**电路与电子III**

**课程报告**

**题 目：Tankbot智能搬运履带机器人**

学 院：求是学部

专业班级：未来智能机器平台1班、2班

姓 名： 石云天、袁萌启、李嘉树、马天翼、李贺航、季若鱼

学 号： 3020205015、3020234149、3020233037、3020001016、3020003013、3020214020

**未来技术学院**

**二零二二 年 08 月**

摘要

随着物流产业的兴起以及科技的发展，智能搬运机器已经成为人们研究的热点话题。如何让智能机器在完成避障、定位、取件、返回等多个任务的同时，实现全自动控制，并兼顾轻便性和环境友好性，是该项目研究的一大挑战。本组项目以Tankbot智能搬运履带机器人为原型，模拟工程中真实应用的智能搬运机器人。Tankbot智能搬运履带机器人是一款以STM32单片机为主控制板，集合红外传感器、超声波传感器、声音传感器、机械臂等多个模块的机器人。经过一个完整的项目设计周期，包括对于项目背景的查找，制定项目规划，硬件设计与实现，软件设计与编程；本组完成了对于智能履带机器人的设计工作，并通过集成与功能测试等环节，探讨如何将其应用于市场。本组着重于软件设计与实现阶段，以C语言为基础，Keil5为编程软件，利用轮巡控制与并行控制的基本思想编写代码，并进行符合实际测试环境的调试工作。最终，实现了预期的全部功能，产品具有极佳的测试效果与应用前景。

**关键词：**智能搬运、STM32单片机**、**软件设计**、**轮巡控制

目 录

[摘要 2](#_Toc138119623)

[第一章 研究背景 4](#_Toc138119624)

[第二章 项目规划 6](#_Toc138119625)

[2.1 研究目标 6](#_Toc138119626)

[2.2 实施计划 6](#_Toc138119627)

[2.2.1 项目作品需求表达 6](#_Toc138119628)

[2.2.2 项目作品计划实施 7](#_Toc138119629)

[2.3 成员及分工 8](#_Toc138119630)

[2.4 进度安排 8](#_Toc138119631)

[第三章 硬件设计与实现 10](#_Toc138119632)

[3.1 硬件设计与器件选型 10](#_Toc138119633)

[3.2 原理图设计 12](#_Toc138119634)

[3.3 仿真及调试结果 13](#_Toc138119635)

[第四章 软件设计与编程 14](#_Toc138119636)

[4.1 软件设计 14](#_Toc138119637)

[4.2程序/算法设计 14](#_Toc138119638)

[4.3 仿真及测试结果 23](#_Toc138119639)

[参考文献 25](#_Toc138119640)

1. 所选课题国内外的研究动态、水平、存在问题，并附主要参考文献。

2. 项目目标、具体实施计划、成员及分工、进度安排

3. 硬件设计方案、器件选型、原理图设计、仿真及调试结果分析

4. 软件设计方案、程序及算法设计过程、仿真及测试结果分析

5. 外观及结构设计、模块整合、功能测试

# 第一章 研究背景

由于当今物流搬运领域的兴起，人们越来越需要设计出一类智能搬运机器人代替人类的大量劳动。另外，在航天探测领域中，智能搬运类型机器人可以将采集到的物品带回研究，因此对于全自动的搬运机器人开发，也成为一个热点课题。国内外目前基于自动控制理论、单片机原理以及机械设计等基础理论开发出的智能机器人，已经能够达到半自动地完成循迹、避障、定位抓取、物料识别并取回等功能。但是目前对于机器人的全自动化控制实现得不够，部分功能仍然需要远程调节和操控来完成。

目前国内最常用的智能机器人控制主板是STM32。该单片机具有较强的控制运算能力，兼具体积小、能耗低的优点，具有USART、I2C、定时器等丰富的 外设便于开发和应用。STM32能够接收循迹模块信号并通过电机驱动模块控制底盘行走机构实现机器人的自主循迹，也能获取二维码识别模块识别的二维码信息（物料搬运顺序）和颜色传感器读取的物料信息进而通过舵机控制机械手臂对物料进行搬运。

机械结构设计方面，机器人整体机械结构包括机械爪、机械臂、载物台、回转装置以及悬挂。其中，机械爪由激光切割制造而成，具有多个自由度，可以随意调节手爪的形状，即使对于不规则的形状也能够轻松实现抓取，具有广泛的实用性。机械臂设计方面，采用4个串行总线舵机控制，实现任意位置任意角度的抓取与放置。同时，使用了总线串口舵机，大大减少了对于芯片资源的占用，使得指令效率更高。载物台与回转装置则根据装置具体需要自行设计。

