Features

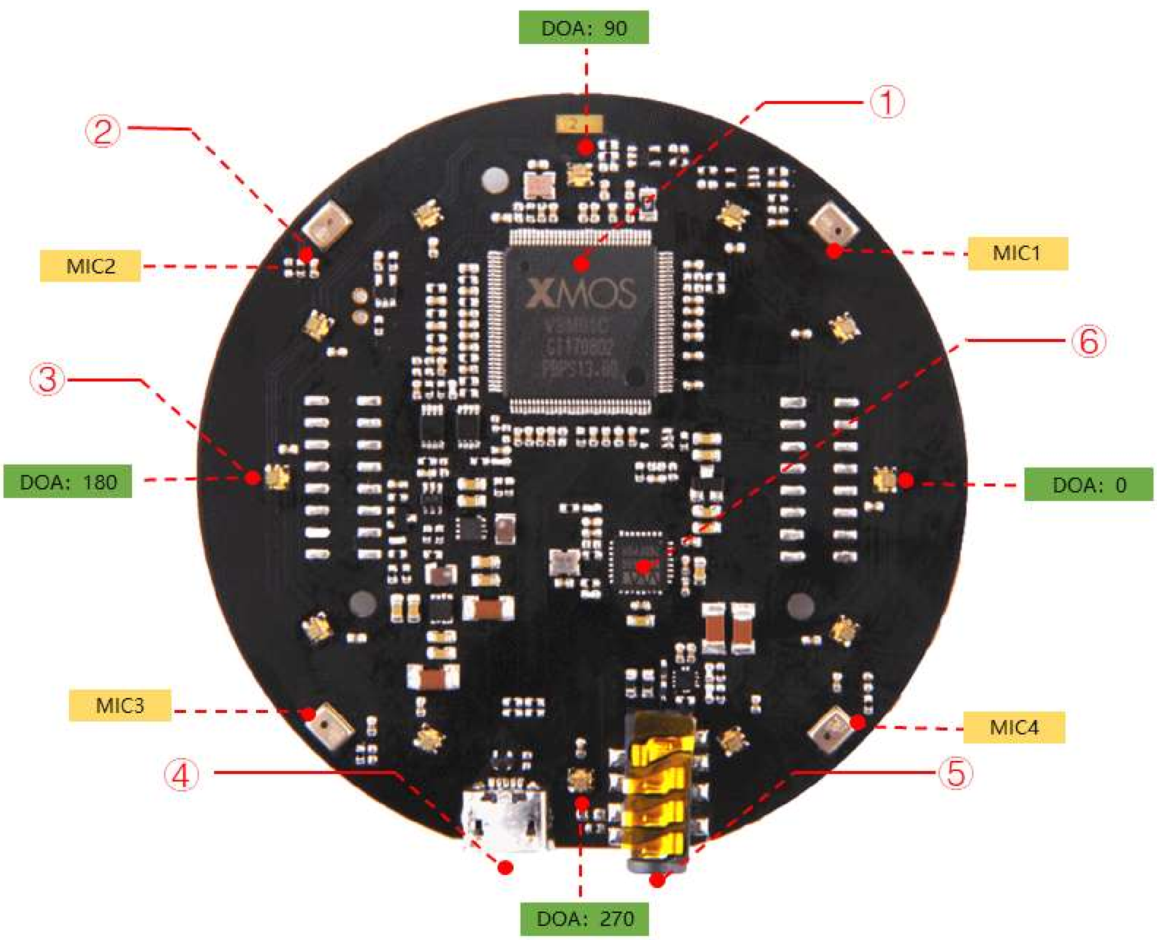
- Far-field voice capture
- Support USB Audio Class 1.0 (UAC 1.0)
- Four microphones array
- 12 programmable RGB LED indicators
- Speech algorithms and features
  - Voice Activity Detection
  - Direction of Arrival
  - Beamforming
  - Noise Suppression
  - De-reverberation
  - Acoustic Echo Cancellation

