文章编号:2096-1472(2023)09-0059-04
DOI:10.19644/j.cnki.issn2096-1472.2023.009.011

基于鸿蒙 OpenHarmony 的语音识别控制系统设计与实现

王 浩

(苏州健雄职业技术学院人工智能学院,江苏 太仓 215411)
✉ 769623995@qq.com

摘 要:随着语音识别技术和国产开源操作系统的飞速发展,人与机器的交互方式也发生巨大改变,从最初的实体键到触摸屏再到语音识别,用户给设备传达命令的方式越来越人性化,其中语音识别能力是判断一个终端设备是否智能化的重要标志。文章设计一种搭载国产鸿蒙 OpenHarmony 轻量级操作系统的 Hi-12F 通信模块,通过串口通信与 ASRPRO 语音识别模块进行数据交互,采用语音识别作为人机接口的通信方式,实现智能家居领域中的设备控制功能,为基于国产 OpenHarmony 操作系统进行各种应用场景设计提供了技术参考。

关键词:OpenHarmony;语音识别;智能家居
中图分类号:TP323 **文献标志码**:A

Design and Implementation of Speech Recognition Control System Based on Hongmeng OpenHarmony

WANG Hao

(*Artificial Intelligence College, Suzhou Chien-Shiung Institute of Technology, Taicang* 215411, *China*)
✉ 769623995@qq.com

Abstract: With the rapid development of speech recognition technology and domestic open source operating system, the way people interact with machines has also changed dramatically. From the initial physical key to touch screens, and then to voice recognition, the way users communicate commands to devices is becoming more and more humanized, among which speech recognition ability is an important indicator to judge whether a terminal device is intelligent. This paper proposes to design a Hi-12F communication module equipped with a domestic Hongmeng OpenHarmony lightweight operating system, which interacts with the ASRPRO speech recognition module through serial communication. The communication method uses speech recognition as a human-machine interface to achieve device control function in the smart home field, providing technical reference for designing various application scenarios based on domestic OpenHarmony operating system.

Key words: OpenHarmony; speech recognition; smart home

0 引言(Introduction)

随着鸿蒙 OpenHarmony 开源操作系统和智能语音技术的飞速发展,基于鸿蒙 OpenHarmony 操作系统的智能家居产业对家庭内部智能化控制开发和应用不断加大力度。OpenHarmony 采用开源的方式实现一个面向全场景、全连接和全智能的终端设备操作系统的框架和平台,可以支持各类设备的系统。为了能适应各种硬件,OpenHarmony 提供了如 LiteOS、Linux 的不同内核,并基于这些内核形成不同的系统类型,其中包括面向

收稿日期:2023-01-05
基金项目:2022 年度苏州健雄职业技术学院科研育人项目(JXZH202203)

MCU类处理器的轻量级操作系统，主要采用LiteOS-M内核，它是面向IoT领域构建的轻量级物联网操作系统内核，具有小体积、低功耗、高性能的特点[1]。这种轻量级操作系统的代码结构简单，主要包括内核最小功能集、内核抽象层、可选组件以及工程目录等，同时在这些系统中构建了一套统一的系统能力[2]。本文提出一个基于Hi3861V100核心处理器的Hi-12F模块，搭载鸿蒙OpenHarmony轻量级操作系统，通过串口方式连接低功耗高性能的ASRPRO语音识别模组，快速实现语音交互及控制方案。通过将语音识别技术应用于智能家居系统中，能极大地方便用户对家中电器设备的控制，创造更加舒适的家庭生活。

1 总体设计(Overall design)

