PWM Guide for BeagleY-AI

LED Emitter and Discrete LED

by Brian Fraser

Last update: Feb 13, 2025

Guide has been tested on

BeagleY-AI (Target):	**Debian 12.x** (and beyond)
PC OS (host):	**Debian 12.x** (and beyond)

This document guides the user through:

1. Loading PWM support.
2. Driving the Zen hat's LED emitter via PWM from a Linux terminal.
3. Driving a discrete LED via PWM.

Table of Contents

Formatting:

1. Commands for the host Linux's console are show as:
```
(host)$ echo "Hello PC world!"
```
2. Commands for the target (BeagleY-AI) Linux's console are shown as:
```
(byai)$ echo "Hello embedded world!"
```
3. `Almost all commands are case sensitive.`

Revision History:

- Nov 1, 2021: Initial version.
- Mar 9, 2023: Updated for Bullseye (kernel 5.10).
- Feb 11, 2024: Added section on discrete LED for PWM.
- Feb 13, 2025: Changed to supporting BeagleY-AI.

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, such as dimming an LED. When the duty cycle of the PWM signal is adjusted it changes how much power the LED gets over time and hence changes its brightness.

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).

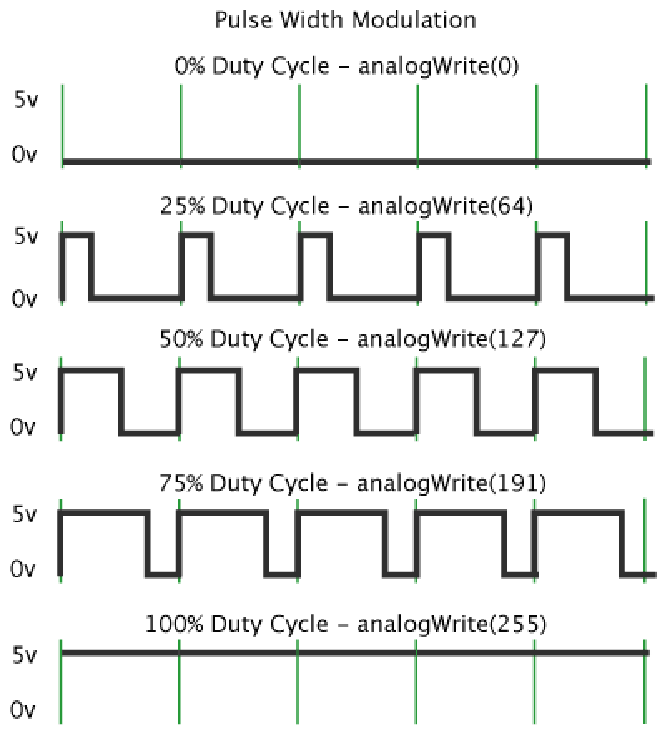

Pulse Width Modulation
0% Duty Cycle - analogWrite(0)
25% Duty Cycle - analogWrite(64)
50% Duty Cycle - analogWrite(127)
75% Duty Cycle - analogWrite(191)
100% Duty Cycle - analogWrite(255)
5v
0v

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

The BeagleY-AI (BYAI) has a number of PWM pins available on its main header. Highlighted rows are either used by Zen Hat for PWM, or are available for use.

Pin #	Pin Name	Zen Cape Use	Linux Path	Notes
8	**GPIO 14**	**Free - Mapped to UART Header for Tx**		
10	**GPIO 15**	**Free - Mapped to UART Header for Rx**		
11	GPIO 17	Used - Rotary Encoder B (Input)		
12	GPIO 18	Used - Audio clock		
29	GPIO 5	Used - Joystick push (Input)		
31	**GPIO 6**	**Free - Mapped to PYMNL header**		
32	**GPIO 12**	**PWM: LED Emitter**		
33	**GPIO 13**	**PWM: LCD Backlight**		
35	GPIO 19	Used - Audio WCLK		
36	GPIO 16	Used - Rotary Encoder A (Input)		
38	GPIO 20	Used - Audio Data In		
40	GPIO 21	Used - Audio Data Out		

1.1 Pick PWM Pins to Enable

In hardware, there are different PWM modules named PWM0, PWM1, and ECAP0[1], ECAP1, and ECAP2. Each module is disabled by default, and can be enabled by editing the /boot/firmware/extlinux/extlinux.conf file (as described in the steps later).

