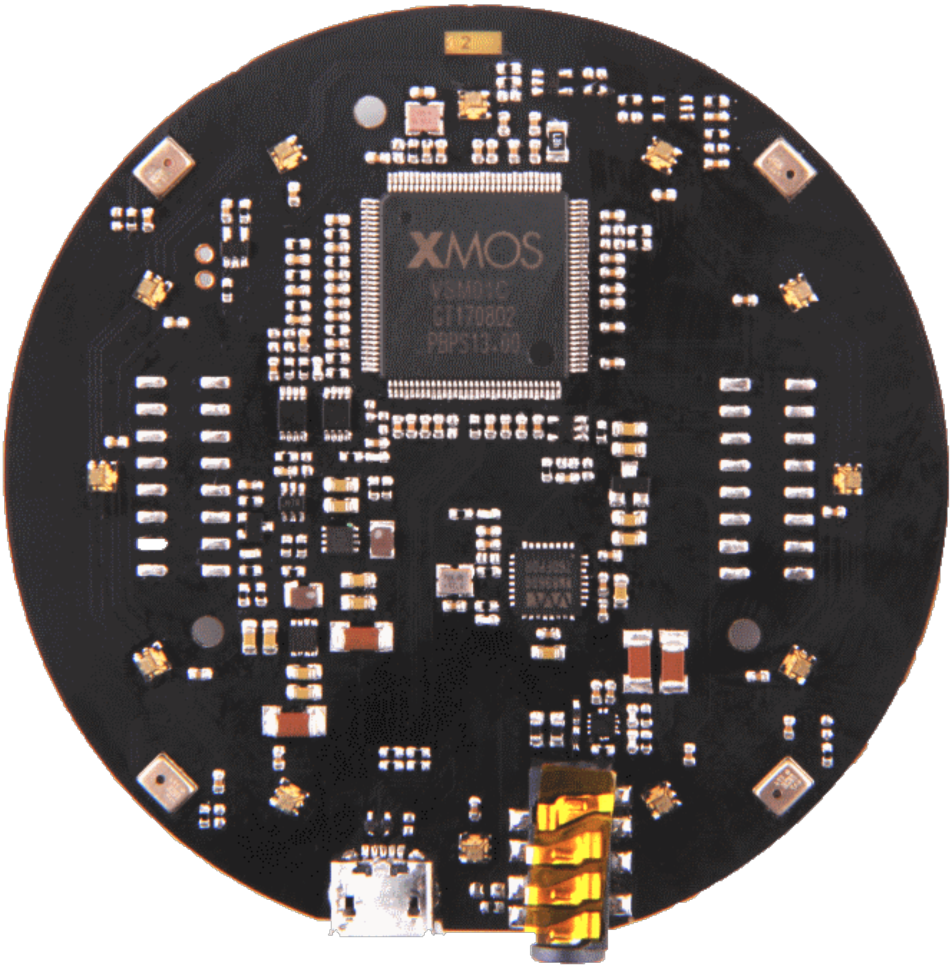
# ReSpeaker Mic Array v2.0



The ReSpeaker Mic Array v2.0 is an upgrade to the original ReSpeaker Mic Array v1.0 [https://www.seeedstudio.com/ReSpeaker-Mic-Array-Far-field-w%2F-7-PDM-Microphones--p-2719.html]. This upgraded version is based on XMOS's XVF-3000, a significantly higher performing chipset than the previously used XVSM-2000. This new chipset includes many voice recognition algorithms to assist in performance. The array can be stacked (connected) right onto the top of the original ReSpeaker Core to significantly improve the voice interaction performance.The microphones have also been improved in this version allowing significant performance improvements over the first generation mic array with only 4 microphones.

The ReSpeaker Mic Array v2.0 supports USB Audio Class 1.0 (UAC 1.0) directly. All major Operating System, including Windows,macOS, and Linux are compatible with UAC 1.0, allowing the mic array to function as a sound card without the ReSpeaker Core,while also retaining voice algorithms, such as DoA, BF, and AEC on those systems.

The ReSpeaker Mic Array v2.0 is a great solution for those who wish to add voice interface into their existing products or future products. It also works well as an entry point to higher level voice interface evaluation. The board allows some flexibility for customization upon request.

The ReSpeaker Mic Array v2.0 has two firmware versions available, one including speech algorithms and a second for raw voice data.

Get One Now [https://www.seeedstudio.com/ReSpeaker-Mic-Array-v2.0-p-3053.html]

Available on amazon.com [https://www.amazon.com/dp/B07D29L3Q1]

## Version

| Product Version | Changes | Released Date |
|---|---|---|
| ReSpeaker Mic Array v1.0 | Initial | Aug 15, 2016 |
| ReSpeaker Mic Array v2.0 | XVSM-2000 is EOL,change MCU to XVF-3000 and reduce the Mics from 7 to 4. | Jan 25, 2018 |

