语音对讲播报器使用手册
（AW-S24FM）

V1.0



竭诚感谢您使用本公司的产品

本手册就产品的使用方法与安全事项进行说明

*熟读本手册，并在使用过程中注意安全。

*保留本手册，放在合适的地方以便随时查阅。

深圳市艾智威科技有限公司

目录

1 安全 1
1.1 安全警示标志 1
1.2 安全使用注意事项 1
2 产品概要 2
3 产品规格 2
3.1 规格参数 2
3.2 安装尺寸 3
4 使用方法 3
4.1 接线图 3
4.2 音频文件下载方法 3
4.3 功能流程图 4
4.4 播放控制 4
- 播报模式：存储音频播放和文本编码播放 4
- 对讲模式：双向对讲、单向对讲（设备->服务端/服务端->设备） 5
5 参数配置 7
5.1 设备连接 7
5.2 参数配置 7
5.3 配置工具参数说明 7
6 通信协议 8
6.1 AW-NTS通信协议 8
- 播放文件夹内音频：#[AWXXX] 9
- 调整播放音量：#[VLXX] 9
- 播放文本内容（TTS）：#[TTS]内容 9
- 获取播放状态：#[GET] 9
- 写入缓存：#[SAVEXX]内容 10
- 播放缓存：#[CHXX] 10
- 停止指令：#[STOP] 10
6.2 Modbus TCP协议播放 10
6.2.1 寄存器地址 10
6.2.2 通信功能码 11
6.3 对讲Demo(Python) 12
7 故障排除 19
8 售后服务 19
8.1 质保期限 19
8.2 质保范围 19
8.3 免责范围 19
8.4 技术支持 19
8.5 版本修订履历 19

1 安全

1.1 安全警示标志

以下安全警示标志，用以警告潜在的人身伤害危险，请务必遵从所有带 有此标志的安全信息，以避免可能的伤害。



警告 在操作时违反本警告事项所示的内容，可能会导致人员死亡或重伤。



注意 在操作时违反本注意事项所示的内容，可能会导致人员负伤或造成物品损坏。



提醒 在操作时使您能正确使用产品，所务必遵守的相关使用的事项。

1.2 安全使用注意事项

这里提示的注意事项，其目的是为了使您能安全、正确地使用产品，并防患于未然，以免给您和他人造成危害和损伤。请您对其内容充分理解以后再使用本产品。



注意

- 请勿在爆炸性气体环境、易燃性气体环境、腐蚀性环境、容易沾水的场所以及可燃物附近使用本产品，否则有可能引起产品故障或火灾与致伤。
- 为了防止火灾和触电的危险，请在额定电压下使用本产品。
- 请按照连接图进行连接，否则有可能引起火灾。
- 保养/检查请务必在切断电源后进行，否则有可能致伤。
- 请不要拆卸和改造本产品，否则有可能致伤或造成装置破损。要检查内部或修理时，请与本公司联系。



提醒

- 请不要在强电波和强电磁辐射的环境中使用本产品，扬声器有可能受到干扰。
- 在您安装各个部件时，请按照我们推荐的拧紧力矩进行安装。
- 请不要在振动超过规定范围的地方使用本产品。会导致产品故障和掉落的危险。
- 在室外使用的时候，请务必将导线的结线部进行绝缘处理，会造成故障和漏电。

2 产品概要

■ 采用高品质数字功放，支持WAV、MP3格式音频文件；
■ 采用稳定可靠NAND Flash存储芯片,容量128M；外置Type-C接口，可自行更换音频文件；
■ 支持RJ45(以太网)接口控制，支持标准Modbus TCP/http协议，可远程修改音频文件；
■ 支持TTS文本语音合成功能，可中英文混合合成，可支持多发音人；
■ 支持AI降噪拾音实现实时对讲；
■ DC12V~36V宽电压输入；
■ 防水设计，防水款防护等级可达IP67；
■ 符合RoHS标准，通过CE认证；
■ 可无级调节音量，最大音量达到115db（1米处），适用于各种嘈杂的工厂环境；
■ 适合用于AGV小车、AGV叉车、机床、自动化设备及自动化生产线的信号语音播报。