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: 63 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: 16Khz

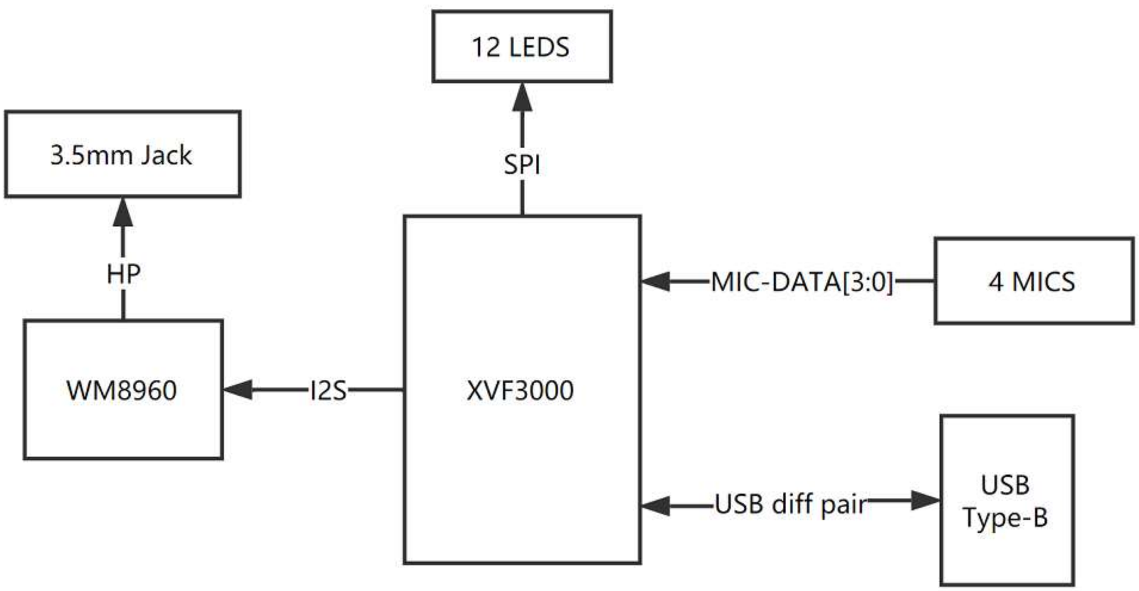
Hardware Overview



- 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.

