Building an Audio Node

Part 1 – Getting the SD card image sorted

You can either download the pre-built image or build the image yourself, if using the prebuilt you will still need to set up your Wifi networks as shown below.

Using the prebuilt image

Download the pre-built image from here, this includes Raspbian, PD, volume control button service, shutdown service that works together with the external circuit and Power Boost board. You will need to set up the wifi credentials to connect to the device:

1. Use 'Pi Filler' app or similar to load the image onto the SD card

2. Mount the SD card from a computer

3. Create a text file in the root folder and rename it: `wpa_supplicant.conf`

4. Use a text editor to edit the wifi settings file you have just created with the entry below, you can add several networks. For example you can set up your home, work, a little portable router and an access point on your phone to cover all bases. You can also add a connection priority (e.g. '`priority=1`') to each entry:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=UK

network={
ssid="YOURSSID"
psk="YOURPASSWORD"
scan_ssid=1
}
```

5. Eject the SD card and run through steps 6 – 11 in the section below

To create your own image:

1. Download Raspbian image and unzip:
   https://www.raspberrypi.org/downloads/raspbian/
   a. [You will need the desktop version if you plan to VNC across to remote control the desktop using the GUI. This can be useful to set up PD and then you can disable the GUI, i.e run 'headless', later]

2. Use PiFiller or similar to load up SD card with the image

3. Add a text file for wifi settings to root folder called: wpa_supplicant.conf

4. Edit the wifi settings file with following, you can add several networks. For example you can set up your home, work, a little portable router and an access point on your phone to cover all bases. You can also add a connection priority (e.g. ‘`priority=1`’) to each entry:

```
a.  ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
b.  update_config=1
c.  country=UK

d.  network={
e.  ssid="YOURSSID"
f.  psk="YOURPASSWORD"
g.  scan_ssid=1
h.  }
```

5. Add a blank text file to root folder simply called ‘ssh’ with no file type

6. Insert SD card and boot up Pi

7. Make sure your computer/device is on the same network as the pi and open a terminal. SSH into the pi with: `ssh pi@raspberrypi.local` password: `raspberry` , or whatever hostname and password you have set up already
(kudos to ‘Termius’ which enables you to store snippets of terminal commands and communicate with the device from your phone easily)

8. Run configuration tool with: `sudo raspi-config`

9. Change hostname (in ‘Network Options’), change password, enable VNC (in ‘Interfacing Options’), you may also want to set the locale and time zone while you are here

10. If you are going to want to schedule things you will need to set the time which you can do manually (`sudo date -s "2/26/2020 9:35"`), but it will be forgotten when the battery is disconnected. Alternatively you could try installing NTP so it updates automatically when you have a network connection: `sudo apt-get install ntp`

11. Reboot Pi

12. Do the updates (it may take a while). Run:
```
sudo apt-get update && sudo apt-get upgrade
```

13. Install PD: `sudo apt-get install puredata`

14. Set up Adafruit Speaker Bonnet: https://learn.adafruit.com/adafruit-speaker-bonnet-for-raspberry-pi/raspberry-pi-usage

15. Open PD, setup audio and SAVE audio settings! I needed to use portaudio driver to make it work, also need to set the output device to ‘softvol’ to be able to control the volume from the alsamixer and via the button code.

16. Copy main PD patch across – Use an FTP client

17. Set up a script to open PD patch and run without the GUI if desired:

    a. Create the script: `sudo nano /home/pi/pdstart`

    b. Add following lines (watch filepath and case!!) :

    ```
    sleep 5
    pd -nogui -nomidi -verbose -stderr
    /home/pi/Documents/Pd/PDSamplePlay/playSample.pd
    sleep 5
    ```

    c. Set privileges on script with:

    ```
    sudo chmod 755 /home/pi/pdstart
    ```

    d. Start script with : `/home/pi/pdstart`
    (Can kill PD with: `sudo killall pd`)

18. You can add script to startup and schedule with crontab if desired:
    https://www.raspberrypi.org/documentation/linux/usage/cron.md
    **NB- using cron reboot function may mean the script starts before home drive is mounted or audio is enabled etc, the lazy can simply add an arbitrary delay to the script and hope it sorts it, or systemd could be used to do it properly and start it later in boot process details for using systemd here:
    https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/

19. If you wish to cut the GUI side of the Pi and save system resources then go into raspi-config and select boot options to go into console, i.e. run headless

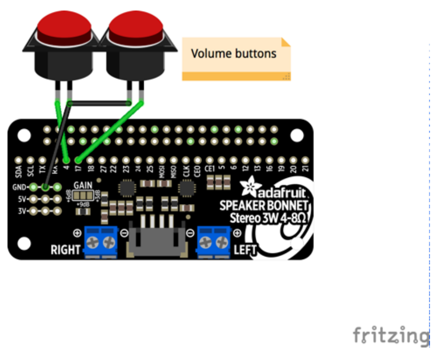
20. Disable Bluetooth if desired to save battery:

21. Edit boot config file: `sudo nano /boot/config.txt`
… and add these lines to bottom:

```
# Disable Bluetooth
dtoverlay=pi3-disable-bt
```

Part 2 – Getting the Volume and Power Button sorted

Volume control – connected buttons to GPIO pins 4 and 17



NB - The below works for system sound control via alsamixer, this means you may need to set the output device in PD to 'softvol' to get it to work

1. To create python script - `sudo nano volButton.py`
2. Insert the Python script code below:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    volUp = GPIO.input(4)
    if volUp == False:
        print('UP Button Pressed')
        #amixer set PCM 3%+
        from subprocess import call
        call(["amixer", "set", "PCM", "3%+"])
        #amixer set Speaker 5%+
        time.sleep(0.2)

    volDwn = GPIO.input(17)
    if volDwn == False:
        print('DOWN Button Pressed')
        #amixer set PCM 3%-
        from subprocess import call
        call(["amixer", "set", "PCM", "3%-"])
        #amixer set Speaker 5%-
        time.sleep(0.2)
```

3. Set the script up to run as a service, follow 'Method 4: SYSTEMD' at this link or the one below with the service code below.

