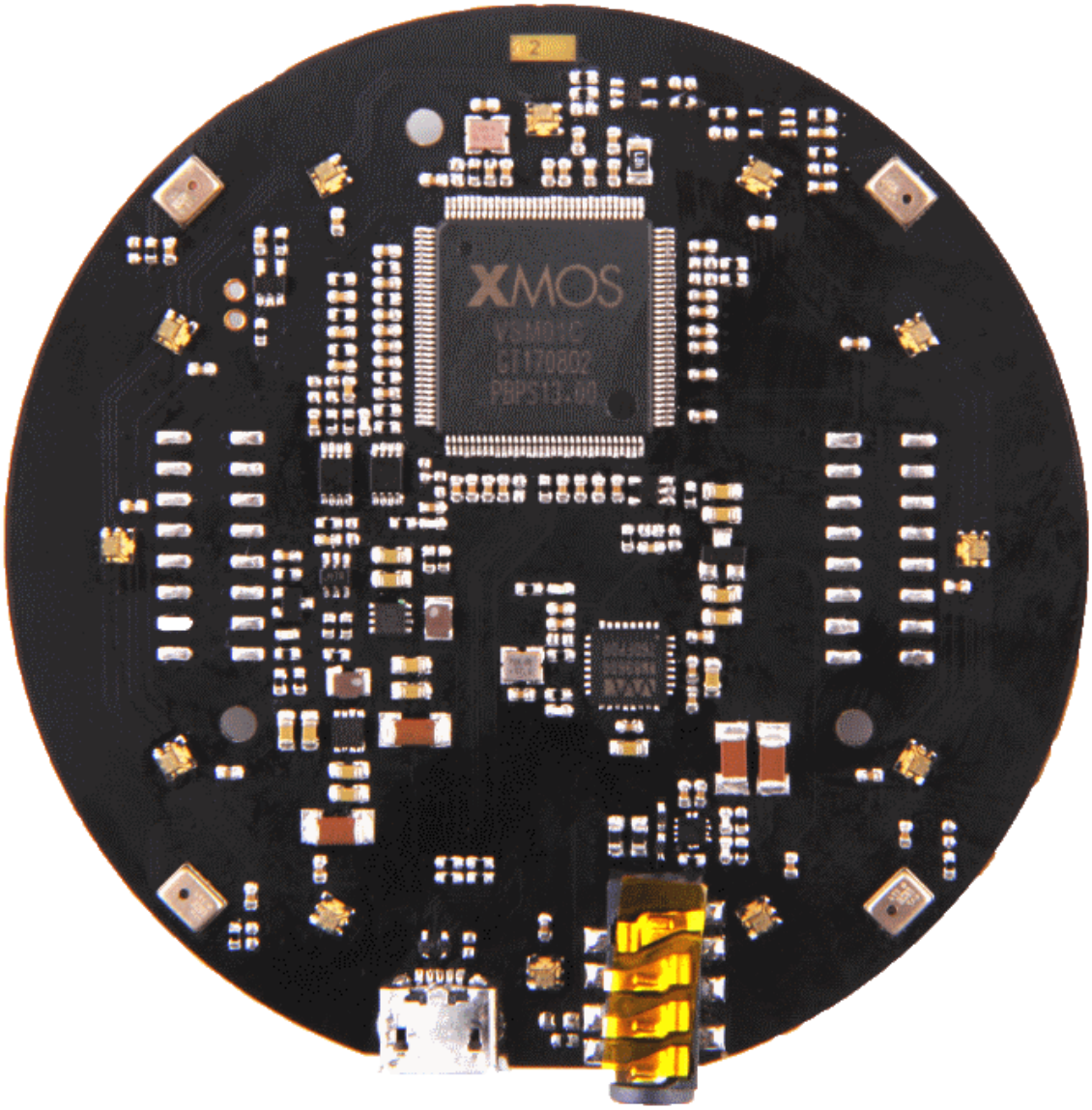
ReSpeaker Mic Array v2.0 SKU: 107990053



Seeed 的 ReSpeaker Mic Array v2.0 是基于 XVSM-2000 的 ReSpeaker Mic Array v1.0 的升级版本。v2.0 是基于 XMOS 的 XVF-3000 开发。它可以堆叠 (连接) 到 ReSpeaker Core 的顶部，显着改善语音交互体验。该主板集成了 4 个 PDM 麦克风，有助于将 ReSpeaker 的声学 DSP 性能提高到更高的水平。

ReSpeaker Mic Array v2.0 直接支持 USB Audio Class 1.0 (UAC 1.0)。所有主流的操作系统，如 Windows，MacOS，Linux 都与 UAC 1.0 兼容，因此它可以作为声卡运行而无需 ReSpeaker Core，但具有语音算法。

ReSpeaker Mic Array v2.0 有两个固件，一个包含语音算法，另一个仅捕获用于特殊用途的原始语音数据。

版本日志

产品版本	版本介绍	发布日期

产品版本	版本介绍	发布日期
ReSpeaker Mic Array v1.0	初始版本	2016 年 8 月 15 日
ReSpeaker Mic Array v2.0	MCU 更换为 XVF-3000，Mic 从 7 减少到 4	2018 年 1 月 25 日