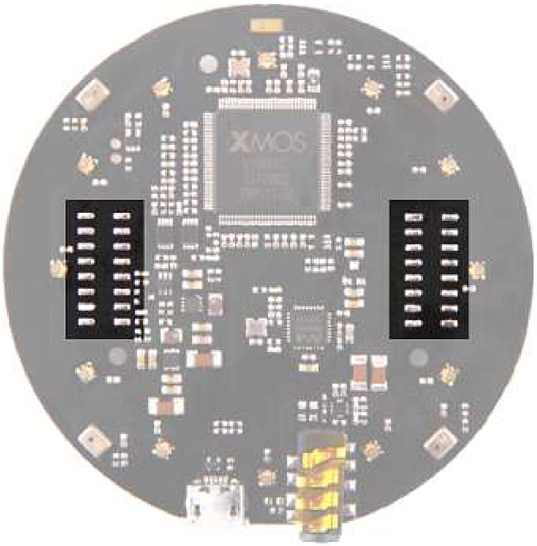
System Diagram



Pin Map

J5

NC	NC
NC	NC
NC	NC
NC	NC
NC	NC
NC	NC
NC	USBDP
NC	USBDM

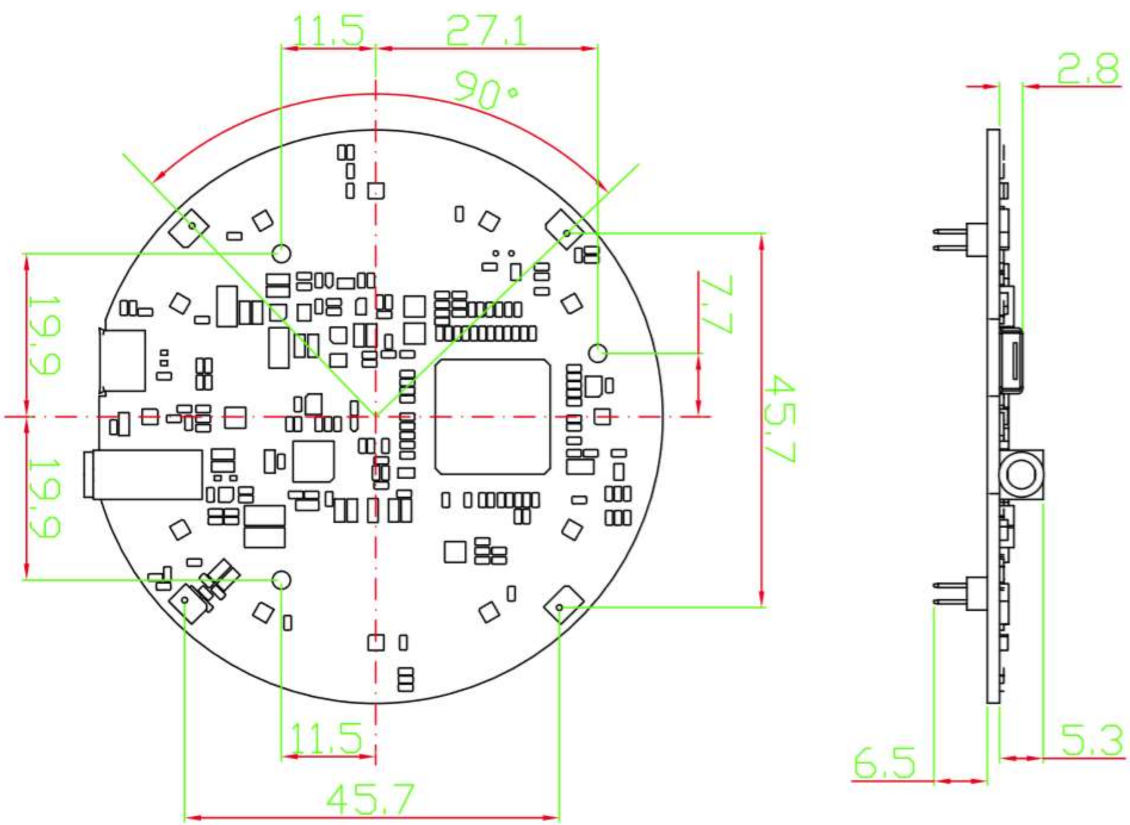