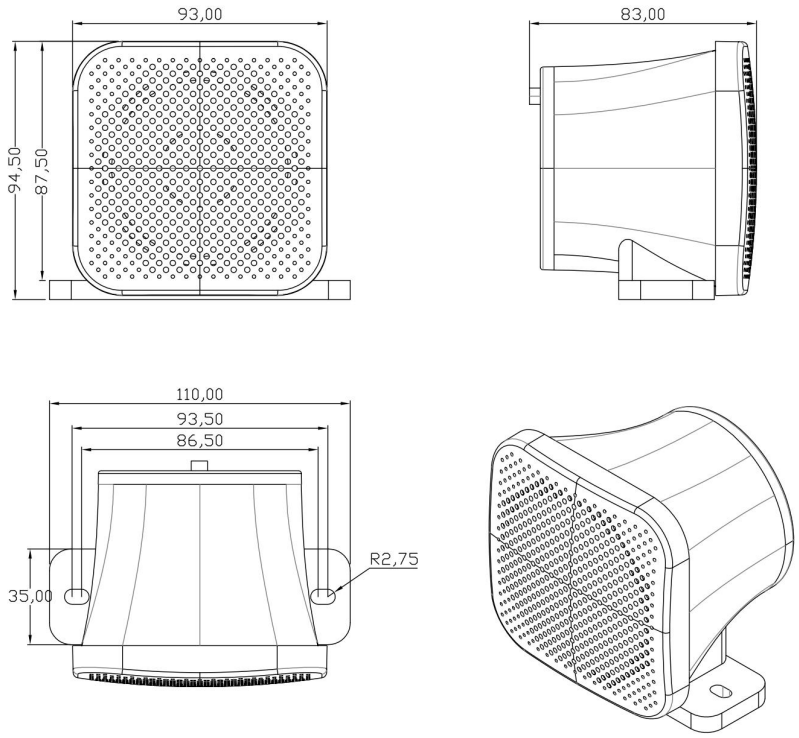
3 产品规格

3.1 规格参数

产品规格	AW-S24FM	AW-S24FM-T
音源类型	预存储音频文件+双向麦克风对讲	预存储音频文件+文本编码+双向麦克风对讲
控制方式	以太网	
配套线束	以太网口/0.5m + AUX/0.5m	
通道数量	255(可增)	
存储介质	NAND Flash	
存储容量	128MB（可定制大容量）	
播放模式	循环播放(默认)/单次播放可打断/单次播放不可打断/定制	
额定电压	DC 24V	
电压范围	DC 12V～36V	
产品音量	Max.115db（1m处），音量可调节 （音量随使用环境、电压及音频文件不同而不同）	
输出功率	10W	
外壳材质	ABS	
防护等级	防水款IP67	
使用环境温度	-40℃～60℃	
使用环境湿度	85%RH（无结露现象）	
重　　量	480g±10%	

注：
1、标准款型号为防水款。
2、不支持旋钮调音，仅支持指令修改音量。
3、AW-S24FM需自配麦克风，AUX接口麦克风。

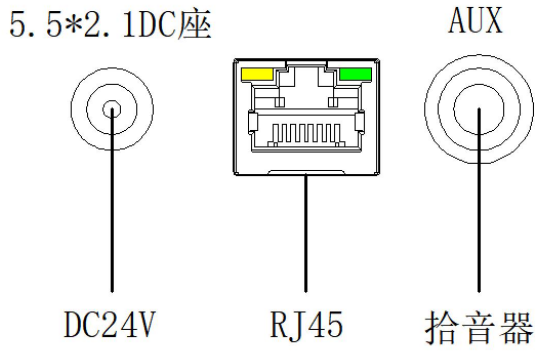
3.2 安装尺寸



4 使用方法

4.1 接线图

- **AW-S24FM/M-T 接线图：**



4.2 音频文件下载方法

本产品背部设有Type-C接口，使用USB数据线连接电脑，电脑即会显示一个可移动磁盘。可移动磁盘内可建多个文件夹，文件夹命名格式固定为“AW***”五位，每个文件夹对应一个音频通道，即AW001-AW255，分别对应第1通道到第255通道，将准备好的音频文件拷贝到对应的文件夹内即可。音频格式支持WAV、MP3格式。

提醒

- 此方法仅适用于音源类型为预存储音频文件的产品。
- USB数据线须确定带数据线芯，否则无法识别。
- 拷贝完资料后请按windows提示正确断开连接，以免造成储存卡损坏或数据丢失。

