

# PWM Guide: Zen Buzzer and Tri-Colour LEDs

For Linux Kernel 4.9+ (BeagleBone 2018-01-28 image)

by Brian Fraser

Last update: November 1, 2019

## This document guides the user through:

1. Loading PWM support.
2. Driving the Zen cape's buzzer via PWM from a Linux terminal.
3. Driving the Zen cape's tri-colour LED via three PWM channels from the Linux terminal.

## Table of Contents

1. PWM Basics.....	2
2. Load PWM Device Tree.....	3
2.1 Load PWM Device Tree – Universal Cape.....	3
2.2 Load PWM Device Tree – Edit uEnv.txt.....	3
3. Linux PWM: Buzzer.....	5
4. Tri-colour LED.....	7

## Formatting:

1. Host (desktop) commands starting with \$ are Linux console commands:  
\$ echo "Hello world"
2. Target (board) commands start with #:  
# echo "On embedded board"
3. Almost all commands are case sensitive.

## Revision History:

- Nov 1: Initial version.

# 1. PWM Basics

Pulse-width modulation (PWM) is a way of generating a digital wave form (think of a clock signal). You can specify two main components of the digital wave form:

1. **Period:** How much time is there between the start of one cycle and the next. This is the time between rising edges of the wave form.
2. **Duty:** This is the percentage of the cycle which the signal is high (or low, depending on its configuration).

Together, these two parameters allow you to generate waves such as those shown in Figure 1.

In some situations an analog voltage is needed. A PWM wave can be used to create such a voltage by applying extra hardware (capacitors) to smooth out, or average out, the wave form. For example, when the signal is between 0 and 3.3V, a 50% duty cycle would average out to 1.65V (half of 3.3V).

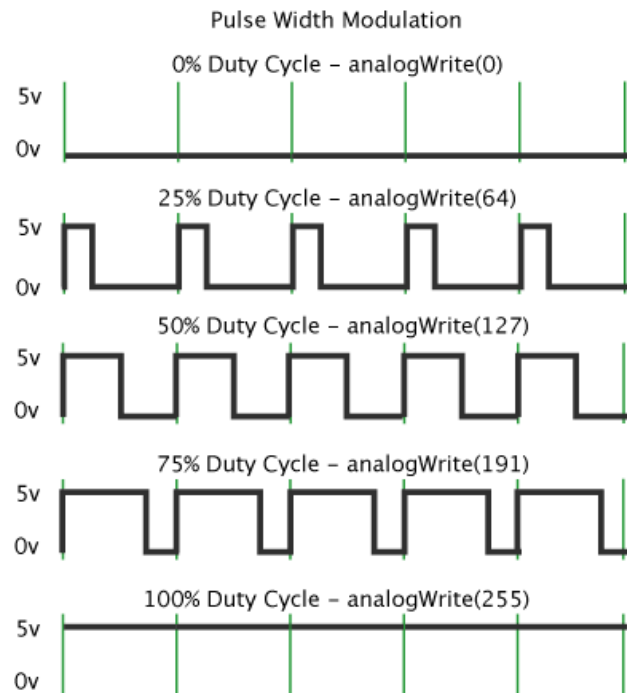


Figure 1: PWM wave forms for different duties, from <https://www.arduino.cc/en/Tutorial/PWM>

The PWM channels used by the Zen cape are listed below. Note that not all PWM channels are used by the Zen cape: some are unused on the BBB, and others are used by the HDMI hardware.

Zen Cape Use	PWM Channel	BBB Pin	Linux Path	Notes
Buzzer	PWM-0A	P9-22	/sys/class/pwm/pwmchip0/pwm0/	
Red LED	PWM-1B	P9-16	/sys/class/pwm/pwmchip2/pwm1/	Shares PWM hardware timer with Blue LED (PWM-1A)
Green LED	PWM-2A	P8-19	/sys/class/pwm/pwmchip4/pwm0/	
Blue LED	PWM-1A	P9-14	/sys/class/pwm/pwmchip2/pwm0/	Shares PWM hardware timer with Red LED (PWM-1B)

Note that for PWM channels which share PWM hardware timers (red and blue), you cannot change the period of these channels independently. See Section 4 for more.

## 2. Load PWM Device Tree

Linux can learn about the PWM hardware in two ways. See next sections for detailed directions for the appropriate way.