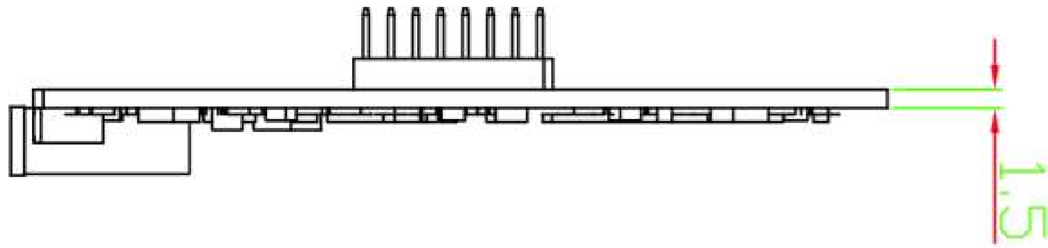
J6

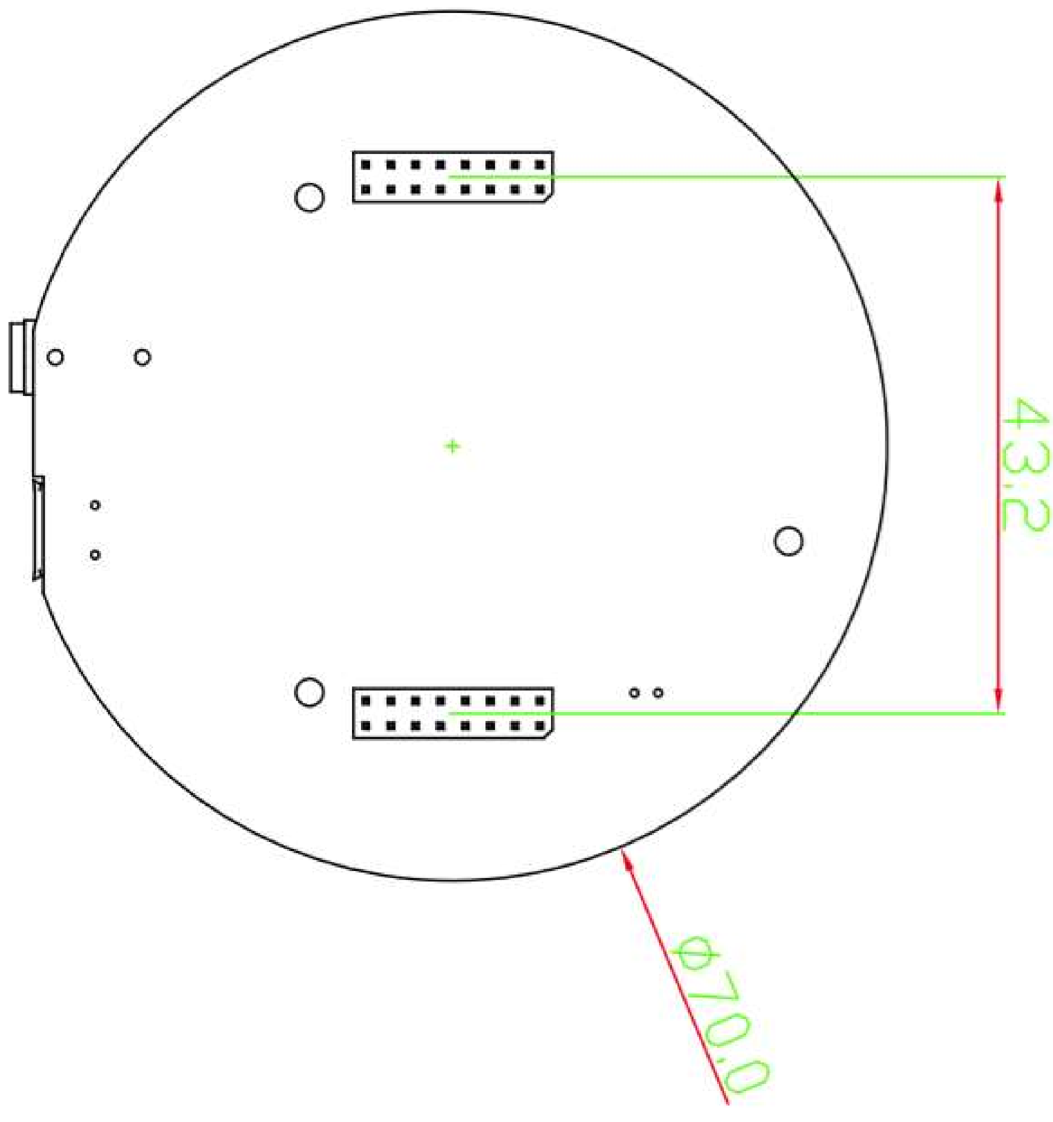
VDD_5V	AUX_SEL
Central_Led	NC
NC	NC
HP_L	HP_R
NC	NC
NC	NC
GND	NC
NC	GND

