

**大学生创新创业项目实训课程报告**

**题目： 基于stm32的语音识别垃圾桶**

**姓 名 郑梦乔、魏昕宇、李子鸣**

**学 号 201910720236、**

**201910720239、**

**201910720228**

**年 级 2019级**

**专 业 电子信息工程**

**教 师 董杰**

**学 院 信息工程学院**

2021年 月 日

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | 郑梦乔 | 魏昕宇 | 李子鸣 |
| **分工** | 软件搭建 | 资料查询、报告填写 | 硬件搭建 |
| **成绩** |  |  |  |

目录

[一、课程设计目的与要求 1](#_Toc73726264)

[（一）设计目的 1](#_Toc73726265)

[（二）设计要求 1](#_Toc73726266)

[二、设计方案 1](#_Toc73726267)

[（一）硬件系统设计 1](#_Toc73726268)

[（二）软件设计 3](#_Toc73726269)

[三、电路仿真和电路的实现 5](#_Toc73726270)

[（一）电路原理图 5](#_Toc73726271)

[(二) PCB图 6](#_Toc73726272)

[（三）万用板焊接 7](#_Toc73726273)

[四、软件调试(核心软件展示) 8](#_Toc73726274)

[五、发现的问题、排除方法和改进措施总结 17](#_Toc73726275)

[六、结论 17](#_Toc73726276)

[七、参考文献 18](#_Toc73726277)

# 

# 一、课程设计目的与要求

# （一）设计目的

(1)巩固所学的相关理论知识；

(2)实践所掌握的电子制作技能；

(3)会运用Multisim工具对所作出的理论设计进行模拟仿真测试,进一步完善理论设计；

(4)通过查阅手册和文献资料,熟悉常用电子器件的类型和特性，并掌握合理选用元器件的原则；

(5)掌握模拟电路的安装\测量与调试的基本技能,熟悉电子仪器的正确使用方法，能力；

(6)分析实验中出现的正常或不正常现象(或数据)独立解决调试中所发生的问题；

(7)学会撰写课程设计报告；

# （二）设计要求

本项目针对垃圾分类问题，通过语音识别、报警、提示功能、单片机的控制功能和垃圾监测功能对办公场所、寝室等环境下进行方便和有效地垃圾分类投放。以单片机作为核心系统，利用电机及其驱动模块控制垃圾桶的开盖及转动角度。考虑到舵机可以简单并且精确地控制其输出轴旋转角度和方向，因此本项目使用步进电机来驱动垃圾桶开盖。在本项目的众多模块中语音识别和控制是关键功能模块，本项目使用了snr6812vr “语音识别播报”专用芯片。当用户使用此垃圾桶时，说出所投垃圾的相关词汇，例如“玻璃”、“塑料” 等，通过语音识别和控制模块，检索snr6812vr芯片中已经录入的垃圾种类关键词可以将桶盖打开并旋转到相应位置，从而实现垃圾分类投放的目的。

# 二、设计方案

# （一）硬件系统设计

本系统的主要硬件设备包括STM32F103C8T6单片机控制系统、snr6812vr 语音识别播报模块、语音自动打开和关闭桶盖、负责垃圾桶盖开合的由SG90舵机模拟开合，电源需要分别给控制板模块和电机模块供电。语音识别芯片snr6812vr是一块基于非特定人语音识别技术的声控芯片。芯片上集成了高精度的A/D和D/A接口，不再需要外接辅助的Flash和RAM，识别的关键词语列表可以动态编辑，事先只需要设定好要识别的关键词语，并把这些关键词语以字符的形式传送到snr6812vr内部进行转换，然后就可以对用户说出的关键词语进行识别，再做出相应的操作，避免了因识别模糊而造成的困扰。语音识别模块直接分析处理原始语音，提高了识别速度，符合垃圾分类对语音识别的要求。系统工作流程如图1所示。