## Features

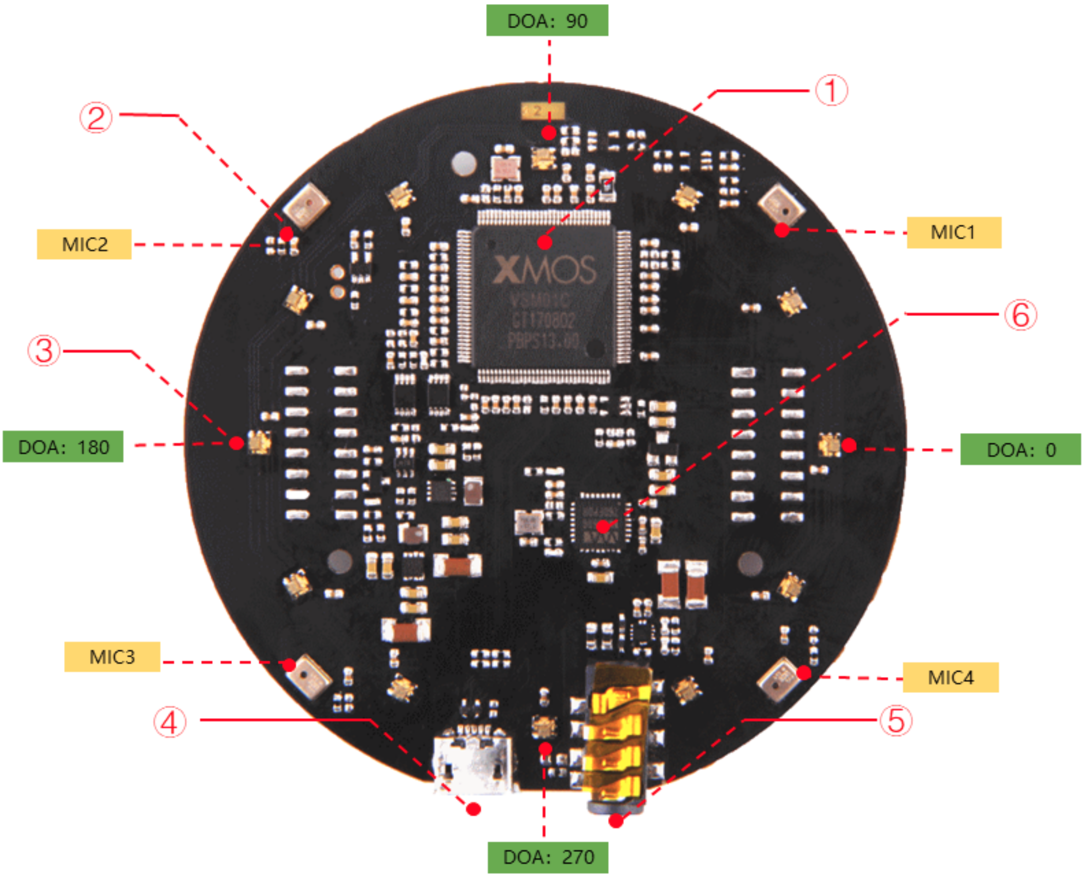
- Far-field voice capture
- Support USB Audio Class 1.0 (UAC 1.0)
- Four microphones array
- 12 programmable RGB LED indicators
- Speech algorithms and features
    - Voice Activity Detection
    - Direction of Arrival
    - Beamforming
    - Noise Suppression
    - De-reverberation
    - Acoustic Echo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

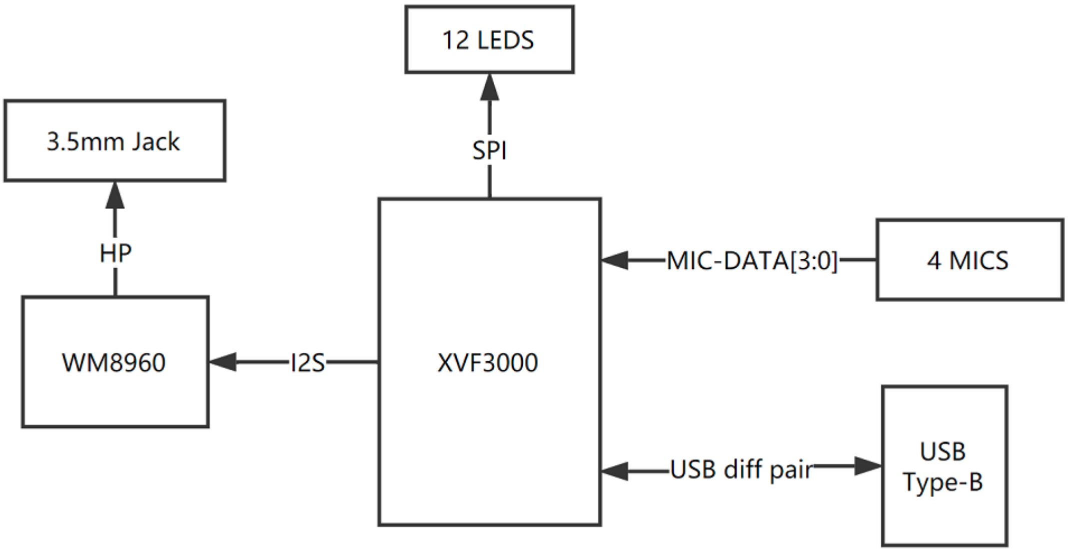
## Hardware Overview



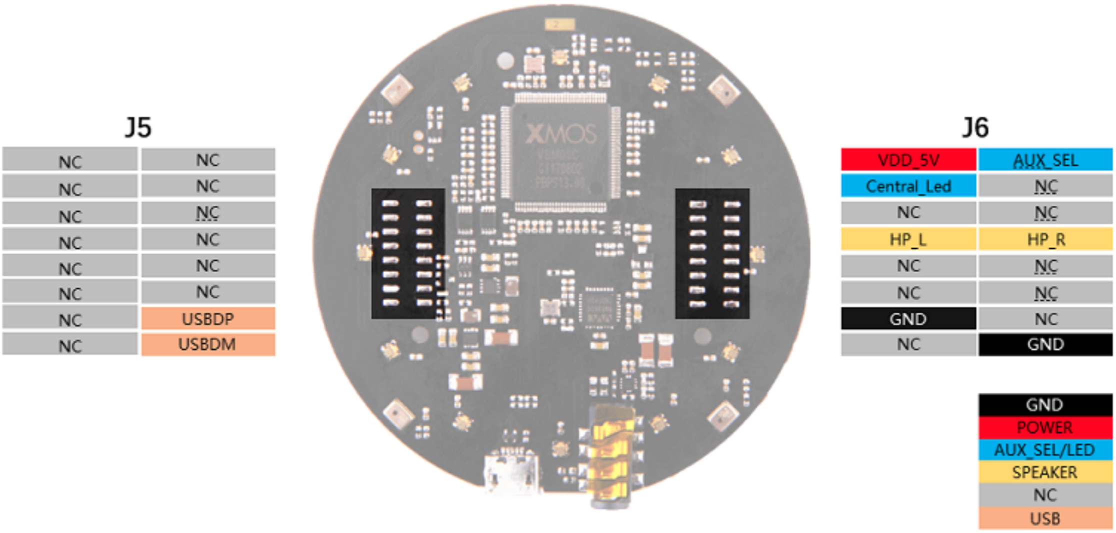
- ① **XMOS XVF-3000:** It integrates advanced DSP algorithms that include Acoustic Echo Cancellation (AEC), beamforming, dereverberation, noise suppression and gain control.
- ② **Digital Microphone:** The MP34DT01-M is an ultra-compact, lowpower, omnidirectional, digital MEMS microphone built with a 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