4.3 功能流程图



4.4 播放控制

设备上电默认为播报模式。

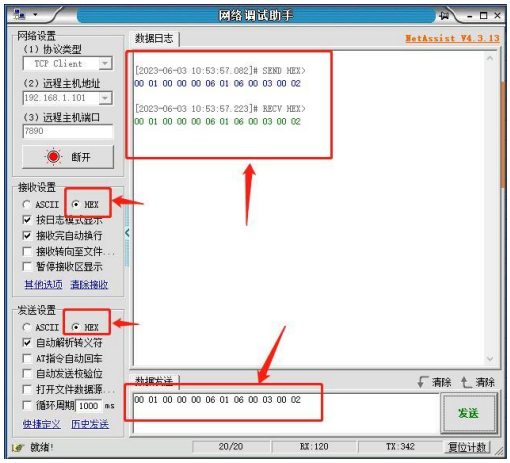
- **播报模式：存储音频播放和文本编码播放**

✧ **Modbus TCP播放**

示例：上位机将寄存器【03】写入数据00 02，即播放文件夹AW002内音频。

发送：00 01 00 00 00 06 01 06 00 03 00 02

返回：00 01 00 00 00 06 01 06 00 03 00 02

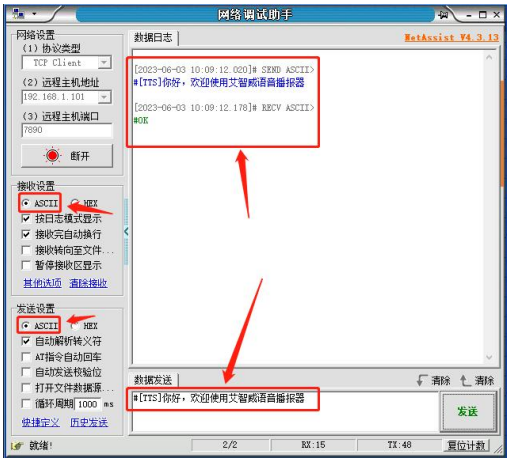


✧ **TTS播放**

示例：播放“你好，欢迎使用艾智威语音播报器”。

发送：#[TTS]你好，欢迎使用艾智威语音播报器

返回：“#OK”

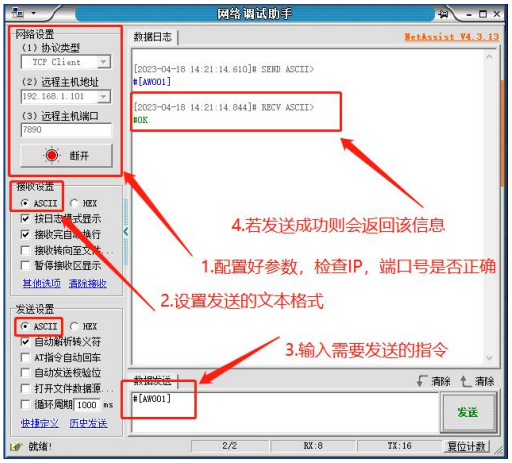


✧ **NTS协议：文件夹播放**

示例1：播放“AW001”文件夹内音频文件。

发送：#[AW001]

返回：“#OK”



