燕山大学

专 业 技 能 实 践 说 明 书

题　　目：　智能化产线单元设计与制作——带取样 AGV

学院（系）：　机械工程学院

年级专业：　18级机械电子工程

组　　号：　5-履带1组

学　　号	姓　　名	成　　绩
201811010426	陈嘉琛（70%）	
201811010240	王玥灏（10%）	
201811010022	刘欣源（10%）	
201811010241	徐世杰（10%）	

指导教师：　张连东　李玉昆

日　　期：　2021.12

燕山大学专业技能实践任务书

院（系）：机械工程学院　　基层教学单位：机械电子工程系

组号	5	学生姓名	陈嘉琛
题目	智能化产线系统集成—履带取样 AGV		
实践目标	基于《机电产品设计》二级项目制作的智能机器人机械本体、传感检测、驱动及控制系统，充分查阅文献，基于工程需求、作业任务、运动轨迹、信息交互等任务，同时考虑社会、安全、法律及环境等因素进行产品再开发，通过编写完整的程序，进行机器人运动控制规划，对作业过程进行模拟仿真，最终高效地完成既定的协同作业任务。		
实践要求	（1）运用现代信息技术获取产品相关信息； （2）动作特点能够充分展示机器人或 AGV 系统的结构特点，体现系统化设计思想； （3）整体方案要体现工作效率和经济性； （4）充分体现协同作业和团队合作； （5）动作时间 3-5 分钟。		
工作量	（1）资料查阅、方案准备、素材整理； （2）详细分析机器人系统的硬件和软件功能，制定运动控制方案； （3）完成机器人动作的总体规划； （4）编制程序设计流程图； （5）编写程序代码，控制机器人实现规划的动作； （6）程序调试、预演； （7）撰写专业综合训练说明书； （8）完成机器人规划动作的现场展示。		
工作计划	（1）资料查阅、分析总结，所需天数 1 天 （2）运动控制方案设计，所需天数 2 天 （3）机器人动作的总体规划，所需天数 2 天 （4）程序流程图的设计，所需天数 1 天 （5）控制程序编码，所需天数 4 天 （6）程序调试，所需天数 2 天 （7）撰写综合训练说明书，所需天数 2 天 （8）答辩考核、演示，所需天数 1 天		
参考资料	通过校园网在我校订阅的电子资料库中可以搜索到大量的有关运动控制、单片机编程的参考资料。同学们也可到学校的图书馆查找纸质期刊资料。机械学院的计算机中心上班时间免费对本院学生开放，查阅外网资料。		
指导教师签字		基层教学单位主任签字	

说明：此表一式四份，学生、指导教师、基层教学单位、系部各一份。

年　月　日

燕山大学专业综合训练评审意见表

指导教师评语：   成绩：_____ 指导教师：_____ 年　　月　　日
答辩小组评语：   成绩：_____ 评阅人：_____ 年　　月　　日
课程设计总成绩：
答辩小组成员签字：   年　　月　　日

目录

第 1 章　产线工作方案介绍 6
1.1 产线简介 6
1.1.1 硬件系统框图 6
1.1.2 各单元硬件功能 6
1.2 产线单元的设计与实现 7
1.2.1　陈嘉琛同学方案设计及分析 7
1.2.2　王玥灏同学方案设计及分析 7
1.2.3　刘欣源同学方案设计及分析 7
1.2.4　徐世杰同学方案设计及分析 8
1.2.5　小组方案讨论 8
1.2.6　小组最终方案 8
第 2 章　单元动作的总体规划详细方案 9
第 3 章　编制程序设计流程及框图 9
第 4 章　编程实现规划动作并编写程序代码 11
4.1 直流电机驱动模块（APO-L1） 11
4.2 步进电机驱动模块（A4988） 13
4.3 末端手爪驱动模块（MG90S） 17
4.4 超声波模块（HC-SP04） 17
4.5 显示模块（OLED） 18
4.6 扫码模块（SH-50） 20
4.7 红外寻迹模块（四路红外循迹模块） 20
4.7 Wifi 蓝牙模块（ESP32） 23
第 5 章　程序调试结果及改进 25
第 6 章　项目思政心得体会 27
参考文献 28
附录 1　程序源代码 29

摘要

AGV 即自动导引小车,它多位一体,集结了当今科技领域先进的理论和应用技术,综合了声、光、电、计算机等技术。AGV 具有灵活可靠、柔性化好、效率高、操作简便、可控性强等许多优点,因此在柔性制造系统和自动化工厂中得到了广泛的应用,极大地提高了企业的生产制造效率和自动化程度。将来的 AGV 必然向着更为简捷高效的方向发展,会得到进一步推广,其应用领域也必然会随之变的更加广泛。

本项目综合研究分析了国内外 AGV 的发展情况和现状,并根据所学知识自主设计一辆履带式 AGV 小车。主要设计工作有:机械总体方案设计（履带式底盘和机械臂的设计或选型）、驱动电机的选取（底盘运行电机及机械臂运动所需电机）、各模块功能设计选取（传感器、32 系统板、减压模块等）、各模块的实际连接设计、各零部件的强度校核、基本控制系统分析、软件系统整体方案设计、硬件电路分析与绘制、小车运动学和动力学分析、控制系统设计以及路径行驶规划策略。最终设计的物流小车能够实现自主导引运行、运动轨迹的控制、自动循迹及避障、物联网监控组态、按照要求抓取和运输工件等功能,目标是使其能够模拟现有的工厂实际使用的 AGV 物料运输小车，并绘制各零件及外部电路原理图、机械装配图及相应的报告，熟悉前沿自动化控制与物联网监控技术，完成此次项目。

关键词: 履带 AGV 小车　运动分析　自动控制　机械设计 物联网

第 1 章　产线工作方案介绍

1.1　产线简介

1.1.1　硬件系统框图

在整个产线的工作任务中，以 STM32F103C8T6 最小系统板为主控板，接收超声波模块、四路红外循迹模块、扫码模块的信息，去控制履带底盘、机械臂、末端执行机构的运行。

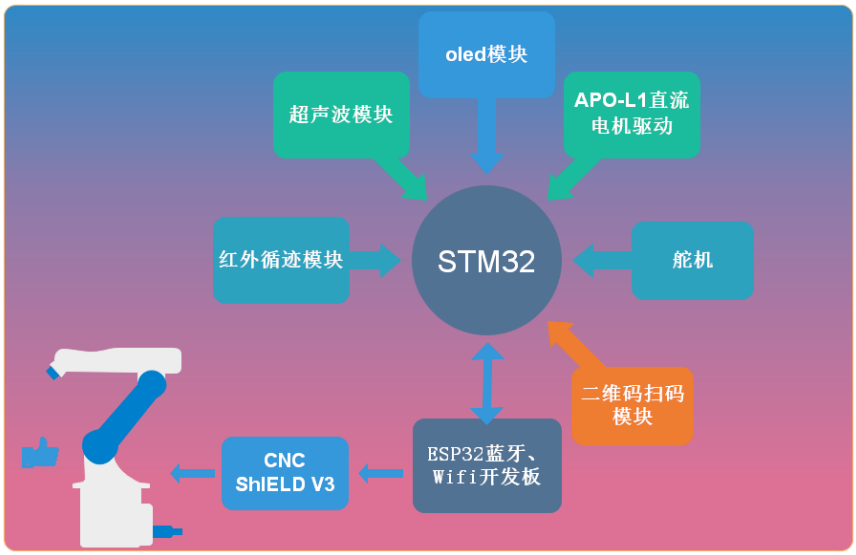

图 1.1 硬件系统框图

1.1.2　各单元硬件功能

表 1 各单元硬件功能

硬件	型号	功能
单片机	STM32F103C8T6	主控单元，协调各模块的工作
固定降压模块	MP1584EN（固定 3.3V）	将电池提供的 12V 降为 3.3V，为板载模块供电
可调降压模块	MP1584EN（0.8~22V 可调）	为舵机组接口单独供电，保证舵机的稳定运行
直流电机驱动模块	APO-L1	驱动底盘的两个直流电机
步进电机驱动模块	A4988	驱动机械臂的三个步进电机
末端手爪驱动模块	MG90S	驱动末端手爪的张合
红外循迹模块	四路红外循迹模块	使小车检测并沿地面黑色迹线行驶
超声波模块	HC-SR04	检测前方物体距离，以便停车避障
扫码模块	SH-50	识别二维码，获取所需夹取的物料信息
显示模块	0.96inc OLED（IIC 显示屏）	将关键信息显示，以便查看
Wifi 蓝牙模块	ESP32	远程操控机械臂

