ESP32-P4

ESP-SR 用户手册

Release master
乐鑫信息科技
2025 年 09 月 01 日

Table of contents

Table of contents i

1 入门指南 3
- 1.1 概述 . . . 3
- 1.2 准备工作 . . . 3
  - 1.2.1 必备硬件 . . . 3
  - 1.2.2 必备软件 . . . 3
- 1.3 编译运行一个示例 . . . 3

2 AFE 声学前端 5
- 2.1 AFE 声学前端算法框架 . . . 5
  - 2.1.1 概述 . . . 5
  - 2.1.2 使用场景 . . . 5
  - 2.1.3 输入格式定义 . . . 5
  - 2.1.4 使用 AFE 框架 . . . 7
  - 2.1.5 资源占用 . . . 8
- 2.2 乐鑫麦克风设计指南 . . . 8
  - 2.2.1 麦克风电器性能推荐 . . . 8
  - 2.2.2 麦克风结构设计建议 . . . 8
  - 2.2.3 麦克阵列设计建议 . . . 9
  - 2.2.4 麦克风结构密封性建议 . . . 9
  - 2.2.5 回声参考信号设计建议 . . . 9
  - 2.2.6 麦克风阵列一致性验证 . . . 9
- 2.3 从 V1.* 迁移到 V2.* . . . 9
  - 2.3.1 输入数据格式修改 . . . 9
  - 2.3.2 配置和初始化 . . . 10

3 唤醒词 11
- 3.1 WakeNet 唤醒词模型 . . . 11
  - 3.1.1 概述 . . . 11
  - 3.1.2 WakeNet 的使用 . . . 12
  - 3.1.3 资源消耗 . . . 13
- 3.2 乐鑫语音唤醒方案客户定制流程 . . . 13
  - 3.2.1 唤醒词定制服务 . . . 13
  - 3.2.2 硬件设计与测试服务 . . . 14

4 语音活动检测模型 15
- 4.1 概述 . . . 15
- 4.2 使用 VADNet . . . 15
- 4.3 资源占用 . . . 16

5 命令词 17
- 5.1 MultiNet 命令词识别模型 . . . 17
- 5.2 命令词识别原理 . . . 17
- 5.3 命令词格式要求 . . . 18
- 5.4 自定义命令词方法 . . . 18
  - 5.4.1 MultiNet6 定义方法： . . . 18

5.4.2 MultiNet5 定义方法： . . . 18
5.4.3 通过调用 API 修改 . . . 19
5.5 MultiNet 的使用 . . . 20
5.5.1 MultiNet 初始化 . . . 20
5.5.2 MultiNet 运行 . . . 21
5.5.3 MultiNet 识别结果 . . . 21
5.6 资源消耗 . . . 21

6 TTS 语音合成模型 23
6.1 简介 . . . 23
6.2 简单示例 . . . 23
6.3 编程指南 . . . 24
6.4 资源消耗 . . . 24

7 模型选择和加载 25
7.1 模型选择 . . . 25
7.2 更新分区表 . . . 25
7.3 模型加载 . . . 25
7.3.1 ESP-IDF 框架 . . . 25
7.3.2 Arduino 框架 . . . 26

8 性能测试结果 27
8.1 AFE . . . 27
8.1.1 资源消耗 . . . 27
8.2 WakeNet . . . 28
8.2.1 资源消耗 . . . 28
8.2.2 性能测试 . . . 28
8.3 MultiNet . . . 28
8.3.1 资源消耗 . . . 28
8.3.2 Word Error Rate 性能测试 . . . 28
8.3.3 Speech Commands 性能测试（空调控制场景） . . . 28
8.4 TTS . . . 29
8.4.1 资源消耗 . . . 29
8.4.2 性能测试 . . . 29
8.5 NSNET . . . 29
8.5.1 性能测试 . . . 29
8.5.2 数据集：array_onemic_nnoise_20230608(按照亚马逊声学认证标准录制测试集) . . . 29

9 测试方法与测试报告 31
9.1 测试场景要求 . . . 31
9.2 测试案例设计 . . . 31
9.3 乐鑫测试与结果 . . . 32
9.3.1 唤醒率测试 . . . 33
9.3.2 语音识别率测试 . . . 33
9.3.3 误唤醒率测试 . . . 33
9.3.4 唤醒打断率测试 . . . 33
9.3.5 响应时间测试 . . . 34

10 术语表 35
10.1 通用术语 . . . 35
10.2 特别术语 . . . 35

本文档仅包含针对 ESP32-P4 芯片的 ESP-SR 使用。

Chapter 1

入门指南

乐鑫 ESP-SR 可以帮助用户基于 ESP32 系列芯片或 ESP32-S3 系列芯片，搭建 AI 语音解决方案。本文档将通过一些简单的示例，展示如何使用 ESP-SR 中的算法和模型。

1.1 概述

ESP-SR 支持以下模块：

- *声学前端算法 AFE*
- *唤醒词检测 WakeNet*
- *命令词识别 MultiNet*
- *语音合成（目前只支持中文）*

1.2 准备工作

1.2.1 必备硬件

- USB 2.0 数据线（标准 A 型转 Micro-B 型）
- 电脑（Linux）

备注： 目前一些开发板使用的是 USB Type C 接口。请确保使用合适的数据线连接开发板！

1.2.2 必备软件

- 下载 ESP-SKAINET，ESP-SR 将作为 ESP-SKAINET 的组件被一起下载。
- 配置安装 ESP-IDF，推荐使用 ESP-SKAINET 中包含的版本。安装方法请参考 ESP-IDF 编程指南 中的 快速入门 小节。

1.3 编译运行一个示例

- 进入 ESP-SKAINET/examples/cn_speech_commands_recognition 目录。
- 参考该示例目录下的配置和编译说明，运行该示例。

- 该示例为中文命令指令识别示例，通过说唤醒词（Hi, 乐鑫），触发语音指令识别。注意，当一段时间没有语音指令后，语音指令识别功能将关闭，并等待下一次唤醒词触发。

Chapter 2

AFE 声学前端

2.1 AFE 声学前端算法框架

2.1.1 概述

智能语音设备需要在远场噪声环境中，仍具备出色的语音交互性能，声学前端 (Audio Front-End, AFE) 算法在构建此类语音用户界面 (Voice-User Interface, VUI) 时至关重要。乐鑫 AI 实验室自主研发了一套乐鑫 AFE 算法框架，可基于功能强大的 ESP32-P4 系列芯片进行声学前端处理，使用户获得高质量且稳定的音频数据，从而构建性能卓越且高性价比的智能语音产品。

名称	简介
AEC (Acoustic Echo Cancellation)	回声消除算法，最多支持双麦处理，能够有效的去除 mic 输入信号中的自身播放声音，从而可以在自身播放音乐的情况下很好的完成语音识别。
NS (Noise Suppression)	噪声抑制算法，支持单通道处理，能够对单通道音频中的非人声噪声进行抑制，尤其针对稳态噪声，具有很好的抑制效果。
BSS (Blind Source Separation)	盲信号分离算法，支持双通道处理，能够很好的将目标声源和其余干扰音进行盲源分离，从而提取出有用音频信号，保证了后级语音的质量。
MISO (Multi Input Single Output)	多输入单输出算法，支持双通道输入，单通道输出。用于在双麦场景，没有唤醒使能的情况下，选择信噪比高的一路音频输出。
VAD (Voice Activity Detection)	语音活动检测算法，支持实时输出当前帧的语音活动状态。
AGC (Automatic Gain Control)	自动增益控制算法，可以动态调整输出音频的幅值，当弱信号输入时，放大输出幅度；当输入信号达到一定强度时，压缩输出幅度。
WakeNet	基于神经网络的唤醒词模型，专为低功耗嵌入式 MCU 设计