产品特性

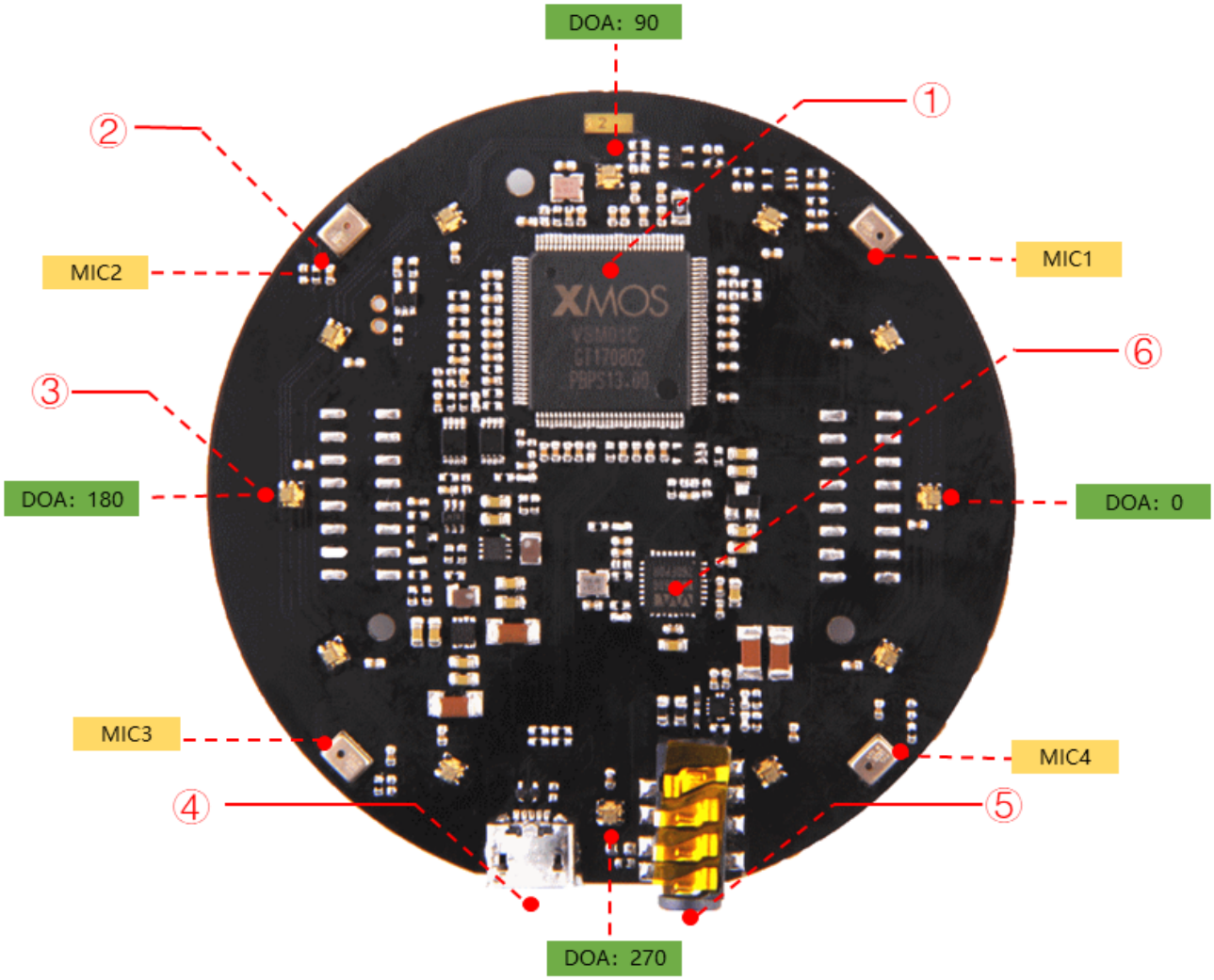
- 远场语音捕获
- UAC 1.0
- 四麦阵列
- 12 个可编程 RGB LED 指示灯
- 语音算法和功能
- 语音活动检测 Voice Activity Detection
- DOA
- 波束成形
- 噪声抑制
- 消混响
- 声学回声消除

规格参数

项	参数
16 内核 XVF-3000	2 个 xCore 磁贴上的 16 个实时逻辑核心
	dual issue 模式下内核共享最高可达 2400 MIPS
	512KB 内部单周期 SRAM 和 2MB 内置闪存
	16KB 内部 OTP (每块最大 8KB)
	USB PHY，完全符合 USB 2.0 规范
	可编程 I/O
	支持 DFU 模式
4 个数字麦克风 (型号：MP34DT01-M)	单电源电压
	低功耗
	120 dB SPL 声学过载点
	61 dB 信噪比
	全方位的灵敏度

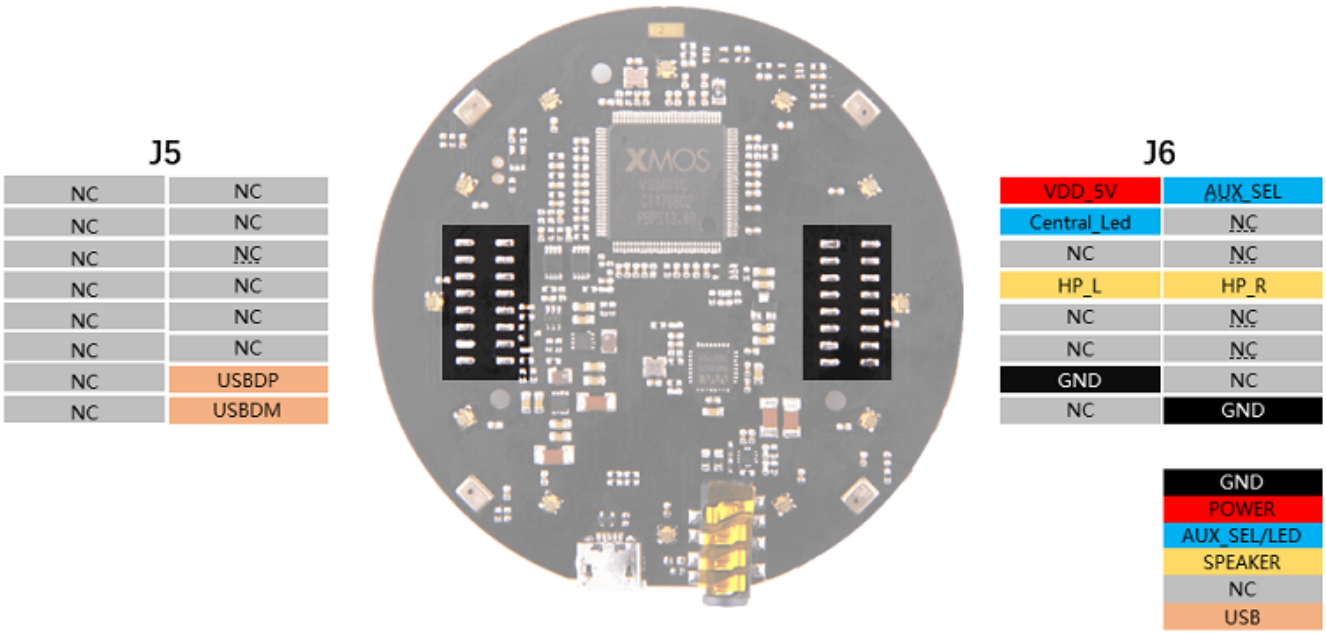
	-26 dB FS 灵敏度
	PDM 输出
12 RGB LEDs (型号：APA102)	256 级亮度
	800kHz 线路数据传输
音频输出	板载 3.5mm Aux 输出
	WOLFSON WM8960
	24 位或 16 位 16kHz 立体声输出
	16 Ω @ 3.3 V 下输出功率为 40mW
尺寸	直径 70mm
功率	Micro USB 或扩展接头供 5V 电源
	功耗 190mA