1.2 产线单元的设计与实现

由于前期已经进行过各模块的选型，因此现阶段主要是根据场地任务要求，确定小车的运行情况。

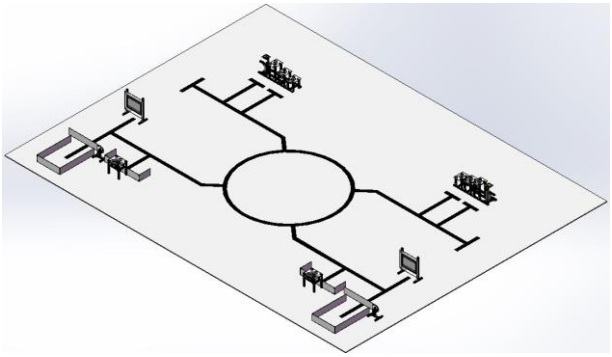

图 1.2 简化的场地模型

1.2.1 陈嘉琛同学方案设计及分析

AGV 履带小车产线单元，以 STM32F103C8T6 最小系统板为主控板，搭配超声波模块、四路红外循迹模块、扫码模块，等外部传感设备，直流电机及其驱动器控制履带底盘，步进电机及其驱动器控制机械臂和末端执行机构的运行。

开启开关后，超声波模块检测到开门后，STM32F103C8T6 最小系统板控制小车沿着红外循迹模块识别到的路线到达扫码区扫描二维码，二维码中包含小车需要夹取货物的信息。

然后继续寻迹至仓库区，使用自己设计的机械手按照二维码信息抓取预先放置好的物块，接着小车携带货物沿预定路线返回。寻迹至组装区，小车在平台中心放料，完成后返回仓库区重复取放货 2 次。

1.2.2 王玥灏同学方案设计及分析

AGV 履带小车产线单元，以 STM32F103C8T6 最小系统板为主控板，搭配超声波模块进行避障功能、摄像头模块进行循迹和扫码功能，直流电机及其驱动器控制履带底盘，步进电机及其驱动器控制机械臂和末端执行机构的运行。开启开关后，超声波模块检测到开门后，STM32F103C8T6 最小系统板控制小车沿着摄像头模块识别到的路线到达扫码区用摄像头扫描二维码，二维码中包含小车需要夹取货物的信息。然后摄像头继续寻迹至仓库区，使用自己设计的机械手按照摄像头扫描到的二维码信息抓取预先放置好的物块，接着小车携带货物沿预定路线返回。寻迹至组装区，小车在平台中心放料，完成后返回仓库区重复取放货 2 次。

1.2.3 刘欣源同学方案设计及分析

AGV 履带小车产线单元，以 STM32F103C8T6 最小系统板为主控板，搭配超声波模块、四路红外循迹模块、扫码模块，等外部传感设备，直流电机及其驱动器控制履带底盘，步进电机及其驱动器控制机械臂和末端执行机构的运行。

开启开关后，小车通过红外循迹模块识别并跟踪路线到达扫码区扫描二维码获取所需夹取的物料，然后继续寻迹至仓库区，使用自己设计的机械手按照二维码信息抓取预

先放置好的物块，将所有需要夹取的物块放置小车上的货舱中，接着小车携带货物沿预定路线返回。寻迹至组装区，小车将货舱中的物料放至平台中心，完成后返回至启动区。

1.2.4 徐世杰同学方案设计及分析

AGV 履带小车产线单元，以 STM32F103C8T6 最小系统板为主控板，搭配超声波模块、四路红外循迹模块、扫码模块，等外部传感设备，直流电机及其驱动器控制履带底盘，步进电机及其驱动器控制机械臂和末端吸盘的运行。

末端执行机构采用吸盘来进行对物块的抓取和释放，在接触物体时气泵吸气，吸盘内部产生气压，将物块牢牢吸住，到达目标地点后放气，释放物块。采用气泵的优点有：排放绿色无污染，物块抓取的程序设计更为简单，操作更方便。

1.2.5 小组方案讨论

陈嘉琛同学：王玥灏同学的方案中采用摄像头识别路线的方式虽然对于路线的跟踪更与加明确和清晰，但较红外循迹而言过于复杂，不便于对设备的设计与调试。

王玥灏同学：刘欣源同学的方案一次抓取两个物块，对于大小物块的识别和抓取有一定困难，且在小车上装货舱会给小车增加负重，若发生物块倾斜甚至倒下的情况，会严重影响整体的完成速度。

刘欣源同学：徐世杰同学的方案中将夹爪换成了吸盘，吸盘需要增加检查整块的密封性，在抓取物块时不能发生偏移，否则会发生不能将物块抓起的情况，若物块不规则，吸盘将不能将物块抓起。

徐世杰同学：陈佳琛同学的方案需要往返抓取物块 2 次，严重增长了整体完成的时间，远不如采用货舱的方式简便节省时间。

1.2.6 小组最终方案

我们最终决定采用以 STM32F103C8T6 最小系统板为主控板，搭配超声波模块、四路红外循迹模块、扫码模块，等外部传感设备，直流电机及其驱动器控制履带底盘，步进电机及其驱动器控制机械臂和末端执行机构的运行。采用末端机械爪并添加货舱只需一次往返就完成的方案。

第 2 章　单元动作的总体规划详细方案

由图 1.2 所示，整个场地主要由数个转弯以及中间的圆形环岛迹线组成，由此规划履带小车的整体动作。

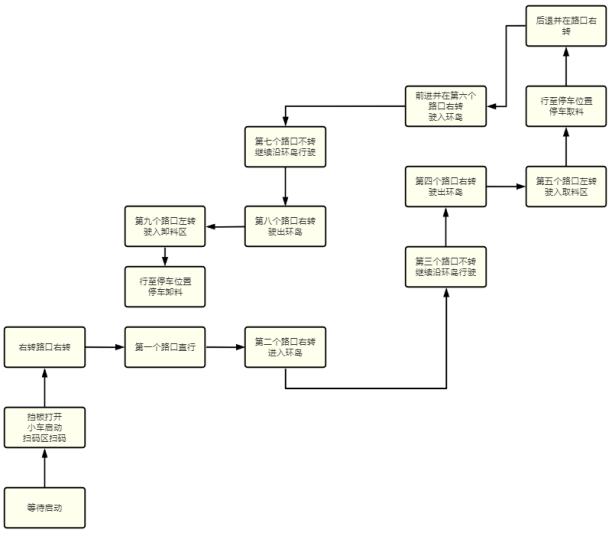

图 2.1 总体规划框图

第 3 章　编制程序设计流程及框图

根据场地组发布的任务要求，AGV 小车的工作流程为：

1、AGV 小车在启动区准备，舵机开门，开始计时，小车出发沿黑线寻迹，前往扫码区扫描二维码，二维码中包含小车需要夹取货物的信息。

2、寻迹至仓库区，小车到达取货区，使用自带的机械手抓取预先放置好的物块，小车携带货物返回。

3、寻迹至组装区，小车在平台中心放料，完成后返回仓库区重复取放货 2 次。

4、往返两次后，装箱结束，小车回到出发点，计时结束。

对小车的要求为：

1、小车尺寸，长：不超过 550mm 宽：不超过 360mm：高能正常抓取工件即可。小车具有机械手用于工件抓取和搬运，手爪抓取最远距离参考需要结合取放货的空间位置和自身车体大小设计，夹爪抓取物料最小重量为 150g。

