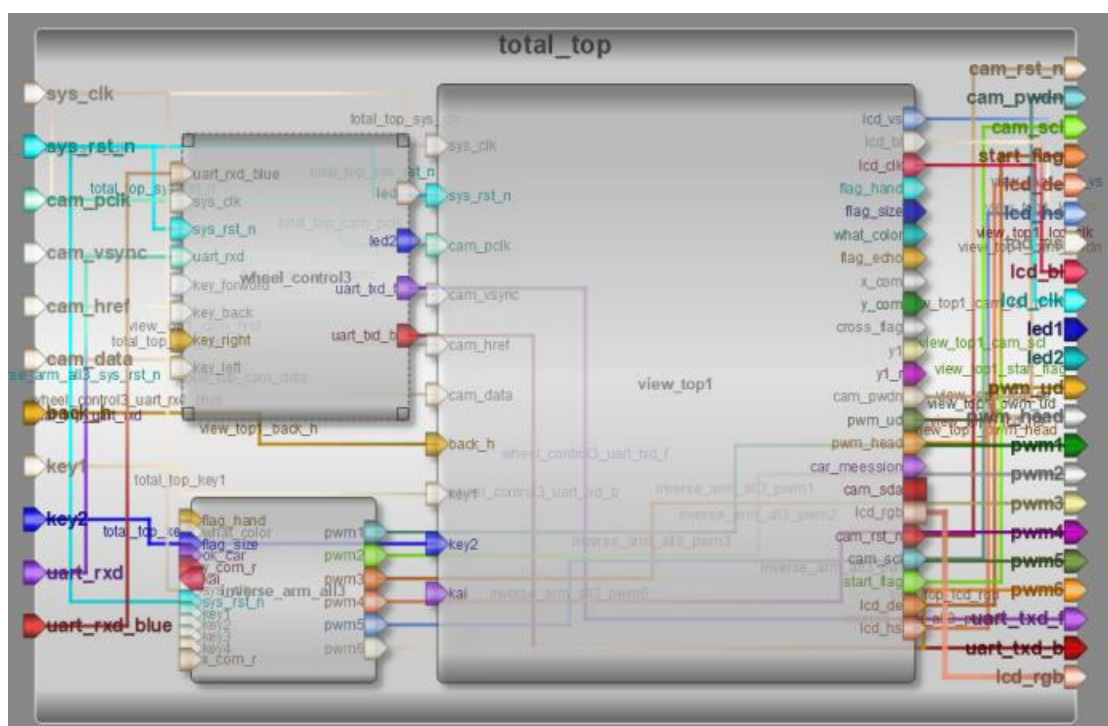


图 2.2 Robei EDA 顶层模块图



Name	Inout	DataType	Datasize	Function
sys_clk	input	wire	1	
sys_rst_n	input	wire	1	
cam_pclk	input	wire	1	
cam_vsync	input	wire	1	
cam_href	input	wire	1	
cam_data	input	wire	8	
back_h	input	wire	1	
key1	input	wire	1	
key2	input	wire	1	
uart_rxd	input	wire	1	
uart_tx_dblue	input	wire	1	
cam_rst_n	output	wire	1	
cam_pwdn	output	wire	1	
cam_scl	output	wire	1	
start_flag	output	wire	1	
lcd_de	output	wire	1	
lcd_hs	output	wire	1	
lcd_vs	output	wire	1	
lcd_bi	output	wire	1	
lcd_clk	output	wire	1	
led1	output	reg	1	
led2	output	reg	1	
pwm_ud	output	wire	1	
pwm_head	output	wire	1	
pwm1	output	wire	1	
pwm2	output	wire	1	
pwm3	output	wire	1	
pwm4	output	wire	1	
pwm5	output	wire	1	
pwm6	output	wire	1	
uart_brd_f	output	wire	1	
uart_tx_b	output	wire	1	
lcd_rgb	output	wire	24	

图 2.3: 顶层模块 Robei EDA 端口定义

下面是本系统实现货品存储收发作业的整体流程：

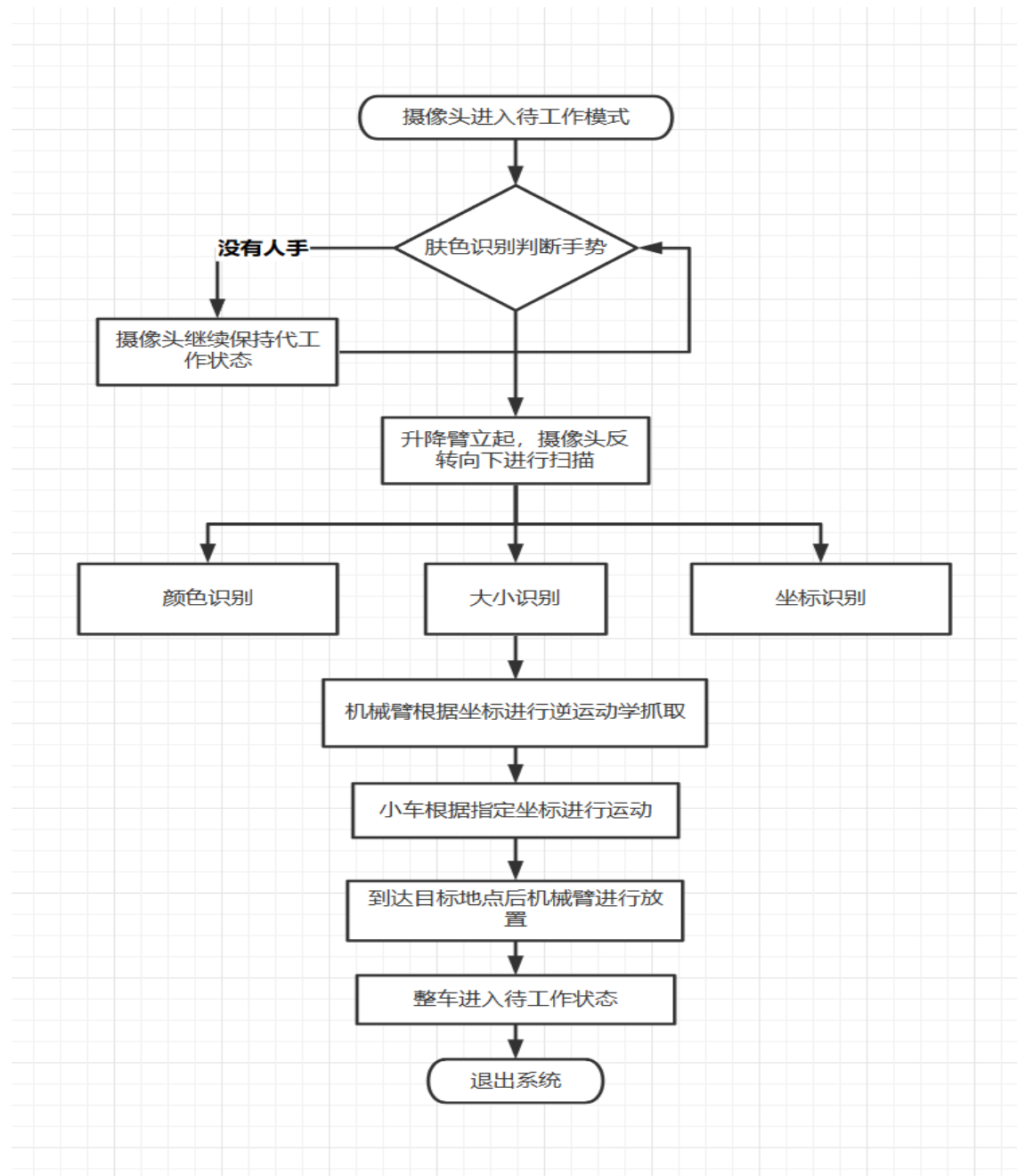


图 2.4：整体工作流程

首先，当没有物品放置或者取出的工作要求时，整车自动进入待工作状态。摄像头斜向上立起，等待唤醒。

当工作人员需要进行货物的投放存储时，只需要将货物放在摄像头前方识物平台上，用手在摄像头前轻轻一扫。摄像头经过肤色识别检测到人机交互之后会

自动抬升并翻转向下对准识物台，并对物品进行颜色，大小和坐标的识别。区分出物块的信息特征之后将对应的目标地点发送给小车，将物块在识物台上的位置发送给机械臂。物块抓取之前，摄像头自动进入蜷缩模式，此时机械臂根据摄像头给出的坐标进行抓取。物块抓取之后，摄像头再进入待工作状态，同时小车会根据预设的目标地点进行运动，运动至对应的柜子。并停下等待机械臂的投放。在收到小车已经到达目标地点的信号之后，机械臂按照预设的投放柜子高度进行投放。整套物体存储过程完成。

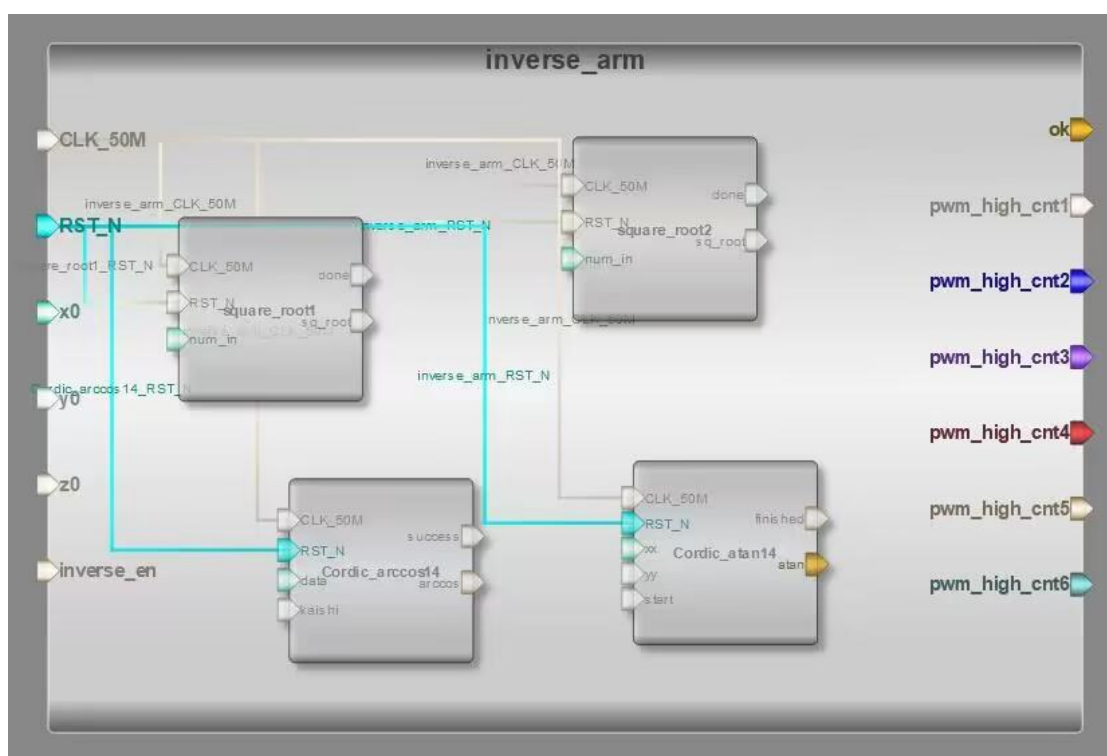


图 2.5: 机械臂模块 RoboEEDA 实现

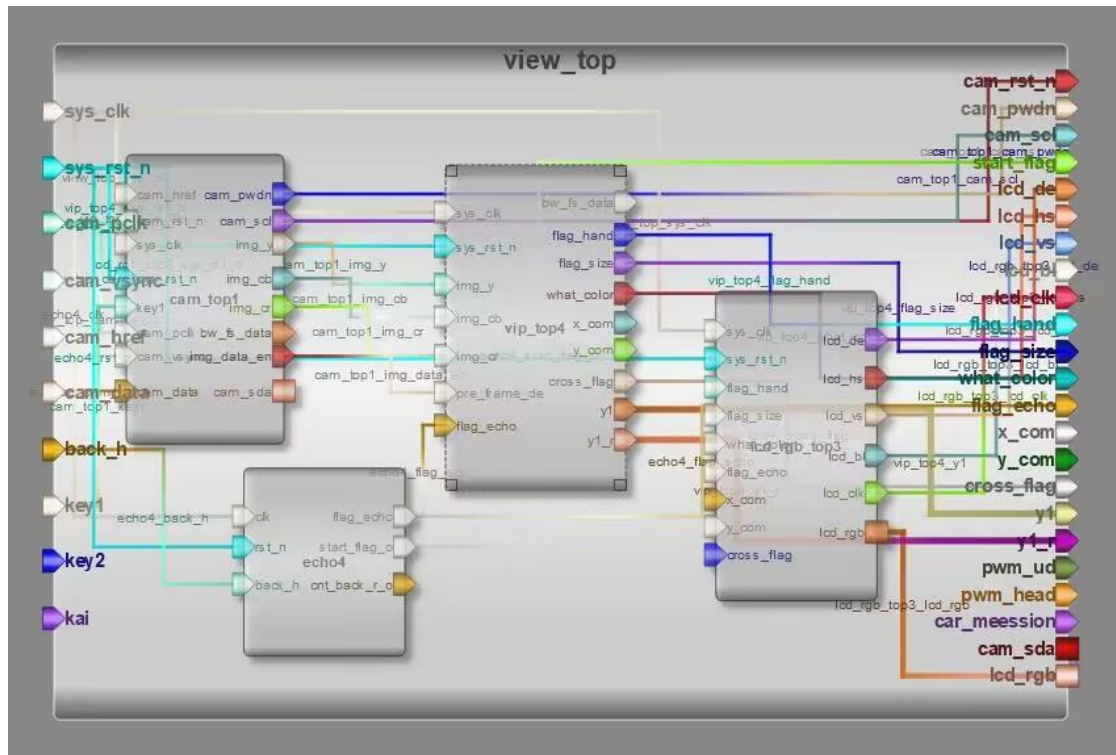


图 2.6: 视觉模块 Robei EDA 实现

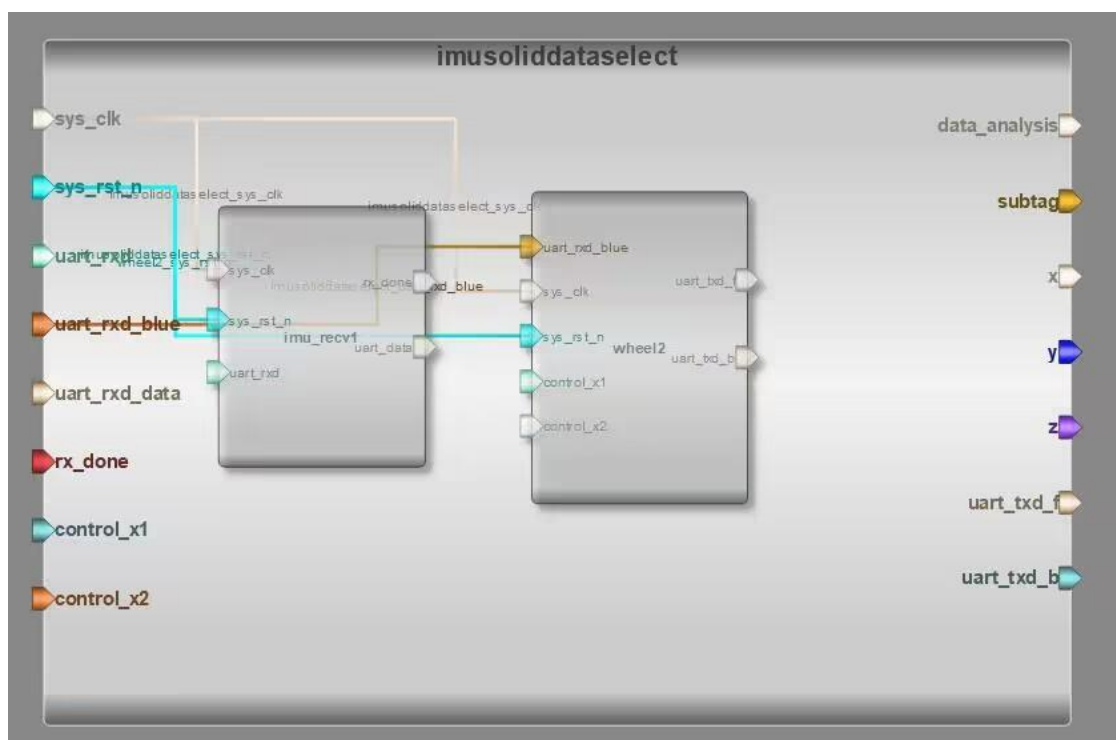


图 2.7: 麦克纳姆轮驱动模块部分 Robei EDA 实现

三、控制电路

3.1 麦克纳姆轮驱动模块

麦克纳姆轮小车模块如图 3.1.1 所示，该模块用于控制总线电机，将上层逆运动学计算部分输出的向量转化为对应电机转速，并将对应电机速度转为相应的字符串进行输出，由于总线电机用 UART 进行传输数据所以需要把相应的 PWM 信号转换为总线所识别的数据。



图3.1.1麦克纳姆轮直接驱动模块



图 3.1.2 总线 370 马达



图 3.1.3 总线 370 马达接口

普通直流电机一般采用 PWM (Pulse Width Modulation)即脉冲宽度调制控制转速外加 L298N 电机驱动控制电机转向,本次设计使用的总线 370 电机，不仅能单独使用 PWM 信号控制车速，而且能通过上位机采用 UART 协议传输转动信号从而实现对电机转速驱动时间控制。

如图 3.1.2 所示为总线 370 马达整体外观以及图 3.1.3 为电机自带接口。

总线转接板 Z-link 是将 TTL-ZX-USB 三种通讯方式集成在一起的转接板，是一块便于总线和控制系统即单片机通讯，总线与 USB 通信的控制板。

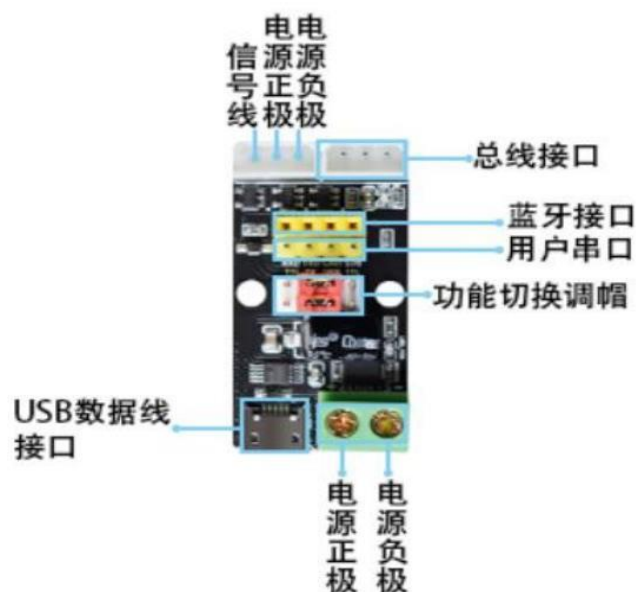


图 3.1.4 总线 Z-link 接口

如图 3.1.4 所示为 Z-link 接口，通过总线接口连接电机传输。实现控制顺序：开发板 -> Z-link -> 总线 370 电机。该串口总线模块自带 CH340 驱动芯片，将跳线帽切换到 ZX-USB 模式，再将 USB 接口连接到电脑，总线连接电机就可以实现电脑对电机的驱动控制，并通过这种方式实现对总线电机内部指令的修改。同时该模块能实现从电源模块到电机缓冲作用，由此处连接外部电源再通过总线实现对电机的供电，这样避免了电源直接与开发板相连，防止由于电压过高或者操作失误从而导致开发板损坏。通过 FPGA 控制总线时只需要将跳线帽转接到 TTL-USB 模式，再将用户串口中的 GND 和 RX 接口连接到开发板 GND 端和模拟 UART 输出 IO 口。

3.1.1 总线指令集解析

电机控制指令：比如#000P1500T1000!

