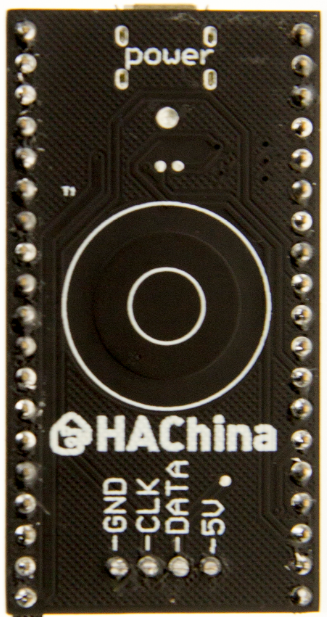
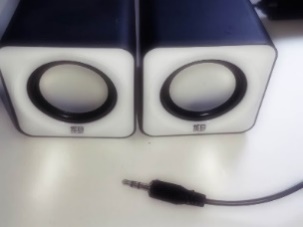
声音信号的采集与播放

【必要硬件】

1. nodemcu-32s
2. 麦克风扩展板(<https://lw.hachina.io/>)
3. 耳机或耳机插口有源音箱

【操作步骤】

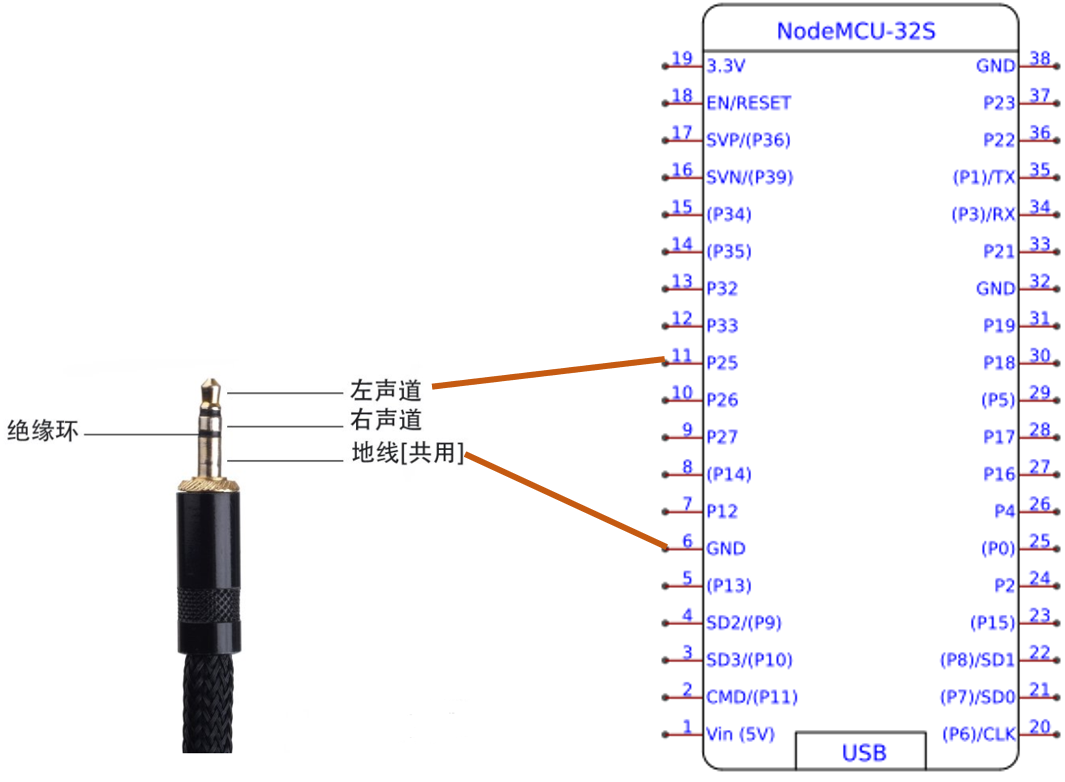
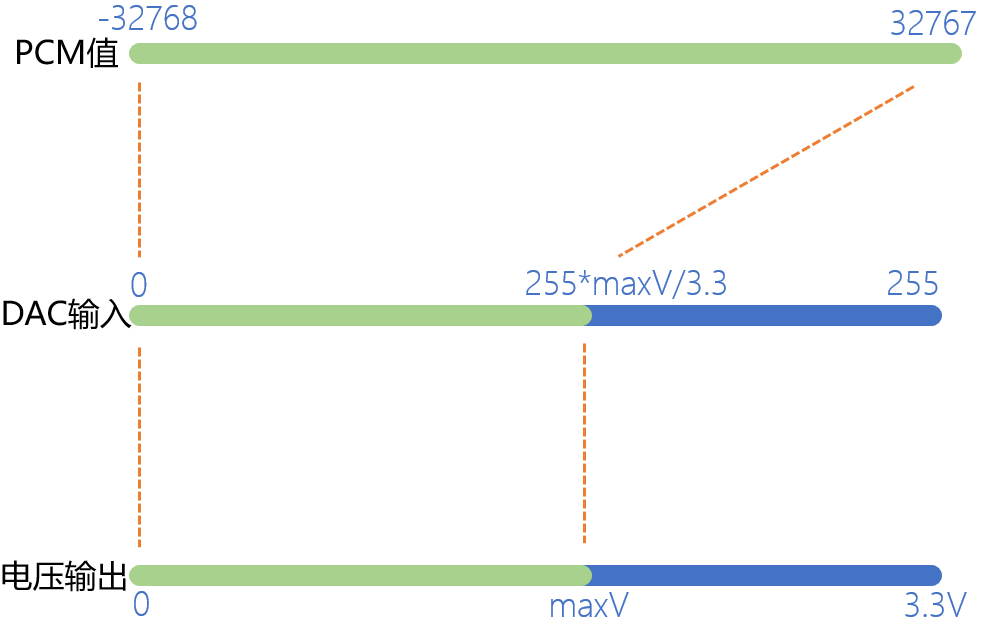
1. 在Arduino环境中观察声音输入与波形



