智能识别垃圾桶的设计

摘　要

当前国内外对于智能识别垃圾桶的研究主要集中在如何智能化的实现垃圾分类和提高处理效率。国内外的研究学者将一些先进的传感器应用在了垃圾桶的运行过程中实现垃圾桶内部和外部的环境检测中，随着互联网和物联网技术的快速进步，垃圾桶的远程控制需求也逐渐被挖掘，使得垃圾桶能够满足智能家居设备的设计和控制要求。

本文基于 STM32F103 单片机实现了款智能识别垃圾桶的设计，该产品能够同时实现厨余、可回收、有害和其他一共 4 个的独立垃圾桶控制。通过超声波传感器和火焰检测传感器分别对垃圾桶内部的垃圾余量和火灾情况进行检测，，通过 OLED 显示模块实现了系统运转过程中垃圾桶内部余量、开关状态等参数的检测结果，使用舵机控制垃圾桶的开合，同时在系统运转过程中，基于语音控制模块可以对使用者所播报垃圾内容进行判断，并打开相应的垃圾桶。此外在远程控制方面通过 ESP8266 模块实现了单片机和手机 APP 的链接，实现了数据的同步和控制信息的传输。

综上所述，本文所设计的智能识别垃圾桶是一款较为可行且智能化程度较高的垃圾桶设计解决方案，具有显著的市场推广价值。

关键词：智能垃圾桶；语音识别；STM32F103；ESP8266

Design of Intelligent Identification Garbage Bin

Abstract

The current research on intelligent identification of garbage bins at home and abroad mainly focuses on how to intelligently achieve garbage classification and improve treatment efficiency. Domestic and foreign researchers have applied some advanced sensors to detect the internal and external environment of garbage bins during their operation. With the rapid progress of Internet and Internet of Things technology, the remote control needs of garbage bins have gradually been explored, enabling garbage bins to meet the design and control requirements of smart home devices.

This article implements the design of an intelligent recognition trash can based on the STM32F103 microcontroller. This product can simultaneously control a total of four independent trash cans: kitchen waste, recyclable, harmful, and others. The ultrasonic sensor and flame detection sensor are used to detect the residual amount of garbage and fire situation inside the garbage bin. The OLED display module is used to achieve the detection results of parameters such as the residual amount and switch status inside the garbage bin during system operation. The servo is used to control the opening and closing of the garbage bin. At the same time, during system operation, the voice control module can judge the content of garbage broadcast by the user, And open the corresponding trash can. In addition, in terms of remote control, the ESP8266 module has been used to link the microcontroller with the mobile app, achieving data synchronization and control information transmission.

In summary, the intelligent recognition garbage bin designed in this article is a feasible and highly intelligent garbage bin design solution, with significant market promotion value.

Keywords: Intelligent trash can; Speech recognition; STM32F103; ESP8266

目　录

1 绪论

1.1 研究背景及意义

随着全球城市化进程的加速，城市人口急剧增长，垃圾产量也随之增加。传统的垃圾处理方式已经无法满足快速增长的垃圾处理需求，而智能识别垃圾桶的设计提供了一种高效率、智能化的垃圾分类和管理解决方案[1]。这种智能化技术不仅提高了垃圾分类的准确性和效率，也大大降低了人工操作的成本，从而缓解了城市垃圾处理压力。此外，技术的不断进步也为智能识别垃圾桶的设计提供了可能[2]。计算机视觉、传感器技术和人工智能的发展使得垃圾桶能够实现对垃圾进行自动识别和分类。机器学习算法的应用，使得智能垃圾桶能够逐渐学习适应不同环境和垃圾种类，提高了分类的准确性和普适性[3]。智能识别垃圾桶的设计具有深远的意义。首先，它可以极大地提高垃圾分类的准确性和效率，降低了传统垃圾分类过程中的人为错误率，推动了垃圾资源的高效回收利用。其次，它减轻了城市垃圾处理压力，降低了处理成本，提高了垃圾处理的效率。此外，通过智能垃圾桶的大规模应用，可以收集到大量的垃圾数据[4]。为城市管理提供科学依据，实现智能化城市管理。最重要的是，它培养了市民的环保意识，推动了社会绿色发展的进程[5]。

1.2 国内外研究现状

1.2.1 国外研究现状

在国外，尤其是一些发达国家，例如美英等国家的城市化进程不断加快。垃圾产生量大幅增加，如果不加以有效管理和分类处理，会导致环境污染。为了提高垃圾分类的准确性，减缓对环境的污染，许多专家和学者对智能垃圾桶的相关领域投入了大量的研究[6]。2017 年，来自美国斯坦福大学设计了基于语音识别的智能垃圾桶。是一种利用语音识别技术，使用户可以通过语音命令控制垃圾桶的开合和操作的智能化设备。这种智能垃圾桶配备了麦克风和语音处理模块，能够识别用户的口头指令，并根据指令自动执行相应的操作[7]。例如打开或关闭垃圾桶盖、启动垃圾压缩机等。同时基于语音识别的智能垃圾桶可以支持多种语言，满足不同地区和用户群体的需求，还可以根据用户的个性化需求进行定制。该垃圾桶的设计完美符合了现代化、智能化的城市管理需求[8]。2019 年英国的帝国理

工学院提出了机器视觉的垃圾分类系统的设计[9]。该系统利用计算机视觉技术，通过摄像头和图像识别算法，实现对垃圾进行自动分类。当有新的垃圾被投放时，摄像头捕捉到的图像会经过预处理，然后输入到训练好的分类模型中[10]。模型会进行分类判断，确定垃圾属于哪个类别，例如可回收垃圾、有害垃圾、厨余垃圾等。一旦系统完成垃圾分类，结果可以通过显示屏、语音提示或网络接口等形式反馈给用户或相关工作人员。这种反馈可以帮助用户确认垃圾是否被正确分类，并在错误分类时进行指导[11]。该垃圾分类系统的设计为城市管理和环境保护提供了有力支持。2021 比西法尼亚大学提出了基于太阳能的智能垃圾桶设计[12]。它结合了太阳能技术和智能传感器技术，旨在提高垃圾桶的效能和便利性。由于智能垃圾桶配备了太阳能电池板，利用太阳能光源充电。因此不需要外部电源，减少了对传统电力资源的依赖，降低了能源消耗和环境污染[13]。当有人接近垃圾桶时，传感器会检测到，触发垃圾桶盖自动打开，方便人们投放垃圾。投放完垃圾后，盖子会自动关闭，避免了传统垃圾桶需要手动开合盖的不便。该设计使得垃圾处理更加智能、高效、方便，有助于提高垃圾分类和回收利用的效率。

1.2.2 国内研究现状

我国是一个发展中的人口大国，每日的城市建设和人们的日常生活同样会产生大量的垃圾。因此，设计一款智能垃圾桶，不仅有助于提高人们的生活质量，还能加快城市的发展。为此，我国大量的研究团队正在致力于开发具有自主知识产权的智能垃圾桶技术[14]。2018 年，南京大学提出了基于物联网的智能垃圾桶。这是利用物联网技术，将传感器、通信模块和垃圾桶结合起来，实现智能化管理和监测的垃圾桶系统。该智能垃圾桶配备各种传感器，如红外线传感器、超声波传感器、重量传感器等[15]。这些传感器可以检测垃圾桶内的垃圾容量、填充状态、垃圾种类等信息。同时还可以收集大量的数据，包括垃圾产生的时间、地点等信息。利用这些数据，垃圾收集服务提供商可以优化垃圾收集路线，提高效率，减少能源消耗和排放[16]。2019 年兰州大学提出了基于 NB-IoT 的智能垃圾桶系统设计。NB-IoT 是一种低功耗、长续航、远程传输距离远的物联网技术。智能垃圾桶系统利用 NB-IoT 技术，将传感器、通信模块和云平台相结合，实现了智能化的垃圾管理。利用传感器监测垃圾桶周围环境的温度和湿度，通过 NB-IoT 模块发送到云平台，云平台接收传感器数据，进行数据处理和存储[17]。可以使用数

据分析算法来预测垃圾桶的填充状态和最佳清空时间。减少了人力和时间成本，同时也有助于垃圾分类和环境保护[18]。2022 年重庆交通大学提出了基于智能垃圾桶的垃圾分类动态收运路径优化问题研究[19]。这是利用智能垃圾桶技术，结合垃圾分类与动态路径规划算法，对城市垃圾收运过程进行优化的研究。通过传感器获取智能垃圾桶的实时数据，包括各个垃圾桶的填充状态和种类信息。利用无线通信技术将传感器数据传输到中央服务器。使用智能路径规划算法，结合实时数据，确定最优的垃圾收运路径。通过研究垃圾分类动态收运路径优化问题，可以实现垃圾收运过程的智能化[20]。提高垃圾分类准确性，降低垃圾收运成本，减少环境污染，推动城市可持续发展。

1.3 本文主要研究内容

本文的主要内容如下：

第一章：主要介绍了智能识别垃圾桶的研究背景及意义。同时介绍了国内外对于智能识别垃圾桶的不同的研究方案，如基于语音识别的智能垃圾桶，基于 NB-IoT 的智能垃圾桶系统设计等等。最后介绍了本文结构。