硬件模块设计方面，主要包括电机驱动模块、循迹模块、颜色识别模块、舵机、电源稳压模块等。电机驱动模块选用东芝半导体公司生产的双驱动TB6612FNG芯片作为直流电机的核心功率模块。通过STM32单片机给予 TB6612FNG电路PWM信号即可控制电机的正、反转以及速度和起停。循迹模块主要采用灰度传感器对于机器人进行循迹引导。将4个一体或者6个一体的灰度传感器安置在机器人底板的前部。模块的每个传感器通过检测到的亮度值将电平置于高低两个位置，将所得到的电信号反馈到主板，再通过主板的指令实现不同的处理，整体上形成一个小的闭环控制。与循迹模块类似的，颜色识别模块也能通过反馈实现闭环控制。选用高分辨率、全彩的TCS3200颜色传感器，并将其相应的控制引脚连接到STM32单片机接上。TCS3200 颜色传感器读取物料的R、G、B值后通过I/O口将此信息传输给STM32单片机。舵机的设置与电源的设置需要根据功能和相应的机械机构进行调整。

程序设计方面，利用功能强大的C语言编写本机器人的软件控制程序，采用模块化编程，以便适应任务多变的需求。编写完成后，需安装相应的程序烧录软件，将程序录入主板中。大体的程序设计流程如图1-1所示。

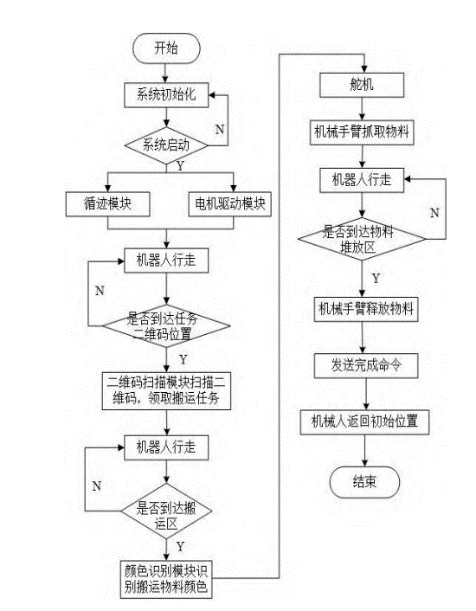


图 1-1 系统设计流程图

从上述程序流程中可以看到，仍有部分环节需要人们手动调节，例如通过操控发送完成命令才能让机器人返回初始位置结束进程。另外，如果系统出现一些扰动情况导致机器人出现偏离路线等问题，也需要人们进行手动调整。在程序编写完成后往往要根据具体的运行环境对其进行适当的修改。例如调整在行进和转弯过程中电机的转速、调整各个传感器的灵敏度、调整舵机的运行角度等。

上述实验室设计的智能搬运机器人可在市场的不同领域具有广泛应用。智能仓储AGV货架搬运机器人通过标签传感器读取地标识别卡,并通过各传感器综合应用实现避障与定位等功能，在快递搬运领域得到应用。机场行李智能搬运机器人通过静力学和运动学分析设计主要结构，再结合搬运机器人的整体设计原理，实现了行李抓取和搬运的自动化。单目视觉智能分拣搬运机器人，以智能搬运机器人的设计为基础，通过对摄像头获取的图像进行中值滤波、去除阴影、图像分割等预处理,结合形状和颜色特征对物料进行识别。提出了基于摄像头角度反馈测距法的目标定位算法,并采用增量式PID控制,使机器人动作更稳定和快速。实验表明该机器人在复杂环境下可准确高效完成物料分拣搬运。

智能搬运机器人在航空航天等科技领域也具有较多应用。例如在月球探测与火星探测的过程中，往往需要实地采样。这就需要机器人根据特定路线前进并避障，完成对样本的采集工作并返回。

目前智能物流机器人在机械结构设计、硬件设计、软件编程等方面均取得了突破性进展。然而控制手段的不足导致部分功能只能通过远程遥控的方式手动调节。因此，我们力求通过轮巡控制与并行控制相结合的方式，实现智能搬运机器人的全自动化。

# 第二章 项目规划

## 2.1 研究目标

我们计划使用Tankbot智能搬运机器人模型，实现相关功能的自动化控制。Tankbot是一款基于STM32编程的智能搬运机器人，它的履带底盘上搭载了1个6自由度机械臂，可以完成夹取、搬运等功能。Tankbot上搭载了超声波、声音、四路巡线、倾角等传感器，支持手机APP、手柄、体感手套控制，支持摄像头拓展。

根据Tankbot提供的传感器，本组实现的功能目标为：按照预订轨道行驶，到达终点后停止运行，听取指令，自动抓取物品。完成任务后，自动返回起点。产品搭建完成实物图如图2-1所示。

