Getting up to Speed

Some useful links for learning all the terminology that we might be using:

Sampling

https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Audio_concepts (up until “Psychoacoustics”)

Linux’s ALSA Architecture

https://jackaudio.org/faq/multiple_devices.html (The method we have tried was number 3 on this list. Currently I am looking into trying the first option, since it is supposed to handle drift correction for us)
https://www.volkerschatz.com/noise/alsa.html (up until “Advanced configuration file features”)
https://alsa.opensrc.org/Udev#Identify_two_identical_audio_devices (NOTE: I have rewritten this page specifically for our hardware. Scroll down to read “Mapping Identical Sound Cards”
https://www.alsa-project.org/main/index.php/Asoundrc#Virtual_multi_channel_devices (Read “Virtual multi channel devices” if you would like to understand how I tied all of the sound cards together through ALSA. This is method number 3 listed on the jackaudio page)

RPi Storage

The free storage space on the RPi can be checked with the command,

```
df -Bh
```

Then, the following table (with equations) can be used to calculate the number of days of recordings that the pi can record. Note that this assumes 4 microphones are used, each recording at 48000 Hz with a bit depth of 16.

USER INPUTS			OUTPUTS	
Recording length [s]	60		Storage per hour [bytes]	46080000
Recordings per knock cycle	3		Storage per hour [GB]	0.04608
Number of microphones	4		Max deployment [hours]	434.0277778
Average interknock time [hour]	1.5		Max deployment [days]	18.08449074
RPi storage [GB]	20			

Common Commands

Linux:

File management commands

Alsa:

```
aplay -L
```

This command lists all of the current alsa devices. The non-indented text is the name of an audio device, and the indented text is information about that device.

```
aplay [path to audio file]
```

This command plays the content of an audio file through an output device (such as speakers you plug in). There is a flag for if you want to manually set the output device, but by default the pi will use the headphone jack (which we usually plug speakers into).

```
arecord -f S16_LE -r 48000 -c 1 -d [number of seconds to record] -D [alsa device name] [recording name]
```

This command will record audio for a given duration from a given audio card. The alsa device name must match the naming of cards given by the aplay -L command. The -f, -r, and -c flags specify format, sampling rate, and channels respectively. Our current microphones require the flags to be the values shown above.

Jack:

```
jack_lsp
```

This command lists all known ports associated with a JACK server.

```
jack_capture
```

This command runs a cli program for recording any audio signal from jack to audio formats supported by the libsndfile library (wav, aiff, ogg, flac, wavex, au, etc.)

SSH:

To copy a file or folder from the pi to the local machine, use

```
scp [-r] pi@raspberrypi.local:/home/pi/[path to file or folder on pi] /Users/[path to destination folder on local machine]
```

Where the flag -r is included if the path is to a folder.

Screen

To open a new screen session for remote access:

```
screen
```

To list all active screen sessions:

```
screen -list
```

To rejoin a screen session:

```
screen -d -r [session #]
```

To leave a screen session after joining:

```
Ctrl + A + D
```

To kill a session:

```
screen -X -S [session # you want to kill] quit
```

To scroll through the terminal:

```
Ctrl + A + [      and     esc     to exit
```

Mapping Identical Sound Cards

With identical sound cards, whenever one is plugged in it is assigned to a random software port, which on the pi is accessible as “hw:?,?”. Whenever one is plugged in, this mapping may change, meaning that there is no good way to identify and distinguish between cards. Luckily, the cards can be mapped based on their hardware port. Specific physical USB ports can be called upon, and as long as a card is always plugged into a certain port, it will always be tied to that port and we can identify it.

To see the current mappings, handled by ALSA:

```
# cat /proc/asound/cards
 4 [UA1A           ]: USB-Audio - EDIROL UA-1A
                      Roland EDIROL UA-1A at usb-0000:00:13.2-6.3, full speed
 5 [UA1A_1         ]: USB-Audio - EDIROL UA-1A
                      Roland EDIROL UA-1A at usb-0000:00:13.2-6.2, full speed
```

This is from a device with two identical sound cards plugged into it, labeled UA1A and UA1A_1.

Run the udevadm tool to output information about devices that are being plugged in / unplugged. Plug a USB sound card into the USB port you want to map. Find the DEVPATH argument:

```
# udevadm monitor --kernel --property --subsystem-match=sound
...
KERNEL[1305470077.926550] add      /devices/pci0000:00/0000:00:13.2/usb2/2-6/2-6.3/2-6.3:1.0/sound/card4 (sound)
UDEV_LOG=3
ACTION=add
DEVPATH=/devices/pci0000:00/0000:00:13.2/usb2/2-6/2-6.3/2-6.3:1.0/sound/card4
SUBSYSTEM=sound
SEQNUM=1434

KERNEL[1305470077.958511] add      /devices/pci0000:00/0000:00:13.2/usb2/2-6/2-6.3/2-6.3:1.0/sound/card4/pcmC4D0p
(sound)
...
KERNEL[1305470078.005760] add      /devices/pci0000:00/0000:00:13.2/usb2/2-6/2-6.3/2-6.3:1.0/sound/card4/pcmC4D0c
(sound)
...
KERNEL[1305470078.055299] add      /devices/pci0000:00/0000:00:13.2/usb2/2-6/2-6.3/2-6.3:1.0/sound/card4/controlC4
(sound)
...
KERNEL[1305470078.104122] change   /devices/pci0000:00/0000:00:13.2/usb2/2-6/2-6.3/2-6.3:1.0/sound/card4 (sound)
...
^C
```