“#”后面的三位数字为总线电机的地址

“P”后面的四位数字为总线电机的转速

“T”后面的四位数字为总线电机的旋转时间

“！”为整条指令的截止位

整条指令是由一个字符串数组组成，每个字符占用的位宽为八

1) id 相当于每个总线电机的“名字”，其范围是 000~254，必须为三位数，不足的位数补 0，255 为广播 ID，所有设备都会响应这个指令；

2) pwm 的范围是 0500~2500，必须为四位数，不足的位数补 0，1500 表示停止，大于 1500 为正转，数值越大转速越快，小于 1500 为反转，数值越小反向转速越快；

3) Time 表示旋转时间(单位 s)，必须为四位数范围：0000-9999，特殊的 0 代表循环执行；

3.2 舵机控制模块

舵机（英文叫 Servo）：它由直流电机、减速齿轮组、传感器和控制电路组成的一套自动控制系统。通过发送信号，指定输出轴旋转角度。舵机一般而言都有最大旋转角度(比如 180 度和 270 度。)我们的机械臂上配置的舵机就是 270°舵机。



图 3.2.1舵机图片

舵机的伺服系统由可变宽度的脉冲来进行控制，控制线是用来传送脉冲的。脉冲的参数有最小值，最大值，和频率。一般而言，舵机的基准信号都是周期为

20ms，宽度为 1.5ms。这个基准信号定义的位置为中间位置。舵机有最大转动角度，中间位置的定义就是从这个位置到最大角度与最小角度的量完全一样。最重要的一点是，不同舵机的最大转动角度可能不相同，但是其中间位置的脉冲宽度是一定的，那就是 1.5ms。如下图：

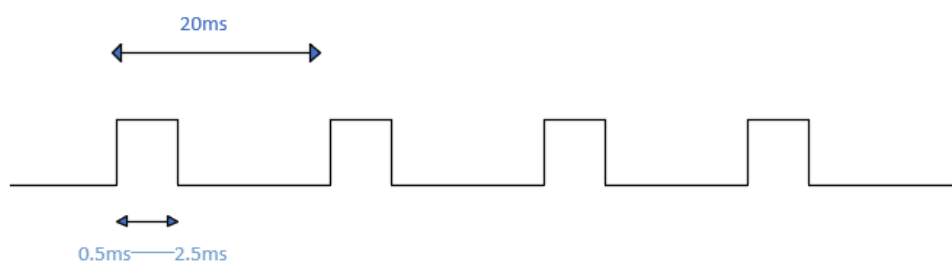


图 3.2.2 脉冲调制图

舵机：角度是由来自控制线的持续的脉冲所产生。这种控制方法叫做脉冲调制。脉冲的长短决定舵机转动多大角度。例如：1.5 毫秒脉冲会到转动到中间位置（对于 180° 舵机来说，就是 90° 位置，对于 270° 舵机来说，就是 135° 位置）。当控制系统发出指令，让舵机移动到某一位置，并让他保持这个角度，这时外力的影响不会让他角度产生变化，但是这个是由上限的，上限就是他的最大扭力。除非控制系统不停的发出脉冲稳定舵机的角度，舵机的角度不会一直不变。

当舵机接收到一个小于 1.5ms 的脉冲，输出轴会以中间位置为标准，逆时针旋转一定角度。接收到的脉冲大于 1.5ms 情况相反。不同品牌，甚至同一品牌的不同舵机，都会有不同的最大值和最小值。一般而言，最小脉冲为 1ms，最大脉冲为 2ms。如下图：

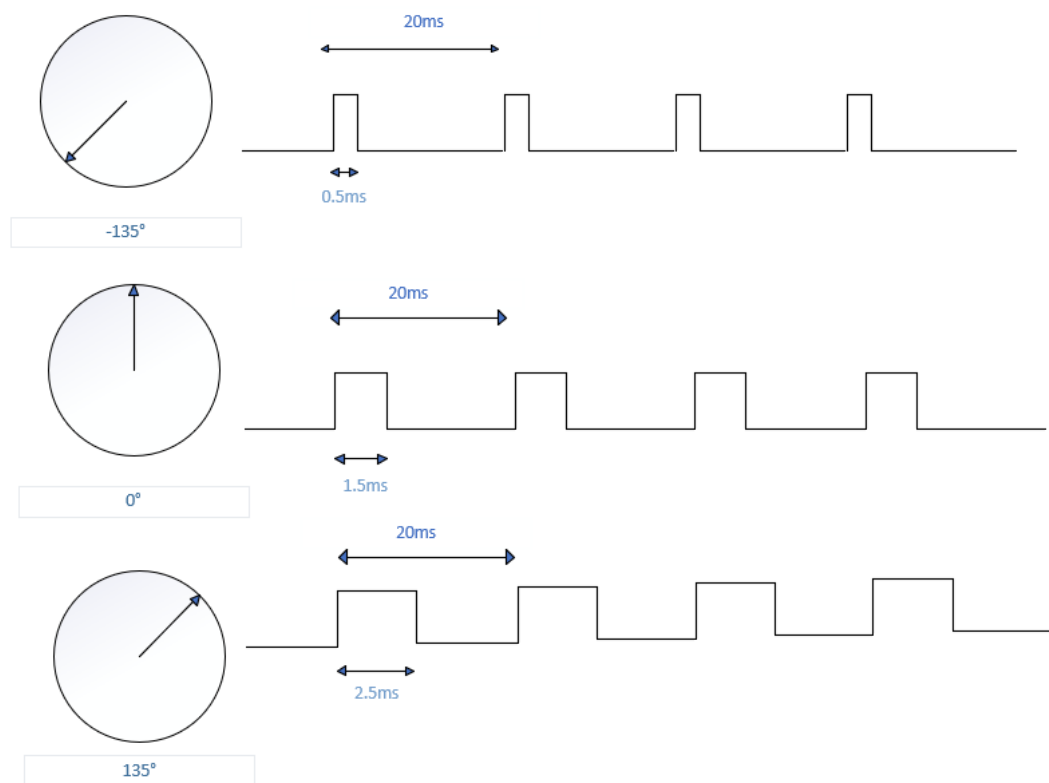


图3.2.3脉冲调制对应角度图

3.3 蓝牙控制模块

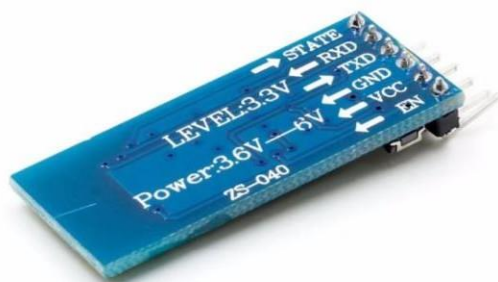


图 3.3.1 HC-05 蓝牙模块实物图

ATK-HC05 蓝牙串口模块是一款高性能的主从一体蓝牙串口模块,可以同各种带蓝牙功能的电脑、蓝牙主机、手机、PDA、PSP 等智能终端配对。该模块支持非常宽的波特率范围: 4800~1382400, 并且模块兼容 5V 或 3.3V 单片机系统, 可以很方便地进行连接。ATK-HC05 蓝牙模块通过串口与 FPGA 进行通信, 完成数据的发送和接收。

该模块用于实现麦克纳姆轮小车的远程控制与数据传输功能。其驱动协议为 uart 协议, 波特率为 9600bps 并将该波特率与 370 总线电机保持一致实现时序同步, 1 位起始位, 8 位数据位, 1 位停止位, 无奇偶校验位。上位机采用 16 进制数据进行传输, 通过手机蓝牙控制从而实现对麦克纳姆轮小车方向驱动控制。

蓝牙模块如图 3.1.1 所示, 用来控制麦克纳姆轮小车的运动, 通过上位机设定输出数据从而实现麦克纳姆轮的前进、倒退、左转、右转、顺时针旋转、逆时针旋转。

四、传感系统

4.1 IMU 惯性传感模块

IMU 全称 Inertial Measurement Unit, 惯性测量单元, 主要用来检测和测量加速度与旋转运动的传感器。其原理是采用惯性定律实现的, 这些传感器从超小型的 MEMS 传感器, 到测量精度非常高的激光陀螺, 无论尺寸只有几个毫米的 MEMS 传感器, 到直径几近半米的光纤器件采用的都是这一原理。

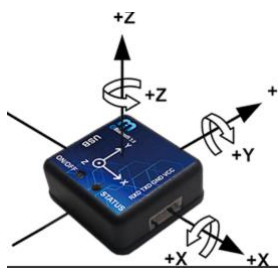


图4.1.1 imu实物示意图

最基础的惯性传感器包括加速度计和角速度计（陀螺仪），他们是惯性系统

的核心部件，是影响惯性系统性能的主要因素。尤其是陀螺仪其漂移对惯导系统的位置误差增长的影响是时间的三次方函数。而高精度的陀螺仪制造困难，成本高昂。因此提高陀螺仪的精度、同时降低其成本也是当前追求的目标。

本次采用的 imu 传感器为辰翼电子科技的 im948 十轴姿态传感器，是一款兼容蓝牙和串口的小尺寸十轴姿态传感器模块。内部集成了 ARM 32 位 DSP 处理器、BLE5.0 蓝牙、高精度加速计、陀螺仪、磁力计、温度气压计。采用自主研发的姿态解算算法，强实时、高精度、稳定不漂移。内置充电管理，超低功耗技术，并支持丰富的功能扩展。工艺上采用 4 层阻抗板，沉金邮票孔接口。同时使用蓝牙 5.0 无线技术，支持 Android、ios、windows、Linux 系统以及其它蓝牙设备连接使用。

典型应用原理图

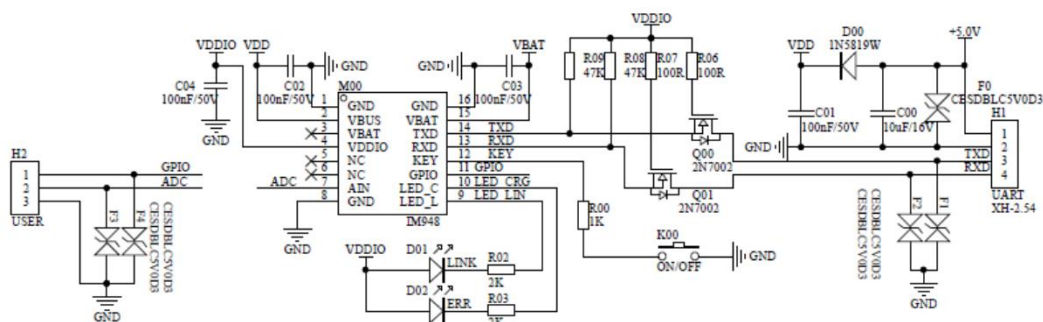


图4.1.2 im948典型应用电路

本模块可以实现的功能有温度，气压，高度，三轴角速度，三轴磁场，包含重力的三轴加速度，四元数，欧拉角，去除重力的三轴加速度，导航系加速度和三维空间位置。

通信方式上支持蓝牙和串口两种通信模式，采用 uart 串口通信协议。

4.1.1 UART 通信协议

UART（Universal Asynchronous Receiver-Transmitter）串口通信是一种常用的串行通信协议，广泛应用于计算机和嵌入式系统中。它是一种简单、可靠的通信方式，用于在设备之间传输数据。

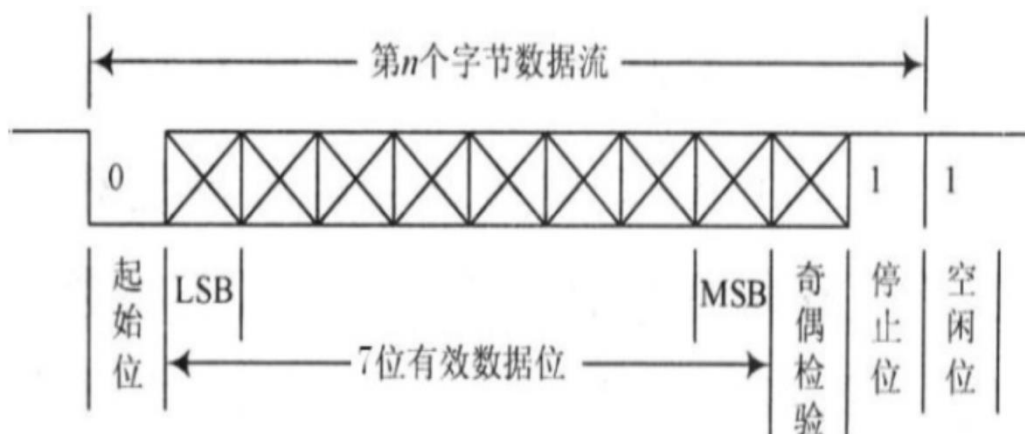


图4.1.3 uart通信数据帧结构

UART 串口通信使用两根信号线，分别是发送线（TX）和接收线（RX），用于双向的数据传输。发送线负责将数据从发送方发送到接收方，接收线负责将数据从接收方传输回发送方。

UART 的数据以数据帧的形式传输，每个数据帧由起始位、数据位、校验位和停止位组成。起始位和停止位用于标识数据帧的开始和结束，数据位用于传输实际的数据，校验位用于验证数据的准确性。使用波特率（Baud Rate）来表示数据传输的速率，即单位时间内传输的比特数。常见的波特率包括 9600、115200 等。发送方和接收方必须使用相同的波特率才能正确地进行数据传输。



图4.1.4 IMU串口接收的Robei EDA 模块图

4.2 超声波传感器

超声波模块采用 HC_SR04，该模块探测距离在 2cm—600cm 之间，感应角度小于 15度，探测精度在 0.1cm 左右(实际使用效果受探测物表面平滑度影响)，输出为 IO 输出。

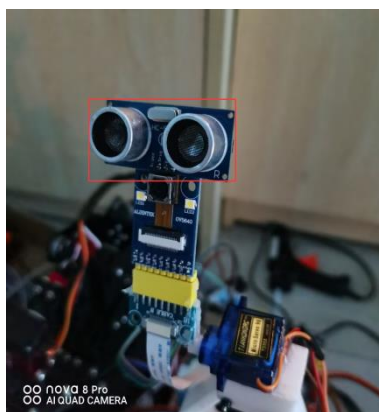


图4.2.1 超声波模块图片

4.3 OV5640 摄像头模块

摄像头采用豪威科技生产的 ov5640 摄像头，像素可达 500w，该模块使用豪威公司自己的 SCCB 通信协议，而标准 SCCB 接口兼容 IIC 接口，所以我们使用 IIC 协议也可以对其进行开发。模块自带两颗高亮闪光灯，可以在低光照的场景下进行补光。然而实际测试使用时发现光照过强，会造成物体表面反光等状况，从而影响识别精度，所以我们没有使用模块自带 led。

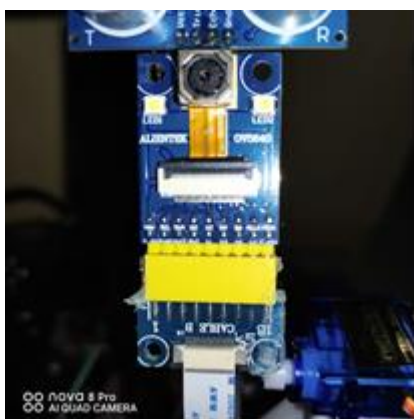


图4.2.2 ov5640摄像头模块图片

五、机械结构

5.1 六自由度机械臂

机械臂：

对于我们所畅想的应用场景，我们决定采用一款六自由度机械臂来实现功能。

六轴机械臂是一种具有六个自由度的机械臂系统，它在工业和科研领域中被广泛应用。以下是六轴机械臂的一些主要好处：

- 多自由度：六轴机械臂具有六个独立自由度，可以在三维空间内进行灵活的运动和操作。这使得它能够完成更加复杂和多样化的任务，如装配、搬运、焊接等。

- **高精度和重复性：**六轴机械臂采用精密的传感器和控制系统，能够实现高精度的运动和定位。它可以精确地控制末端执行器的位置和姿态，保证任务的准确性和稳定性。

- **载重能力：**六轴机械臂通常具有较高的载重能力，可以承担重量较大的物体。这使得它在物流、生产线和仓储等领域中可以进行大规模的物体搬运和装配工作。

- **灵活性和可编程性：**六轴机械臂的动作和路径可以通过编程进行灵活控制。它可以根据不同的任务需求进行快速调整和重新编程，适应不同的工作场景和产品变化。

- **自动化和提高生产效率：**六轴机械臂能够自动执行任务，减少了人力投入，提高了生产效率和工作质量。它可以在不间断的工作环境下连续运行，实现高速、高效的流水线操作。

- **人机协作和安全性：**六轴机械臂可以与人员进行安全的协作，实现人机共同操作。通过搭载传感器和智能控制系统，它能够感知周围环境和人员的存在，并做出相应的反应，确保工作的安全性和人员的安全。

