Adding Basic Audio Output to Raspberry Pi Zero

Created by lady ada

https://learn.adafruit.com/adding-basic-audio-output-to-raspberry-pi-zero

Last updated on 2021-11-15 06:35:37 PM EST
# Table of Contents

## Overview 3
- How Audio Works on Other Pi Computers 3

## Pi Zero PWM Audio 4
- Let's go! 4
- Option 1. Use Device Tree Overlay 5
- Option 2. Manually Assigning PWM pins 5
- Changing the GPIO ALTs 10
- Low Pass Filter Wiring 14
- Set Audio Output 15
- First Test 17
- Adjusting Volume 17
- Automate it! 18
Overview

To keep the Raspberry Pi Zero as low cost and small as possible, the Pi foundation didn't include a 3.5mm audio jack. There's also no breakout pads for the audio output. This made us a little :( at first but then we thought "hey you know, we can probably figure out how to get audio out with a little hacking!

How Audio Works on Other Pi Computers

The Broadcom chipset used for the Pi does not have a true analog output. Instead, two pins are PWM (pulse-width-modulated) at very high speeds, and filtered. The PWM frequency has to be at least 10x as high as the highest frequency we want to replicate in audio. Then, by adjusting the duty cycle of the PWM, we can 'fake' an audio signal.

Audio is 20Hz to 20KHz, and the PWM output from the Pi is 50MHz so we can easily filter the high 50MHz out (and anyways it can't be heard).

Looking at the Pi B schematic, we can see PWM0_OUT and PWM1_OUT are the left and right channels. R21 and R20 are voltage dividers to get the 3.3V signal down to about 1.1V max (that's the max peak-to-peak voltage you want for audio line level.)
C20/C26 works with R21/R27 to create an "RC low-pass filter". You can calculate the cut-off frequency with $1/(2\pi\times RC) = 1/(2\pi\times270\times33\times10^{-9}) = 17865$ Hz which is pretty close to 20KHz!

C48/C34 acts as a DC-filter capacitor, it only allows AC through - speakers and headphones don't like DC voltage!

Finally BAV99 are ESD protection diodes. That's to protect the Pi from static coming in and zapping the PWM pins..

Pi Zero PWM Audio

On the Pi Zero, we don't have pins PWM0 (pin #40) and PWM1 (pin #45)- those are not available on the PCB. That would normally be super :-( but it turns out that while those pads are not exposed, we can re-route those signals to other pins that we can get to!

You can get to PWM0 on GPIO #18 (ALT5) and PWM1 on GPIO #13 (ALT0) or GPIO #19 (ALT5) - see the full list of pins and alternate functions here (https://adafruit.it/jEa)

Let's go!

Get started by burning and booting a Raspbian OS card in your Zero, we'll be using Raspbian Jessie. You don't need network connectivity to get this going but its handy so if you can configure WiFi or Ethernet thru a USB adapter, do that too.

Then log into a command line console.
Option 1. Use Device Tree Overlay

There’s two options, the easy way is using a DTO which will set up all your pins and PWMs. However, we have not tested it.

The other way is to manually set the pin GPIO alt functions. We’ve tested it but it’s more difficult!

Here’s two DTO options that people have written in to suggest. The first uses a DTO that was recently added it seems:

1. There is a much simpler way to configure the Pi GPIO pins for PWM audio. Simply adding the following line to your /boot/config.txt will reconfigure the pins at boot without any external software or services: `dtoverlay=pwm-2chan,pin=18,func=2,pin2=13,func2=4`

Or, you can craft your own DTO


Either way, if you use a DTO you can skip down to the low pass wiring section ([https://adafruit.it/rlB](https://adafruit.it/rlB))

Option 2. Manually Assigning PWM pins

This is how to manually set the pin functionality - you can do this if you want to set up the individual pins and usages

Before we start its handy to use [wiringpi's gpio utility](https://adafruit.it/jWD) to list all the GPIO pins and their current set functions/alternates.