2、小车应具有二维码扫描模块和黑线寻迹模块，自行选择红外避障模块和超声波测距模块等其他传感器。

从 AGV 小车的工作流程及相关的硬件要求中，得出编程的流程：

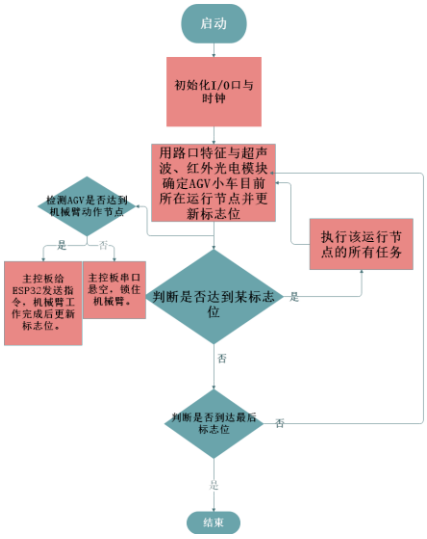

图 3.1 程序设计流程框图（陈嘉琛）

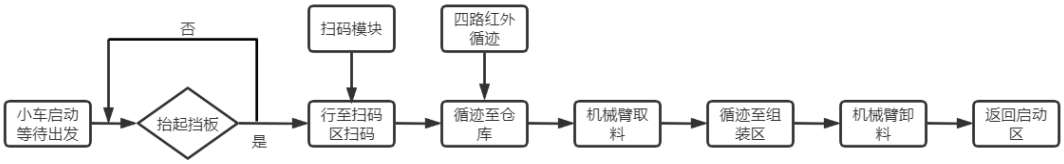

图 3.2 程序设计流程框图（王玥灏）

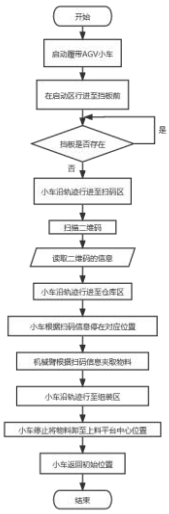

图 3.3 程序设计流程框图（刘欣源）

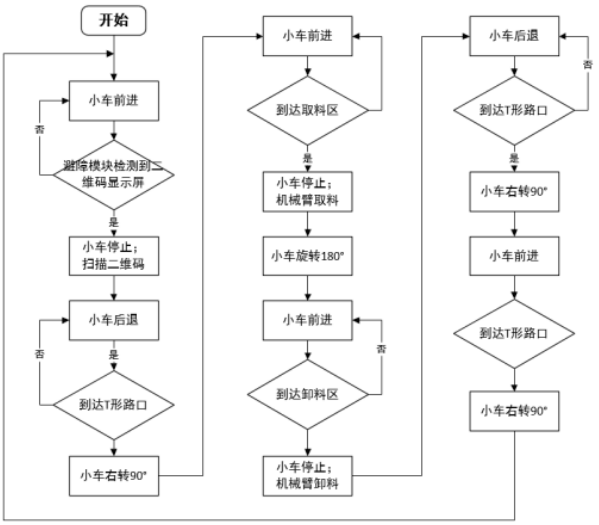

图 3.4 程序设计流程框图（徐世杰）

第4章 编程实现规划动作并编写程序代码

表 2 各硬件模块及负责人

负责人	硬件	型号
陈嘉琛	直流电机驱动模块	TB6612FNG
	步进电机驱动模块	A4988
	Wifi 蓝牙模块	ESP32
王玥灏	超声波模块	HC-SR04
	扫码模块	SH-50
刘欣源	显示模块	0.96 寸 OLED（IIC 显示屏）
	红外寻迹模块	四路红外循迹模块
徐世杰	红外循迹模块	四路红外循迹模块
	末端手爪驱动模块	MG995

4.1 直流电机驱动模块（APO-L1）

直流电机驱动模块用来驱动履带底盘的两个直流减速电机，该模块如图 4.1 所示。

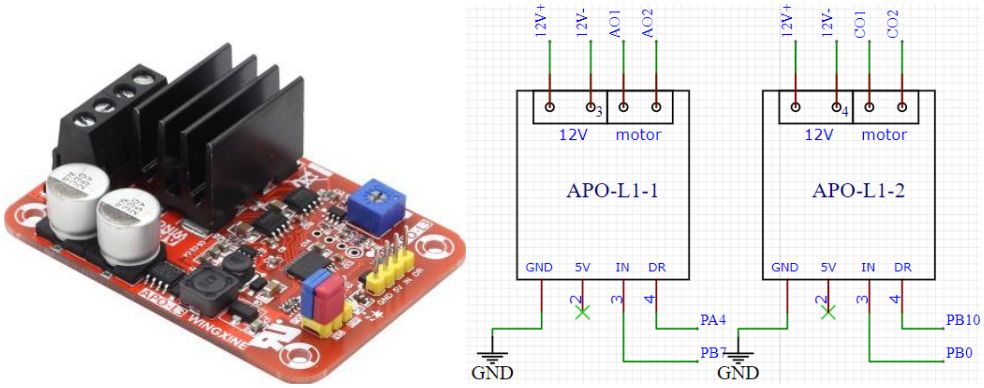

图 4.1 直流电机驱动模块及电路图

驱动程序：

//定义引脚使主程序更加简单易懂

```
#define LEFT_MOTOR_PIN              GPIO_PIN_4
#define LEFT_MOTOR_GPIO             GPIOA

//右电机的两个IO分配
//右电机1号IO
#define RIGHT_MOTOR_PIN              GPIO_PIN_10
#define RIGHT_MOTOR_GPIO             GPIOB
```

//小车停止

```
void STOP(int C_COUNTER,int TIME)
{     //左电机
  //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);      //开启左电机的pwm调压
  //HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);      //开启右电机的pwm调压

  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER);    //设定左电机的占空比=C_COUNTER/1000
  __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER);    //设定右电机的占空比=C_COUNTER/1000
   HAL_Delay(TIME);
  // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);     //关闭左电机的pwm输出，防止怠速运行
   //HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);     //关闭右电机的pwm输出，防止怠速运行
}
```

//小车左转 90°

```
void left_90(int C_COUNTER_L,int C_COUNTER_R,int TIME)
{     //左电机
  //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);      //开启左电机的pwm调压
  //HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);      //开启右电机的pwm调压
  HAL_GPIO_WritePin(LEFT_MOTOR_GPIO , LEFT_MOTOR_PIN, GPIO_PIN_SET); //左轮反转       /* 设定IO的值使左转电机正转  */
  HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO, RIGHT_MOTOR_PIN, GPIO_PIN_RESET);        /* 设定IO的值使右转电机正转  */

  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER_L);    //设定左电机的占空比=C_COUNTER/1000
  __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER_R*1.12);    //设定右电机的占空比=C_COUNTER/1000

  HAL_Delay(TIME);
  // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);     //关闭左电机的pwm输出，防止怠速运行
   //HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);     //关闭右电机的pwm输出，防止怠速运行
}
```

//小车右转 90°

```
void right_90(int C_COUNTER_L,int C_COUNTER_R,int TIME)
{     //左电机
  //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);      //开启左电机的pwm调压
  //HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);      //开启右电机的pwm调压
  HAL_GPIO_WritePin(LEFT_MOTOR_GPIO , LEFT_MOTOR_PIN, GPIO_PIN_RESET); //左轮反转       /* 设定IO的值使左转电机正转  */
  HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO, RIGHT_MOTOR_PIN, GPIO_PIN_SET);         /* 设定IO的值使右转电机正转  */

  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER_L);   //设定左电机的占空比=C_COUNTER/1000
  __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER_R);   //设定右电机的占空比=C_COUNTER/1000

  HAL_Delay(TIME);
  // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);     //关闭左电机的pwm输出，防止怠速运行
   //HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);     //关闭右电机的pwm输出，防止怠速运行
}
```

//小车掉头

```
void rotate_180(int C_COUNTER_L,int C_COUNTER_R,int TIME)
{     //左电机
  //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);      //开启左电机的pwm调压
  //HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);      //开启右电机的pwm调压
  HAL_GPIO_WritePin(LEFT_MOTOR_GPIO , LEFT_MOTOR_PIN, GPIO_PIN_RESET); //左轮反转       /* 设定IO的值使左转电机正转  */
  HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO, RIGHT_MOTOR_PIN, GPIO_PIN_SET);         /* 设定IO的值使右转电机正转  */
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER_L);   //设定左电机的占空比=C_COUNTER/1000
  __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER_R);   //设定右电机的占空比=C_COUNTER/1000

  HAL_Delay(TIME);
  // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);     //关闭左电机的pwm输出，防止怠速运行
   //HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);     //关闭右电机的pwm输出，防止怠速运行
}
```

4.2 步进电机驱动模块（A4988）

步进电机驱动模块用来驱动机械臂的三个步进电机，控制机械臂的动作，该模块如图 4.5 所示。

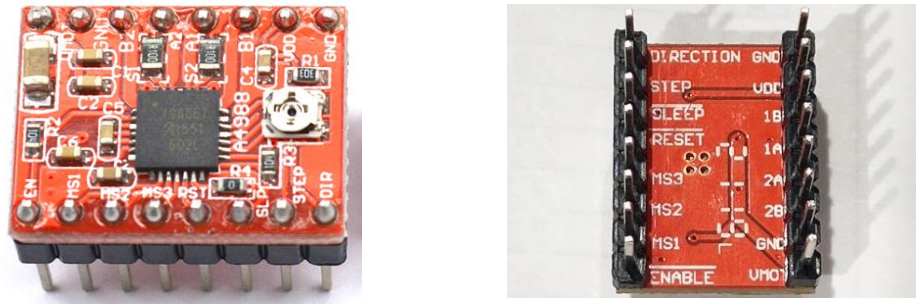

图 4.2 步进电机驱动模块（A4988）

对于 A4988 来说，ENA 为使能端，低电平有效；STEP 为步进电机信号的输入端，可以采用 PWM 波输入，也可以采用 IO 引脚高低电平转换；MS1、MS2、MS3 控制一个脉冲旋转多少度，分别是全步进，1/2 步进，1/4 步进，1/8 步进，1/16 步进模式，如表所示；DIR 为步进电机方向控制 IO 高电平为顺时针，低电平为逆时针；VDD 为模块的逻辑电压 3V3 或 5V，VDD 必须接电；VMOT 为步进电机的输入电压（8V-35V）；1A、1B 为同一相，2A、2B 为同一相，不能接反，否则电机会出现抖动的情况。对于步进电机同一相的确定方法是：将电压表打到蜂鸣器档位，将正负表笔随意接步进电机的两个引脚，若蜂鸣器响起，则该两引脚为一相。