它在提高生产效率、减少劳动力成本、改善工作环境和提升产品质量等方面发挥着重要作用，被广泛应用于各个领域。

我们采用的机械臂由散件组装，根据本小组所演算的公式，组装成能够得到契合公式的舵机角度的机械臂。



图5.1.1拼装前零件图



图 5.1.2机械臂实物图

5.2 ov5640 摄像头升降臂

摄像头固定采用自行 3D 打印的 X 型升降台结构，让摄像头可以脱离固定位置带来的一些弊端，比如只放在高处容易在车体行进过程中发生大幅晃动从而影响整体系统稳定性，只放置在低处会导致识别范围太小，且容易在机械臂抓取时相互阻碍，造成结构性损伤。因此引入可升降机械结构，在识别物块时升起，高位观察，识别结束后，折叠摄像头，方便机械臂抓取以及提高车体行进稳定性。

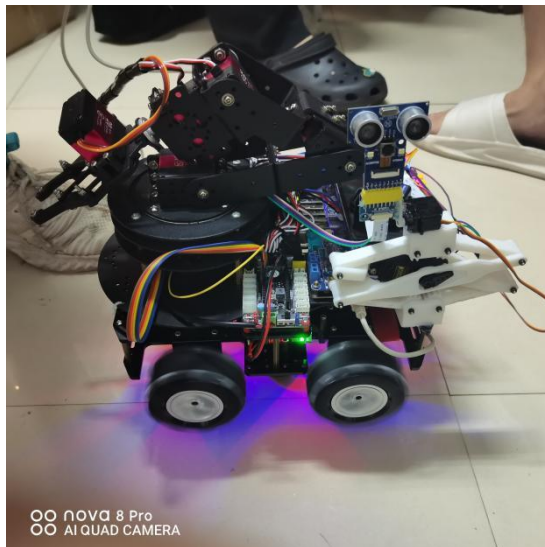


图5.2.1 升降台折叠状态图

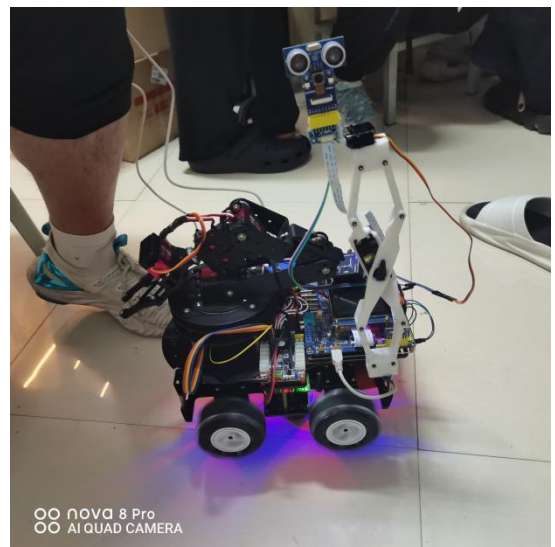


图5.2.2 升降台升起状态图

5.3 麦克纳姆轮小车底盘



图 5.3.1 麦克纳姆轮小车

麦克纳姆轮是一种可以全向移动的全向轮，又叫麦轮，由轮毂和围绕轮毂的辊子组成，麦轮辊子轴线和轮毂轴线夹角成 45° 。在轮毂的轮缘上斜向分布着许多小轮子，即辊子，故轮子可以横向滑移。辊子是一种没有动力的小滚子，小滚子的母线很特殊，当轮子绕着固定的轮心轴转动时，各个小滚子的包络线为圆柱面，所以该轮子能够连续的向前滚动。由四个这种轮加以组合，可以使机构实现全方位移动的功能。麦克纳姆轮根据镜像关系分为 A 轮和 B 轮，如图 5.3.4 所示：

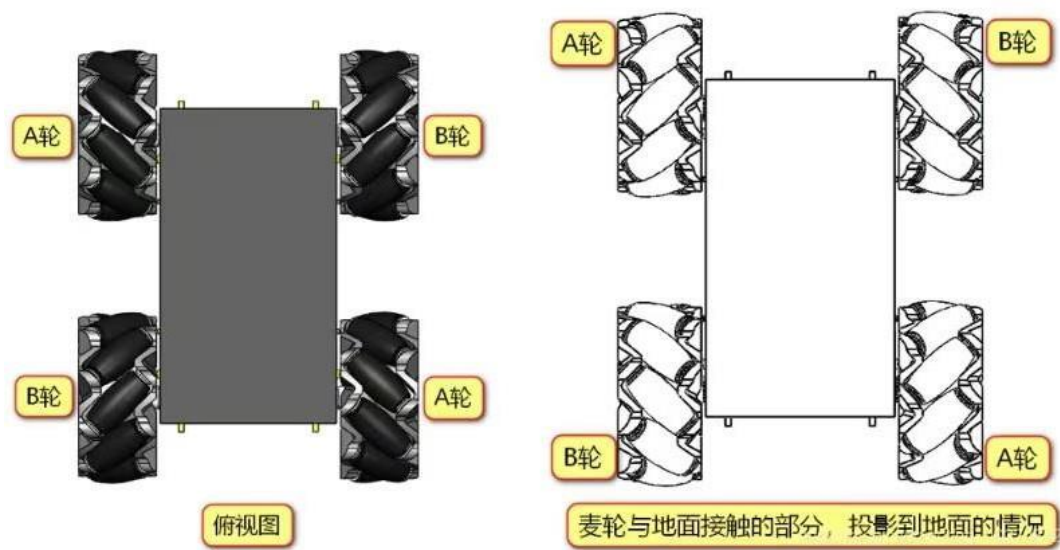


图 5.3.2 麦克纳姆轮示意图

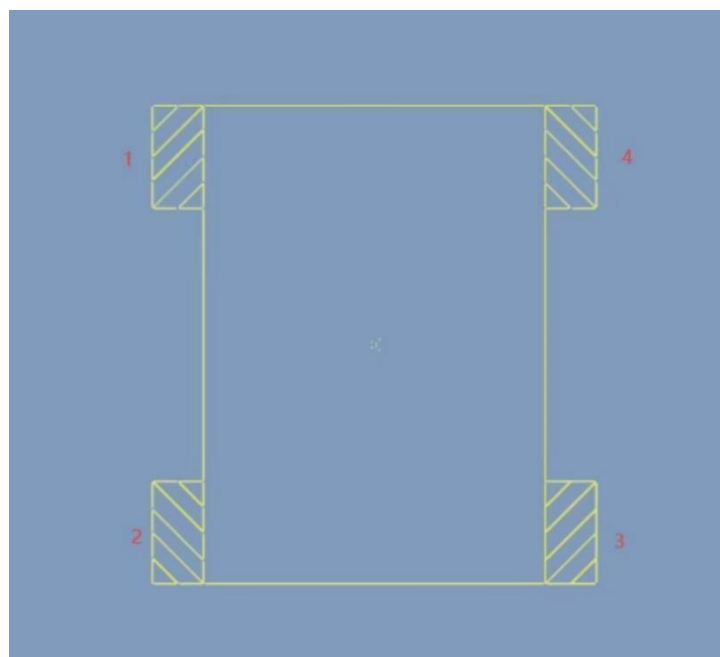


图 5.3.3 轮子标记

设四个轮子分别为 1 号 2 号 3 号 4 号，如图 3.2.3 所示：

下面红色箭头表示车轮的运动正反方向。

当小车向前运动时：1 号、2 号、3 号、4 号车轮都正方向转动，小车前进。

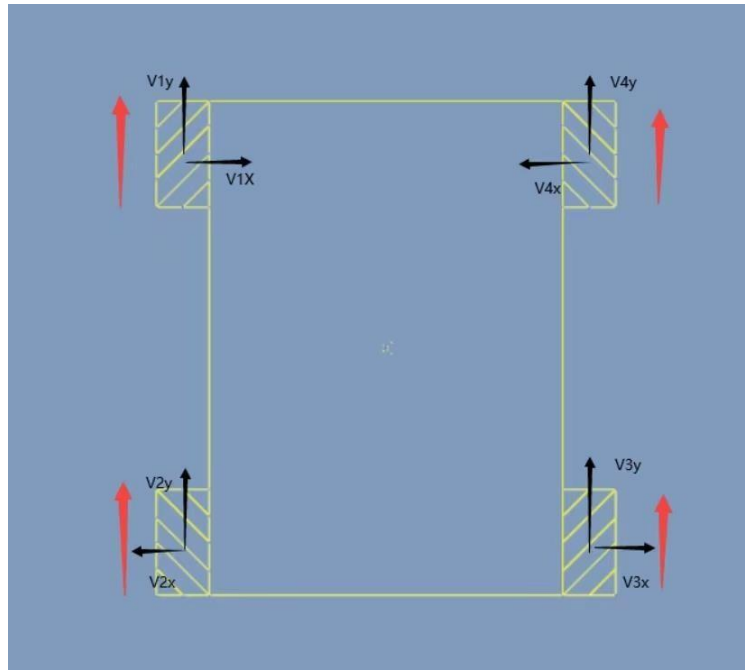


图 5.3.4 轮子转动方向

当小车向右运动时，1 号、3 号车轮正方向转动，2 号、4 号车轮反方向转动。

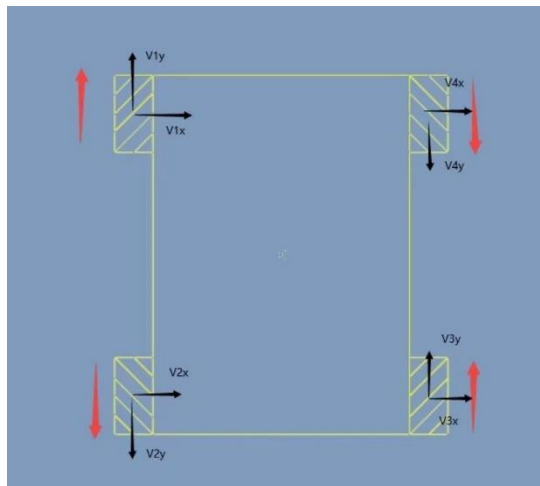


图 5.3.5 轮子转动方向

当小车顺时针转动时，1 号、2 号车轮正方向转动，3 号、4 号车轮反方向转动。

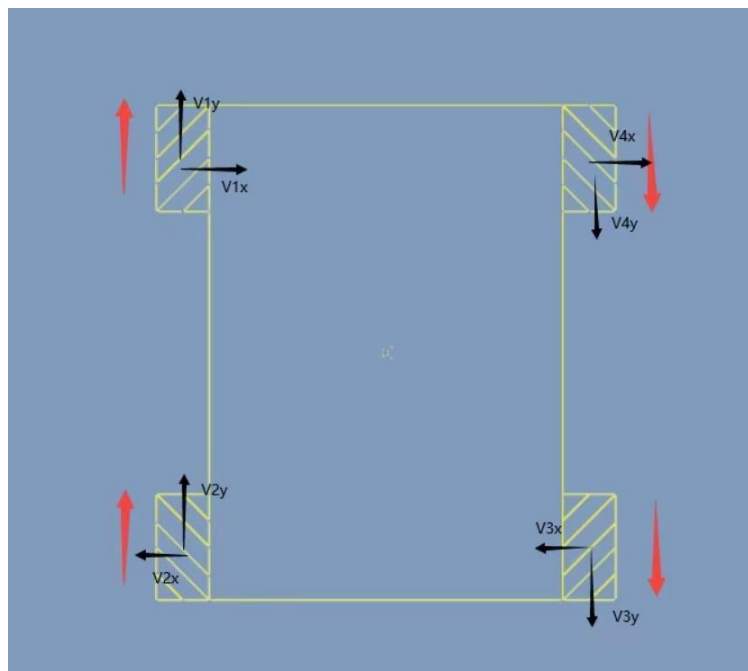


图5.3.6 轮子转动方向

六、算法系统

6.1 视觉识别算法

6.1.1 RGB 格式与 YCbCr 格式换算算法

RGB 颜色空间是最常见的颜色空间，基于 RGB 三原色的不通亮度掺杂，可以显示出最多 16 万种颜色，然而 RGB 三色之间存在高相关性，直接对 RGB 数据处理会导致误差很大，对应的，YCbCr 颜色空间在分割 RGB 颜色空间中难以分离的对象时有优势，所以我们将其转换为 YCbCr 格式以待后续处理。

转换标准公式为：

$$Y = 0.257R + 0.564G + 0.098B + 16$$

$$Cb = -0.148 R - 0.291G + 0.439 B + 128$$

$$Cr = 0.439R - 0.368 G - 0.071* B + 128$$

当然，标准公式是面对 RGB888 格式的，而我们摄像头输出的是 RGB565 格式，所以要进行先
前处理，将 RGB565 格式转换为 RGB888 格式。

```
//RGB565 转 RGB 888  
assign rgb888_r   = {img_red , img_red[4:2] };  
assign rgb888_g   = {img_green, img_green[5:4]};  
assign rgb888_b   = {img_blue , img_blue[4:2] };
```

图6.1.1 rgb565转rgb888核心程序

```

/*****
  RGB888 转 YCbCr
  Y = 0.299R + 0.587G + 0.114B
  Cb = 0.568(B-Y) + 128 = -0.172R-0.339G + 0.511B + 128
  CR = 0.713(R-Y) + 128 = 0.511R-0.428G -0.083B + 128

  Y = (77 *R   + 150 *G   + 29 *B)>>8
  Cb = (-43 *R  - 85 *G   + 128 *B)>>8 + 128
  Cr = (128 *R  - 107 *G  - 21 *B)>>8 + 128

  Y = (77 *R   + 150 *G   + 29 *B   )>>8
  Cb = (-43 *R  - 85 *G   + 128 *B + 32768)>>8
  Cr = (128 *R  - 107 *G  - 21 *B + 32768)>>8
*****/

```

图6.1.2 rgb转YCbCr核心程序示例

由于 fpga 无法进行浮点运算，所以不能像写软件编程一样直接整式写入，我们需要先将式子右端整体扩大 256 倍，然后右移 8 位，这样就可以进行 fpga 擅长的乘法和加法运算了。

并且我们利用 fpga 的高并行运算特点，引入三级流水线设计思想，以 Y 值计算为例，用一个时钟计算括号内的每个乘积，再用一个时钟计算三个值的和，最后用一个时钟右移 8 位就得到了 Y 值。流水线运算每一级运算相差一个时钟，每一级都在不断运算，这样的流水线运算可以将运算速度提高三倍。

```

always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        rgb_r_m0 <= 16'd0;
        rgb_r_m1 <= 16'd0;
        rgb_r_m2 <= 16'd0;
        rgb_g_m0 <= 16'd0;
        rgb_g_m1 <= 16'd0;
        rgb_g_m2 <= 16'd0;
        rgb_b_m0 <= 16'd0;
        rgb_b_m1 <= 16'd0;
        rgb_b_m2 <= 16'd0;
    end
    else begin
        rgb_r_m0 <= rgb888_r * 8'd77 ;
        rgb_r_m1 <= rgb888_r * 8'd43 ;
        rgb_r_m2 <= rgb888_r << 3'd7 ;
        rgb_g_m0 <= rgb888_g * 8'd150;
        rgb_g_m1 <= rgb888_g * 8'd85 ;
        rgb_g_m2 <= rgb888_g * 8'd107;
        rgb_b_m0 <= rgb888_b * 8'd29 ;
        rgb_b_m1 <= rgb888_b << 3'd7 ;
        rgb_b_m2 <= rgb888_b * 8'd21 ;
    end
end

always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        img_y0 <= 16'd0;
        img_cb0 <= 16'd0;
        img_cr0 <= 16'd0;
    end
    else begin
        img_y0 <= rgb_r_m0 + rgb_g_m0 + rgb_b_m0;
        img_cb0 <= rgb_b_m1 - rgb_r_m1 - rgb_g_m1 + 16'd32768;
        img_cr0 <= rgb_r_m2 - rgb_g_m2 - rgb_b_m2 + 16'd32768;
    end
end

always @(posedge clk or negedge rst_n) begin
    if(!rst_n) begin
        img_y1 <= 8'd0;
        img_cb1 <= 8'd0;
        img_cr1 <= 8'd0;
    end
    else begin
        img_y1 <= img_y0 [15:8];
        img_cb1 <= img_cb0[15:8];
        img_cr1 <= img_cr0[15:8];
    end
end

```

图6.1.3 三级流水线实现rgb转YCbCr具体程序

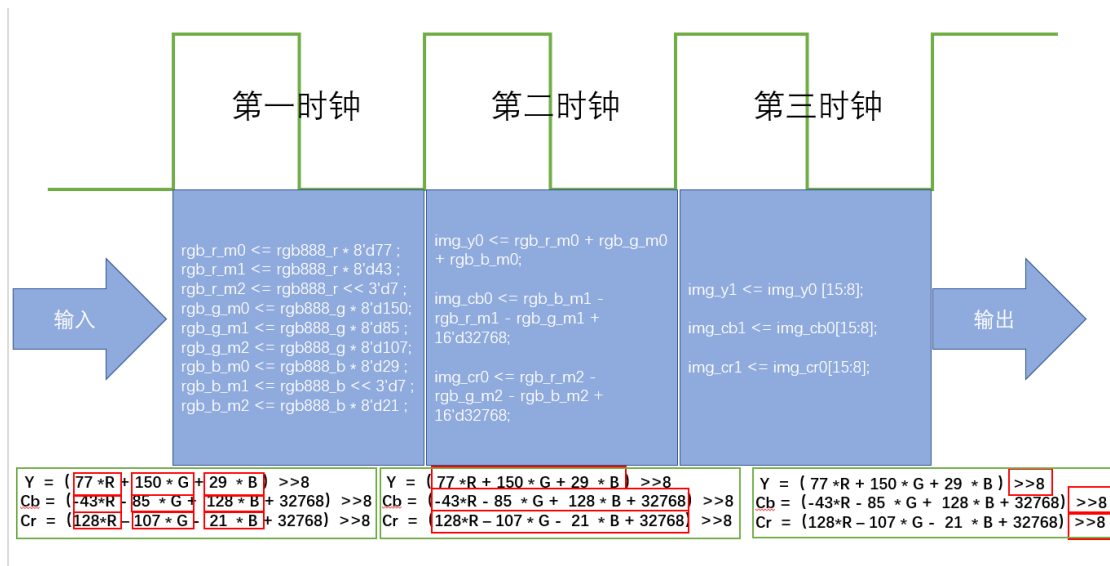
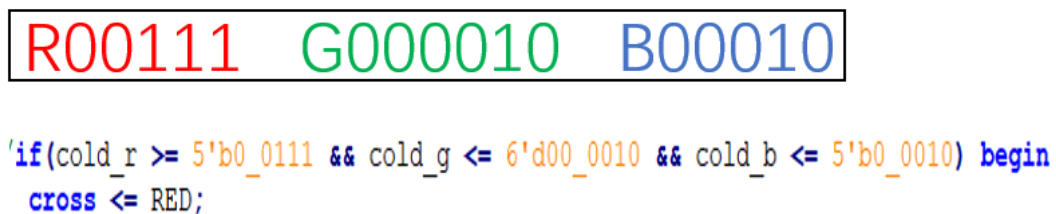