第二章：介绍了智能识别垃圾桶的总体设计方案。主要包括核心控制芯片介绍、超声波检测模块介绍、无线数据传输模块介绍等等。

第三章：根据系统的总体设计方案设计硬件电路，并介绍各个子功能模块的电路。例如显示模块设计，舵机控制模块设计以及 WiFi 模块设计等等。

第四章：介绍本文所需的仿真软件和编程语言。绘制流程图并跟根据流程图设计程序。

第五章：进行系统调试，主要包括软件调试，硬件调试以及测试各个功能。

第六章：对智能识别垃圾桶的展望和总结。

2 系统总体设计方案

2.1 方案设计

本文所设计的基于单片机的智能识别垃圾桶设计方案，如图 2.1 所示。

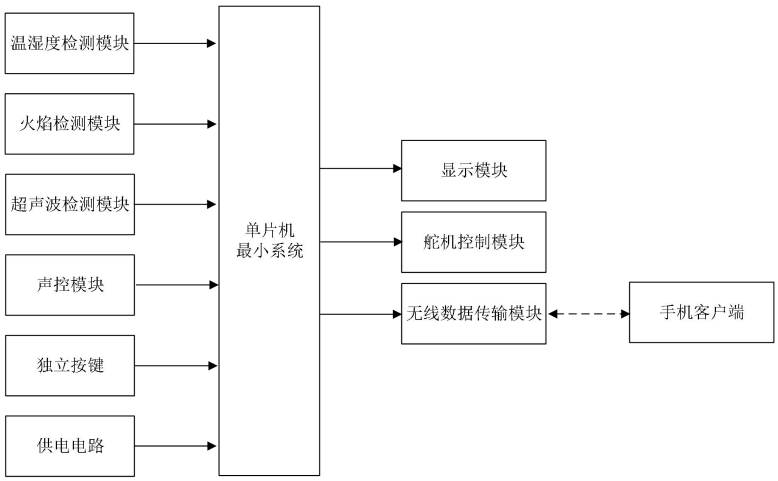


图 2.1 方案设计图

本文的总体系统设计方案包含了输入、输出和单片机最小系统三个部分。输入模块包含了温湿度检测模块、火焰检测模块、超声波检测模块、声控模块、独立按键和供电电路 6 个主要模块。温湿度检测模块，对垃圾桶所处环境的温度和湿度进行检测；火焰检测传感器对垃圾桶内部的火灾情况进行检测；超声波检测模块对垃圾桶内部的余量进行检测；声控模块基于语音指令，实现垃圾的识别以及垃圾桶的开合自动控制；独立按键实现了基于按键的垃圾桶控制；充电电路对系统的各功能模块进行供电。单片机最小系统实现了输入模块的输入，数据的控制以及输出模块执行机构的控制功能。输出模块包含了显示模块、舵机控制模块、无线数据传输模块以及手机客户端 4 个主要部分。其中显示模块用于显示系统运行过程中各参数的检测结果；舵机控制模块实现了 4 个独立垃圾桶盒盖的开合控制；无线数据传输模块基于 WiFi 模块实现了手机客户端和单片机之间的数据的同步以及控制信号的同步功能。以下对各主要功能模块进行介绍。

2.2 各模块芯片介绍

2.2.1 核心控制芯片介绍

本文选择 STM32F103 作为系统的控制核心，该控制核心作为当前主流的控

制核心之一，其是一种嵌入式的微控集成电路，主要以 32 位的内核处理器为核心，最高运行速度可达 72 兆赫兹，在内存储存类型方面，通过闪存储存的方式实现 48k 的内存容量。在电源引脚方面。其主要的引脚包含了 VDD、VBAT 和 GND 等。其中 VDD 和 GND 构成了外部电源供电电路，VBAT 实现了系统基于电池的供电方式，确保了系统的运行稳定性。同时，该系统还具备 NRST 复位引脚，在系统运行故障且无法修复时，可以通过该引脚实现相应的软件重置功能，该引脚通常处于高电平工作状态，在低电平时触发系统的复位功能，而复位功能可以将系统初始化至空状态， 确保其能够重新执行所有程序。该 STM32F103 实物如图 2-2 所示。

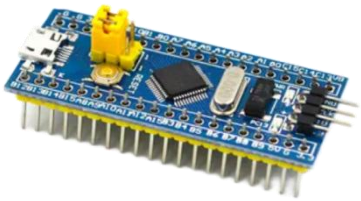


图 2.2 STM32F103 实物图

2.2.2 温湿度检测模块介绍

本文选择的 DHT11 传感器是一个基于 NPC 测温元器件的数字式温湿度检测传感器，通过其内部的电感式元器件，可以实现 8 位高精度的检测结果的传输，通过其内置的校准模块，可以实现较为精准的检测结果的校准，通过其内存中所存储的校准系数可以对不同环境下的环境检测结果进行调整，在校准完之后，将检测信号通过数据传输的 DATA 上传至单片机，实现较高的检测精度的回传。综上所述 DHT11 传感器是一款精度高且性能稳定的温湿度检测传感器能够具备宜家的使用性能。该 DHT11 传感器实物如图 2.3 所示。

图 2.3 DHT11 实物图

2.2.3 超声波检测模块介绍

本文选择的超声波检测模块的型号为 HC-SR04，该传感器的内部芯片能够具备自动增益和温度补偿的功能，尤其适合近距离的数据检测，对于物体追踪和

液位变化的检测精度较高。在工作温度方面，该模块能够支持零下 40 度至零上 125 度温度范围内的稳定运行，且能够根据温度的变化实现基于温度补偿的自动检测精度校准功能。在抗干扰能力方面，该模块内置的噪声抑制技术能够提高，测量的信噪比从而实现检测精度的提高。该模块的实物如图 2.4 所示。

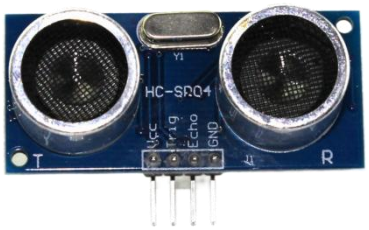


图 2.4 HC-SR04 实物图

2.2.4 声控模块介绍

本文选择 SU-03T 声控模块实现垃圾桶的垃圾分类语音的控制功能。此次研究所选用的语音识别模块是一种集成的语音控制模块，同时具有播报控制模块的功能。其主要功能是通过芯片和单片机进行语音的识别和相应指令的控制功能，可识别的语音指令多达五十条。该模块通过对语音指令的识别，并通过芯片将语音指令转换为数字信号输入给单片机执行相应的控制操作。同时，通过该模块的控制引脚，可以实现基于串口通讯协议的数据写入，提高了该模块的控制能力。该模块拥有一定的噪声过滤功能，能够在嘈杂的环境当中进行使用。该模块实物如图 2.5 所示。

图 2.5 SU-03T 实物图

2.2.5 显示模块介绍

OLED 模块作为当前最为常用的显示模块之一，其主要目的是实现设备运行参数的监控功能，本文选择的 OLED 显示模块主要由 OLED 显示屏幕、PCB 板和外围铁框共同构成，在显示方面，能够该显示模块能够具备自发光大视角和高

对比度的特点，尤其在刷新速度和刷新率方面，能够满足大多数使用账底下的需求，通过大范围的视角显示和快速的适度响应，可以有效提高图像的稳定性和亮度，在显示各种图片动画的过程中，能够通过其丰富的色彩提高其显示分辨率，此外在驱动电压方面，由于其发光原理较为简单，因此驱动电压交替能够满足各种不同的使用场景，通过太阳能电池板即可实现供电功能。综上所述，OLED 显示模块能够充分满足本文的设计功能时限要求，该 OLED 显示模块实物如图 2.6 所示。

图 2.6 OLED 实物图

2.2.6 舵机模块介绍

本文选择常规的 9G 舵机实现垃圾桶的开合控制，该型号的舵机的工作电压和单片机最小系统保持一致为 3.3-5V，通过单片机引脚发出不同占空比的 PWM 信号，实现舵机旋转角度的控制，该舵机模块实物如图所示。

图 2.7 9G 舵机实物图

2.2.7 无线数据传输模块介绍

本文选择 ESP8266WIFI 模块实现数据传输功能，该模块相较于其他数据通信芯片，其主要特点是能够支持 AT 指令、MAQTT 透传和局域网配网等功能，通过先进的封装工艺，实现了高度的集成和稳定的性能，基于屏蔽罩极大增强了

该模块在数据传输过程中的稳定性，通过压缩空间体。在提升抗干扰能力的同时提高了其开发的适应性。在天线形式方面主要采用了 PCB 板载天线，基于 ESP826EX 芯片实现最高 72.2Mbps 下的数据通信速度，当前该芯片被主要应用于智能家居、人工智能和智能工厂等领域。该模块的实物如图 2.8 所示。

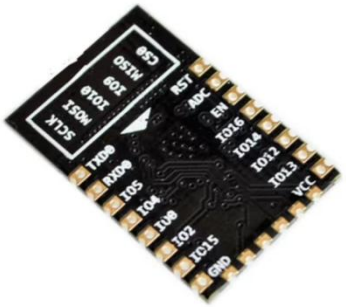


图 2.8 ESP8266 实物图

3 硬件电路设计

3.1 单片机最小系统模块设计

本文选用的单片机型号为 STM32F103，由于该单片机能够适用于复杂的应用，具有丰富的外设可以满足各种应用的需求，且拥有丰富的开发工具等优点，因此十分适用于本次实验，STM32F103 各个引脚的配置各个功能模块电路设计如图 3.1 所示。

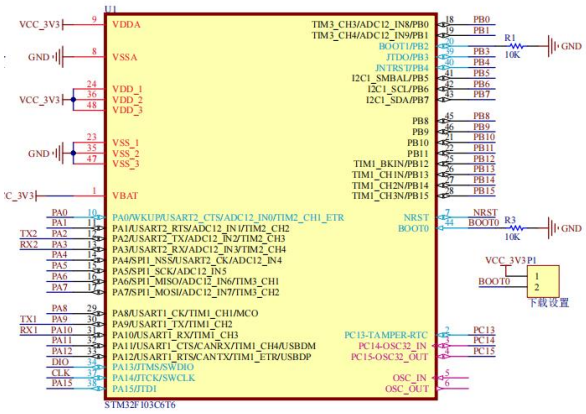


图 3.1 STM32F103 电路设计

在设计本次实验最小系统时，除了对于单片机引脚进行设计外，还需设计一个供电模块，为准确对系统经行供电，因此，在设计供电模块时加入了电源适配器实现对实验系统的供电需求。供电模块主要由 Type-C 电源母座和开关组成，Type-C 电源母座通过数据线与电源适配器的 USB 接口相连接，同时 Type-C 电源母座的 VBUS 引脚与电源适配器模块相连接。为控制引脚 1-6，本文则以六角按键作为开关模块，当按键还未按下时，引脚 1 和引脚 2 导通，当按键按下时，引脚 2 和引脚 3 导通。同时加入+5V 电源用于实现对电路的供电功能，并将其与电源适配器的 VIN 引脚连接。为直观准确的观测出电路是否供电，在该模块电路上加入了 LED 指示灯模块，当电路通电后，指示灯亮起时，供电正常，若指示灯未，则说明该电路存在问题。该模块设计如图 3.2 所示。

