

ReSpeaker USB Mic Array



An out-of-the-box voice pick-up device is the voice of the customer.

During the past year, [Respeaker Mic Array V2.0](#) has been sold out for more than 10K units in the format of the development board. Customers keep requesting a complete device with an enclosure, which is challenging for them to design it, considering the acoustic principles.

And here Seeed provides the answer with ReSpeaker USB Mic Array:

- An out-of-box device with a well-designed acoustic structure brings the flexibility for the customer to build in their solution.
- Mold injected enclosure available, saves the time to go to the market and the mold cost.

The difference between the PCBA inside ReSpeaker USB Mic Array and Respeaker Mic Array V2.0:

- Optimized power circuit
- Move the audio jack and micro USB port to the backside.

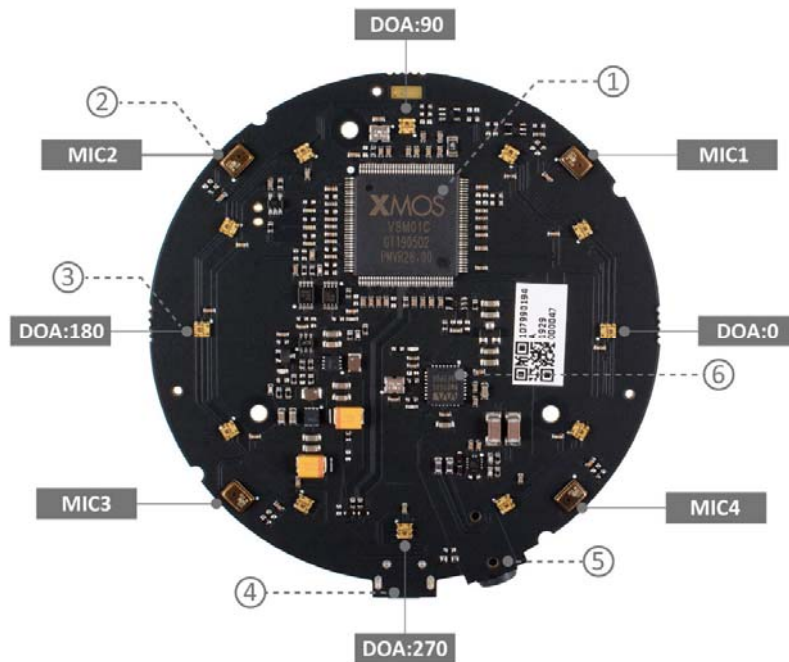
Features

- Far-field voice capture
- Support USB Audio Class 1.0 (UAC 1.0)
- Four microphones array
- 12 programmable RGB LED indicators
- Speech algorithms and features
 - Voice Activity Detection
 - Direction of Arrival
 - Beamforming
 - Noise Suppression
 - De-reverberation
 - Acoustic Echo Cancellation