图6.1.4 三级流水线示意图

在最初的设计中，为了效率，设想用更快的速度去实现功能，直接用 RGB 颜

色空间去区分颜色和二值化图像，但颜色识别会很容易有误差，RGB 是从颜色发光的原理来设计定的，它的颜色混合方式为三个独立的单色光想混合，色彩相混，而亮度却等于两者亮度之总和，越混合亮度越高，即加法混合。

红、绿、蓝三个颜色通道每种色各分为 256 阶亮度，在 0 时该颜色最弱，是关掉的，而在 255 时最亮。当三色灰度数值相同时，产生不同灰度值的灰色调，即三色灰度都为 0 时，是最暗的黑色调；三色灰度都为 255 时，是最亮的白色调。虽然在判断条件中加入另外两种颜色的限制，但是识别效果依旧不尽人意，且在最强白光时，会出现颜色识别混乱。最终不得不改变策略。



```

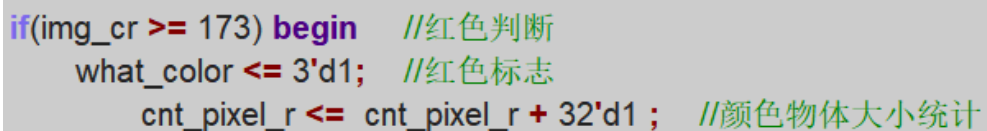
R00111  G000010  B00010

`if(cold_r >= 5'b0_0111 && cold_g <= 6'd00_0010 && cold_b <= 5'b0_0010) begin
  cross <= RED;

```

图6.1.5 基于rgb565颜色空间实现的颜色识别

对直接获取的 RGB 数据不经过任何多余处理直接识别并没有带来过多的效率提升，所以在经过抉择后使用了 YCbCr 颜色空间进行颜色区分。修改后的方案不仅颜色识别变得十分准确，二值化的噪点也有所降低。



```

if(img_cr >= 173) begin //红色判断
  what_color <= 3'd1; //红色标志
  cnt_pixel_r <= cnt_pixel_r + 32'd1 ; //颜色物体大小统计

```

图6.1.6 优化后基于YCbCr颜色空间的颜色识别核心代码

6.1.2 二值化处理

二值化是后续视觉识别的一项不可少的操作。通过二值化可以剔除大部分不需要的杂物,从而减少影响。二值化处理有两种方式,一种是直接设定固定阈值,每次修改只适用于特定需要,另一种是自适应阈值将像素点的像素值与该点所在区域的像素平均值进行比较,来决定究竟是黑还是白,当然,这个区域的产生是需要 IP 核的,比如 3*3 矩阵,出于遵守出题方的规则,所以我们的选择只有前者,通过多次调整阈值达到合适的位置,来尽量减少噪声,判断标准为是否会大面积识别错误,如图 x.x,背景会有很明显的深色误判,在修改后如图 x.x,虽然对手势本身识别效果也有所减弱,但是背景的误判几乎可以消除,此时认为手势识别几乎不会受到异物影响,阈值效果为优。同时,由于 fpga 的可重构性,在后期拓展时,如果需要对其他单一颜色的物体进行区分,只需要改动合适的阈值即可。

这里 if 的判断阈值即是应用 YCbCr 的分量数据,以肤色为例,在经过阈值调整后,可以使得实际显示的图像只有与人体肤色有关,可以做到人体检测功能。

```
//颜色数据属于范围内为白, 否则为黑
if ((img_cb > 77) && (img_cb < 127) && (img_cr > 133) && (img_cr < 173)) begin
    bw_data <= WHITE;
end
else begin
    bw_data <= BLACK;
end
```

图6.1.7 二值化核心程序



图 6.1.8 肤色识别修改前



图 6.1.9 肤色识别修后

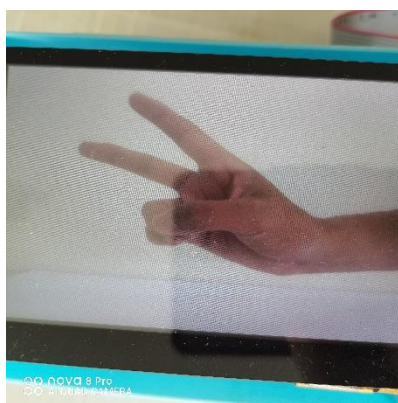


图6.1.10 手势识别原图

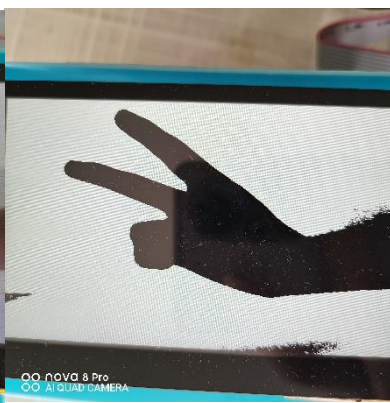


图6.1.11二值化效果图



图6.1.12肤色识别效果图

6.1.3 边缘检测

采用创新思想，在实现二值化的基础上，只需要及少量的代码与资源就可以实现简单的边缘检测。边缘检测对于图像识别具有重大帮助，它可以进一步提取出物体本身的特征信息，提高识别效率。

目前主流的边缘检测算法往往都用到一阶和二阶导数，并且大多有数组矩阵的计算。过多的计算会导致逻辑门资源占用率增高，秉承简单高效的理念，我们所使用的边缘检测算法不需要计算，仅仅是简单的逻辑判断即可。

```
//二值化后的两个前后像素数据颜色值不同，判断为边缘
if(sb_ck1 != sb_ck2) begin
    bw_sb_data_r <= WHITE;
end
else begin
    bw_sb_data_r <= BLACK;
end
```

图6.1.13 边缘检测核心代码

通常情况下的边缘检测是对 3*3 矩阵作为判断，采用“8 与得 1”的判断方式，只有周围 8 个像素点都相同才置中间像素点为 1 或 0，而该作品采用降维方式，将平面矩阵降为一维矩阵，仅通过对比前后两个像素点的相同与否，即可判断当前像素点的 1/0 值。有两个寄存器来分别存储前后两个像素数据，二值化的图像数据非黑即白，当前后两个寄存器一黑一白时，代表此处为图像边缘，于是可以将边缘置为白色或者黑色。该方法的优点是使用简单，资源占用少，处理速度快，但缺点是容易受到单个噪声干扰，但是并不会放大噪声。在处理水平的边界时能力较弱，但是在处理连续曲折的图形边缘时可以展现出相较优良的效果。

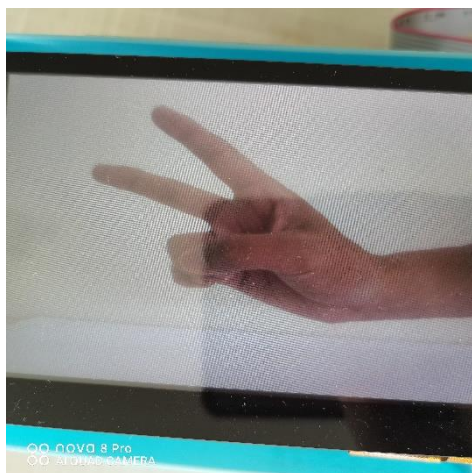


图6.1.14 手势检测原图

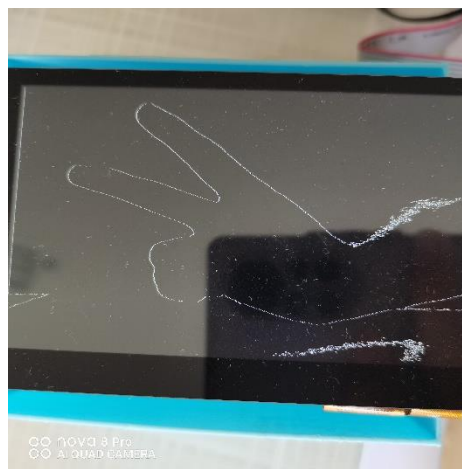
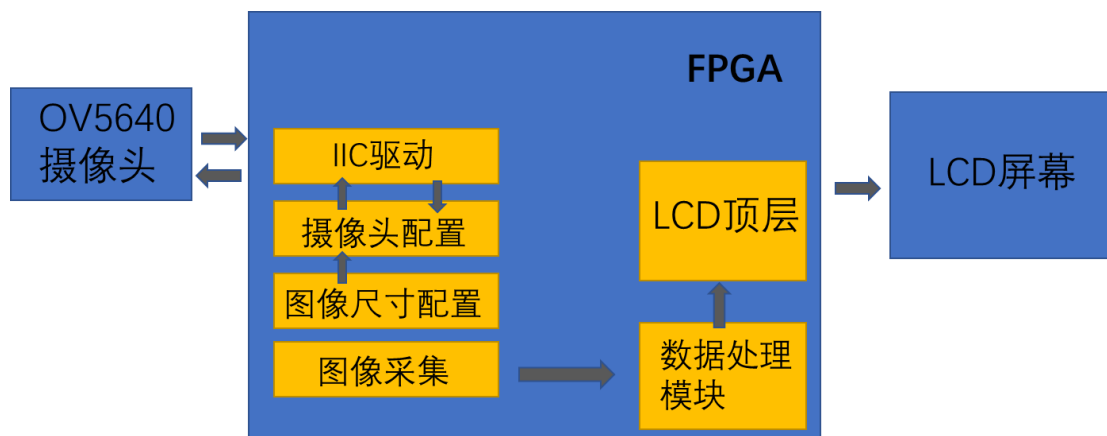


图6.1.15 边缘检测效果图

6.1.4 物块坐标给出

机械臂的逆运动学算法需要由图像识别部分提供物块的位置信息。当图像显示在屏幕上时，由于有行列像素点信息可以利用，得到坐标相对简单，然而在项目中后期的问题是，机械臂的逆运动学算法需要占用极大的资源，导致原本使用的 altera E10 芯片资源不足，采用 zynq 的芯片后却发现，用于图像显示的存储芯片 ddr3 接线连接在 arm 的 ps 端，使用 lcd 屏幕实时显示画面必定要用 ddr3 存储芯片，会导致违规，采用网线连接电脑显示又会导致车体过于臃肿不便行动。

在多番考量后决定摒弃实时显示的功能，对于无人智能设备，在其本身上安装显示屏幕实时显示是非必要的，只要显示到相应重要参数信息即可，所以本项目创新性的提出直接将摄像头读入的数据进行实时处理，只将处理后得到的有效信息进行保留与传递，将处理后的信息抛弃。如此一来，不但节省了储存资源，



在量化生产时可以减少一颗储存芯片从而降低成本，还可以降低电路复杂程度，极大的提高处理速度，，在刚得数据时就对其进行处理，使延时最小化。

图6.1.16 低延迟图像处理流程图

且摒弃 ddr3 的使用后，其他设备亦可以统一工作时钟，（模块自带时钟除外）因此，关于 pll 时钟锁相环的 ip 模块也可以省略，从而做到了整个项目纯使

用 fpga 可编程逻辑资源完成的设计，具有优越的可迁移性。不会受到任何板载芯片型号等限制。从而更好的突出了可重构性。使其能在多个不同的平台上快速部署。

坐标提取：

采用模拟显示端坐标计数法，由数学公式模拟出像素点在屏幕中的位置，从而得到需求坐标。由于只是去除了摄像头的显示功能，在摄像头读取方面并未改动，所以采集进入的数据依旧是以原先 480*272 分辨率的屏幕为前提输出的，由此我们可以得到，每一帧的准确像素点的个数为 130560 个，所以我们可以用一个整体的帧计数器来计时，用摄像头数据输入使能作为开关，每当处理 130560 个像素点后代表已经处理完一帧的数据，计数器清零开始处理下一帧数据。

```
//*****一帧像素点计数器*****//
always@ (posedge sys_clk or negedge sys_rst_n) begin
    if(!sys_rst_n) begin
        t_cnt <= 32'd0;
        tw_cnt <= 1'b0;
    end
    else if(pre_frame_de)begin
        if(t_cnt == 130560-1)begin //每帧的总像素点数，计够清0
            t_cnt <= 32'd0;
            tw_cnt <= ~tw_cnt;
        end
        else
            t_cnt <= t_cnt + 1'd1;
    end
end
end
```

图6.1.17 帧计数器程序

当我们需要传输具体某一点的坐标值时，我们可以通过如下公式得到相应
在屏幕中的模拟坐标：

$y = \text{当前像素点计数} / \text{行分辨率}$;

$X = \text{当前像素点计数} \% \text{行分辨率}$

如此便可以做到精准输出屏幕中任何一个像素点的坐标值。

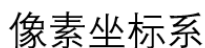


图 6.1.18 模拟显示屏幕输出坐标图

坐标传输:

设计中由于硬件问题，摄像头延长线的长度达不到安装在机械臂前端的要求，因此通过 3D 打印技术，制作出合适的升降台，采用识别执行分立设计，摄像头识别部分和机械臂抓取部分单独放置，通过对不同独立坐标转换设计，使机械臂最终能得到正确的坐标从而实现逆运动学的抓取。在识别坐标与机械臂坐标中，二者都是以自身为主创建坐标系的，如图是两个坐标系的对比。图中可以看出他们的坐标系原点不同，y 轴方向也不同，在实际的项目作品中，由于像素坐标系是采用模拟显示端计数法输出坐标，所以最终输出的坐标的范围是 $1 < x < 480$, $1 < y < 272$, 而机械臂的坐标范围在 $1 < x < 3000$, $1 < y < 3000$, $z = \text{定值}$ 。

因此在坐标转换中还需要进行适当的扩大与调整。

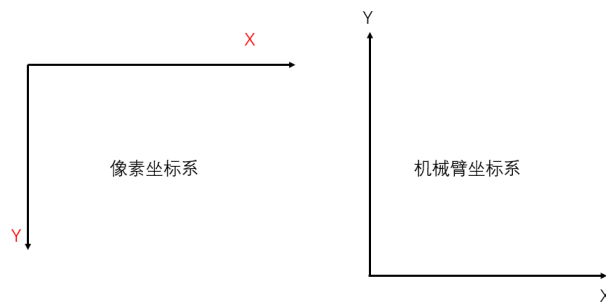


图6.1.19像素坐标系与机械臂坐标系差异对比图

```
assign x_com_rr=(5*x_com)-2500-450 ;//扩大坐标系,最后加减为平移调整
assign y_com_rr = 2500 - (5*y_com)-150 ;
```

图6.1.20 坐标系转换程序

x_com, y_com 是像素坐标, x_com_rr, y_com_rr 是输入给机械臂坐标的中间缓存, 有了物体坐标的传递, 视觉识别和机械臂就可以进行基础的联合调试。详细流程图为

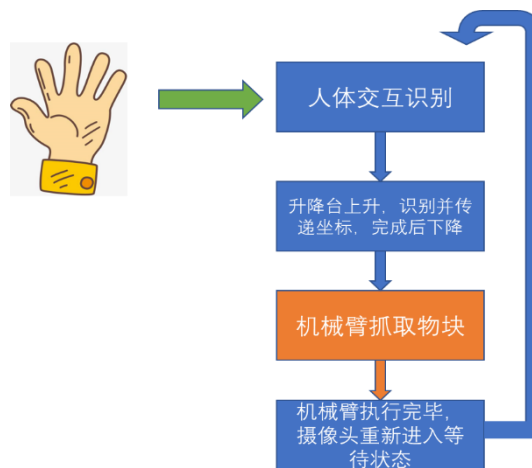


图6.1.21 视觉与机械臂交互流程图

6.1.5 LCD 显示屏驱动模块

采用正点原子 4:3 的 4.3 寸 TFTLCD 屏幕模块,

屏幕驱动时钟:

由于不同尺寸的屏幕所需要的驱动时钟不同,由于每个型号发的屏幕都有统一的 ID 信息,所以我们可以根据对应的 ID 信息产生所需要的时钟信号,

M2 LCD_B7	M1 LCD_G7	M0 LCD_R7	LCD_ID	时钟	对应分辨率
0	0	0	4342	9Mhz	480*272
0	0	1	7084	33.3Mhz	800*480
0	1	0	7016	50Mhz	1024*600
1	0	0	4384	33.3Mhz	800*480
1	0	1	1018	70Mhz	1280*800
x	x	x	NC		无

图6.1.22 屏幕对应信息表

本项目使用的屏幕对应的应该为 9Mhz,然而由于 fpga 板载时钟为 50Mhz,9Mhz 不方便直接产生,所以我们用四分频的 12.5Mhz 代替 9Mhz 标准时钟。

ID 的读取:

对于屏幕的 ID 读取,我们使用一个简单的 case 模块即可实现。由上图可知,屏幕的 ID 是由 RGB 信号线的三根线输出的,对应的管脚分别为 R,G,B 的最高位,如果输出的是 RGB565 格式的数据,则三位分别为 lcd_rgb[4] lcd_rgb[10] lcd_rgb[15],如果输出的是 RGB888 格式,则是三色通道的最高位 lcd_rgb[7] lcd_rgb[15] lcd_rgb[23]。上电时屏幕会给出电平信号表示自己的 ID,即可通过读取得到对应的信息。

```
case({lcd_rgb[7],lcd_rgb[15],lcd_rgb[23]})
  3'b000 : lcd_id <= 16'h4342; //4.3' RGB LCD RES:480x272
  3'b001 : lcd_id <= 16'h7084; //7' RGB LCD RES:800x480
  3'b010 : lcd_id <= 16'h7016; //7' RGB LCD RES:1024x600
  3'b100 : lcd_id <= 16'h4384; //4.3' RGB LCD RES:800x480
  3'b101 : lcd_id <= 16'h1018; //10' RGB LCD RES:1280x800
  default : lcd_id <= 16'd0;
endcase
```