➢ **对讲模式：双向对讲、单向对讲（设备->服务端/服务端->设备）**

✧ **切换双向对讲模式**

发送：41 57 03 03 9d 53 59 53

返回：41 57 03 FE 98 53 59 53



✧ **切换单向对讲模式（设备->服务端）**

发送：41 57 03 02 9c 53 59 53

返回：41 57 03 FE 98 53 59 53



✧ 切换单向对讲模式（设备->服务端）

发送：41 57 03 01 9b 53 59 53

返回：41 57 03 FE 98 53 59 53



✧ 切换播报模式

发送：41 57 04 00 9b 53 59 53

返回：41 57 04 01 9C 53 59 53



5 参数配置

5.1 设备连接

用USB数据线将语音播报器的Type-C口与电脑USB连接，打开“艾智威播报器配置工具.exe”(以下统称配置工具)，启动配置工具。



5.2 参数配置

A. 打开配置工具，左下角显示设备已连接，左上角选择对应的型号（默认自动选择），右侧显示该型号的可配置参数，如图5.2-A所示。

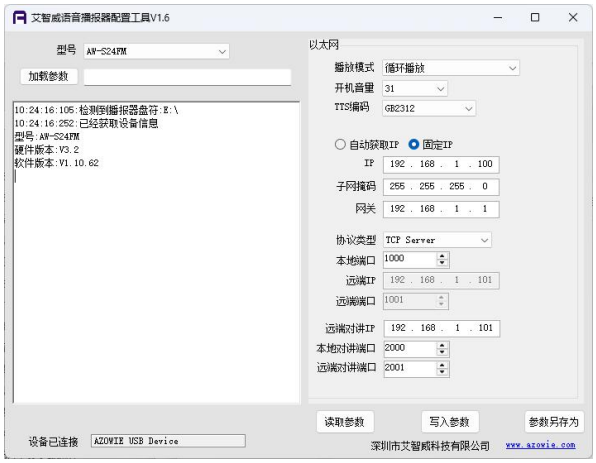


图5.2-A

B. 根据需求修改对应的参数后，点击写入参数，待左侧状态栏显示写入参数成功后，即修改配置完成，如图5.2-B所示

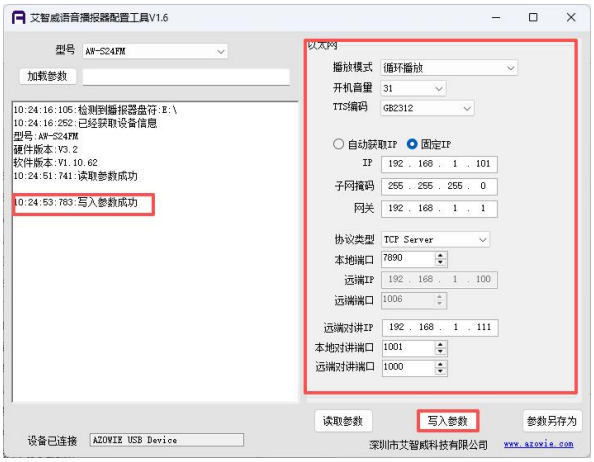


图5.2-B

C. 如需批量操作，可待右侧的参数修改完成后，点击 “参数另存为”，将修改的参数另存为awdc文件，可在下次配置时通过点击“加载参数”，选择awdc文件后，写入参数即可。

注意：写入参数以配置工具右侧的具体数据为准写入。

5.3 配置工具参数说明

参数名称		参数说明	
播报参数	播放模式	**循环播放**：播报器在接收到上位机发送的播报指令后，循环播报当前指令对应文件夹内的音频或对应文本内容。若播报器在播报期间接收到新的播报指令，会立即结束当前播报，执行新的播放指令。 **单次播放（可打断）**：播报器在接收到上位机发送的播报指令后，播报当前指令对应文件夹内的音频或对应文本内容，播报完一次即停止播报。若播报器在播报期间接收到新的播报指令，会立即结束当前播报，执行新的播放指令。 **单次播放（不可打断）**：播报器在接收到上位机发送的播报指令后，播报当前指令对应文件夹内的音频或对应文本内容，播报完一次即停止播报。若在播报期间接收到上位机发送的新指令，则需等待当前指令播报完成后，才执行新的播报指令。若播报期间收到多条指令，则播报器只会记录最后一条指令。	
	开机音量	0-31；开机音量范围0-31，写入参数后，开机音量会写入到寄存器4（见章节6 通信协议）	
	TTS编码	选择TTS播放发送的编码类型	
	自动获取IP	设备的IP地址由局域网自动分配	
	固定IP	设备的IP地址手动设置	
	IP	如果选择固定IP，需要根据局域网参数手动填写设备IP地址、网关、子网掩码，确保设备IP地址符合网段要求。	
	子网掩码		
	网关		
	协议类型 切换模式及播报模式使用	TCP Client客户端	设备工作在TCP Client模式，上位机系统工作在TCP Server模式，需配置远端IP、远端端口
		TCP Server服务器	设备工作在TCP Server模式，上位机系统工作在TCP Client模式，需配本地端口
		UDP	设备和上位机系统都工作在UDP模式，需配本地端口（设备的端口）和远端IP、远端端口
	本地端口	设备做 TCP Server 或 UDP 时设置（切换模式及播报模式使用）	
	远端IP	设备做 TCP Client 或 UDP 时设置（切换模式及播报模式使用）	
	远程端口	设备做 TCP Client 或 UDP 时设置（切换模式及播报模式使用）	
	远端对讲IP	设备对讲时连接的远端对讲IP地址（与设备对讲的服务器IP）	
	本地对讲端口	设备对讲时本机的对讲端口（本机的对讲端口）	
	远端对讲端口	设备对讲时连接的远端对讲端口（与设备对讲的服务器端口）	

6 通信协议

6.1 AW-NTS通信协议

指令格式	说明	范围	返回值
#[AWXXX]	播放文件夹内音频	#[AW001]～#[AW999]	#OK
#[VLXX]	调整播放音量	#[VL00]～#[VL10]	#OK
#[TTS]内容	播放文本内容（TTS）	1～255字节	#OK
#[GET]	获取播放状态	-	#PLAY / #STOP
#[SAVEXX]内容	写入缓存	#[SAVE01]～#[SAVE16]	#OK
#[CHXX]	播放缓存	#[CH01]～#[CH16]	#OK
#[STOP]	停止播放	-	#OK

➢ 播放文件夹内音频：#[AWXXX]

播放指令格式对应表：

播报指令格式	播放对应文件夹编号
#[AW001]	AW001
#[AW002]	AW002
...	...
#[AW999]	AW999

■ 示例：播放AW004文件夹内音频

发送：#[AW004]