本系统主要由Hi-12F通信模组、ASRPRO语音识别模组、TTL转USB串口模块、风扇控制模块、MIC麦克风模块和扬声器模块组成，其中Hi-12F通信模组和ASRPRO语音识别模组通过串口通信完成数据交互和数据处理。首先，ASRPRO语音识别模组通过麦克风模块采集用户发出的语音控制风扇设备命令，在对声音进行分析后能从“命令列表”匹配出最接近的命令，通过扬声器模块提示用户已识别语音命令，接着进行语音识别转换处理，将每条语音指令转换成对应的文本字符串，以此达到语音识别的功能。其次，通过串口将风扇控制命令的字符串发送给Hi-12F通信模组，Hi-12F通信模组搭载鸿蒙OpenHarmony轻量级操作系统之后，通过应用层串口通信函数接收控制风扇命令字符串，对连接Hi-12F通信模组IO引脚的风扇设备进行控制，实现对风扇的开启或者关闭功能，系统的整体架构如图1所示。

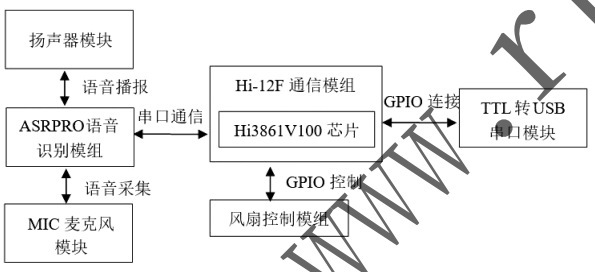

图1 系统整体架构图

Fig. 1 Overall architecture of the system

2 系统的硬件设计(Hardware design of the system)

2.1 鸿蒙OpenHarmony硬件设计

对于能搭载和支持鸿蒙OpenHarmony轻量级操作系统的硬件电路主要是面向MCU类处理器，例如Arm Cortex-M、RISC-V 32位的设备，其硬件资源极其有限，支持的设备最小内存为128 kB，并可以提供多种轻量级网络协议、轻量级的图形框架以及丰富的IOT总线读写部件等[3]。本文采用Hi-12F通信模组，该模组已获得了鸿蒙HarmonyOS Connect的生态模组技术认证，可以快速地将智能产品接入华为鸿蒙生态，以便帮助开发者提供开放、易用的开发和调试运行环境。

Hi-12F模块搭载Hi3861V100核心处理器芯片，该芯片是一款高度集成的2.4 GHz低功耗SoC WiFi芯片，集成IEEE 802.11b/g/n基带和射频(Radio Frequency，RF)电路。Hi-12F模块的Hi3861V100芯片同时集成高性能32 bit微处理器、硬件安全引擎以及丰富的外设接口，外设接口包括SPI、UART、I2C、PWM、GPIO和多路ADC芯片，同时支持高速SDIO2.0 Slave接口，最高时钟可达50 MHz[4]；芯片内置静态随机存取存储器(Static Random-Access Memory，SRAM)和闪存(Flash Memory)，可独立运行，并支持在Flash Memory上运行程序[5]。IO03引脚和IO04引脚分别连接到USB转串口模块(CH340C模块)的RX引脚和TX引脚，一旦按下PWR引脚上的按键，就可以将PC端的OpenHarmony轻量级操作系统编译完成的bin文件通过D+引脚和D−引脚传输到CH340C模块，并最终通过TX串口和RX串口方式下载至Hi3861模块中的Hi3861V100芯片的Flash Memory中。

2.2 风扇控制设计

为了能够通过OpenHarmony硬件电路驱动大功率的风扇设备运行，需要将OpenHarmony硬件电路模块的GPIO2引脚与直流电机L9110S驱动芯片相连，L9110S芯片内部集成“H”桥电路，是一个可以直接驱动直流有刷电机的芯片，VCC电源电压工作在2.2～6.5 V，输出的电流最大支持200 mA。本文只需要将风扇的两端连接芯片的OA引脚和OB引脚，它的两个输出端能直接驱动直流风扇电机的运行和停止。