GND
POWER
AUX_SEL/LED
SPEAKER
NC
USB

Dimensions




1.5


43.2
Φ70.0

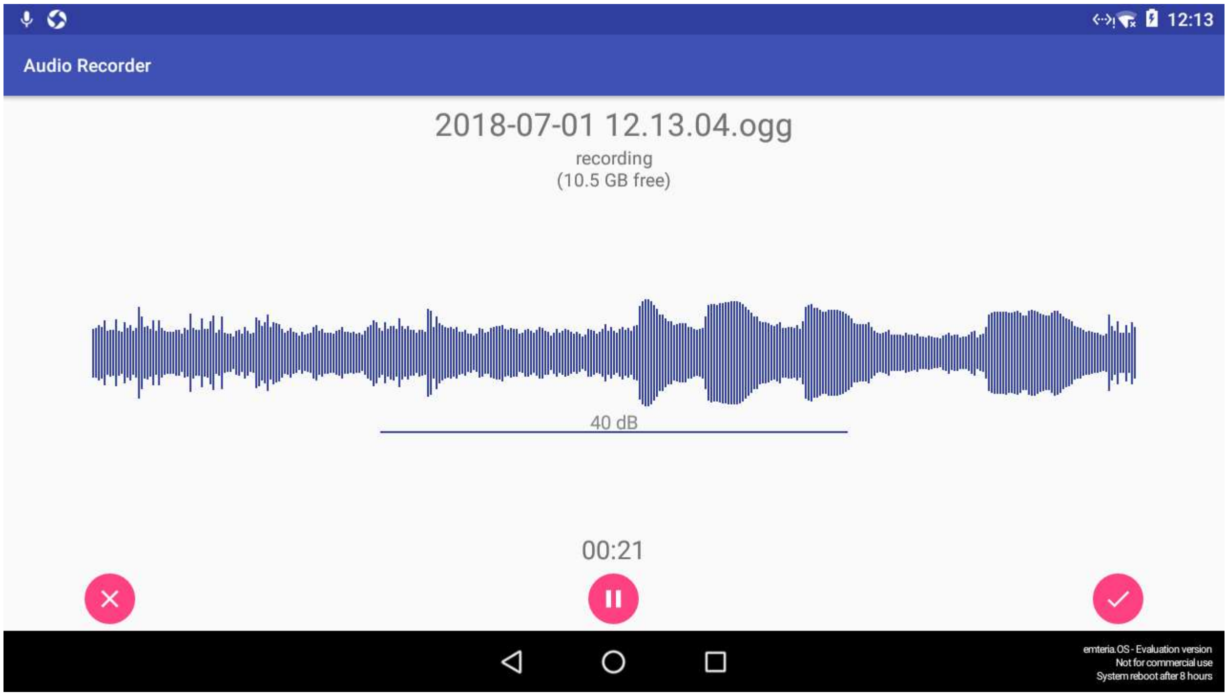
Applications

- 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

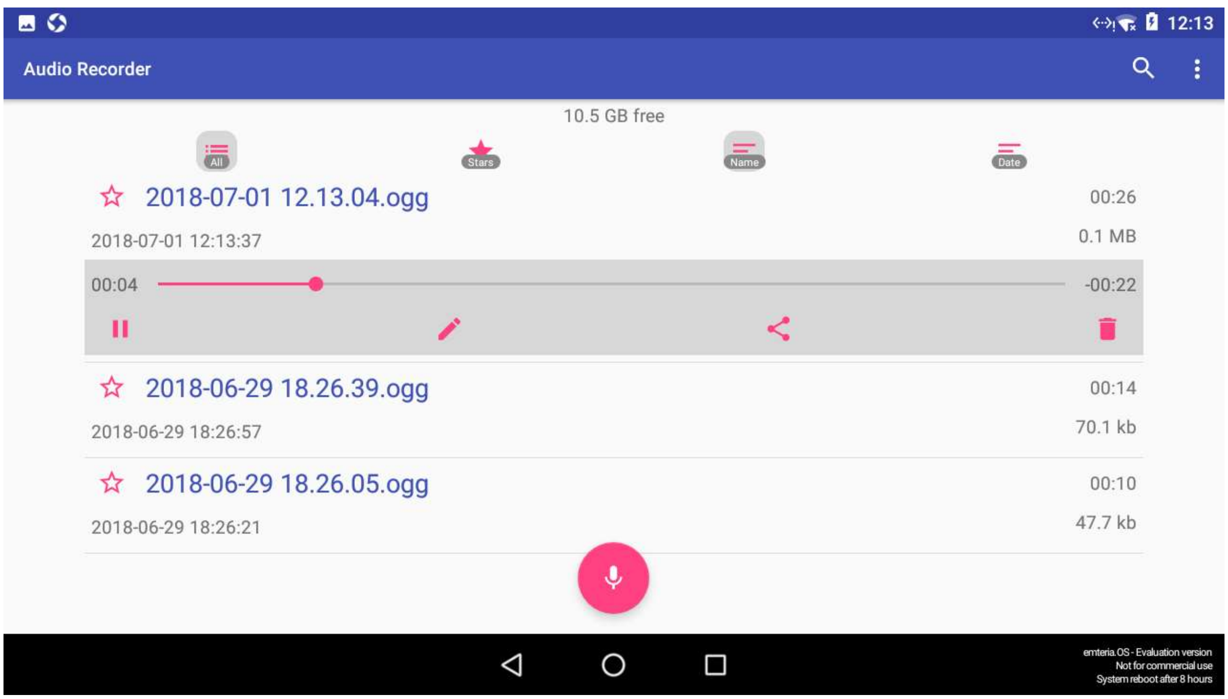
Getting Started

> **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(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.