Each module can be configured to set the desired pins as PWM output. Note that in some cases, such as GPIO5 and GPIO15, there is only one real PWM channel in hardware which is then mapped to two different possible pins. In this case, it is PWM0 Channel A that can be mapped to either GPIO5 or GPIO15.

1 ECAP is the Texas Instruments Enhanced Capture Module: https://www.ti.com/lit/ug/spru807b/spru807b.pdf?ts=1739432450195

In the following table, no more than one GPIO pin per column can be enabled.

Table 1: Hardware support for GPIO pins (https://pinout.beagleboard.io/pinout/pwm)

	PWM0		**PWM1**		**ECAP**		
	A	**B**	**A**	**B**	**0**	**1**	**2**
GPIO5	X						
GPIO6			X				
GPIO12		X			X		
GPIO13				X			
GPIO14		X					
GPIO15	X						
GPIO16						X	
GPIO17							X
GPIO18							X
GPIO20				X			
GPIO21			X			X	

To enable a PWM configuration, we will load an overlay (steps below). Available overlays are found by:

```
(byai)$ ls /boot/firwmare/overlays/*beagley-ai-pwm*
k3-am67a-beagley-ai-pwm-ecap0-gpio12.dtbo
k3-am67a-beagley-ai-pwm-ecap1-gpio16.dtbo
k3-am67a-beagley-ai-pwm-ecap1-gpio21.dtbo
k3-am67a-beagley-ai-pwm-ecap2-gpio17.dtbo
k3-am67a-beagley-ai-pwm-ecap2-gpio18.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio12.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio14.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio15-gpio12.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio15-gpio14.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio15.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio5-gpio12.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio5-gpio14.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio5.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio13.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio20.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio21-gpio13.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio21-gpio20.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio21.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio6-gpio13.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio6-gpio20.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio6.dtbo
```

You can load multiple overlays at once, as long as they refer to different modules. I.e., you cannot load two “epwm0” modules at once, but you could load one for each of “ecap0”, “ecap1”, “ecap2”, “epwm0”, “epwm1” and “epwm2” all at once.

2. Linux PWM: Zen Hat LED Emitter

1. By default, the PWM functionality is not enabled on the BeagleY-AI. We must first tell Linux to load the correct driver and dedicate the necessary pins to PWM.

```
(byai)$ sudo nano /boot/firmware/extlinux/extlinux.conf
```

   - To load PWM support for GPIO12, make the last entry in the file look like the following (note one line has been trimmed for space):

```
# Enable PWM on GPIO12 (LED Emitter) = HAT pin32
label microSD (default)
   kernel /Image
   append console=ttyS2,115200n8 root=/dev/mmcblk1p3 ro ... <trimmed>
   fdtdir /
   fdt /ti/k3-am67a-beagley-ai.dtb
   fdtoverlays /overlays/k3-am67a-beagley-ai-pwm-epwm0-gpio12.dtbo
   initrd /initrd.img
```

   - If you want to load support for multiple overlays at once, you can add additional "fdtoverlays" lines, or list them as a space separated list on one line.
   - Reboot the board

```
(byai)$ sudo reboot
```

   - This setting will persist so you only need to edit the file once.

2. Each time you reboot the board, you must configure the HAT pin PWM symlink pin using the `beagle-pwm-export` command:

```
(byai)$ sudo beagle-pwm-export --pin hat-32
```

   - We are using GPIO12 which is HAT pin 32.

4. View the PWM files:

```
(byai)$ ls /dev/hat/pwm/GPIO12
capture  duty_cycle  enable  period  polarity  power  uevent
```

5. Set the period of a cycle (via `period`, in ns), duration of each "on" pulse (`duty_cycle`, in ns), and then enable it (`enable`, a 1 for on):

```
(byai)$ cd /dev/hat/pwm/GPIO12/
(byai)$ echo 0       > duty_cycle
(byai)$ echo 1000000 > period
(byai)$ echo 500000  > duty_cycle
(byai)$ echo 1       > enable
```

   - If you see the error: "`echo: write error: Invalid argument`" it means that you are writing a value that is not allowed based on the current configuration. By setting the duty_cycle to zero each time first it helps ensure you can set values correctly.