2.3 语音识别采集模块设计

语音识别采集模块选用ASRPRO语音识别芯片，它是针对低成本离线语音应用方案开发的一款通用、便携、低功耗高性能的语音识别模组，其内置神经网络处理器，能支持DNN、TDNN及RNN等神经网络及卷积运算，具备强劲的回声消除和环境噪声抑制能力，模块主芯片支持离线神经网络计算，ASRPRO语音识别模组的MIC+引脚和MIC−引脚连接麦克风模块进行语音数据采集，并将语音数据经过内置神经网络处理器进行卷积运算，实现语音识别转换为文本字符串，ASRPRO语音识别模组的PA6和PA5分别连接OpenHarmony硬件电路中的IO06引脚和IO05引脚，实现将文本字符串通过串口发送至Hi-12F模块，并最终完成对风扇的控制功能。

3 系统的软件设计(Software design of the system)

3.1 OpenHarmony轻量级操作系统应用开发环境搭建

首先，系统采用虚拟机环境下基于Linux环境的Ubuntu操作系统作为编译服务器，而应用层的功能代码编写需要在Windows环境下的VSCode编辑器中进行，因此需要将获取的鸿蒙OpenHarmony轻量级操作系统源码通过共享文件夹方式映射到Windows平台上进行编写[6]。其次，将VSCode编辑器中编辑完成的功能代码映射回虚拟机环境下的Ubuntu进行编译，编译成功之后会生成.bin的二进制文件。最后，使用华为海思Hi3861芯片的专用烧写工具HiBurn，通过串口方式烧写至Hi-12F模块中，OpenHarmony轻量级操作系统应用开发流程如图2所示。一旦OpenHarmony轻量级操作系统镜像烧写完成后，按下Hi-12F模块的复位键就可以启动OpenHarmony轻量级操作系统运行，并可以通过PC端串口调试助手与OpenHarmony硬件电路进行串口通信。

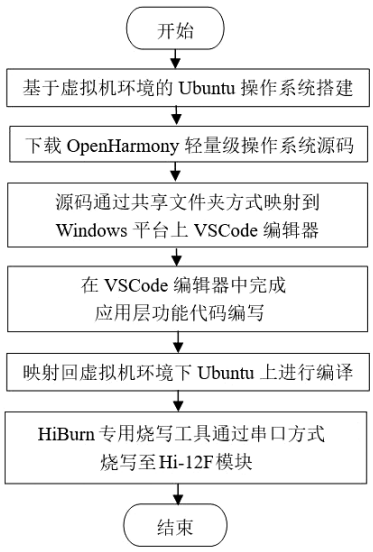

图 2　OpenHarmony 轻量级操作系统应用开发流程图

Fig. 2 The application development flowchart of OpenHarmony lightweight operating system

3.2　基于 OpenHarmony 语音识别控制应用程序设计

本设计是通过创建任务的方式完成业务逻辑功能代码，任务入口函数的作用就是把任务函数创建成一个任务，并使之进入就绪状态，接受操作系统的调度，主要包括两个步骤：设置任务属性和创建任务[7]。本文创建一个通过接收串口数据的主任务，具体代码如下：

```
static void UART_ExampleEntry(void)
{
    osThreadAttr_t attr;
    attr.name = "UART_Task";//任务名称
    attr.attr_bits = 0U;
    attr.cb_mem = NULL;
    attr.cb_size = 0U;
    attr.stack_mem = NULL;
    attr.stack_size = UART_TASK_STACK_SIZE; //任务堆栈的大小
    attr.priority = UART_TASK_PRIO; //任务优先级
    //创建任务，失败则报错
    if (osThreadNew((osThreadFunc_t)UART_Task, NULL, &attr) == NULL)
    {printf("[VoiceExample] Falied to create UART_Task! \n");}
}
    APP_FEATURE_INIT(UART_ExampleEntry);
```

APP_FEATURE_INIT 是一个在头文件 ohos_init.h 中定义的带参数的宏，这个宏的作用就是指示编译器把这里的 UART_ExampleEntry 函数编译到某个代码段中，系统启动之后，会遍历这个代码段中的所有函数并执行[8]。