2.1.2 使用场景

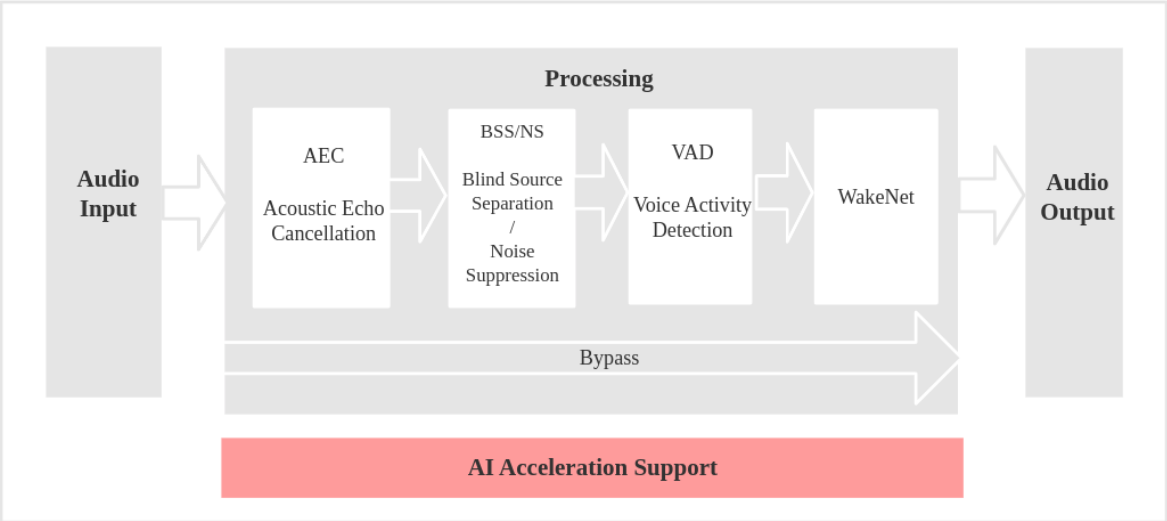
本节介绍 Espressif AFE 框架的两种典型应用场景。

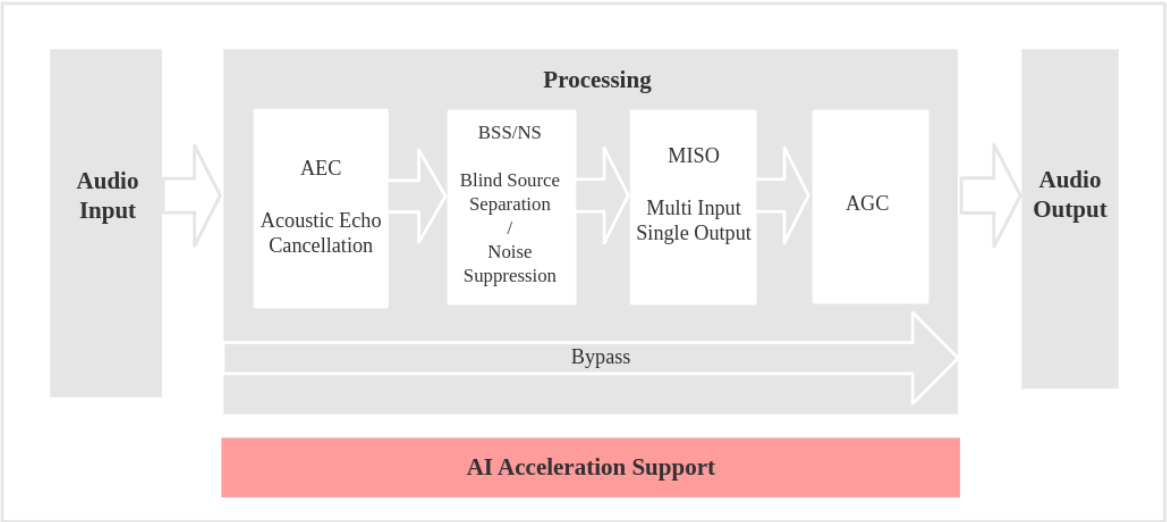
语音识别

语音通话

2.1.3 输入格式定义

`input_format` 参数定义了输入数据中音频通道的排列方式。字符串中的每个字符代表一个通道类型：


Audio Input
Processing
AEC
Acoustic Echo Cancellation
BSS/NS
Blind Source Separation / Noise Suppression
VAD
Voice Activity Detection
WakeNet
Audio Output
Bypass
AI Acceleration Support


Audio Input
Processing
AEC
Acoustic Echo Cancellation
BSS/NS
Blind Source Separation / Noise Suppression
MISO
Multi Input Single Output
AGC
Audio Output
Bypass
AI Acceleration Support

字符	描述
M	麦克风通道
R	播放参考通道
N	未使用或未知通道

示例： "MMNR"：表示四通道排列，依次为麦克风通道、麦克风通道、未使用通道和播放参考通道。

备注： 输入数据必须采用 **通道交错排列格式**。

2.1.4 使用 AFE 框架

根据 menuconfig->ESP Speech Recognition 选择需要的 AFE 的模型，比如 WakeNet 模型，VAD 模型，NS 模型等，然后在代码中使用以下步骤调用 AFE 框架。代码可以参考 test_apps/esp-sr/main/test_afe.cpp 或是 or esp-skainet/examples.。

步骤 1：初始化 AFE 配置

使用 afe_config_init() 获取默认配置并根据需求调整参数：

```
srmodel_list_t *models = esp_srmodel_init("model");
afe_config_t *afe_config = afe_config_init("MMNR", models, AFE_TYPE_SR, AFE_MODE_
↪HIGH_PERF);
```

- input_format：定义通道排列（如 MMNR）。
- models：模型列表（如 NS、VAD 或 WakeNet 模型）。
- afe_type：AFE 类型（如 AFE_TYPE_SR 表示语音识别场景）。
- afe_mode：性能模式（如 AFE_MODE_HIGH_PERF 表示高性能模式）。

步骤 2：创建 AFE 实例

通过配置创建 AFE 实例：

```
// 获取句柄
esp_afe_sr_iface_t *afe_handle = esp_afe_handle_from_config(afe_config);
// 创建实例
esp_afe_sr_data_t *afe_data = afe_handle->create_from_config(afe_config);
```

步骤 3：输入音频数据

将音频数据输入 AFE 进行处理。输入数据格式需与 input_format 匹配：

```
int feed_chunksize = afe_handle->get_feed_chunksize(afe_data);
int feed_nch = afe_handle->get_feed_channel_num(afe_data);
int16_t *feed_buff = (int16_t *) malloc(feed_chunksize * feed_nch * sizeof(int16_
↪t));
afe_handle->feed(afe_data, feed_buff);
```

- feed_chunksize：每帧输入的样本数。
- feed_nch：输入数据的通道数。
- feed_buff：通道交错的音频数据（16 位有符号，16 kHz）。

步骤 4：获取处理结果

获取处理后的单通道音频输出和检测状态：

```
afe_fetch_result_t *result = afe_handle->fetch(afe_data);
int16_t *processed_audio = result->data;
vad_state_t vad_state = result->vad_state;
wakenet_state_t wakeup_state = result->wakeup_state;

// if vad cache is exists, please attach the cache to the front of processed_audio␣
↪to avoid data loss
if (result->vad_cache_size > 0) {
    int16_t *vad_cache = result->vad_cache;
}
```

```
// get the processed audio with specified delay, default delay is 2000 ms
afe_fetch_result_t *result = afe_handle->fetch_with_delay(afe_data, 100 / portTICK_
↪PERIOD_MS);
```

2.1.5 资源占用

关于 AFE 的资源占用情况，请参阅资源占用。

2.2 乐鑫麦克风设计指南

本指南基于乐鑫的 ESP32-P4 系列语音开发板，对于整机 mic 设计做出要求。

2.2.1 麦克风电器性能推荐

- 麦克类型：全向型 MEMS 麦克风
- 灵敏度
  - 1 Pa 声压下模拟麦灵敏度不低于 -38 dBV，数字麦灵敏度要求不低于 -26 dB。
  - 公差控制在 ±2 dB，对于麦克阵列推荐采用 ±1 dB 公差。
- 信噪比 (SNR)
  - 信噪比：信噪比不低于 62 dB，推荐 > 64 dB
  - 频率响应在 50~16 KHz 范围内的波动在 ±3 dB 之内
  - 麦克风的 PSRR 应大于 55 dB