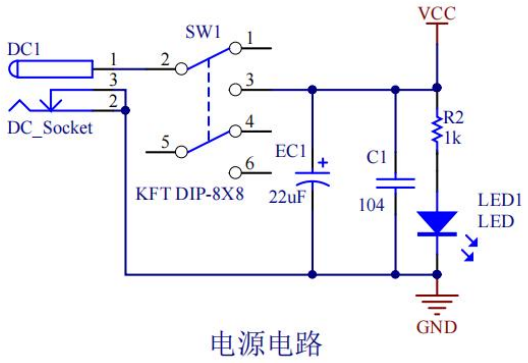


图 3.2 外部电源供电模块设计

本文的降压模块选择 AMS1117-3V3，通过该模块实现了外部电源供电模块 5V 向 3.3V 的转换功能，该模块的电路设计如图 3.3 所示。

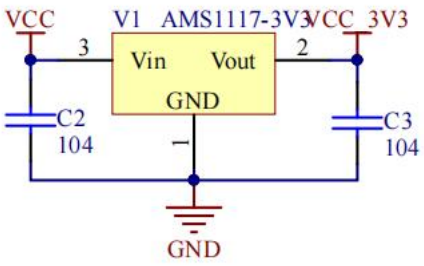


图 3.3 降压模块电路设计

3.2 温湿度检测模块设计

DHT11 电路设计方面，使用 5V 电压供电，DATA 是数据引脚，用于从传感器读取温度和湿度数据，通常，它与其他控制设备的数字输入引脚相连，本文则与单片机的 P20 端口连接，用于发送采集的信号，DHT11 模块通过 DATA 引脚与控制器通信，数据的传输通常采用单总线数字信号，遵循 DHT11 的通信协议，本文传输的数据格式为：粮仓湿度数据的 8 位整数+8 位小数+粮仓温度数据的 8 位整数+8 位小数。GND 引脚则是用于接地处理。温湿度检测 DHT11 模块的电路设计如图 3.4 所示。

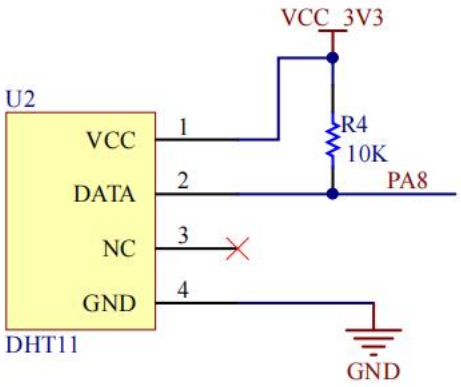


图 3.4 湿度检测模块电路设计

3.3 显示模块设计

显示模块的主要功能是显示实验中各种结果，例如湿度、温度、垃圾桶内部余量等等，OLED 显示屏通常通过 I2C 或 SPI 等通信协议与其他控制设备进行通信，以显示图像或文本等内容，引脚 SDA 和 SCL 分别为用于传输数据的数据线以及用于同步数据传输的时钟线。引脚 VCC 和 GND 分别连接电源的正负极。OLED 显示模块的电路设计如图 3.8 所示。

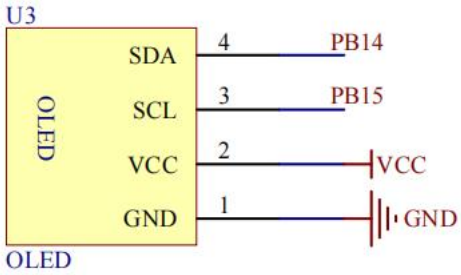


图 3.5 OLED 模块电路设计

3.4 独立按键设计

本文设计了 4 个独立按键分别为 S1-S4，对应 STM32F103 单片机的 PA0、PC13、PC14 和 PC15 共四个引脚，在未触发状态下，上述 4 个引脚均上拉为高电平，在按键触发之后，对应的引脚由高电平下拉为低电平，并触发相应的函数控制功能。该独立按键电路设计如图 3.6 所示。

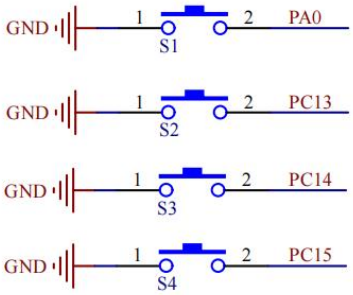


图 3.6 独立按键电路设计

3.5 超声波检测模块设计

为了对垃圾桶内部情况进行检测，本次实验选用了型号为 HC-SR04 的超声波模块。HC-SR04 是一款常用的超声波传感器模块，用于测量物体与传感器之间的距离。它通过发射超声波脉冲并测量脉冲返回的时间来计算距离。VCC 引脚和 GND 引脚连接电源的正负极。Trig 为触发引脚分别连接单片机的 PB3、PB5、PB7、PB9 引脚。通过该引脚发送触发信号，当该引脚接收到一个 10 微秒以上的高电平脉冲时，传感器开始工作。Echo 为回波引脚分别连接单片机 PB4、PB6、PB8、PB10 引脚，该引脚用于接收回波信号。当传感器发射超声波后，当回波信号返回时，该引脚会输出一个高电平脉冲，脉冲的宽度表示回波信号的时间。通过控制 Trig 引脚的高低电平信号以及测量 Echo 引脚的脉冲宽度，可以实现与目标物体的距离测量。该检测模块电路设计如图 3.7 所示。

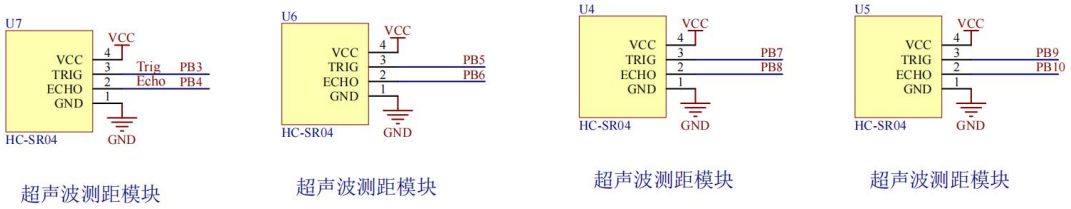


图 3.7 超声波检测模块电路设计

3.6 舵机控制模块设计

本文通过 4 个独立的舵机控制模块，实现舵机的控制模拟垃圾桶盖的开合操作。在舵机控制模块设计方面，此次研究选取了 SG90 型号的舵机，该型号舵机在市面上较为常见。在电路设计方面，VCC 和 GND 两个引脚用于给模块供电，I/O 引脚用于接收来自核心单片机的控制信号，舵机旋转角度的大小与控制信号

PWM 的占空比相关。PWM 控制信号的输出引脚分别为 PA6、PA7、PB0 和 PB1。舵机控制模块电路设计图如图 3.8 所示。

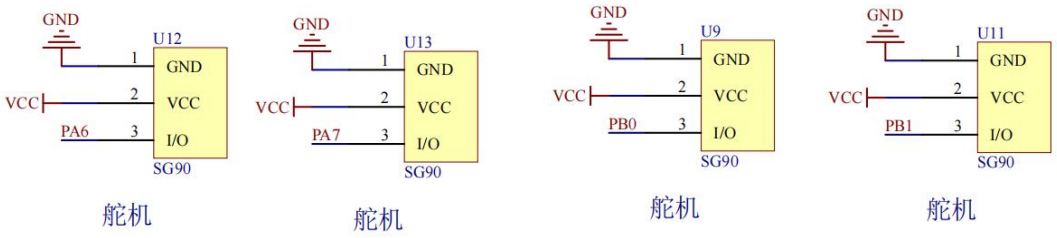


图 3.8 舵机模块电路设计

3.7 火焰检测模块设计

本文设计了 4 个独立的火焰检测传感器，分别实现垃圾桶内部的火焰检测功能，在输出方面使用 DQ 引脚，在检测到有火焰时输出高电平，使得单片机的对应角由低电平上拉为高电平，4 个引脚分别对应 STM32F103 的 PB11、PB12、PB13 和 PA11。该模块的电路设计如图 3.9 所示。

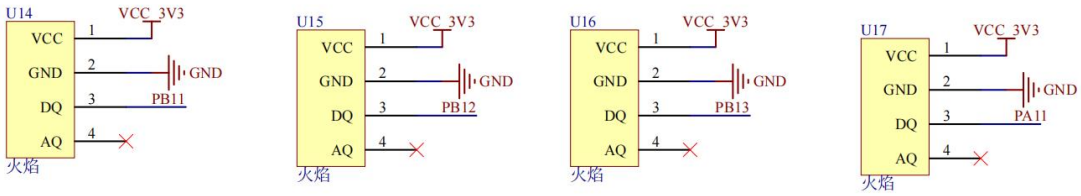


图 3.9 火灾检测模块电路设计

3.8 声控模块设计

本次研究所采用的语音识别模块型号为 SU-03T，其具备一定的存储功能，内部内置了语音识别所需要的唤醒词，及其他控制功能识别所需要的识别语句。在与核心的面积组成系统之前，可以通过 USB 接口与计算机相连，进行基于串口通信协议的数据传输和写入。在电路设计上，VCC 和 GND 接口用于给模块供电，RXD 和 TXD 引脚与核心单片机的 TX1 和 RX1 引脚相连，用于实现语音识别数据和控制数据的传输。语音识别模块的电路设计如图 3.10 所示。

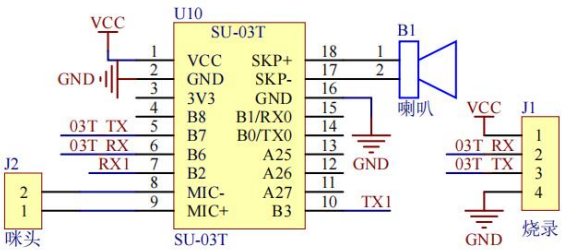


图 3.10 SU-03T 声控模块电路设计

3.9 WiFi 模块设计

本文选用 ESP8266 无线传输模块，用于实现单片机系统和手机之间的无线通信。该模块具有传输信号、稳定传输距离远的优点，最大传输距离达 50 米。在电路设计上，VCC 和 GND 接口用于给模块供电，RXD 和 TXD 引脚与核心单片机的 TX2 和 RX2 引脚相连，用于实现基于串口通信协议的数据传输。该模块在使用过程中基于 3.3V 的外部独立供电，通过 SW2 的拨杆实现了改模块数据传输模式的选择和切换，无线传输模块的电路设计如图 3.11 所示。