返回：#OK

➢ 调整播放音量：#[VLXX]

音量调节指令格式对应表：

音量调节指令格式	对应音量等级
#[VL00]	00
#[VL01]	01
...	...
#[VL10]	10

注：音量指令与播放指令可组合使用，但播放指令须在前。

■ 示例1：音量调节为1

发送：#[VL01]

返回：#OK

■ 示例2：播放AW001文件夹内音频，音量调节为1

发送：#[AW001][VL01]

返回：#OK

■ 示例3：播放文本内容为“你好，欢迎使用艾智威语音播报器”，音量调节为10

发送：#[TTS][VL10]你好，欢迎使用艾智威语音播报器

返回：#OK

■ 示例4：播放第16缓存，音量调节为5

发送：#[CH16][VL05]

返回：#OK

➢ 播放文本内容（TTS）：#[TTS]内容

■ 示例：播报“你好,欢迎使用艾智威语音播报器”

发送：#[TTS]你好,欢迎使用艾智威语音播报器

返回：#OK

➢ 获取播放状态：#[GET]

该指令获取播报状态回复的内容有：#PLAY：正在播报； #STOP：空闲状态/停止播报。

发送：#[GET]

返回：#PLAY 或 #STOP

➢ 写入缓存：#[SAVEXX]内容

写入缓存指令格式对应表：

写入缓存指令格式	对应缓存序号
#[SAVE01]内容	01
#[SAVE02]内容	02
#[SAVE03]内容	03
...	...
#[SAVE16]内容	16

■ 示例1：将“你好,欢迎使用艾智威语音播报器”写入缓存1

发送：#[SAVE01]你好,欢迎使用艾智威语音播报器

返回：#OK

➢ 播放缓存：#[CHXX]

播放缓存指令格式对应表：

播放缓存指令格式	对应缓存序号
#[CH01]	01
#[CH02]	02
#[CH03]	03
...	...
#[CH16]	16

■ 示例1：播放第16缓存

发送：#[CH16]

返回：#OK

➢ 停止指令：#[STOP]

发送：#[STOP]

返回：#OK

发送该指令，则播报器便会停止播放。

6.2 Modbus TCP协议播放

注：Modbus TCP 协议报文内容为十六进制格式（HEX）

AW-S24EN系列播报器使用标准的Modbus TCP协议，支持功能码 03，06，10（十六进制）。

6.2.1 寄存器地址

播报器仅工作在 slave 模式，Modbus TCP 寄存器地址表如下:

寄存器地址（十进制）	功能	取值范围（十进制）	读写权限	备注
1	程序版本	-	只读	
2	播放状态	0～1	只读	0停止，1播放
3	音频文件夹编号	0～999	读写	0停止播放，1～999播放对应的AW001～AW999文件夹
4	播放音量	0～10	读写	0～10
5～8	备用	-	-	-
9	播报器编号	0～65535	只读	播报器编号

6.2.2 通信功能码

➢ **功能码 03：读取多个寄存器数据**

请求报文：

编号	名称	长度	描述
0	通信帧号	2	此部分数据参考帧基本格式
1			
2	协议类型	2	
3			
4	长度	2	
5			
6	设备地址	1	
7	功能码	1	03，读取多个寄存器数据
8	起始地址	2	读取目标数据的起始地址
9			
10	数量	2	读取目标寄存器的数量
11			

应答报文：

编号	名称	长度	描述
0	通信帧号	2	此部分数据参考帧基本格式
1			
2	协议类型	2	
3			
4	长度	2	
5			
6	设备地址	1	
7	功能码	1	03，读取多个寄存器数据
8	字节个数	1	后面数据的总字节个数(m)，m = 2 * n
9-N	数据	m	数据，每两字节表示一个地址数据，高字节在前

■ 示例：上位机读寄存器【1】、【2】、【3】的值，即获取播报器的程序版本、播放状态、音频文件夹编号，起始地址是00 01，寄存器数量是00 03，如图7.2.3.1所示。