硬件概述

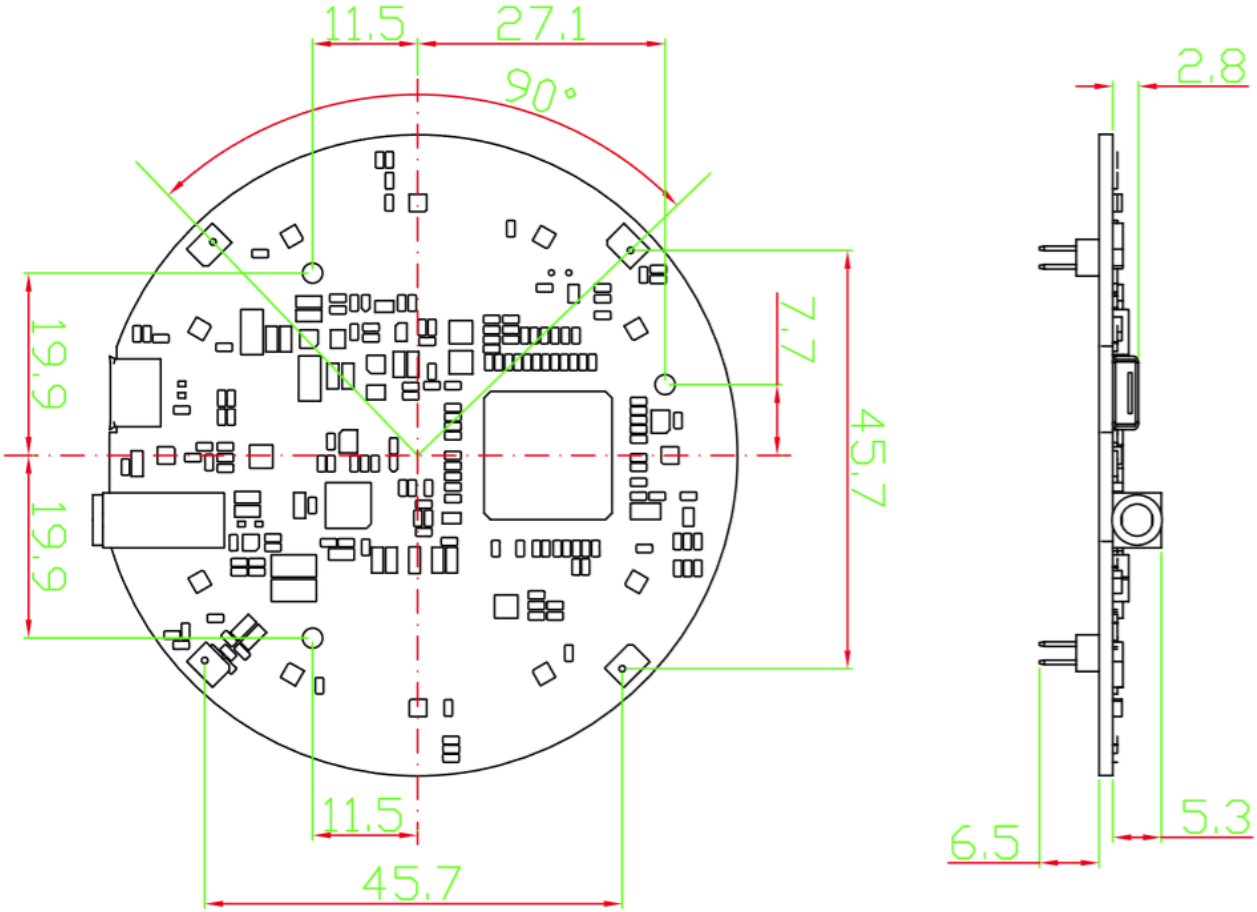


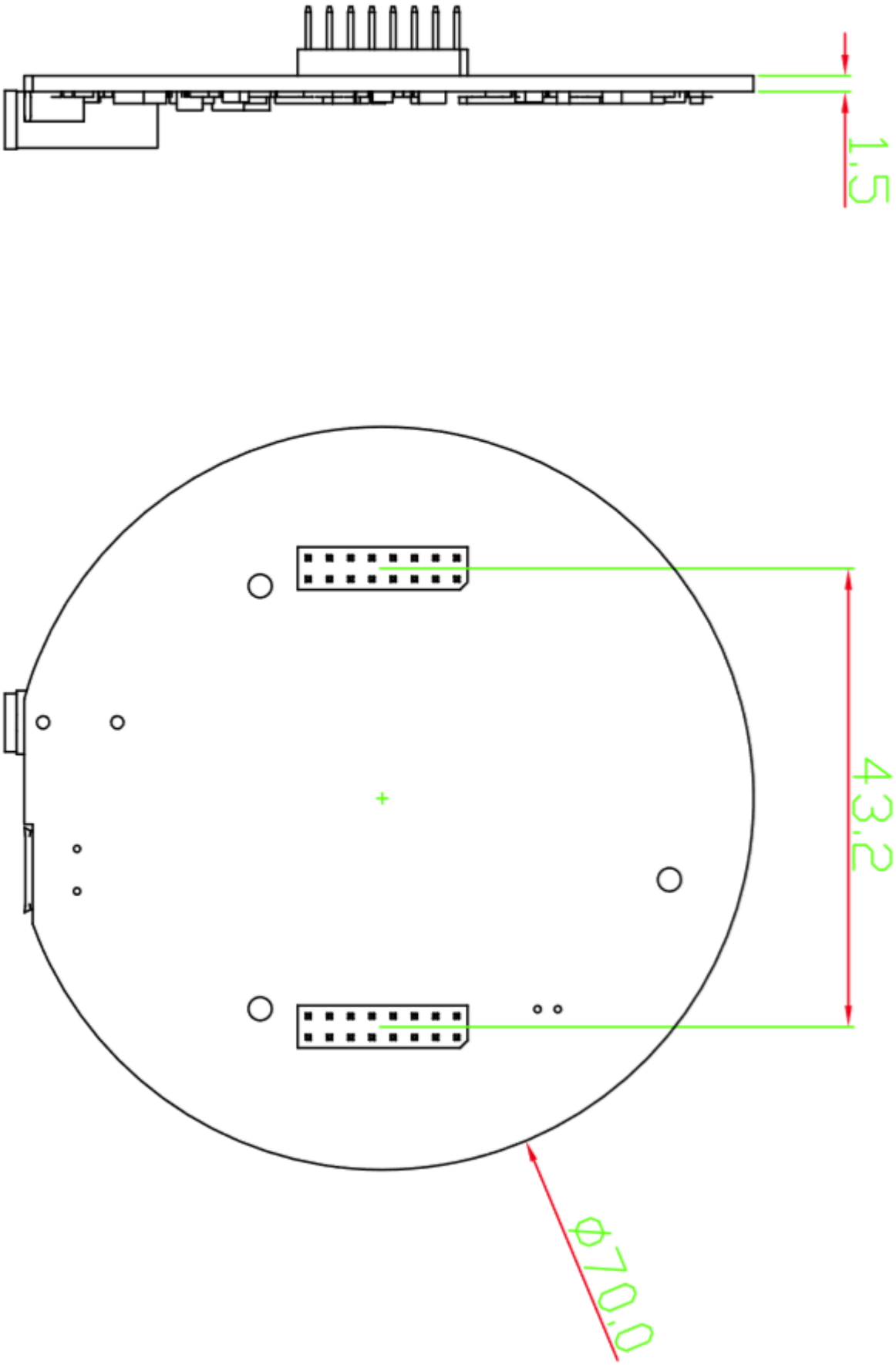
- ① **XMOS XVF-3000 :** 它集成了先进的 DSP 算法，包括声学回声消除 (AEC)，波束成形，去混响，噪声抑制和增益控制。
- ② **Digital Microphone :** MP34DT01-M 是一款超小型，低功耗，全方位的数字 MEMS 麦克风，内置电容式感应元件和 IC 接口。
- ③ **RGB LED :** 三色 RGB LED。
- ④ **USB Port :** 提供电源并控制麦克风阵列。
- ⑤ **3.5mm Headphone jack :** 输出音频，我们可以将有源音响或耳机插入此端口。
- ⑥ **WM8960 :** WM8960 是一款低功耗立体声编解码器，采用 D 类扬声器驱动器，可为每个通道提供 1 W，总计 8 W 的负载。