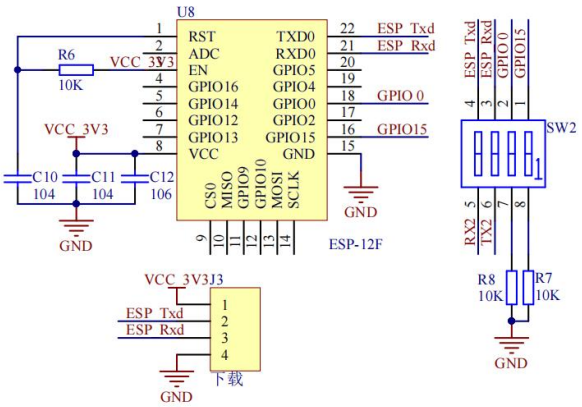


图 3.11 ESP8266 模块电路设计

3.10 系统总体技术方案设计

结合前文对各功能模块进行设计，为了合理有效的分配各个引脚的设计，避免某些引脚存在重复使用，本文对该系统经行了合理的设计，电路设计的方案如如图 3.12 所示。

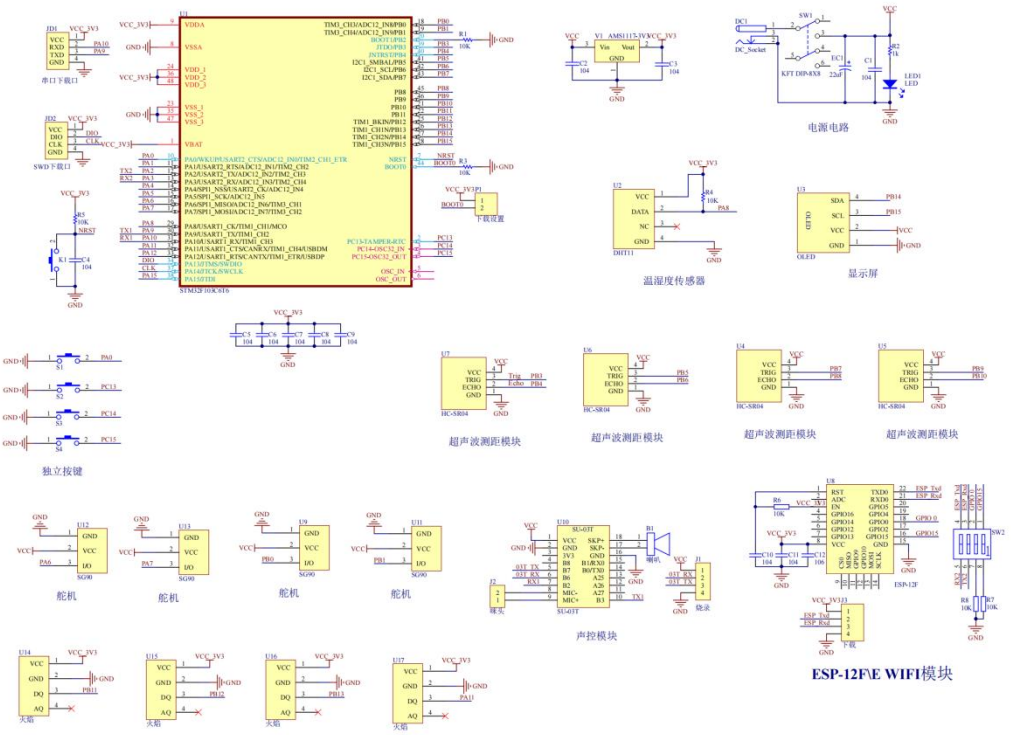


图 3.12 系统总体技术方案设计

4 系统程序设计

4.1 编程软件介绍

Keil5 作为一个完整的开发工具，在使用过程中能够为嵌入式系统提供包括编译调试仿真在内的全套，嵌入式开发应用功能，该开发环境具有以下几个方面的特点，一是能够支持各种不同的主流嵌入式系统，满足各种不同场景下的开发需求，二是具备强大的变异功能，能够对 C 语言的各种开发语言的规范进行检测并生成优化课代码，以提高编译的效率。三是在调试器的设计方面，该软件能够支持实时的调试和仿真，以便于开发人员在应用过程中快速的找到所存在的问题。四是该软件内置了工程管理功能，以便于开发人员能够从源码调试等各个不同的方面对项目进行管理，综上所述，本文所选取的 Keil5 开发人员是一道高效可靠的嵌入式开发软件，能够适用于本文的软件开发需求和仿真需求。Keil5 开发界面如图 4.1 所示。

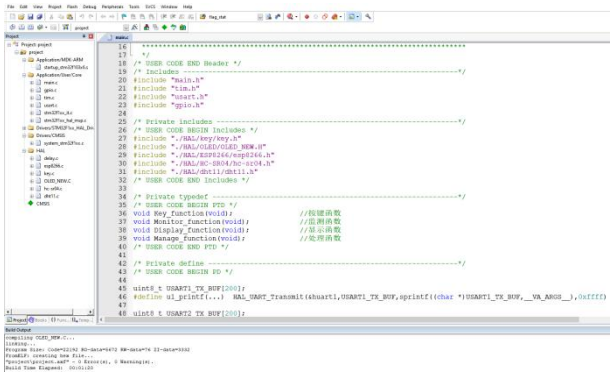


图 4.1 Keil5 开发界面

4.2 各程序设计

系统在程序设计中包含了主程序、按键控制子程序、显示控制三个主要的程序设计，本文通过流程图的方式对上述三个程序的设计方案进行介绍。

4.2.1 系统主程序设计

对于本文的主程序设计，首先是初始化各个模块。其次是利用按键扫描函数扫描按键是否被按下。如果按键被按下，根据按下的键值启动垃圾桶相应的功能，包括联网，开盖等等。同时，利用检测函检测垃圾桶内部情况，包括温度、湿度、

垃圾类别等等。并利用显示函数在 OLED 显示屏显示传感器检测的情况。如果检测到垃圾桶内部存在火焰，则通过语音播报系统将存在火焰的情况告知周围人。最后，通过无线传输模块，将检测量上传至系统后台。该系统的主程序设计流程图如图 4.2 所示。

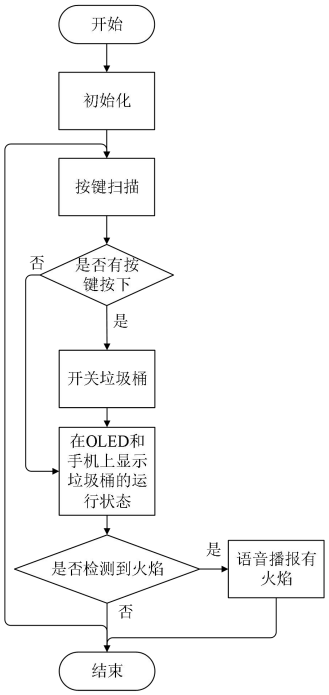


图 4.2 系统主程序设计流程图

4.2.2 系统按键控制程序设计

在设计系统按键程序时，首先判断按键是否按下以及是否接收语音指令。如果按键按下，通过扫描函数获取按下的键值。如果键值为 1 到 4，则打开对应序号的垃圾桶会或者关闭对应序号的垃圾桶。如果键值为 5，则将联网功能打开。该按键控制程序设计如图 4.3 所。

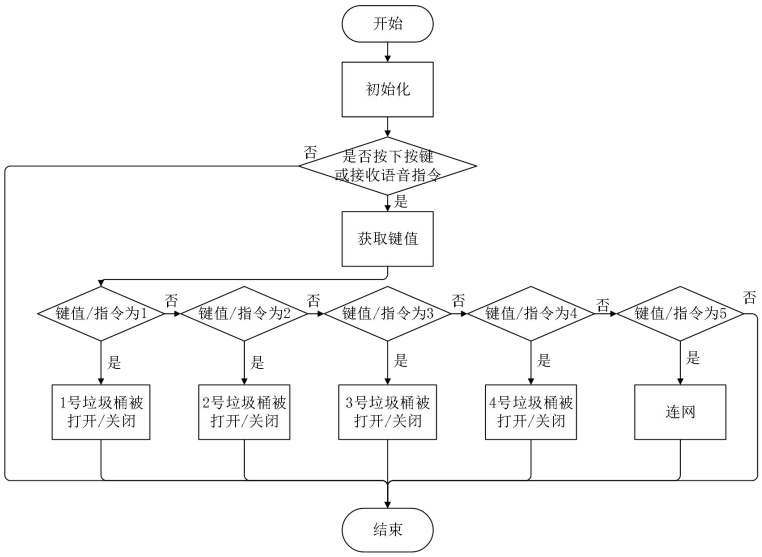


图 4.3 按键控制程序设计流程图

4.2.3 系统显示控制程序设计

系统的显示控制程序设计，首先在程序中对 OLED 显示模块进行初始化，若系统控制垃圾桶开启，则 OLED 显示模块的第三行显示对应的垃圾桶开启状态，如果没有垃圾桶开启，则继续重新刷新 4 个独立垃圾桶的垃圾余量、环境温湿度结果。该程序设计流程图如图 4.4 所示。

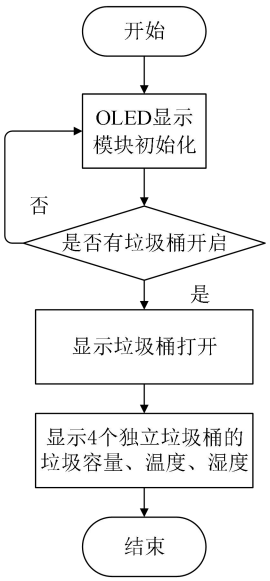


图 4.4 显示控制程序设计流程图

5 系统调试与实现

在进行系统的调试和实现过程中，主要实现了硬件调试、软件调试和软硬件联合调试三个主要部分。

5.1 硬件调试

本文所涉及的硬件调试部分组主要分为三个步骤：首先，将使用到包括电容、电阻在内的所有硬件进行仪器的测试，确保其性能良好。其次，在电路板焊接完成之后，对各功能模块按照功能设计流程图进行测试，试验其是否能满足设计要求。最后，在所有的电器元件和功能模块焊接完毕之后，对系统整体功能进行调试，确保所有部分能正常工作，以免影响后续的软件联合调试。例如，使用万用表测试电阻、电容、各开关元件是否故障，并采用相对应的档位测量其性能指标是否与标称规格相符合，是否能满足设计要求。在此次硬件调试过程中，使用万用表发现两个电容被击穿，不能起到应有的隔绝效果，更换故意电容后问题得到解决。另外在开关元器件检测过程中，若发现六角按键内部的接线问题，在按键被按下之后，不能控制整个系统的电源导通状态，则需要更换新的六角按键，在此次研究过程中，未发现此类问题。此次研究的硬件实物展示如图 5-1 所示。