发送：00 01 00 00 00 06 01 03 00 01 00 03

返回：00 01 00 00 00 09 01 03 06 00 68 00 01 00 01

说明：读取到的数据长度为00 09，字节个数为03，程序版本为00 68，00 01表示正在播放，00 01播放文件夹AW001。

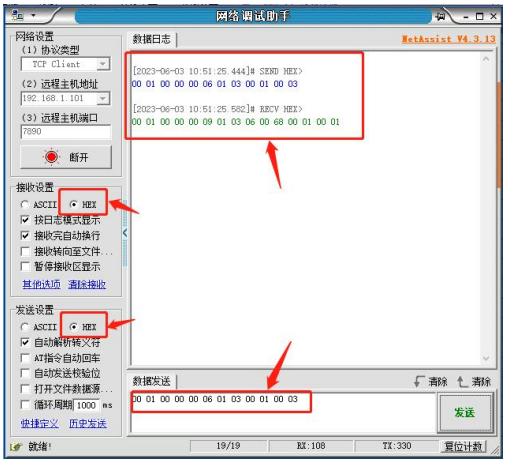


图6.3.2.3-A 功能码03读取多个寄存器数据

➢ **功能码 06：写入单个寄存器数据**

请求报文：

编号	名称	长度	描述
0	通信帧号	2	此部分数据参考帧基本格式
1			
2	协议类型	2	
3			
4	长度	2	
5			
6	设备地址	1	
7	功能码	1	06，写入单个寄存器数据
8	寄存器地址	2	写入目标数据的地址
9			
10	数据	2	数据

应答报文：同请求报文。

■ 示例1：上位机将寄存器【03】写入数据00 02，即播放文件夹AW002内音频，如图7.2.3.2所示。

发送：00 01 00 00 00 06 01 06 00 03 00 02

返回：00 01 00 00 00 06 01 06 00 03 00 02

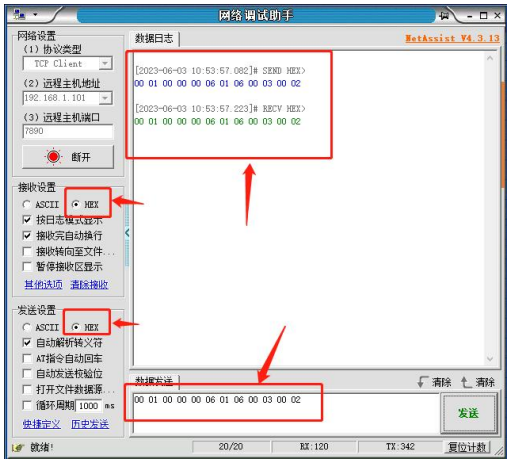


图6.3.2.3-B 功能码06写入单个数据

6.3 对讲Demo(Python)

6.3.1 概述

本DEMO是一个基于 Python 开发的实时语音对讲工具。它利用 UDP 协议进行双向音频传输，具备低延迟、轻量化的特点。通过 Tkinter 提供的图形用户界面（GUI），用户可以方便地配置网络参数并启动对讲。