图6.1.23 LCD ID读取程序

屏幕驱动：

屏幕驱动部分主要是确定显示范围以及数据请求使能等信号，可以理解为此时的屏幕为一个二维的画板，我们可以得到相应的二维坐标之后就可以进行标记，最终将一个区域集成为一个精简的使能信号，当然，这个区域甚至可以移动，本作中的显示准心部分即是根据此原理实现。将物块实时变动的坐标与屏幕反复刷新坐标做对比，从而实现准心线始终跟踪物块的效果。

```
//RGB LCD 采用DE模式时，行场同步信号需要拉高
assign lcd_hs = 1'b1; //LCD行同步信号
assign lcd_vs = 1'b1; //LCD场同步信号

assign lcd_bl = 1'b1; //LCD背光控制信号
assign lcd_clk = lcd_pclk; //LCD像素时钟
assign lcd_de = lcd_en; //LCD数据有效信号

//识别物块使能
assign check_flag = ((x_com_r >= pixel_xpos && x_com_r <= pixel_xpos + 1) || (y_com_r >= pixel_ypos && y_com_r <= pixel_ypos + 1)) ? 1'b1 : 1'b0;

//使能RGB888数据输出
assign lcd_en = ((h_cnt >= h_sync + h_back) && (h_cnt < h_sync + h_back + h_disp)
&& (v_cnt >= v_sync + v_back) && (v_cnt < v_sync + v_back + v_disp))
? 1'b1 : 1'b0;

//请求像素点颜色数据输入
assign data_req = ((h_cnt >= h_sync + h_back - 1'b1) && (h_cnt < h_sync + h_back + h_disp - 1'b1)
&& (v_cnt >= v_sync + v_back) && (v_cnt < v_sync + v_back + v_disp))
? 1'b1 : 1'b0;

//像素点坐标
assign pixel_xpos = data_req ? (h_cnt - (h_sync + h_back - 1'b1)) : 11'd0;
assign pixel_ypos = data_req ? (v_cnt - (v_sync + v_back - 1'b1)) : 11'd0;

//RGB888数据输出
assign lcd_rgb = lcd_en ? pixel_data : 24'd0;

always@(posedge lcd_clk) begin
    x_com_r <= x_com;
    y_com_r <= y_com;
end
```

图6.1.24 屏幕驱动程序

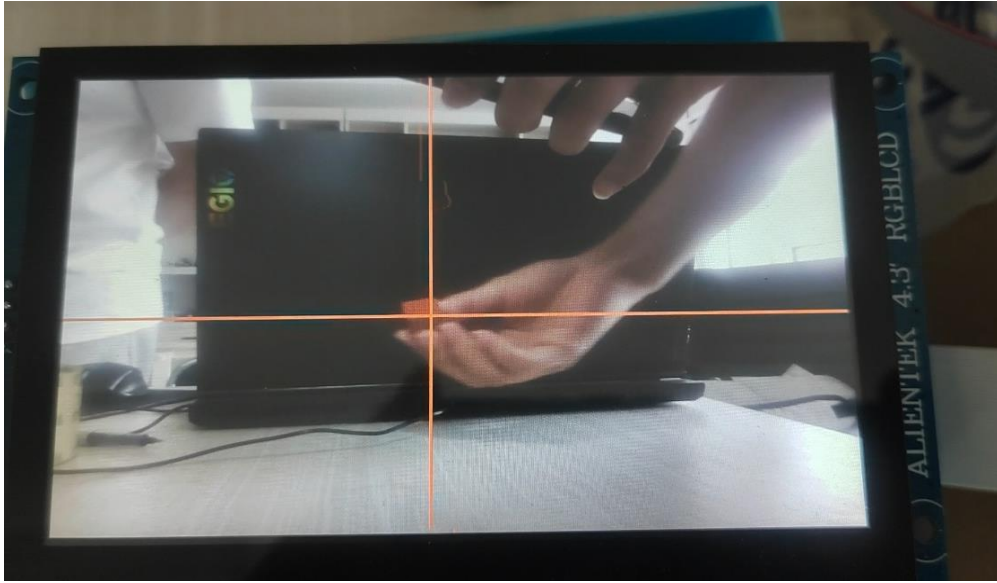


图6.1.25 物块标记效果图

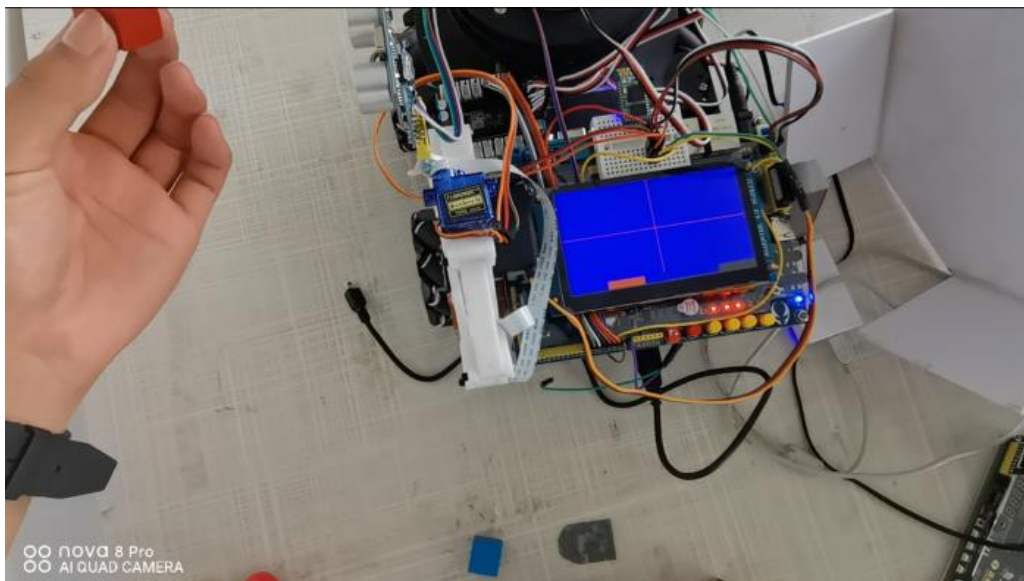


图6.1.26 去显示识别效果图

屏幕显示：

显示与驱动程序互补互通，继续以画板为例，驱动程序规定了每个一些区间以及显示范围，显示程序则为之“涂色”，将我们需要显示的画面放置在对应的区域里。最简单的应用为，驱动范围设置为全屏范围，显示图像为摄像头采集到

的画面，就可以做到实时显示画面的功能。本项目中借屏幕来实时显示一些传感器模块的信息。显示信息如图标注，其中，每个块都有红绿蓝三色，超声波模块的红色为距离最近，绿色为合适距离，蓝色为过远；物块大小显示为红色最小，绿色中等，蓝色最大；颜色识别则红绿蓝对应相应的颜色；手势识别为，红色为手掌肤色面积最先，视为握拳，绿色为过度，视为伸出手指或者距离过近的误差，蓝色为面积最大，视为手掌。



图6.1.27 屏幕显示各区域说明

6.2 机械臂逆运动学算法

机械臂控制模块采用逆运动学算法，根据摄像头的图像处理得到物块所在图片的具体像素点，然后换算为坐标，根据逆运动学算法坐标解算得到机械臂六个舵机对应的 pwm 输出，转动到物块上方进行夹取。

6.2.1 CORDIC 算法

而要实现逆运动学算法，cordic 是必不可少的。而要实现此机械臂的逆运动学我们主要用到了反正切和开根号的 cordic 算法。

首先介绍简单的 cordic 在三角函数中的原理。

1.如下图所示，点(x2,y2)可以通过点(x1,y1)旋转来得到，根据 givens 旋转可以得到如下公式： $x_2 = x_1 \cdot \cos \theta - y_1 \cdot \sin \theta$;

$$y_2 = x_1 \cdot \sin \theta + y_1 \cdot \cos \theta ;$$

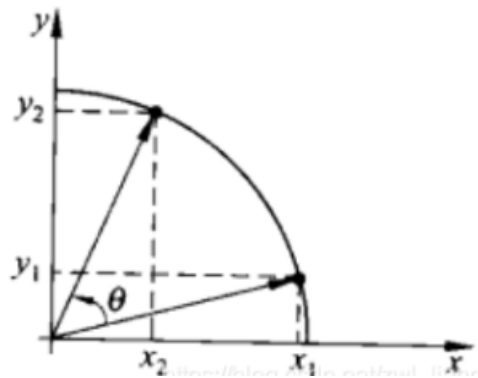


图 6.2.1

2. 上式提出 $\cos \theta$ 后

$$X_2 = \cos \theta \cdot (x_1 - y_1 \cdot \tan \theta);$$

$$Y_2 = \cos \theta \cdot (x_1 + y_1 \cdot \tan \theta);$$

3. 去掉 $\cos \theta$ 后可得到伪旋转方程式如下

$$X_3 = x_1 - y_1 \cdot \tan \theta;$$

$$Y_3 = x_1 + y_1 \cdot \tan \theta;$$

伪旋转后， x 与 y 的值都相比实际值增加了 $\cos^{-1} \theta$, 且 $\cos^{-1} \theta > 1$ 。

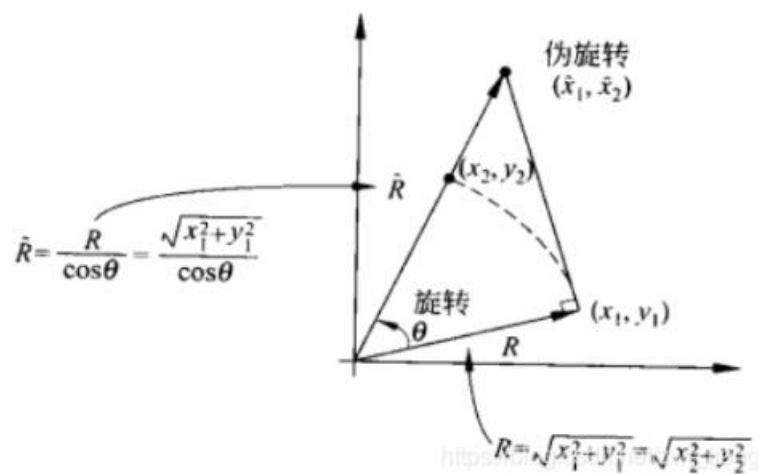


图 6.2.2

伪旋转的主要作用是得到只含正切三角函数的等式，正切三角函数可以使其用 2^{-i}

来表示，即令 $\tan \theta = 2^{-i}$ 。此时伪旋转公式可以变为如下形式

$$X_4 = x_1 - y_1 * 2^{-i};$$

$$Y_4 = x_1 + y_1 * 2^{-i};$$

这样使得对任意角度的旋转变成了通过一些小角度的旋转 i 次来完成的，也迭代了 i 次，适合硬件电路实现，下表是迭代 13 次后各次迭代的 θ ， \cos ， \tan 函数值

i	$\tan \theta$	θ^i	$\cos \theta$
1	1	45.0	0.707106781
2	0.5	26.555051	0.894427191
3	0.25	14.0362434679	0.9701425
4	0.125	7.1250163489	0.992277877
5	0.0625	3.5763343750	0.998052578
6	0.03125	1.7899106082	0.999512078
7	0.015625	0.8951737102	0.999877952
8	0.0078125	0.4476141709	0.999969484
9	0.00390625	0.2238105004	0.999992371
10	0.001953125	0.1119056771	0.999999093
11	0.000976563	0.0559528919	0.999999523
12	0.000488281	0.0279764526	0.999999881
13	0.000244141	0.0139882271	0.99999997

图 6.2.3

如上图所示，此方法支持的角度范围为 $(-99.7, 99.7)$ ，二三象限的角度值可以通过三角函数的恒等变换来转换到此范围内，在进行计算。

4. 此时的伪旋转公式可以改为如下迭代公式 $\sin \quad \cos \quad \tan$

$$x^{(i+1)} = x^i - d_i(2^{-i} * y^i)$$

$$y^{(i+1)} = y^i - d_i(2^{-i} * x^i)$$

同时设置一个角度累加方程，来记录角度的累加值

$$z^{(i+1)} = z^i - d_i * \theta^i$$

注： d_i 的值为 1 或者 -1，主要是用来确定旋转方向，例如要计算 $\sin 30$ ，当 $z^{(i+1)} > 30$ 时减少角度，小于 30 则加大角度。

5. 令伸缩因子

$$k_n = \cos 45 * \cos 26.5 * \cos 14.03 * \dots * \cos 0.0139 = 0.60725941$$

而根据 cordic 原理我们仍然可以用在反正切上实现功能：基于 cordic 算法的旋转模式。在 always 块内，使用连续的旋转操作，并在每个旋转操作后更新极角“z”和坐标“x”“y”。在这个过程中，通过使用已知的三角函数来计算旋转后的向量在坐标轴上的投影，进而得到旋转向量的大小和角度。每一次迭代都会将向量旋转一定角度，直到达到所需的精度为止。

6.2.2 机械臂平滑

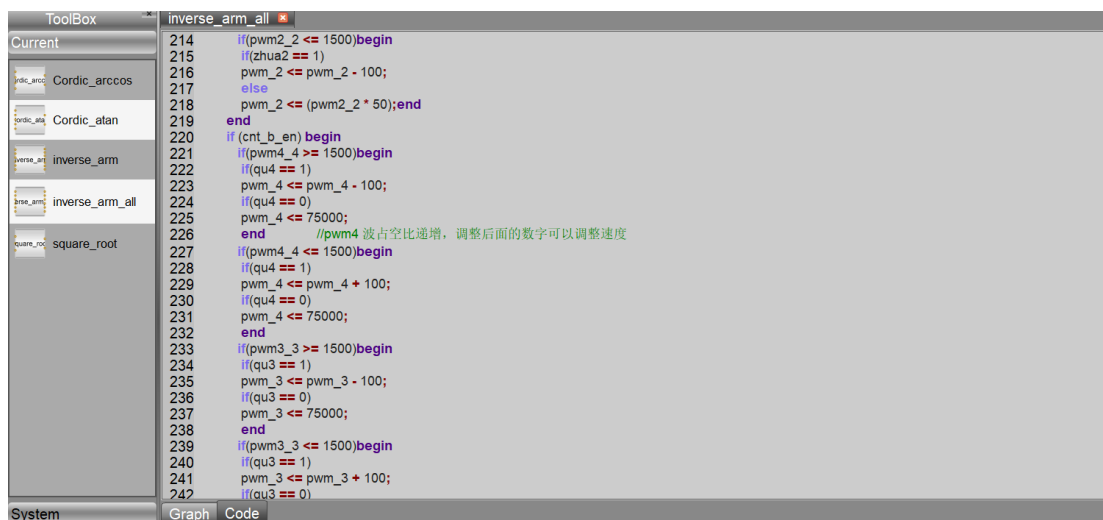
完成这个逆运动学算法之后，我们就已经可以根据摄像头给的物块坐标让机械臂到达指定位置了，但是这样的话会很僵硬，所以我们加上了机械臂平滑模块，也就是让机械臂从直接瞬间到达指定位置改成了，一步一步顺滑的到达指定位置。

已经我们日夜兼程的研究和实验，我们发现可以每 1ms 给定一个的变化的 pwm 可以让他比较顺滑的执行动作。以下是部分核心代码的展示：



```
166   pwm_2 <= 25'd1500_000/20;  
167   end  
168   else begin  
169     if(ok == 'b1)  
170       begin  
171         cnt_1 <= cnt_1 + 1'd1;  
172         if (cnt_1 == 500_000) begin //计数器, 1ms变化一次//if (cnt_1 == 500_000) begin  
173           cnt_1 <= 0;  
174           if(pwm1_1 <= 1500)begin  
175             if(zhua1 == 1)  
176               pwm_1 <= pwm_1 - 200; //pwm1是最底层转向的舵机, 先地下转完再其他的转, 所以这里需要独立把pwm1快速转到位置, 再转其他的舵机  
177             else  
178               pwm_1 <= (pwm1_1 * 50);  
179             end  
180             if(pwm1_1 >= 1500)begin  
181               if(zhua1 == 1)  
182                 pwm_1 <= pwm_1 + 200; //pwm1是最底层转向的舵机, 先地下转完再其他的转, 所以这里需要独立把pwm1快速转到位置, 再转其他的舵机  
183               else  
184                 pwm_1 <= (pwm1_1 * 50);  
185             end  
186             if (cnt_en) begin  
187               if(pwm4_4 >= 1500)begin  
188                 if(zhua4 == 1)  
189                   pwm_4 <= pwm_4 + 100; //pwm4 波占空比递增, 调整后面的数字可以调整速度  
190                 else  
191                   pwm_4 <= (pwm4_4 * 50);end  
192               if(pwm4_4 <= 1500)begin  
193                 if(zhua4 == 1)  
194                   pwm_4 <= (pwm4_4 * 50);end  
195               end  
196             end  
197           end  
198         end  
199       end  
200     end  
201   end  
202 end
```

图 6.2.1 机械臂平滑部分核心代码 1



```
214   if(pwm2_2 <= 1500)begin  
215     if(zhua2 == 1)  
216       pwm_2 <= pwm_2 - 100;  
217     else  
218       pwm_2 <= (pwm2_2 * 50);end  
219   end  
220   if (cnt_b_en) begin  
221     if(pwm4_4 >= 1500)begin  
222       if(qu4 == 1)  
223         pwm_4 <= pwm_4 + 100;  
224       if(qu4 == 0)  
225         pwm_4 <= 75000;  
226       end //pwm4 波占空比递增, 调整后面的数字可以调整速度  
227       if(pwm4_4 <= 1500)begin  
228         if(qu4 == 1)  
229           pwm_4 <= pwm_4 + 100;  
230         if(qu4 == 0)  
231           pwm_4 <= 75000;  
232         end  
233       if(pwm3_3 >= 1500)begin  
234         if(qu3 == 1)  
235           pwm_3 <= pwm_3 - 100;  
236         if(qu3 == 0)  
237           pwm_3 <= 75000;  
238         end  
239       if(pwm3_3 <= 1500)begin  
240         if(qu3 == 1)  
241           pwm_3 <= pwm_3 + 100;  
242         if(qu3 == 0)  
243           pwm_3 <= 75000;  
244         end  
245       end  
246     end  
247   end  
248 end
```

图 6.2.2 机械臂平滑部分核心代码2

6.3 麦克纳姆轮驱动算法

6.3.1 麦克纳姆轮逆运动学算法

麦克纳姆轮小车的控制主要采用了自主开发的底盘向量逆运动学算法和基于 imu 的 pid 算法。

底盘向量的逆运动学算法主要原理和算法推导的主要过程如下：

第一步：根据麦克纳姆轮的本身特性，分析车轮触地面对小车整体施加的作用力，当四个轮子全部正转的时候，可以得出如下的向量图：

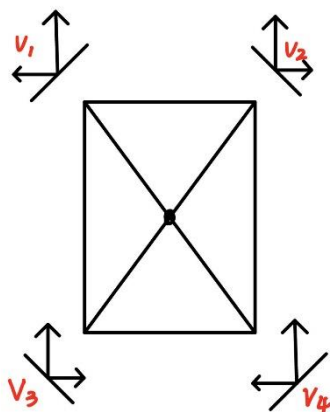


图6.3.1 全部正转向量图示意

在不要求车体旋转的前提下，即沿车体的中心对称分布的力必须大小相等方向相同才能使得车体不发生旋转，所以就必须要要求对角线分布的两个车轮的转速必须保持一致，即

$$V_1 = V_4$$

$$V_2 = V_3$$

所以我们可以将四个轮子两两一组化简， V_1, V_4 一组， V_2, V_3 一组，构成了两个以小车中心为起点，方向沿着小车底盘对角线的向量 Z_1, Z_2 .