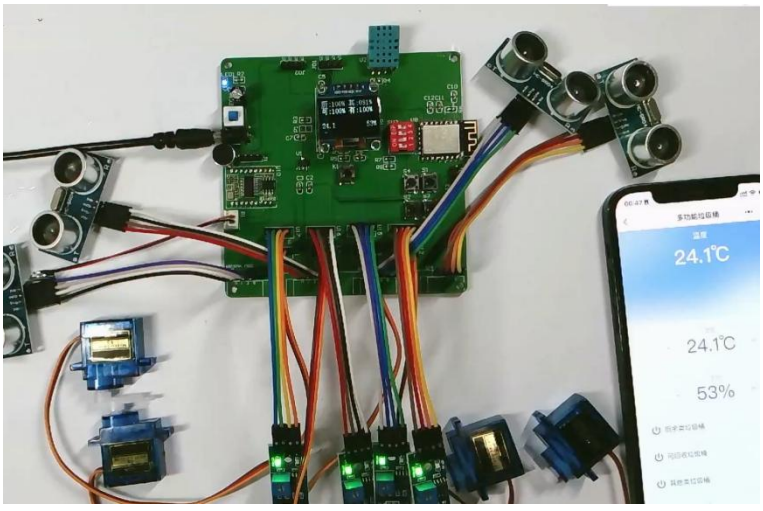


图 5.1 系统硬件实物

5.2 软件调试

软件的调试主要是依据第四章的软件设计流程图的顺序进行各自功能的代

码编写和调试，整个过程遵循从局部到整体的思想，先完成子功能的代码调试，使各子功能模块的性能满足设计要求，再进行系统整体的调试。在软件调试过程中，发现了以下几个问题，并得到了处理解决：一是循环无法直接跳出的问题，在软件调试过程中，发现部分 While 循环无法跳出，经测试发现部分代码在循环设计上存在缺陷，循环跳出的条件设置不合理，在循环过程中无法达到循环跳出条件，导致出现故障，经重新修改后该问题得到了解决。二是函数无法编译问题，经分析发现，存在部分函数名称使用错误，导致在函数库内无法查找到相应函数，以及不同代码中同一函数名称前后不对应，统一修改函数名称及重新调整函数对应关系之后，上述问题得到解决。

5.3 系统各功能模块测试

5.3.1 初始化功能模块测试

在系统初始化功能模块的测试过程中，在按下外部电源供电模块，实现供电之后，对设备各功能模块进行初始化，本文的初始化过程包含了厨余垃圾、可回收垃圾、有害垃圾和其他垃圾一共 4 个独立垃圾桶的控制程序，包含了垃圾桶的超声波检测传感器、火焰检测传感器、舵机以及系统 WiFi 模块的初始化功能，经初始化完成后，各功能模块均能够正常运行，且 OLED 显示模块能够正常显示系统对于环境温度和湿度的检测结果。在完成初始化功能模块测试后，对该系统的其他功能进行检测。该功能的测试过程如图 5.2 所示。

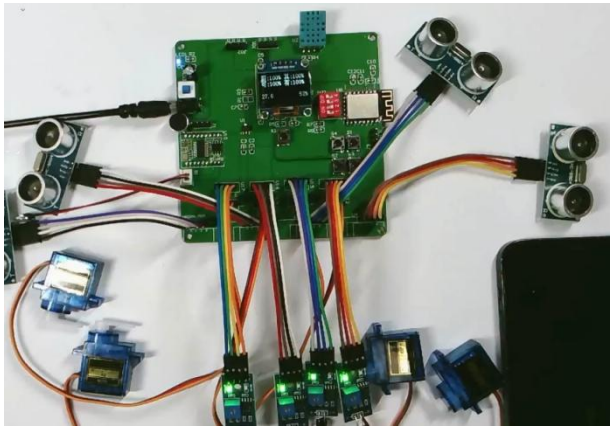


图 5.2 初始化功能测试

5.3.2 垃圾桶自动控制功能测试

垃圾桶的自动控制功能测试包含了垃圾桶的余量、火灾检测、垃圾桶盖手动

开关、远程开关等功能的测试。首先对垃圾桶的余量检测功能进行测试，因厨余垃圾、可回收垃圾、有害垃圾和其他垃圾 4 个独立垃圾桶的内部配置完全一致，因此举例说明其中一个垃圾桶的测试过程。在进行垃圾余量检测时，主要依赖于垃圾桶顶部的超声波检测传感器，检测传感器距离垃圾桶内容量，顶部的高度距离，根据该距离检测结果通过百分比的方式评估垃圾桶的余量，并将检测结果显示在 OLED 显示模块上，该功能的测试过程如图 5.3 所示。

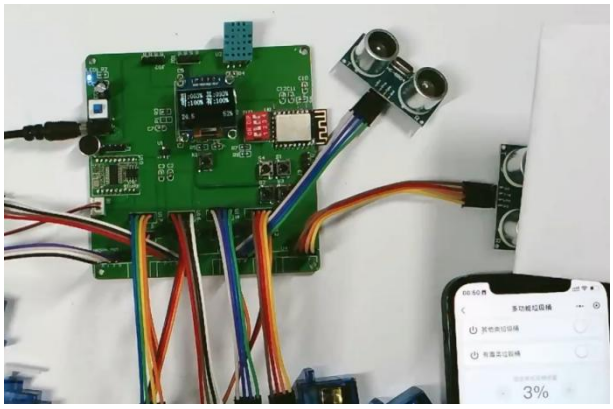


图 5.3 垃圾桶余量检测功能测试

在垃圾桶火灾检测功能测试中，主要依赖于垃圾桶内部的火焰检测传感器，该火焰检测传感器放置在垃圾桶内部，若检测到垃圾桶内部火焰，则通过该传感器向单片机的对应引脚发送高电平，实现火焰的检测功能该功能的测试过程如图 5.4 所示。

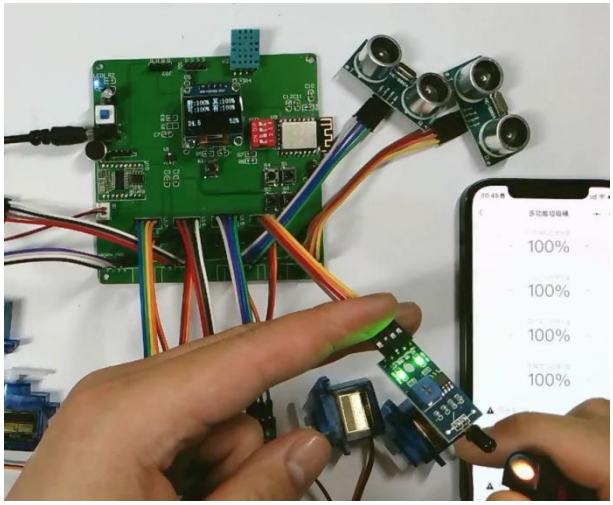


图 5.4 垃圾桶火灾检测功能测试

垃圾桶的开关功能测试包含了基于独立按键的垃圾桶开关测试以及基于 WiFi 模块的手机客户端远程开关功能测试。在测试过程中通过舵机的旋转角度模拟垃圾桶盖的开合状态，在控制过程中，垃圾桶打开一定时间后，自动反向旋

转舵机完成垃圾桶的关闭。在开关过程中，OLED 显示模块均能够显示出对应垃圾桶的开关状态。该功能的测试过程如图 5.5 所示。

图 5.5 垃圾桶盖开关功能测试

5.3.3 温湿度检测功能测试

本文的温湿度检测功能依赖于安装在系统上的 DHT11 温湿度检测传感器对环境中的温度和湿度进行检测。在检测完成后，温度和湿度数据分别显示在 oled 显示模块第 4 行的左侧和右侧，在测试过程中，通过吹气和用手按压的方式，模拟环境中温度和湿度的变化，并观察显示模块上的显示数据是否发生变化。若同样发生变化，则说明该模块能够正常检测且系统该功能运行正常，该功能的测试过程如图 5.6 所示。

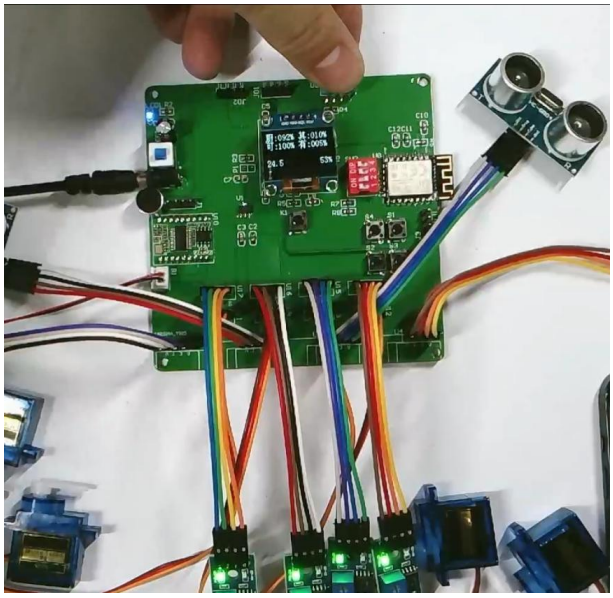


图 5.6 温湿度检测功能测试

5.3.4 语音控制模块功能测试

本文通过语音控制模块进行垃圾桶的开合语音开合操作，其中语音控制的触发指令为“小爱同学”，在测试过程中分别通过“剩菜”“中药”测试打开厨余垃圾桶、“玻璃瓶”“书籍”“塑料篮子”“报纸”测试打开可回收垃圾桶、“脏衣服”“渣土”测试打开其他垃圾桶、“酒精”“杀虫剂”测试打开有害垃圾。在上述开启关闭过程中 OLED 显示模块同步显示开关状态。该功能的测试过程如图 5.7 所示。

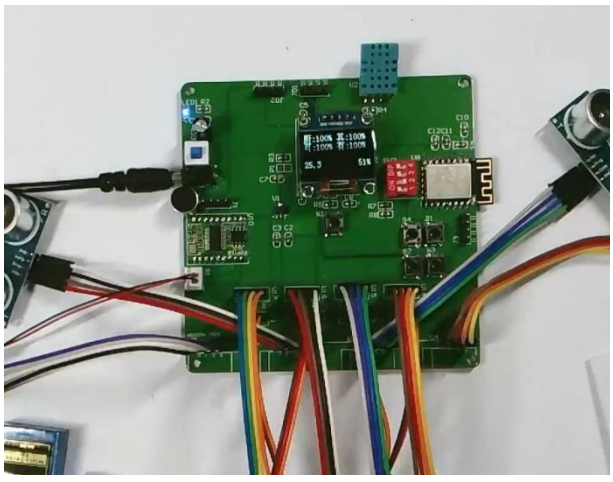


图 5.7 语音控制模块功能测试

5.3.5 WiFi 无线数据传输功能测试

WiFi 无线数据传输功能的测试过程，包含了配网和数据同步控制两个部分。其中配网功能通过 WiFi 扫码的方式实现设备的配网功能。该功能的测试过程如图 5.8 所示。