综上，控制 MS1、MS2、MS3、STEP、DIR 就可以控制一个步进电机的运行了，甚至只用 STEP 就可以了。

引脚	MS1	MS2	MS3	步进模式
引脚	L	L	L	全步进
电平	H	L	L	1/2 步进

	L	H	L	1/4 步进
	H	H	L	1/8 步进
	H	H	H	1/16 步进

//将接收二维码传递的数据进行转换，确定夹取物块的顺序

```
// 新建组件对象
BlinkerButton Button1( (char*)"key1" );
BlinkerButton Buttonre0( (char*)"btn-0" );

BlinkerNumber Num1( (char*)"num-1" );
BlinkerNumber Num_D1( (char*)"num-D1" );
BlinkerNumber Num_D2( (char*)"num-D2" );
BlinkerNumber Num_D3( (char*)"num-D3" );

// BlinkerSlider bar1( (char*)"bar1" );
// BlinkerSlider bar2( (char*)"bar2" );
// BlinkerSlider bar3( (char*)"bar3" );

int16_t D1=0;int16_t D2=0;int16_t D3=0;
int16_t getbig_deg1[2];
int16_t getbig_deg2[2];
int16_t getbig_deg3[2];
int16_t getsmall_deg1[2];
int16_t getsmall_deg2[2];
int16_t getsmall_deg3[2];
int16_t savebig1_deg1[2];
int16_t savebig1_deg2[2];
int16_t savebig1_deg3[2];
//int16_t savebig2_deg1;int16_t savebig2_deg2;int16_t savebig2_deg3;
int16_t savesmall1_deg1[2];
int16_t savesmall1_deg2[2];
int16_t savesmall1_deg3[2];
//int16_t savesmall2_deg1;int16_t  savesmall2_deg2;int16_t  savesmall2_deg3;
int16_t put_deg1[2];
int16_t put_deg2[2];
int16_t put_deg3[2];
```

```
case 2://取用物块

  while(digitalRead(getinpin1)==1)
  {
   //抓放大底盘
   motor_run(getbig_deg1[0],getbig_deg2[0],getbig_deg3[0]);//抓大底盘
   delay(500);
   motor_run(getbig_deg1[1],getbig_deg2[1],getbig_deg3[1]);//抓大底盘
   delay(500);
   myservo.write(40);
   delay(500);
   motor_run(savebig1_deg1[0],savebig1_deg2[0],savebig1_deg3[0]);//放大底盘
   delay(500);
   motor_run(savebig1_deg1[1],savebig1_deg2[1],savebig1_deg3[1]);//放大底盘
   delay(500);
   myservo.write(0);
   delay(500);

   //抓放小物块
   motor_run(getsmall_deg1[0],getsmall_deg2[0],getsmall_deg3[0]);//抓大底盘
   delay(500);
   motor_run(getsmall_deg1[1],getsmall_deg2[1],getsmall_deg3[1]);//抓大底盘
   delay(500);
   myservo.write(80);
   delay(500);
   motor_run(savesmall1_deg1[0],savesmall1_deg2[0],savesmall1_deg3[0]);//放大底盘
   delay(500);
   motor_run(savesmall1_deg1[1],savesmall1_deg2[1],savesmall1_deg3[1]);//放大底盘
   delay(500);
   myservo.write(0);
   delay(500);

   //
  }
  roboflag++;
  break;
```

```
  case 3://放回物块

  while(digitalRead(getinpin1)==1)
  {
    //抓放小物块
    motor_run(getsmall_deg1[0],getsmall_deg2[0],getsmall_deg3[0]);//抓小物块
    delay(500);
    motor_run(getsmall_deg1[1],getsmall_deg2[1],getsmall_deg3[1]);//抓小物块
    delay(500);
    myservo.write(80);
    delay(500);
    motor_run(put_deg1[0],put_deg2[0],put_deg3[0]);//放小物块
    delay(500);
    motor_run(put_deg1[1],put_deg2[1],put_deg3[1]);//放小物块
    delay(500);
    myservo.write(0);
    delay(500);


    //抓放大底盘
    motor_run(getbig_deg1[0],getbig_deg2[0],getbig_deg3[0]);//抓大底盘
    delay(500);
    motor_run(getbig_deg1[1],getbig_deg2[1],getbig_deg3[1]);//抓大底盘
    delay(500);
    myservo.write(40);
    delay(500);
    motor_run(put_deg1[0],put_deg2[0],put_deg3[0]);//放大底盘
    delay(500);
    motor_run(put_deg1[1],put_deg2[1],put_deg3[1]);//放大底盘
    delay(500);
    myservo.write(0);
    delay(500);

    motor_return_to_zero();
  }
    roboflag++;
    break;

default:
  break;
}

Blinker.run();
```

4.3 末端手爪驱动模块（MG995）

输入信号脉冲宽度	舵机输出轴转角
0.5ms	-90°
1ms	-45°
1.5ms	0°
2ms	45°
2.5ms	90°

图 4.3 舵机转角数据图

图 4.4 末端手抓驱动模块

程序已加入步进电机的夹取部分，详细如 4.2 步进电机驱动模块（A4988）所述程序

4.4 超声波模块（HC-SP04）

超声波模块装在履带底盘小车的前方，用来检测与前方物体的距离，以便小车及时减速避让或停车。该模块如图 4.6 所示。

图 4.5 超声波模块（HC-SR04）

超声波模块程序的编写参考图 4.7 的时序图，时序图表明，只需要提供一个 10uS 以

上脉冲触发信号，该模块内部将发出 8 个 40kHz 周期电平并检测回波。一旦检测到有回波信号则输出回响信号。回响信号的脉冲宽度与所测的距离成正比。由此通过发射信号到收到的回响信号时间间隔可以计算得到距离。计算公式如下：

$$distance = Time * 340 * 1000/2(mm)$$

其中，distance 为距离，Time 为高电平时间。

超声波模块引脚的连接如下：

VCC—5V	GND—GND
Trig—输出引脚（PA8）	Echo—输入引脚（PA15）

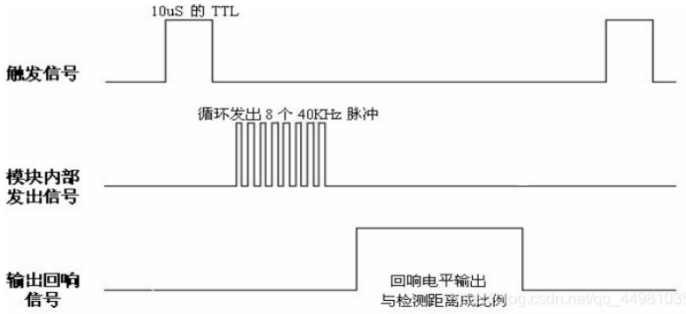

图 4.6 超声波模块时序图

根据时序图编写超声波模块的驱动函数如下：

```
#include "HC_SR04.h"

//声明一个模块的数据结构体
HCSR04 hcsr04;
//重新编写输入捕获回调函数
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{

  if(htim->Instance == HCSR04_TIM)
  {
    //如果上升沿检测
    if(hcsr04.STATE == RISING)
    {
      //设置定时器CNT为0
      __HAL_TIM_SET_COUNTER(&HCSR04_TIM_HAL,0);
      //读取上升沿时的CNT值到buf
      hcsr04.buf[0] = __HAL_TIM_GetCounter(&HCSR04_TIM_HAL);
      //设置下一个捕获为下降沿
      __HAL_TIM_SET_CAPTUREPOLARITY(&HCSR04_TIM_HAL,HCSR04_TIM_CHANNEL,TIM_ICPOLARITY_FALLING);
      //改变运行模式
      hcsr04.STATE=FALLING;
    }else if(hcsr04.STATE == FALLING)//如果下降沿检测
    {
      //获取当前的CNT到buf2，这样高电平维持的时间长度就记录了
      hcsr04.buf[1] = __HAL_TIM_GetCounter(&HCSR04_TIM_HAL);
      //将运行标志设置为完成
      hcsr04.STATE=OVER;

    }
  }

}
```

4.5 显示模块（OLED）

显示模块采用的是 0.96 寸的由 IIC 驱动的 OLED 屏，如图 4.9 所示。

图 4.7 OLED 显示屏

该模块只有 4 个引脚，各引脚的功能及连接如下表所示。

VCC—3V3	GND—GND
SCL—I2C1_SCL（PB8）	SDA—I2C1_SDA（PB9）

对于 OLED 模块的使用，首先需要其底层驱动函数库，该库从其显示原理上编写，可以显示数字、字符、字符串、汉字、图片等，其次这些显示的内容需要通过取模软件对其取字模才能在 OLED 上显示。综上，使用 OLED 模块只需要准备其底层函数驱动库和取模软件即可。

```
void OLED_ShowStr(unsigned char x, unsigned char y, unsigned char ch[], unsigned char TextSize)
{
  unsigned char c = 0,i = 0,j = 0;
  switch(TextSize)
  {
    case 1:
    {
      while(ch[j] != '\0')
      {
        c = ch[j] - 32;
        if(x > 126)
        {
          x = 0;
          y++;
        }
        OLED_SetPos(x,y);
        for(i=0;i<6;i++)
          WriteDat(F6x8[c][i]);
        x += 6;
        j++;
      }
    }break;
    case 2:
    {
      while(ch[j] != '\0')
      {
        c = ch[j] - 32;
        if(x > 120)
        {
          x = 0;
          y++;
        }
        OLED_SetPos(x,y);
        for(i=0;i<8;i++)
          WriteDat(F8X16[c*16+i]);
        OLED_SetPos(x,y+1);
        for(i=0;i<8;i++)
          WriteDat(F8X16[c*16+i+8]);
        x += 8;
        j++;
      }
    }break;
  }
}
```