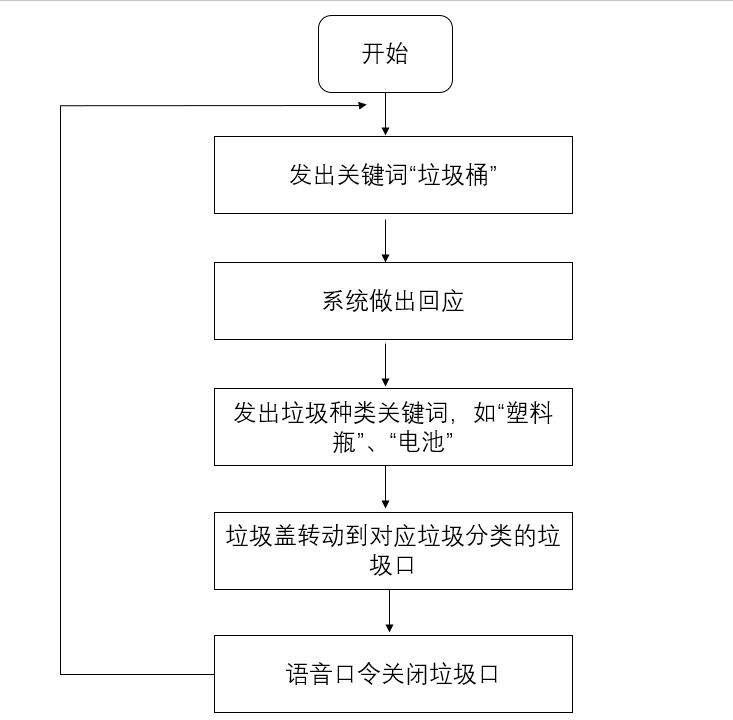


图1 总体结构图

snr6812vr模块控制系统根据识别到的关键词直接控制电机打开桶盖并旋转到相应位置，并且集成了播报系统可将识别的结果通过喇叭播报出来。垃圾投放完毕之后，系统会自动关闭垃圾桶盖，避免垃圾异味扩散。5V舵机马达操作简单。系统配有SG90舵机的标准接口，检修以及使用时可以随时插拔。

# （二）软件设计

**1.语音识别模块驱动设计**

本项目中的语音识别模块控制中以中断响应为主，具体流程如图2所示

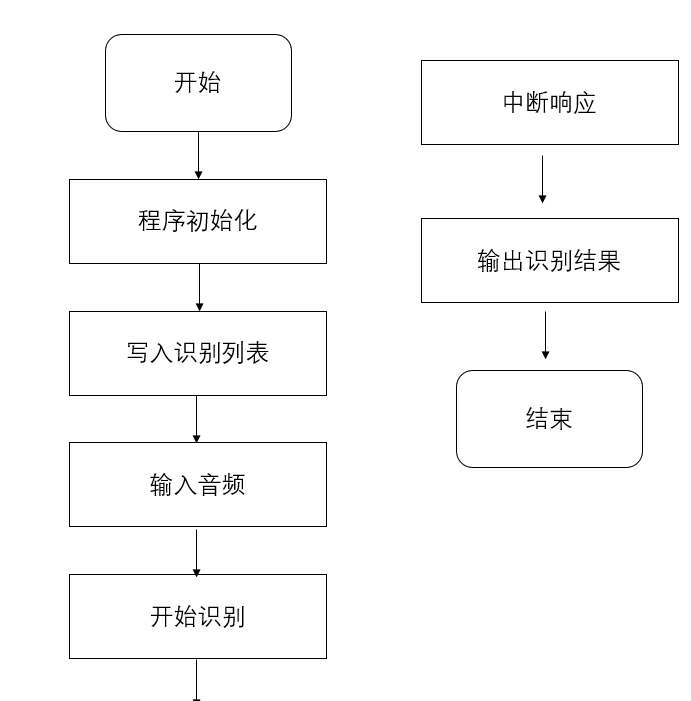


图2 语音识别模块流程图

首先是程序的初始化，包括寄存器的设置、软复位、时钟频率设定、FIFO 设定等，用于激活芯片内部的数据模块，等待输入。

第二步是写入识别列表，即需要识别的语音关键词，项目中使用的snr6812vr芯片可支持最多50个识别条目，每个识别条目必须是标准普通话的汉语小写拼音，每2个字(汉语拼音)之间需插入-一个空格间隔。同时每个识别条目对应一个编号， 编号可以任意设置，但在数值上必须小于256 （O0H-FFH),将识别条目按照顺序存储在寄存器中。本项目共设置了16个识别条目，即各种垃圾的名称拼音，如su liao、dian chi等。

然后，输入音频。对单片机说出某些语音，系统开始识别，程序进入中断，如图3所示。在经历了语音频谱分析、特征值提取后，将特征值保存在另一寄存器中，与写入的识别列表进行比较匹配，找到匹配度最高的识别条目作为输出。