If you have network access, run `sudo apt-get update`

Then install it (in the off chance it isn't installed/updated) with `sudo apt-get install raspi-gpio`
On our Pi Zero, it didn't have a Zero-compatible version, and said "unable to determine board type". If you have network access, and can run sudo apt-get update and then the install line, this will likely fix it by grabbing a new version. However...I didn't have network access for my Pi and figured its worth documenting how to get around this. sudo shutdown -h now your Pi Zero, eject the SD card. Then on your desktop computer, download the latest snapshot from the wiringPi Git repo (https://adafru.it/jWE)

In case its not available, you can download it by clicking this button:

[button]

wiringPi-78b5c32.tar.gz
https://adafru.it/jWF

Transfer the file to the root directory of the Jessie SD card
Re-boot your Pi Zero up, and get back to the command line. Then `mv` the file over and un-tar it

```
sudo mv /boot/wiringPi-78b5c32.tar.gz .
tar -zxvf wiringPi-78b5c32.tar.gz
```

then go into the uncompressed directory and run the compile/install script with

```
cd wiringPi-78b5c32.tar.gz
./build
```
OK now you can run the updated version of gpio to read the states of all the pins:

```
gpio -v
```

(to make sure it works with the Pi Zero now, it should properly detect the board type)
gpio readall

The table has a ton of information going on. What you want to look for is BCM pins #1 8 and #13
As you can see, these two pins have MODE type IN - that means they are just plain inputs. You can see that pins like TxD and RxD are 'ALT0', that's the built in serial console.

Anyways, time to change these pins!

### Changing the GPIO ALTs

We can manually tweak the GPIO ALTs using a very handy tool by TimG in the Pi forums (https://adafru.it/jEc)

Here's the code in entirety:

```c
/*
 Utility to switch Raspberry-Pi GPIO pin functions
 Tim Giles 01/04/2013

 Usage:
 $ gpio_alt -p PIN_NUMBER -f ALT_NUMBER

 Based on RPi code from Dom and Gert, 15-Feb-2013, &lt;http://elinux.org/RPi_Low-level_peripherals#C_2&gt;
 */

#include &lt;ctype.h&gt;
#include &lt;stdio.h&gt;
#include &lt;stdlib.h&gt;
#include &lt;unistd.h&gt;
#include &lt;fcntl.h&gt;
```

©Adafruit Industries
```c
#include "sys/mman.h"

#define BCM2708_PERI_BASE 0x20000000
#define GPIO_BASE (BCM2708_PERI_BASE + 0x200000) /* GPIO controller */
#define PAGE_SIZE (4*1024)
#define BLOCK_SIZE (4*1024)

int  mem_fd;
void *gpio_map;
volatile unsigned *gpio;
void setup_io();

// GPIO setup macros. Always use INP_GPIO(x) before using OUT_GPIO(x) or SET_GPIO_ALT(x,y)
#define INP_GPIO(g) *(gpio+((g)/10)) &= ~(7<<(((g)%10)*3))
#define OUT_GPIO(g) *(gpio+((g)/10)) |=  (1<<(((g)%10)*3))
#define SET_GPIO_ALT(g,a) *(gpio+(((g)/10))) |= (((a)<=3?(a)+4:(a)==4?3:2)&lt;&lt;(((g)%10)*3))

#define GPIO_SET *(gpio+7)  // sets bits which are 1 ignores bits which are 0
#define GPIO_CLR *(gpio+10) // clears bits which are 1 ignores bits which are 0

int main (int argc, char **argv) {
    int opt, flag, n_pin, n_alt;
    flag=0;

    while ((opt = getopt (argc, argv, "hp:f:")) != -1) {
        switch (opt) {
            case 'h':
                break;
            case 'p':
                n_pin = atoi(optarg); flag |= 0b0001; break;
            case 'f':
                n_alt = atoi(optarg); flag |= 0b0010; break;
            case '?':
                // getopt() prints error messages, so don't need to repeat them here
                return 1;
            default:
                abort ();
        }
    }

    if (flag != 0b0011) {
        fprintf (stderr, "Usage:
            "$ gpio_alt -p PIN_NUM -f FUNC_NUM
"");
        return 1;
    }

    setup_io(); // Set up gpio pointer for direct register access
    INP_GPIO(n_pin);  // Always use INP_GPIO(x) before using SET_GPIO_ALT(x,y)
    SET_GPIO_ALT(n_pin, n_alt);
    printf("Set pin %i to alternative-function %i\n", n_pin, n_alt);
    return 0;
}

void setup_io() {
    /* open /dev/mem */
    if ((mem_fd = open("/dev/mem", O_RDWR|O_SYNC) ) &lt; 0) {
        printf("can't open /dev/mem \n");
        exit(-1);
    }
    /* mmap GPIO */
```