### Pin Map



J5

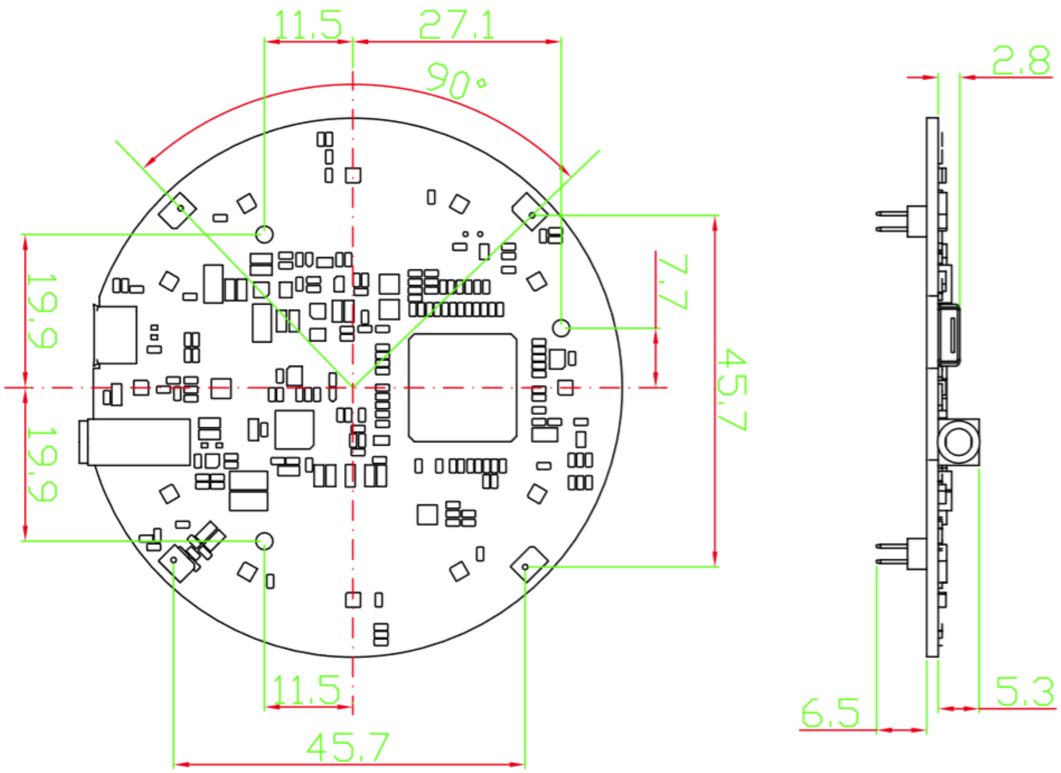
| | |
|---|---|
| NC | NC |
| NC | NC |
| NC | NC |
| NC | NC |
| NC | NC |
| NC | NC |
| NC | USBDP |
| NC | USBDM |