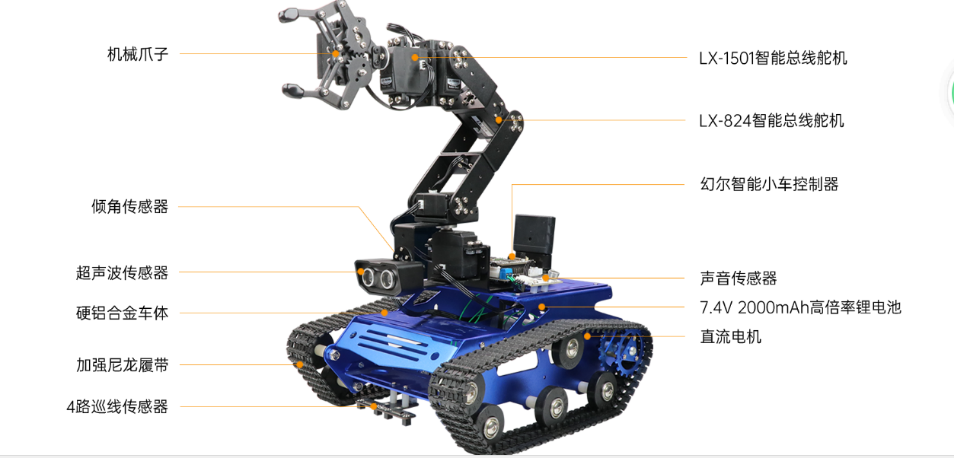


图 2-1 产品实物图

## 2.2 实施计划

### 2.2.1 项目作品需求表达

#### 2.2.1.1 需求列举

经过查找资料与实际分析，我们列举了以下几条用户对于产品的不同方面的需求可能：

a)抓取物体过程中的稳定性

b)装置的快速性与准确性

c)可运送货物的重量范围尽可能大

d)装置自身的质量不能过大

e)装置制作所耗费的能量要尽可能少

#### 2.2.1.2 组织需求的层级

货物需求：抓取物体过程中的稳定性；

装置本身的功能性：运行速度快、循迹准确、承重大；

装置自身性质：智能搬运机器人自身质量轻便且能耗较少为佳。

#### 2.2.1.3 建立需求的相对重要性

1. 货物需求（☆☆☆☆☆）：抓取物体过程中的稳定性（☆☆☆☆☆）
2. 装置本身的功能性（☆☆☆）：运行速度快（☆☆☆），循迹准确（☆☆☆），承重大（☆☆）
3. 装置自身性质（☆☆）：智能搬运机器人自身质量轻便且能耗较少（☆☆）

**注：**☆多少代表对应需求重要性程度，☆☆☆☆☆较重要，☆☆较不重要

### 2.2.2 项目作品计划实施

在进行项目规划之前，我们首先对于用户的具体需求进行了简单的调研。调研对象主要面向各个高校的师生以及的科研工作者，共收集到100份有效数据，汇总如表2-1所示。

表2-1 用户需求统计表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 需求 | 重要性 |  |  |  |
| 1 | 完成抓取任务 | 1 |  |  |  |
| 2 | 减小产品质量以及空间体积，方便较小空间运行 | 2 |  |  |  |
| 3 | 减震、抗震设计 | 2 |  |  |  |
| 4 | 科学设计自动控制系统 | 1 |  |  |  |
| 5 | 提高电池容量，增强续航 | 2 |  |  |  |
| 6 | 保证制动能力 | 3 |  |  |  |
| 7 | 降低能耗 | 1 |  |  |  |

基于上述对于项目作品需求的分析与调研，我们制定了如下项目计划。在项目实施的第一阶段，我们将首先对于产品的硬件进行组装和搭建，过程中应该注意减小不必要的装置从而达到减轻质量的效果。在项目实施的第二阶段，我们会对于产品进行软件编程。在程序中需要稳定运行并循迹、到定点抓取物品、返回原点等多个功能。随后进入项目综合调试阶段。首先，我们将对相关硬件，也就是传感器的灵敏度进行调整，使其达到符合运行轨道的目的。其次我们将对我们的程序进行调试，力求采用轮巡控制与并行控制相结合的方式，实现整个过程的全自动控制。同时，对于设定的电机与舵机的控制方式与速度参数等，也将根据实际情况进行相应的调整。最后我们将通过实际的运行检验设计成果。