2.2.2 麦克风结构设计建议

- 麦克孔孔径或宽度推荐大于 1 mm，拾音管道尽量短，腔体尽可能小，保证麦克和结构组件配合的谐振频率在 9 KHz 以上。
- 拾音孔深度和直径比小于 2:1，壳体厚度推荐 1 mm，如果壳体过厚，需增大开孔面积。
- 麦克孔上需通过防尘网进行保护。
- 麦克风与设备外壳之间必须加硅胶套或泡棉等进行密封和防震，需进行过盈配合设计，以保证麦克的密封性。
- 麦克孔不能被遮挡，底部拾音的麦克风需结构上增加底部凸起，保证麦克风与桌面等平面有一定间隙。
- 麦克需远离喇叭等会产生噪音或振动的物体摆放，且与喇叭音腔之间通过橡胶垫等隔离缓冲。

2.2.3 麦克阵列设计建议

客户可采用双麦克或三麦克方案：

- 双麦克方案：2 个麦克风之间间距保持 4~6.5 cm，连接 2 个麦克风的轴线应平行于水平线，且 2 个麦克的中心尽量靠近产品水平方向的中心。
- 三麦克方案：3 个麦克风等间距并且成正圆分布（夹F互成 120 度），间距要求 4~6.5 cm。

在选择阵列中的麦克风时，有如下注意事项：

- 麦克类型：全向型硅麦，推荐同一个阵列内的麦克应使用同一厂家的同一型号，不建议混用。
- 灵敏度：麦克阵列中各麦克灵敏度差异在 3 dB 之内。
- 相位差：多麦克阵列中麦克之间的相位差控制在 10° 以内。
- 麦克阵列中各麦克的结构设计，推荐采用相同的设计，以保证结构设计的一致性。

2.2.4 麦克风结构密封性建议

用橡皮泥等材料封堵麦克拾音孔，密封前后麦克风采集信号的幅度衰减 25 dB 为合格，推荐 30 dB。测试方法：

1. 麦克风正上方 0.5 米处，播放白噪声，麦克风处音量 90 dB。
2. 使用麦克风阵列录制 10s 以上，存储为录音文件 A。
3. 用橡皮泥等材料封堵麦克拾音孔，使用麦克风阵列录制 10s 以上，存储为录音文件 B。
4. 对比两个文件的频谱，需保证 100~8 KHz 频段内整体衰减 25dB 以上。

2.2.5 回声参考信号设计建议

- 回声参考信号推荐尽量靠近喇叭侧，推荐从 DA 后级 PA 前级回采。
- 扬声器音量最大时，输入到麦克的回声参考信号不能有饱和失真，最大音量下喇叭功放输出 THD 满足 100 Hz 小于 10%，200 Hz 小于 6%，350 Hz 以上频率，小于 3%。
- 扬声器音量最大时，麦克处拾音的声压不超过 102 dB (1KHz)。
- 回声参考信号电压不超过 ADC 的最大允许输入电压，电压过高需增加衰减电路。
- 从 D 类功放输出引参考回声信号需增加低通滤波器，滤波器的截止频率推荐 > 22 KHz。
- 音量最大播放时，回采信号峰值 -3 到 -5 dB。

2.2.6 麦克风阵列一致性验证

要求各个麦克风采样信号幅度相差小于 3 dB，测试方法：

1. 麦克风正上方 0.5 米处，播放白噪声，麦克风处音量 90 dB。
2. 使用麦克风阵列录制 10s 以上，查看各 mic 录音幅度和音频采样率是否一致。

2.3 从 V1.* 迁移到 V2.*

2.3.1 输入数据格式修改

新版本通过 `input_format` 参数定义了输入数据中音频通道的排列方式。字符串中的每个字符代表一种通道类型：

字符	描述
M	麦克风通道
R	播放参考通道
N	未使用或未知通道

示例： MMNR 表示四个通道，依次为：麦克风通道、麦克风通道、未使用通道、播放参考通道。

2.3.2 配置和初始化

- 1. 旧的配置初始化方法 AFE_CONFIG_DEFAULT() 已被移除。请使用 `afe_config_init` 来初始化配置：

```
afe_config_t *afe_config = afe_config_init("MMNR", models, AFE_TYPE_SR,
↪ AFE_MODE_HIGH_PERF);
afe_config_print(afe_config); // print all configurations
```

- 2. ESP_AFE_SR_HANDLE 和 ESP_AFE_VC_HANDLE 已被移除。使用 `esp_afe_handle_from_config` 来创建实例：

```
esp_afe_sr_iface_t *afe_handle = esp_afe_handle_from_config(afe_
↪config);
```

备注： AFE v2.0 引入了额外的配置选项。详细信息请参阅*AFE* 和*VAD*。

Chapter 3

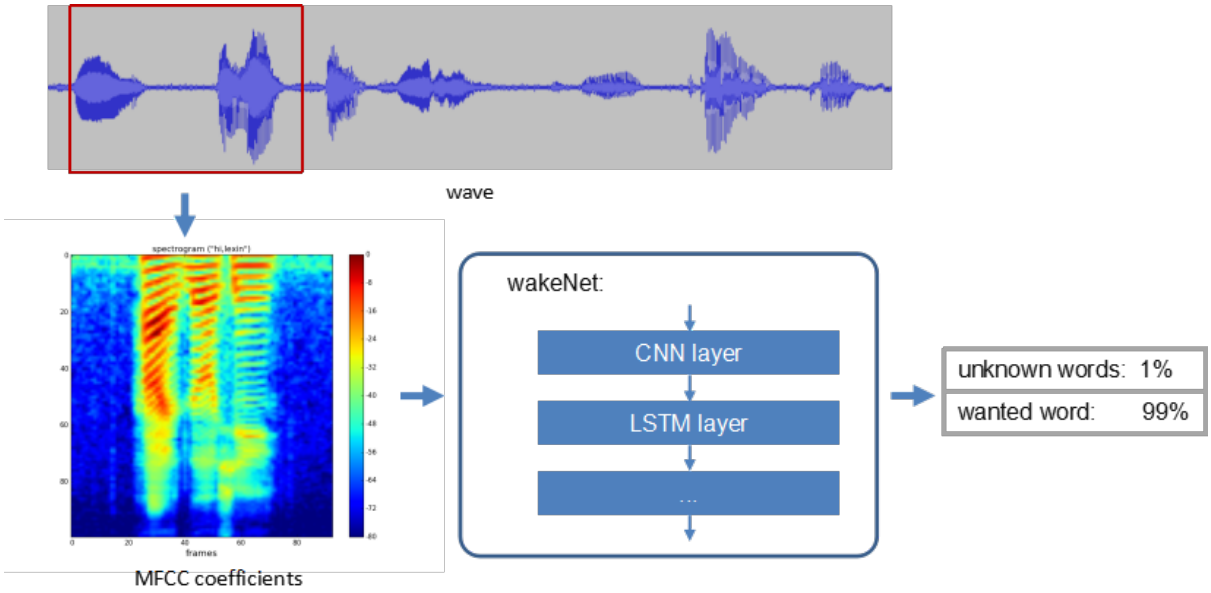
唤醒词

3.1 WakeNet 唤醒词模型

WakeNet 是一个基于神经网络，为低功耗嵌入式 MCU 设计的唤醒词模型，目前支持 5 个以内的唤醒词识别。

3.1.1 概述

WakeNet 的流程图如下：



- **语音特征 (Speech Feature)** 我们使用 MFCC 方法提取语音频谱特征。输入的音频文件采样率为 16 KHz，单声道，编码方式为 signed 16-bit。每帧窗宽和步长均为 30 ms。
- **神经网络 (Neural Network)** 神经网络结构已经更新到第 9 版，其中：
  - WakeNet1、WakeNet2、WakeNet3、WakeNet4、WakeNet6 and WakeNet7 已经停止使用。
  - WakeNet5 应用于 ESP32 芯片。
  - WakeNet8 和 WakeNet9 应用于 ESP32-S3 芯片，模型基于 Dilated Convolution 结构。