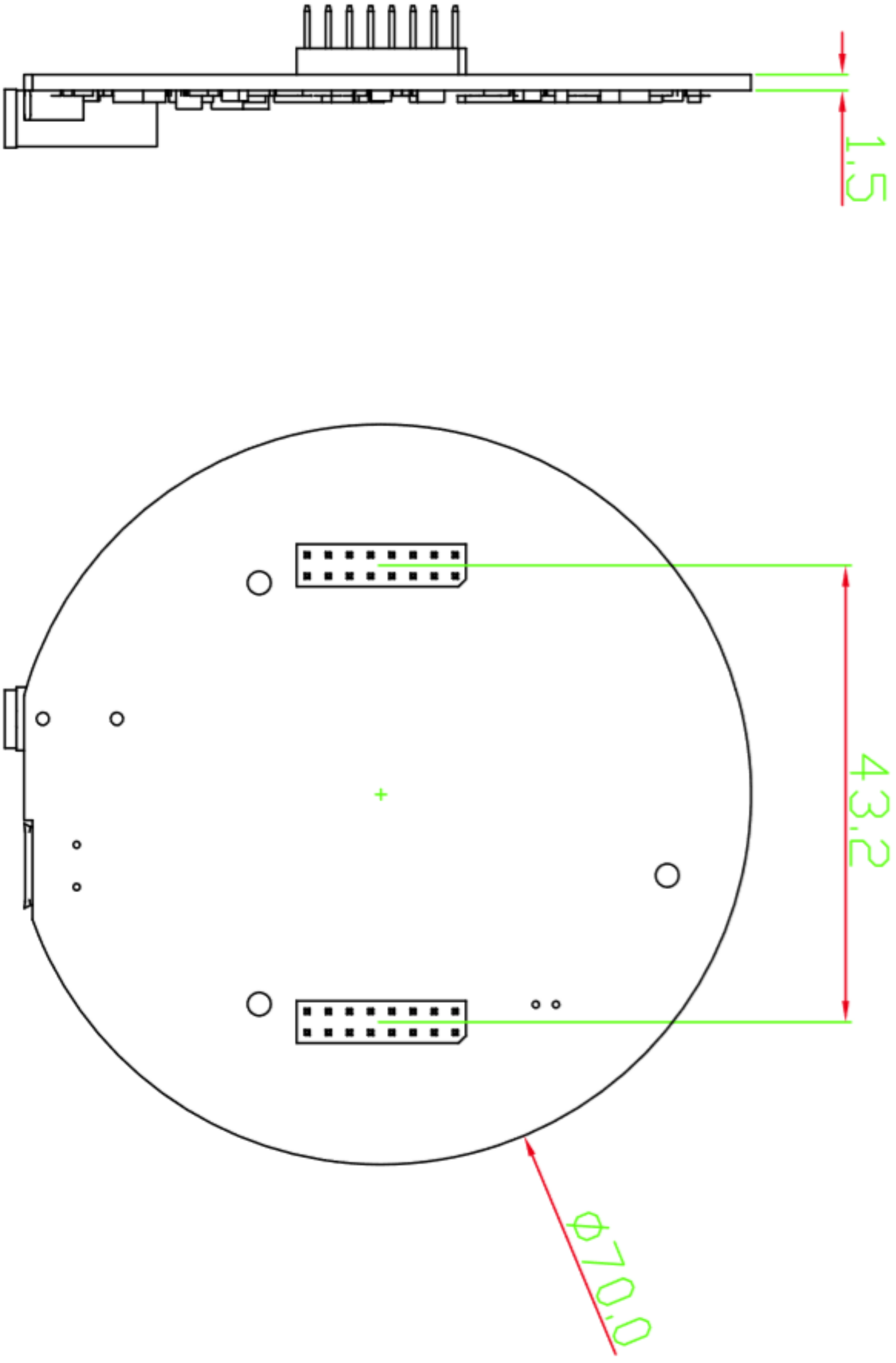
J6

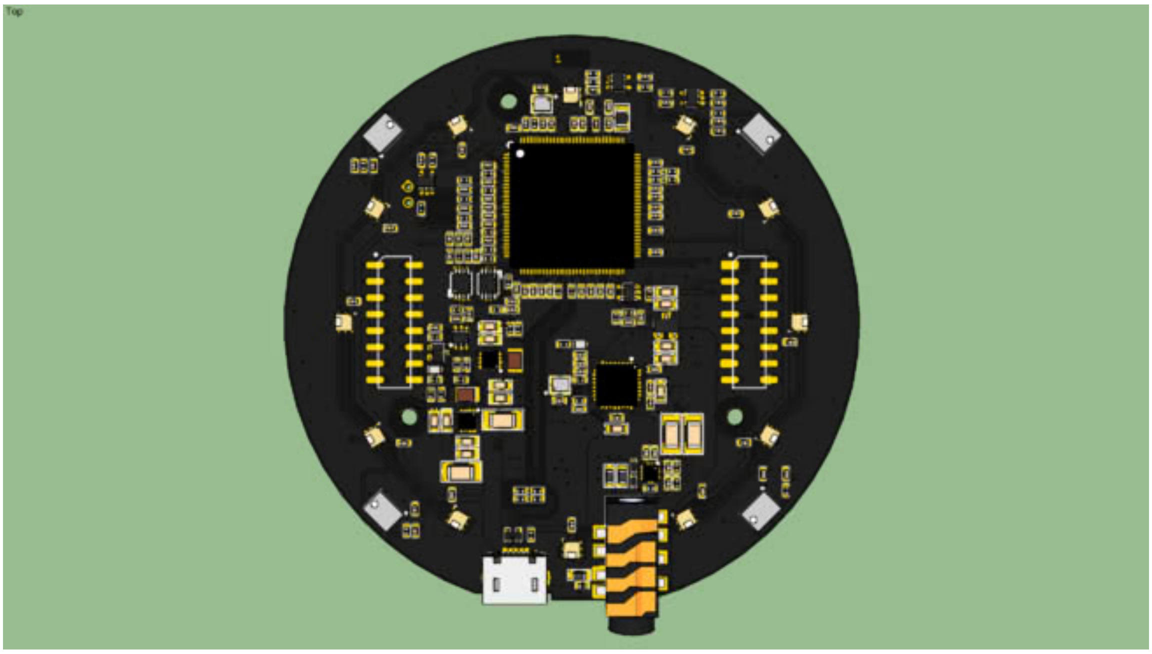
| | |
|---|---|
| VDD_5V | AUX_SEL |
| Central_Led | NC |
| NC | NC |
| HP_L | HP_R |
| NC | NC |
| NC | NC |
| GND | NC |
| NC | GND |

| |
|---|
| GND |
| POWER |
| AUX_SEL/LED |
| SPEAKER |
| NC |
| USB |

### Dimensions


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


1.5
43.2
Φ70.0



Click for 3D Model Viewer

3D Warehouse

## Applications

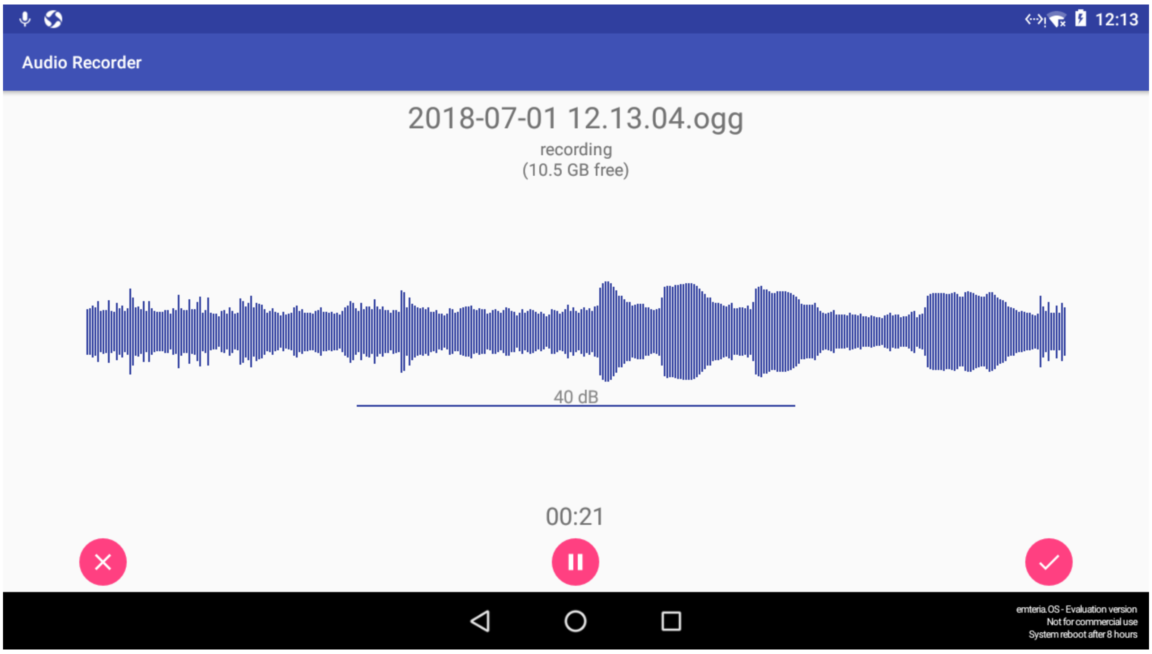
- USB Voice Capture
- Smart Speaker
- Intelligent Voice Assistant Systems
- Voice Recorders
- Voice Conferencing System
- Meeting Communicating Equipment
- Voice Interacting Robot
- Car Voice Assistant