引脚图


J5
NC NC
NC NC
NC NC
NC NC
NC NC
NC NC
NC USBDP
NC USBDM
J6
VDD_5V AUX_SEL
Central_Led NC
NC NC
HP_L HP_R
NC NC
NC NC
GND NC
NC GND
GND
POWER
AUX_SEL/LED
SPEAKER
NC
USB

尺寸图


11.5
27.1
90°
19.9
19.9
7.7
45.7
11.5
45.7
2.8
6.5
5.3



创意应用

- USB 语音捕获
- 智能音响

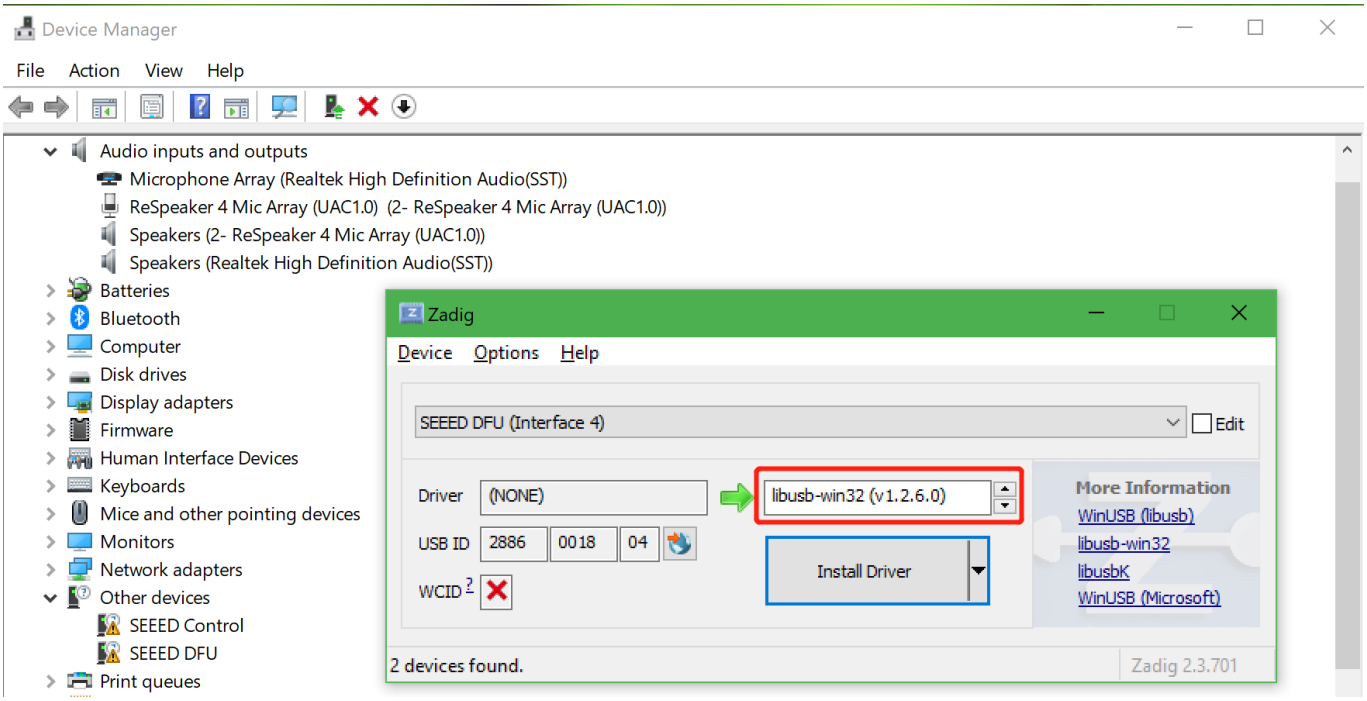
- 智能语音助理系统
- 录音机
- 语音会议系统
- 会议通信设备
- 语音互动机器人
- 车载语音助手
- 其他需要语音命令的场景