## 2.3 成员及分工

本组的组长是石云天，组员为袁萌启、李贺航、马天翼、季若鱼、李嘉树。经过组内会议讨论，拟定在接下来的项目中分工如下：

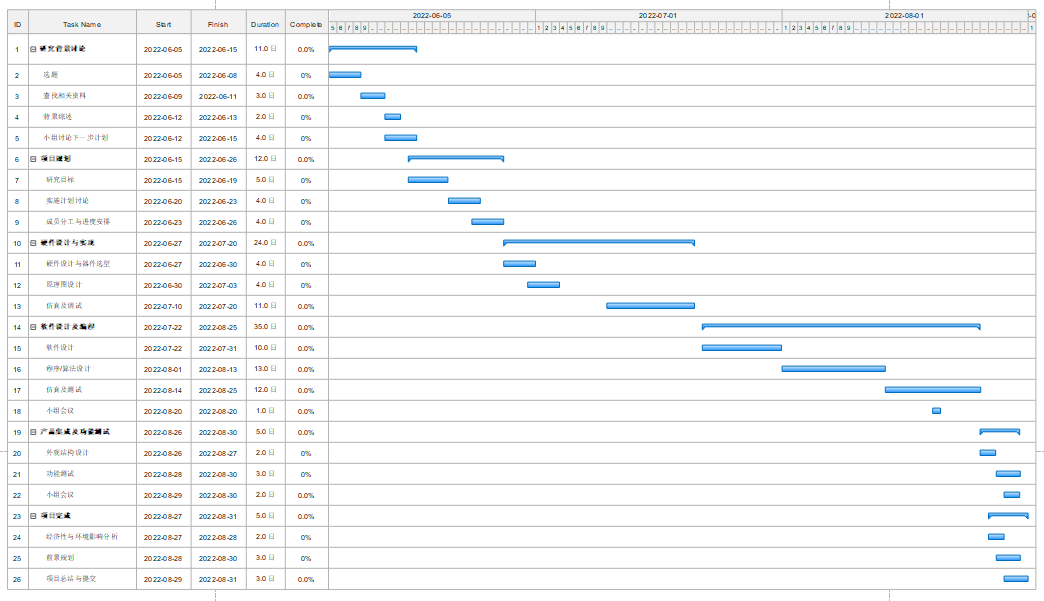
石云天：项目整体规划与进度管理；循迹与转向模块程序设计与功能测试；撰写课程报告。  
 袁萌启：原理图设计；声音模块设计；声控模块程序设计与功能测试。  
 李嘉树：硬件设计与器件选型；抓取模块程序设计与功能测试；撰写课程报告。  
 马天翼：外观及结构设计；声音模块设计；声控模块程序设计与功能测试。  
 李贺航：原理图设计；循迹与转向模块程序设计与功能测试；制作答辩PPT。  
 季若鱼：研究背景调研；抓取模块程序设计与功能测试；制作答辩PPT。

## 2.4 进度安排

按本项目的实际规模和建设要求，大致的项目进度见表2-2所示，按任务需求调研安排大致需要88个工作日，以6月5日为工作启动，基本将在8月31日左右结束。

表 2-2 项目进度表

按本期项目的实际规模和建设要求，计划大致的项目进度甘特图如图2-2所示。

图 2-2 项目进度甘特图

其中，由于软件设计及编程部分为我们组的主要目标，因此本组将在该部分投入大量时间。在项目进行过程中，本组还将定期进行组会，总结前一阶段出现的问题和不足，制定改进计划；对后一阶段进行细化分工和适当调整项目目标计划。

# 第三章 硬件设计与实现

## 3.1 硬件设计与器件选型

硬件主要由厂商提供，组装后智能履带车的整体功能参数如表3-1所示。接下来对于主要的传感器模块参数进行逐一介绍。

表3-1 Tankbot规格参数

|  |  |  |
| --- | --- | --- |
| 名称 | Tankbot | |
| 机体参数 | 产品尺寸（长\*宽\*高） | 230\*300\*570mm |
| 机体重量（未含电池） | 2.16kg |
| 机体材料 | 硬铝合金 |
| 供电部分 | 正常工作电压 | 6.9V-7.4V |
| 电池 | 7.4V锂电池 |
| 充电时间 | 3-4h |
| 满电续航能力 | 100分钟 |
| 主控部分 | 控制板 | 开源控制板 |
| 单片机 | STM32单片机模板 |
| 功能参数 | 承重能力 | 0.5kg |
| 极限车速 | 0.27m/s |
| PS2手柄控制 | 至少10m（空旷20m） |
| 手机app控制 | 至少10m（空旷15m） |
| 爪子举重能力 | 0.22kg |
| 爬坡能力 | 25° |
| 超声波探测距离 | 2cm-400cm（精度3mm） |
| 控制部分 | 见下文所述 | |

总线舵机包括两种。LX-824舵机是由幻尔科技研发的双轴三端口串行总线舵机。舵机精度高，虚位小，扭矩大(17KG.cm)，具有反馈功能，三端口接线更便捷。LX-1501舵机是由幻尔科技研发的单轴三端口串行总线舵机。舵机精度高，虚位小，扭矩大(17KG.cm)，具有反馈功能，三端口接线更便捷。两个舵机分别安装在机械臂的不同位置，其实物图如图3-1所示。