- https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/#local

- https://github.com/bruceiow/rPiVolumeControl/blob/master/setup.md

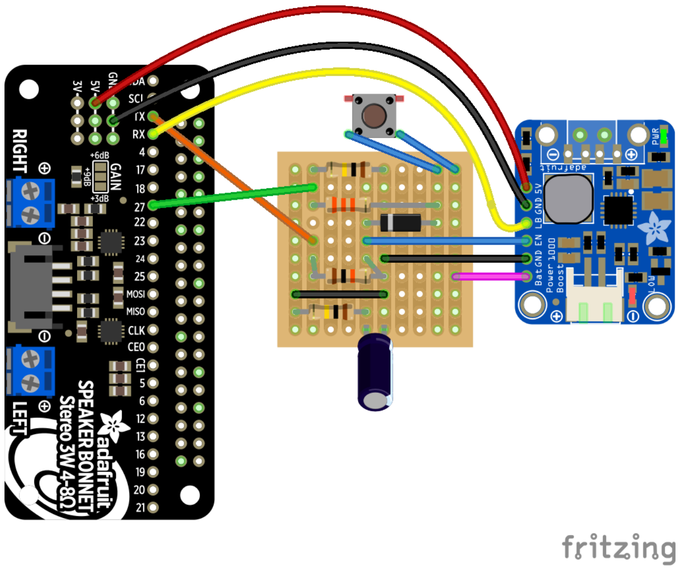
```
[Unit]
Description=Volume Controller
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/python /home/pi/volButton.py > /home/pi/volButton.log 2>&1

[Install]
WantedBy=multi-user.target
```

On/off button – using ‘lipopi’ resource

NB – Note use of pin 27 instead of pin 18 as used in the ‘lipopi’ resource. This is because the speaker bonnet makes use of pin 18



LiPoPi
https://github.com/NeonHorizon/lipopi/blob/master/README.power_up_power_down.md

[NB I found that the service was failing when I tried to run it and had to chmod on the original script to make it executable (probably because I added the script myself instead of copying the repository one over): `sudo chmod +x lipopi.py`]

I wanted the shutdown button press to also kill PD so I added to the lipopi.py script where shutdown or low battery is triggered: `os.system("sudo killall pd")`
e.g:

```
# Detect when the switch is pressed - wait shutdown_wait seconds - then shutdown

def lipopi_user_shutdown(channel):
    global lipopi

    cmd = "sudo wall 'System shutting down in %d seconds'" % lipopi['shutdown_wait']
    os.system(cmd)
    os.system("sudo killall pd")
    time.sleep(lipopi['shutdown_wait'])

    msg = time.strftime("User Request - Shutting down at %a, %d %b %Y %H:%M:%S +0000\n",
time.gmtime())
    lipopi['logfile_pointer'].write(msg)
    lipopi['logfile_pointer'].close()
    GPIO.cleanup()
    os.system("sudo shutdown now")


# Respond to a low battery signal from the PowerBoost and shutdown
# Pin goes low on low battery

def lipopi_low_battery_shutdown(channel):
    global lipopi
    cmd = "sudo wall 'System shutting down in %d seconds'" % lipopi['shutdown_wait']
    os.system(cmd)
    os.system("sudo killall pd")
    time.sleep(lipopi['shutdown_wait'])

    msg = time.strftime("Low Battery - Shutting down at %a, %d %b %Y %H:%M:%S +0000\n", 
time.gmtime())
    lipopi['logfile_pointer'].write(msg)
    lipopi['logfile_pointer'].close()
    GPIO.cleanup()
    os.system("sudo shutdown now")
```

Part 3 – The Hardware

Download the laser cutter files from the repository.

These designs are based on 3mm stock, though remember that acrylic sheet is not always uniform so it is possible you may end up adapting the design. You can use wood stock, but the sections that hold components have some delicate thin structures that require some strength and are best suited to acrylic.

Note the use of deep engraves to allow components to be held snugly in the case and allow for a smaller overall size. On my laser cutter I use full power and a speed of 150 to go a good halfway into the 3mm stock, this leaves a lot of dust residue so go over the engraved sections with a flat head screwdriver of a small file to clear it out.

The bill-of-materials (BOM) can also be found in the repository, this lists all the components needed, costs and potential suppliers.

The design file lays out the parts in assembled order so it can be used as a blueprint, the volume button and power button need to be connected as shown above with the wires threaded through the case to the required location. I find it best to cut them a little long and then trim them to size on assembly. I find it useful to use header pins and sockets when assembling the circuits as everything for ease of assembly/disassembly/troubleshooting, but you can solder wires directly. NB – you can use thin or ribbon wire for most connections, but use some decent wires for the 5V and GND connections.

Assembly photos can also be found in the repository along with circuit design files.

Charging the battery

The PowerBoost

To do:
Startup sound, shutdown sound, button sound
Startup sound - https://raspberrypi.stackexchange.com/questions/3632/running-headless-how-do-i-create-a-boot-sound

N.B:

If you get SSH error connecting :
May need to delete hostname/IP from known hosts file if you get ssh refusal due to suspected spoofing: /Users/yourUserName/.ssh/known_hosts

Change volume once in headless mode:

> Type: alsamixer to enter command line mixer, use F6 to swap between sound cards. If you already have an ssh window open and PD is taking over, you can open another ssh window to edit the volume at the same time.