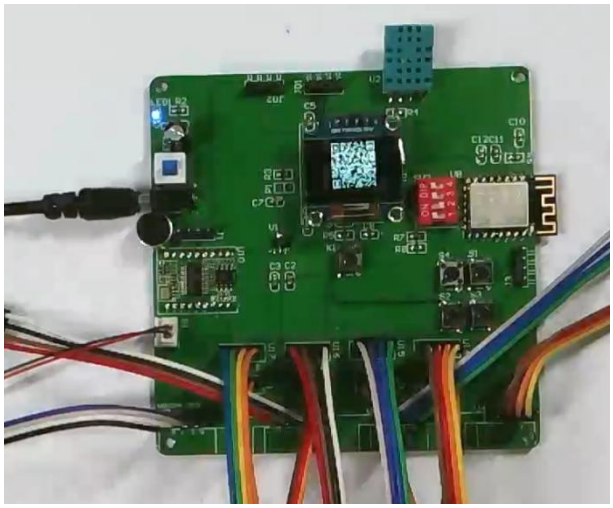


图 5.8 WiFi 配网功能测试

在配网完成后，通过手机客户端实现温湿度、4 个独立垃圾桶的开合状态以及垃圾桶内部余量等参数的检测结果。同时在控制过程中，根据 APP 的按键也可以实现垃圾桶的远程开关控制。该功能的测试过程如图 5.9 所示。

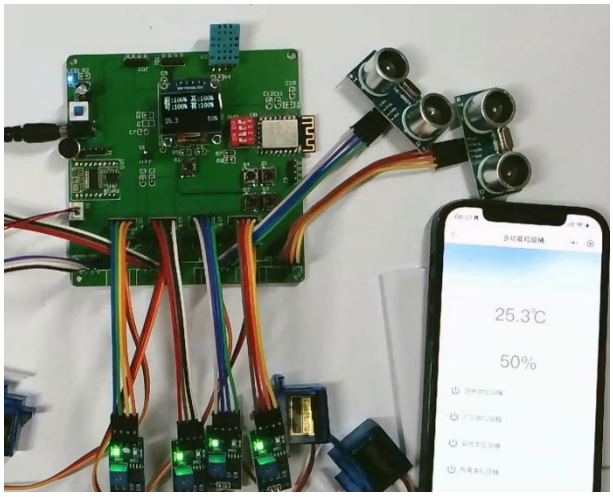


图 5.9 数据同步和控制功能测试

6 结论和展望

6.1 结论

本文所设计的基于单片机的智能垃圾桶控制系统，相较于传统的智能垃圾桶能够实现多个垃圾桶的综合管控，通过融入各种环境监测传感器实现了垃圾桶的环境和内部数据检测功能，通过语音控制模块，实现了基于语音的声控功能，通过 WiFi 模块将检测结果和手机 APP 进行同步实现基于手机 APP 的数据同步显示和控制信号的传输。在单片机上 OLED 显示模块对垃圾桶的余量、开关状态和环境温湿度进行显示，在垃圾桶内，火焰检测传感器能够实现垃圾桶内部火灾情况的检测功能。综上所述，本文所涉及的智能垃圾桶控制系统相较于传统的垃圾桶能够实现更为丰富的环境数据的检测和控制功能，极大地满足了人们对于垃圾桶的智能化控制需求，是一款较为可行且可靠性较高的解决方案。

6.2 展望

随着城市化进程的不断加速，垃圾处理问题日益凸显，传统的垃圾处理方式已经难以满足日益增长的城市人口和垃圾产量。在这一背景下，智能识别垃圾桶的设计成为了解决这一问题的有效途径。首先智能识别垃圾桶可以通过视觉识别、传感器技术等手段，准确判断垃圾的种类，帮助人们更准确地进行垃圾分类。这将提高垃圾分类的准确性，降低垃圾混合率，推动资源的有效回收和再利用。其次智能识别垃圾桶的使用可以有效减少垃圾的混合处理，降低对土壤、水源等自然环境的污染，提高城市环境质量。最后通过智能识别垃圾桶收集的数据，可以为城市管理提供宝贵的参考。可以分析垃圾产生的规律，优化垃圾桶的设置位置，提高垃圾收集的效率，减少资源浪费。总的来说，智能识别垃圾桶的设计在提高城市垃圾处理效率、保护环境、推动循环经济、促进社会参与等方面有着广泛的应用前景。随着技术的不断进步和社会的不断需求，相信智能识别垃圾桶将在未来得到更广泛的应用，为人类创造更清洁、美好的生活环境。

参考文献

[1]姚依铭,何巧红,韦延锋,曹晶,陈杰,邸国辉. 基于深度学习的语音识别垃圾桶的设计[J]. 长江信息通信,2022,35(12):78-79+82.

[2]卫宣伶. 基于计算机视觉的智能废料瓶分类系统[J]. 长江信息通信,2022,35(12):107-109.

[3]冯寿岗,刘娅楠,李圣君. 智能杀菌分类垃圾桶的设计与实现[J]. 电子制作,2022,30(13):27-30+84.

[4]丁梦思,王琰,潘可可,陈学进,李桂荣. 智能识别分类垃圾箱的认识与设计[J]. 科技创新与应用,2022,12(14):78-81.

[5]李祺,曾明,卢向哲,聂为之. 基于机器视觉的智能垃圾分类实验平台设计[J]. 实验室研究与探索,2022,41(04):68-73.

[6]毛长丹. 基于 RFID 的智能垃圾分类回收系统设计[J]. 信息系统工程,2022,(03):96-99.

[7]于雯,王艳,张佳佳,陈思思. 多功能智能垃圾桶结构设计和功能实现[J]. 工业仪表与自动化装置,2022,(01):117-120.

[8]朱效恒,禹素萍,许武军,范红. 智能垃圾设备监测系统[J]. 软件导刊,2022,21(02):149-153.

[9]左露洁,张幔,狄雨洋,肖权珈,刘冬阳,李硕. 易分宝——智慧城市自动分类垃圾桶系统设计[J]. 资源信息与工程,2022,37(01):117-120.

[10]陈志伟,李志超,刘天丽,张道信,王国松,王辰悉. 基于 STM32 的智能家居垃圾桶设计[J]. 机械,2022,49(02):67-72.

[11]张鸿,陈怡. 智慧景区智能垃圾桶管控系统的设计[J]. 黄河科技学院学报,2022,24(02):44-50.

[12]李琳,胡方圆,邹青龙,刘淑妍,邹欣. 基于 STM32 的校园内智能垃圾桶测满系统[J]. 黑龙江科学,2022,13(02):44-45.

[13]贺连升,周扬理,于新畅,牛子夫,赵一林. 基于语音识别的智能垃圾桶设计与实现[J]. 机电工程技术,2022,51(01):122-125.

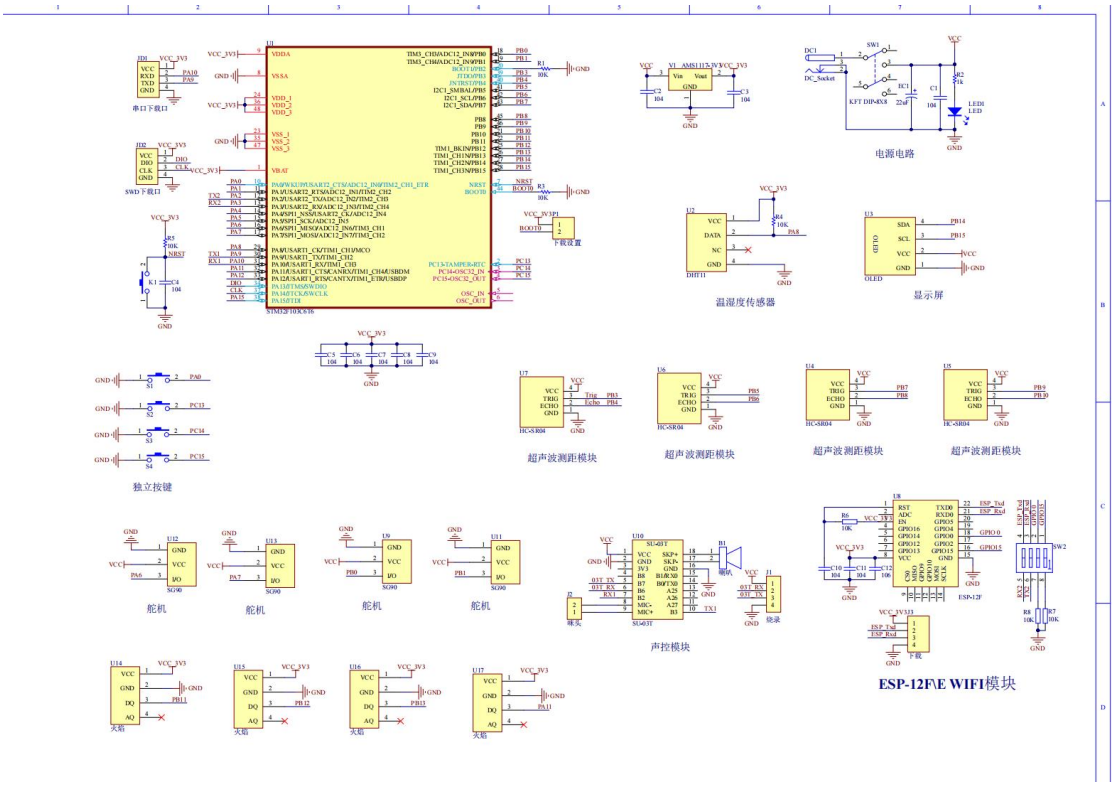
[14]吴伟烈,刘如军. 便于智能化管理的公共垃圾桶的设计与实现[J]. 物联网技