     Constraints:

     - `duty_cycle` <= `period`
     - `0 <= period <= 469,754,879`

   - Note on times:
     1 second
     = 1,000 miliseconds [ms]
     = 1,000,000 microseconds [us]

= 1,000,000,000 nano-seconds [ns]

6. Turn off with:
```
(byai)$ echo 0 > enable
```

7. Make it flash slowly (4 Hz)

```
(byai)$ cd /dev/hat/pwm/GPIO12/
(byai)$ echo 0         > duty_cycle
(byai)$ echo 250000000 > period
(byai)$ echo 125000000 > duty_cycle
(byai)$ echo 1         > enable
```

8. Troubleshooting:

   - If `beagle-pwm-export` fails with a message like the following, it likely means that the overlay is not yet enabled in the `extlinux.conf` file.
```
$ sudo beagle-pwm-export --pin hat-33
Unknown pin name: [--pin hat-33]
Possible PIN Options:
---------------
gpio12.pin
hat-32.pin
---------------
Possible Overlay Options:
---------------
k3-am67a-beagley-ai-pwm-ecap0-gpio12.dtbo
k3-am67a-beagley-ai-pwm-ecap1-gpio16.dtbo
k3-am67a-beagley-ai-pwm-ecap1-gpio21.dtbo
k3-am67a-beagley-ai-pwm-ecap2-gpio17.dtbo
k3-am67a-beagley-ai-pwm-ecap2-gpio18.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio12.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio14.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio15-gpio12.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio15-gpio14.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio15.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio5-gpio12.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio5-gpio14.dtbo
k3-am67a-beagley-ai-pwm-epwm0-gpio5.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio13.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio20.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio21-gpio13.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio21-gpio20.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio21.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio6-gpio13.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio6-gpio20.dtbo
k3-am67a-beagley-ai-pwm-epwm1-gpio6.dtbo
---------------
```

   - While trying to change the period (or other operations), if you get:
`"bash: echo: write error: Invalid argument"`

     - You may be trying to change `period` to a value less than the duty cycle. First change `duty_cycle` to 0, then retry your change.
     - You may be trying to change the period to a value greater than its maximum.
     - You may be trying to change the period for a PWM channel when the other channel that shares that PWM timer has already set the period (both channels running on the same timer must have the same period, but may have different `duty_cycle` values). For

example, if you have set the period for PWM0 channel A and then later try to change the period for PWM0 channel B, you are likely to get this warning.

Workarounds include:

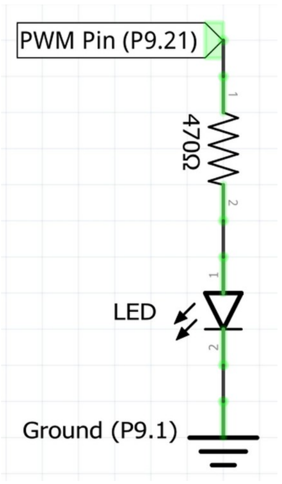
- Set both the A or B channels to the desired period, and then set the other channel to that exact same period. You may alter the `duty_cycle` independently.

3. PWM an LED *[Optional]*

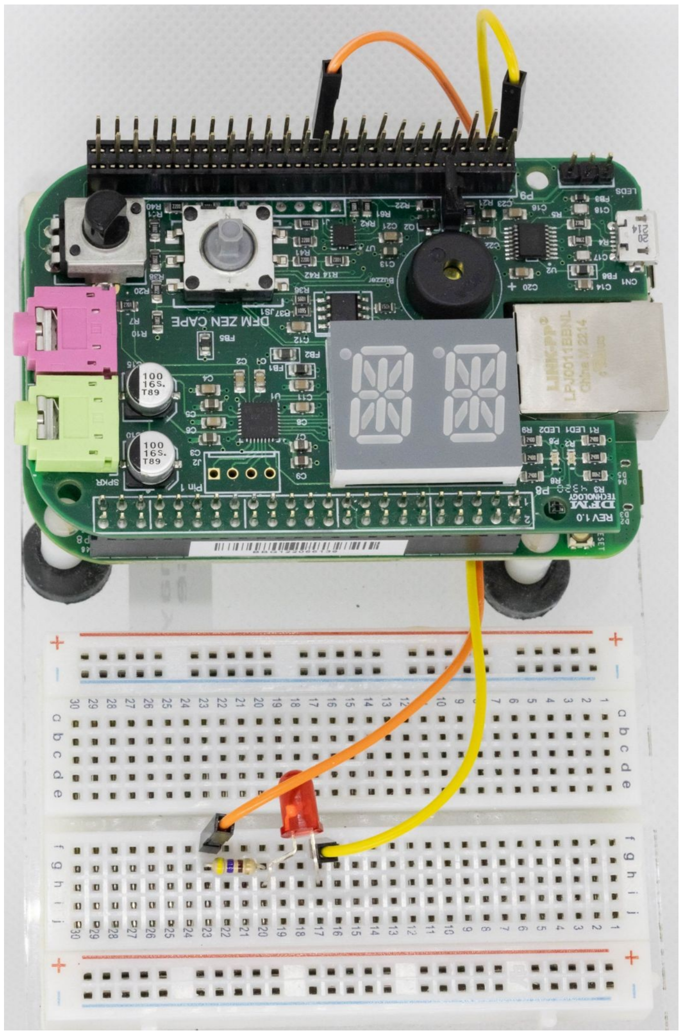
This is not necessary for the class assignments; it may be useful for the project.