4.6 扫码模块（SH-50）

在场地组的要求中，小车首先要在启动区进行扫码，得到需要夹取的物料的信息，然后小车才能出发。

扫码模块使用的是上海斯恳赫公司生产的 SH-50 型扫码模块，如图 4.11 所示。该模块可以直接通过串口将扫码信息发送到上位机，中间无需人为转换。但在项目实施中，小车要自我识别扫码信息，不能人为操控，因此该模块的使用主要是将扫码得到的信息进行处理，使单片机能识别。

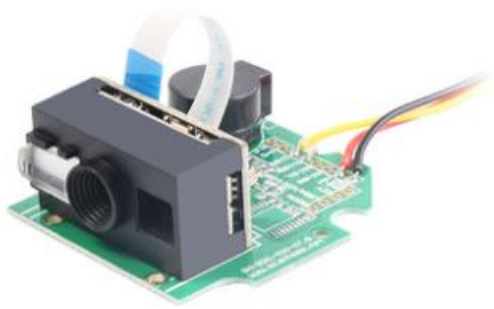

图 4.8 扫码模块（SH-50）

在该扫码模块上自带串口功能，因此单片机需要通过串口与该模块通讯。如图 4.12 所示，首先在单片机中预设好扫码要得到的信息，以“01234567”为例，将信息进行拆分提取。

```
/* USER CODE BEGIN PV */
        uint8_t x1[8]=“01234567”;        //定义数组存放收到的二维码
        uint8_t x2[8];
        uint8_t t0;                      //定义t0为提取出来的数组中的第1位
        uint8_t t1;                      //定义t1为提取出来的数组中的第2位
        uint8_t t2;                      //定义t2为提取出来的数组中的第3位
        uint8_t t3;                      //定义t3为提取出来的数组中的第4位
        uint8_t t4;                      //定义t4为提取出来的数组中的第5位
        uint8_t t5;                      //定义t5为提取出来的数组中的第6位
        uint8_t t6;                      //定义t6为提取出来的数组中的第7位
        uint8_t t7;                      //定义t7为提取出来的数组中的第8位
        uint8_t str[40];
        uint8_t buf[8];                  //储存收到的字符
/* USER CODE END PV */
```

4.7 红外寻迹模块（四路红外循迹模块）

红外循迹模块负责检测地面的黑色迹线，控制履带底盘的前进、左转、右转等动作，如图 4.14 所示。

图 4.9 红外循迹模块（四路红外循迹）

对于该循迹模块，黑色 led 发出红外光，经反射后由白色 led 接收。当物体的颜色较白时，反射的红外光较多，此时模块上对应的 D 口 led 亮，此时输出低电平；当物体的颜色较深时，吸收的红外光较多，反射的较少，此时模块上对应的 D 口 led 灭，此时输出高电平。模块上有对应的电位器，调节电位器即可调节相应循迹模块的灵敏度。因此，对于该模块的编程，主要是读取相应引脚的高低电平去改变履带底盘电机的转速和转向即可。

四路红外循迹模块控制履带底盘直流电机的部分程序如图 4.15 所示，当中间两路检测到高电平时，说明此时为直线，车体居中，底盘直走；当中间两路任意一路为低电平时，说明此时车体向该路偏转，需要将该路侧的电机转速加大，使其归正，以此类推。

```
#define INF_port    GPIOB
#define INF_PIN_1 GPIO_PIN_12
#define INF_PIN_2 GPIO_PIN_13
#define INF_PIN_3 GPIO_PIN_14
#define INF_PIN_4 GPIO_PIN_15


#define INF1 HAL_GPIO_ReadPin(INF_port,INF_PIN_1)//右1
#define INF2 HAL_GPIO_ReadPin(INF_port,INF_PIN_2)//右2
#define INF3 HAL_GPIO_ReadPin(INF_port,INF_PIN_3)//左2
#define INF4 HAL_GPIO_ReadPin(INF_port,INF_PIN_4)//左1
```

//小车循迹

```
void line_follow(void)
{
if(INF1==0 && INF2==1 && INF3==1 && INF4==0)
     {
        FORWARD(450,10);//直行
     }

else if ( (INF1==0 && INF2==0 && INF3==1 && INF4==0) )
     {
      RIGHTTURN(590,350,10);//右小转
     }
else if( (INF1==0 && INF2==0 && INF3==1 && INF4==1) )
     {
      FORWARD(500,300);
      while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
          right_90(460,460,5);
     }
else if( (INF1==0 && INF2==1 && INF3==0 && INF4==0)  )
   {
     LEFTTURN(350,590,10);//左小转
    }
else if( (INF1==1 && INF2==1 && INF3==0 && INF4==0) )
  {
  //LEFTTURN(350,670,10);
    //FORWARD(500,300);
    while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 ){
      left_90(455,455,5);
    }
  }
else if( INF1==0 && INF2==0 && INF3==0 && INF4==0 ){
     BACK(320,250);
   }
```

//小车左转 90°

```
void left_90turn(void)
{

    // while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
   while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
  {
        left_90(450,450,2);

    }
}

void right_90turn(void)
{


     while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
     {
           right_90(450,450,2);
     }
}
```

//小车环岛循迹

```
void circle_turn(void)
{
  if(INF1==0 && INF2==1 && INF3==1 && INF4==0)
     {
        FORWARD(420,2);//直行
     }

else if ( (INF1==0 && INF2==0 && INF3==1 && INF4==0)
        ||(INF1==0 && INF2==0 && INF3==1 && INF4==1) )
     {
       RIGHTTURN(675,350,5);//右小转

     }
 else if( (INF1==0 && INF2==1 && INF3==0 && INF4==0)
         ||(INF1==1 && INF2==1 && INF3==0 && INF4==0) )
   {
      LEFTTURN(350,670,5);//左小转
    }
   else if( INF1==0 && INF2==0 && INF3==0 && INF4==0 ){
      BACK(330,40);
   }
}
```

4.8 Wifi 蓝牙模块（ESP32）

Wifi 蓝牙模块 ESP32 用以辅助控制机械臂的运动。

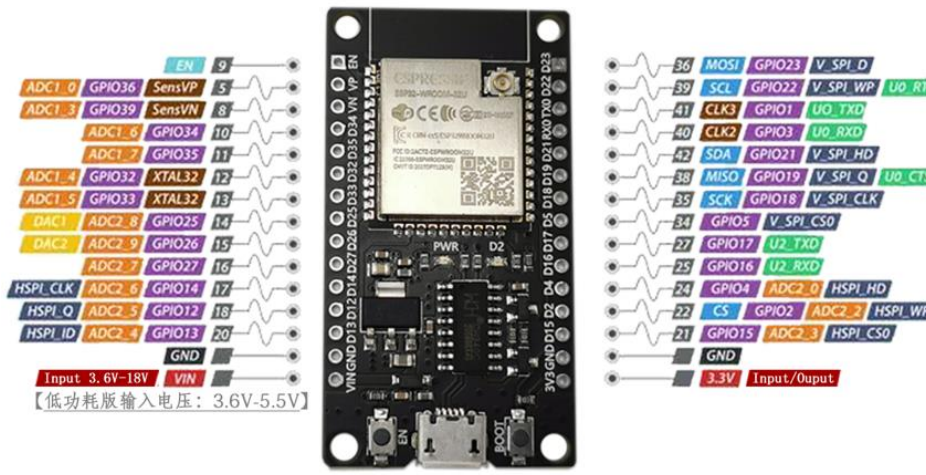

图 4. 10 ESP32 引脚分配功能图

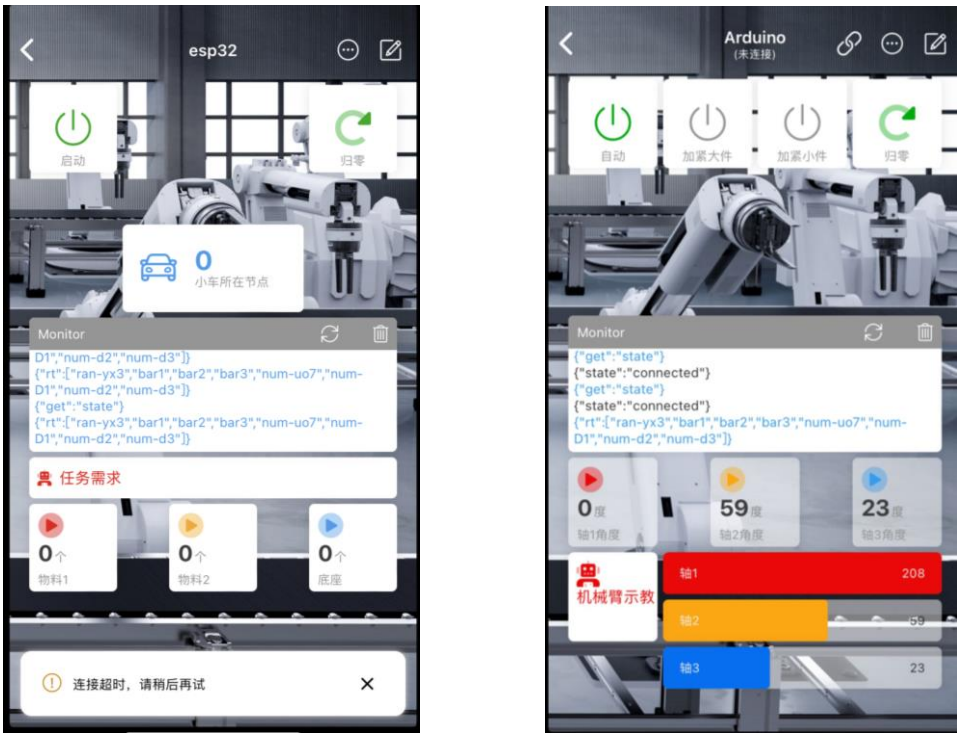

图 4.11 物联网半实时监控组态（左）和蓝牙实时调试组态（右）

程序：

```
// 按下按键即会执行该函数
void button1_callback(const String & state)
{
    if (state=="on") {
        //digitalWrite(LED_BUILTIN, HIGH);
        // 反馈开关状态
        button_on_flag=1;
        Button1.print("on");

    } else if(state=="off"){
        //digitalWrite(LED_BUILTIN, LOW);
        // 反馈开关状态
        //Blinker.vibrate();
        button_on_flag=0;
        Button1.print("off");
    }

}
void buttonre0_callback(const String & state)
{
  button_re0_flag=1;
}
```