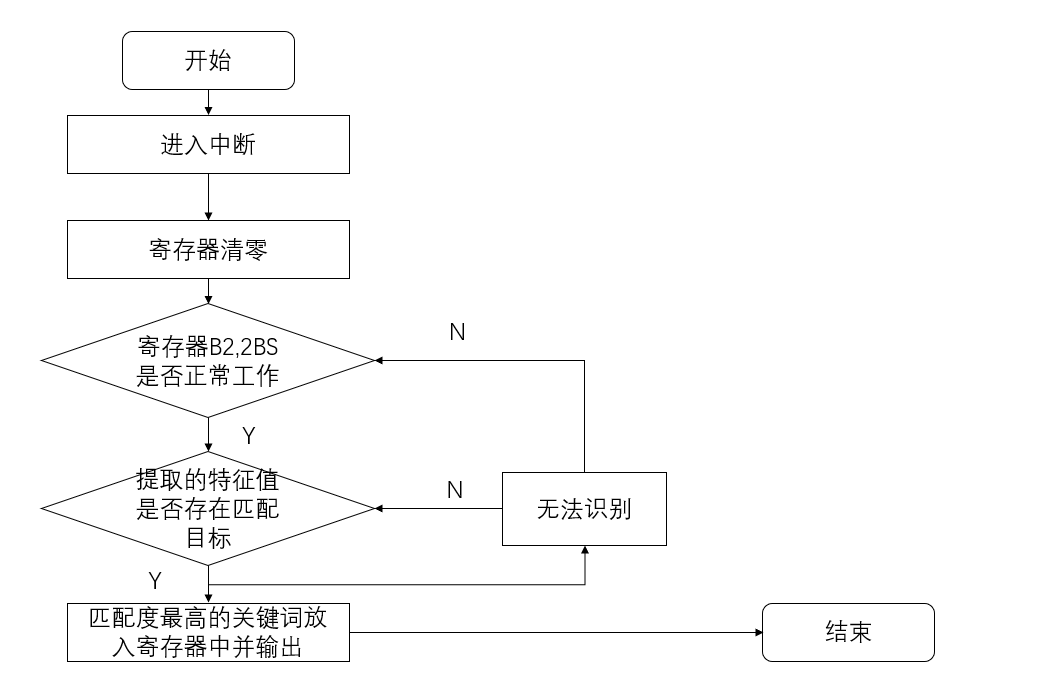


图3 中断程序流程图

**2.电机模块驱动设计**

本项目采用的舵机电机，首先在主函数外编写-一个控制电机转过指定角度的函数，这个角度值由函数的形式参数给出，然后在主函数中就可以方便的通过更改调用时的实际参数来控制电机转过任意的角度。更改节拍和角度，从而达到控制电机的转速和频率。

理论上电机的减速比为164,但在设计的过程中，我们发现实测的减速比为1 63.684，所以在实际电机的运行中会产生0.0049的误差，即电机转100圈就会产生半圈的误差，所以我们按照1 63.684的实际减速比，设定转过一圈所需要节拍数为64X 63.684≈4076。

电机初始化确认了角度和节拍，再进入语音识别中断程序，识别到了正确关键词后，进入对应的电机驱动程序，根据不同的识别条目分属的垃圾种类，电机转子转动不同的圈数，使得对应垃圾口移动到对应的垃圾筐上，其工作流程如图4所示。