Specification

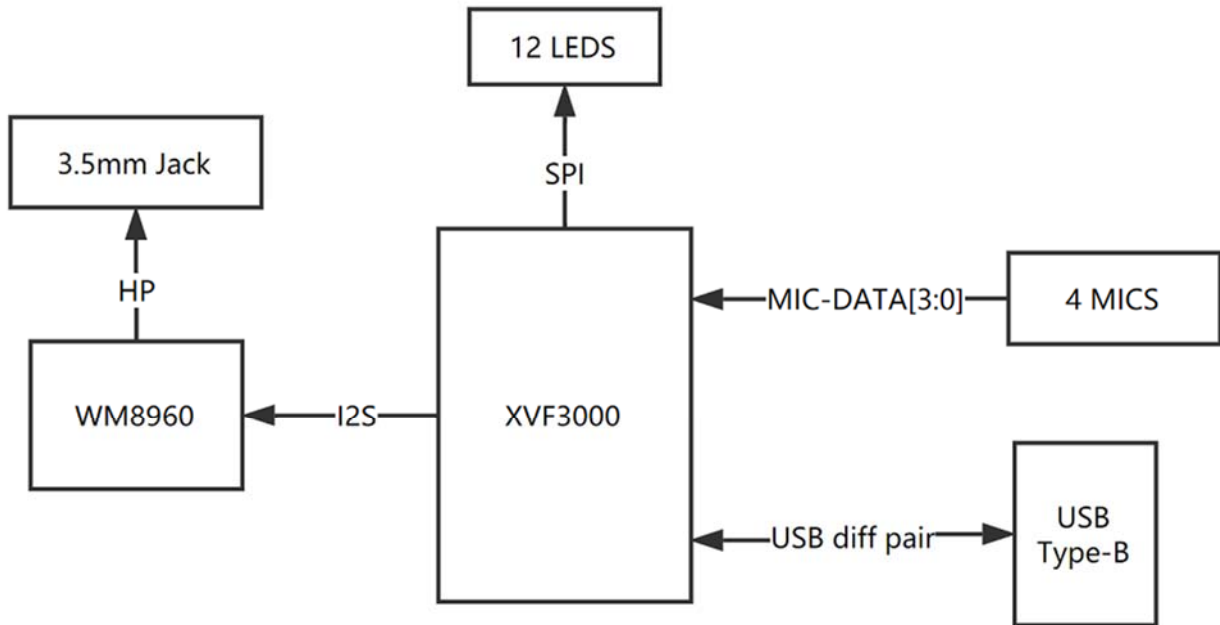
- XVF-3000 from XMOS
- 4 high performance digital microphones
- Supports Far-field Voice Capture
- Speech algorithm on-chip
- 12 programmable RGB LED indicators
- Microphones: ST MP34DT01TR-M
- Sensitivity: -26 dBFS (Omnidirectional)
- Acoustic overload point: 120 dB SPL
- SNR: 61 dB
- Power Supply: 5V DC from Micro USB
- Dimensions: 70mm (Diameter)
- 3.5mm Audio jack output socket
- Power consumption: 5V, 180mA with led on and 170mA with led off
- Max Sample Rate: 48Khz

Hardware Overview



- **① XMOS XVF-3000:** It integrates advanced DSP algorithms that include Acoustic Echo Cancellation (AEC), beamforming, dereverberation, noise suppression and gain control.
- **② Digital Microphone:** The MP34DT01-M is an ultra-compact, lowpower, omnidirectional, digital MEMS microphone built with a capacitive sensing element and an IC interface.
- **③ RGB LED:** Three-color RGB LED.
- **④ USB Port:** Provide the power and control the mic array.
- **⑤ 3.5mm Headphone jack:** Output audio, We can plug active speakers or Headphones into this port.
- **⑥ WM8960:** The WM8960 is a low power stereo codec featuring Class D speaker drivers to provide 1 W per channel into 8 W loads.

System Diagram



Applications

- USB Voice Capture
- Smart Speaker
- Intelligent Voice Assistant Systems
- Voice Recorders
- Voice Conferencing System
- Meeting Communicating Equipment
- Voice Interacting Robot
- Car Voice Assistant
- Other Voice Interface Scenarios

Getting Started

Note

ReSpeaker USB Mic Array is compatible with Windows, Mac, Linux systems and android. The below scripts are tested on Python2.7.

Update Firmware

Here is the table for the differences.

Firmware	Channels	Note
1_channel_firmware.bin	1	processed audio for ASR

Firmware	Channels	Note
1_channel_firmware_6.02dB.bin	1	same as 1_channel_firmware.bin, but 4 microphones have a 6.02dB gain
1_channel_firmware_12.06dB.bin	1	same as 1_channel_firmware.bin, but 4 microphones have a 12.04dB gain
48k_1_channels_firmware.bin	1	48k sample rate, 1 input channel
48k_1_channel_firmware_6.02dB.bin	1	48k sample rate, 1 input channel, but 4 microphones have a 6.02dB gain
6_channels_firmware.bin	6	channel 0: processed audio for ASR, channel 1-4: 4 microphones' raw data, channel 5: playback (factory firmware)
6_channels_firmware_6.02dB.bin	6	same as 6_channels_firmware.bin, but 4 microphones have a 6.02dB gain
6_channels_firmware_12.04dB.bin	6	same as 6_channels_firmware.bin, but 4 microphones have a 12.04dB gain
48k_6_channels_firmware.bin	6	48k sample rate, 6 input channels
48k_6_channels_firmware_6.02dB.bin	6	48k sample rate, 6 input channels, 6.02dB gain