注意，WakeNet5、WakeNet5X2 和 WakeNet5X3 的网络结构一致，但是 WakeNet5X2 和 WakeNet5X3 的参数比 WakeNet5 要多。请参考资源消耗 来获取更多细节。

- **Keyword Trigger Method** 对连续的音频流，为准确判断关键词的触发，我们通过计算若干帧内识别结果的平均值 M，来判断是否触发。当 M 大于指定阈值，则发出触发的命令。

以下表格展示在不同芯片上的模型支持：

Chip	ESP32			ESP32S3			
model	WakeNet 5			WakeNet 8		WakeNet 9	
	WakeNet 5	WakeNet 5X2	WakeNet 5X3	Q16	Q8	Q16	Q8
Hi,Lexin	√	√	√				√
nihaoxiaozhi	√		√				√
nihaoxiaoxin			√				
xiaoaitongxue							√
Alexa				√			√
Hi,ESP							√
Customized word							√

3.1.2 WakeNet 的使用

- WakeNet 模型选择
  WakeNet 模型选择请参考*flash model 介绍* 。
  自定义的唤醒词，请参考*乐鑫语音唤醒词定制流程* 。
- WakeNet 模型运行
  WakeNet 目前包含在语音前端算法*AFE* 中，默认为运行状态，并将识别结果通过 AFE fetch 接口返回。
  如果用户无需使用 WakeNet 唤醒，请在 AFE 配置时选择：

```
afe_config.wakenet_init = False.
```

  如果用户想临时关闭/打开 WakeNet, 请在运行过程中调用：

```
afe_handle->disable_wakenet(afe_data)
afe_handle->enable_wakenet(afe_data)
```

3.1.3 资源消耗

有关本模型的资源消耗情况，请见资源消耗。

3.2 乐鑫语音唤醒方案客户定制流程

3.2.1 唤醒词定制服务

乐鑫提供 **语音唤醒词定制**服务，详情如下：

1. “HI 乐鑫”，“你好小鑫”等官方开放的唤醒词，客户可直接商用
   - 同时，乐鑫会逐渐开放更多可免费商用的唤醒词
2. 除官方开放的唤醒词，乐鑫还可为客户提供 **唤醒词定制服务**，主要分如下两种情况：
   - 客户提供唤醒词语料
     - 需要提供大于 2 万条合格的语料，具体语料需求见训练语料要求
     - 语料提供给乐鑫后，需要 2 ~ 3 周进行模型训练及调优
     - 根据量级收取少量模型定制费用
   - 如果客户不提供唤醒词语料
     - 所有训练语料由乐鑫采集提供
     - 乐鑫需要一定时间收集语料，具体需要分别讨论；语料准备好后，需要 2 ~ 3 周进行模型训练及调优
     - 根据量级收取少量模型定制费用，语料采集费用另算
   - 定制的具体时间和费用取决于 **唤醒词定制的数量**以及 **产品量产数量**，详情请联系 乐鑫销售人员 。
3. 对于乐鑫唤醒词模型：
   - 目前单个模型最多支持 5 个及以内的唤醒词识别
   - 每个唤醒词通常由 3-6 音节组成，比如 “hi 乐鑫”，“Alexa”，“小爱同学”，“你好天猫” 等
   - 可多个唤醒模型一起使用，具体需根据客户应用的资源消耗确定
   - 更多详情，请见 *WakeNet* 唤醒词模型

训练语料要求

客户可自备训练语料或向第三方采购，对于语料有以下要求：

- 语料音频格式要求
  - 采样率（sample rate）：16 KHz
  - 编码（encoding）：16-bit signed int
  - 通道数（channel）：mono
  - 格式：wav
- 语料采集要求
  - 采样人数：最好样本可以大于 500 人，其中男女，年龄分布均衡，儿童不小于 100 人
  - 采样环境：环境噪声低（< 40 dB），建议在语音室等专业环境下录制
  - 录制设备：高保真麦克风
  - **录制场景：**
    - 距离麦克风 1 m 处每人录制 15 遍，其中 5 遍快语速，5 遍正常语速，5 遍慢语速；
    - 距离麦克风 3 m 处每人录制 15 遍，其中 5 遍快语速，5 遍正常语速，5 遍慢语速
  - 样本命名需体现样本信息：如 `female_age_fast_id.wav` 或有单独表格记录每个样本的年龄，性别等信息

3.2.2 硬件设计与测试服务

语音唤醒效果与硬件设计以及腔体结构有很大关系。因此，请认真阅读以下内容：

1. 硬件设计要求
   - 各类语音音箱类设计：乐鑫可提供 **原理图** / **PCB** 等设计参考，客户可以根据自身具体需求设计修改，设计完毕后，乐鑫还可提供审阅服务，避免常见设计问题。
   - 腔体结构：建议有专门的声学人员参与设计，乐鑫不提供 ID 设计类参考，客户可参考市面上的主流音箱腔体设计，例如天猫精灵、小度音箱、谷歌音箱等。
2. 硬件设计好后，客户可通过以下简单测试，验证硬件设计效果（下列测试都是基于语音室环境，客户可以根据自身测试环境做调整）
   - 录音测试，验证 mic、codec 录音增益以及失真情况
     - 音源 90 dB，距离 0.1 m 播放样本，调节增益，保证录音样本不饱和
     - 使用扫频文件 (0~20 KHz)，使用 16 KHz 采样率录音，音频不会出现明显频率混叠
     - 录制 100 个语音样本，使用公开的云端语音识别端口识别，识别率达到指定标准
   - 播音测试，验证功率放大器 (PA)、喇叭的失真情况
     - 测试 PA 功率 @1% 总谐波失真 (THD)
   - 语音算法测试，验证 AEC、BFM、NS 效果
     - 首先需要注意下参考信号延时，不同的 AEC 算法有不同的要求
     - 以实际产品场景为测试指标，例如 mic 播放 `85DB-90DB 大梦想家.wav`，设备回采
     - 保存回声参考信号、回声消除后的信号分析，对比查看 AEC、BFM、NS 等效果
   - DSP 性能测试，验证 DSP 参数是否合适，同时尽可能减少 DSP 算法中的非线性失真
     - 降噪 (Noise Suppression) 算法性能测试
     - 回声消除 (Acoustic Echo Cancellation) 算法性能测试
     - 语音增强 (Speech Enhancement) 算法性能测试
3. 硬件设计完毕后，**可寄送** 1-2 台硬件至乐鑫，乐鑫会基于客户整机做唤醒词性能调优。

Chapter 4

语音活动检测模型

VADNet 是一个基于神经网络的语音活动检测模型，专为低功耗嵌入式 MCU 设计。

4.1 概述

VADNet 采用了与 WakeNet 相似的模型结构和数据处理流程，更多实现细节可参考 *WakeNet* 中的说明。

VADNet 训练数据包括了大约 5000 小时中文数据，5000 小时英文数据，还有 5000 小时的多语言数据。

4.2 使用 VADNet

- 选择 VADNet 模型

```
idf.py menuconfig
ESP Speech Recognition -> Select voice activity detection -> voice␣
↪activity detection (vadnet1 medium).
```

- 运行 VADNet

  VADNet 当前集成在音频前端处理模块 中，默认处于启用状态，通过 AFE 的 fetch 接口返回检测结果。

  常用 VAD 参数配置如下：

```
afe_config->vad_init = true            //␣
↪是否在AFE流水线中初始化VAD，默认启用
afe_config->vad_min_noise_ms = 1000;   // 噪声/
↪静音段的最短持续时间（毫秒）
afe_config->vad_min_speech_ms = 128;   // 语音段的最短持续时间（毫秒）
afe_config->vad_delay_ms = 128;        //␣
↪VAD首帧触发到语音首帧数据的延迟量
afe_config->vad_mode = VAD_MODE_1;     // 模式值越大，语音触发概率越高
```

  如需临时启用/禁用/重置 VADNet，可使用以下接口：

```
afe_handle->disable_vad(afe_data);  // 禁用VAD
afe_handle->enable_vad(afe_data);    // 启用VAD
afe_handle->reset_vad(afe_data);    // 重置VAD状态
```