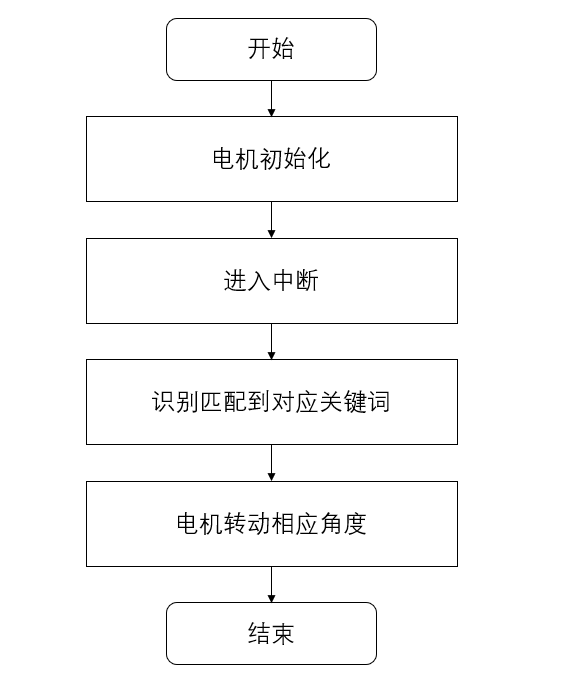


图4 电机驱动流程图

# 三、电路仿真和电路的实现

## （一）电路原理图

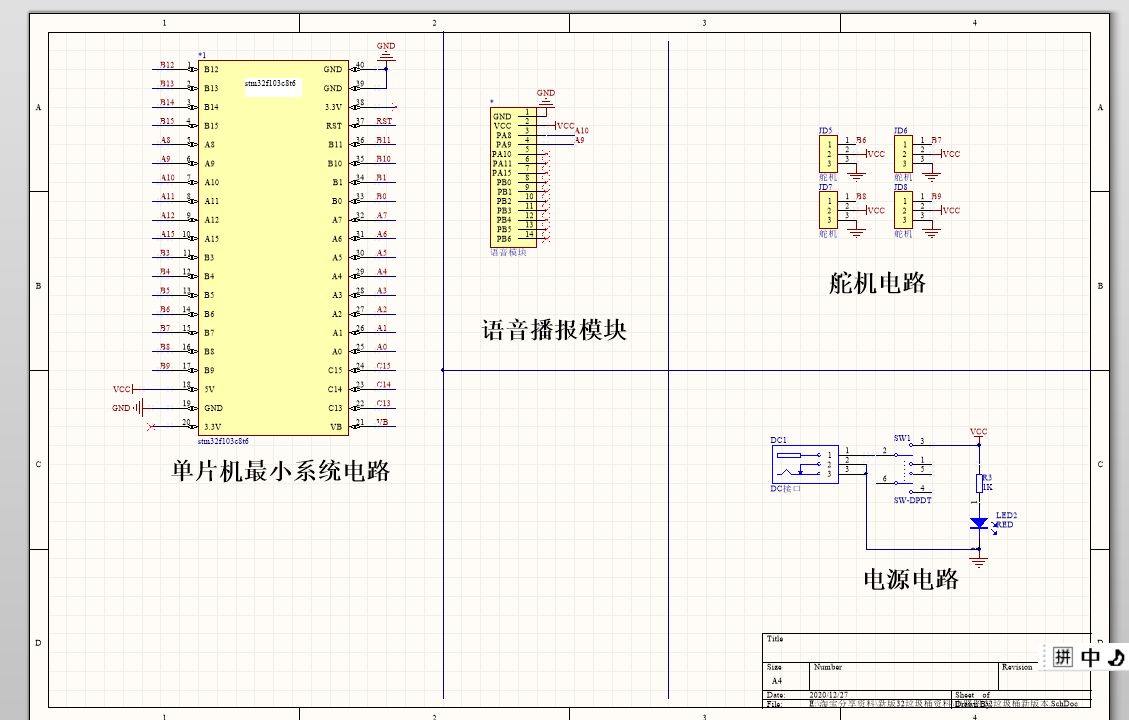


图5电路仿真图

## (二) PCB图

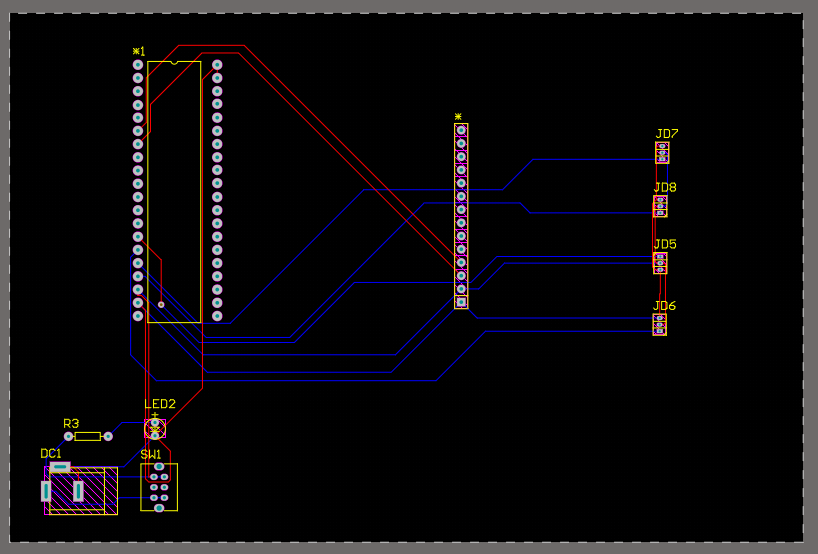


图5-1PCB 2D图