第 5 章　程序调试结果及改进

程序调试过程中，对于有些模块，只要给定固定的程序就能够正确运行。因此只对其中直流电机模块、超声波模块、红外寻迹模块和显示的调试进行阐述。

5.1 直流电机模块的调试（陈嘉琛）

在直流电机模块的调试过程中，最开始采用的是 TB6612FNG，多次调试却总是只有一个履带可以运行，所以想到是否是这种模块的电压太小，所以就更换成了 APO-L1，发现的确是 TB6612FNG 带不动直流电机。把左右两个直流电机的占空比设置为相同值，理论上两个直流电机的转速应该相同，但经过数十次的循迹过后，发现在相同占空比的条件下，履带小车在走直线时，车体总会往左偏，即左轮速度会比右轮速度快。随后经过数次尝试，找出了合适的占空比，如图 5.1 所示，使履带小车在进行直线循迹时，能尽量保持在中位。

5.2 超声波模块的调试（王玥灏）

在超声波模块的调试中，很重要的一个因素是电源的供给必须是 5V，如果使用的是 3V3 的电源，即使程序编写正确，超声波测出的距离也会不准确或者没有变化，对于排查程序的错误也会增加很多困难。

最开始是在 NUCLEO 开发板上进行程序的编写实验的，成功驱动之后需要移植到 STM32F103C8T6 最小系统板上，但在移植之后，发现超声波模块不能正常使用，连值都不能输出。在经过两部分程序及 CubeMX 设置的对比之后发现，NUCLEO 开发板上自带旁路时钟源，而 STM32F103C8T6 最小系统板上没有，因此照搬 NUCLEO 的设置的行不通，最后把 HSE 选项选到 Disable 之后，如图 5.3 所示，超声波模块就能正常运行了。

5.3 红外寻迹模块的调试（刘欣源）

红外循迹模块的调试主要是灵敏度和模块之间的间距。红外循迹模块的灵敏度是单片机能正常触发相关程序的关键，其灵敏度是通过模块上相关 led 小灯的亮灭来体现的，有一个最大的误区就是当小灯微微亮时，输出的是高电平，很容易导致信号错误，使小车发生错误判断。所以在调试时，相关 led 小灯只能出现非亮即灭两种情况，否则容易导致单片机错误判断。另外是在直射光较强时，地面的黑色迹线反射的光线也较强，会使红外循迹模块接收到相关的红外信号，即把此处判断为低电平，会使小车莫名停车，但遮挡相关干扰光线之后，就能使小车正常工作。环境过暗也同理，会使红外循迹模块把本该识别为低电平的地方识别成高电平，使小车误判，因此环境的光线也极为重要。

红外循迹模块的间距的主要是配合场地的 50mm 黑色迹线，在直线情况时，本组使用中间两个模块位于黑色迹线中间，两边的红外模块在黑色迹线之外，四个红外循迹模块在水平和竖直方向的同一平面上。若红外循迹模块之间的间距过宽，会使履带小车在过圆形环岛迹线时，不容易判断路口，进行计数识别；若红外循迹模块之间的间距过窄，会使履带小车频繁的调整车体姿态，影响履带小车的行走，并且会增大红外循迹模块之间的干扰，造成误检测。

5.4 显示模块的调试（徐世杰）

显示模块即 OLED 模块程序的调试主要是换行显示及动态刷新的程序调试。

图 5.1　OLED 模块

换行显示的目的是使 OLED 屏能显示更多相关信息。如图 5.4 所示，在绿色箭头方向，OLED 有 128 个像素点；在蓝色箭头方向，OLED 分为 8 页，每页共 8 个像素点。在取模软件输出的字的大小一般为 16×16 个像素的大小，因此可以知道，每个字或数字等，在绿色箭头方向占 16 个像素点，在蓝色箭头方向占 2 页，所以可以写出换行程序如图 5.5 所示。

```
OLED_ShowCHinese (0,4,7);                    // “当”
OLED_ShowCHinese (16,4,8);                   // “前”
OLED_ShowCHinese (32,4,14);                  // “距”
OLED_ShowCHinese (48,4,15);                  // “离”
OLED_ShowCHinese (62,4,6);                   // “：”
```

图 5.5 OLED 换行程序（部分）

其次是 OLED 显示动态刷新内容，此功能可以配合超声波模块的测距和检测四路红外循迹模块的使用。调试的思路是：首先定义一个数组，将外部输入进来的信号存到此数组之中，再采用 sprintf 函数将其打印，最后通过 OLED 显示函数将其发送到显示页面上。

```
/* USER CODE BEGIN 0 */
uint16_t str_buff[64];                       //定义数组，用来存储数据，发送到OLED上
/* USER CODE END 0 */

While(1)
{
sprintf ((char *)str_buff ,"%d.%d",distance,distance%10);      //将 distance 处理
OLED_ShowString (70,4,(uint8_t *)str_buff ,16);               //发送到OLED上
}
```

图 5.2　OLED 动态刷新显示内容

第 6 章　项目思政心得体会

陈嘉琛：

经过本次二级项目，我们将二级项目中遇到的问题和困难进行了思考总结，对所使用的 stm32cube，keil 等软件和机械臂机器人，还有红外循迹模块，超声波模块等传感器有了进一步的认识；通过本次二级项目，我们除了提高了软件的操作能力，也明白了团队合作的重要性，没有大家的共同努力，是无法将本次项目做好的。相信我们日后对待更加复杂的问题时，也能处理地更加完善。

王玥灏：

通过此次二级项目的实施，我们对机器人技术，机械动力学分析，嵌入式，传感器，机电一体化等课程有了更深的认识。“纸上得来终觉浅，绝知此事要躬行”。在没有开始实验之前，对于二级项目有点自己上了就能行的感觉。等到真正的上手操作时，却发现自己终究是眼高手低，各种各样的问题层出不穷，最终在通力合作下才完成了项目。学习的路上任重道远，我们要保持终身学习的好习惯，才能站在时代的风口而屹立不倒。

刘欣源：

通过查阅相关文献资料与小组内部讨论，在小组的共同努力下，完成了本次二级项目。本次项目涉及了嵌入式，机器人技术等课程，对这些学过的课程有了更深刻的认识。二级项目训练了我们如何查资料及绘图、数据处理、创新能力，培养了我们独立设计机电一体化产品的能力，提高我们综合应用已有知识解决问题的能和综合素质。同时加强团队协作能力，促进交流与合作，拓展视野，勇于创新，提高思考与决策水平，形成解决实际问题的能力和终身学习的能力。

徐世杰：

在本次二级项目中，合理分工，通力合作，每位成员都发挥了自己相对应的价值，对本次课程设计做出了很大的贡献，我们了收获很多，在设计过程中大家一起讨论，分工合作，相互帮助，使得项目顺利完成。在这个项目过程中同样也会有很多的意见，通过团结合作，我们最终完善了这个项目。同时，这次项目使我们把过去一个段时间所学的理论知识融会贯通，真正了解理论知识在工程实际中的应用，提高了解决实际问题的能力，锻炼了主动学习及查阅资料的能力。