入门指导

!!!Note ReSpeaker Mic Array v2.0 兼容 Windows，Mac 和 Linux 系统。以下脚本在 Python2.7 上进行了测试。

安装 DFU 和 LED 控制驱动程序

- **Windows :** 音频录制和播放运行良好。Windows 上仅需 Libusb-win32 驱动程序来控制 LED 指示灯。我们使用 一个方便的工具 - Zadig为 `SEEED DFU` 和 `SEEED Control` 安装 libusb-win32 驱动程序 (ReSpeaker Mic Array 在 Windows 设备管理器中会显示 2 个设备)。



!!!Warning 请确保选择 libusb-win32，而不是 WinUSB 或 libusbK。

- **MAC :** 无需安装驱动
- **Linux :** 无需安装驱动

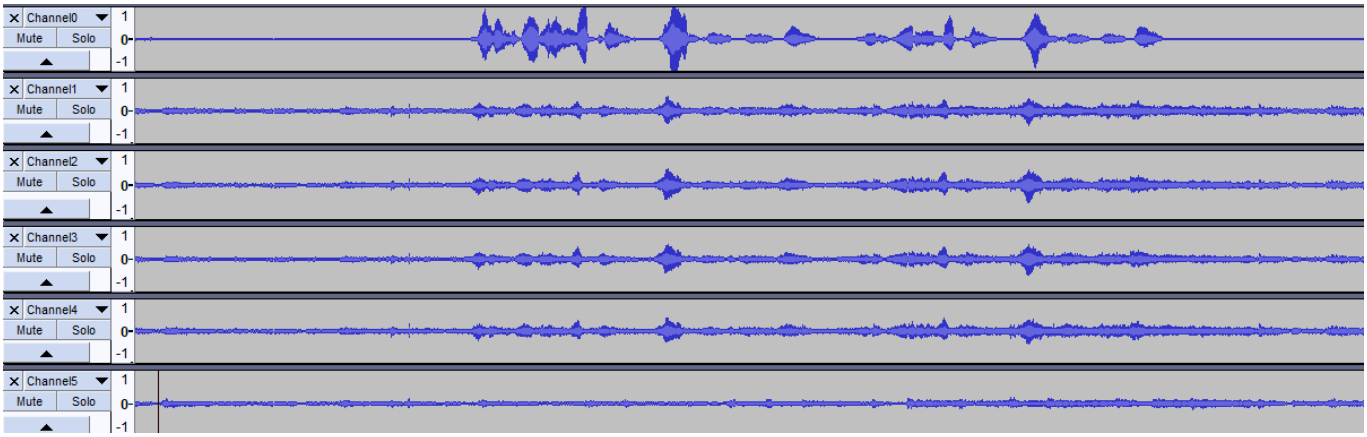
更新固件

有2个固件。 一个包含 1 个通道数据，另一个包含 6 个通道数据。这里是差异表 :

固件版本	通道数	说明
1_channel_firmware.bin	1	为 ASR 处理音频

固件版本	通道数	说明
6_channels_firmware.bin	6	通道 0 : ASR 处理音频，通道 1 : mic1 原始数据，通道 2 : mic2 原始数据，通道 3 : mic3 原始数据，通道 4 : mic4 原始数据，通道 5 : 合并播放

这是 audacity 刷 i6_firmware 后的录音



对于 **Linux 系统 :** ReSpeaker Mic Array v2.0 支持 USB DFU。我们开发了一个 python script dfu.py 来通过 USB 更新固件。

```
sudo apt-get update
sudo pip install pyusb click
git clone https://github.com/respeaker/usb_4_mic_array.git
cd usb_4_mic_array
sudo python dfu.py --download default_firmware.bin  # Change the bin names base on
needs
```

这是固件下载结果。

```
pi@raspberrypi:~/usb_4_mic_array $ sudo python dfu.py --download default_firmware.bin
entering dfu mode
found dfu device
downloading
150336 bytes
done
```

对于 **Windows 用户 :** 我们不建议使用 Windows 更新固件。

控制 LED

我们可以通过 USB 控制 ReSpeaker Mic Array V2 的 LED。USB 设备具有供应商特定的等级接口，可用于通过 USB 控制传输发送数据。我们引用 pyusb python library 并发布 usb_pixel_ring python library。

LED 控制命令由 pyusb 的 usb.core.Device.ctrl_transfer() 发送，其参数如下 :

```
ctrl_transfer(usb.util.CTRL_OUT | usb.util.CTRL_TYPE_VENDOR |
usb.util.CTRL_RECIPIENT_DEVICE, 0, command, 0x1C, data, TIMEOUT)
```

这里是 usb_pixel_ring APIs。

Command	Data	API	Note
0	[0]	pixel_ring.trace()	trace mode, LEDs changing depends on VAD* and DOA*
1	[red, green, blue, 0]	pixel_ring.mono()	mono mode, set all RGB LED to a single color, for example Red(0xFF0000), Green(0x00FF00)， Blue(0x0000FF)
2	[0]	pixel_ring.listen()	listen mode, similar with trace mode, but not turn LEDs off
3	[0]	pixel_ring.speak()	wait mode
4	[0]	pixel_ring.think()	speak mode
5	[0]	pixel_ring.spin()	spin mode
6	[r, g, b, 0] * 12	pixel_ring.custimize()	custom mode, set each LED to its own color
0x20	[brightness]	pixel_ring.set_brightness()	set brightness, range: 0x00~0x1F
0x21	[r1, g1, b1, 0, r2, g2, b2, 0]	pixel_ring.set_color_palette()	set color palette, for example, pixel_ring.set_color_palette(0xff0000, 0x00ff00) together with pixel_ring.think()
0x22	[vad_led]	pixel_ring.set_vad_led()	set center LED: 0 - off, 1 - on, else - depends on VAD
0x23	[volume]	pixel_ring.set_volume()	show volume, range: 0 ~ 12
0x24	[pattern]	pixel_ring.change_pattern()	set pattern, 0 - Google Home pattern, others - Echo pattern