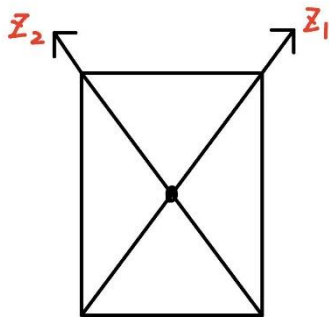


图6.3.2 合成向量示意图

当我们给出一个目标地点的坐标时通常都是在平面直角坐标系内给出，如在刚刚的合成向量的坐标系中表示这个坐标比较复杂，需要经过复杂的坐标变换，大量调用 \sin 和 \cos 函数，占用资源多。所以我们采用向量分解的方法来进行计算。首先建立一个基于小车中心的平面直角坐标系，在这个坐标系中，由于两个合成向量与下 x, y 轴的夹角都是固定的，两个向量在 x 和 y 轴的映射的长度呈固定比例即两个对角线的斜率。设对角线斜率为 k ，另一条为负 k 。

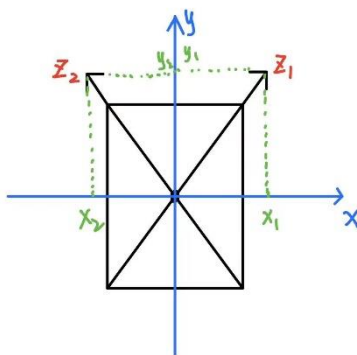


图6.3.3 向量分解示意图

在建立了直角坐标系后，将目标的坐标值代入，求出相对应的向量的斜率，从而可以列出一个由 x_1 , x_2 组成的二元一次方程组：

$$X_1 + X_2 = 1$$

$$Y_1 + Y_2 = \frac{Y}{X}$$

其中 X, Y 为目标的坐标值， $Y_1 = kX_1$, $Y_2 = -kX_2$ 。

解开方程组即可表示出两组轮子的速度关系，从而得出了小车按照这种速度关系进行运动，运动方向是朝向目标地点的。而具体的轮子的速度大小则需按照实际的物理换算和后续的 pid 算法来控制。

6.3.2 Imu 数据解析模块

在 imu 传感器上配置好开机主动上报的数据和波特率等参数后，利用 FPGA 和 imu 模块的串口 ttl 端进行连接。即 imu 的 tx 口接到 FPGA 的接收端口上，imu 的 rx 口接到 FPGA 的发送端口上。

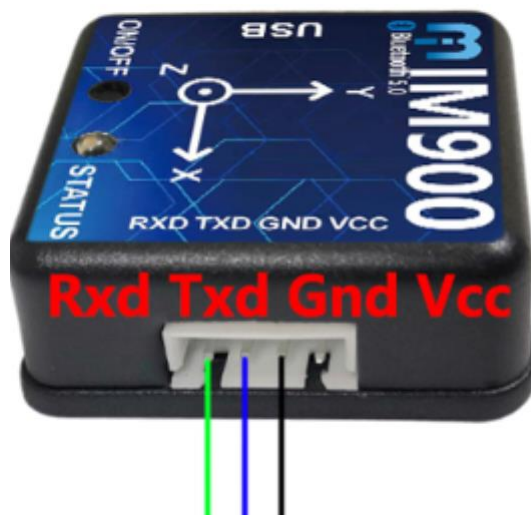


图6.3.4 imu连线示意图

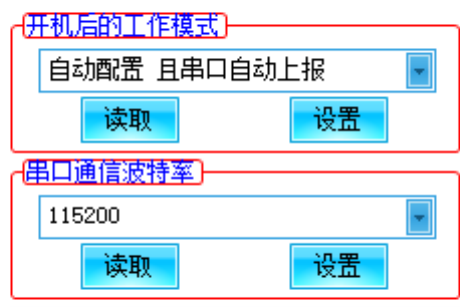


图 6.3.5 imu 的主动上报设置页面

在硬件连线 and 配置都以完善的条件下需要进行的的就是 FPGA 对于 imu 传感器传回的串口数据进行解析。

Imu 传感器的串口参数为 115200bps、8 数据位、1 停止位、无校验位。数据包格式: 前导码+起始码+地址+长度+数据体+校验码+结束码

前导码 为固定数据, 用于唤醒可能处于睡眠状态的模块, 只有发送给模块串口的指令包才需加上前导码

起始码 固定为 0x49

地址 为串口通信地址, 当扩展 485 总线使用时, 用于标识模块身份, 设备地址取值 0-254, 广播地址 255

长度 为数据体的字节数

数据体 为具体内容的负载

校验码 指从地址到数据体(包含地址、长度和数据体)每字节数据的和校验

结束码 固定为 0x4D

数据解析模块需要将串口传回的数据进行数据包的整体识别, 剔除杂乱数据并将准确的数据体拆分出来, 最后读取数据体内的内容进行组合, 并将数据分类

传出。下面给出一个例子：

如串口收到以下数据包（仅仅订阅含重力加速度）

49 00 0D 11 02 00 4F 5B 16 00 36 00 1F 00 FB 07 37 4D

如下：

起始码：0x49

地址：0x00

数据体的长度：0x0D

数据体：0x11 0x02 0x00 0x4F 0x5B 0x16 0x00 0x36 0x00 0x1F 0x00 0xFB

0x07

：0x11 表示本数据包为传感器上报的测量结果数据

0x02 0x00 表示订阅标识值为 0x0002(即对应的重力加速度)，

0x4F 0x5B 0x16 0x00 表示运行时间戳，小端对齐还原为 32 位数即

0x00165B4F=1,465,167(十进制)毫秒

接下来的数据根据订阅标识 0x0002，即二进制的 0000 0000 0000 0010,重力加速度。

由于 imu 中传出的数据为默认十六进制数据，需要组合并转化为有符号数，再乘以相应系数才能正常使用，此时重力加速度的系数 0.00478515625，经过处理之后可以得出对应数值，如下：

0x36 0x00 即 AX=0x0036(即有符号数 0x0036)*0.00478515625f=0.2583 m/s²

0x1F 0x00 即 $A_y = 0x0018$ (即有符号数 0x001F) * 0.00478515625f = 0.1483 m/s²

0xFB 0x07 即 $A_z = 0x07FB$ (即有符号数 0x07FB) * 0.00478515625f = 9.7761 m/s²

校验码:

0x37 = 0x00 + 0x0D + 0x11 + 0x02 + 0x00 + 0x4F + 0x5B + 0x16 + 0x00 + 0x36 + 0x00 + 0x1F

+ 0x00 + 0xFB 0x07 = 0x0237 (保留最后 1 字节即 0x37)

结束码: 0x4D

```

imusoliddataselct
106 subtag1<=1'b1;
107 end
108 else begin
109 subtag1<=1'b0;
110 end
111 if(rx_done) begin
112 if(dataout==1'b1)begin
113 c<=b;
114 b<=a;
115 a<=uart_rxd_data;
116 if(datain==1'b1)begin
117 arrayimudata[0]<=c;
118 arrayimudata[1]<=b;
119 arrayimudata[tag]<=uart_rxd_data;
120 tag<=tag+8'd1;
121 end
122 end
123
124
125 if(arrayimudata[detect]==8'h4D)begin
126 data_analysis<=1'b1;
127 tag<=8'd2;
128 subtag<=16'd0;
129 length<=8'd0;
130 detect<=8'd0;
131 datain<=1'b0;
132 for (i = 0; i < 20; i = i + 1) begin
133 arrayimudata[i] = 0;
134 end
135 end
136 /*
137 if(tag==8'd2)begin
138 data_analysis <= 1'b0;
139 end*/

```

图6.3.6imu数据解析部分代码

下面以伪代码解释如何本模块数据处理流程:

创建一个 16 位宽, 长度为 32 的数组, 用于存放解析后的数据。创建三个寄

寄存器 a, b, c 来充当串口接收到的数据的缓存区。

将串口接收到的数据进行数据包包头的的数据对比

If (串口接收数据标识==1)

进行

c<=b;

b<=a;

a<=uart_rxd_data;即将串口的数据进行缓存。

进行数据包包头判断（数据包包头为固定数值）

arrayimudata[0]<=c;

arrayimudata[1]<=b;

arrayimudata[tag]<=uart_rxd_data;

tag<=tag+8'd1;

当数据包识别成功之后 data_in 信号拉高，此时串口接收的数据开始存入缓存区。

同时根据数据包中的数据长度位来计算出数据包的包尾位置，即

arrayimudata[detect]==8'h4D;

当检测到包尾位置的数据正确后，数据储存区正常清零并输出对应处理好的传感器数据。

如果包尾位置数据不正确，则整体数据清零并不输出传感器数据，进入下次接收准备阶段。

6.3.3 速度数值转字符串

当上层逆运动学解算出四个轮子应有的速度关系和具体数值之后,就需要将该转速发送给对应的电机。而本次使用的电机为总线指令电机,所有经过 uart 串口通信的数据指令都必须为字符串格式。所以本部分解决的问题就是格式转化的问题。

```
assign a1=motor1/1000;  
assign b1=motor1%1000/100;  
assign c1=motor1%1000%100/10;  
assign d1=motor1%1000%100%10/1;  
assign a2=motor2/1000;  
assign b2=motor2%1000/100;  
assign c2=motor2%1000%100/10;  
assign d2=motor2%1000%100%10/1;  
assign a3=motor3/1000;  
assign b3=motor3%1000/100;  
assign c3=motor3%1000%100/10;  
assign d3=motor3%1000%100%10/1;  
assign a4=motor4/1000;  
assign b4=motor4%1000/100;  
assign c4=motor4%1000%100/10;  
assign d4=motor4%1000%100%10/1;
```

图 6.3.7 四个电机速度取位部分程序

```
parameter [7:0] ling ="0"; //拼音0, 下列全为拼音  
parameter [7:0] yi ="1";  
parameter [7:0] er ="2";  
parameter [7:0] san ="3";  
parameter [7:0] si ="4";  
parameter [7:0] wu ="5";  
parameter [7:0] liu ="6";  
parameter [7:0] qi ="7";  
parameter [7:0] ba ="8";  
parameter [7:0] jiu ="9";  
  
case(a1) //赋值对应的字符串数组  
4'd0: array1[5]<=ling; //前000  
4'd1: array1[5]<=yi;  
4'd2: array1[5]<=er;  
4'd3: array1[5]<=san;  
4'd4: array1[5]<=si;  
4'd5: array1[5]<=wu;  
4'd6: array1[5]<=liu;  
4'd7: array1[5]<=qi;  
4'd8: array1[5]<=ba;  
4'd9: array1[5]<=jiu;  
endcase  
case(b1)
```

图6.3.8 十进制与字符串转换部分

完成电机速度按位取整之后，将对应位数制进行 case 语句的判定，从而实现了对应电机字符串数组内的字符串修改。

七、系统整合与调试

7.1 imu 数据解析仿真

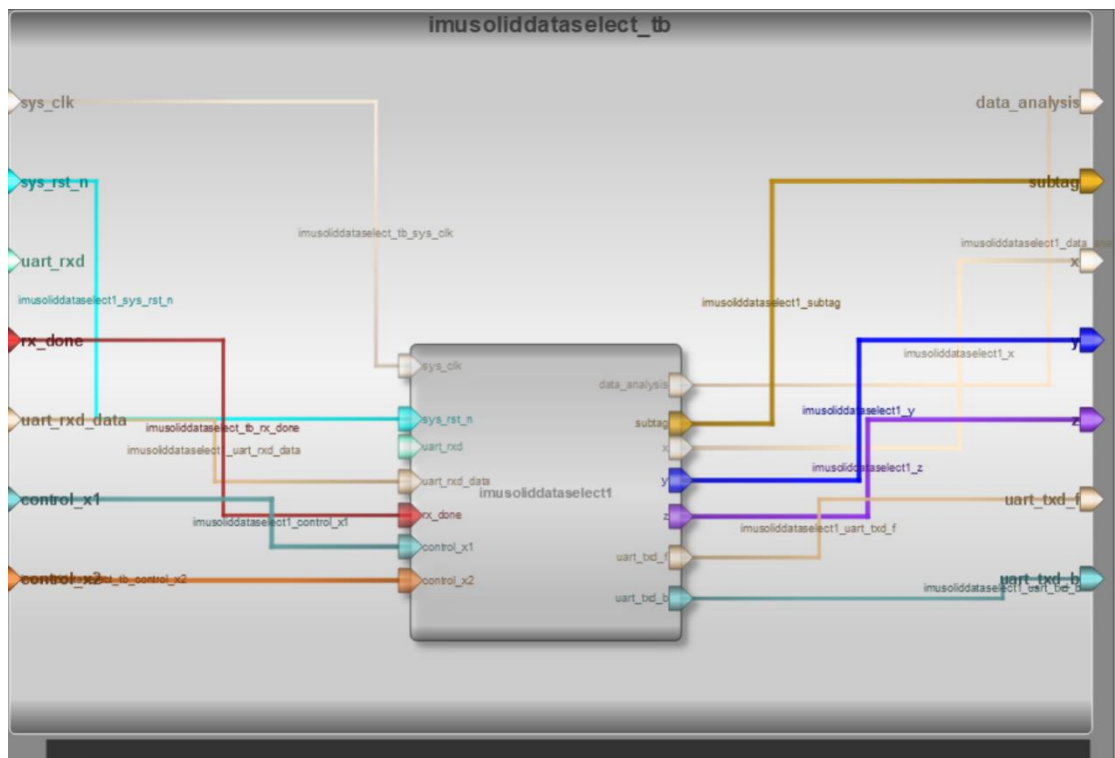


图7.1.1 imu数据解析仿真模块框图

对 imu 数据解析模块输入如下数据包，串口每一百纳秒更新一次串口接收

数据，同时串口通信接收标志位拉高，代表有一个新的串口数据进入。

7.2 cordic 的代码解释与仿真

1.这是正余弦的 cordic 算法：



图 7.2.1 Robei EDA的Cordic_sin_cos框图

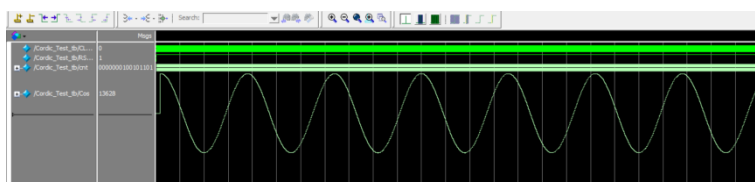


图 7.2.2 cordic_sin_cos仿真

它通过一系列的迭代计算来逼近给定角度的正弦和余弦值。

输入端口包括 CLK_50M 和 RST_N，输出端口包括 Sin、Cos 和 Error。

在这个代码中，使用了一系列的 always 块来实现 Cordic 算法的迭代计算。每个 always 块中都根据上一个阶段的计算结果和旋转角度进行计算，得到下一个阶段的结果。计算是在每个上升沿时钟触发或者复位信号变为低电平时触发的。

代码中的每个 always 块都根据当前阶段的旋转角度的最高位（即 $zX[31]$ ）来判断旋转方向，然后使用移位和加减操作来计算新的 x 、 y 和 z 的值。

最后，根据 CORDIC 算法的性质，通过迭代计算，最终可以得到输入角度对应的正弦值和余弦值。这里的结果保存在 Sin 和 Cos 寄存器中。

需要注意的是，本代码中的 CORDIC 算法是固定迭代次数的，并且使用了 16 级流水线结构，迭代次数由 Pipeline 参数指定。此外，K 参数表示一个常数，用于缩放输入角度。

2. 其中开根号的 cordic 算法在这里的作用是用迭代的方式来逐步逼近平方根。每次迭代，都会通过比较输入数和当前逼近值的大小关系，来选择加/减一个部分平方根值。最终，经过多次迭代，逼近值会逐渐趋近于真实的平方根值。