---

Adafruit Industries
Page 11 of 20
gpio_map = mmap(
    NULL,             //Any address in our space will do
    BLOCK_SIZE,       //Map length
    PROT_READ|PROT_WRITE, // Enable reading & writing to mapped memory
    MAP_SHARED,       //Shared with other processes
    mem_fd,           //File to map
    GPIO_BASE         //Offset to GPIO peripheral
);

close(mem_fd); //No need to keep mem_fd open after mmap

if (gpio_map == MAP_FAILED) {
    printf("mmap error %d\n", (int)gpio_map);//errno also set!
    exit(-1);
}

// Always use volatile pointer!
gpio = (volatile unsigned *)gpio_map;

On your Pi Zero, create a new file and edit it with

nano gpio_alt.c

(it doesn't matter what directory you are in)

Then paste in the entire code above

Save it...
Compile it & install with

```
gcc -o gpio_alt gpio_alt.c
sudo chown root:root gpio_alt
sudo chmod u+s gpio_alt
sudo mv gpio_alt /usr/local/bin/
```

Now you can set the ALT functions of the two GPIO!

```
gpio_alt -p 13 -f 0
```

```
gpio_alt -p 18 -f 5
```

Sweet. Now go back to wiringPi to check that we did it!
Yep! You can see the new ALT settings.

**Low Pass Filter Wiring**

Now wire up the schematic to GPIO #13 as PWM1 and #18 PWM0 on a breadboard. You can skip the diodes. If you don't have the exact values it's ok. I built it with a 10nF capacitor (0.01uF) rather than 33nF and it worked just fine (the cutoff frequency is higher but the speakers don't hear those high frequencies anyways)
I'm using a 3.5mm audio jack terminal block (http://adafruit.it/2791) to wire up the left & right channels + ground.

Set Audio Output

You'll also want to 'fix' the Pi so the audio is definitely getting piped out the 'headphone jack' (PWM output) rather than HDMI. From the console run sudo raspi-config

Go to Advanced
Then Audio

Finally Force 3.5mm (Headphone)

You only have to do this once, you can hit return and then Finish to exit
First Test

You can now play audio! Plug in powered speakers or headphones. We will use the built in aplay audio player. Run

```
aplay /usr/share/sounds/alsa/Front_Center.wav
```

You should hear audio!

Adjusting Volume

You may notice a bit of hum, or maybe its just not very loud. To get the best quality audio, you'll want to have the audio level out of the Pi be as high as possible. You can do this with alsamixer which is easier in my opinion than amixer although both work.

On my terminal the ascii art is a bit tough to read, just press the up arrow until the volume is at 100% Then you can hit Esc to save & quit

You only have to do this once, alsamixer saves the settings between reboots.
Automate it!

OK now you have that working, you probably want it to happen at boot, and not need to run gpio_alt! No biggies, we have a fine tutorial on how to automate things at boot on Jessie with systemd (https://adafruit.it/jXa)

A 'better' way to do it is to adapt the dt-blob device tree, however at the time of writing, the Zero dts is not documented and this works without needing internet so...thats why we're doin' it this way!

Here's how to do it.

Start by creating a shell script with sudo nano /root/pwmaudio.sh

```bash
#!/bin/bash
/usr/local/bin/gpio_alt -p 13 -f 0
/usr/local/bin/gpio_alt -p 18 -f 5
```

©Adafruit Industries  Page 18 of 20
run `sudo chmod +x /root/pwmaudio.sh` and then create another script with `sudo nano /lib/systemd/system/pwmaudio.service`

and in there stick:

```
[Unit]
Description=PWM Audio Service

[Service]
ExecStart=/root/pwmaudio.sh
StandardOutput=null

[Install]
WantedBy=multi-user.target
Alias=pwmaudio.service
```

save the file. Now enable the service with

```
sudo systemctl enable pwmaudio.service
```

and test-start the service with

```
sudo systemctl start pwmaudio.service
```
Reboot the pi (sudo reboot) and re-log in. You can run the gpio readall commands to verify that the alts are set

That's it! You're done, enjoy :)}