对于 **Linux 系统 :** 这里是控制 LED 的示例。

```
git clone https://github.com/respeaker/pixel_ring.git
cd pixel_ring
sudo python setup.py install
sudo python examples/usb_mic_array.py
```

这里是 usb_mic_array.py 的代码。

```
import time
from pixel_ring import pixel_ring
```

```
if __name__ == '__main__':
    pixel_ring.change_pattern('echo')
    while True:

        try:
            pixel_ring.wakeup()
            time.sleep(3)
            pixel_ring.think()
            time.sleep(3)
            pixel_ring.speak()
            time.sleep(6)
            pixel_ring.off()
            time.sleep(3)
        except KeyboardInterrupt:
            break


    pixel_ring.off()
    time.sleep(1)
```

对于 **Windows** 系统 这里是控制 LED 的示例

```
git clone https://github.com/respeaker/pixel_ring.git
cd pixel_ring/pixel_ring
```

用下面的代码创建一个 led_control.py 并运行 'python led_control.py'

```
from usb_pixel_ring_v2 import PixelRing
import usb.core
import usb.util
import time

dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)
print dev
if dev:
    pixel_ring = PixelRing(dev)

    while True:
        try:
            pixel_ring.wakeup(180)
            time.sleep(3)
            pixel_ring.listen()
            time.sleep(3)
            pixel_ring.think()
            time.sleep(3)
            pixel_ring.set_volume(8)
            time.sleep(3)
            pixel_ring.off()
```

```
            time.sleep(3)
        except KeyboardInterrupt:
            break

    pixel_ring.off()
```

!!!Note 如果您在屏幕上看到 "None"，请重新安装 libusb-win32 驱动程序。

调音

我们可以配置一些内置算法的参数。它适用于 Linux 和 Windows。

- 获取完整列表参数 :

```
git clone https://github.com/respeaker/usb_4_mic_array.git
cd usb_4_mic_array
python tuning.py -p
```

- 例如，我们可以关闭自动增益控制 (AGC) :

```
python tuning.py AGCONOFF 0
```

提取语音

使用 PyAudio python library 通过 USB 提取语音。

对于 Linux 系统 : 可以使用下面的命令来录制或播放语音。

```
arecord -D plughw:1,0 -f cd test.wav # record, please use the arecord -l to check
the card and hardware first
aplay -D plughw:1,0 -f cd test.wav # play, please use the aplay -l to check the
card and hardware first
arecord -D plughw:1,0 -f cd |aplay -D plughw:1,0 -f cd # record and play at the
same time
```

我们也可以使用 python 脚本来提取语音。

- 步骤 1，我们需要运行以下脚本来获取麦克风阵列的设备索引号 :

```
sudo pip install pyaudio
cd ~
nano get_index.py
```

- Step 2, 复制下面的代码并粘贴到 get_index.py.

```
import pyaudio

p = pyaudio.PyAudio()
info = p.get_host_api_info_by_index(0)
numdevices = info.get('deviceCount')

for i in range(0, numdevices):
        if (p.get_device_info_by_host_api_device_index(0,
i).get('maxInputChannels')) > 0:
            print "Input Device id ", i, " - ",
p.get_device_info_by_host_api_device_index(0, i).get('name')
```

- Step 3, 按 Ctrl + X 退出并按 Y 保存。
- Step 4, 运行 'sudo python get_index.py'，我们将看到如下的设备 ID。

```
Input Device id  2  -  ReSpeaker 4 Mic Array (UAC1.0): USB Audio (hw:1,0)
```

