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.

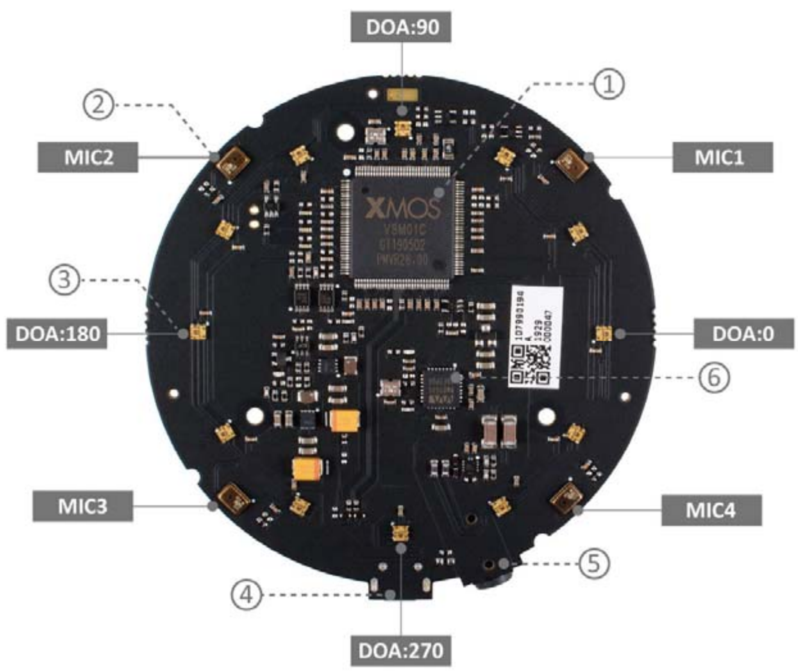
Features

- 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

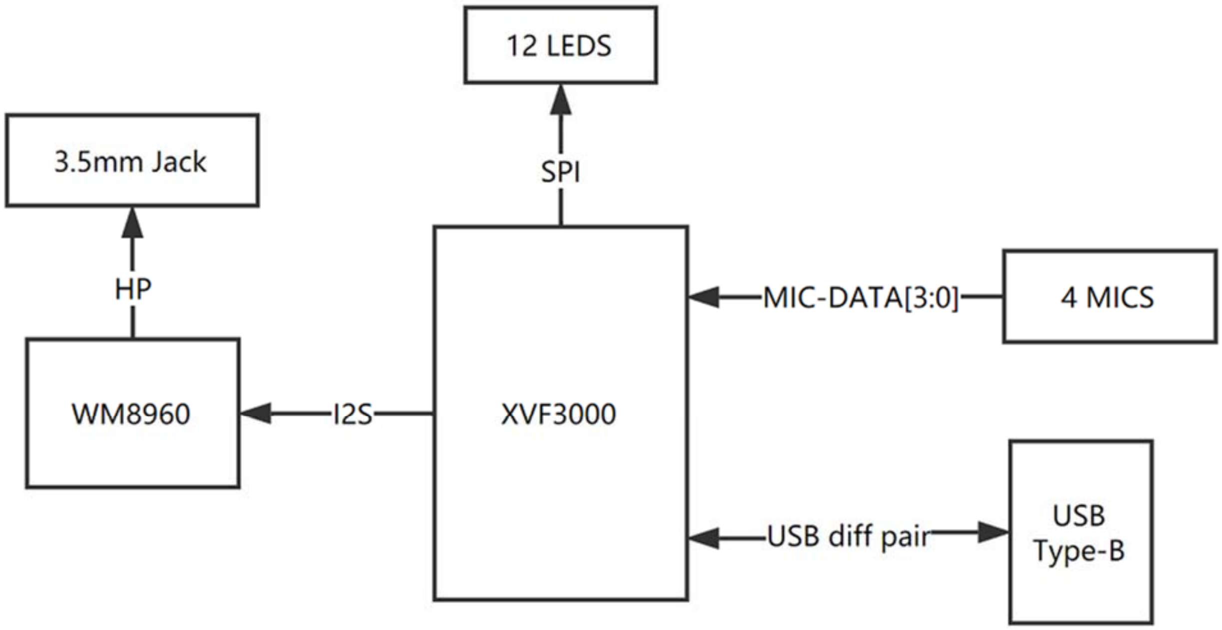
- XVF-3000 from XMOS
- 4 high performance digital microphones
- Supports Far-field Voice Capture
- Speech algorithm on-chip
- 12 programmable RGB LED indicators
- Microphones: ST MP34DT01TR-M
- Sensitivity: -26 dBFS (Omnidirectional)
- Acoustic overload point: 120 dBSPL
- SNR: 61 dB
- Power Supply: 5V 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 compatiable with Windows, Mac, Linux systems andriod. 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.

```
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 6_channels_firmware.bin  # The 6 channels version

# if you want to use 1 channel,then the command should be like:

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
```