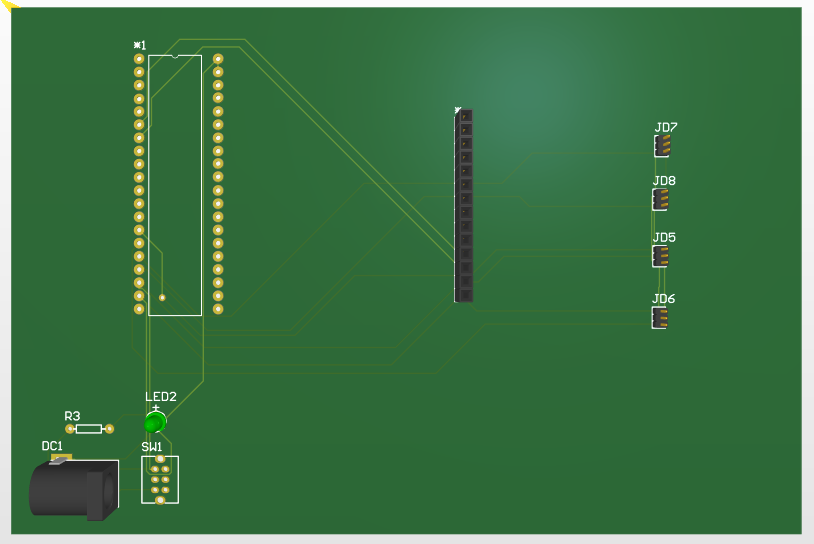


图5-2 PCB 3D图

## （三）万用板焊接

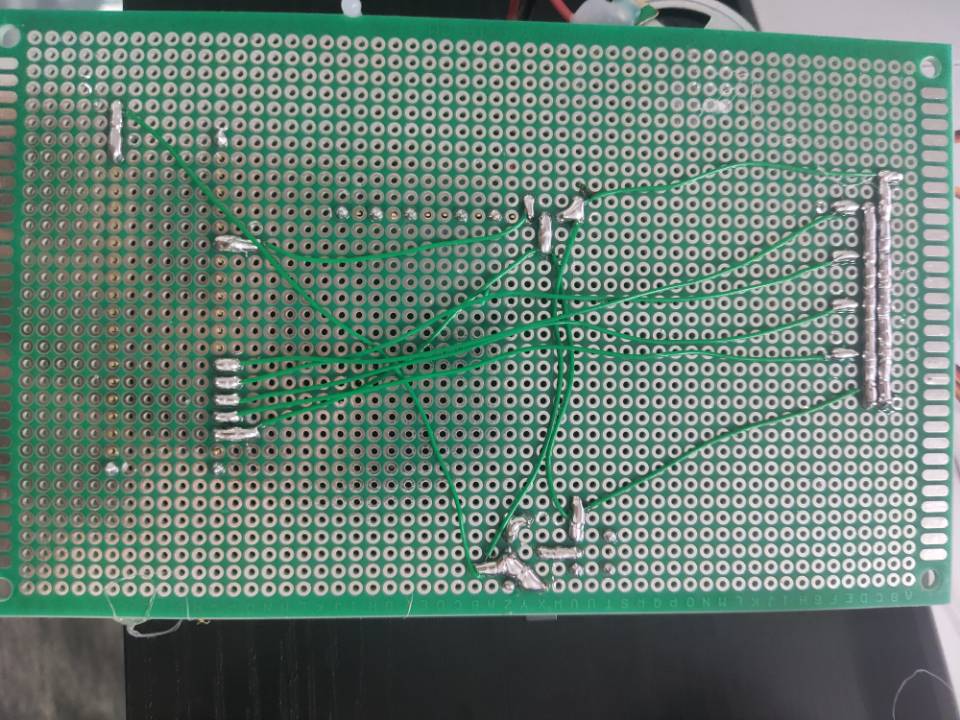


图5-3万用焊接板制作图

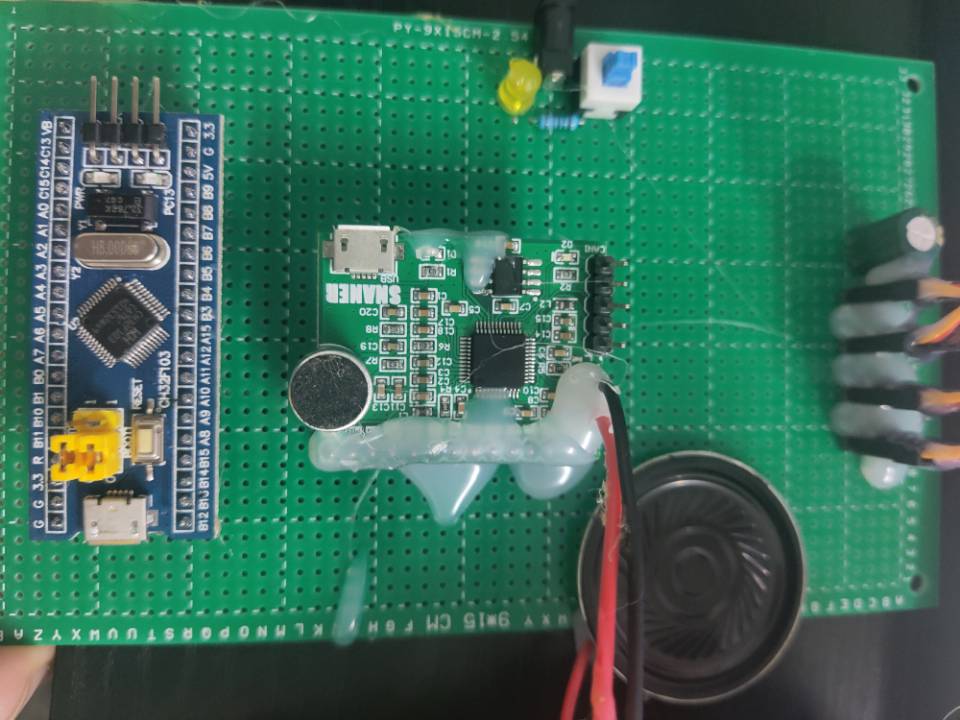


图5-4万用焊接板制作图

# 四、软件调试(核心软件展示)

**Main.c**

#include "sys.h"