- Step 5, 将 `RESPEAKER_INDEX = 2` 更改为索引号。运行 python 脚本 record.py 来录制语音。

```
import pyaudio
import wave

RESPEAKER_RATE = 16000
RESPEAKER_CHANNELS = 1 # change base on firmwares, default_firmware.bin as 1 or
i6_firmware.bin as 6
RESPEAKER_WIDTH = 2
# run getDeviceInfo.py to get index
RESPEAKER_INDEX = 2  # refer to input device id
CHUNK = 1024
RECORD_SECONDS = 5
WAVE_OUTPUT_FILENAME = "output.wav"

p = pyaudio.PyAudio()

stream = p.open(
            rate=RESPEAKER_RATE,
            format=p.get_format_from_width(RESPEAKER_WIDTH),
            channels=RESPEAKER_CHANNELS,
            input=True,
            input_device_index=RESPEAKER_INDEX,)

print("* recording")

frames = []
```

```
for i in range(0, int(RESPEAKER_RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data)

print("* done recording")

stream.stop_stream()
stream.close()
p.terminate()

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(RESPEAKER_CHANNELS)
wf.setsampwidth(p.get_sample_size(p.get_format_from_width(RESPEAKER_WIDTH)))
wf.setframerate(RESPEAKER_RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

对于 **Windows 系统:** 我们首先运行命令 'pip install pyaudio'，然后使用 get_index.py 和 record.py 来提取语音。

!!!Warning 如果看到 "Error: %1 is not a valid Win32 application."，请安装 Python Win32 版本。

实时声源定位跟踪

ODAS 代表 Open embeddeD Audition System。这是一个专门用于声源定位，追踪，分离和后期过滤的库。

对于 **Linux 用户:**

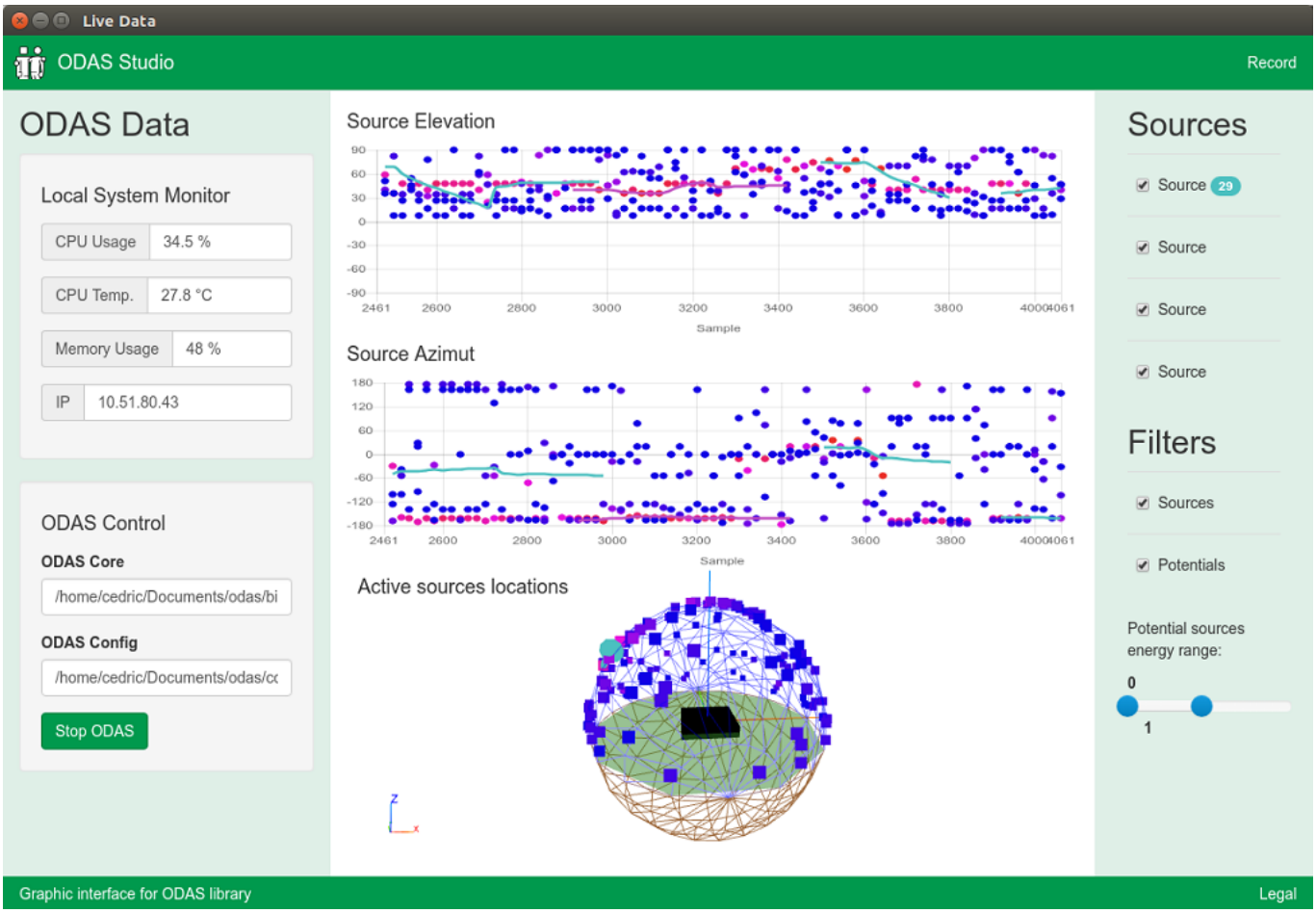
- Step 1. 获得 ODAS 并构建它。

```
sudo apt-get install libfftw3-dev libconfig-dev libasound2-dev
sudo apt-get install cmake
git clone https://github.com/introlab/odas.git --branch=dev
mkdir odas/build
cd odas/build
cmake ..
make
```

- Step 2. 获得 ODAS Studio 然后打开.
- Step 3. odascore 将位于 odas/bin/odascore，配置文件是 odas.cfg。请根据您的声卡号更改 odas.cfg。

```
interface: {
    type = "soundcard";
    card = 1;
    device = 0;
}
```

- Step 4. 使用包含 4 声道原始音频数据的 i6_firmware.bin 更新麦克风阵列。



对于 **Windows 系统：**请参考 ODAS.

FAQ

Q1: 内置算法的参数

```
pi@raspberrypi:~/usb_4_mic_array $ python tuning.py -p
name                    type    max     min     r/w     info
-------------------------------
AECFREEZEONOFF          int     1       0       rw      Adaptive Echo Canceler
updates inhibit.
                                                        0 = Adaptation enabled
                                                        1 = Freeze adaptation,
filter only
AECNORM                 float   16      0.25    rw      Limit on norm of AEC
filter coefficients
AECPATHCHANGE           int     1       0       ro      AEC Path Change Detection.
                                                        0 = false (no path
change detected)
                                                        1 = true (path change
detected)
AECSILENCELEVEL         float   1       1e-09   rw      Threshold for signal
detection in AEC [-inf .. 0] dBov (Default: -80dBov = 10log10(1x10-8))
AECSILENCEMODE          int     1       0       ro      AEC far-end silence
detection status.
```

```
                                                                0 = false (signal
detected)
                                                                1 = true (silence
detected)
AGCDESIREDLEVEL         float   0.99    1e-08   rw      Target power level of the
output signal.
                                                                [-inf .. 0] dBov
(default: -23dBov = 10log10(0.005))
AGCGAIN                 float   1000    1       rw      Current AGC gain factor.
                                                                [0 .. 60] dB (default:
0.0dB = 20log10(1.0))
AGCMAXGAIN              float   1000    1       rw      Maximum AGC gain factor.
                                                                [0 .. 60] dB (default
30dB = 20log10(31.6))
AGCONOFF                int     1       0       rw      Automatic Gain Control.
                                                                0 = OFF
                                                                1 = ON
AGCTIME                 float   1       0.1     rw      Ramps-up / down time-
constant in seconds.
CNIONOFF                int     1       0       rw      Comfort Noise Insertion.
                                                                0 = OFF
                                                                1 = ON
DOAANGLE                int     359     0       ro      DOA angle. Current value.
Orientation depends on build configuration.
ECHOONOFF               int     1       0       rw      Echo suppression.
                                                                0 = OFF
                                                                1 = ON
FREEZEONOFF             int     1       0       rw      Adaptive beamformer
updates.
                                                                0 = Adaptation enabled
                                                                1 = Freeze adaptation,
filter only
FSBPATHCHANGE           int     1       0       ro      FSB Path Change Detection.
                                                                0 = false (no path
change detected)
                                                                1 = true (path change
detected)
FSBUPDATED              int     1       0       ro      FSB Update Decision.
                                                                0 = false (FSB was not
updated)
                                                                1 = true (FSB was
updated)
GAMMAVAD_SR             float   1000    0       rw      Set the threshold for
voice activity detection.
                                                                [-inf .. 60] dB
(default: 3.5dB 20log10(1.5))
GAMMA_E                 float   3       0       rw      Over-subtraction factor of
echo (direct and early components). min .. max attenuation
GAMMA_ENL               float   5       0       rw      Over-subtraction factor of
non-linear echo. min .. max attenuation
GAMMA_ETAIL             float   3       0       rw      Over-subtraction factor of
echo (tail components). min .. max attenuation
GAMMA_NN                float   3       0       rw      Over-subtraction factor of
non- stationary noise. min .. max attenuation
```