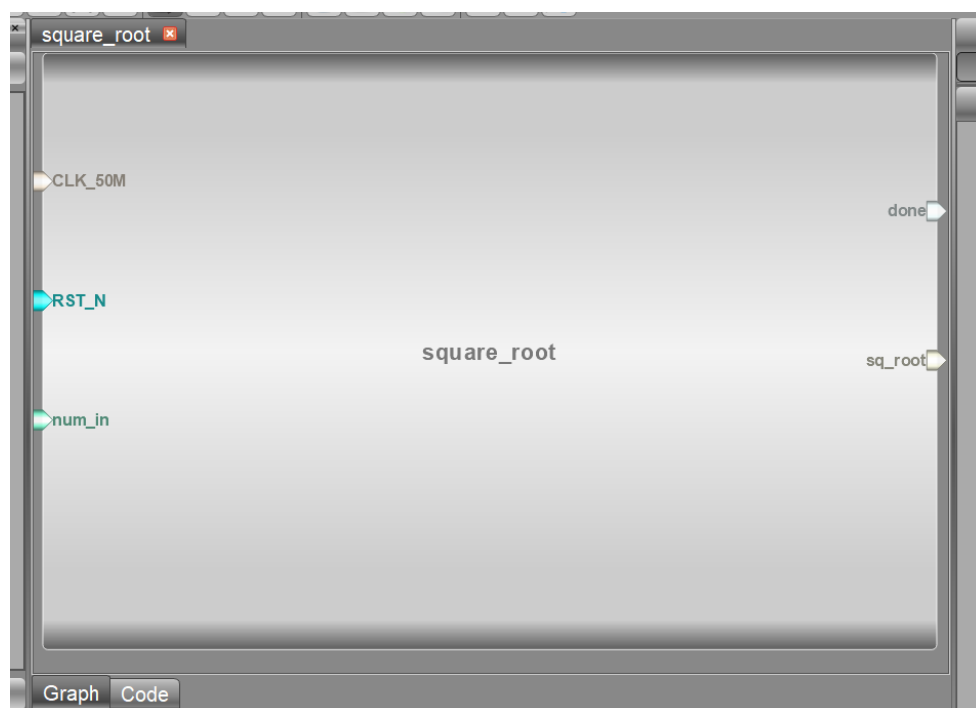


图7.2.3 Robei EDA的cordic_sqrt框图

仿真展示：

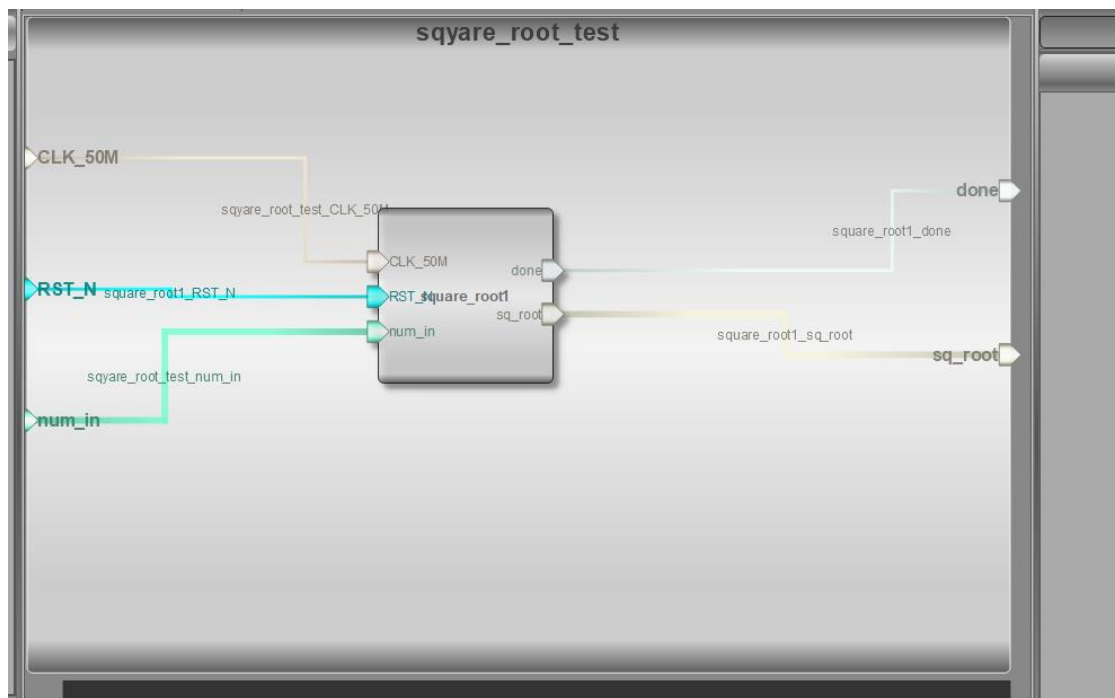


图7.2.4 cordic_sqrt 测试框图

```
ToolBox
current
are_roo square_root

sqyare_root_test
10 initial begin
11   CLK_50M = 0;
12 end
13 always #10 CLK_50M = ~CLK_50M;
14 initial begin
15   RST_N = 0;
16   #100;
17   RST_N = 1;
18 end
19 initial begin
20   #10;
21   num_in = 7500;
22   #1000;
23   num_in = 1000;
24   $finish;
25 end
26
27
28
29
```

图7.2.5 cordic_sqrt 仿真代码

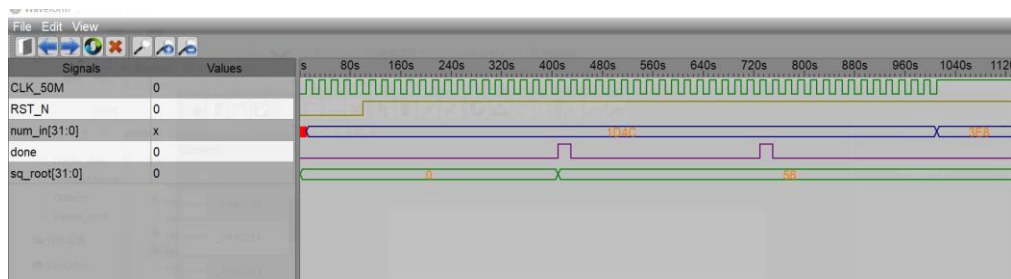


图7.2.6 cordic_sqrt 仿真波形图

该模块的大致功能：

- 接收一个时钟信号 CLK_50M 和一个复位信号 RST_N，并通过这些信号的边沿触发进行计算。

- 在输入 num_in 中提供需要计算平方根的数字。

- 当计算完成后，将输出信号 done 置高，表示输出结果准备完毕。

- 将计算得到的平方根存储在输出端口 sq_root 中。

该模块的计算过程如下：

- 在第一个时钟周期内，将输入数字存储到变量 a 中，并将输出信号 done 置零。

- 从第二个时钟周期开始，根据循环索引和状态进行迭代计算。

- 使用加法和减法操作更新中间变量 left 和 right。

- 对变量 a 进行左移 2 位，相当于乘以 4。

- 根据中间变量 r 的最高位决定进行加法还是减法运算。

- 更新结果变量 q。

- 当循环达到设定的最大值时，表示计算完成，将输出信号 done 置高，并

将计算得到的平方根赋值给输出端口 `sq_root`。

而基于 `cordic` 算法的反正切函数的实现：



图7.2.7 Robei EDA的`cordic_atan`框图

3.在这个反正切函数的实现中，输入是 y/x ，即待求的角度的正切值。通过对输入进行一系列的转换和迭代，最终可以得到反正切值。在代码中，通过调整 K 值来控制精度， K 越大，精度越高，但计算时间也会相应增加。如果对精度要求不高或需要更快的计算速度，可以选择较小的 K 值。不过在我们的机械臂中，还是尽量让 k 值大一些，保证精确度。

下面是反正切的 `arctan` 算法（部分）：

```
Cordic_atan
28
29 reg [15:0] x; //无符号数 位宽最小可以设置为3bit
30 reg [15:0] y;
31 always@(posedge CLK_50M)
32 begin
33   x <= xx[31] ? ~xx+1 : xx;
34   y <= yy[31] ? ~yy+1 : yy;
35 end
36
37
38 `define angle_0 32'd2949120 //45°*2^16
39 `define angle_1 32'd1740992 //26.5651°*2^16
40 `define angle_2 32'd919872 //14.0362°*2^16
41 `define angle_3 32'd466944 //7.1250°*2^16
42 `define angle_4 32'd234368 //3.5763°*2^16
43 `define angle_5 32'd117312 //1.7899°*2^16
44 `define angle_6 32'd58688 //0.8952°*2^16
45 `define angle_7 32'd29312 //0.4476°*2^16
46 `define angle_8 32'd14656 //0.2238°*2^16
47 `define angle_9 32'd7360 //0.1119°*2^16
48 `define angle_10 32'd3648 //0.0560°*2^16
49 `define angle_11 32'd1856 //0.0280°*2^16
50 `define angle_12 32'd896 //0.0140°*2^16
51 `define angle_13 32'd448 //0.0070°*2^16
52 `define angle_14 32'd256 //0.0035°*2^16
53 `define angle_15 32'd128 //0.0018°*2^16
54
55 parameter Pipeline = 16;
56 parameter K = 32'h09b74 //K=0.607253*2^16 32'h09b74

Graph Code
```

图7.2.8 arctan_cordic算法1

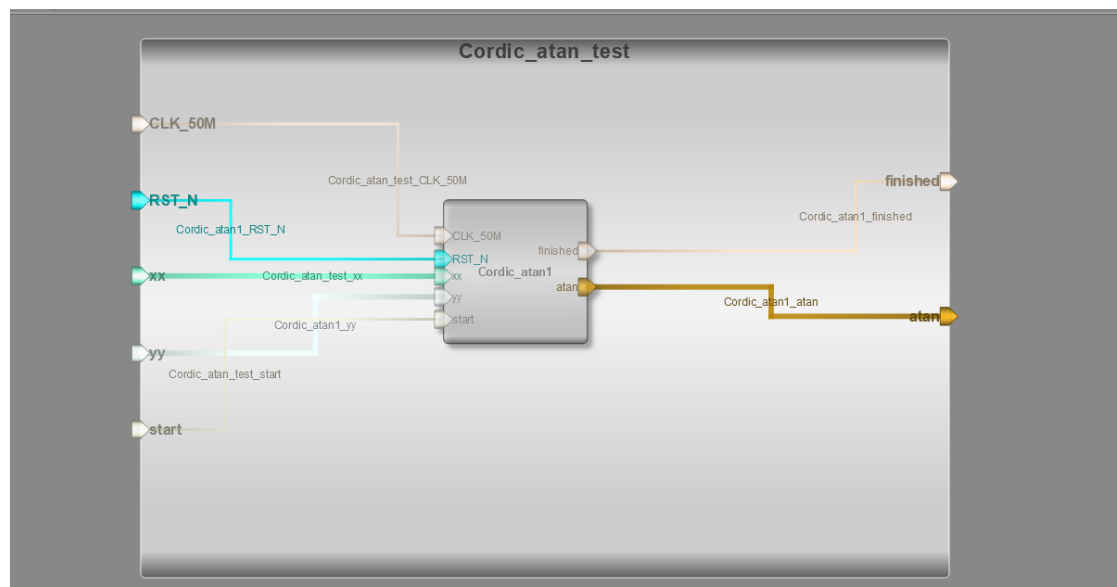


图 7.2.9 cordic_atan 测试框图

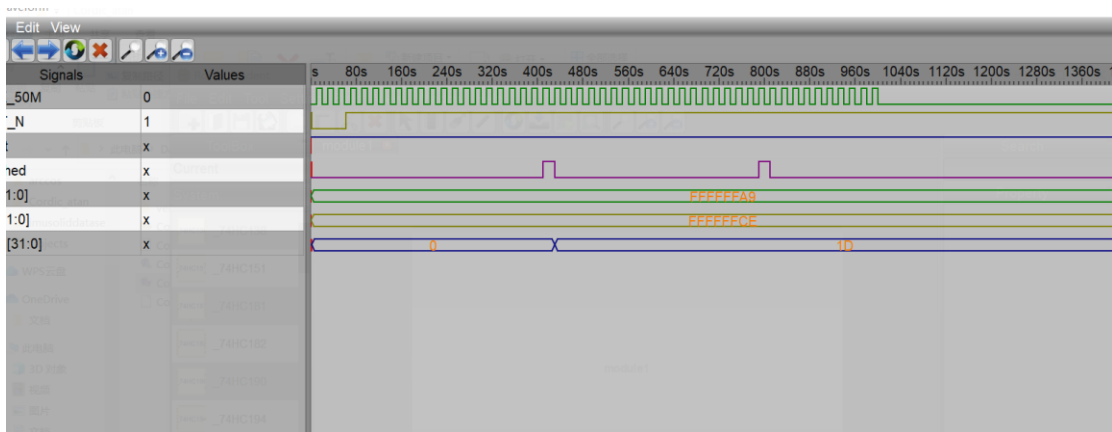


图7.2.10 cordic_atan 仿真波形图

而在逆运动学中我们还用到了反余弦函数，而根据数学算式我们知道反余弦可以根据反正切和开根号算出来。其数学公式为： $\cos^{-1} x = \frac{\pi}{2} - \tan^{-1} \frac{x}{\sqrt{1-x^2}}$

之后我们可以根据开根号和反正切模块的使能配合得到反余弦值。

反余弦的 robei 框图:

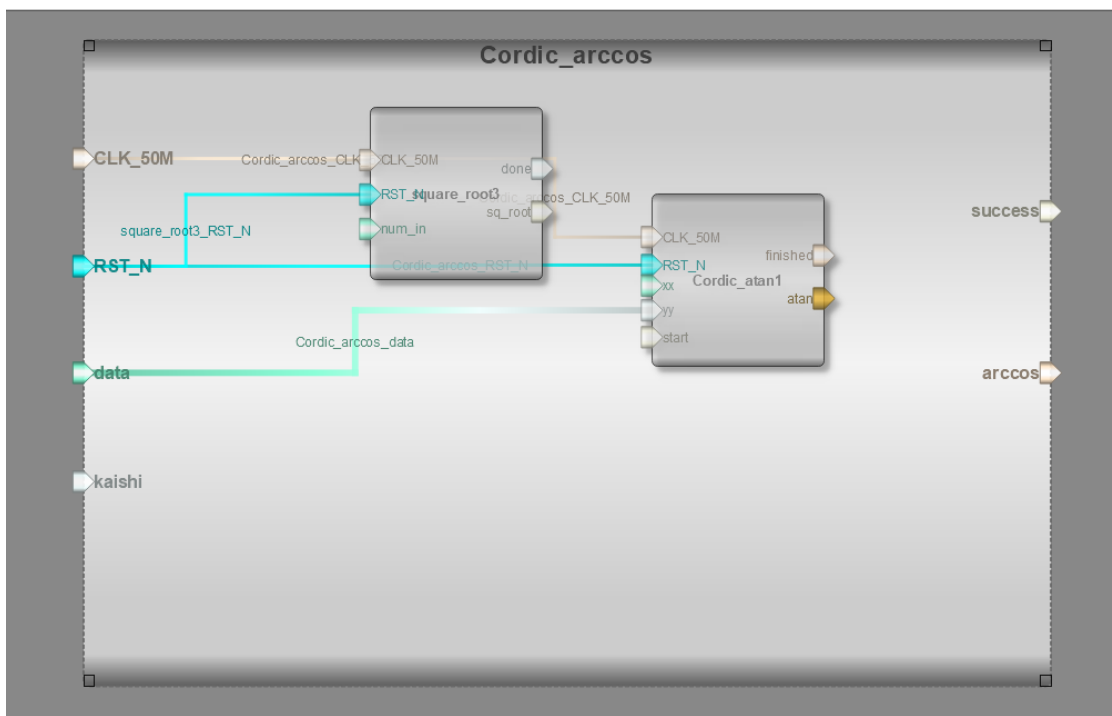


图7.2.11 Robei EDA的cordic_arccos框图

在得到这些算法之后我们就可以着手准备逆运动学算法了，而逆运动学我们用的是三角函数+向量法分析。所用到的公式/步骤为：

首先：给出预期末端执行器与地面的夹角 Alpha

其次：根据摄像头提供的 x、y 坐标算出角度 θ_6 （也就是底座旋转角度）

$$\theta_6 = \tan^{-1} \frac{x}{y} ;$$

再者：然后在根据机械臂参数，得出必要的中间值：

$$a_1 = \sqrt{x^2 + y^2} - l_3 * \cos \text{Alpha};$$

$$a_2 = z - l_0 - l_3 * \sin \text{Alpha} ;$$

$$a_3 = \cos^{-1} \frac{b_1}{\sqrt{b_1^2 + c_1^2}} ;$$

$$a_4 = \frac{(b_1^2 + c_1^2 + l_1^2 - l_2^2)}{2 * l_1 * \sqrt{b_1^2 + c_1^2}} ;$$

$$a_5 = \frac{b_1^2 + c_1^2 - l_1^2 - l_2^2}{2 * l_1 * l_2} ;$$

最后得到下面三个角度：

$$\theta_5 = \text{ccc} + \cos^{-1} \text{bbb};$$

$$\theta_4 = \cos^{-1} \text{aaa};$$

$$\theta_3 = \text{Alpha} - \theta_5 + \theta_4;$$

注： θ_2 和 θ_1 根据抓取物和环境而定。

逆运动学的 pwm 输出框图:

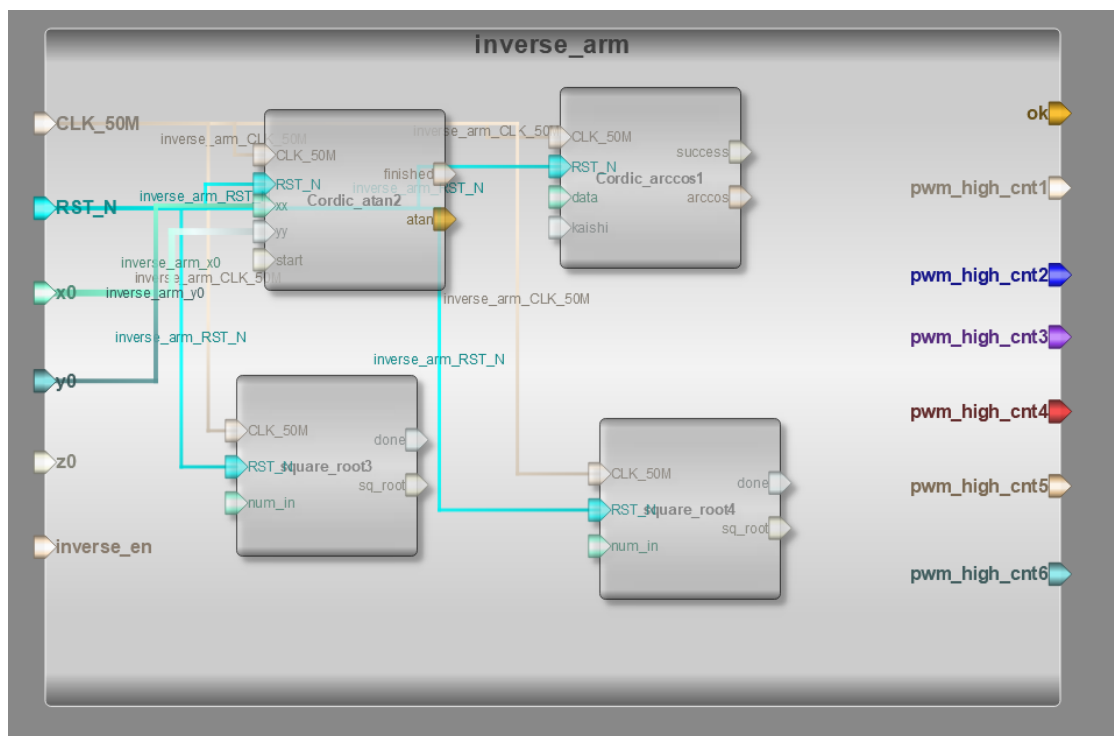


图7.2.12 Robei EDA 的inverse_arm 框图

```

244 end
245 end
246 always@(posedge CLK_50M or negedge RST_N)begin
247   if(!RST_N)
248     begin
249       pwm_high_cnt1 <= 1500;
250       pwm_high_cnt2 <= 1500;
251       pwm_high_cnt3 <= 1500;
252       pwm_high_cnt4 <= 1500;
253       pwm_high_cnt5 <= 1500;
254       pwm_high_cnt6 <= 1500;
255       ok <= 0;
256     end
257   else
258     begin
259       if(a == 5 && inverse_en)
260         begin
261           ok <= 1;
262           pwm_high_cnt1 <= 1500-(2000 * theta6 / 270);
263           pwm_high_cnt2 <= 1500+(2000 * (theta5 - 90) / 270);
264           pwm_high_cnt3 <= 1500+(2000 * theta4 / 270);
265           pwm_high_cnt4 <= 1500-(2000 * theta3 / 270);
266           pwm_high_cnt5 <= 1500;
267           pwm_high_cnt6 <= 1500;
268         end
269       else
270         begin
271           pwm_high_cnt1 <= 1500;
272           pwm_high_cnt2 <= 1500;