u8 Target1=0;//舵机控制时间

u8 Target2=0;

u8 Target3=0;

u8 Target4=0;

u8 Open\_Time=0;//开垃圾桶单位时间

u8 WakeUp\_Flag=0;//唤醒标志位

char CSBStr[20] = {0}; //超声波字符串形式

float csb = 0; //超声波

float rongqishen = 10; //容器深度

float distance = 0; //

unsigned char rekey=0;

int main(void)

{

SystemInit(); //参数为向量表的偏移地址

delay\_init();

TFT\_Init();//TFT初始化

gpio\_Init();

HC05\_Init();

MP3\_Init();//初始化MP3语音播报模块

Voice\_Init(); //初始化语音识别模块

SysTick\_Init();//初始化多功能定时器

PWM\_Init();//初始化PWM

TFT\_Clear(BLACK);

TFT\_PutString(10,10,"READY......",RED,BLACK);

delay\_ms(10000);

delay\_ms(10000);

TFT\_Clear(BLACK);

TFT\_PutString(5,0," 智能垃圾桶 ",RED,BLACK);

TFT\_PutString(25,0,"垃圾桶状态 关闭 ",RED,BLACK);

while(1)

{

csb = GET\_Distance(); //获得超声波距离

if(csb>10)

{

csb = 0;

TFT\_PutString(65,16,"\* \* \* \* ",RED,BLACK);

}

else if(csb>=8&&csb<=10)

{

TFT\_PutString(65,16," ",RED,BLACK);

}

else if(csb>=6&&csb<=8)

{

TFT\_PutString(65,16,"\* ",RED,BLACK);

}

else if(csb>=4&&csb<=6)

{

TFT\_PutString(65,16,"\* \* ",RED,BLACK);

}

else if(csb>=2&&csb<=4)

{

TFT\_PutString(65,16,"\* \* \* ",RED,BLACK);

}

else if(csb<=2)

{

TFT\_PutString(65,16,"\* \* \* \* ",RED,BLACK);

}

else

{

TFT\_PutString(65,16,"\* \* \* \* ",RED,BLACK);

}

CSBStr[0] = 0;

sprintf(CSBStr,"垃圾桶余量:%.1f",csb);

TFT\_PutString(45,0,CSBStr,RED,BLACK);

if(Infrared1 == 0)

{

if(rekey == 0)

{

delay\_ms(200);

if(Infrared1 == 0)

{

rekey =1;

MP3\_Star(2);

Target1 = 1;

}

}

}

if(Voice\_RX\_STA == 1)//收到语音指令

{

if(MP3\_State() == 0)

{

if(Voice\_RX\_CMD==49 || Voice\_RX\_CMD==50)//唤醒

{

WakeUp\_Flag = 1;//唤醒词标志位置位

MP3\_Star(1);

}

else if(WakeUp\_Flag == 1)

{

WakeUp\_Flag=0;

if(Voice\_RX\_CMD%4 == 1)//开启垃圾桶

{

MP3\_Star(2);

Target1 = 1;

}

else if(Voice\_RX\_CMD%4 == 2)//关闭垃圾桶

{

MP3\_Star(3);

Target2 = 1;

}

// else if(Voice\_RX\_CMD%4 == 3)//有害垃圾

// {

// MP3\_Star(3);

// Target3 = 1;

// }

else//可回收物

{

MP3\_Star(4);

Target4 = 1;

}

}

}

Voice\_RX\_STA = 0;

}

if(SysTick\_JumpEdge(&Open\_Time, &SysTickBit.Time1s)==1)//开垃圾桶单位时间到

{

if(Target1 != 0)//开启垃圾桶

{

TFT\_PutString(25,0,"垃圾桶状态 开启 ",RED,BLACK);

if(rekey == 0)

Target1++;

TIM\_SetCompare1(TIM4,1000);//打开

if(Target1>15)

{

TIM\_SetCompare1(TIM4,400);//时间到关闭

Target1 = 0;

rekey =0;

TFT\_PutString(25,0,"垃圾桶状态 关闭 ",RED,BLACK);

}

}

if(Target2 != 0)//关闭垃圾桶

{

TFT\_PutString(25,0,"垃圾桶状态 关闭 ",RED,BLACK);