当串口接收任务创建完成并开始启动任务的执行，包括 UartInit()函数对串口 1 进行配置、设置控制风扇 GPIO02 引脚的功能模式为输出模式以及进入循环体内开始调用 UartRead()函数将语音识别的字符串数据进行读取，当串口接收到“ON”字符串数据时，将 GPIO02 引脚设置为高电平，实现对风扇设备的开启，如果串口接收到“OFF” 字符串数据时，完成对风扇设备的关闭操作，具体代码如下：

```
static void UART_Task(void)
{
    uint8_t uart_buff[UART_BUFF_SIZE] = {0};
    uint8_t * uart_buff_ptr = uart_buff;
    uint32_t ret;
    WifiIotUartAttribute uart_attr = {
        .baudRate = 9600,
        .dataBits = 8,
        .stopBits = 1,
        .parity = 0,
    };
    ret = UartInit(WIFI_IOT_UART_IDX_1, &uart_attr, NULL);
    IoSetFunc(WIFI_IOT_IO_NAME_GPIO_2, WIFI_IOT_IO_FUNC_GPIO_2_GPIO);//设置 GPIO_2 的复用功能为普通 GPIO
    GpioSetDir(WIFI_IOT_IO_NAME_GPIO_2, WIFI_IOT_GPIO_DIR_OUT);//设置 GPIO_2 为输出模式
    while (1)
    {
        //通过串口 1 接收数据
        UartRead(WIFI_IOT_UART_IDX_1, uart_buff_ptr, UART_BUFF_SIZE);
        if (uart_buff_ptr[0]=='O'&&uart_buff_ptr[1]=='N')
        {
        GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_2, 1);//设置 GPIO_2 输出高电平打开风扇
        }
        if (uart_buff_ptr[0]=='O'&&uart_buff_ptr[1]=='F'&&uart_buff_ptr[2]=='F')
        {
      GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_2, 0);//设置 GPIO_2 输出低电平关闭风扇
        }
        printf("Uart1 read data:%s", uart_buff_ptr);
        usleep(1000000);//延时 1s
    }
}
```

3.3　基于 OpenHarmony 语音识别控制程序配置与编译

3.3.1　编写编译脚本

一旦编写完成语音识别控制程序功能代码后，还需要通过编译脚本通知鸿蒙 OpenHarmony 工程的编译构建子系统处理编写的源程序，这个编译脚本的文件名必须是 BUILD.gn，这是由编译构建子系统决定的[9]。BUILD.gn 文件就相当于 Makefile，但它的编译速度要比 Makefile 快。BUILD.gn 文件在 C 文件同级目录下，BUILD.gn 文件由三个部分内容（目标、源文件、头文件路径）构成，BUILD.gn 文件主要是描述了软件包的相关信息，包括编译哪些源文件、头文件路径、编译方式。这个 BUILD.gn 文件中的 static_library 是告诉编译构建子系统，将应用程序编译生成一个静态库，主要内容包括以下三个部分[10]。

（1）在 static_library 后面的括号中，指定要编译生成的静

态库的目标名称。

(2)在 sources 列表中,列出编译生成静态库所需要用到的源文件,多个文件之间用逗号隔开。

(3)在 include_dirs 列表中,列出源文件中所引用的头文件的路径,多个路径之间用逗号隔开。下面是 BUILD.gn 文件内容:

```
static_library("uart_example") {
    sources = [
        "uart_example.c"
    ]
    include_dirs = [
        "//utils/native/lite/include",
        "//kernel/liteos_m/components/cmsis/2.0",
        "//base/iot_hardware/interfaces/kits/wifiiot_lite",
    ]
}
```

3.3.2 修改 app 文件夹下的 BUILD.gn 文件

在 BUILD.gn 文件的 features 列表中,添加一项程序,代码如下:

```
import("//build/lite/config/component/lite_component.gni")
lite_component("app") {
    features = [
        #"B6_YUYIN_uart:uart_example",
    ]
}
```

这里的鸿蒙 OpenHarmony 轻量级操作系统是按照"系统(子系统集)—子系统(Subsystem)—组件/模块(Component)"逐次展开的,其中一个组件/模块(Component)也可以进一步分成若干个 Feature[11]。本文中,由源程序 uart_example.c 生成的静态库 B6_YUYIN_uart 就被作为一个 feature 加入名为 app 的 Component 中,app 的 Component 又是在子系统 applications 中。BUILD.gn 文件内容帮助编译构建子系统对一个模块的构建[12]。