- VAD 缓存与检测

  VAD 配置中的两个特性可能导致语音首帧触发延迟：

  1. VAD 算法固有延迟：VAD 无法在首帧精准触发，可能有 1-3 帧延迟
  2. 防误触机制：需持续触发时间达到配置参数 ‘vad_min_speech_ms‘才会正式触发

为避免上述原因导致语音首字截断，AFE V2.0 新增了 VAD 缓存机制。可通过检查 vad_cache_size 判断是否需要保存 VAD 缓存：

```
afe_fetch_result_t* result = afe_handle->fetch(afe_data);
if (result->vad_cache_size > 0) {
    printf("vad cache size: %d\n", result->vad_cache_size);
    fwrite(result->vad_cache, 1, result->vad_cache_size, fp);
}

printf("vad state: %s\n", res->vad_state==VAD_SILENCE ? "noise" :
↪"speech");
```

4.3 资源占用

本模型的资源占用情况请参考资源占用说明。

Chapter 5

命令词

5.1 MultiNet 命令词识别模型

MultiNet 是为了在 ESP32-P4 系列上离线实现多命令词识别而设计的轻量化模型，目前支持 200 个以内的自定义命令词识别。

- 支持用户自定义命令词
- 支持运行过程中增加/删除/修改命令词语
- 最多支持 200 个命令词
- 支持单次识别和连续识别两种模式
- 轻量化，低资源消耗
- 低延时，延时 500 ms 内
- 模型单独分区，支持用户应用 OTA

MultiNet 输入为经过前端语音算法（AFE）处理过的音频（格式为 16 KHz，16 bit，单声道）。通过对音频进行识别，则可以对应到相应的汉字或单词。

请参考*Models Benchmark* 去查看当前不同芯片支持的模型。

用户选择不同的模型的方法请参考模型加载 。

备注： 其中以 Q8 结尾的模型代表模型的 8 bit 版本，表明该模型更加轻量化。

5.2 命令词识别原理

命令词识别原理如下图所示：

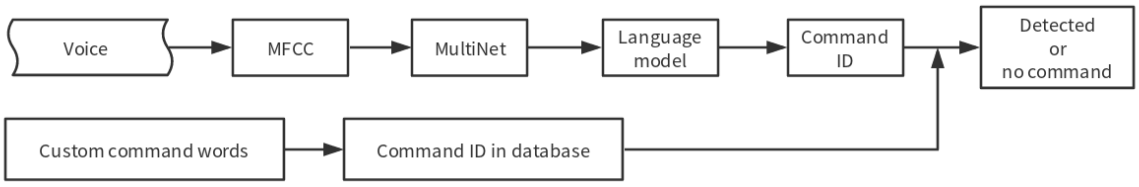


图 1: speech_command-recognition-system

5.3 命令词格式要求

不同版本的 MultiNet 命令词格式不同。命令词需要满足特定的格式，具体如下：

5.4 自定义命令词方法

备注： 不支持中英文混合

不能含有阿拉伯数字和特殊字符

英语自定义命令词方法请参考英文文档

MultiNet 支持多种且灵活的命令词设置方式，可通过在线或离线方法设置命令词，还允许随时动态增加/删除/修改命令词。

MultiNet5 和 MultiNet6 使用汉语拼音作为基本识别单元。比如 "打开空调"，应该写成 "da kai kong tiao"，请使用以下工具将汉字转为拼音：tool/multinet_pinyin.py 。

5.4.1 MultiNet6 定义方法：

- 中文通过修改 model/multinet_model/fst/commands_cn.txt
  格式如下，第一个数字代表 command id, 后面为指令的中文拼音，两者由空格隔开，拼音间也由空格隔开，Command id 不能为 0

```
# command_id command_sentence
1 da kai kong tiao
2 guan bi kong tiao
```

5.4.2 MultiNet5 定义方法：

- 通过 `menuconfig`
  1. `idf.py menuconfig` > `ESP Speech Recognition` > `Add Chinese speech commands/Add English speech commands`，添加命令词。具体也可参考 ESP-Skainet 中的 example。

  注意，单个 Command ID 可以支持多个短语，比如 "打开空调" 和 "开空调" 表示的意义相同，则可以将其写在同一个 Command ID 对应的词条中，用英文字符 "," 隔开相邻词条（"," 前后无需空格）。

  2. 在代码里调用以下 API：

```
/**
* @brief Update the speech commands of MultiNet by menuconfig
*
* @param multinet            The multinet handle
*
* @param model_data          The model object to query
*
* @param langugae            The language of MultiNet
*
* @return
*     - ESP_OK                  Success
*     - ESP_ERR_INVALID_STATE   Fail
*/
esp_err_t esp_mn_commands_update_from_sdkconfig(esp_mn_iface_t␣
↪*multinet, const model_iface_data_t *model_data);
```

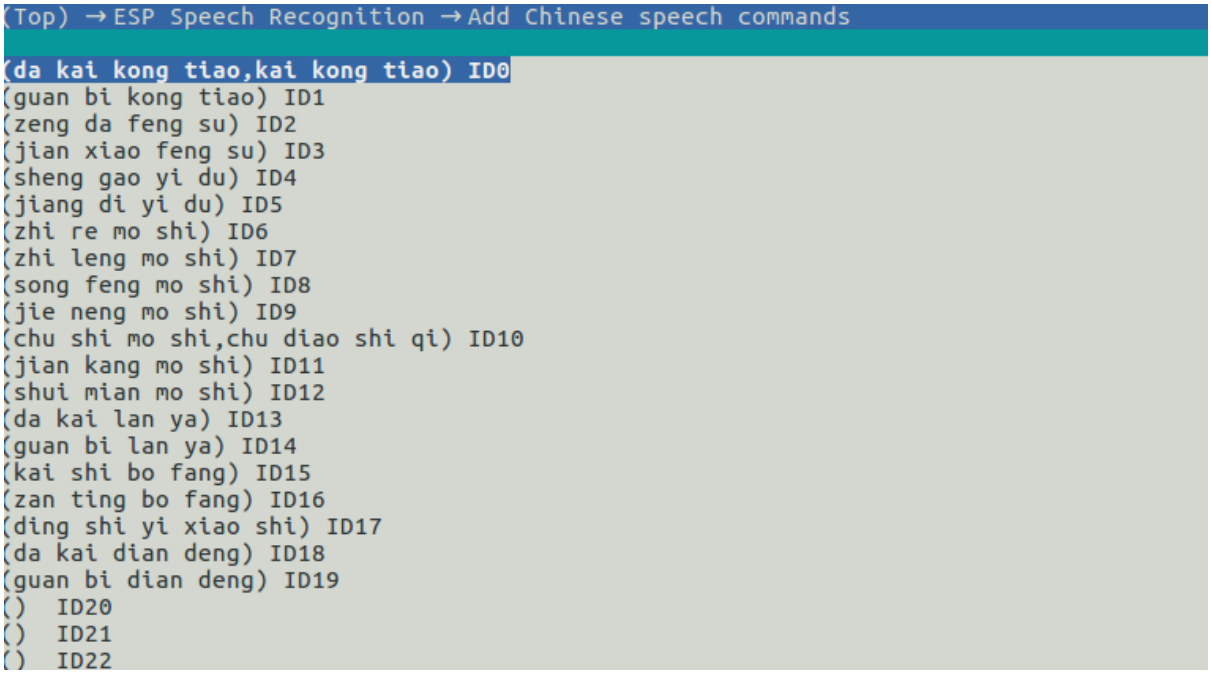


图 2: menuconfig_add_speech_commands

5.4.3 通过调用 API 修改

指令还可以通过调用 API 修改，这种方法对于 MultiNet5 和 MultiNet6 都适用。

- 应用新的修改操作，所有添加、移除、修改及清空操作在调用后才会被应用。

```
/**
* @brief Update the speech commands of MultiNet
*
* @Warning: Must be used after [add/remove/modify/clear] function,
*           otherwise the language model of multinet can not be␣
↪updated.
*
* @return
*     - NULL                  Success
*     - others                The list of error phrase which can not be␣
↪parsed by multinet.
*/
esp_mn_error_t *esp_mn_commands_update();
```