图3-1 LX-1501舵机实物图

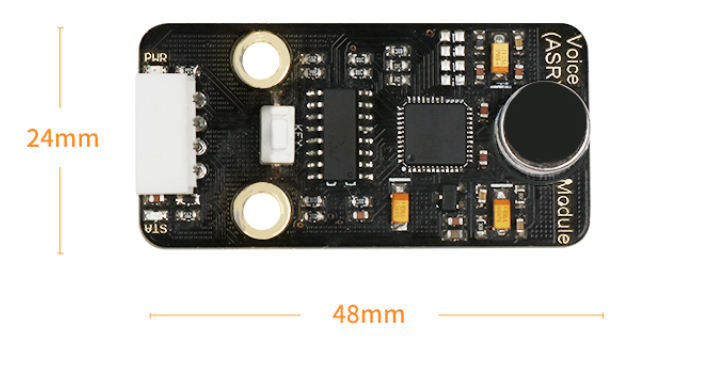
语音识别模块的实物图如图3-2所示，其可通过编程语言控制，使用方法简单，用户无需深入了解语音识别原理，通过IIC通信，即可识别语音，输出结果，可直接应用于多种人机交互领域的开发实验中。

图3-2 语音识别模块实物图

语言识别模块的规格参数表如表3-2所示。

表 3-2 语音识别模块规格参数表

|  |  |
| --- | --- |
| 工作电压 | 5V |
| 长度 | 48mm |
| 宽度 | 24mm |
| 通讯方式 | IIC |

4路循线传感器的实物图如图3-3所示。该传感器利用红外原理，能提前探测到大角度转弯的黑线，可以完成高难度巡线，且I2C输出不占用IO口



图3-3 4路循线传感器实物图

4路循线传感器模块的规格参数表如表3-3所示。

表3-3 4路循线传感器规格参数表

|  |  |
| --- | --- |
| 工作电压 | 5V |
| 工作电流 | 10mA-50mA |
| 工作温度 | 0℃-50℃ |
| 检测距离 | 5mm-15mm（可调） |
| 安装孔径 | M4铜柱 |
| 长度 | 72.5mm |
| 宽度 | 44mm |
| 输出接口 | 4PIN接口 |
| 输出信号 | I2C总线 |

## 3.2 原理图设计

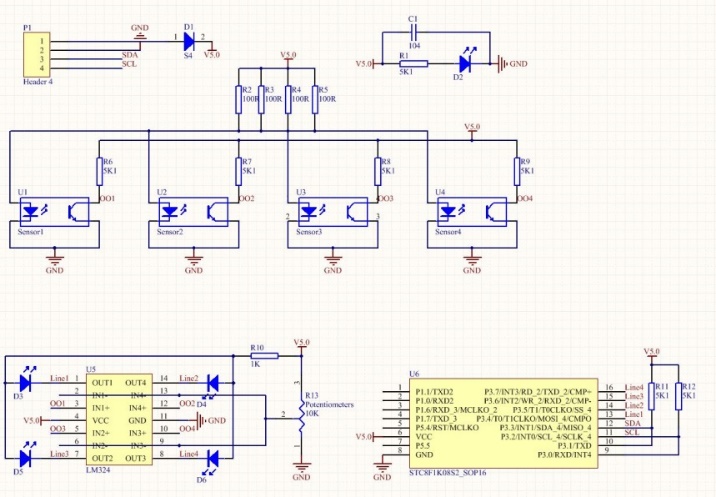
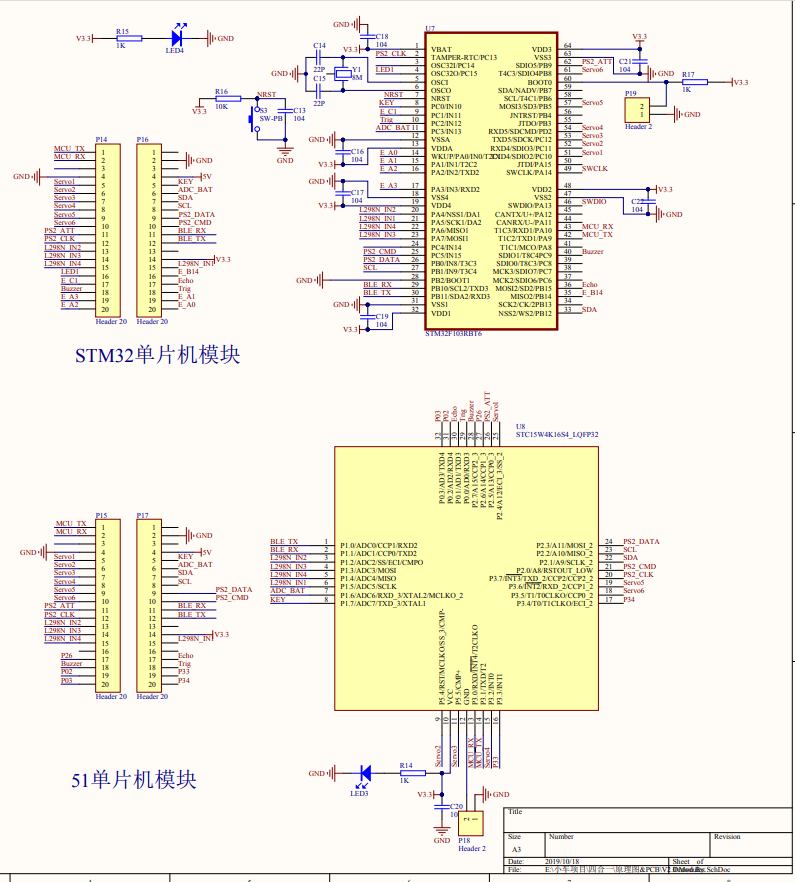
4路循线传感器的电路原理图如图3-4所示。其中各路传感器灯的亮灭均采用二极管控制。