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	Proces for ASF
6_channels_firmware.bin	6	Channe process for ASF Channe raw dat Channe raw dat Channe raw dat Channe raw dat Channe merged

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

# 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.

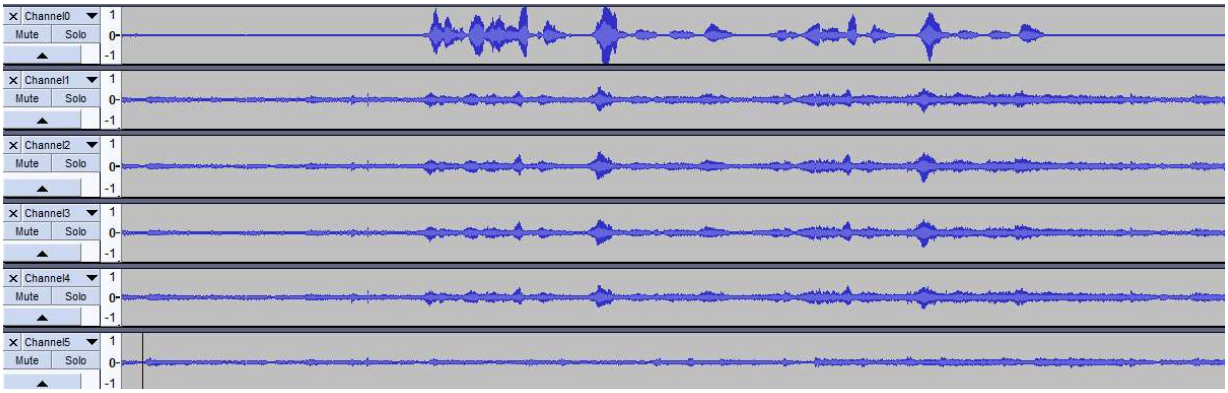
Out of Box Demo

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):