Arduino中麦克风信号采集程序

<https://github.com/zhujisheng/Home-Assistant-DIY/blob/master/%E5%8F%82%E8%80%83%E6%96%87%E6%A1%A3%EF%BC%8821-30%EF%BC%89/example_24_audio_input/example_24_audio_input.ino>

1. 将PCM信号转化为模拟信号输出给喇叭

Arduino中PCM信号播放程序

<https://github.com/zhujisheng/Home-Assistant-DIY/blob/master/%E5%8F%82%E8%80%83%E6%96%87%E6%A1%A3%EF%BC%8821-30%EF%BC%89/example_24_audio_output/example_24_audio_output.ino>

【参考】

* 课程视频《Arduino与ESP硬件》

接入HomeAssistant的远程麦克风

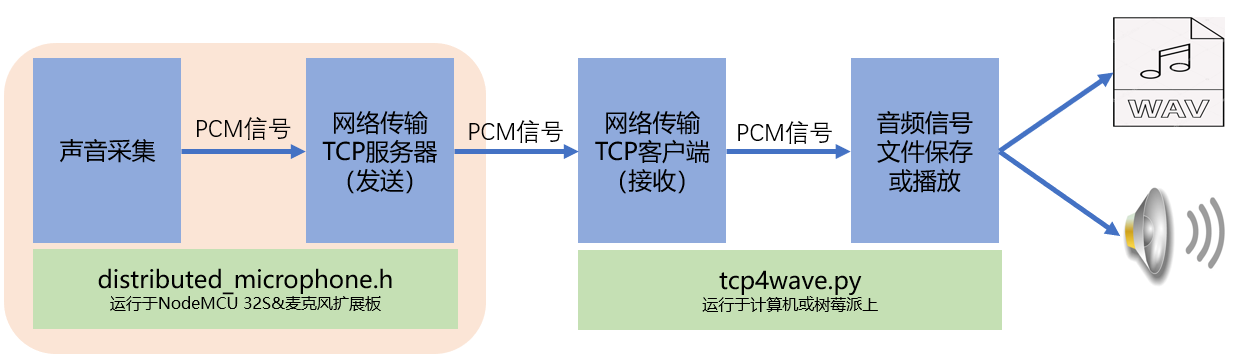
【操作步骤】

1. 硬件连接

1. 在ESPHome中接入自定义的分布式麦克风程序
   1. 下载distributed\_microphone.h

<https://raw.githubusercontent.com/zhujisheng/audio-reactive-led-strip/master/DistributedMicrophone/distributed_microphone.h>



* 1. 生成ESPHome的yaml配置文件
  2. 编译、UPLOAD

1. 使用nc进行连接测试
2. 接入HomeAssistant进行控制

【参考】

* ESPHome操作视频
  + 《ESPHome——不编程，集成ESP8266》
  + 《使用NFC识别不同的ID卡》
  + 《音乐灯带》
* 参考配置

esphome:

name: distributed\_microphone

platform: ESP32

board: nodemcu-32s

includes:

- distributed\_microphone.h

......

switch:

- platform: custom

lambda: |-

auto my\_custom\_switch = new MicrophoneSwitch();

App.register\_component(my\_custom\_switch);

return {my\_custom\_switch};

switches:

name: "RemoteMic"

# a LED on GPIO17, optional for indicating microphone's state

output:

- platform: ledc

id: mic\_led

pin: GPIO17

light:

- platform: monochromatic

name: "MicLight"

output: mic\_led

default\_transition\_length: 0s

effects:

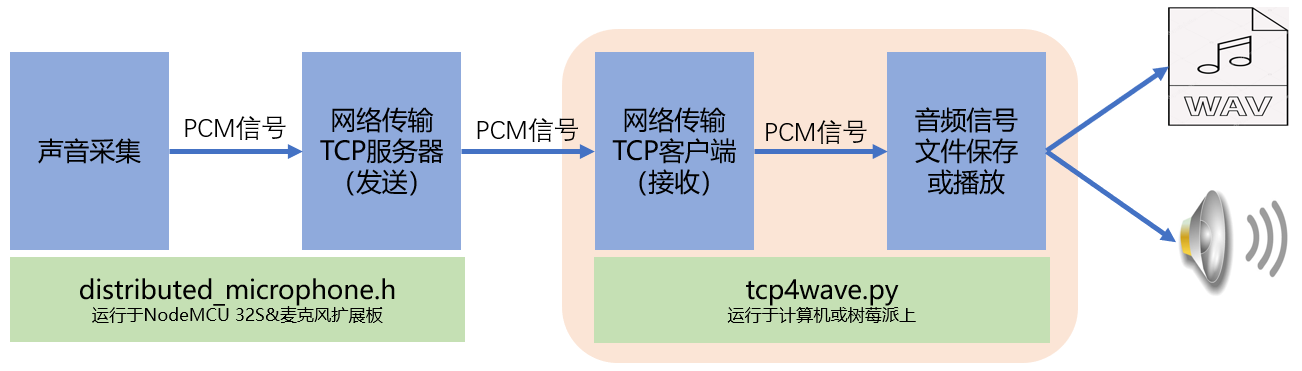
- strobe:

* ESPHome中自定义开关组件

<https://esphome.io/components/switch/custom.html>

使用远程麦克风——监听与录音

【架构】



【操作步骤】

1. 安装pyaudio库  
   pip install pyaudio或sudo pip3 install pyaudio（linux下python3环境）
2. 下载tcp4wave.py程序

<https://github.com/zhujisheng/audio-reactive-led-strip/blob/master/DistributedMicrophone/tcp4wave.py>

1. 运行tcp4wave.py
2. 解释tcp4wave.py

【参考】

* pyaudio库

<https://people.csail.mit.edu/hubert/pyaudio/>

* wave库

<https://docs.python.org/zh-cn/3/library/wave.html>

给智能音箱配上远程麦克风

【操作步骤】

1. 以前智能音箱必要环节回顾
2. 本地麦克风
3. SpeechRecongnition库
4. Snowboy唤醒词服务
5. 具有tts和媒体播放能力的HomeAssistant
6. 配置HomeAssistant的conversation和intent\_script组件
7. 图灵机器人的API服务
8. 阅读新的voice\_assistant.py程序

<https://github.com/zhujisheng/audio-reactive-led-strip/blob/master/DistributedMicrophone/%E6%99%BA%E8%83%BD%E9%9F%B3%E7%AE%B1/voice_assistant.py>

<https://github.com/zhujisheng/audio-reactive-led-strip/blob/master/DistributedMicrophone/%E6%99%BA%E8%83%BD%E9%9F%B3%E7%AE%B1/tcp_mic.py>

1. 程序下载与配置
2. 演示

【参考】

* REMOTE\_MIC\_CONFIG配置说明
  + 当没有远程麦克风时，配置为：

REMOTE\_MIC\_CONFIG = []

* + 当有多个远程麦克风时，配置为：

REMOTE\_MIC\_CONFIG = [('IP1',port1, RemoteMicCB1), ('IP2',port2, RemoteMicCB2),…]

其中，IP和port是远程麦克风的IP地址与端口号，RemoteMicCB是状态指示函数

RemoteMicCB(0) 在进入等待唤醒时被调用

RemoteMicCB(1) 在被唤醒后被调用

RemoteMicCB(2) 在输入语音指令后被调用

* LOCAL\_MIC\_CONFIG配置说明

LOCAL\_MIC\_CONFIG=True 使用本地麦克风

LOCAL\_MIC\_CONFIG=False 不使用本地麦克风