图 3-4 4路循线传感器原理图

STM32单片机原理图如图3-5所示。小车主控器原理图如图3-6所示。

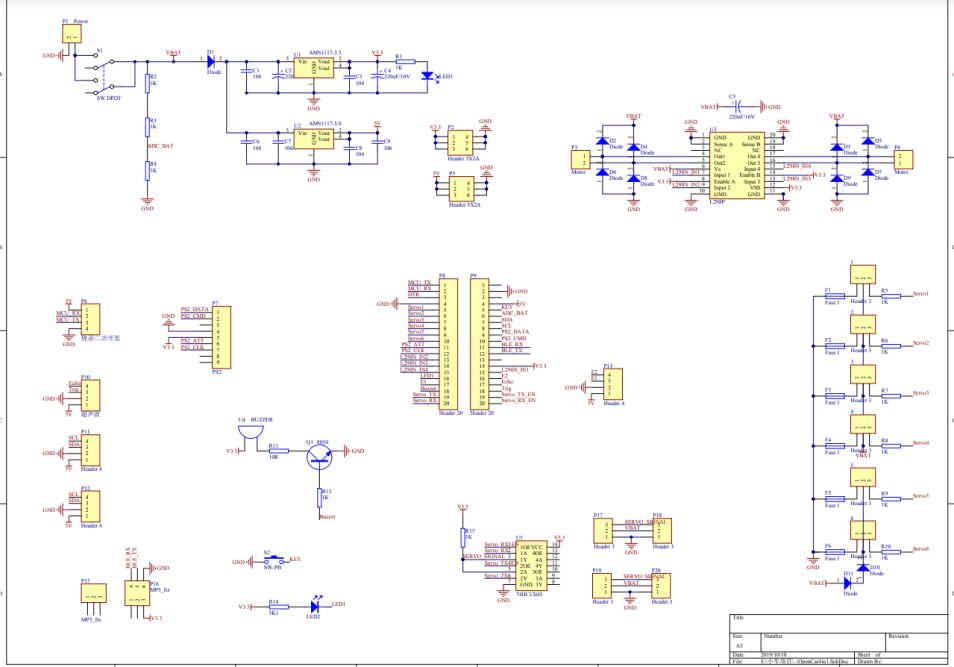
图 3-5 STM32单片机原理图

图3-6 小车主控器原理图

## 3.3 仿真及调试结果

首先，对于4路循线传感器进行调试。根据传感器原理，当传感器检测到暗光时，应反馈灯灭，检测到亮光时，应反馈灯亮。根据循线传感器放置的位置，调节灵敏度旋钮，使得每路传感器在检测到黑线时，反馈灯灭；在未检测到黑线时，反馈灯亮。

其次，对机械臂舵机进行调节，使其能够平稳地抓起并放下物体。

# 第四章 软件设计与编程

## 4.1 软件设计

在本项目中，我们需要操控智能搬运机器人模拟现实物流过程的操作步骤，在功能上设计了自动循迹、声控抓取、抓取后回程这三大主要功能。在软件设计方面，stm32以C语言为主体，以及少量的汇编语言，支持模块化编程，以此来实现我们的功能。

功能图如下图4-1所示：

图4-1 软件设计功能图

## 4.2程序/算法设计

1、算法流程框图

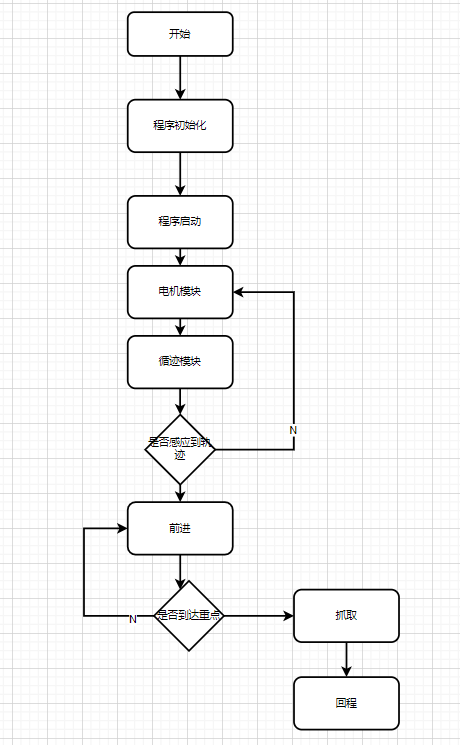
根据目标功能，设计算法流程框图如图4-2所示。图中，由两个闭环控制构成整个系统的全自动化，从而达到预期设定的目标。