TIM\_SetCompare1(TIM4,400);//打开

Target1 = 0;

Target2=0;

rekey =0;

}

}

}

}

#include "voice.h"

//加入以下代码,支持printf函数,而不需要选择use MicroLIB

#if 0

#pragma import(\_\_use\_no\_semihosting)

//标准库需要的支持函数

struct \_\_FILE

{

int handle;

};

FILE \_\_stdout;

//定义\_sys\_exit()以避免使用半主机模式

\_sys\_exit(int x)

{

x = x;

}

//重定义fputc函数

int fputc(int ch, FILE \*f)

{

while((USART1->SR&0X40)==0);//循环发送,直到发送完毕

USART1->DR = (u8) ch;

return ch;

}

#endif

u8 Voice\_RX\_CMD;//接收缓冲,最大USART\_REC\_LEN个字节.末字节为换行符

u8 Voice\_RX\_STA;//接收状态标记

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

USART初始化

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Voice\_Init()

{

//GPIO端口设置

GPIO\_InitTypeDef GPIO\_InitStructure;

USART\_InitTypeDef USART\_InitStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1, ENABLE); //使能USART1

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE); //GPIOB时钟

USART\_DeInit( USART1);

//USART1\_TX

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//USART1\_RX

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;//浮空输入

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//Usart1 NVIC 配置

NVIC\_InitStructure.NVIC\_IRQChannel = USART1\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=2 ;//抢占优先级3

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0; //子优先级3

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //IRQ通道使能

NVIC\_Init(&NVIC\_InitStructure); //根据指定的参数初始化VIC寄存器

//USART 初始化设置

USART\_InitStructure.USART\_BaudRate = 9600;//串口波特率

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;//字长为8位数据格式

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;//一个停止位

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;//无奇偶校验位

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;//无硬件数据流控制

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx; //收发模式

USART\_Init(USART1, &USART\_InitStructure); //初始化串口

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);//开启串口接受中断

USART\_Cmd(USART1, ENABLE); //使能串口

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

串口2中断接收数据

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void USART1\_IRQHandler(void) //串口2中断服务程序

{

u8 res=0;

if(USART\_GetITStatus(USART1, USART\_IT\_RXNE) != RESET) //接收中断(接收到的数据必须是0x0d 0x0a结尾)

{

res=USART\_ReceiveData(USART1); //读取接收到的数据

if(Voice\_RX\_STA==0)

{

Voice\_RX\_CMD = res;

Voice\_RX\_STA=1;

}

}

}

#include "play.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MP3\_USART\_Init(void)

{

//GPIO端口设置

GPIO\_InitTypeDef GPIO\_InitStructure;

USART\_InitTypeDef USART\_InitStructure;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, ENABLE); //使能USART2

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC, ENABLE);

//USART2\_TX

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//USART2\_RX

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_3;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;//浮空输入

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//Busy

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_14;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);

//USART 初始化设置

USART\_InitStructure.USART\_BaudRate = 9600;//串口波特率

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;//字长为8位数据格式

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;//一个停止位

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;//无奇偶校验位

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;//无硬件数据流控制

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx; //收发模式

USART\_Init(USART2, &USART\_InitStructure);

USART\_Cmd(USART2, ENABLE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

串口发送数据

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MP3\_Com(u8 com)

{

USART\_ClearFlag(USART2,USART\_FLAG\_TC);

USART\_SendData(USART2,com);

while(!USART\_GetFlagStatus(USART2,USART\_FLAG\_TC));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MP3初始化

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MP3\_Init(void)

{

MP3\_USART\_Init();

delay(1000);

MP3\_Com(0xAA);//选择Flash播放

MP3\_Com(0x0B);

MP3\_Com(0x01);

delay(1000);

MP3\_Com(0xAA);//音量30

MP3\_Com(0x13);

MP3\_Com(0x01);

delay(1500);

MP3\_Com(0xAA);//音量30

MP3\_Com(0x13);

MP3\_Com(0x01);

delay(1500);

}

void MP3\_Star(u32 Number)

{

u8 check=0;

check=0xB3+Number;

MP3\_Com(0xAA);

MP3\_Com(0x07);

MP3\_Com(Number);

MP3\_Com(check);

}

u8 MP3\_State(void)

{

return GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_14);

}

#include "pwm.h"

void PWM\_Init(void)

{

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStruct;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB,ENABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4,ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO,ENABLE);

GPIO\_InitStructure.GPIO\_Pin=GPIO\_Pin\_6|GPIO\_Pin\_7|GPIO\_Pin\_8|GPIO\_Pin\_9;

GPIO\_InitStructure.GPIO\_Speed=GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode=GPIO\_Mode\_AF\_PP;

GPIO\_Init(GPIOB,&GPIO\_InitStructure);

