# Getting Started with Raspberry Pi

CAUTION

This wiki is wrote for ReSpeaker 2-Mics Pi HAT **v2**, to distinguish the v1 and v2 device, please refer to How to Distinguish ReSpeaker 2-Mics Pi HAT Hardware Revisions.

On the latest Raspberry Pi OS, the traditional method of driver installation (For v1 device) is no longer available and may lead to the following known issues:

- The desktop environment may be corrupted after installation.
- The ReSpeaker device may not able to be detected by `aplay` / `arecord`.

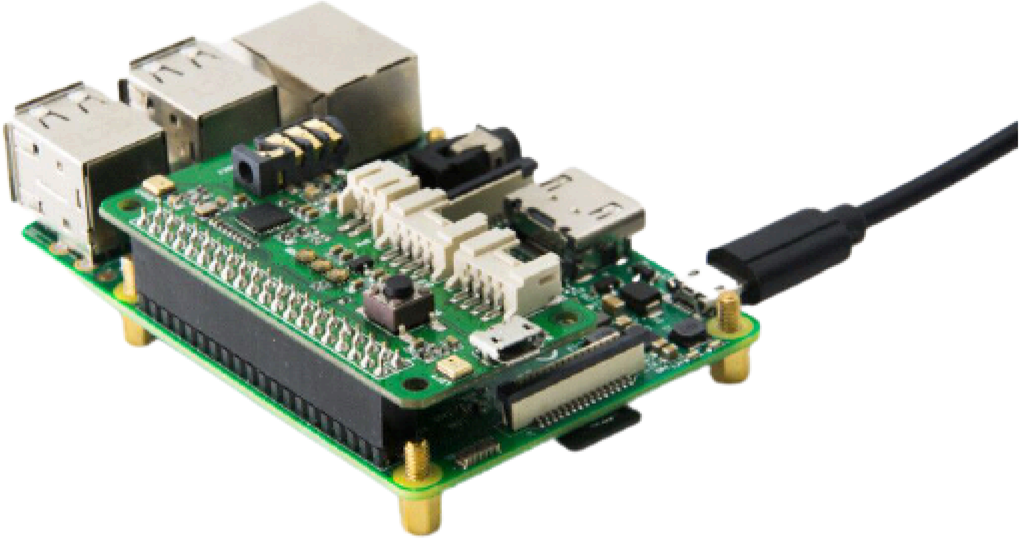
We have therefore re-released a new wiki on this issue, if you are using the more modern Raspberry Pi OS rather than previous releases, please follow these steps to get your ReSpeaker working.

## Driver installation and configuration

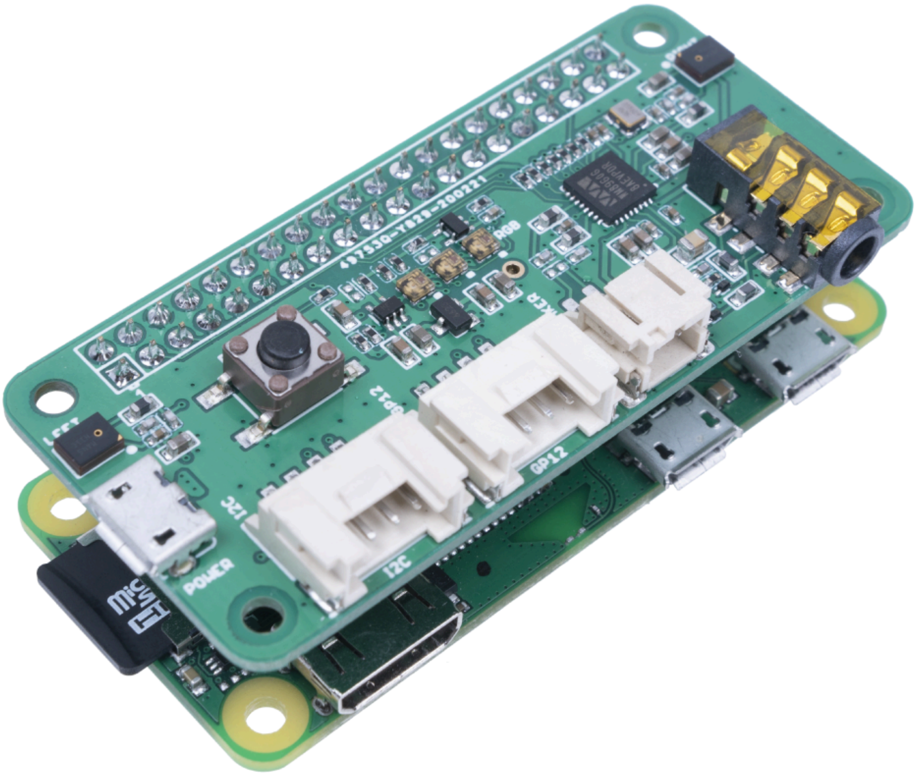
### 1. Connect ReSpeaker 2-Mics Pi HAT to Raspberry Pi