参考文献

[1] 闻邦椿 机械设计手册.[M](第一版)，北京，机械工业出版社，2010 年 1 月，

[2] 钱晓捷 [程 楠] 嵌入式系统导论[M]（第四版），北京，电子工业出版社，2020 年 2 月

[3] 许立忠 [周玉林] 机械设计[M] （第三版），北京，中国标准出版社，2019 年 9 月

[4] 龚溎义 [严国良] 设计课程设计图册（第三版），北京，高等教育出版社，2019 年 12 月

附录 1　程序源代码

```
//linetracking.h 宏定义部分程序，可以使主程序更简洁
#define INF1 HAL_GPIO_ReadPin(INF_port,INF_PIN_1)//右 1
#define INF2 HAL_GPIO_ReadPin(INF_port,INF_PIN_2)//右 2
#define INF3 HAL_GPIO_ReadPin(INF_port,INF_PIN_3)//左 2
#define INF4 HAL_GPIO_ReadPin(INF_port,INF_PIN_4)//左 1

//linetracking.c 程序
#include "linetracking.h"
#include "motor.h"
#include "stm32f1xx_hal.h"

extern uint8_t flag_reach;
void linetrack(void)
{
  if(INF1==0 && INF2==1 && INF3==1 && INF4==0)
                {
                  FORWARD(450,10);//直行
              }
else if ( (INF1==0 && INF2==0 && INF3==1 && INF4==0)
                  )
              {
                            RIGHTTURN(500,350,10);//右小转
                        }
else if ( (INF1==0 && INF2==0 && INF3==0 && INF4==1)
                            ||(INF1==0 && INF2==0 && INF3==1 && INF4==1))
              {
                            RIGHTTURN(650,370,10);//右 da 转
                        }
else if( (INF1==0 && INF2==1 && INF3==0 && INF4==0) )
    {
                            LEFTTURN(350,500,10);//左小转
                        }
else if( (INF1==1 && INF2==0 && INF3==0 && INF4==0)
                        ||(INF1==1 && INF2==1 && INF3==0 && INF4==0) )
    {
                            LEFTTURN(350,680,10);//左 da 转
                        }
```

```
else if(INF1==0 && INF2==0 && INF3==0 && INF4==0)
    {
                STOP(0,100);
                BACK(350,400);
       }
else if(INF1==1 && INF2==1 && INF3==1 && INF4==1)
    {
            STOP(0,50);//停转
                if(INF1==1 && INF2==1 && INF3==1 && INF4==1){
                            STOP(0,50);//停转
                            flag_reach=1;
                           HAL_GPIO_WritePin(led_GPIO_Port,led_Pin,GPIO_PIN_SET);
                           rotate_180(420,420,3000);
                      while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
                           rotate_180(420,420,10);
                      }
           }
else if(INF1==1 && INF2==1 && INF3==1 && INF4==0)
    {
            STOP(0,50);//停转
            if(INF1==1 && INF2==1 && INF3==1 && INF4==0)
    {
            while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
                            left_90(330,580,10);
           }
           }
void line_follow(void)
{
if(INF1==0 && INF2==1 && INF3==1 && INF4==0)
            {
                    FORWARD(450,10);//直行
            }

else if ( (INF1==0 && INF2==0 && INF3==1 && INF4==0) )
            {
                RIGHTTURN(590,350,10);//右小转
            }
else if( (INF1==0 && INF2==0 && INF3==1 && INF4==1) )
            {
```

```
                FORWARD(500,300);
                while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
                        right_90(460,460,5);
            }
else if( (INF1==0 && INF2==1 && INF3==0 && INF4==0)    )
    {
                LEFTTURN(350,590,10);//左小转
            }
else if( (INF1==1 && INF2==1 && INF3==0 && INF4==0) )
      {
      //LEFTTURN(350,670,10);
            //FORWARD(500,300);
            while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 ){
                left_90(455,455,5);
            }
      }
else if( INF1==0 && INF2==0 && INF3==0 && INF4==0 ){
                BACK(320,250);
       }
}
void left_90turn(void)
{
            // while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
       while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
      {
                        left_90(450,450,2);
            }
}
void right_90turn(void)
{
             while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
             {
                        right_90(450,450,2);
             }
}
void circle_turn(void)
{
      if(INF1==0 && INF2==1 && INF3==1 && INF4==0)
             {
```

```
                    FORWARD(420,2);//直行
            }
else if ( (INF1==0 && INF2==0 && INF3==1 && INF4==0)
          ||(INF1==0 && INF2==0 && INF3==1 && INF4==1) )
            {
                RIGHTTURN(675,350,5);//右小转
            }
 else if( (INF1==0 && INF2==1 && INF3==0 && INF4==0)
                        ||(INF1==1 && INF2==1 && INF3==0 && INF4==0) )
    {
                LEFTTURN(350,670,5);//左小转
          }
       else if( INF1==0 && INF2==0 && INF3==0 && INF4==0 ){
                BACK(330,40);
       }
}
uint8_t T_check(void)
{
     if(INF1==1 && INF2==1 && INF3==1 && INF4==1)
    {
            STOP(0,50);//停转
            if(INF1==1 && INF2==1 && INF3==1 && INF4==1)
    {
          return 1;
          }
       else
            return 0;
     }
       else
            return 0;
}
uint8_t left90_check(void)
{
     if(INF1==1 && INF2==1 && INF3==1 && INF4==0)
    {
            STOP(0,50);//停转
            if(INF1==1 && INF2==1 && INF3==1 && INF4==0)
                {
                return 1;
```

```
                }
            else
              return 0;
       }
        else
              return 0;
}
uint8_t right90_check(void)
{
      if(INF1==0 && INF2==1 && INF3==1 && INF4==1)
     {
              STOP(0,50);//停转
              if(INF1==0 && INF2==1 && INF3==1 && INF4==1)
                  {
                  return 1;
                  }
            else
              return 0;
      }
        else
              return 0;
}
uint8_t right_180turn(void)
{
      right_90(440,440,800);
      while( (INF1==0 && INF2==1 && INF3==1 && INF4==0)==0 )
      {
            right_90(440,440,2);
      }
      return 1;
}
void line_follow_slowdown(void)
{
      if(INF1==0 && INF2==1 && INF3==1 && INF4==0)
              {
                        FORWARD(380,5);//直行
              }
else if ( (INF1==0 && INF2==0 && INF3==1 && INF4==0) )
              {
```

```
                RIGHTTURN(490,340,10);//右小转
            }
else if( (INF1==0 && INF2==0 && INF3==1 && INF4==1) )
            {
                FORWARD(500,300);
                while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 )
                        right_90(460,460,5);
            }
else if( (INF1==0 && INF2==1 && INF3==0 && INF4==0)    )
    {
                LEFTTURN(340,490,10);//左小转
            }
else if( (INF1==1 && INF2==1 && INF3==0 && INF4==0) )
      {
      //LEFTTURN(350,670,10);
            //FORWARD(500,300);
            while( (INF1==0 && INF2==1 && INF3==1 && INF4==0) ==0 ){
                left_90(455,455,5);
            }
      }
else if( INF1==0 && INF2==0 && INF3==0 && INF4==0 ){
                BACK(320,250);
        }
}
uint8_t middle_check(void)
{
      if(INF1==0 && INF2==1 && INF3==1 && INF4==0)
    {

            HAL_Delay(20);//
            if(INF1==0 && INF2==1 && INF3==1 && INF4==0)
                {
                return 1;
                }
           else
            return 0;
      }
        else
            return 0;
```

```
}
void dongyidong(void)
{
        FORWARD(400,2000);
        BACK(400,2000);
        left_90(450,450,1000);
        right_90(450,450,1000);
}

//motor.c 直流电机运行程序
#include "motor.h"
#include "tim.h"
//左转函数
/*  注：电机输入 IO 口 1 表示正转，0 表示反转
                PWM 口输入 0--1000 值进行速度调节
*/
void LEFTTURN(int C_COUNTER_L,int C_COUNTER_R,int TIME_L)
{       //左电机
      //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);
//开启左电机的 pwm 调压
      __HAL_TIM_SET_COMPARE(&htim4,  TIM_CHANNEL_2,  C_COUNTER_L);
//设定左电机的占空比=C_COUNTER/1000
      HAL_GPIO_WritePin(LEFT_MOTOR_GPIO , LEFT_MOTOR_PIN,
GPIO_PIN_RESET); //左轮正转        /* 设定 IO 的值使左转电机正转或反转  */
          //右电机
    //    HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);
//开启右电机的 pwm 调压
      __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER_R*1.2);
//设定右电机的占空比=C_COUNTER/1000
      HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO,  RIGHT_MOTOR_PIN,
GPIO_PIN_RESET);          /* 设定 IO 的值使右转电机正转或反转  */
    //设定运行时间
        HAL_Delay(TIME_L);
//      HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);    //关闭左电机的 pwm 输
出，防止怠速运行
//      HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);    //关闭右电机的 pwm 输
出，防止怠速运行
}
```

```
//右转函数
void RIGHTTURN(int C_COUNTER_L,int C_COUNTER_R,int TIME_R)
{       //左电机
     //                          HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);
//开启左电机的 pwm 调压
            __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER_L);
//设定左电机的占空比=C_COUNTER/1000
       HAL_GPIO_WritePin(LEFT_MOTOR_GPIO          ,          LEFT_MOTOR_PIN,
GPIO_PIN_RESET);          /* 设定 IO 的值使左转电机正转或反转   */

          //右电机
        //                          HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);
//开启右电机的 pwm 调压
            __HAL_TIM_SET_COMPARE(&htim3,                    TIM_CHANNEL_3,
C_COUNTER_R*1.12);                                  //设定右电机的占空比
=C_COUNTER/1000
       HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO,                RIGHT_MOTOR_PIN,
GPIO_PIN_RESET);          /* 设定 IO 的值使右转电机正转或反转   */
     //设定运行时间
          HAL_Delay(TIME_R);