- 添加一条新指令，如果指令格式不正确则返回 ESP_ERR_INVALID_STATE。

```
/**
* @brief Add one speech commands with command string and command ID
*
* @param command_id      The command ID
* @param string  The command string of the speech commands
*
* @return
*     - ESP_OK                  Success
*     - ESP_ERR_INVALID_STATE   Fail
*/
esp_err_t esp_mn_commands_add(int command_id, char *string);
```

- 移除一条指令，如果该指令不存在则返回 ESP_ERR_INVALID_STATE。

```
/**
```

(下页继续)

(续上页)

```
* @brief Remove one speech commands by command string
*
* @param string  The command string of the speech commands
*
* @return
*     - ESP_OK                  Success
*     - ESP_ERR_INVALID_STATE   Fail
*/
esp_err_t esp_mn_commands_remove(char *string);
```

- 修改一条指令，如果该指令不存在则返回 ESP_ERR_INVALID_STATE。

```
/**
* @brief Modify one speech commands with new command string
*
* @param old_string  The old command string of the speech commands
* @param new_string  The new command string of the speech commands
*
* @return
*     - ESP_OK                  Success
*     - ESP_ERR_INVALID_STATE   Fail
*/
esp_err_t esp_mn_commands_modify(char *old_string, char *new_string);
```

- 清空所有指令。

```
/**
* @brief Clear all speech commands in linked list
*
* @return
*     - ESP_OK                  Success
*     - ESP_ERR_INVALID_STATE   Fail
*/
esp_err_t esp_mn_commands_clear(void);
```

- 打印缓存的指令, 只有当调用 esp_mn_commands_update() 缓存指令才会被应用.

```
/**
* @brief Print all commands in linked list.
*/
void esp_mn_commands_print(void);
```

- 打印当前已经被应用的指令.

```
/**
* @brief Print all commands in linked list.
*/
void esp_mn_active_commands_print(void);
```

5.5 MultiNet 的使用

MultiNet 命令词识别建议和 ESP-SR 中的 AFE 声学算法模块一起运行，具体请参考*AFE 介绍及使用*)。当用户配置完成 AFE 后，请按照以下步骤配置和运行 MultiNet。

5.5.1 MultiNet 初始化

- 模型加载与初始化，请参考模型加载
- 设置命令词，请参考命令词格式要求

5.5.2 MultiNet 运行

当用户开启 AFE 且使能 WakeNet 后，则可以运行 MultiNet。但需要注意以下几点要求：

- 传入帧长和 AFE fetch 帧长长度相等
- 支持音频格式为 16 KHz，16 bit，单通道。AFE fetch 拿到的数据也为这个格式
- 确定需要传入 MultiNet 的帧长

  ```
  int mu_chunksize = multinet->get_samp_chunksize(model_data);
  ```

  `mu_chunksize` 是需要传入 MultiNet 的每帧音频的 `short` 型点数，这个大小和 AFE 中 fetch 的每帧数据点数完全一致。
- MultiNet 识别

  我们将 AFE 实时 `fetch` 到的数据送入以下 API：

  ```
  esp_mn_state_t mn_state = multinet->detect(model_data, buff);
  ```

  `buff` 的长度为 `mu_chunksize * sizeof(int16_t)`。

5.5.3 MultiNet 识别结果

命令词识别必须和唤醒搭配使用，当唤醒后可以运行命令词的检测。

命令词模型在运行时，会实时返回当前帧的识别状态 `mn_state` ，目前分为以下几种识别状态：

- ESP_MN_STATE_DETECTING

  该状态表示目前正在识别中，还未识别到目标命令词。
- ESP_MN_STATE_DETECTED

  该状态表示目前识别到了目标命令词，此时用户可以调用 `get_results` 接口获取识别结果。

  ```
  esp_mn_results_t *mn_result = multinet->get_results(model_data);
  ```

  识别结果的信息存储在 `get_result` API 的返回值中，返回值的数据类型如下：

  ```
  typedef struct{
      esp_mn_state_t state;
      int num;                    // The number of phrase in list, num<=5.␣
  ↪When num=0, no phrase is recognized.
      int phrase_id[ESP_MN_RESULT_MAX_NUM];       // The list of phrase␣
  ↪id.
      float prob[ESP_MN_RESULT_MAX_NUM];          // The list of␣
  ↪probability.
  } esp_mn_results_t;
  ```

  其中，
  - `state` 为当前识别的状态
  - `num` 表示识别到的词条数目，`num` <= 5，即最多返回 5 个候选结果
  - `phrase_id` 表示识别到的词条对应的 Phrase ID
  - `prob` 表示识别到的词条识别概率，从大到到小依次排列

  用户可以使用 `phrase_id[0]` 和 `prob[0]` 拿到概率最高的识别结果。
- ESP_MN_STATE_TIMEOUT

  该状态表示长时间未检测到命令词，自动退出。等待下次唤醒。

单次识别模式和连续识别模式：当命令词识别返回状态为 `ESP_MN_STATE_DETECTED` 时退出命令词识别，则为单次识别模式；当命令词识别返回状态为 `ESP_MN_STATE_TIMEOUT` 时退出命令词识别，则为连续识别模式；

5.6 资源消耗

有关本模型的资源消耗情况，请见资源消耗。

Chapter 6

TTS 语音合成模型

乐鑫 TTS 语音合成模型是一个为嵌入式系统设计的轻量化语音合成系统，具有如下主要特性：

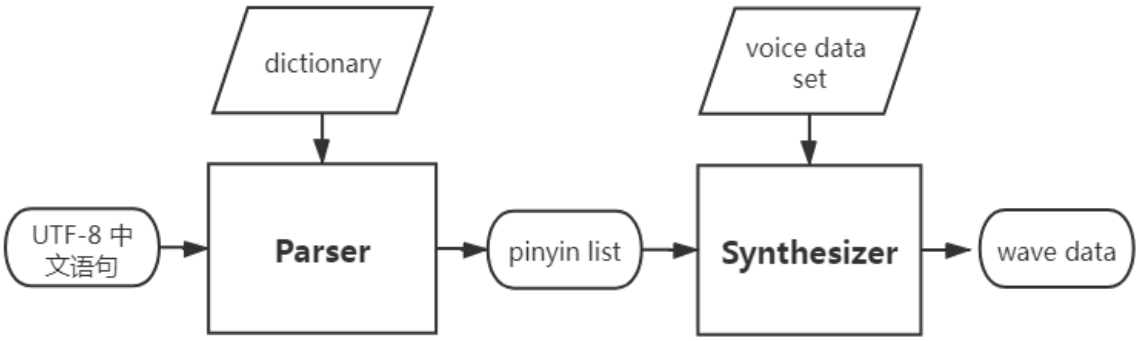
- 目前 **仅支持中文**
- 输入文本采用 UTF-8 编码
- 输出格式采用流输出，可减少延时
- 多音词发音自动识别
- 可调节合成语速
- 数字播报优化
- 自定义声音集（敬请期待）

6.1 简介

乐鑫 TTS 的当前版本基于拼接法，主要组成部分包括：

- 解析器 (Parser)：根据字典与语法规则，将输入文本（采用 UTF-8 编码）转换为拼音列表。
- 合成器 (Synthesizer)：根据解析器输出的拼音列表，结合预定义的声音集，合成波形文件。默认输出格式为：单声道，16 bit @ 16000Hz。

系统框图如下：



6.2 简单示例

- esp-tts/samples/xiaoxin_speed1.wav (voice=xiaoxin, speed=1)：欢迎使用乐鑫语音合成，支付宝收款 72.1 元，微信收款 643.12 元，扫码收款 5489.54 元
- esp-tts/samples/S2_xiaole_speed2.wav (voice=xiaole, speed=2)：支付宝收款 1111.11 元

6.3 编程指南

```
#include "esp_tts.h"
#include "esp_tts_voice_female.h"
#include "esp_partition.h"

/*** 1. create esp tts handle  ***/


// initial voice set from separate voice data partition

const esp_partition_t* part=esp_partition_find_first(ESP_PARTITION_TYPE_DATA, ESP_
↪PARTITION_SUBTYPE_DATA_FAT, "voice_data");
if (part==0) printf("Couldn't find voice data partition!\n");
spi_flash_mmap_handle_t mmap;
uint16_t* voicedata;
esp_err_t err=esp_partition_mmap(part, 0, part->size, SPI_FLASH_MMAP_DATA, (const␣
↪void**)&voicedata, &mmap);
esp_tts_voice_t *voice=esp_tts_voice_set_init(&esp_tts_voice_template, voicedata);

// 2. parse text and synthesis wave data
char *text="欢迎使用乐鑫语音合成";
if (esp_tts_parse_chinese(tts_handle, text)) {  // parse text into pinyin list
    int len[1]={0};
    do {
        short *data=esp_tts_stream_play(tts_handle, len, 4); // streaming synthesis
        i2s_audio_play(data, len[0]*2, portMAX_DELAY);  // i2s output
    } while(len[0]>0);
    i2s_zero_dma_buffer(0);
}
```

更多参考，请前往 esp-tts/esp_tts_chinese/include/esp_tts.h 查看 API 定义，或参考 ESP-Skainet 中 chinese_tts 示例.