Mount ReSpeaker 2-Mics Pi HAT on your Raspberry Pi, make sure that the pins are properly aligned when stacking the ReSpeaker 2-Mics Pi HAT.

Raspberry Pi Connection



Raspberry Pi Zero Connection



## 2. Setup the driver on Raspberry Pi

Make sure that you are running the latest Raspberry Pi OS on your Pi. *(updated at 2024.11.19)*

Prepare for Raspberry Pi Zero W

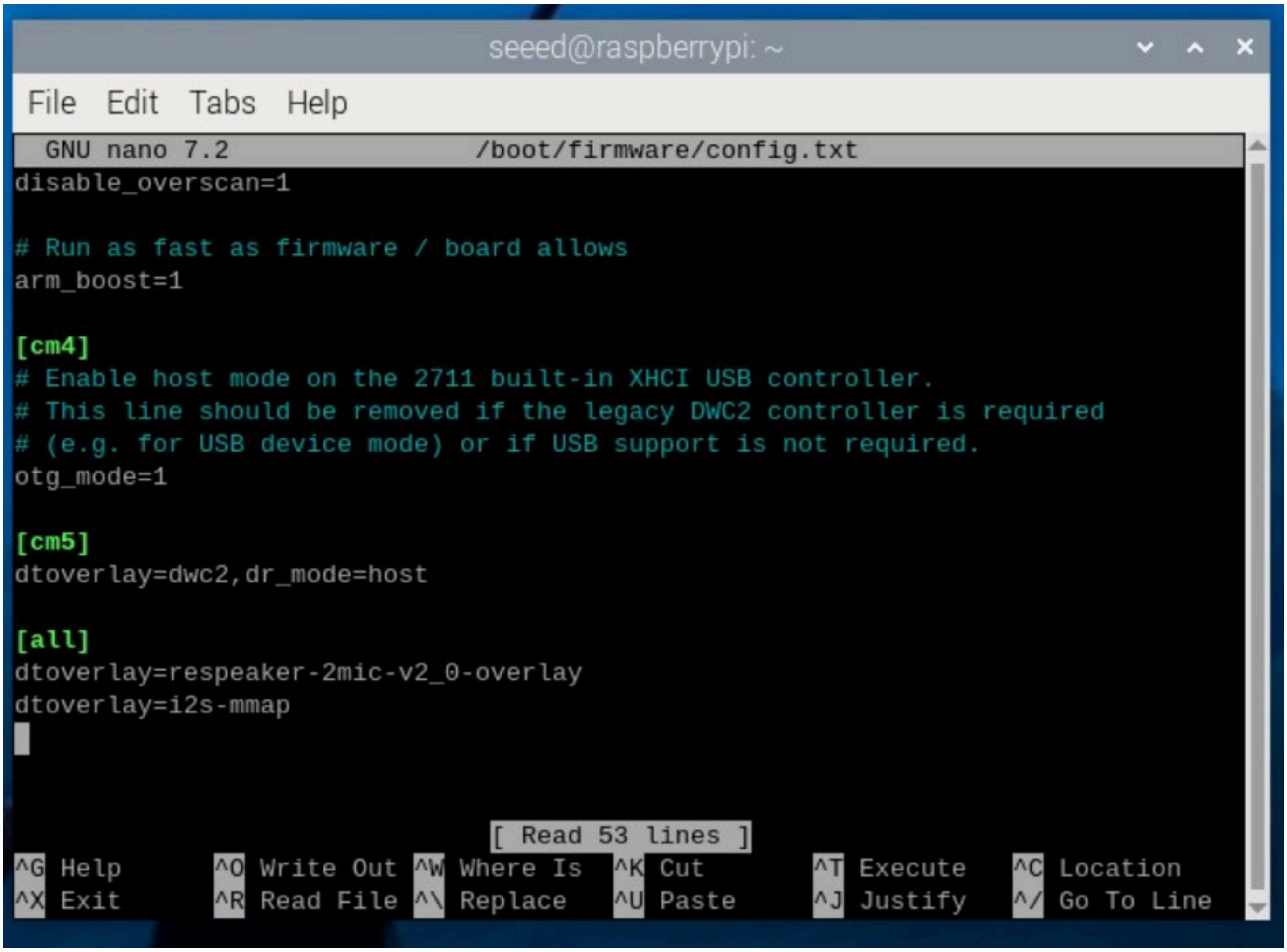
```
## Install kernel
sudo apt install flex bison libssl-dev bc build-essential
libncurses5-dev libncursesw5-dev linux-headers-6.6.51+rpt-rpi-v6
git clone --depth=1 --branch rpi-6.6.y
https://github.com/raspberrypi/linux.git

## Make target directory
mkdir ~/tlv320aic3x_i2c_driver
cd ~/tlv320aic3x_i2c_driver
## Copy code
cp ~/linux/sound/soc/codecs/tlv320aic3x.c ~/tlv320aic3x_i2c_driver/
cp ~/linux/sound/soc/codecs/tlv320aic3x.h ~/tlv320aic3x_i2c_driver/
cp ~/linux/sound/soc/codecs/tlv320aic3x-i2c.c
~/tlv320aic3x_i2c_driver/
## Modify Makefile
nano Makefile
-------------------
obj-m += snd-soc-tlv320aic3x-i2c.o
snd-soc-tlv320aic3x-i2c-objs := tlv320aic3x.o tlv320aic3x-i2c.o

KDIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

all:
        $(MAKE) -C $(KDIR) M=$(PWD) modules

clean:
        $(MAKE) -C $(KDIR) M=$(PWD) clean

install:
        sudo cp snd-soc-tlv320aic3x-i2c.ko /lib/modules/$(shell
uname -r)/kernel/sound/soc/codecs/
        sudo depmod -a

-------------------

## Compile the driver
make
sudo make install
sudo modprobe snd-soc-tlv320aic3x-i2c

## Check logs
lsmod | grep tlv320
dmesg | grep tlv320
```