Now, create a file like `/etc/udev/rules.d/85-my-usb-audio.rules` and use the following contents (this will require root permission):

```
SUBSYSTEM!="sound", GOTO="my_usb_audio_end"
ACTION!="add", GOTO="my_usb_audio_end"

DEVPATH=="/devices/pci0000:00/0000:00:13.2/usb2/2-6/2-6.3/2-6.3:1.0/sound/card
?", ATTR{id}="MY_AUDIO_CARD_1"

LABEL="my_usb_audio_end"
```

But replace the DEVPATH with the one gathered earlier from udevadm. Like done above, replace the card number with a “?”. Replace the ATTR{id} with a name of your choice.

Unplug and plug the sound card back into its original port, and udev should take care of the rest:

```
# cat /proc/asound/cards
 4 [MY_AUDIO_CARD_1  ]: USB-Audio - EDIROL UA-1A
                        Roland EDIROL UA-1A at usb-0000:00:13.2-6.3, full speed
```

Use the command `aplay -L` to check the audio devices currently plugged in. The format for calling upon a card to record looks like:

```
arecord -f S16_LE -r 48000 -c 1 -d 5 -D front:CARD=MY_AUDIO_CARD_1, DEV=0
testRecording.wav
```

Where the flags being used are for format, sample rate, number of channels, duration of recording, and device, respectively. These are the flags I have found that enable the Fifine audio cards we have to work properly.

alsa_in

It turns out that using a USB splitter with audio cards is problematic. When doing this bits of data are lost and the resulting audio is distorted, and it turns out it makes the audio higher pitched and garbled. We need to check if using a proper USB hub would work, or if we need to be using discrete USB ports.
Update 11/30: after using a proper usb hub (Anker 4-Port Ultra Slim), it seems like everything is working. However, not only is jack working for capture, but alsa is now working too. The audio recorded is good and is synced well. We need to do new metronome tests to determine how well each solution works.

Using discrete ports, I was able to get jack to record from multiple devices at once using the alsa_in and jack_capture interfaces:

```
alsa_in -d "hw:CARD=Device,DEV=0" -q 0 -r 48000 -c 1 -v
```

One terminal needed for each device to execute this command

```
jack_capture --port "*"
```

In another terminal

Note that I have not tested using udev to rename cards, and then feeding the new name into alsa_in. This should be tested to make sure there are no issues with sample rate or format.

We also need to test the auto-syncing capabilities of alsa_in with a long-term test, such as another metronome test.

Current Status of the pi

OS

- Currently I am running a new distribution on the pi that runs headless. We are using ssh to access the pi. We still have the old SD card that had the GUI OS on it, and it still has all of the work we did on it.
- Use of the pi's serial port is disabled through two measures - first, with the following command:

```
sudo systemctl disable serial-getty@ttyAMA0.service
```

  As well as by removing the following text from the beginning of /boot/cmdline.txt:

```
console=serial0,115200
```

  This was done to allow communication with an accelerometer.
  On the pi 3, the following lines must be added to /boot/config.txt for serial communication to work properly:

```
dtoverlay=pi3-disable-bt

enable_uart=1
```

GPIO

- There are easy commands which can be run from the command line to change the state of GPIO pins. With the setup shown below (purple signal, black ground) the commands would be

```
raspi-gpio set 21 op
raspi-gpio set 21 [dl/dh]
```

  where dl would drive the pin to low, and dh would drive the pin high.

Auto Record Script

- As of 3/5/24, the script works. We can change how long we want each sample to be and the interval between samples we take. By editing the /etc/rc.local bash script, we can set the pi to start recording on boot. There is a line commented out currently that we can uncomment to get the recording to work. An important thing to note is that inside the script, there is an **absolute** path to the location for our recordings to be stored. If we ever want to change this we will need to edit the script.

Alsa

- Currently there are udev rules to assign the names 'HUB_MIC_1,' 'HUB_MIC_2', 'HUB_MIC_3' and 'HUB_MIC_4' to 4 mics plugged into the same USB hub. These rules work for when the USB hub is plugged into the top right USB port.

Jack

- Currently I am trying to dive into how ports work with jack. Stay tuned for now

RPi Storage Capacity

- Each microphone is recording 16 bit samples and 48000 samples per second
    - With 4 microphones recording simultaneously, we have a bit rate of 3072 kbit/s
    - With ~20Gb of free memory we can only record around 14 hours of sound
- Working on using hard drive adapter as external memory
    - 3/7 added a line to /etc/fstab to try and support automatically mounting the hdd adapter to /media/hdd, had to use the 'nowait' option so that booting continues in the absence of the drive
    - The UUID of each hard drive is different, which means that /etc/fstab must be edited accordingly
        - To check the UUID of a hard drive, use

```
lsblk -fs
```