6.3.2 核心功能及实现原理

库支持：使用 PyAudio 库。

采样参数：

采样率：8000 Hz。

通道：单声道（Mono）。

位深：16-bit (paInt16)。

包大小：512 字节。

6.3.2 网络传输

协议：UDP（User Datagram Protocol）。

抖动缓冲：使用 queue.Queue 实现了一个深度为 5 的音频数据队列，过滤网络抖动，保证播放平滑性。

超时控制：Socket 设置了 0.5s 的超时时间，避免接收阻塞导致界面无响应。

6.3.3 安装要求

由于涉及音频硬件驱动，请确保已安装以下依赖：

```
pip install pyaudio
```

6.3.4 使用说明

- **界面配置**

运行程序后，在主界面配置以下参数：

本地端口：本程序监听音频流入的端口（即播报器设置的远端对讲端口）。

目标 IP：播报器的 IP 地址。

目标端口：播报器设置的对讲端口。

- **建立通话**

确认双方的网络参数填写正确（通常本地端口与对方的目标端口需对应）。

点击 “启动连接” 按钮。状态栏显示“已连接”后，即可开始实时对讲。

按下 “断开连接” 可停止音频流并释放硬件资源。

6.3.5 源码结构说明

UDPVoiceChat 类：负责音频 I/O 和 UDP 套接字管理。

__init__：初始化音频流和线程。

send_audio：读取麦克风数据并发送。

receive_audio：接收 UDP 数据包并推入缓冲队列。

play_audio：从队列中提取音频并输出到扬声器。

APP 类：基于 Tkinter 的窗体布局和逻辑绑定。

6.3.6 注意事项

防火墙：使用时请确保防火墙已放行所选的 UDP 端口。

音量反馈：若在同一环境下测试，请降低音量或佩戴耳机，以防止产生音频回馈啸叫。

以下为Demo源码：

```
import socket
import threading
import pyaudio
import tkinter as tk
from tkinter import ttk


import queue


class UDPVoiceChat:
    def __init__(self, server_ip, server_port, listen_port):
        # 初始化网络和音频参数
        self.server_ip = server_ip
        self.server_port = server_port
        self.running = threading.Event()
        self.running.set()
        self._closed = False
        self.threads = []
```

```
        # 音频参数配置
        self.chunk_size = 256
        self.audio_format = pyaudio.paInt16
        self.channels = 1
        self.rate = 8000

        self.p = pyaudio.PyAudio()

        # 新增音频数据队列（最多缓存50个数据包）
        self.audio_queue = queue.Queue(maxsize=5)

        try:
            self.input_stream = self.p.open(
                format=self.audio_format,
                channels=self.channels,
                rate=self.rate,
                input=True,
                frames_per_buffer=self.chunk_size
            )

            self.output_stream = self.p.open(
                format=self.audio_format,
                channels=self.channels,
                rate=self.rate,
                output=True,
                frames_per_buffer=self.chunk_size
            )

            # 配置UDP套接字
            self.socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
            self.socket.bind(('0.0.0.0', listen_port))
            self.socket.settimeout(0.5)

            # 启动线程
            self.receive_thread = threading.Thread(target=self.receive_audio, name="receive-audio", daemon=True)
            self.send_thread = threading.Thread(target=self.send_audio, name="send-audio", daemon=True)
            self.play_thread = threading.Thread(target=self.play_audio, name="play-audio", daemon=True)
            self.threads = [self.receive_thread, self.send_thread, self.play_thread]

            for thread in self.threads:
                thread.start()

        except Exception as e:
            self.close()
            raise RuntimeError(f"初始化失败: {str(e)}")
```

```
def send_audio(self):
    while self.running.is_set():
        try:
            data = self.input_stream.read(self.chunk_size, exception_on_overflow=False)
            self.socket.sendto(data, (self.server_ip, self.server_port))
        except Exception as e:
            if self.running.is_set():
                print(f'发送错误: {e}')
            break

def receive_audio(self):
    while self.running.is_set():
        try:
            data, _ = self.socket.recvfrom(2048)
            if data:
                if self.audio_queue.full():
                    try:
                        self.audio_queue.get_nowait()
                    except queue.Empty:
                        pass
                self.audio_queue.put_nowait(data)
        except socket.timeout:
            continue
        except queue.Full:
            continue
        except Exception as e:
            if self.running.is_set():
                print(f'接收错误: {e}')
            break

def play_audio(self):
    while self.running.is_set():
        try:
            data = self.audio_queue.get(timeout=0.1)
            self.output_stream.write(data)
        except queue.Empty:
            continue
        except Exception as e:
            if self.running.is_set():
                print(f'播放错误: {e}')
            break

def _close_stream(self, stream):
    if stream is None:
        return
    try:
        if stream.is_active():
            stream.stop_stream()
```

```
            except Exception:
                pass
            try:
                stream.close()
            except Exception:
                pass


    def close(self):
        if self._closed:
            return


        self._closed = True
        self.running.clear()


        while not self.audio_queue.empty():
            try:
                self.audio_queue.get_nowait()
            except queue.Empty:
                break


        socket_obj = getattr(self, 'socket', None)
        if socket_obj:
            try:
                socket_obj.close()
            except Exception:
                pass


        for thread in self.threads:
            if thread.is_alive() and thread is not threading.current_thread():
                thread.join(timeout=1)


        if hasattr(self, 'input_stream'):
            self._close_stream(self.input_stream)
        if hasattr(self, 'output_stream'):
            self._close_stream(self.output_stream)
        if hasattr(self, 'p'):
            self.p.terminate()
class APP:
    def __init__(self):
        self.win = tk.Tk()
        self.win.title("语音对讲DEMO v2.0")
        self.win.geometry("500x250")
        self.voice_chat = None


        self._create_widgets()


        # 窗口关闭事件绑定
        self.win.protocol("WM_DELETE_WINDOW", self.on_close)
```

```
def _create_widgets(self):
    """创建界面控件"""
    main_frame = ttk.Frame(self.win, padding=20)
    main_frame.pack(fill=tk.BOTH, expand=True)

    ttk.Label(main_frame, text="本地端口").grid(row=0, column=0, sticky=tk.W, pady=8)
    self.local_port_entry = ttk.Entry(main_frame, width=15)
    self.local_port_entry.insert(0, "2001")
    self.local_port_entry.grid(row=0, column=1, sticky=tk.W, pady=8)

    ttk.Label(main_frame, text="目标IP").grid(row=1, column=0, sticky=tk.W, pady=8)
    self.target_ip_entry = ttk.Entry(main_frame, width=25)
    self.target_ip_entry.insert(0, "192.168.1.101")
    self.target_ip_entry.grid(row=1, column=1, sticky=tk.W, pady=8)

    ttk.Label(main_frame, text="目标端口").grid(row=2, column=0, sticky=tk.W, pady=8)
    self.target_port_entry = ttk.Entry(main_frame, width=15)
    self.target_port_entry.insert(0, "2000")
    self.target_port_entry.grid(row=2, column=1, sticky=tk.W, pady=8)

    self.connect_btn = ttk.Button(main_frame, text="启动连接", command=self.toggle_connection)
    self.connect_btn.grid(row=1, column=2, rowspan=2, padx=(20, 0))

    self.status_var = tk.StringVar(value="未连接")
    ttk.Label(main_frame, text="状态:").grid(row=3, column=0, sticky=tk.W, pady=(16, 8))
    ttk.Label(main_frame, textvariable=self.status_var).grid(row=3, column=1, sticky=tk.W, pady=(16, 8))

    ttk.Button(main_frame, text="退出", command=self.on_close).grid(row=4, column=2, sticky=tk.E, pady=(24, 0))

    main_frame.columnconfigure(1, weight=1)

def _set_inputs_state(self, state):
    self.local_port_entry.config(state=state)
    self.target_ip_entry.config(state=state)
    self.target_port_entry.config(state=state)

def toggle_connection(self):
    """连接/断开连接切换"""
    if self.voice_chat is None:
        self.connect()
    else:
        self.disconnect()

def connect(self):
    """建立语音连接"""
    try:
        target_ip = self.target_ip_entry.get()
```

```
            target_port = int(self.target_port_entry.get())
            local_port = int(self.local_port_entry.get())

            if not (0 < target_port < 65536 and 0 < local_port < 65536):
                raise ValueError

            self.voice_chat = UDPVoiceChat(target_ip, target_port, local_port)
            self.status_var.set("已连接")
            self.connect_btn.config(text="断开连接")
            self._set_inputs_state('disabled')

        except ValueError:
            self.status_var.set("端口必须是 1-65535 的数字!")
        except Exception as e:
            self.status_var.set(f"连接失败: {str(e)}")
            if self.voice_chat:
                self.voice_chat.close()
            self.voice_chat = None

    def disconnect(self):
        """断开语音连接"""
        if self.voice_chat:
            self.voice_chat.close()
            self.voice_chat = None
        self.status_var.set("已断开")
        self.connect_btn.config(text="启动连接")
        self._set_inputs_state('normal')

    def on_close(self):
        """窗口关闭事件处理"""
        self.disconnect()
        self.win.destroy()


if __name__ == "__main__":
    app = APP()
    app.win.mainloop()
```