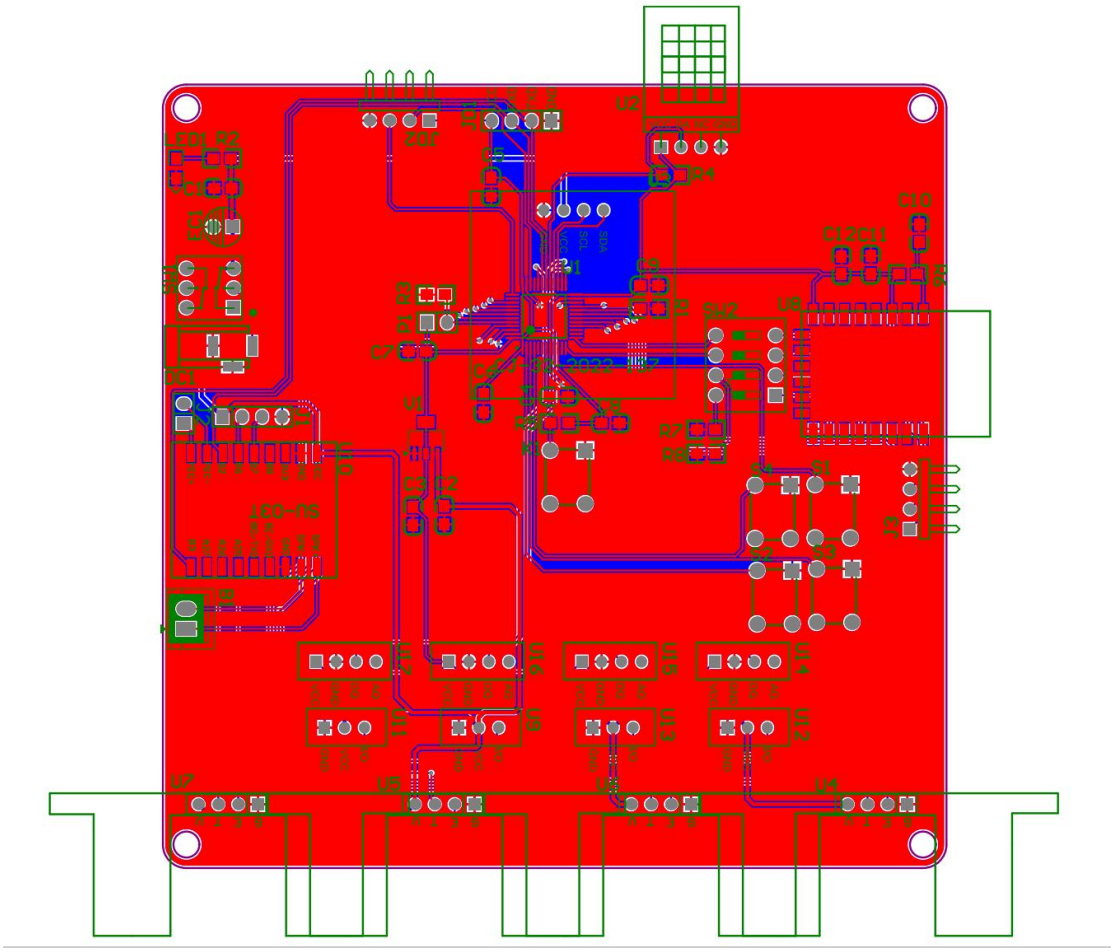
术,2022,12(01):67-69.
[15]段昆昕,江秋仪,陈锋楠,杨家来,付羿. 基于树莓派下智能分类垃圾桶设计[J]. 中国新技术新产品,2022,(01):37-39.
[16]潘仲勋,葛宇童,董京伟,牛爽,刘振晔. 基于深度学习的智能垃圾桶设计[J]. 科技与创新,2022,(01):66-68.
[17]Abuga Dominic,Raghava N.S. Real-time smart garbage bin mechanism for solid waste management in smart cities[J]. Sustainable Cities and Society,2021,75.
[18]Sen Gupta, Y.,Mukherjee, S.,Dutta, R.,Bhattacharya, S.. A blockchain-based approach using smart contracts to develop a smart waste management system[J]. International Journal of Environmental Science and Technology,2021,(prepublish).
[19]. Predictive Analysis based Efficient Routing of Smart Garbage Bins for Effective Waste Management[J]. International Journal of Recent Technology and Engineering,2019,8(3).
[20]. Development f Smart Garbage Bins for Automated Segregation of Waste with Real-Time Monitoring using Iot[J]. International Journal of Engineering and Advanced Technology,2019,8(6S).

致谢

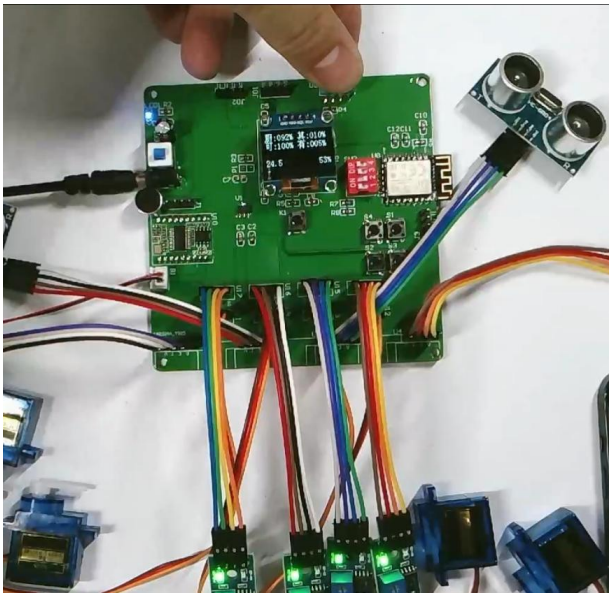
转眼间，四年的学习生涯即将结束，在毕业之际，以毕业论文的方式来结束大学四年的美好时光。我要感谢我的导师，从最初对论文的选题到后面的最终稿，老师都以最认真、最负责的态度对我进行指导，尤其是在之前的系统设计、方案制定、系统仿真以及元器件焊接过程中可谓是困难重重，多亏指导老师对我进行了无私的指导和帮助，耐心的帮助我制定整体系统方案，指导系统的仿真以及焊接，不厌其烦的帮我指出论文中的不当之处，并且热心的帮助我指导修改论文。在此，向给予我莫大帮助的老师表示最衷心的感谢！

附录一：原理图


电源电路
温湿度传感器
显示屏
超声波测距模块
超声波测距模块
超声波测距模块
超声波测距模块
独立按键
舵机
舵机
舵机
舵机
声控模块
ESP-12F\E WIFI模块