- Other Voice Interface Scenarios

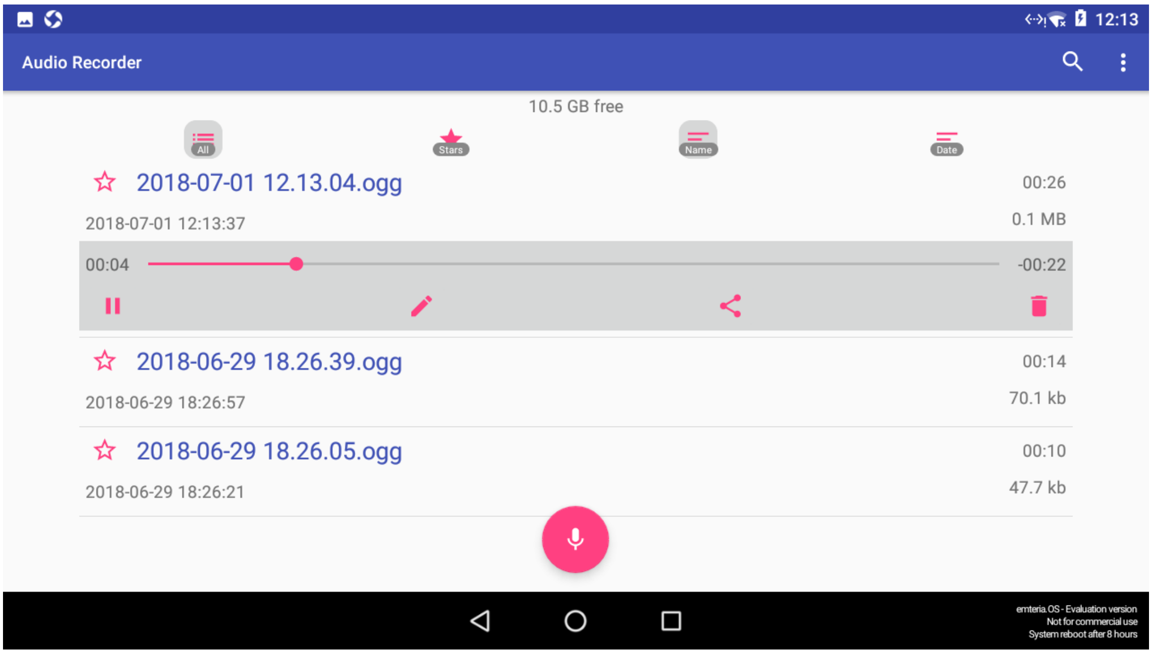
## Getting Started

**Note**
ReSpeaker Mic Array v2.0 is compatiable with Windows, Mac, Linux systems andriod.
The below scripts are tested on Python2.7.

For andriod, we tested it with emteria.OS [https://help.emteria.com/kb/emteria-os-installation](andriod 7.1) on Raspberry. We plug the mic array v2.0 to raspberry pi USB port and select the ReSpeaker mic array v2.0 as audio device. Here is the audio recording screen.



Here is the audio playing screen. We plug speaker to ReSpeaker mic array v2.0 3.5mm audio jack and hear what we record.



2018-07-01 12.13.04.ogg 00:26
2018-07-01 12:13:37 0.1 MB

2018-06-29 18.26.39.ogg 00:14
2018-06-29 18:26:57 70.1 kb

2018-06-29 18.26.05.ogg 00:10
2018-06-29 18:26:21 47.7 kb

## Update Firmware

There are 2 firmwares. One includes 1 channel data, while the other inlcudes 6 channels data (factory firmware). Here is the table for the differences.

| Firmware | Channels | Note |
|---|---|---|
| 1_channel_firmware.bin | 1 | Processed audio for ASR |
| 6_channels_firmware.bin | 6 | Channel 0: processed audio for ASR<br>Channel 1: mic1 raw data<br>Channel 2: mic2 raw data<br>Channel 3: mic3 raw data<br>Channel 4: mic4 raw data<br>Channel 5: merged playback |

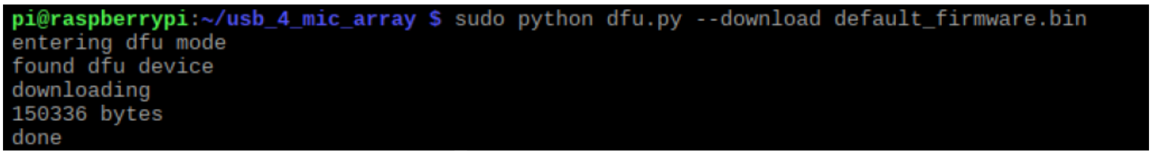
**For Linux:** The Mic array supports the USB DFU. We develop a python script dfu.py to update the firmware through USB.

```
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 char

# 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
```

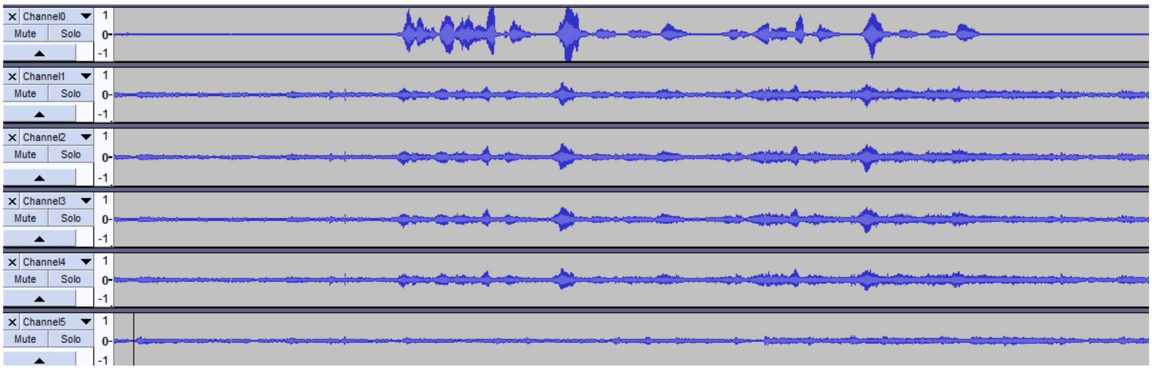
**For Windows/Mac:** We do not suggest use Windows/Mac and Linux vitual machine to update the firmware.