For Linux: The Mic array supports the USB DFU. We develop a python script dfu.py to update the firmware through USB.

```

1sudo apt-get update
2sudo pip install pyusb click
3git clone https://github.com/respeaker/usb_4_mic_array.git
4cd usb_4_mic_array
5sudo python dfu.py --download 6_channels_firmware.bin # The 6 channels
6version
7
8# if you want to use 1 channel,then the command should be like:
9
  sudo python dfu.py --download 1_channel_firmware.bin

```

Here is the firmware downloading result.

```

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

```

For Windows/Mac: We do not suggest use Windows/Mac and Linux virtual machine to update the firmware.

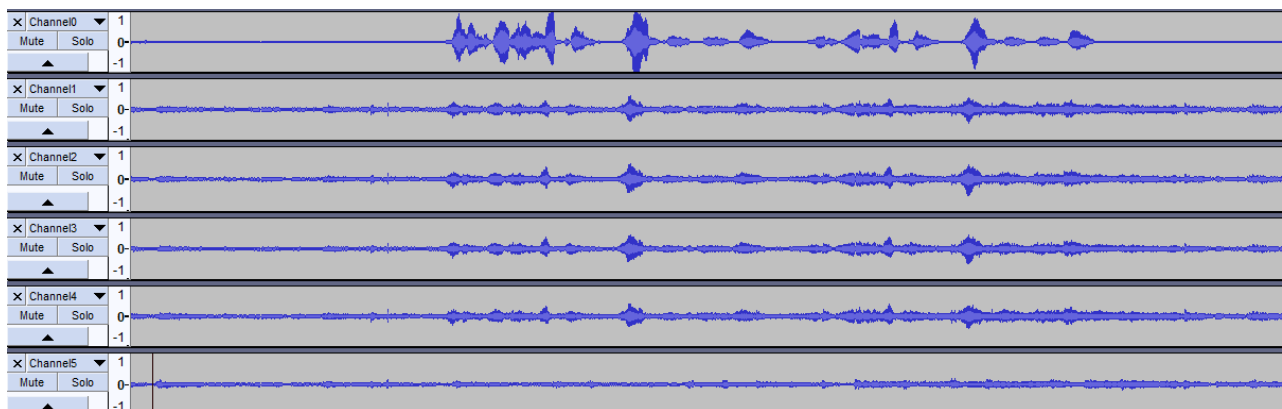
Out of Box Demo

Here is the Acoustic Echo Cancellation example with 6 channels firmware.

- Step 1. Connect the USB cable to PC and audio jack to speaker.



- Step 2. Select the mic array v2.1 as output device in PC side.
- Step 3. Start the audacity to record.
- Step 4. Play music at PC side first and then we talk.
- Step 5. We will see the audacity screen as below, Please click **Solo** to hear each channel audio.



Channel0 Audio(processed by algorithms):