- Step 1: Get Device Tree Source (DTS) for the ReSpeaker 2-Mics Pi HAT (V2.0), compile it and install the device tree overlay.

```
curl https://raw.githubusercontent.com/Seeed-Studio/seeed-linux-dtoverlays/refs/heads/master/overlays/rpi/respeaker-2mic-v2_0-overlay.dts -o respeaker-2mic-v2_0-overlay.dts
dtc -I dts respeaker-2mic-v2_0-overlay.dts -o respeaker-2mic-v2_0-overlay.dtbo
sudo dtoverlay respeaker-2mic-v2_0-overlay.dtbo
sudo cp respeaker-2mic-v2_0-overlay.dtbo /boot/firmware/overlays
```

- Step 2: Edit `/boot/firmware/config.txt` and add the following lines:

```
dtoverlay=respeaker-2mic-v2_0-overlay
dtoverlay=i2s-mmap
```



```
 GNU nano 7.2                 /boot/firmware/config.txt
disable_overscan=1

# Run as fast as firmware / board allows
arm_boost=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[cm5]
dtoverlay=dwc2,dr_mode=host

[all]
dtoverlay=respeaker-2mic-v2_0-overlay
dtoverlay=i2s-mmap


                              [ Read 53 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```

- Step 3: Reboot your Pi.