## 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 v2.0 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):

○ 0:00 / 0:13 ○

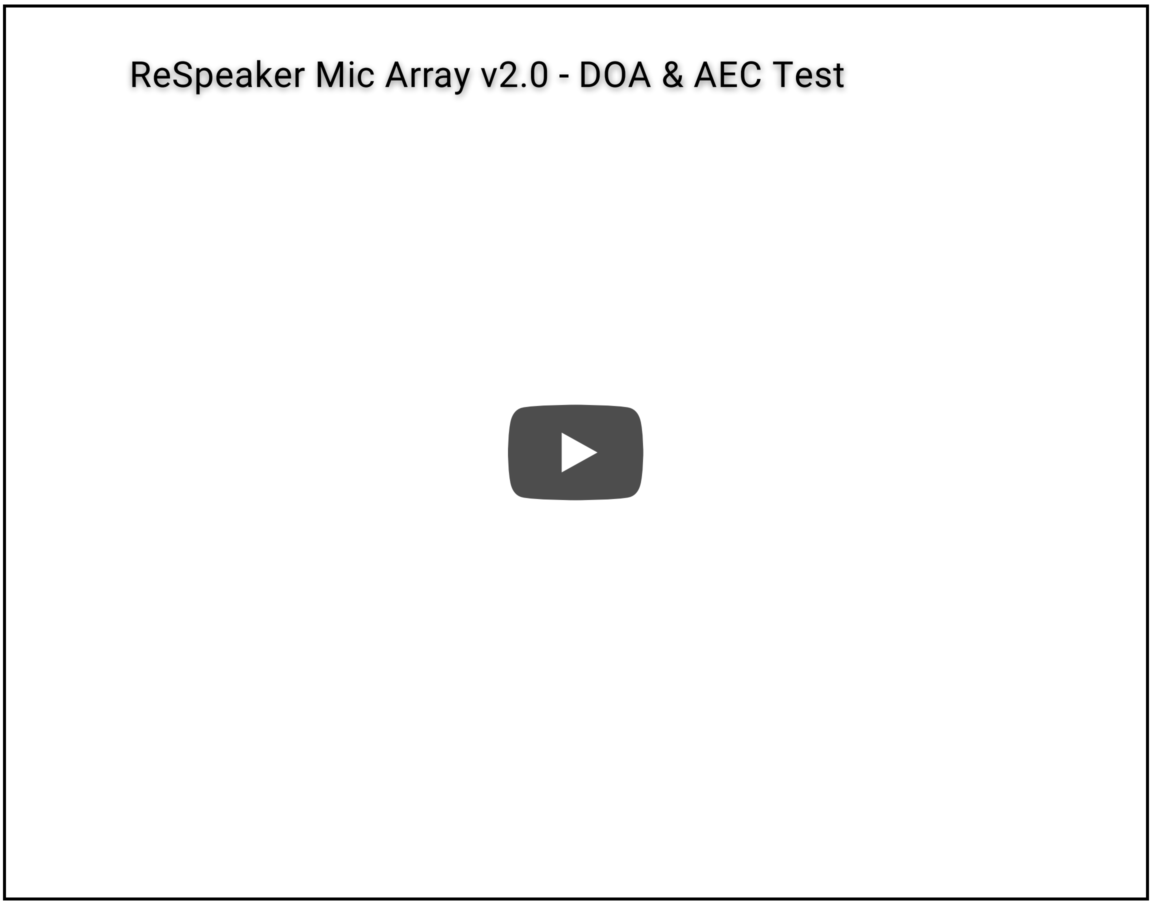
Channel1 Audio(Mic1 raw data):

○ 0:00 / 0:13 ○

Channel5 Audio(Playback data):

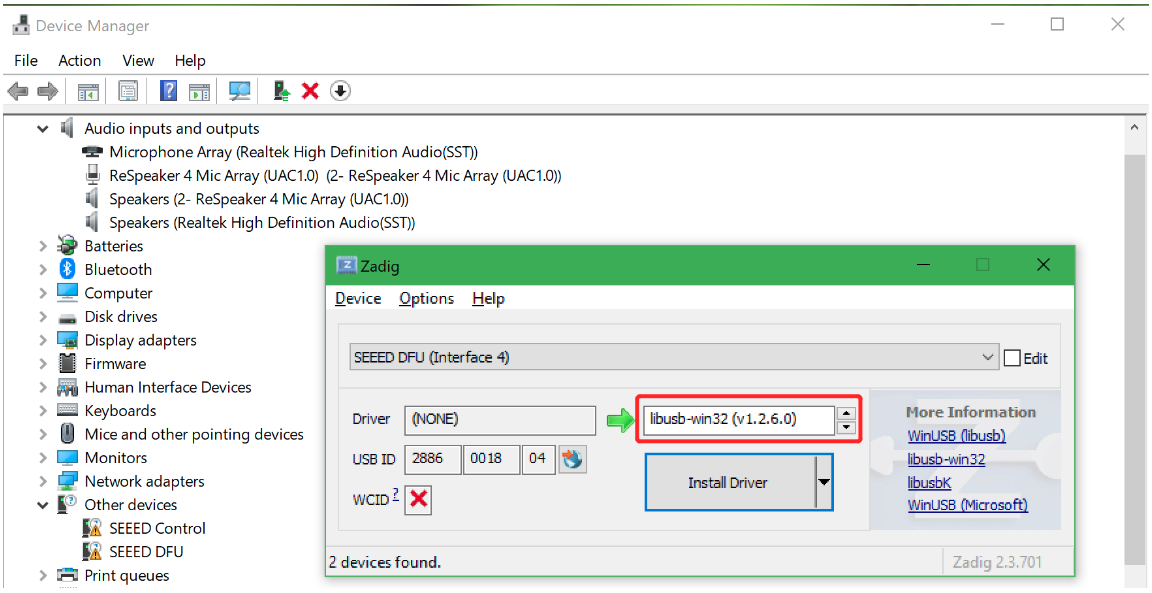
○ 0:00 / 0:13 ○

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 [http://zadig.akeo.ie/] to install the libusb-win32 driver for both `SEEED DFU` and `SEEED Control` (ReSpeaker Mic Array has 2 devices on Windows Device Manager).