图 4-2 算法设计流程图

2、程序代码

如下：

bool grab\_s = FALSE; //??????

static bool left = FALSE;

static bool right = FALSE;

bool getSound(void)

{

CheckSound();

if (GetSound() > 60)

{

DelayMs(5);

if (GetSound() > 60)

return TRUE;

}

return FALSE;

}

void Grab(void)

{

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(2, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 140, 1000);

BusServoCtrl(2, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 810, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 765, 1000);

DelayMs(1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 600, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

DelayMs(1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 130, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 810, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 765, 1000);

DelayMs(1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 0, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

DelayMs(1000);

grab\_s = FALSE;

}

void leftGrab(void)

{

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(2, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

DelayMs(1000);

// BusServoCtrl(6,SERVO\_MOVE\_TIME\_WRITE,875,1000);

// DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 200, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 250, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 780, 1000);

DelayMs(1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 600, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

DelayMs(1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 140, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 250, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 780, 1000);

DelayMs(1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 200, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(2, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

DelayMs(1000);

DelayMs(1000);

left = FALSE;

}

void rightGrab(void)

{

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(2, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

DelayMs(1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 140, 1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 200, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 250, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 780, 1000);

DelayMs(1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 600, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

DelayMs(1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 875, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 250, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 780, 1000);

DelayMs(1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 200, 1000);

DelayMs(1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

DelayMs(1000);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(2, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

DelayMs(1000);

DelayMs(1000);

right = FALSE;

}

int main(void)

{

SystemInit(); //系统时钟初始化为72M SYSCLK\_FREQ\_72MHz

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2); //设置NVIC中断分组2:2位抢占优先级，2位响应优先级

InitDelay(72); //初始化延时函数

InitServo(); //舵机初始化

InitBuzzer(); //蜂鸣器初始化

InitLED(); //初始化LED指示灯

InitADC(); // ADC电池电量初始化

InitTimer2(); //定时器2初始化

InitMotor(); //电机初始化

IIC\_init(); // IIC初始化

DelayMs(100);

Usart1\_Init(); //串口初始化

InitBusServoCtrl();

DelayMs(100);

BusServoCtrl(1, SERVO\_MOVE\_TIME\_WRITE, 600, 1000);

BusServoCtrl(2, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

BusServoCtrl(3, SERVO\_MOVE\_TIME\_WRITE, 350, 1000);

BusServoCtrl(4, SERVO\_MOVE\_TIME\_WRITE, 100, 1000);

BusServoCtrl(5, SERVO\_MOVE\_TIME\_WRITE, 300, 1000);

BusServoCtrl(6, SERVO\_MOVE\_TIME\_WRITE, 500, 1000);

DelayMs(1000);

while (1)

{

u8 val;

bool voice = FALSE;

TaskTimeHandle(); //舵机驱动与ADC检测

IIC\_start(); //起始信号

IIC\_send\_byte(0xF0); //发送写指令

IIC\_wait\_ack(); //应答

IIC\_send\_byte(0x01); //发送寄存器地址

IIC\_wait\_ack(); //应答

IIC\_start(); //重新启动

IIC\_send\_byte(0xF1); //发送读指令

IIC\_wait\_ack(); //应答

val = IIC\_read\_byte(0); //读取一个字节

IIC\_stop(); //停止

// printf("%d%d%d%d",val & 0x01,(val>>1) & 0x01, (val>>2) & 0x01, (val>>3) & 0x01);

// //串口打印4个传感器的状态，串口波特率9600，1表示检测到黑线，0表示检测到白线，从左到右对应sensor1-sensor4

// printf("\n");

// DelayMs(500);

if (((val & 0x01) == 1) & (((val >> 1) & 0x01) == 1) & (((val >> 2) & 0x01) == 1) & (((val >> 3) & 0x01) == 1))

{

MotorControl(0, 0);

voice = getSound(); //检测声音

if (voice)

{

leftGrab();

MotorControl(-80, -80);

DelayMs(1000);

MotorControl(80, -80);

DelayMs(5000);

while (1)

{

TaskTimeHandle(); //舵机驱动与ADC检测

IIC\_start(); //起始信号

IIC\_send\_byte(0xF0); //发送写指令

IIC\_wait\_ack(); //应答

IIC\_send\_byte(0x01); //发送寄存器地址

IIC\_wait\_ack(); //应答

IIC\_start(); //重新启动

IIC\_send\_byte(0xF1); //发送读指令

IIC\_wait\_ack(); //应答

val = IIC\_read\_byte(0); //读取一个字节

IIC\_stop(); //停止

MotorControl(80, -80);

if ((((val >> 1) & 0x01) == 1) | (((val >> 2) & 0x01) == 1))

break;