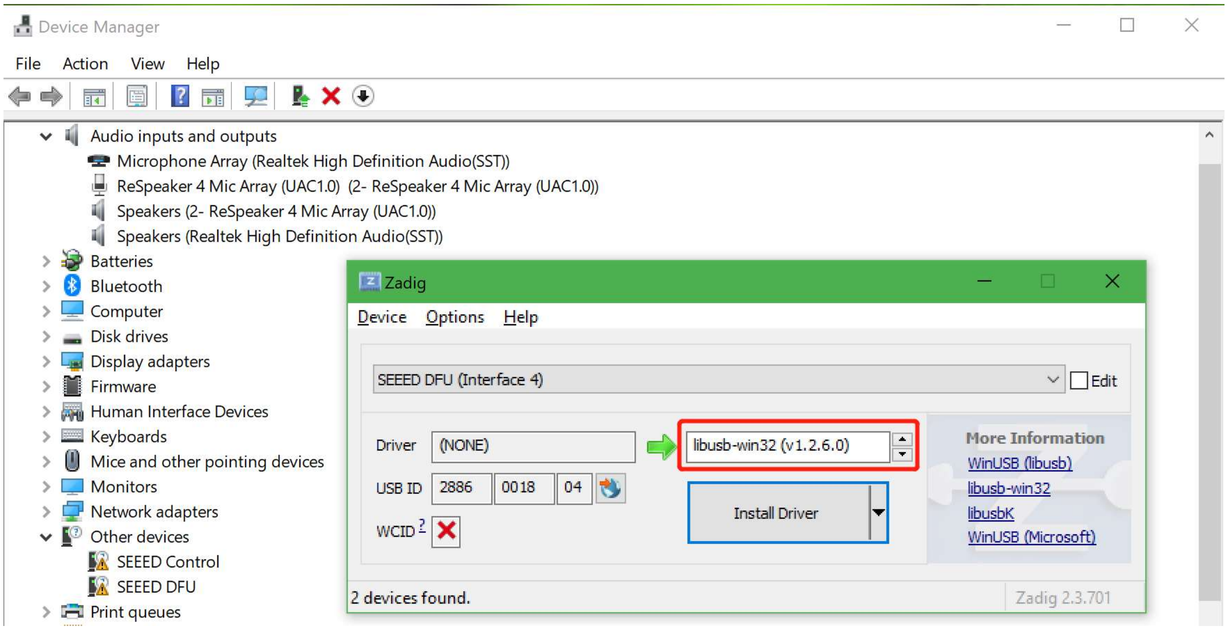
Channel1 Audio(Mic1 raw data):

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):

```
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 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.
```

Here are the usb_pixel_ring APIs.

Command	Data	API
0	[0]	pixel_ring.trac

Command	Data	API
1	[red, green, blue, 0]	pixel_ring.mon
2	[0]	pixel_ring.liste
3	[0]	pixel_ring.spea
4	[0]	pixel_ring.think
5	[0]	pixel_ring.spin
6	[r, g, b, 0] * 12	pixel_ring.cust
0x20	[brightness]	pixel_ring.set_
0x21	[r1, g1, b1, 0, r2, g2, b2, 0]	pixel_ring.set_
0x22	[vad_led]	pixel_ring.set_

0x23	[volume]	pixel_ring.set_
0x24	[pattern]	pixel_ring.char

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 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 'python DOA.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)
    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 '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 -
aplay -D plughw:1,0 -f cd test.wav # play, please use the aplay -l to c
arecord -D plughw:1,0 -f cd |aplay -D plughw:1,0 -f cd # record and pla
```

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('maxIn
            print "Input Device id ", i, " - ", p.get_device_info_by_ho
```

- 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,
```

- 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.b
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_WID
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:
* 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
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 and open it.
- Step 3. The odascore will be at odas/bin/odascore, the config file is odas.cfg. Please change odas.cfg based on your sound card number.