7 故障排除

故障现象	检查和确认	处理措施
不能播放	电源是否正常接通	确认传输信号线是否配线正确
	信号线是否正确配线	确认传输信号线是否配线正确
	串口及指令是否正确	确认串口是否正常，指令是否正确
	USB线是否断开	确认USB线有断开，断开后重新上电测试
	存储器内是否有对应音频文件	确认存储器内音频文件
	音量是否为最小状态	调整音量大小
播放与指定不同的语音信息	编码格式是否正确	确认编码格式是否正确
	串口指令是否正确	确认指令是否正确
	存储器内音频文件是否配置正确	确认存储器内音频文件是否正确
储存器无法识别	电源是否断开	确认电源有断开，断开后重新连接USB
	USB数据线是否正常	检查USB数据线是否有数据传输功能
	播放器与电脑USB端口是否正常	检查播放器与电脑USB端口是否正常

8 售后服务

8.1 质保期限

语音播报器提供限期质保，质保期间因产品本身质量问题，设计缺陷等原因造成的不能正常使用的，我们将免费进行售后维护。

■保固期：自出售起 1 年

8.2 质保范围

产品在质保期内，属于质保条件范围内的，我们将免费进行维修或者更换。

■本产品质保及售后只限在中国大陆境内；

■由于运输途中造成的产品开箱无法正常使用；

■产品本身元器件损坏造成的无法正常工作；

■产品设计缺陷造成无法正常使用。

8.3 免责范围

产品在使用过程中请注意以下条件范围内，我们将不提供无偿售后和质保。

■未正确按照说明书安装产品，造成产品损坏；

■在不适合的环境和条件下使用本产品，造成产品损坏；

■因不按产品说明书规范操作导致产品的损坏；

■未经本公司允许，擅自拆解或维修产品；

■自然灾害，火灾等不可抗拒的外界力造成的产品损坏。

8.4 技术支持

服务网址：www.azowie.com

服务电话：13128762505

服务微信：AZW13128762505

服务邮箱：zhaox@azowie.cn

8.5 版本修订履历

版本号	修订日期	修订作者	修订内容
V1.0	2026/05/23	Zyh	初版