```
sudo reboot
```

- Step 4: Check if the device is detected by `aplay` / `arecord`.

The expected output for `aplay` should be:

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: vc4hdmi0 [vc4-hdmi-0], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: vc4hdmi1 [vc4-hdmi-1], device 0: MAI PCM i2s-hifi-0 [MAI PCM i2s-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: seeed2micvoicec [seeed2micvoicec], device 0: 1f000a4000.i2s-tlv320aic3x-hifi tlv320aic3x-hifi-0 [1f000a4000.i2s-tlv320aic3x-hifi
tlv320aic3x-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

The expected output for `arecord` should be:

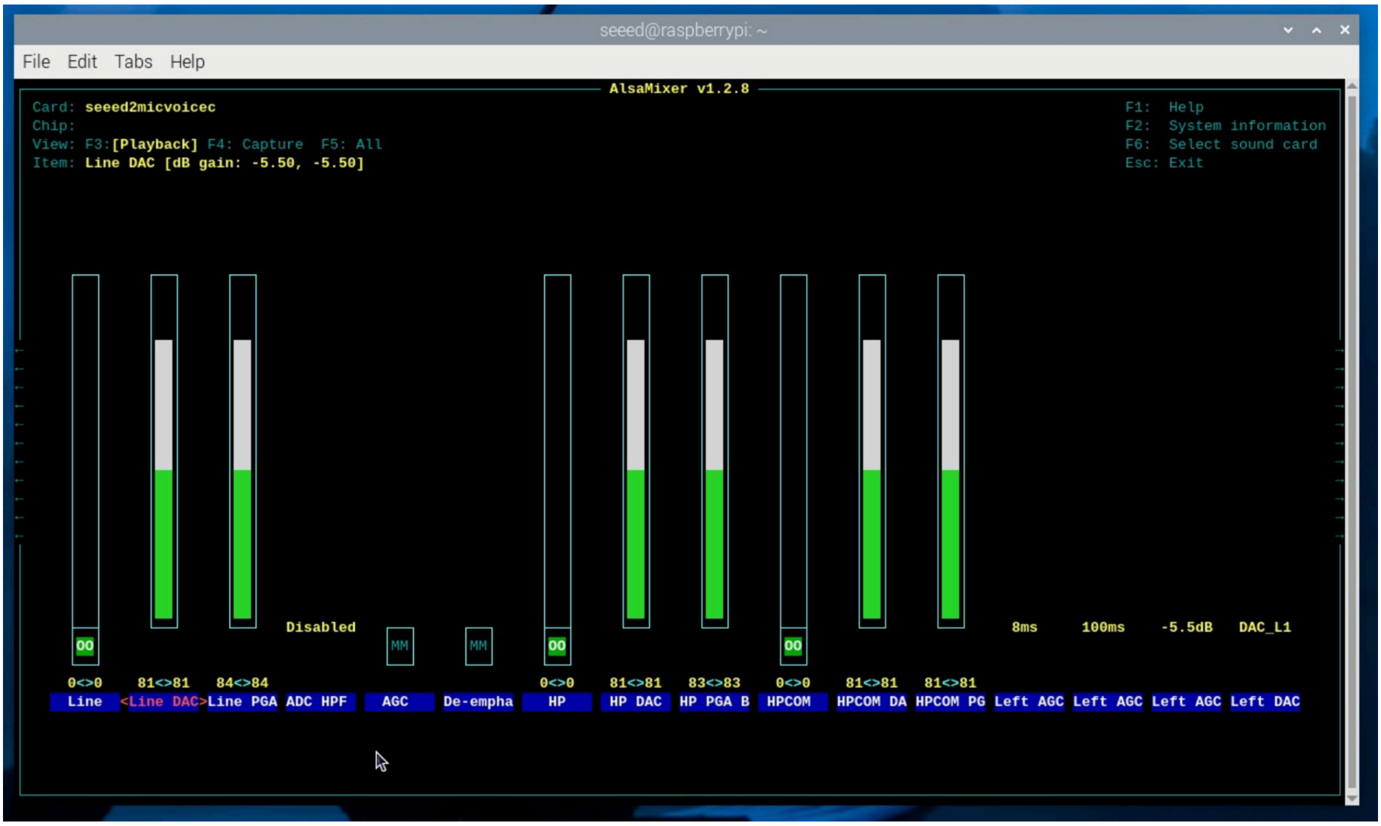
```
$ arecord -l
**** List of CAPTURE Hardware Devices ****
card 2: seeed2micvoicec [seeed2micvoicec], device 0: 1f000a4000.i2s-
tlv320aic3x-hifi tlv320aic3x-hifi-0 [1f000a4000.i2s-tlv320aic3x-hifi
tlv320aic3x-hifi-0]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

**Where the `card 2` is the index of the ReSpeaker 2-Mics Pi HAT, depending on your system this number may differ. To access the ReSpeaker in this example, you can use `arecord -D plughw:2,0` or `aplay -D plughw:2,0`.**

## 3. Configure sound settings and adjust the volume with alsamixer

`alsamixer` is a terminal user interface mixer program for the Advanced Linux Sound Architecture (ALSA) that is used to configure sound settings and adjust the volume.

```
alsamixer
```



```
AlsaMixer v1.2.8
Card: seeed2micvoicec                                  F1:  Help
Chip:                                                  F2:  System information
View: F3:[Playback] F4: Capture  F5: All               F6:  Select sound card
Item: Line DAC [dB gain: -5.50, -5.50]                 Esc: Exit

                    Disabled                                            8ms    100ms   -5.5dB  DAC_L1
 OO                            MM    MM    OO                OO
0<>0   81<>81  84<>84                     0<>0   81<>81 83<>83  0<>0   81<>81   81<>81
 Line <Line DAC>Line PGA ADC HPF  AGC  De-empha  HP  HP DAC HP PGA B HPCOM HPCOM DA HPCOM PG Left AGC Left AGC Left AGC Left DAC
```