```
interface: {
    type = "soundcard";
```

```
    card = 1;
    device = 0;
}
```

- Step 4. Upgrade mic array with 6_channels_firmware.bin which includes 4 channels raw audio data.

For Windows/Mac: Please refer to 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 inhi
                                                        0 = Adaptat
                                                        1 = Freeze
AECNORM             float   16  0.25    rw  Limit on norm of AEC filter
AECPATHCHANGE       int 1   0   ro  AEC Path Change Detection.
                                                        0 = false (
                                                        1 = true (p
AECSILENCELEVEL     float   1   1e-09   rw  Threshold for signal detect
AECSILENCEMODE      int 1   0   ro  AEC far-end silence detection statu
                                                        0 = false (
                                                        1 = true (s
AGCDESIREDLEVEL     float   0.99    1e-08   rw  Target power level of t
                                                        [-inf .. 0]
AGCGAIN             float   1000    1   rw  Current AGC gain factor.
                                                        [0 .. 60] d
AGCMAXGAIN          float   1000    1   rw  Maximum AGC gain factor.
                                                        [0 .. 60] d
AGCONOFF            int 1   0   rw  Automatic Gain Control.
                                                        0 = OFF
                                                        1 = ON
AGCTIME             float   1   0.1 rw  Ramps-up / down time-constant i
```

```
CNIONOFF            int 1   0   rw  Comfort Noise Insertion.
                                                        0 = OFF
                                                        1 = ON
DOAANGLE            int 359 0   ro  DOA angle. Current value. Orientati
ECHOONOFF           int 1   0   rw  Echo suppression.
                                                        0 = OFF
                                                        1 = ON
FREEZEONOFF         int 1   0   rw  Adaptive beamformer updates.
                                                        0 = Adaptat
                                                        1 = Freeze
FSBPATHCHANGE       int 1   0   ro  FSB Path Change Detection.
                                                        0 = false (
                                                        1 = true (p
FSBUPDATED          int 1   0   ro  FSB Update Decision.
                                                        0 = false (
                                                        1 = true (F
GAMMAVAD_SR         float   1000    0   rw  Set the threshold for voice
                                                        [-inf .. 60
GAMMA_E             float   3   0   rw  Over-subtraction factor of echo
GAMMA_ENL           float   5   0   rw  Over-subtraction factor of non-
GAMMA_ETAIL         float   3   0   rw  Over-subtraction factor of echo
GAMMA_NN            float   3   0   rw  Over-subtraction factor of non-
GAMMA_NN_SR         float   3   0   rw  Over-subtraction factor of non-
                                                        [0.0 .. 3.0
GAMMA_NS            float   3   0   rw  Over-subtraction factor of stat
GAMMA_NS_SR         float   3   0   rw  Over-subtraction factor of stat
                                                        [0.0 .. 3.0
HPFONOFF            int 3   0   rw  High-pass Filter on microphone sign
                                                        0 = OFF
                                                        1 = ON - 70
                                                        2 = ON - 12
                                                        3 = ON - 18
MIN_NN              float   1   0   rw  Gain-floor for non-stationary n
                                                        [-inf .. 0]
MIN_NN_SR           float   1   0   rw  Gain-floor for non-stationary n
                                                        [-inf .. 0]
MIN_NS              float   1   0   rw  Gain-floor for stationary noise
                                                        [-inf .. 0]
MIN_NS_SR           float   1   0   rw  Gain-floor for stationary noise
                                                        [-inf .. 0]
NLAEC_MODE          int 2   0   rw  Non-Linear AEC training mode.
                                                        0 = OFF
```

```
                                                        1 = ON - ph
                                                        2 = ON - ph
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 suppressio
                                                        0 = OFF
                                                        1 = ON
RT60                float   0.9 0.25    ro  Current RT60 estimate in se
RT60ONOFF           int 1   0   rw  RT60 Estimation for AES. 0 = OFF 1
SPEECHDETECTED      int 1   0   ro  Speech detection status.
                                                        0 = false (
                                                        1 = true (s
STATNOISEONOFF      int 1   0   rw  Stationary noise suppression.
                                                        0 = OFF
                                                        1 = ON
STATNOISEONOFF_SR   int 1   0   rw  Stationary noise suppression for AS
                                                        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 (
                                                        1 = true (v
```

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/baf08bc0dac02ba
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 v2.0 with ROS system?

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