}

}

}

// 停

else if (((((val >> 3) & 0x01) == 1)))

MotorControl(80, -80);

else if (((((val >> 2) & 0x01) == 1) & (((val >> 1) & 0x01) == 0)))

MotorControl(80, -80);

else if (((((val >> 3) & 0x01) == 1) & (((val >> 2) & 0x01) == 1) & (((val >> 1) & 0x01) == 1) & ((val & 0x01) == 0)))

{

MotorControl(80, -80);

}

// 左右转分界线

else if (((((val >> 3) & 0x01) == 0) & (((val >> 2) & 0x01) == 1) & (((val >> 1) & 0x01) == 1) & ((val & 0x01) == 1)))

{

MotorControl(-80, 80);

}

else if (((((val >> 1) & 0x01) == 1) & (((val >> 2) & 0x01) == 0)))

MotorControl(-80, 80);

else if ((((val & 0x01) == 1)))

MotorControl(-80, 80);

// 执行

else

MotorControl(65, 65);

}

}

## 4.3 仿真及测试结果

在各模块程序设计与仿真测试的过程中，由于理想状况与实际测试环境之间的差异，出现了许多意料之外的状况，经过反复调整与试验，最终实现了所有预设目标，具体问题与调整措施如下：

1.调节直行前进速度，避免直角转弯速度过快，来不及转弯冲出，经过调试转速从80降至50。  
 2.调节转弯逻辑，使其能顺利通过直角弯，从只使用中间两个传感器来控制循迹，到使用外面两个传感器来控制循迹。  
 3.使用外面两个传感器循迹，导致直线行驶调整不敏感，难以直线行走，速度慢，改用同时使用四个传感器来控制循迹。中间两个传感器主要作用在直线行走阶段，外面两个传感器主要作用在直角转弯阶段，能同时使用这两种策略的原因是，直线循迹阶段，外面两个传感器始终保持未检测到黑线状态，避免了两种逻辑产生冲突。  
 4.在判断停止阶段，判断逻辑中，黑线停止的条件达成的同时，一定会达成左右转条件，在修改循迹过程中，误把左右转判断放在停止判断之前，导致永远无法停止，后经过多次调试和代码的重新阅读，发现问题，通过修改条件判断的先后顺序解决问题。  
 5.在抓取阶段，从只能抓取具有一定高度的物体，通过修改舵机6的参数到可以抓取放在地面上的小球。  
 6.在抓取阶段，增加语音控制，增加声控模块，并修改声音阈值的大小，可以方便的捕捉到声音，实现通过声音控制抓取的实行与否。  
 7.增加抓取后返回的功能，首先，考虑抓取后直接进行原地旋转，通过控制延时来控制旋转过程，虽此方法简单易行，但此方法代码需要通过多次尝试才能得到可行的延时时间，同时其没有拓展性，更换场地后便无法正常工作。然后考虑，原地旋转使用传感器判断停止，当传感器脱离停止状态后重新回到执行状态，停止旋转，但此方法难以需要考虑两次判断状态的关联性，代码难度较高。之后考虑避免停止黑线区域的影响，在完成抓取后，先控制车身后退，退出停止黑线区域，在控制进行原地旋转，使传感器脱离循迹线，再执行沿相同方向旋转循环，直至传感器重新回到直线行驶状态，退出循环，继续执行循迹程序，此方法可行性较高，泛用性较强，逻辑清晰，最终选用此方法完成。

# 参考文献

[1] 宋康,薛凯阳,卢帆,杨鸣远,谭洪.基于STM32的智能搬运机器人设计[J].信息技术与信息化,2021

[2] 彭周萍,王丽芳,满达虎.基于STM32控制的竞赛用智能物流搬运机器人设计[J].轻工科技,2022

[3] 姚思嘉,刘芸,邵铭旭,王鹏家.基于STM32的智能搬运机器人的研究与设计[J].电子测试,2021

[4] 李爽,张志勇,于跃,杨威,赵航.智能应用型搬运机器人设计[J].科学技术创新,2021

[5] 牟胜辉,王有亮,张成,蒋修华,何伟锋.应用于智能仓储的AGV货架搬运机器人[J].机电工程技术,2021

[6] 翟优. 机场行李智能搬运机器人末端执行器的研究与设计[D].吉林大学,2021

[7] 朱颖,黎伟健,郭国辉,严宗灿.单目视觉智能分拣搬运机器人的设计[J].国外电子测量技术,2020

[8] 赵炳生.基于Arduino智能搬运机器人控制系统设计[J].南方农机,2019

[9] 原焕林. 搬运机器人智能控制系统的设计[D].上海交通大学,2014.

[10] 苏鹏升,吕东旭,徐教礼,李沪.搬运机器人控制系统设计[J].中外企业