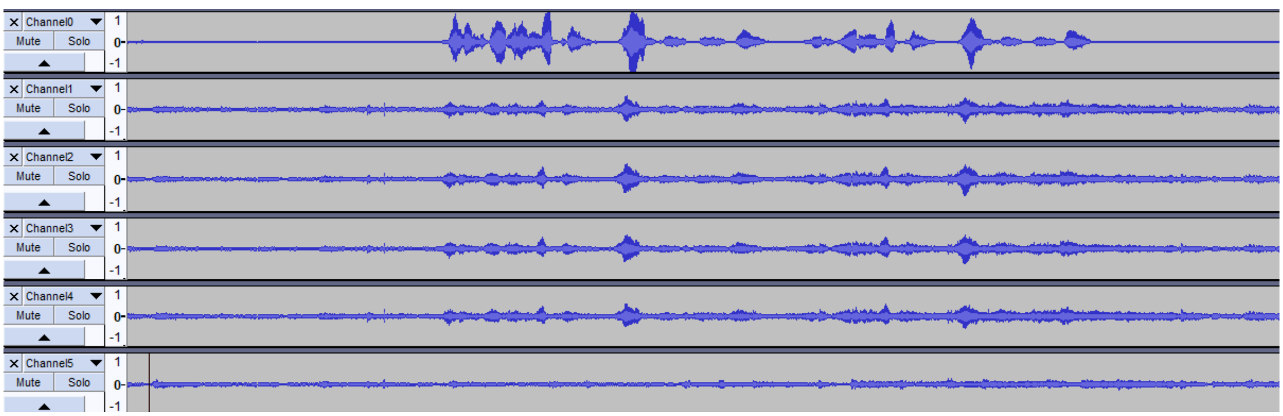
For Windows/Mac: We do not suggest use Windows/Mac and Linux vitual machine to update the firmware.

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):

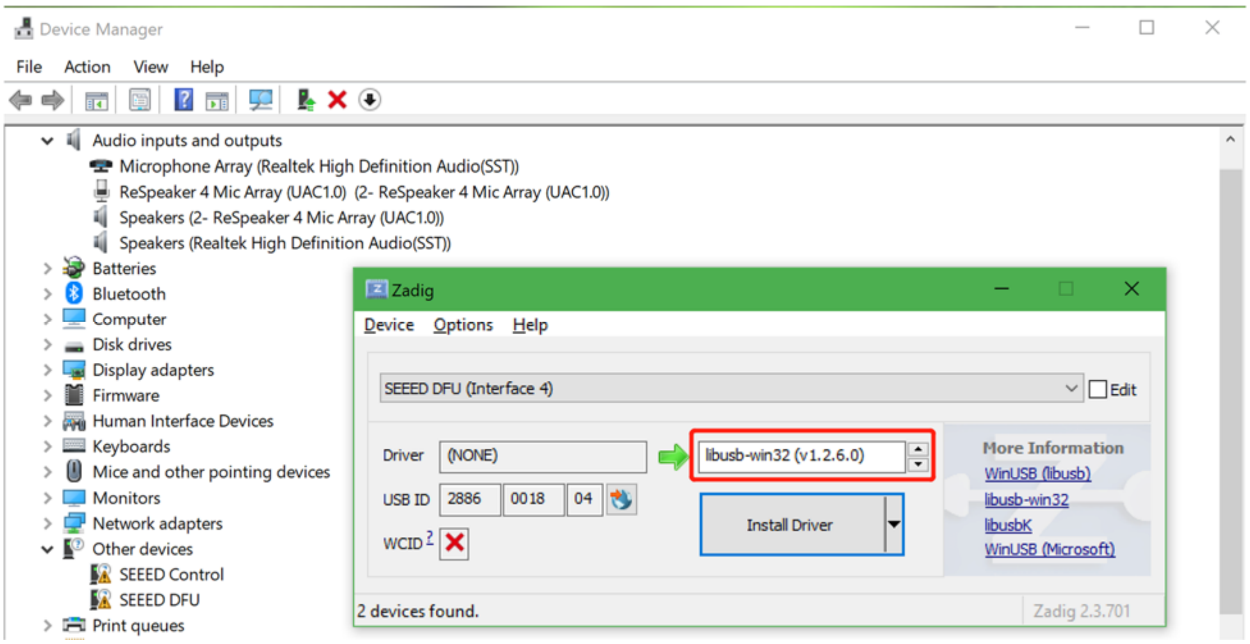
Channel1 Audio(Mic1 raw data):

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 an DSP parameters on Windows. We use a handy tool - Zadig to install the libusb-win32 driver for both `SEEED DFU` and `SEEED 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.