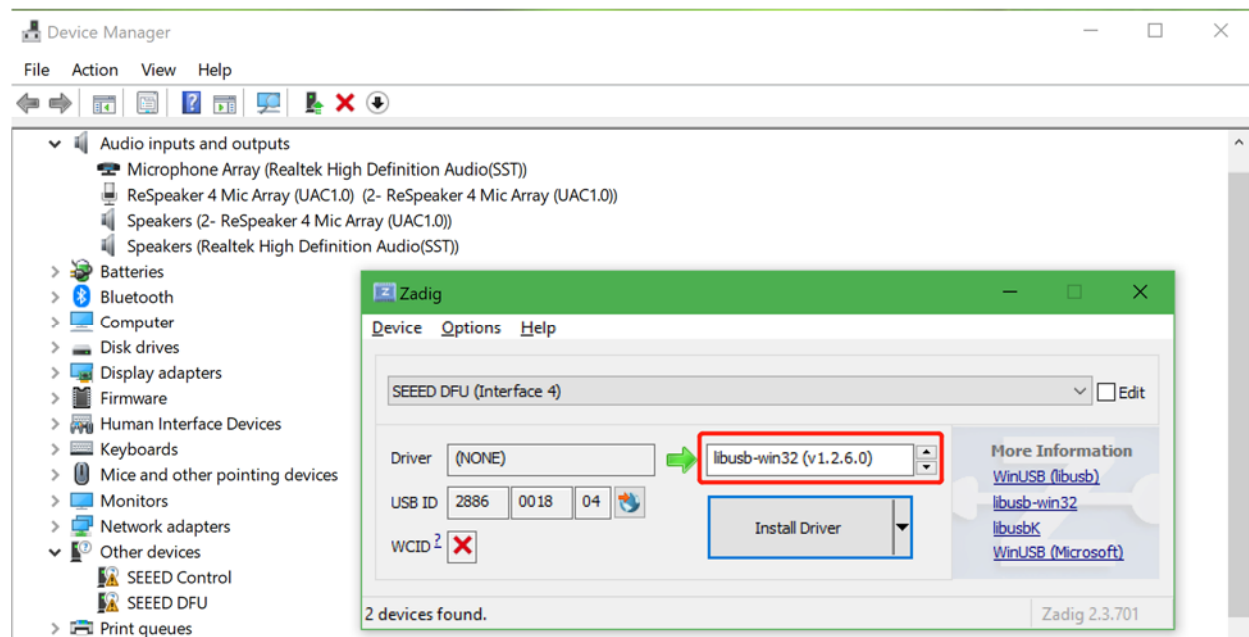
Channel1 Audio(Mic1 raw data):

Channel5 Audio(Playback data):

Here is the video about the DOA and AEC.

Install DFU and LED Control Driver

- **Windows:** Audio recording and playback works well by default. Libusb-win32 driver is only required to control LEDs and DSP parameters on Windows. We use a handy tool - Zadig to install the libusb-win32 driver for both SEED DFU and SEED Control (ReSpeaker Mic Array has 2 devices on Windows Device Manager).



Warning

Please make sure that libusb-win32 is selected, not WinUSB or libusbK.

- **MAC:** No driver is required.
- **Linux:** No driver is required.

Tuning

For Linux/Mac/Windows: We can configure some parameters of built-in algorithms.

- Get the full list parameters, for more info, please refer to FAQ.

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

- Example#1, we can turn off Automatic Gain Control (AGC):

```
1 sudo python tuning.py AGCONOFF 0
```

- Example#2, We can check the DOA angle.

```
1 pi@raspberrypi:~/usb_4_mic_array $ sudo python tuning.py DOAANGLE
2 DOAANGLE: 180
```

Control the LEDs

We can control the ReSpeaker USB Mic Array's LEDs through USB. The USB device has a Vendor Specific Class Interface which can be used to send data through USB Control Transfer. We refer [pyusb python library](#) and come out the [usb_pixel_ring python library](#).

The LED control command is sent by pyusb's `usb.core.Device.ctrl_transfer()`, its parameters as below :

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

Here are the `usb_pixel_ring` APIs.

Command	Data	API	Note
0	[0]	<code>pixel_ring.trace()</code>	trace mode, LEDs changing depends on VAD* and DOA*
1	[red, green, blue, 0]	<code>pixel_ring.mono()</code>	mono mode, set all RGB LED to a single color, for example Red(0xFF0000), Green(0x00FF00), Blue(0x0000FF)
2	[0]	<code>pixel_ring.listen()</code>	listen mode, similar with trace mode, but not turn LEDs off
3	[0]	<code>pixel_ring.speak()</code>	wait mode
4	[0]	<code>pixel_ring.think()</code>	speak mode
5	[0]	<code>pixel_ring.spin()</code>	spin mode
6	[r, g, b, 0] * 12	<code>pixel_ring.customize()</code>	custom mode, set each LED to its own color
0x20	[brightness]	<code>pixel_ring.set_brightness()</code>	set brightness, range: 0x00~0x1F

Command	Data	API	Note
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

For Linux: Here is the example to control the leds. Please follow below commands to run the demo.