SEEED DFU (Interface 4)

Driver (NONE) → libusb-win32 (v1.2.6.0)

USB ID 2886 0018 04

WCID ? ☒

2 devices found.

**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):

```
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 Mic Array V2'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 [https://github.com/pyusb/pyusb] and come out the usb_pixel_ring python library [https://github.com/respeaker/pixel_ring/blob/master/pixel_ring/usb_pixel_ring_v2.py].

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.u
```

Here are the usb_pixel_ring APIs.

| Command | Data | API | Note |
|---|---|---|---|
| 0 | [0] | pixel_ring.trace() | trace mode, LEDs changing d on VAD* and DOA* |
| 1 | [red, green, blue, 0] | pixel_ring.mono() | mono mode, set all RGB LED single color, for example Red(0xFF0000), Green(0x00F Blue(0x0000FF) |
| 2 | [0] | pixel_ring.listen() | listen mode, similar with trace 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 t own color |
| 0x20 | [brightness] | pixel_ring.set_brightness() | set brightness, range: 0x00~0 |
| 0x21 | [r1, g1, b1, 0, r2, g2, b2, 0] | pixel_ring.set_color_palette() | set color palette, for example pixel_ring.set_color_palette(0 0x00ff00) together with pixel_ring.think() |
| 0x22 | [vad_led] | pixel_ring.set_vad_led() | set center LED: 0 - off, 1 - on, 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 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__':
    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)
```

**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 [https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/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 [https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/DOA.py] with below code under usb_4_mic_array folder and run '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 [https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/VAD.py] with below code under usb_4_mic_array folder and run '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 [http://wiki.seeedstudio.com/ReSpeaker_Mic_Array_v2.0/#tuning] for more detail.

## Extract Voice

We use PyAudio python library [https://people.csail.mit.edu/hubert/pyaudio/] 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 arec
aplay -D plughw:1,0 -f cd test.wav # play, please use the aplay -l
arecord -D plughw:1,0 -f cd |aplay -D plughw:1,0 -f cd # record an
```

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 [https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/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('r
            print "Input Device id ", i, " - ", p.get_device_info_b
```

- 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 (h
```

- Step 5, change `RESPEAKER_INDEX = 2` to index number. Run python script record.py [https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2

/raw/master/res/record.py] to record a speech.

```
import pyaudio
import wave

RESPEAKER_RATE = 16000
RESPEAKER_CHANNELS = 6 # change base on firmwares, 1_channel_firmw
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(RESPEAKE
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_firmw
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 extra
    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(RESPEAKE
wf.setframerate(RESPEAKER_RATE)
wf.writeframes(b''.join(frames))
```

**For Windows:**

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

```
pip install pyaudio
```

- Step 2. Use get_index.py [https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/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 [https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/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 [https://github.com/SeeedDocument

/ReSpeaker_Mic_Array_V2/raw/master/res/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 [https://github.com/SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master/res/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:
* recording
* done recording
```

## Realtime Sound Source Localization and Tracking

ODAS [https://github.com/introlab/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.

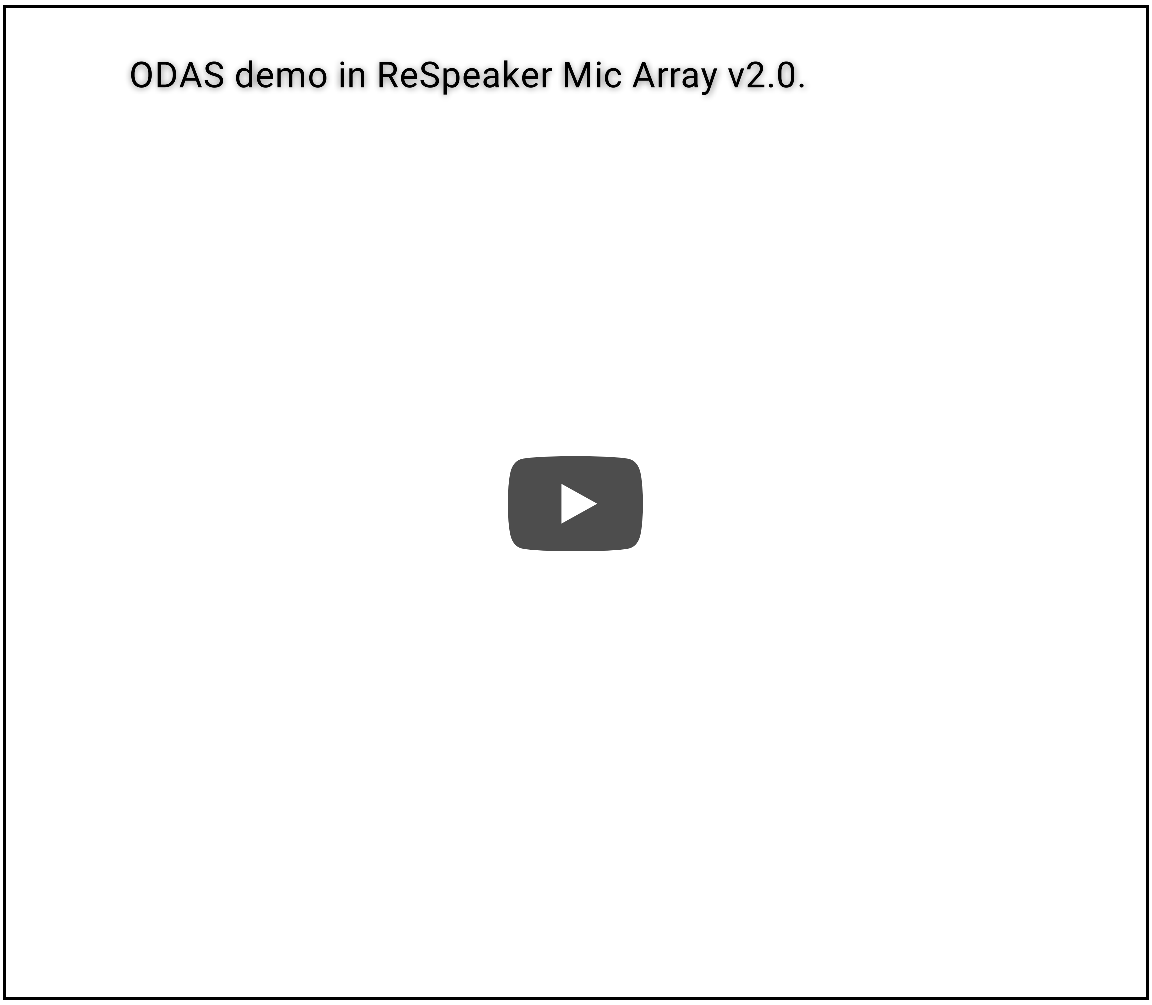
**For Linux:**

- Step 1. Get ODAS and build it.

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

- Step 2. Get ODAS Studio [https://github.com/introlab/odas_web/releases] and open it.

- Step 3. The odascore will be at **odas/bin/odaslive**, the **config file** is odas.cfg [https://raw.githubusercontent.com/respeaker/usb_4_mic_array/master/odas.cfg].
- Step 4. Upgrade mic array with 6_channels_firmware.bin which includes 4 channels raw audio data.



ODAS demo in ReSpeaker Mic Array v2.0.

**For Windows/Mac:** Please refer to ODAS [https://github.com/introlab/odas].

## 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
                                                            0 = Ad
                                                            1 = Fr
AECNORM             float   16  0.25    rw  Limit on norm of AEC f
AECPATHCHANGE       int 1   0   ro  AEC Path Change Detection.
                                                            0 = fa
                                                            1 = tr
AECSILENCELEVEL     float   1   1e-09   rw  Threshold for signal d
AECSILENCEMODE      int 1   0   ro  AEC far-end silence detection
                                                            0 = fa
                                                            1 = tr
AGCDESIREDLEVEL     float   0.99    1e-08   rw  Target power level
                                                            [-inf
AGCGAIN             float   1000    1   rw  Current AGC gain facto
                                                            [0 ..
AGCMAXGAIN          float   1000    1   rw  Maximum AGC gain facto
                                                            [0 ..
AGCONOFF            int 1   0   rw  Automatic Gain Control.
                                                            0 = OF
                                                            1 = ON
AGCTIME             float   1   0.1 rw  Ramps-up / down time-const
CNIONOFF            int 1   0   rw  Comfort Noise Insertion.
                                                            0 = OF
                                                            1 = ON
DOAANGLE            int 359 0   ro  DOA angle. Current value. Orie
ECHOONOFF           int 1   0   rw  Echo suppression.
                                                            0 = OF
                                                            1 = ON
FREEZEONOFF         int 1   0   rw  Adaptive beamformer updates.
                                                            0 = Ad
                                                            1 = Fr
FSBPATHCHANGE       int 1   0   ro  FSB Path Change Detection.
                                                            0 = fa
                                                            1 = tr
FSBUPDATED          int 1   0   ro  FSB Update Decision.
                                                            0 = fa
                                                            1 = tr
GAMMAVAD_SR         float   1000    0   rw  Set the threshold for
                                                            [-inf
GAMMA_E             float   3   0   rw  Over-subtraction factor of
```