The Left and right arrow keys are used to select the channel or device and the Up and Down Arrows control the volume for the currently selected device. Quit the program with ALT+Q, or by hitting the Esc key. More information

CAUTION

```
Please use the F6 to select seeed-2mic-voicecard device first.
```

# Usage overview

To get started, clone https://github.com/respeaker/mic_hat.git repository to your Raspberry Pi.

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

All the Python scripts, mentioned in the examples below can be found inside this repository. To install the necessary dependencies, from mic_hat repository folder, run

```
sudo apt-get install portaudio19-dev libatlas-base-dev
pip3 install -r requirements.txt
```

## APA102 LEDs

To use the LEDs, you need enable SPI interface first. To enable SPI interface, open the Raspberry Pi software configuration tool:

```
sudo raspi-config
```

Choose "3 Interface Options" -> "I4 SPI" to enable SPI interface. Then reboot your Raspberry Pi.

```
sudo reboot
```

Each on-board APA102 LED has an additional driver chip. The driver chip takes care of receiving the desired color via its input lines, and then holding this color until a new command is received.

```
cd mic_hat
python3 interfaces/pixels.py
```

0:00 / 0:12

## User Button

There is an on-board User Button, which is connected to GPIO_17.

⚠ CAUTION

```
The demo code from repository is not available for Raspberry Pi 5
due to `RPI.GPIO` incompatibility. But we have provided an
alternative demo code for Raspberry Pi 5 using `gpiozero` library.
```

### Non-Raspberry Pi 5 device

Execute the example script from the repository which you cloned at Step 4, and it should display "on" when you press the button:

```
$ python3 button.py
off
off
on
on
off
```

### Raspberry Pi 5 device

Copy the following code and save it to `~/button.py`:

```python
from gpiozero import DigitalInputDevice
from time import sleep

def main():
    pin = DigitalInputDevice(pin=17, pull_up=True)
    try:
        while True:
            print("on" if pin.value == 1 else "off")
            sleep(1)
    finally:
        pin.close()

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

It should also display "on" when you press the button:

```
$ python3 ~/button.py
off
on
on
```

```
off
on
off
```

> NOTE
>
> It does not work in a virtual environment, you need to exit it first:
>
> ```
> deactivate
> python3 ~/button.py
> ```

## Record sound with Python

We use PyAudio python library to record sound with Python.

First, run the following script to get the device index number of ReSpeaker:

```
cd mic_hit
python3 recording_examples/get_device_index.py
```

You will see the device ID as below.

```
Input Device id  1  -  seeed2micvoicec: 1f000a4000.i2s-tlv320aic3x-hifi
tlv320aic3x-hifi-0 (hw:2,0)
```

To record the sound, open `recording_examples/record.py` file with `nano`, `vim` or other text editor and change `RESPEAKER_INDEX = 2` to index number of ReSpeaker on your system. Then run python script `record.py` to make a recording:

```
python3 recording_examples/record.py
```

If you want to extract channel 0 data from 2 channels, have a look at the content of `record_one_channel.py`. For other channel X, please change [0::2] to [X::2].

```
python3 recording_examples/record_one_channel.py
```

To play the recorded samples you can either use aplay system utility, for example

```
aplay -f cd -D hw:2,0 output.wav # for Stereo sound
aplay -D plughw:2,0 output_one_channel.wav #for Mono sound from one
channel
```

Alternatively you can use recording_examples/play.py script to play the .wav files with PyAudio.

```
python3 recording_examples/play.py path-to-wav-file
```

Make sure to specify the right output device index in play.py - otherwise PyAudio will freeze!

# Tech Support & Product Discussion

Thank you for choosing our products! We are here to provide you with different support to ensure that your experience with our products is as smooth as possible. We offer several communication channels to cater to different preferences and needs.

Edit this page

*Last updated on **Apr 18, 2025** by **Jiahao***

**0 reactions**

☺

**7 comments** · 9 replies *– powered by giscus*

**Oldest** Newest