```

图7.2.13 逆运动学部分算法1

7.3 视觉仿真

摄像头配置部分：

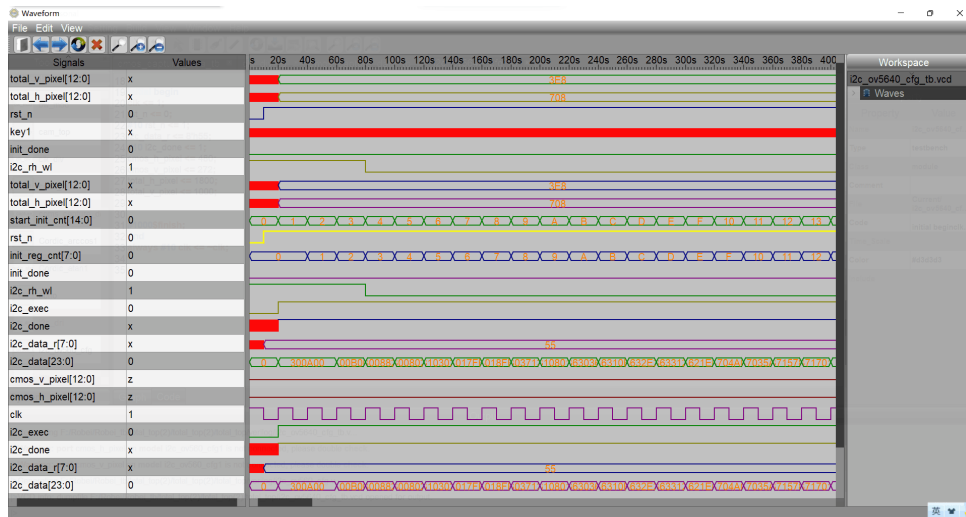


图7.3.1 摄像头配置仿真图

图像尺寸仿真部分：

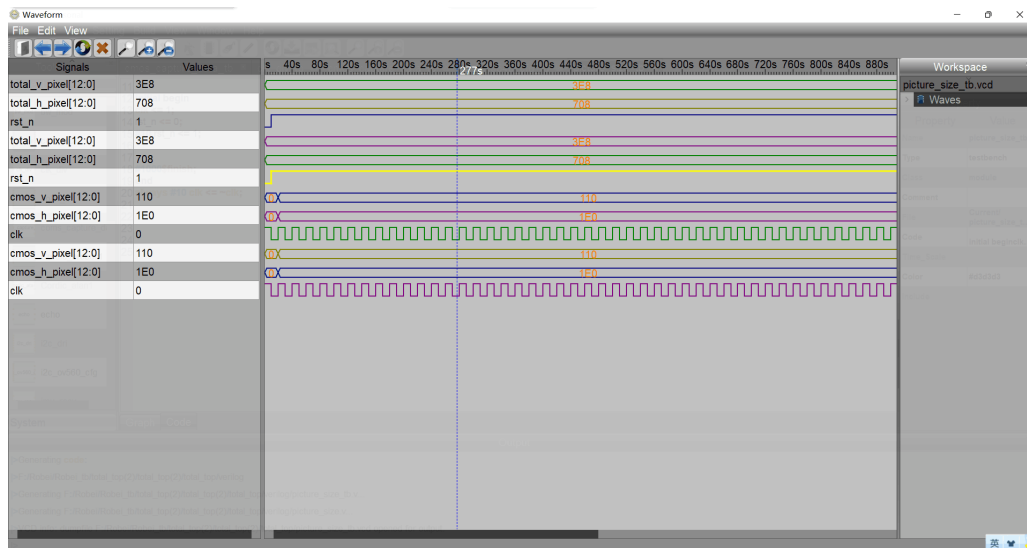


图7.3.2图像尺寸仿真图

GRB转YCbCr部分：

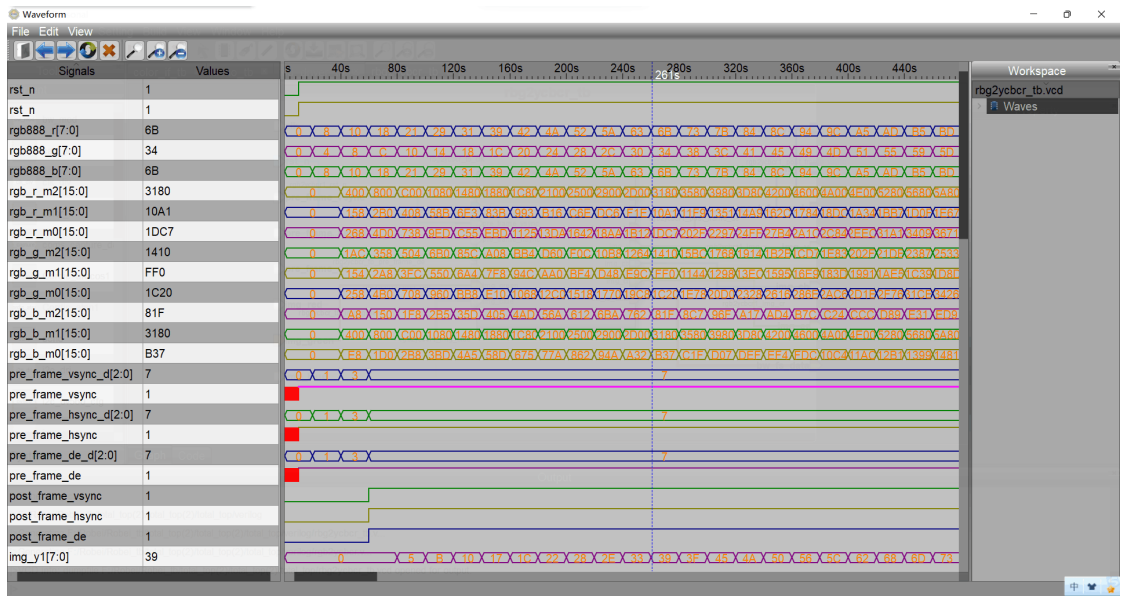


图7.3.3 RGB转YCbCr仿真图

颜色识别部分：

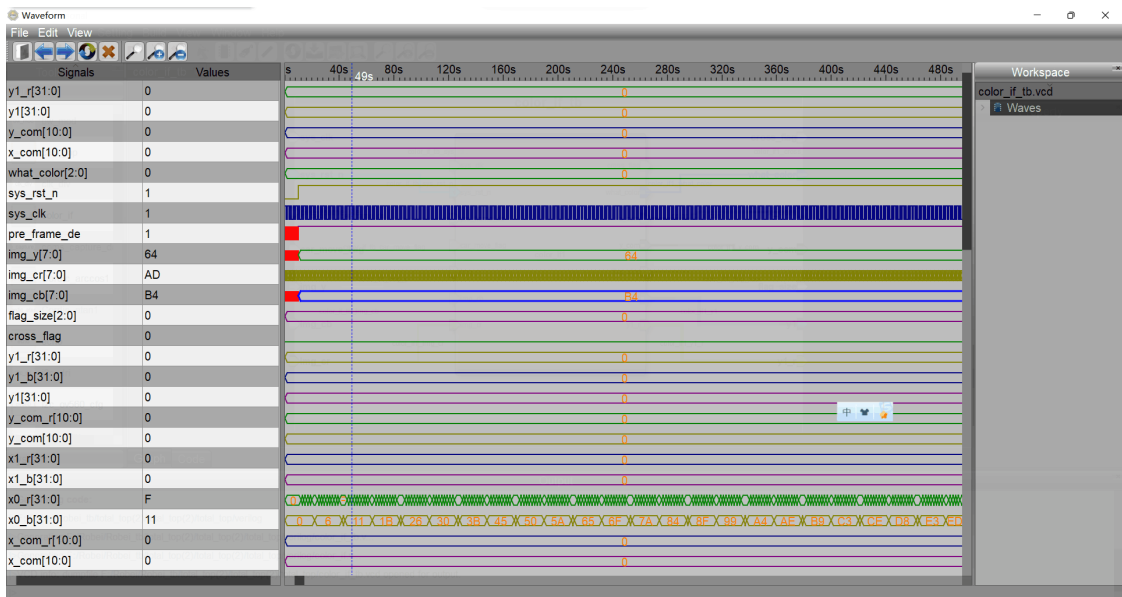


图7.3.4 颜色识别仿真图

二值化与边缘检测部分：

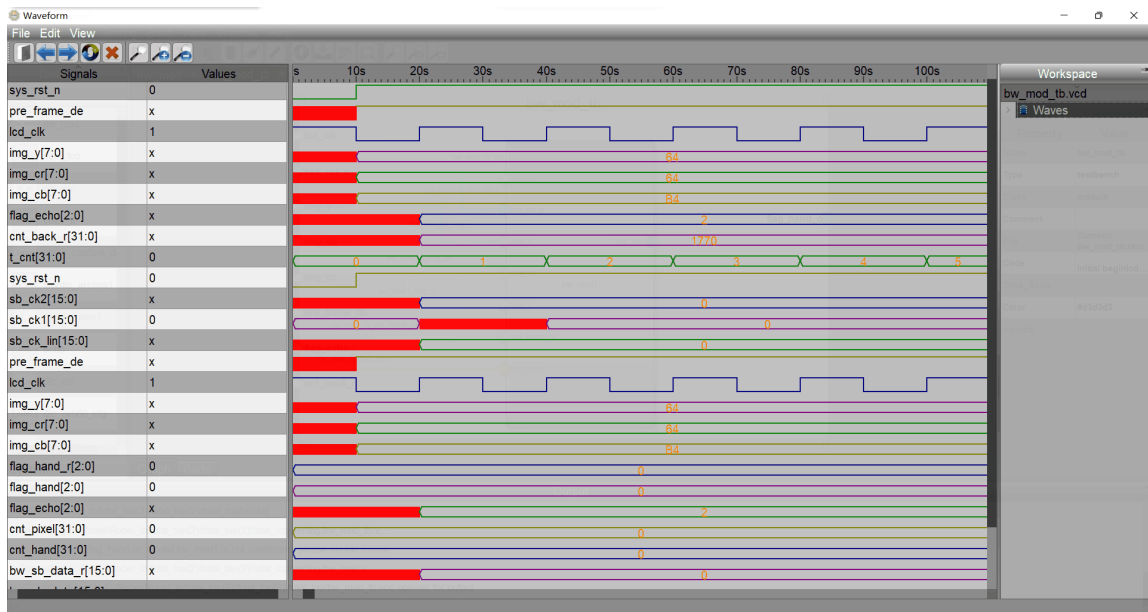


图7.3.5二值化和边缘检测仿真图

屏幕驱动部分：

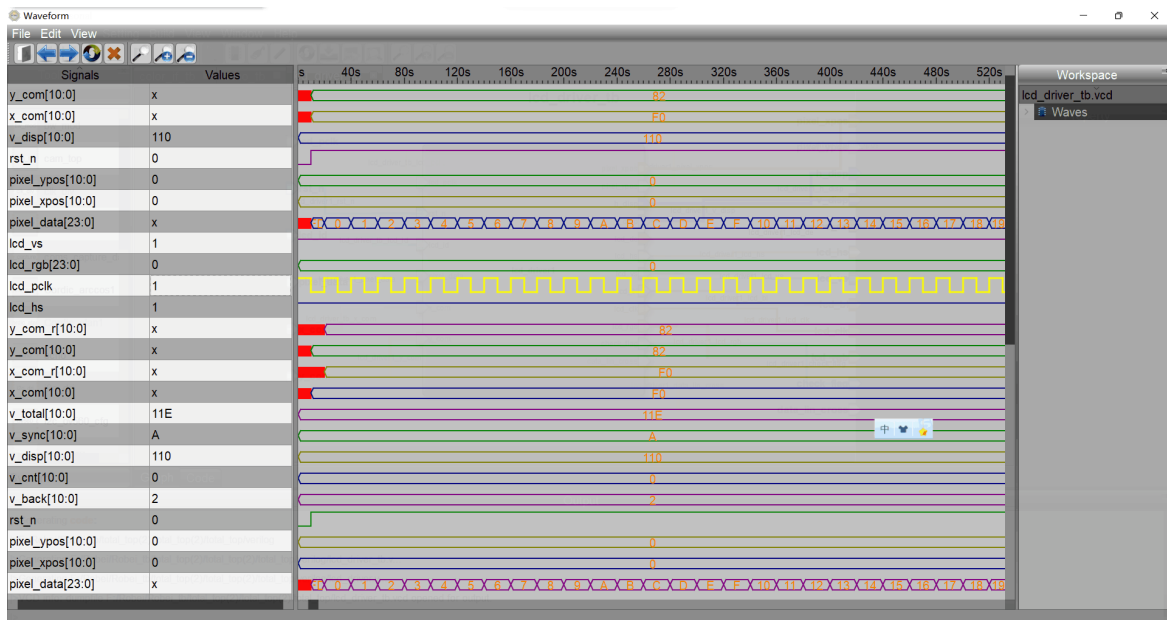


图7.3.6 屏幕驱动仿真图

超声波部分：

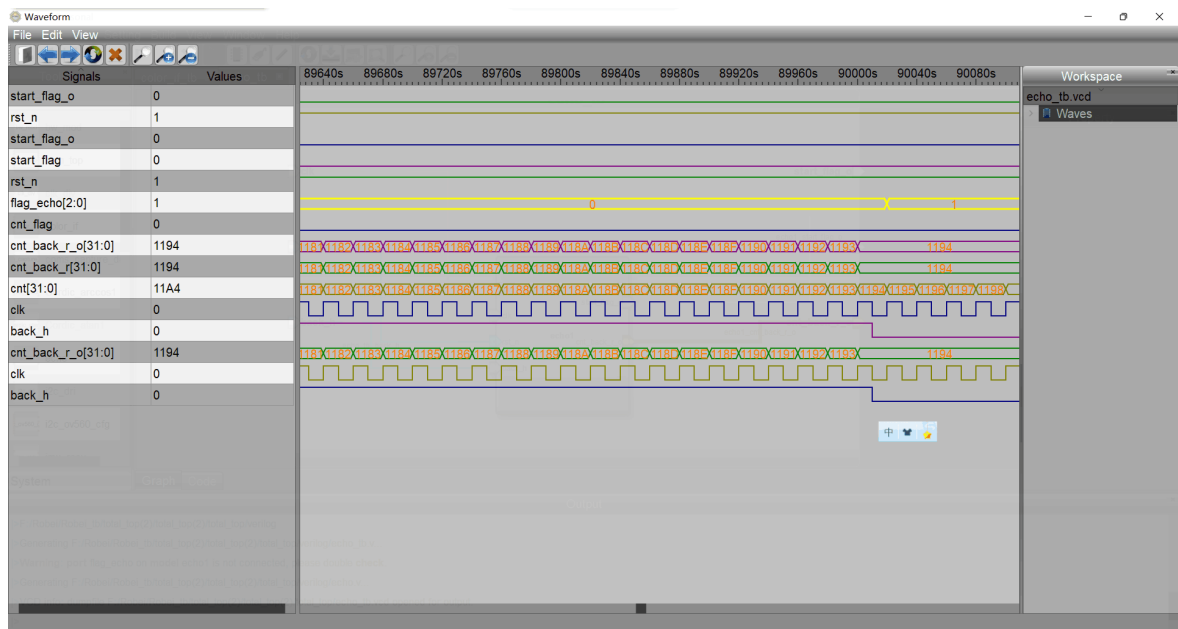


图7.3.7超声波仿真图

八、未来发展与提升

8.1 IMU 姿态融合空间定位算法

现阶段使用的 imu 直接输出的三维位置信息会因为 imu 自带的求解算法中的加速度迭代导致误差产生较大累积，我们希望在未来使用误差较小的姿态信息融合编码器的数据建立的空间定位算法。使得导航定位更为准确。

8.2 语音控制温湿度传感器

在未来我们会添加更多的传感器以及语音控制，增强整套系统功能的全面性。

九、总结（Conclusion）

通过本次比赛，我们学习使用了 Robei EDA 软件，软件灵活小巧但功能强大，尤其是用图形化界面下的连线功能，非常的便捷好用，让我们再一次的感受到了国产 EDA 工具的灵活便捷，对于我们初学者非常友好。

在查找相关资料时，我们了解了当前机器人产业的技术发展水平、应用方向以及市场规模。我们学习了解了机器人相关的控制方式、传感器应用、定位技术、滤波算法、数据融合处理、图像处理等一系列的知识，同时对于一些机器人发展的新趋势有了更多的认识，这些对于我们以后的学习有着指导性的意义。

在当下的时代背景下，我们自己的 EDA 软件正在起势，预计在不久的将来 EDA 软件将会打破三大巨头的垄断，同时在中国市场的催生下，新型机器人技术也会逐渐走到世界前列。

十、参考资料（References）

- [1] 张伟,张建,高岩,孙天澳,李旭升,王云亮.基于 FPGA 的增量式 PID 算法的设计与改进[J].电子测试, 2017(19):41-42+45.DOI:10.16520/j.cnki.1000-8519.2017.19.019.
- [2] 裴正雄,彭安金.基于 FPGA 和 OV5640 的图像采集和处理系统设计[J].机电信息,2019(32):159-160.DOI:10.19514/j.cnki.cn32-1628/tm.2019.32.089.
- [3] 吴国盛.7 天搞定 FPGA:Robei 与 Xilinx 实战[M]. 电子工业出版社,2016.
[4]吴万欣,田军委,丁良华,张云辉.六自由度机械臂运动学分析与仿真[J].天津理工大学学报,2022,38(01):8-12+22.
- [4] 温芮,陈锦鸿,王丽.基于蓝牙控制技术的智能小车控制系统设计[J].汽车零部件,2019(12):1-4.DOI:10.19466/j.cnki.1674-1986.2019.12.001.
- [5] 罗冬旭. 基于 FPGA 的超声波测距系统设计[D].吉林大学,2021.DOI:10.27162/d.cnki.gjlin.2021.003492.
- [6] 于波,陈先瑞,张强,李建成.基于 FPGA 的印刷体数字采集识别系统[J].电子测量技术,2021,44(16):7-11.DOI:10.19651/j.cnki.emt.2107098.
- [7] 赵林军. 流水线CORDIC算法的FPGA实现[J]. 电脑知识与技术, 2008 (30) :716-717.