```
GAMMA_NN_SR             float   3       0       rw      Over-subtraction factor of
non-stationary noise for ASR.
                                                            [0.0 .. 3.0] (default:
1.1)
GAMMA_NS                float   3       0       rw      Over-subtraction factor of
stationary noise. min .. max attenuation
GAMMA_NS_SR             float   3       0       rw      Over-subtraction factor of
stationary noise for ASR.
                                                            [0.0 .. 3.0] (default:
1.0)
HPFONOFF                int     3       0       rw      High-pass Filter on
microphone signals.
                                                            0 = OFF
                                                            1 = ON - 70 Hz cut-off
                                                            2 = ON - 125 Hz cut-
off
                                                            3 = ON - 180 Hz cut-
off
MIN_NN                  float   1       0       rw      Gain-floor for non-
stationary noise suppression.
                                                            [-inf .. 0] dB
(default: -10dB = 20log10(0.3))
MIN_NN_SR               float   1       0       rw      Gain-floor for non-
stationary noise suppression for ASR.
                                                            [-inf .. 0] dB
(default: -10dB = 20log10(0.3))
MIN_NS                  float   1       0       rw      Gain-floor for stationary
noise suppression.
                                                            [-inf .. 0] dB
(default: -16dB = 20log10(0.15))
MIN_NS_SR               float   1       0       rw      Gain-floor for stationary
noise suppression for ASR.
                                                            [-inf .. 0] dB
(default: -16dB = 20log10(0.15))
NLAEC_MODE              int     2       0       rw      Non-Linear AEC training
mode.
                                                            0 = OFF
                                                            1 = ON - phase 1
                                                            2 = ON - phase 2
NLATTENONOFF            int     1       0       rw      Non-Linear echo
attenuation.
                                                            0 = OFF
                                                            1 = ON
NONSTATNOISEONOFF       int     1       0       rw      Non-stationary noise
suppression.
                                                            0 = OFF
                                                            1 = ON
NONSTATNOISEONOFF_SR    int     1       0       rw      Non-stationary noise
suppression for ASR.
                                                            0 = OFF
                                                            1 = ON
RT60                    float   0.9     0.25    ro      Current RT60 estimate in
seconds
RT60ONOFF               int     1       0       rw      RT60 Estimation for AES. 0
```

```
= OFF 1 = ON
SPEECHDETECTED          int     1       0       ro      Speech detection status.
                                                            0 = false (no speech
detected)
                                                            1 = true (speech
detected)
STATNOISEONOFF          int     1       0       rw      Stationary noise
suppression.
                                                            0 = OFF
                                                            1 = ON
STATNOISEONOFF_SR       int     1       0       rw      Stationary noise
suppression for ASR.
                                                            0 = OFF
                                                            1 = ON
TRANSIENTONOFF          int     1       0       rw      Transient echo
suppression.
                                                            0 = OFF
                                                            1 = ON
VOICEACTIVITY           int     1       0       ro      VAD voice activity status.
                                                            0 = false (no voice
activity)
                                                            1 = true (voice
activity)
```

Q2: ImportError: No module named usb.core

A2: Run sudo pip install pyusb to install the pyusb.

```
pi@raspberrypi:~/usb_4_mic_array $ sudo python tuning.py DOAANGLE
Traceback (most recent call last):
  File "tuning.py", line 5, in <module>
    import usb.core
ImportError: No module named usb.core
pi@raspberrypi:~/usb_4_mic_array $ sudo pip install pyusb
Collecting pyusb
  Downloading pyusb-1.0.2.tar.gz (54kB)
    100% |████████████████████████████████| 61kB 101kB/s
Building wheels for collected packages: pyusb
  Running setup.py bdist_wheel for pyusb ... done
  Stored in directory:
/root/.cache/pip/wheels/8b/7f/fe/baf08bc0dac02ba17f3c9120f5dd1cf74aec4c54463bc85cf
9
Successfully built pyusb
Installing collected packages: pyusb
Successfully installed pyusb-1.0.2
pi@raspberrypi:~/usb_4_mic_array $ sudo python tuning.py DOAANGLE
DOAANGLE: 180
```

Q3: 有没有树莓派的**Alexa**的样例**?**

A3: Yes, we can connect the mic array v2.0 to raspberry usb port and follow Raspberry Pi Quick Start Guide with Script to do the voice interaction with alexa.

Q4: 有在**ROS**系统上面运行**Mic array v2.0** 的历程吗

A4: Yes, thanks for Yuki sharing the package for integrating ReSpeaker Mic Array v2 with ROS (Robot Operating System) Middleware.

Q5: 能直接通过**3.5mm**耳机孔听到采样的声音吗**?**

A5 不行的。3.5mm耳机孔音源来自于上位机，但是如果用树莓派的话可以通过执行arecord -D plughw:1,0 -f cd |aplay -D plughw:1,0 -f cd 达到目的

资源下载

- **[产品简介]** ReSpeaker MicArray v2.0 Product Brief
- **[产品简介]** XVF3000 Product Brief
- **[芯片数据手册]** XVF3000 Datasheet

技术支持

如果您有任何技术问题，请联系 techsupport@seeed.cc。或者将问题提交到我们的 论坛。