GPIO\_ResetBits(GPIOB,GPIO\_Pin\_9);

//定时器初始化

TIM\_TimeBaseStruct.TIM\_Period=9999;//初值

TIM\_TimeBaseStruct.TIM\_Prescaler=143;//预分频

TIM\_TimeBaseStruct.TIM\_ClockDivision=0;

TIM\_TimeBaseStruct.TIM\_CounterMode=TIM\_CounterMode\_Up;//向上

TIM\_TimeBaseInit(TIM4,&TIM\_TimeBaseStruct);

//pwm 初始化

TIM\_OCInitStructure.TIM\_OCMode=TIM\_OCMode\_PWM1;

TIM\_OCInitStructure.TIM\_OutputState=TIM\_OutputState\_Enable;

TIM\_OCInitStructure.TIM\_OCPolarity=TIM\_OCPolarity\_High;

TIM\_OC1Init(TIM4,&TIM\_OCInitStructure);

TIM\_OC2Init(TIM4,&TIM\_OCInitStructure);

TIM\_OC3Init(TIM4,&TIM\_OCInitStructure);

TIM\_OC4Init(TIM4,&TIM\_OCInitStructure);

TIM\_OC1PreloadConfig(TIM4,TIM\_OCPreload\_Enable);

TIM\_OC2PreloadConfig(TIM4,TIM\_OCPreload\_Enable);

TIM\_OC3PreloadConfig(TIM4,TIM\_OCPreload\_Enable);

TIM\_OC4PreloadConfig(TIM4,TIM\_OCPreload\_Enable);

TIM\_Cmd(TIM4,ENABLE);

TIM\_SetCompare1(TIM4,400);

TIM\_SetCompare2(TIM4,400);

TIM\_SetCompare3(TIM4,400);

TIM\_SetCompare4(TIM4,400);

}

# 五、发现的问题、排除方法和改进措施总结

在本次课程设计的过程中碰到了很多的困难，比如一开始不知道设计思路、不懂很多仿真原理，到后来不知道怎么使用面包板等等。一开始还比较迷茫，但是我们没有失去信心，通过不断地查阅文献，不断地思考，问老师、同学，最终大体上算是成功,了，这让我们明白了学习理论知识的重要性，做任何设计都需要强大的理论知识。

# 六、结论

通过这次对项目设计的课程设计，让我们更加明白了设计电路的程序，更加深入了解了项目设计的设计理念和思路，也更加扎实的掌握了有关电子产品设计。尽管这次实验做得并不是很顺利，在设计过程中碰到了很大的困难。一开始还比较迷茫，但是我们没有失去信心，通过不断地查阅文献，不断地思考，最终大体上算是成功,了，让我们明白了学习理论知识的重要性，做任何设计都需要强大的理论知识。

通过本次设计，不仅有效巩固了本学期所学数电的相关知识，加强了对重要知识点的记忆和理解。首先，加深了我对数字电路知识的了解，尤其是数字芯片的性能和使用。其次，使我对数字电路设计更感兴趣，原来它是很奥秘的，一些电子功能可以通过组合各种各样的元器件，从而产生想要实现的功能，关键在于能够设计出符合要求的电路。再次，使我对课本知识得到巩固和加强，毕竟课本知识是抽象的，只有真正在实践中利用它，做到学以致用，才能加深对它的理解，所以设计出这个题目后，我发现以前对课本有疑问的地方都随着在课程设计中豁然开朗了。最后，加强了我思考和解决问题的能力。

此次电路设计让我得到了多方面的锻炼。一方面我电子技术专业知识有了进一步的了解，并熟悉了更多的电子仪器的使用方法，了解常用的电子器件的类型和特征，同时学会了如何合理选用电子器件的原则。也通过电路设计、安装、调试等各环节，也培养了我运用课程中所学的理论与实践紧密相结合，独立地解决实际问题的能力。让我知道了在以后的学习中，生活中，我会勤于思考，勤于动手，不拍困难的好习惯。再一方面电子电路的安装与调试技能培养了我创新能力和对治学要严谨的态度。

# 七、参考文献

[1]鲁菁.智能垃圾桶中的语音识别技术应用研究[J].电脑编程技巧与维护,2021(05):114-116.

[2]郭建军,林丽君,陈红斌,王克强,蔡肯.基于声音识别的智能声控垃圾桶[J].科学技术创新,2020(03):95-96.

[3]张家欣,许树华,姜彦吉.基于TMS320C5416的非特定人识别智能声控垃圾桶系统设计[J].电子世界,2018(07):122-123.

[4]阮顺,徐如意,赵飞洋,郑阿勇.一种基于物联网的智能垃圾箱[J].科学技术创新,2019(27):84-85.