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

Here is the code of the usb_mic_array.py.

```
1 import time
2 from pixel_ring import pixel_ring
3
4
5 if __name__ == '__main__':
6     while True:
7
8         try:
9             pixel_ring.wakeup()
10            time.sleep(3)
11            pixel_ring.think()
12            time.sleep(3)
13            pixel_ring.speak()
14            time.sleep(6)
15            pixel_ring.off()
16            time.sleep(3)
17        except KeyboardInterrupt:
18            break
19
20
21 pixel_ring.off()
22 time.sleep(1)
```

For Windows/Mac: Here is the example to control the leds.

- Step 1. Download pixel_ring.

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

- Step 2. Create a led_control.py with below code and run 'python led_control.py'

```
1 from usb_pixel_ring_v2 import PixelRing
2 import usb.core
3 import usb.util
4 import time
5
6 dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)
7 print dev
8 if dev:
9     pixel_ring = PixelRing(dev)
10
11 while True:
12     try:
13         pixel_ring.wakeup(180)
14         time.sleep(3)
15         pixel_ring.listen()
16         time.sleep(3)
17         pixel_ring.think()
18         time.sleep(3)
19         pixel_ring.set_volume(8)
20         time.sleep(3)
21         pixel_ring.off()
22         time.sleep(3)
23     except KeyboardInterrupt:
24         break
25
26 pixel_ring.off()
```

Note

If you see "None" printed on screen, please reinstall the libusb-win32 driver.

DOA (Direction of Arrival)

For Windows/Mac/Linux: Here is the example to view the DOA. The Green LED is the indicator of the voice direction. For the angle, please refer to hardware overview.

- Step 1. Download the usb_4_mic_array.

```
1 git clone https://github.com/respeaker/usb_4_mic_array.git
2 cd usb_4_mic_array
```

- Step 2. Create a `DOA.py` with below code under `usb_4_mic_array` folder and run 'sudo python DOA.py'

```
1 from tuning import Tuning
2 import usb.core
3 import usb.util
4 import time
5
6 dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)
7
8 if dev:
9     Mic_tuning = Tuning(dev)
10    print Mic_tuning.direction
11    while True:
12        try:
13            print Mic_tuning.direction
14            time.sleep(1)
15        except KeyboardInterrupt:
16            break
```

- Step 3. We will see the DOA as below.

```
1 pi@raspberrypi:~/usb_4_mic_array $ sudo python doa.py
2 184
3 183
4 175
5 105
6 104
7 104
8 103
```

VAD (Voice Activity Detection)

For Windows/Mac/Linux: Here is the example to view the VAD. The Red LED is the indicator of the VAD.

- Step 1. Download the `usb_4_mic_array`.

```
1 git clone https://github.com/respeaker/usb_4_mic_array.git
2 cd usb_4_mic_array
```

- Step 2. Create a `VAD.py` with below code under `usb_4_mic_array` folder and run 'sudo python VAD.py'

```
1 from tuning import Tuning
2 import usb.core
3 import usb.util
4 import time
5
6 dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)
```

```

7 #print dev
8 if dev:
9     Mic_tuning = Tuning(dev)
10    print Mic_tuning.is_voice()
11    while True:
12        try:
13            print Mic_tuning.is_voice()
14            time.sleep(1)
15        except KeyboardInterrupt:
16            break

```

- Step 3. We will see the DOA as below.

```

1 pi@raspberrypi:~/usb_4_mic_array $ sudo python VAD.py
2 0
3 0
4 0
5 1
6 0
7 1
8 0

```

Note

For the threshold of VAD, we also can use the GAMMAVAD_SR to set. Please refer to [Tuning](#) for more detail.

Extract Voice

We use [PyAudio python library](#) to extract voice through USB.

For Linux: We can use below commands to record or play the voice.

```

1 arecord -D plughw:1,0 -f cd test.wav # record, please use the arecord -l to
2 check the card and hardware first
3 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

```

We also can use python script to extract voice.