6.4 资源消耗

有关本模型的资源消耗情况，请见资源消耗。

Chapter 7

模型选择和加载

本文档解释了如何为 ESP-SR 选择和加载模型。

7.1 模型选择

ESP-SR 允许您通过 `menuconfig` 界面选择所需的模型。要配置模型：

1. 运行 `idf.py menuconfig`
2. 导航到 **ESP Speech Recognition**
3. 配置以下选项：- **噪声抑制模型** - **VAD 模型** - **WakeNet 模型** - **MultiNet 模型**

```
    model data path (Read model data from flash)  --->
    Afe interface (afe interface(version: v1))  --->
    Select noise suppression model (noise suppression (WebRTC))  --->
    Select voice activity detection (voice activity detection (WebRTC))  --->
    Load Multiple Wake Words  --->
    Chinese Speech Commands Model (None)  --->
    English Speech Commands Model (None)  --->
```

7.2 更新分区表

您必须添加一个 *partition.csv* 文件，并确保有足够的空间来存储所选的模型。在项目的 `partitions.csv` 文件中添加以下行，以分配模型所需的空间：

```
model,  data,        ,         ,     6000K
```

- 将 `6000K` 替换为您根据所选模型自定义的分区大小。
- `model` 是分区标签（固定值）。

7.3 模型加载

7.3.1 ESP-IDF 框架

ESP-SR 通过其 CMake 脚本自动处理模型加载：

1. 烧写设备并包含所有组件：`idf.py flash` 此命令会自动加载所选模型。

2. 在代码调试时（不重新烧写模型）：`idf.py app-flash`

备注： 模型加载脚本在 `esp-sr/CMakeLists.txt` 中定义。模型在初始烧写时会被写入标签为 `model` 的分区。

7.3.2 Arduino 框架

手动生成和加载模型：

1. 使用提供的 Python 脚本生成 `srmodels.bin`：

```
python {esp-sr_path}/movemodel.py -d1 {sdkconfig_path} -d2 {esp-sr_path} -d3
↪{build_path}
```

   参数：
   - `esp-sr_path`：您的 ESP-SR 组件目录路径
   - `sdkconfig_path`：项目的 `sdkconfig` 文件路径
   - `build_path`：项目的构建目录（通常是 `your_project_path/build`）
2. 生成的 `srmodels.bin` 将位于：`{build_path}/srmodels/srmodels.bin`
3. 将生成的二进制文件烧写到设备上。

重要： 仅在 `menuconfig` 中更改模型配置后，请重新生成 `srmodels.bin`。

Chapter 8

性能测试结果

8.1 AFE

8.1.1 资源消耗

表 1: AFE 配置和算法流程

Config	Pipeline
MR, SR, LOW_COST	`|AEC(SR_LOW_COST)| -> |VAD(vadnet1_medium)| -> |WakeNet(wn9_hilexin,)|`
MR, SR, HIGH_PERF	`|AEC(SR_HIGH_PERF)| -> |VAD(vadnet1_medium)| -> |WakeNet(wn9_hilexin,)|`
MR, VC, LOW_COST	`|AEC(VOIP_LOW_COST)| -> |NS(nsnet2)| -> |VAD(vadnet1_medium)|`
MR, VC, HIGH_PERF	`|AEC(VOIP_HIGH_PERF)| -> |NS(nsnet2)| -> |VAD(vadnet1_medium)|`
MMNR, SR, LOW_COST	`|AEC(SR_LOW_COST)| -> |SE(BSS)| -> |VAD(vadnet1_medium)| -> |WakeNet(wn9_hilexin,)|`
MMNR, SR, HIGH_PERF	`|AEC(SR_HIGH_PERF)| -> |SE(BSS)| -> |VAD(vadnet1_medium)| -> |WakeNet(wn9_hilexin,)|`

备注:

- **MR:** 一个麦克风通道和一个播放通道
- **MMNR:** 两个麦克风通道和一个播放通道
- **Models:** nsnet2, vadnet1_medium, wn9_hilexin

表 2: AFE 配置和性能

Config	Internal RAM (KB)	PSRAM (KB)	Feed CPU usage (1 core,%)	Fetch CPU usage (1 core,%)
MR, SR, LOW_COST	73.6	733.2	10.6	11.2
MR, SR, HIGH_PERF	73.3	733.2	10.6	11.2
MR, VC, LOW_COST	74.4	821.3	40.2	5.7
MR, VC, HIGH_PERF	116.7	823.9	42.4	5.7
MMNR, SR, LOW_COST	78.0	1173.0	28.2	24.8
MMNR, SR, HIGH_PERF	78.0	1173.0	28.2	24.8

8.2 WakeNet

8.2.1 资源消耗

Model Type	RAM	PSRAM	Average Running Time per Frame	Frame Length
Quantised WakeNet9 @ 2 channel	16 KB	324 KB	2.6 ms	32 ms
Quantised WakeNet9 @ 3 channel	20 KB	347 KB	3.1 ms	32 ms

8.2.2 性能测试

Distance	Quiet	Stationary Noise (SNR = 4 dB)	Speech Noise (SNR = 4 dB)	AEC I nterruption (-10 dB)
1 m	98%	96%	94%	96%
3 m	98%	96%	94%	94%

误触发率：12 小时 1 次

备注： 以上测试结果基于 ESP32-S3-Korvo V4.0 开发板和 WakeNet9(Alexa) 模型。

8.3 MultiNet

8.3.1 资源消耗

Model Type	Internal RAM	PSRAM	Average Running Time per Frame	Frame Length
MultiNet 7	18 KB	2920 KB	8 ms	32 ms

8.3.2 Word Error Rate 性能测试

Model Type	aishell test
MultiNet 5_cn	9.5%
MultiNet 6_cn	5.2%

备注： 中文使用没有声调的拼音单元去计算 WER。

8.3.3 Speech Commands 性能测试（空调控制场景）

Model Type	Distance	Quiet	Stationary Noise (SNR=5~10dB dB)	Speech Noise (SNR=5~10dB dB)
MultiNet 5_cn	3 m	88.9%	66.1%	67.5%
MultiNet 6_cn	3 m	98.8%	88.3%	88.0%
MultiNet 6_cn_ac	3 m	97.1%	95.1%	96.8%