4 系统测试(System testing)

当系统软硬件设计完成后,首先将硬件模块组装集成后放置在家居环境中,给系统上电,这时可以通过 ASRPRO 语音识别模组连接的 MIC 麦克风进行语音数据采集;其次通过语音识别转换成对应的控制风扇的字符串命令数据,以串口方式将数据传输至包含 OpenHarmony 轻量级操作系统的 Hi-12F 模块中;最后通过 GPIO02 引脚完成高低电平的控制,实现对风扇的转动和停止操作。

5 结论(Conclusion)

本文设计了一种基于鸿蒙 Openharmony 的语音识别风扇控制系统,该系统以 Hi-12F 模块搭载华为海思 Hi3861V100 核心处理器芯片作为核心设计,通过构建和编程 OpenHarmony 语音识别控制工程中应用层功能代码以及编写 BUILD.gn 文件,指示编译构建子系统构建一个 Feature,以及将 Feature 加入 Component 中构建一个新的 Component[13]。把 Component 加入 Subsystem 中,让 Subsystem 参与整个鸿蒙系统的构建,实现了与 ASRPRO 语音识别模组的数据交互,达到了精准的语音识别控制功能,发挥了鸿蒙国产操作系统的技术特点和优势,解决了目前智能家居中普遍存在的安全性和效率低及舒适性差的问题,为用户带来全新的智能家居控制新体验。

参考文献(References)

[1] 吴祖顺. 基于 OpenHarmony 的金融终端产品标准化现状研究与思考[J]. 金融科技时代,2023,31(5):65-70.

[2] 陈元亮,戴洋毅,瑚琦. 基于 OpenHarmony 的新型智能电动吸痰器[J]. 软件导刊,2023,22(5):78-83.

[3] 马建为,李江荣,田原. 基于 OpenHarmony 的环境参数监测系统设计[J]. 延安大学学报(自然科学版),2022,41(2):36-41,51.

[4] 孟斌. 一种 OpenHarmony 操作系统的 UI 界面自动化测试方法[J]. 电脑知识与技术,2022,18(13):128-131.

[5] 欧建深. 开源开放地构建 OpenHarmony[J]. 软件和集成电路,2021(6):28-29.

[6] 郭德财,彭石林. 基于海思芯片与鸿蒙操作系统的智慧灌溉系统设计[J]. 单片机与嵌入式系统应用,2022,22(3):11-15.

[7] 李苏,何巍. 鸿蒙 Hi3861 IoT WiFi 模组的智能家居设计[J]. 单片机与嵌入式系统应用,2022,22(12):80-82,87.

[8] 沈周锋,郑慧珍. 基于鸿蒙系统的农业监测器的研究[J]. 九江学院学报(自然科学版),2022,37(1):59-62.

[9] 张洋,刘建粉. 基于鸿蒙 Hi3861 和 STM32 双控的智能餐厅系统设计[J]. 物联网技术,2023,13(1):85-88.

[10] 程永红,王萌. 基于语音识别处理的植保无人机航行路径控制研究[J]. 农机化研究,2024,46(2):44-48.

[11] GAO R P,XING W W,ZHAO H Y,et al. Teaching reform for harmony os mobile application development[J]. Computer Education,2021(12):62-67.

[12] DU N,SCHMIDT H,POLIAN I. Low-power emerging memristive designs towards secure hardware systems for applications in Internet of Things[J]. Nano Materials Science,2021,3(2):186-204.

[13] HU X F,SHI W Q,ZHOU Y,et al. Quantized and adaptive memristor based CNN(QA-mCNN) for image processing[J]. Science China Information Sciences,2022,65(1):273-275.

作者简介:

王 浩(1971-),男,硕士,副教授。研究领域:物联网工程应用研究。