- Step 1, We need to run the following script to get the device index number of Mic Array:

```

1 sudo pip install pyaudio
2 cd ~
3 nano get_index.py

```

- Step 2, copy below code and paste on [get_index.py](#).

```

1 import pyaudio
2
3 p = pyaudio.PyAudio()
4 info = p.get_host_api_info_by_index(0)
5 numdevices = info.get('deviceCount')
6
7 for i in range(0, numdevices):
8     if (p.get_device_info_by_host_api_device_index(0,
9 i).get('maxInputChannels')) > 0:
10         print "Input Device id ", i, " - ",
11         p.get_device_info_by_host_api_device_index(0, i).get('name')

```

- Step 3, press Ctrl + X to exit and press Y to save.
- Step 4, run 'sudo python get_index.py' and we will see the device ID as below.

```

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

```

- Step 5, change RESPEAKER_INDEX = 2 to index number. Run python script [record.py](#) to record a speech.

```

import pyaudio
1 import wave

2 RESPEAKER_RATE = 16000
3 RESPEAKER_CHANNELS = 6 # change base on firmwares, 1_channel_firmware.bin
4 as 1 or 6_channels_firmware.bin as 6
5 RESPEAKER_WIDTH = 2
6 # run getDeviceInfo.py to get index
7 RESPEAKER_INDEX = 2 # refer to input device id
8 CHUNK = 1024
9 RECORD_SECONDS = 5
10 WAVE_OUTPUT_FILENAME = "output.wav"

11
12 p = pyaudio.PyAudio()

13
14 stream = p.open(
15     rate=RESPEAKER_RATE,
16     format=p.get_format_from_width(RESPEAKER_WIDTH),
17     channels=RESPEAKER_CHANNELS,
18     input=True,
19     input_device_index=RESPEAKER_INDEX,)
20
21 print("* recording")
22
23 frames = []
24
25

```

```

1 for i in range(0, int(RESPEAKER_RATE / CHUNK * RECORD_SECONDS)):
4     data = stream.read(CHUNK)
1     frames.append(data)
5
1 print("* done recording")
6
1 stream.stop_stream()
7 stream.close()
1 p.terminate()
8
1 wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
9 wf.setnchannels(RESPEAKER_CHANNELS)
2 wf.setsampwidth(p.get_sample_size(p.get_format_from_width(RESPEAKER_WIDTH))
0 )
2 wf.setframerate(RESPEAKER_RATE)
1 wf.writeframes(b''.join(frames))
2 wf.close()
2
2
3
2
4
2
5
2
6
2
7
2
8
2
9
3
0
3
1
3
2
3
3
3
3
4
3
5
3
6
3
7
3
8
3
9
4
0
4
1

```

- Step 6. If you want to extract channel 0 data from 6 channels, please follow below code. For other channel X, please change [0::6] to [X::6].

```

import pyaudio
1 import wave
import numpy as np
2
RESPEAKER_RATE = 16000
3 RESPEAKER_CHANNELS = 6 # change base on firmwares, 1_channel_firmware.bin
as 1 or 6_channels_firmware.bin as 6
4 RESPEAKER_WIDTH = 2
# run getDeviceInfo.py to get index
5 RESPEAKER_INDEX = 3 # refer to input device id
CHUNK = 1024
6 RECORD_SECONDS = 3
WAVE_OUTPUT_FILENAME = "output.wav"
7
p = pyaudio.PyAudio()
8
stream = p.open(
9     rate=RESPEAKER_RATE,
1     format=p.get_format_from_width(RESPEAKER_WIDTH),
0     channels=RESPEAKER_CHANNELS,
1     input=True,
1     input_device_index=RESPEAKER_INDEX,)
1
2 print("* recording")
1
3 frames = []
1
4 for i in range(0, int(RESPEAKER_RATE / CHUNK * RECORD_SECONDS)):
1     data = stream.read(CHUNK)
5     # extract channel 0 data from 6 channels, if you want to extract
1 channel 1, please change to [1::6]
6     a = np.fromstring(data, dtype=np.int16)[0::6]
1     frames.append(a.tostring())
7
1 print("* done recording")
8
1 stream.stop_stream()
9 stream.close()
2 p.terminate()
0
2 wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
1 wf.setnchannels(1)
2 wf.setsampwidth(p.get_sample_size(p.get_format_from_width(RESPEAKER_WIDTH))
2 )
2 wf.setframerate(RESPEAKER_RATE)
3 wf.writeframes(b''.join(frames))
2 wf.close()
4
2
5
2
6
2
7