**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/baf08bc0da
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 [https://github.com/alexa/avs-device-sdk/wiki/Raspberry-Pi-Quick-Start-Guide-with-Script] to do the voice interaction with alexa.

**Q4: Do you have the example for Mic array v2.0 with ROS system?**

A4: Yes, thanks for Yuki sharing the package for integrating ReSpeaker Mic Array v2 with ROS (Robot Operating System) Middleware [https://github.com/furushchev/respeaker_ros].

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

A5: Please download the new firmware [https://github.com/SeeedDocument

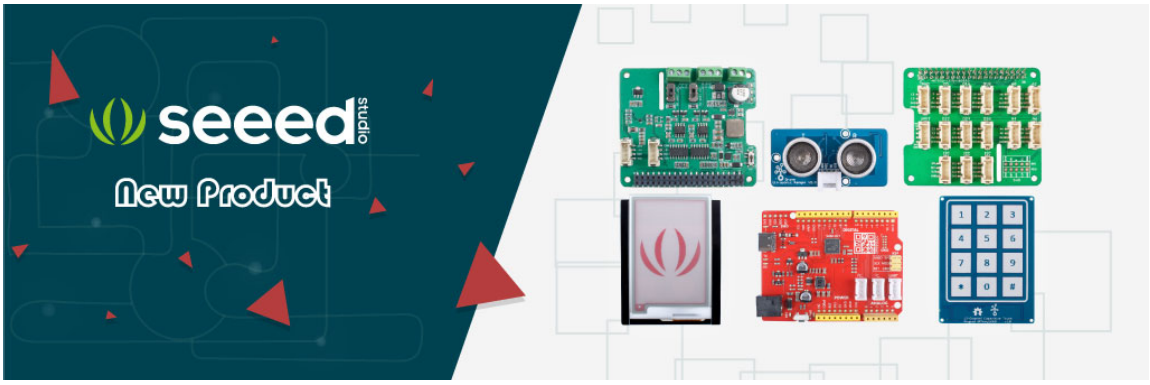
/ReSpeaker_Mic_Array_V2/raw/master/res/i2s_i1o2.bin] and burn the XMOS by following How to update firmware [http://wiki.seeedstudio.com /ReSpeaker_Mic_Array_v2.0/#update-firmware].

## Resource

- **[PDF]** ReSpeaker MicArray v2.0 Product Brief [https://github.com /SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master /res/ReSpeaker%20MicArray%20v2.0%20Product%20Brief.pdf]
- **[PDF]** ReSpeaker MicArray v2.0 3D Model [https://github.com /SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master /res/RESPEAKER%20MIC%20v2.0.pdf]
- **[SKP]** ReSpeaker MicArray v2.0 3D Model [https://github.com /SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master /res/Respeaker%20Microphone%20Array%20v2.0_20180316.skp.zip]
- **[STP]** ReSpeaker MicArray v2.0 3D Model [https://github.com /SeeedDocument/ReSpeaker_Mic_Array_V2/raw/master /res/RESPEAKER%20MIC-3D%20v2.0.stp.zip]
- **[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]
- **[Github]** ReSpeaker Mic Array v2 with ROS (Robot Operating System) Middleware [https://github.com/furushchev/respeaker_ros]

## Tech Support

Please submit any technical issue into our forum [http://forum.seeedstudio.com/].



[https://www.seeedstudio.com/act-4.html?utm_source=wiki&utm_medium=wikibanner&utm_campaign=newproducts]