### 1. Universal cape

If you have not edited the `/boot/uEnv.txt` file to load any device tree overlays, nor have any overlays loading for installed capes, then the universal cape will automatically be loaded for you.

### 2. Edit `uEnv.txt` to load PWM device tree overlays

If you are loading custom device tree overlay files via `/boot/uEnv.txt` then you'll need to once again edit `uEnv.txt` to load more device tree overlays.

Unsure which applies to you? Try the first step of the universal cape section and see if it works!

## 2.1 Load PWM Device Tree – Universal Cape

If using the universal cape, do the following:

### 1. Set the buzzer's pin to PWM

```
# sudo config-pin p9_22 pwm
```

- If this fails with a message like the following, it likely means that the universal cape is not loaded: jump to the next section

```
$ sudo config-pin p9_22 pwm
```

```
P9_22 pinmux file not found!
```

```
bash: /sys/devices/platform/ocp/ocp*P9_22_pinmux/state: No such file or directory
```

```
Cannot write pinmux file:
```

```
/sys/devices/platform/ocp/ocp*P9_22_pinmux/state
```

- This configures the CPU's pin connected to the buzzer for use with PWM (each pin has multiple possible functions to select from).
- You can list all the functions the pin can be configured for using:  

```
# config-pin -l P9_22
```
- You can check the current state of the pin:<sup>1</sup>  

```
# config-pin -q P9_22
```

### 2. If you are using the LEDs on the Zen cape, you will need to execute the following commands:

```
# config-pin P9_14 pwm
```

```
# config-pin P9_16 pwm
```

```
# config-pin P8_19 pwm
```

- See table at start of this guide listing mapping of each LED to PWM, pins, and paths.
- NOTE: **P9\_14 must be configured before P9\_16** or else you might not be able to export some PWM channels in the next step.

## 2.2 Load PWM Device Tree – Edit `uEnv.txt`

If loading the overlay via `uEnv.txt`, do the following

### 1. Update the cape overlays:

<sup>1</sup> This is similar to the following command:

```
# cat /sys/devices/platform/ocp/ocp\:P9_22_pinmux/state
```

```
# sudo apt-get update
# sudo apt-get install bb-cape-overlays
```

- You must have an internet connection for this to succeed.

2. Ensure you have an up-to-date version:

```
# apt-cache show bb-cape-overlays
Package: bb-cape-overlays
Version: 4.4.20190922.0-0rcnee0~stretch+20190922
Architecture: armhf
....
```

3. Copy the current uEnv.txt file. This is necessary for recovering from anything going wrong!

```
# sudo cp /boot/uEnv.txt /boot/uEnv-BeforePwm.txt
```

4. Edit the uEnv.txt file to enable the PWM overlays

```
# sudo nano /boot/uEnv.txt
```

- Edit the Additional custom capes section.

Below shows the setup you'll have if you are already loading the audio cape and the I2C-1 cape. If you are not loading these, you may comment out those lines

```
###Additional custom capes
uboot_overlay_addr4=/lib/firmware/BB-BONE-AUDI-02-00A0.dtbo
uboot_overlay_addr5=/lib/firmware/BB-I2C1-00A0.dtbo
uboot_overlay_addr6=/lib/firmware/BB-PWM0-00A0.dtbo
uboot_overlay_addr7=/lib/firmware/BB-PWM1-00A0.dtbo
uboot_overlay_addr3=/lib/firmware/BB-PWM2-00A0.dtbo
```

- Note that you can use `uboot_overlay_addr0` through `uboot_overlay_addr7` for any of the capes being loaded. If you use 0-3, it replaces the cape support for any automatically detected capes configured to load at that slot. If you have no physical capes connected (which are automatically detected), then you can use any of the 8 slots you like. This is why the above snippet uses 3 through 7. The number dictates the order they are loaded, which should not matter here.
- You only need to load the capes you need.

5. Reboot the target

6. Troubleshooting

- See the Audio guide's last section (on the course website) to recover from a corrupted `uEnv.txt`. You should be able to copy back the `/boot/uEnv-BeforePwm.txt` to restore your previous working state.

### 3. Linux PWM: Buzzer

**NOTE ON SILENCE:** The buzzer can be manually turned off by removing the jumper (black rectangle) just below the buzzer (left of the joystick) on the Zen cape.