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

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

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

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

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

```
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]	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.customize()	custom mode, set each LED to its own color
0x20	[brightness]	pixel_ring.set_brightness()	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.

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

Here is the code of the usb_mic_array.py.

```
import time
from pixel_ring import pixel_ring


if __name__ == '__main__':
    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)
```

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

- Step 1. Download pixel_ring.

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

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

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.

```
git clone https://github.com/respeaker/usb_4_mic_array.git
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'

```
from tuning import Tuning
import usb.core
import usb.util
import time

dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)

if dev:
    Mic_tuning = Tuning(dev)
    print Mic_tuning.direction
    while True:
        try:
            print Mic_tuning.direction
            time.sleep(1)
        except KeyboardInterrupt:
            break
```

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

```
pi@raspberrypi:~/usb_4_mic_array $ sudo python doa.py
184
183
175
105
104
104
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.

```
git clone https://github.com/respeaker/usb_4_mic_array.git
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'

```
from tuning import Tuning
import usb.core
import usb.util
import time

dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)
```

```
#print dev
if dev:
    Mic_tuning = Tuning(dev)
    print Mic_tuning.is_voice()
    while True:
        try:
            print Mic_tuning.is_voice()
            time.sleep(1)
        except KeyboardInterrupt:
            break
```

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

```
pi@raspberrypi:~/usb_4_mic_array $ sudo python VAD.py
0
0
0
1
0
1
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.

```
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
```

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:

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

- Step 2, copy below code and paste on 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, 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.

```
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
import wave

RESPEAKER_RATE = 16000
RESPEAKER_CHANNELS = 6 # change base on firmwares, 1_channel_firmware.bin
as 1 or 6_channels_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()
```

- 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
import wave
import numpy as np

RESPEAKER_RATE = 16000
RESPEAKER_CHANNELS = 6 # change base on firmwares, 1_channel_firmware.bin as 1 or 6_channels_firmware.bin as 6
RESPEAKER_WIDTH = 2
# run getDeviceInfo.py to get index
RESPEAKER_INDEX = 3  # refer to input device id
CHUNK = 1024
RECORD_SECONDS = 3
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)
    # extract channel 0 data from 6 channels, if you want to extract channel 1, please change to [1::6]
    a = np.fromstring(data,dtype=np.int16)[0::6]
    frames.append(a.tostring())

print("* done recording")

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

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(1)
wf.setsampwidth(p.get_sample_size(p.get_format_from_width(RESPEAKER_WIDTH)))
wf.setframerate(RESPEAKER_RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

For Windows:

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

```
pip install pyaudio
```

- Step 2. Use get_index.py to get device index.

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

- Step 3. Modify the device index and channels of record.py and then extract voice.

```
C:\Users\XXX\Desktop>python record.py
* recording
* 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.

```
pip install pyaudio
```

- Step 2. Use get_index.py to get device index.

```
MacBook-Air:Desktop XXX$ python get_index.py
Input Device id  0  -  Built-in Microphone
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.

```
MacBook-Air:Desktop XXX$ python record.py
2018-03-24 14:53:02.400 Python[2360:16629] 14:53:02.399 WARNING:  140: This
application, or a library it uses, is using the deprecated Carbon Component
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.

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

```
make
```

- 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

```
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/baf08bc0dac02ba17f3c9120f5dd1cf74aec4c5446
3bc85cf9
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: Do you have the example for Raspberry alexa application?

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: Do you have the example for Mic array v2.1 with ROS system?

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

Q5: How to enable 3.5mm audio port to receive the signal as well as usb port?

A5: Please download the new firmware and burn the XMOS by following How to update firmware.

Q6: #include "portaudio.h" Error when run "sudo pip install pyaudio".

A6: Please run below command to solve the issue.

```
sudo apt-get install portaudio19-dev
```

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.