```

2
8
2
9
3
0
3
1
3
2
3
3
3
4
3
5
3
6
3
7
3
8
3
9
4
0
4
1
4
2
4
3
4
4

For Windows:

- Step 1. We run below command to install pyaudio.

```
1 pip install pyaudio
```

- Step 2. Use [get_index.py](#) to get device index.

```
1 C:\Users\XXX\Desktop>python get_index.py
2 Input Device id 0 - Microsoft Sound Mapper - Input
3 Input Device id 1 - ReSpeaker 4 Mic Array (UAC1.0)
4 Input Device id 2 - Internal Microphone (Conexant I)
```

- Step 3. Modify the device index and channels of [record.py](#) and then extract voice.

```
1 C:\Users\XXX\Desktop>python record.py
2 * recording
3 * done recording
```

Warning

If we see "Error: %1 is not a valid Win32 application.", please install Python Win32 version.

For MAC:

- Step 1. We run below command to install pyaudio.

```
1 pip install pyaudio
```

- Step 2. Use [get_index.py](#) to get device index.

```
1 MacBook-Air:Desktop XXX$ python get_index.py
2 Input Device id 0 - Built-in Microphone
3 Input Device id 2 - ReSpeaker 4 Mic Array (UAC1.0)
```

- Step 3. Modify the device index and channels of [record.py](#) and then extract voice.

```
1 MacBook-Air:Desktop XXX$ python record.py
2 2018-03-24 14:53:02.400 Python[2360:16629] 14:53:02.399 WARNING: 140: This
3 application, or a library it uses, is using the deprecated Carbon Component
4 Manager for hosting Audio Units. Support for this will be removed in a
  future release. Also, this makes the host incompatible with version 3 audio
  units. Please transition to the API's in AudioComponent.h.
  * recording
  * done recording
```

Realtime Sound Source Localization and Tracking

[ODAS](#) stands for Open embeddeD Audition System. This is a library dedicated to perform sound source localization, tracking, separation and post-filtering. Let's have a fun with it.

For Linux:

- Step 1. Get ODAS and build it.

```
1 sudo apt-get install libfftw3-dev libconfig-dev libasound2-dev libgconf-2-4
2 git clone https://github.com/introlab/odas.git
3 mkdir odas/build
4 cd odas/build
5 cmake ..
```


6make

- Step 2. Get [ODAS Studio](#) and open it.
- Step 3. The odascore will be at `odas/bin/odaslive`, the **config file** is `odas.cfg`.
- Step 4. Upgrade mic array with `6_channels_firmware.bin` which includes 4 channels raw audio data.