1. Export the PWM channel:  

```
# echo 0 > /sys/class/pwm/pwmchip0/export
```

  - The 0 tells Linux to export the config files for part A of the PWM (P8\_19). Write a 1 for part B (P8\_13).
  - The `pwmchip0` directory means this is for PWM A.
2. View the PWM files in the sysfs:  

```
# cd /sys/class/pwm/pwmchip0/pwm0/  
# ls  
capture  duty_cycle  enable  period  polarity  power  uevent
```
3. Set the period of a cycle (via `period`, in ns), duration of each “on” pulse (`duty_cycle`, in ns), and if it is running (`run`, a 1 for on):  

```
# echo 100000 | sudo tee period  
# echo 50000 | sudo tee duty_cycle  
# echo 1 | sudo tee enable
```

  - You can manually silence the buzzer by pulling the jumper beside it. When put the jumper back in, make sure you connect it to the correct pins! (not the big P8 header!)
  - Note on times:  
1 second  
= 1,000 milliseconds [ms]  
= 1,000,000 microseconds [us]  
= 1,000,000,000 nano-seconds [ns]
  - For the buzzer, an easy way to work is always make the `duty_cycle` half of the `period`. It's the period that controls the frequency of the sound played (the note).
4. Turn off with:  

```
# echo 0 | sudo tee enable
```
5. Make it play a lower pitched sound by giving it a larger period (ns):  

```
# echo 1000000 | sudo tee period  
# echo 500000 | sudo tee duty_cycle
```

  - Ensure it's enabled (write a 1 to `enable`) for sound to be generated.
6. Make it play a higher pitched sound by giving it a smaller period. Since the period cannot be less than the duty cycle (period cannot be on longer than a single cycle), we must first set the duty to be less than we want the period to be:  

```
# echo 0 | sudo tee duty_cycle  
# echo 100000 | sudo tee period  
# echo 50000 | sudo tee duty_cycle
```

7. You can play specific notes on the buzzer. First find the frequency for the note you want (try online) and then compute the period by:

Period = (1 / Frequency [cycles per s]) \* 1,000,000,000 [ns per s]

Set the duty to be half of the period.

- For example, middle C is 261.6Hz. This gives a period of 3,822,256ns and duty 1,911,128ns:

```
# echo 0 | sudo tee duty_cycle
# echo 3822256 | sudo tee period
# echo 1911128 | sudo tee duty_cycle
```

8. Troubleshooting:

- Unknown folder when you try to export the PWM? You likely don't yet have the hardware support loaded yet. Follow the steps in section 2 - Load PWM Device Tree.
- No sound?
  - Ensure you have the jumper correctly installed connecting the two pins just below the buzzer (left of the joystick).
  - Ensure you have it running (`# echo 1 | sudo tee enable`)
  - Ensure you have set the duty to be half of the period.
- While trying to change the period, if you get:  
"bash: echo: write error: Invalid argument"

You are likely trying to change the period to a value less than the duty cycle. First change the duty cycle to 0, then retry your change.

## 4. Tri-colour LED

### 1. Export the PWM functionality for LEDs:

- **Blue**  
`# echo 0 | sudo tee /sys/class/pwm/pwmchip2/export`
- **Red**  
`# echo 1 | sudo tee /sys/class/pwm/pwmchip2/export`
- **Green**  
`# echo 0 | sudo tee /sys/class/pwm/pwmchip4/export`

### 2. Set the period for the LED PWM channels.

```
# echo 100000 | sudo tee /sys/class/pwm/pwmchip2/pwm0/period
# echo 100000 | sudo tee /sys/class/pwm/pwmchip2/pwm1/period
# echo 100000 | sudo tee /sys/class/pwm/pwmchip4/pwm0/period
```

- The red and blue PWM channels share hardware so you cannot change the period of these channels independently. This is not an issue for us: we just need to set the duty cycle.
- In fact, the software won't let you change the red/blue's period at all if you have both PWM channels' periods set. Specifically, you can set and change the period of one of the red or blue PWM channels until the other one is given a period. Then, you can only change the period of one to match the other.
- So, in other words, set the period to something reasonable to start and then don't change it.