//        HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);      //关闭左电机的 pwm 输
出，防止怠速运行
//        HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);      //关闭右电机的 pwm 输
出，防止怠速运行
}

//前进函数
void FORWARD(int C_COUNTER,int TIME)
{       //左电机
       //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);
//开启左电机的 pwm 调压
            __HAL_TIM_SET_COMPARE(&htim4,   TIM_CHANNEL_2,   C_COUNTER);
//设定左电机的占空比=C_COUNTER/1000
       HAL_GPIO_WritePin(LEFT_MOTOR_GPIO          ,          LEFT_MOTOR_PIN,
GPIO_PIN_RESET);          /* 设定 IO 的值使左转电机正转或反转   */
          //右电机
       //                          HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);
//开启右电机的 pwm 调压
            __HAL_TIM_SET_COMPARE(&htim3,                    TIM_CHANNEL_3,
```

```
C_COUNTER*1.12);                    //设定右电机的占空比=C_COUNTER/1000
    HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO,          RIGHT_MOTOR_PIN,
GPIO_PIN_RESET);          /* 设定 IO 的值使右转电机正转或反转 */
   //设定运行时间
      HAL_Delay(TIME);
     // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);
     // HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);    //关闭左右电机的 pwm
输出，防止怠速运行
}

//后退函数
void BACK(int C_COUNTER,int TIME)
{     //左电机
    //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);
//开启左电机的 pwm 调压
        __HAL_TIM_SET_COMPARE(&htim4,  TIM_CHANNEL_2,  C_COUNTER);
//设定左电机的占空比=C_COUNTER/1000
    HAL_GPIO_WritePin(LEFT_MOTOR_GPIO      ,      LEFT_MOTOR_PIN,
GPIO_PIN_SET);          /* 设定 IO 的值使左转电机正转或反转 */

       //右电机
    //                  HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);
//开启右电机的 pwm 调压
        __HAL_TIM_SET_COMPARE(&htim3,                TIM_CHANNEL_3,
C_COUNTER*1.12);                    //设定右电机的占空比=C_COUNTER/1000
    HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO,          RIGHT_MOTOR_PIN,
GPIO_PIN_SET);          /* 设定 IO 的值使右转电机正转或反转 */
   //设定运行时间
      HAL_Delay(TIME);
     // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);
     // HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);    //关闭左右电机的 pwm
输出，防止怠速运行
}

//制动函数
void STOP(int C_COUNTER,int TIME)
{     //左电机
   //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);     //开启左电机的 pwm 调压
   //HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);     //开启右电机的 pwm 调压
```

```
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER);    //设定左电机的占空比=C_COUNTER/1000
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER);    //设定右电机的占空比=C_COUNTER/1000
     HAL_Delay(TIME);
    // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);      //关闭左电机的 pwm 输出，防止怠速运行
     //HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);      //关闭右电机的 pwm 输出，防止怠速运行
}
void left_90(int C_COUNTER_L,int C_COUNTER_R,int TIME)
{       //左电机
    //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);      //开启左电机的pwm调压
    //HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);      //开启右电机的pwm调压
  HAL_GPIO_WritePin(LEFT_MOTOR_GPIO , LEFT_MOTOR_PIN, GPIO_PIN_SET); //左轮反转          /* 设定 IO 的值使左转电机正转  */
    HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO,                RIGHT_MOTOR_PIN, GPIO_PIN_RESET);          /* 设定 IO 的值使右转电机正转  */
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER_L);    //设定左电机的占空比=C_COUNTER/1000
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER_R*1.12); //设定右电机的占空比=C_COUNTER/1000
    HAL_Delay(TIME);
    // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);      //关闭左电机的 pwm 输出，防止怠速运行
     //HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);      //关闭右电机的 pwm 输出，防止怠速运行
}
void right_90(int C_COUNTER_L,int C_COUNTER_R,int TIME)
{       //左电机
    //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);      //开启左电机的pwm调压
    //HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);      //开启右电机的pwm调压
  HAL_GPIO_WritePin(LEFT_MOTOR_GPIO , LEFT_MOTOR_PIN, GPIO_PIN_RESET); //左轮反转          /* 设定 IO 的值使左转电机正转  */
    HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO,                RIGHT_MOTOR_PIN, GPIO_PIN_SET);          /* 设定 IO 的值使右转电机正转  */
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER_L);    //设定左电机的占空比=C_COUNTER/1000
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER_R);    //
```

```
设定右电机的占空比=C_COUNTER/1000
    HAL_Delay(TIME);
    // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);      //关闭左电机的 pwm 输出，防止怠速运行
     //HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);      //关闭右电机的 pwm 输出，防止怠速运行
}
void rotate_180(int C_COUNTER_L,int C_COUNTER_R,int TIME)
{       //左电机
    //HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_2);      //开启左电机的 pwm 调压

    //HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);      //开启右电机的 pwm 调压
  HAL_GPIO_WritePin(LEFT_MOTOR_GPIO , LEFT_MOTOR_PIN, GPIO_PIN_RESET); //左轮反转          /* 设定 IO 的值使左转电机正转  */
    HAL_GPIO_WritePin(RIGHT_MOTOR_GPIO,                RIGHT_MOTOR_PIN, GPIO_PIN_SET);          /* 设定 IO 的值使右转电机正转  */
    __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, C_COUNTER_L);    //设定左电机的占空比=C_COUNTER/1000
    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, C_COUNTER_R);    //设定右电机的占空比=C_COUNTER/1000

    HAL_Delay(TIME);
    // HAL_TIM_PWM_Stop(&htim4,TIM_CHANNEL_2);      //关闭左电机的 pwm 输出，防止怠速运行
     //HAL_TIM_PWM_Stop(&htim3,TIM_CHANNEL_3);      //关闭右电机的 pwm 输出，防止怠速运行
}
```

```
while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

        sprintf ((char *)str1 ,"point:%d",vechile1.Trend);
```

```
{
    case 1:
        line_follow();
        if(right90_check())
            {
                FORWARD(500,550);
                right_90(450,450,500);
                right_90turn();
                vechile1.Trend=2;
                close_singal;
            }
            break;
    case 2:
        line_follow();
        if( T_check() )
        {
            FORWARD(500,500);
            right_90(450,450,500);
            right_90turn();
            vechile1.Trend=3;
        }
        break;
    case 3:
            circle_turn();
        if( (right90_check() ) && (vechile1.turn_situ==0) )
        {
            LEFTTURN(350,675,200);//左小转
            vechile1.turn_situ = 1;
        }

        else if( (right90_check() || T_check() ) && (vechile1.turn_situ==1) )
        {
            RIGHTTURN(520,450,650);
            right_90(450,450,650);
            right_90turn();
            vechile1.turn_situ = 0;
            vechile1.Trend=4;
            //line_follow();
        }
```

```
        break;
    case 4:     //进到第一个路口抓物块
        line_follow();
        if( left90_check() )
            {
            FORWARD(500,530);
            left_90(450,450,300);
            left_90turn();
            vechile1.Trend=16;
            }
            break;
    case 16:
        line_follow_slowdown();
        if( T_check() )
        {
            STOP(0,500);
            send_singal;
            STOP(0,1500);
            close_singal;
            vechile1.Trend=5;
        }
        break;
    case 5:
        BACK(350,2600);
        left_90(450,450,650);
        left_90turn();
        vechile1.Trend=7;
        break;
    case 7:
        line_follow();
        if( T_check() )
        {
            FORWARD(500,540);
            right_90(450,450,2);
            right_90turn();
            vechile1.Trend=8;
        }
        break;
    case 8:
```

```
            circle_turn();
        if( (right90_check() ) && (vechile1.turn_situ==0) )
        {
            LEFTTURN(350,670,150);//左小转
            vechile1.turn_situ = 1;
        }
        if( (right90_check() || T_check() ) && (vechile1.turn_situ==1) )
        {
            RIGHTTURN(520,450,650);
            right_90(450,450,650);
            right_90turn();
            vechile1.turn_situ = 0;
            vechile1.Trend=9;
        }
        break;

case 9:
        line_follow();
if( left90_check() )
{
        FORWARD(500,520);
        left_90(450,450,500);
        left_90turn();
        vechile1.Trend=10;
}
        break;
case 10:
        line_follow();
if( T_check() )
        {
            STOP(0,500);
            send_singal;
            STOP(0,1500);
            close_singal;
            vechile1.Trend=11;
        }
        break;
case 11:
        BACK(350,500);
```

```
                if( right_180turn() )
                {
                }
                break;

            case 13:
                line_follow();
                if( T_check() )
                {
                    FORWARD(500,500);
                    left_90(450,450,500);
                    left_90turn();
                    vechile1.Trend=14;
                }
            break;
            case 14:
                line_follow();
                if(left90_check())
                    {
                        FORWARD(500,530);
                        left_90(450,450,500);
                        left_90turn();
                        vechile1.Trend=15;
                    }
            case 15:
                line_follow();
            if( T_check() )
                {
                    STOP(0,500);
                }
                break;
            }
    }
    /* USER CODE END 3 */
}
```