附录二：实物图



附录三：程序

```
/* USER CODE BEGIN Header */
/**
******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
******************************************************************************
  * @attention
  *
  * Copyright (c) 2022 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
```

```
  *
**********
*********
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "./HAL/key/key.h"
#include "./HAL/OLED/OLED_NEW.H"
#include "./HAL/ESP8266/esp8266.h"
#include "./HAL/HC-SR04/hc-sr04.h"
#include "./HAL/dht11/dht11.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
void Key_function(void);                    //按键函数
void Monitor_function(void);                //监测函数
void Display_function(void);                //显示函数
void Manage_function(void);                 //处理函数
/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

uint8_t USART1_TX_BUF[200];
```

```
#define u1_printf(...) HAL_UART_Transmit(&huart1,USART1_TX_BUF,sprintf((char *)USART1_TX_BUF,__VA_ARGS__),0xffff)

uint8_t USART2_TX_BUF[200];
#define u2_printf(...) HAL_UART_Transmit(&huart2,USART2_TX_BUF,sprintf((char *)USART2_TX_BUF,__VA_ARGS__),0xffff)

#define FRI1 HAL_GPIO_ReadPin(FRI1_GPIO_Port,FRI1_Pin)
#define FRI2 HAL_GPIO_ReadPin(FRI2_GPIO_Port,FRI2_Pin)
#define FRI3 HAL_GPIO_ReadPin(FRI3_GPIO_Port,FRI3_Pin)
#define FRI4 HAL_GPIO_ReadPin(FRI4_GPIO_Port,FRI4_Pin)
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
uint8_t key_num,flag_display;          //按键与显示变量
uint16_t time_1ms,time_500ms;          //计时变量 1ms,500ms

//串口 1 的数据获取
uint8_t uart1_value;                   //串口传的单个数据
//串口的储存数组，串口的接收时间，串口存值的数量
uint8_t uart1_buf[128],uart1_time,uart1_num;
uint8_t uart1_rx_flag;                 //串口的获取值的标志位
```

```
uint16_t sg90_1,sg90_2,sg90_3,sg90_4;   //四个舵机角度和垃圾桶状态
uint8_t flag_stat_1,flag_stat_2,flag_stat_3,flag_stat_4;

uint8_t flag_full_1,flag_full_2,flag_full_3,flag_full_4; //垃圾桶是否已满标志位
uint16_t distence1,distence2,distence3,distence4;                    //垃圾桶剩余空间
uint16_t autoclose_num;
                //自动关闭计时

uint8_t flag_fire_1,flag_fire_2,flag_fire_3,flag_fire_4;            //4 个着火标志位
uint8_t fire_1,fire_2,fire_3,fire_4,full_1,full_2,full_3,full_4;//着火和垃圾桶已满播报标志位

uint16_t temp,humi;
                        //温湿度变量
uint8_t flag_clearoled;
                    //清理标志位
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/****
*******按键设置函数
```

```
*****/
void Key_function(void)
{
    key_num = Chiclet_Keyboard_Scan(); //按键扫描
    if(key_num != 0)                                    //有按键按下
    {
        switch(key_num)
        {
            case 1:                                                 // 按 键
1，手动打开/关闭厨余类垃圾桶
              flag_stat_1==0?flag_stat_1=1:(flag_stat_1=0);
              break;

            case 2:                                                 // 按 键
2，手动打开/关闭可回收垃圾桶
              flag_stat_2==0?flag_stat_2=1:(flag_stat_2=0);
              break;

            case 3:                                                 // 按 键
3，手动打开/关闭其他类垃圾桶
                flag_stat_3==0?flag_stat_3=1:(flag_stat_3=0);
              break;

            case 4:                                                 // 按 键
4，手动打开/关闭有害类垃圾桶
                flag_stat_4==0?flag_stat_4=1:(flag_stat_4=0);
              break;

          case 5:                                                   //按键 5，配
网
                SmartConfig=1;
              OLED_Clear();
```

```
                break;
                default:
                break;
            }
        }
    }

    /****
    *******监测函数
    *****/
    void Monitor_function(void)
    {
        if(time_500ms == 1)                    //每 500ms 获取一次垃圾桶余量和温湿度
        {
            time_500ms = 0;
            distence4=Hcsr04_GetDistance(&htim2,25,1);
          distence2=Hcsr04_GetDistance(&htim2,25,2);
          distence1=Hcsr04_GetDistance(&htim2,25,3);
          distence3=Hcsr04_GetDistance(&htim2,25,4);
          DHT11_Read_TempAndHumidity(&DHT11_Data);//调用获取温湿度函数
            temp = DHT11_Data.temperature; //获取温度
            humi = DHT11_Data.humidity;      //获取湿度
        }
       if(FRI1==0)                                    //厨余垃圾桶检测到火焰，语音提示“厨余垃圾桶着火”
        {
          flag_fire_1=1;
          if(fire_1 == 0)
          {
            fire_1 = 1;
            u1_printf("%c%c%c%c%c",0xAA,0x55,0x05,0x55,0xAA);
          }
```

```
    }
    else
    {
      flag_fire_1=0;
      fire_1 = 0;
    }

    if(FRI2==0)                                    //可回收垃圾桶检
测到火焰，语音提示“可回收垃圾桶着火”
    {
      flag_fire_2=1;
      if(fire_2 == 0)
      {
        fire_2 = 1;
        u1_printf("%c%c%c%c%c",0xAA,0x55,0x06,0x55,0xAA);
      }
    }
    else
    {
      flag_fire_2=0;
      fire_2 = 0;
    }

    if(FRI3==0)                                    //其他类垃圾桶检
测到火焰，语音提示“其他类垃圾桶着火”
    {
      flag_fire_3=1;
      if(fire_3 == 0)
      {
        fire_3 = 1;
        u1_printf("%c%c%c%c%c",0xAA,0x55,0x07,0x55,0xAA);
      }
    }
```

```
        else
        {
          flag_fire_3=0;
          fire_3 = 0;
        }

        if(FRI4==0)                                    //有害类垃圾桶检
测到火焰，语音提示“有害类垃圾桶着火”
        {
          flag_fire_4=1;
          if(fire_4 == 0)
          {
            fire_4 = 1;
            u1_printf("%c%c%c%c%c",0xAA,0x55,0x08,0x55,0xAA);
          }
        }
        else
        {
          flag_fire_4=0;
          fire_4 = 0;
        }



        if(distence1<=10)          //垃圾桶剩余空间小于 10，语音播报其垃圾桶打
开
        {
          flag_full_1=1;
          if(full_1 == 0)
          {
            full_1 = 1;
            u1_printf("%c%c%c%c%c",0xAA,0x55,0x01,0x55,0xAA);
          }
```

```
}
else
{
   full_1 = 0;
   flag_full_1=0;
}
if(distence2<=10)
{
   flag_full_2=1;
   if(full_2 == 0)
   {
      full_2 = 1;
      u1_printf("%c%c%c%c%c",0xAA,0x55,0x02,0x55,0xAA);
   }
}
else
{
   full_2 = 0;
   flag_full_2=0;
}
if(distence3<=10)
{
   flag_full_3=1;
   if(full_3 == 0)
   {
      full_3 = 1;
      u1_printf("%c%c%c%c%c",0xAA,0x55,0x03,0x55,0xAA);
   }
}
else
{
   full_3 = 0;
   flag_full_3=0;
}
```

```
if(distence4<=10)
{
  flag_full_4=1;
  if(full_4 == 0)
  {
    full_4 = 1;
    u1_printf("%c%c%c%c%c",0xAA,0x55,0x04,0x55,0xAA);
  }
}
else
{
  full_4 = 0;
  flag_full_4=0;
}

 if(uart1_rx_flag == 1)                                    //接收到语音指令
{
  uart1_rx_flag = 0;
  if(uart1_buf[0]==0xf4&& uart1_buf[1]==0x06&& uart1_buf[3]==0xff)
  {
    switch(uart1_buf[2])
    {
      case 0x05:                                           //回复语“正在为您
打开厨余类垃圾桶”
        if(flag_full_1==0)
          flag_stat_1=1;
      break;
      case 0x06:                                           //回复语“正在为您
打开可回收垃圾桶”
        if(flag_full_2==0)
          flag_stat_2=1;
      break;
      case 0x07:                                           //回复语“正在为您
```

```
打开其他类垃圾桶”
                    if(flag_full_3==0)
                        flag_stat_3=1;
                break;
                case 0x08:                                          //回复语“正在为您
打开有害类垃圾桶”
                    if(flag_full_4==0)
                        flag_stat_4=1;
                break;
                case 0x09:                                          //配网，显示配网二
维码
                    SmartConfig=1;
                    OLED_Clear();
                break;
                default:
                    break;
            }
        }
    }

}


/****
*******显示函数
*****/
void Display_function(void)
{
   if(SmartConfig==0)                                               //不在配网
界面，显示温湿度、四类垃圾桶的剩余空间和状态
    {
      OLED_Show_Temp(0,6,temp);
      OLED_Show_Humi(96,6,humi/10);
```

```
Oled_ShowCHinese(0,0,"厨");
Oled_ShowString(16,0,":");
OLED_ShowNum(24,0,distence1,3,16);
Oled_ShowString(48,0,"%");

Oled_ShowCHinese(0,2,"可");
Oled_ShowString(16,2,":");
OLED_ShowNum(24,2,distence2,3,16);
Oled_ShowString(48,2,"%");

Oled_ShowCHinese(64,0,"其");
Oled_ShowString(80,0,":");
OLED_ShowNum(88,0,distence3,3,16);
Oled_ShowString(112,0,"%");

Oled_ShowCHinese(64,2,"有");
Oled_ShowString(80,2,":");
OLED_ShowNum(88,2,distence4,3,16);
Oled_ShowString(112,2,"%");

if(flag_stat_1==1)
{
  Oled_ShowCHinese(8,4,"厨余类垃圾已开");
}
if(flag_stat_2==1)
{
  Oled_ShowCHinese(8,4,"可回收垃圾已开");
}
if(flag_stat_3==1)
{
  Oled_ShowCHinese(8,4,"其他类垃圾已开");
}
```

```
        if(flag_stat_4==1)
        {
          Oled_ShowCHinese(8,4,"有害类垃圾已开");
        }
        if(flag_clearoled==1)
        {
          flag_clearoled=0;
          OLED_Clear();
        }
      }
      else
    //配网界面，显示配网二维码
      {
        OLED_Drwa_QRCode();
      }
    }


    /****
    *******处理函数
    *****/
    void Manage_function(void)
    {
       if(flag_stat_1==1)                          //标志位等于 1，舵机转动 180
度，模拟垃圾桶打开
      {
        sg90_1=2500;
      }
      else
      {
        sg90_1=500;
      }
      if(flag_stat_2==1)
```

```
    {
        sg90_2=2500;
    }
    else
    {
        sg90_2=500;
    }
    if(flag_stat_3==1 )
    {
        sg90_3=2500;
    }
    else
    {
        sg90_3=500;
    }
    if(flag_stat_4==1 )
    {
        sg90_4=2500;
    }
    else
    {
        sg90_4=500;
    }
}

/* USER CODE END 0 */

/**
    * @brief    The application entry point.
    * @retval int
    */
int main(void)
{
    /* USER CODE BEGIN 1 */
```

```
/* USER CODE END 1 */

/* MCU Configuration--------------------------------------------------------*/

/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM1_Init();
MX_TIM3_Init();
MX_USART1_UART_Init();
MX_USART2_UART_Init();
MX_TIM2_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim1);
HC_SR04_TIM_init(&htim2);
HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_2);
HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_3);
HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_4);
  HAL_UART_Receive_IT(&huart1, &uart1_value, 1);
```

```
    HAL_UART_Receive_IT(&huart2, &uart2_value, 1);

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    OLED_Init();                                    //OLED 初始化
    OLED_Clear();                                   //OLED 清屏

    while (1)
    {
      Key_function();                   //按键函数
        Monitor_function();             //监测函数
        Display_function();             //显示函数
        Manage_function();              //处理函数
      ESP8266_App();
      /* USER CODE END WHILE */

      /* USER CODE BEGIN 3 */
      __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,sg90_1);
//舵机 1
      __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_2,sg90_2);
//舵机 2
      __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_3,sg90_3);
//舵机 3
      __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_4,sg90_4);
//舵机 4
    }
    /* USER CODE END 3 */
  }

  /**
```

```
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL16;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
```

```
    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) !=
HAL_OK)
    {
      Error_Handler();
    }
  }

  /* USER CODE BEGIN 4 */

  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
  {
     if(htim->Instance == htim1.Instance)        //定时器 1 触发中断
     {
         time_1ms++;
         if(time_1ms>= 500)
         {
             time_1ms= 0;
             time_500ms = 1;
  //500ms 标志位
         }
         if(uart1_num != 0)
         {
             uart1_time++;
             if(uart1_time >= 10)                              //一帧数据接受
完成
             {
                 uart1_time = 0;
                 uart1_num = 0;
                 uart1_rx_flag = 1;
             }
         }
       time_flag_1ms++;
         if(time_flag_1ms>3000)
```

```
            {
                time_flag_1ms=0;
                sec_3 = 1;
    //3s 标志位
            }
        if(flag_stat_1==1|| flag_stat_2==1||flag_stat_3==1||flag_stat_4==1)   //垃圾
桶打开 5 秒后自动关闭
        {
          autoclose_num++;
          if(autoclose_num>=5*1000)
          {
            autoclose_num=0;
            flag_stat_1=0;
            flag_stat_2=0;
            flag_stat_3=0;
            flag_stat_4=0;
            flag_clearoled=1;
          }
        }
      }
    }

    void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
    {
      if(huart->Instance == huart1.Instance)//串口 1 触发中断
      {
          HAL_UART_Receive_IT(&huart1, &uart1_value, 1);
          uart1_buf[uart1_num++] = uart1_value;
          uart1_time = 0;
      }
     if(huart->Instance == huart2.Instance)//串口 2 触发
      {
          HAL_UART_Receive_IT(&huart2, &uart2_value, 1);
```

```
            if(esp8266_cnt >= sizeof(esp8266_buf))     esp8266_cnt = 0; //防止串口
被刷爆
            esp8266_buf[esp8266_cnt++] = uart2_value;
        }


    }
    /* USER CODE END 4 */

    /**
      * @brief  This function is executed in case of error occurrence.
      * @retval None
      */
    void Error_Handler(void)
    {
      /* USER CODE BEGIN Error_Handler_Debug */
      /* User can add his own implementation to report the HAL error return state */
      __disable_irq();
      while (1)
      {
      }
      /* USER CODE END Error_Handler_Debug */
    }

    #ifdef  USE_FULL_ASSERT
    /**
      * @brief  Reports the name of the source file and the source line number
      *         where the assert_param error has occurred.
      * @param  file: pointer to the source file name
      * @param  line: assert_param error line source number
      * @retval None
      */
    void assert_failed(uint8_t *file, uint32_t line)
```

```
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```