### 3. Enable the PWM

```
# echo 1 | sudo tee /sys/class/pwm/pwmchip2/pwm0/enable
# echo 1 | sudo tee /sys/class/pwm/pwmchip2/pwm1/enable
# echo 1 | sudo tee /sys/class/pwm/pwmchip4/pwm0/enable
```

### 4. LED brightness is controlled by how much time it is being turned on per cycle (the duty cycle).

Off     Set `duty_cycle` to 0

50%     Set `duty_cycle` to 50000

100%    Set `duty_cycle` to 100000

- For example: blue to 0%, red to 100%, green to 0%):  
`# echo 0 | sudo tee /sys/class/pwm/pwmchip2/pwm0/duty_cycle`  
`# echo 100000 | sudo tee /sys/class/pwm/pwmchip2/pwm1/duty_cycle`  
`# echo 0 | sudo tee /sys/class/pwm/pwmchip4/pwm0/duty_cycle`

### 5. Generate your own colours by specifying the RGB components. For example, purple is 50% red, 0% green, 50% blue.

```
# echo 50000 | sudo tee /sys/class/pwm/pwmchip2/pwm0/duty_cycle
# echo 50000 | sudo tee /sys/class/pwm/pwmchip2/pwm1/duty_cycle
# echo 0 | sudo tee /sys/class/pwm/pwmchip4/pwm0/duty_cycle
```

- The colours of the LED don't mix together very well, so it can hard to create exactly the colour you have in mind. However, try putting a piece of paper over it to defuse the light and you may find it seems to mix better.

6. You may find the following script useful. Copy to a file `driveZenLEDs.sh`, change to executable and run with:

```
# ./driveZenLEDs.sh 100000 50000 0
#!/bin/sh
## set -x
echo "Can pass R G B parameters (0 = off; 100,000 = on)"

## Figure out values for writing
RED=50000
GREEN=50000
BLUE=50000
if [ $# -eq 3 ]; then
    RED=$1
    GREEN=$2
    BLUE=$3
fi

## IF USING UNIVERSAL CAPE:
## Configure pins for PWM
## MUST have P9_14 before P9_16!
#config-pin P9_14 pwm
#config-pin P9_16 pwm
#config-pin P8_19 pwm

#config-pin -q P9_14
#config-pin -q P9_16
#config-pin -q P8_19

## Export the PWM folders to work with.
if [ ! -d /sys/class/pwm/pwmchip2/pwm0 ]; then
    echo 0 | sudo tee /sys/class/pwm/pwmchip2/export
fi
if [ ! -d /sys/class/pwm/pwmchip2/pwm1 ]; then
    echo 1 | sudo tee /sys/class/pwm/pwmchip2/export
fi
if [ ! -d /sys/class/pwm/pwmchip4/pwm0 ]; then
    echo 0 | sudo tee /sys/class/pwm/pwmchip4/export
fi

## Setup period
echo 100000 | sudo tee /sys/class/pwm/pwmchip2/pwm0/period
echo 100000 | sudo tee /sys/class/pwm/pwmchip2/pwm1/period
echo 100000 | sudo tee /sys/class/pwm/pwmchip4/pwm0/period

## Set the color (duty-cycle)
echo "Red: $RED, Green: $GREEN, Blue: $BLUE"
echo $BLUE | sudo tee /sys/class/pwm/pwmchip2/pwm0/duty_cycle
echo $RED | sudo tee /sys/class/pwm/pwmchip2/pwm1/duty_cycle
echo $GREEN | sudo tee /sys/class/pwm/pwmchip4/pwm0/duty_cycle

## Enable
echo 1 | sudo tee /sys/class/pwm/pwmchip2/pwm0/enable
echo 1 | sudo tee /sys/class/pwm/pwmchip2/pwm1/enable
echo 1 | sudo tee /sys/class/pwm/pwmchip4/pwm0/enable
```



## 7. Troubleshooting:

- If you get the messages for an unknown folder when you try to export the PWM? You likely don't yet have the hardware support loaded yet. Follow the steps in section 2 - Load PWM Device Tree.
- "Permission denied" error trying to write to a file: You either incorrectly used sudo tee, or have not yet exported the PWM for that pin.
- "Device or resource busy" when trying to export: the PWM is likely already exported.
- "Invalid argument" when writing 1 to the `enable` file likely means you have not yet set the `period` or `duty_cycle` correctly.
- Unable to change period of red/blue LED: This is by design, since both PWMs are linked and the period cannot be changed. It is best to not change the period of any of the LED PWM channels for this reason.
- When exporting the red LED channel (`pwm1` from `pwmchip4`) if you get the following:  

```
# echo 1 | sudo tee /sys/class/pwm/pwmchip2/export
sh: echo: I/O error
```

then ensure that you have configured the pins for PWM, and that P9\_14 was configured before before P9\_16.