备注： MultiNet6_cn_ac 在空调场景数据集上进行了进一步的微调，所以在空调控制场景具有更好的性能。

8.4 TTS

8.4.1 资源消耗

Flash image size: 2.2 MB

RAM runtime: 20 KB

8.4.2 性能测试

CPU 负载测试（ESP32 @240 MHz）：

Speech Rate	0	1	2	3	4	5
Times faster than real time	4.5	3.2	2.9	2.5	2.2	1.8

8.5 NSNET

8.5.1 性能测试

8.5.2 数据集：array_onemic_nnoise_20230608(按照亚马逊声学认证标准录制测试集)

	dnsmos
nsnet1	2.4
nsnet2	2.71

Chapter 9

测试方法与测试报告

为了保证 DUT 的性能，可通过测试确定 DUT 在以下方面的表现：

- 唤醒率
- 识别率
- 误唤醒率
- 唤醒打断率
- 响应时间

9.1 测试场景要求

以上测试需在合适的测试房间中进行，测试房间应满足如下要求：

- **房间大小**
  - 面积：不小于 4 m * 3.2 m
  - 高度：不低于 2.3 m
- **房间装饰**
  - 地板需配有地毯，天花板需配备常见声学阻尼材料，墙面应有 1 到 2 面墙上挂有窗帘，防止强反射。
  - 房间混响 (RT60) 3 在 [125, 8k] 范围内，要满足 0.2-0.7s 的要求。
  - 不要使用消音室。
- **环境底噪**：必须 < 35 dBA，最好 < 30 dBA
- **温度和湿度要求**：20±10°C，相对湿度为 50%±20%。
- **DUT、外部噪声、人声的放置**：
  - DUT、外部噪声、人声的具体位置和相对位置应根据 DUT 的实际应用场景进行安排。

备注： RT60、环境底噪和 DUT、外部噪声、人声的放置应在所有测试中保持不变。

9.2 测试案例设计

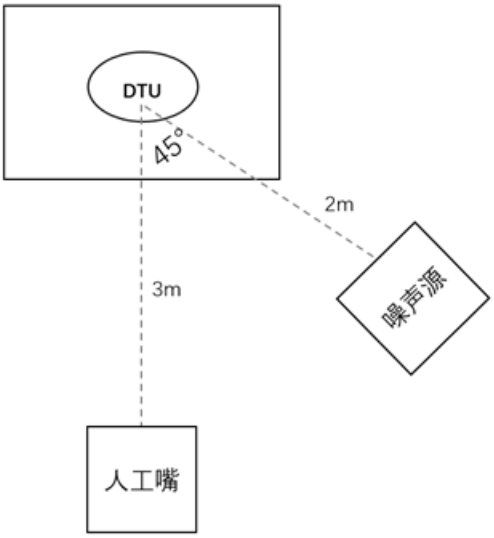
在设计测试案例时，建议按照产品的实际应用场景，考虑 **以下部分或全部参数**：

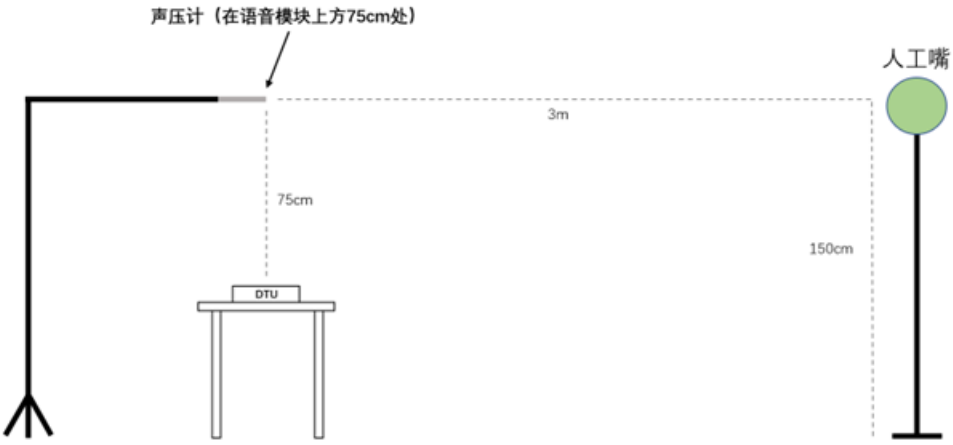
- **不同噪声源**
  - 白噪声
  - 人类噪声
  - 音乐
  - 新闻
  - ……

  - 有必要时还可以增加多噪声源的测试场景。
- **不同噪声分贝**
  - < 35 dBA
  - 45 dBA
  - 55 dBA
  - 65 dBA
- **不同人声分贝**
  - 54 dBA
  - 59 dBA
  - 64 dBA
- **不同 SNR**
  - 9 dBA
  - 4 dBA
  - -1 dBA

9.3 乐鑫测试与结果

在本章节描述的所有测试中，DUT、外部噪声、人声的具体位置和相对位置如下所述。





具体来说，

- DUT 距离地面高度 0.75 米
- 人工嘴（人声源）距离地面高度 1.5 米，距离 DUT 直线水平距离 3 米
- 噪声源在相对人工嘴 45°C 角处，距离地面高度 1.2 米，距离 DUT 直线水平距离 2 米
- 声压计位于 DUT 正上方 0.75 米

9.3.1 唤醒率测试

唤醒率：指当设备处于待唤醒状态时被唤醒成功的概率。

乐鑫唤醒率测试和结果

测试案例	噪声类型	噪声分贝	人声分贝	SNR	唤醒率
1	/	/	59 dBA	/	99%
2	白噪声	55 dBA	59 dBA	≥4 dBA	99%
3	人声噪声	55 dBA	59 dBA	≥4 dBA	99%

9.3.2 语音识别率测试

语音识别率：指当设备处于识别状态时，成功识别词表里包含的命令词的概率。

乐鑫语音识别率测试和结果

测试案例	噪声类型	噪声分贝	人声分贝	SNR	语音识别率
1	/	/	59 dBA	/	91.5%
2	白噪声	55 dBA	59 dBA	≥4 dBA	78.25%
3	人声噪声	55 dBA	59 dBA	≥4 dBA	82.77%

9.3.3 误唤醒率测试

误唤醒率：指设备在产品定义的应用场景下被非唤醒词成功唤醒的概率。

乐鑫误唤醒率测试和结果

测试案例	噪声类型	噪声分贝	测试时长	误唤醒次数
1	音乐	55 dBA	12 小时	1 次
2	新闻	55 dBA	12 小时	1 次

9.3.4 唤醒打断率测试

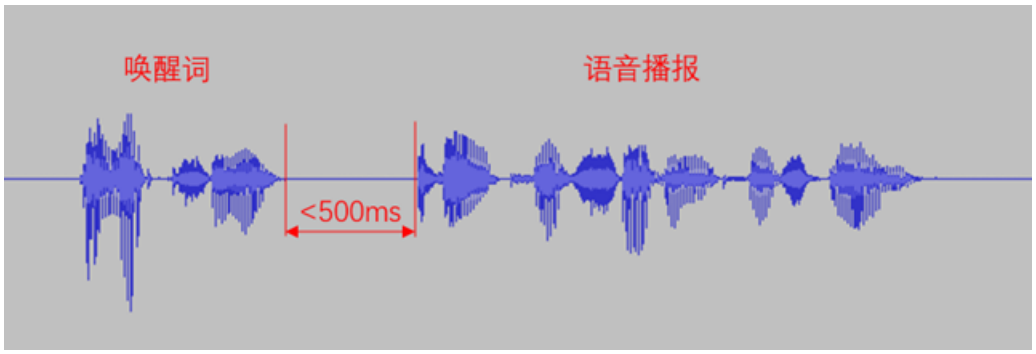
唤醒打断率：唤醒打断率是指设备有自噪时，即有 TTS3 播报或播放音频时，被唤醒成功的概率。对于有 AEC 功能的产品需要进行该测试。

乐鑫唤醒打断率测试和结果

测试案例	噪声类型	噪声 / 人声分贝	SNR	唤醒率	语音识别率
1	音乐	69 dBA / 59 dBA	≥10 dBA	100%	96%
2	TTS	69 dBA / 59 dBA	≥10 dBA	100%	96%

9.3.5 响应时间测试

响应时间：代表 DUT 响应语音命令的时间，具体测量为语音指令与播报之间的时间间隔（见下图）。



乐鑫响应时间测试和结果

测试案例	噪声 / 人声分贝	SNR	响应时间
1	NA / 59 dBA	/	< 500 ms

Chapter 10

术语表

10.1 通用术语

ESP-SR 仓库中的大多数术语均与 乐鑫高级开发框架 (ESP-ADF) 共用，具体请访问 ESP-ADF 中英术语库。

10.2 特别术语

ESP-SR 仓库独有术语，请见下方。

语音用户界面 (Voice-User Interface, VUI)