FAQ¶

Q1: Parameters of built-in algorithms

```
1 pi@raspberrypi:~/usb_4_mic_array $ python tuning.py -p
2 name                type      max min r/w info
3 -----
4 AECFREEZEONOFF      int 1    0   rw  Adaptive Echo Canceler updates inhibit.
5                                     0 = Adaptation
6 enabled
7                                     1 = Freeze
8 adaptation, filter only
9 AECNORM              float  16  0.25  rw  Limit on norm of AEC filter
10 coefficients
11 AECPATHCHANGE       int 1    0   ro  AEC Path Change Detection.
12                                     0 = false (no
13 path change detected)
14                                     1 = true (path
15 change detected)
16 AEC SILENCE LEVEL   float  1    1e-09  rw  Threshold for signal detection
17 in AEC [-inf .. 0] dBov (Default: -80dBov = 10log10(1x10-8))
18 AEC SILENCE MODE    int 1    0   ro  AEC far-end silence detection status.
19                                     0 = false
20 (signal detected)
21                                     1 = true
22 (silence detected)
23 AGC DESIRED LEVEL   float  0.99  1e-08  rw  Target power level of the
24 output signal.
25                                     [-inf .. 0]
26 dBov (default: -23dBov = 10log10(0.005))
27 AGC GAIN            float  1000  1     rw  Current AGC gain factor.
28                                     [0 .. 60] dB
29 (default: 0.0dB = 20log10(1.0))
30 AGC MAX GAIN        float  1000  1     rw  Maximum AGC gain factor.
31                                     [0 .. 60] dB
32 (default 30dB = 20log10(31.6))
33 AGC ON OFF         int 1    0   rw  Automatic Gain Control.
34                                     0 = OFF
35                                     1 = ON
36 AGC TIME            float  1    0.1  rw  Ramps-up / down time-constant in
37 seconds.
38 CN ION OFF         int 1    0   rw  Comfort Noise Insertion.
39                                     0 = OFF
40                                     1 = ON
41 DOA ANGLE           int 359  0    ro  DOA angle. Current value. Orientation
42 depends on build configuration.
43 ECHO ON OFF         int 1    0   rw  Echo suppression.
```

```

44                                     0 = OFF
45                                     1 = ON
46 FREEZEONOFF          int 1    0    rw  Adaptive beamformer updates.
47                                     0 = Adaptation
48 enabled
49                                     1 = Freeze
50 adaptation, filter only
51 FSBPATHCHANGE        int 1    0    ro  FSB Path Change Detection.
52                                     0 = false (no
53 path change detected)
54                                     1 = true (path
55 change detected)
56 FSBUPDATED           int 1    0    ro  FSB Update Decision.
57                                     0 = false (FSB
58 was not updated)
59                                     1 = true (FSB
60 was updated)
61 GAMMAVAD_SR          float  1000  0    rw  Set the threshold for voice
62 activity detection.
63                                     [-inf .. 60] dB
64 (default: 3.5dB 20log10(1.5))
65 GAMMA_E              float   3    0    rw  Over-subtraction factor of echo
66 (direct and early components). min .. max attenuation
67 GAMMA_ENL            float   5    0    rw  Over-subtraction factor of non-
68 linear echo. min .. max attenuation
69 GAMMA_ETAIL          float   3    0    rw  Over-subtraction factor of echo
70 (tail components). min .. max attenuation
71 GAMMA_NN             float   3    0    rw  Over-subtraction factor of non-
72 stationary noise. min .. max attenuation
73 GAMMA_NN_SR          float   3    0    rw  Over-subtraction factor of non-
74 stationary noise for ASR.
75                                     [0.0 .. 3.0]
76 (default: 1.1)
77 GAMMA_NS             float   3    0    rw  Over-subtraction factor of
78 stationary noise. min .. max attenuation
79 GAMMA_NS_SR          float   3    0    rw  Over-subtraction factor of
80 stationary noise for ASR.
81                                     [0.0 .. 3.0]
82 (default: 1.0)
83 HPFONOFF             int 3    0    rw  High-pass Filter on microphone signals.
84                                     0 = OFF
85                                     1 = ON - 70 Hz
86 cut-off
87                                     2 = ON - 125 Hz
88 cut-off
89                                     3 = ON - 180 Hz
90 cut-off
91 MIN_NN              float   1    0    rw  Gain-floor for non-stationary noise
92 suppression.
93                                     [-inf .. 0] dB
94 (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
1 Traceback (most recent call last):
  File "tuning.py", line 5, in <module>
    import usb.core
2 ImportError: No module named usb.core

```


Resource

- **[PDF]** ReSpeaker USB Mic Array Dimension
- <https://github.com/SeeedDocument/ReSpeaker-USB-Mics/raw/master/res/dimension.pdf>
- **[DWG]** ReSpeaker USB Mic Array Case 3D Model
- <https://github.com/SeeedDocument/ReSpeaker-USB-Mics/raw/master/res/dimension.pdf>
- **[PDF]** XVF3000 Product Brief
- https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/XVF3000-3100-product-brief_1.4.pdf
- **[PDF]** XVF3000 Datasheet
- https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/XVF3000-3100-TQ128-Datasheet_1.0.pdf

Tech Support

Please submit any technical issue into our forum.

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for [Seeed Studio Accessories](#) category:

Click to view products by [Seeed Studio](#) manufacturer:

Other Similar products are found below :

[101990565](#) [114992282](#) [114992268](#) [101990647](#) [103060000](#) [101020004](#) [101020006](#)

[101020012](#) [101020018](#) [101020028](#) [101020038](#) [101020045](#) [101020049](#) [101020052](#)

[101020055](#) [101020056](#) [101020058](#) [101020472](#) [101020580](#) [101020603](#) [101990007](#)

[101990028](#) [101990053](#) [101990058](#) [101990059](#)