```
import sys
sys.path.append('/home/pi/Documents/18500/usb_4_mic_array')
from tuning import Tuning
import usb.core
import usb.util
import queue
import time
import pyaudio
import wave
import numpy as np

dev = usb.core.find(idVendor=0x2886, idProduct=0x0018)
mic_tuning = Tuning(dev)
userInputQ = Queue.queue()
speakerOutputQ = Queue.queue()

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 = 1  # refer to input device id
CHUNK = 1024
RECORD_SECONDS = 3
WAVE_OUTPUT_FILENAME = "output.wav"
p = pyaudio.PyAudio()

#operating on its own thread, while audio records
def callback(in_data, frame_count, time_info, status):
    if(mic_tuning.is_voice()):
        print('voice')
       # queue.put_nowait((in_data, doa)) testing this

    #put audio in queue for signal processing
    userInputQ.put_nowait((in_data, doa))

    print(mic_tuning.direction) #got doa
    return(None, pyaudio.paContinue)

def playAudio(rate=16000, channels=1, width=2, spectrum=True):
        stream = p.open(
            format=p.get_format_from_width(width),
            channels=channels,
            rate=rate,
            output=True,
            # output_device_index=1,
            frames_per_buffer=CHUNK_SIZE,
        )
```

```
        while !speakerOutputQ.empty():
            stream.write(speakerOutputQ.get_nowait)

        stream.close()

def startIO(quit_event, userInputQ, speakerOutputQ):
    global userInputQ = userInputQ
    global speakerOutputQ = speakerOutputQ

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


    try:
        #handle recording when playing audio
        stream.start_stream()
        while True:
            if(!speakerOutputQ.empty()):
                stream.stop_stream()
                playAudio()

            #restart listening on mic
            stream.start_stream()
            time.sleep(.1)
    except quit_event.isSet():
        #cleanup
        stream.stop_stream()
        stream.close()
        p.terminate()
        pass

# frames = []

# for i in range(0, int(RESPEAKER_RATE / CHUNK * RECORD_SECONDS)):
#     data = stream.read(CHUNK)
#     print(mic_tuning.direction)
#     # 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())
```

```
# 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()

###TUNING INFO
# if dev:
#     Mic_tuning = Tuning(dev)
#     print (Mic_tuning.direction)
#     while True:
#         try:
#             print(Mic_tuning.direction)
#             time.sleep(1)
#         except KeyboardInterrupt:
#             break
# # def main():

#     quit_event = Event()
#     thread = Thread(target=listenOnMic, args=(quit_event,))
#     thread.start()

#     while True:
#         try:
#             time.sleep(1)
#         except KeyboardInterrupt:
#             print('Quit')
#             quit_event.set()
#             break
#     thread.join()

# if __name__ == '__main__':
#     main()
```