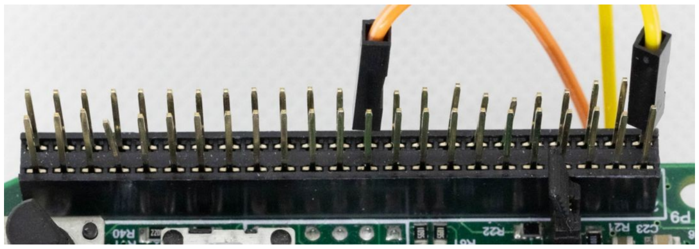
We can wire a discrete LED (single component) into a PWM pin on the target to turn on and off the LED at a desired frequency.

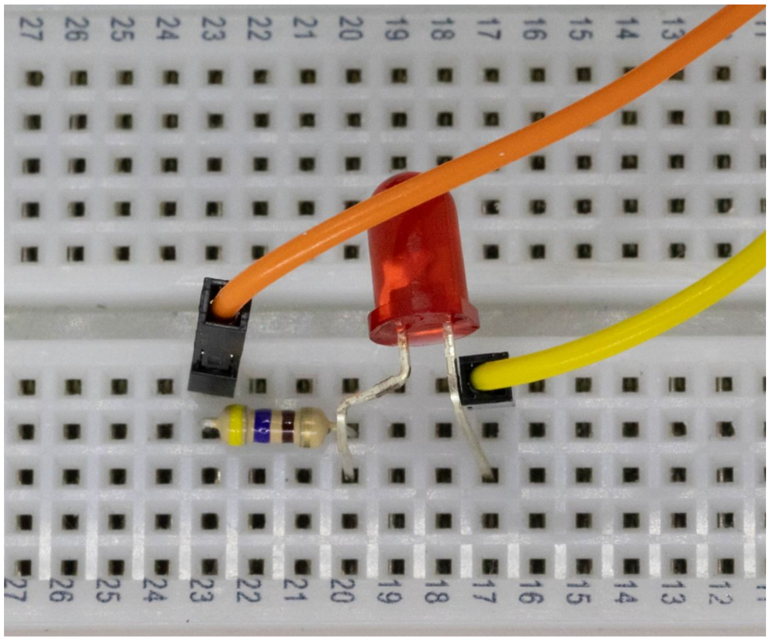
1. **Critical Tips**
   - Do all your wiring and wiring changes with the power to the target turned ***off*** (safest!)
   - Try to have some space between the components you place on the breadboard to prevent them from unexpectedly shorting (touching the wire leads). For example, on the breadboard, wire your resistor into column A, and your LED into column E so that they don't touch.
   - Note that it's OK for the plastic coated parts of the wires to touch; it's only the bare metal that can conduct electricity.
   - Avoid static! Before working on the components, try grounding yourself by touching some unpainted metal on your computer, such as the USB port.
2. Turn power off to the target.
3. If not done so already, place breadboard onto plastic mounting plate beside the BeagleY-AI.
   - Peel paper off back of breadboard and stick it to the mounting plate; discard extra metal plate if any.
4. The circuit we are going to wire up is shown on the right. The LED is connected in series (one into the next) with the 470 ohm resistor. Note the LED has one long and one short lead (wire).
   - Connect the "top" of the 470 ohm resistor to an available PWM pin: (Diagram refers to P9.21 of an older board).
   - Connect the "middle" between the two components together. This will be one end of the resistor (either end), and **the longer** lead of the LED.
   - Connect the "bottom" of the LED (**the shorter** lead) to ground: (Diagram refers to P9.1 of older board).
   - It does not matter whether the LED or the resistor is "on-top" (i.e., connected to the PWM pin).



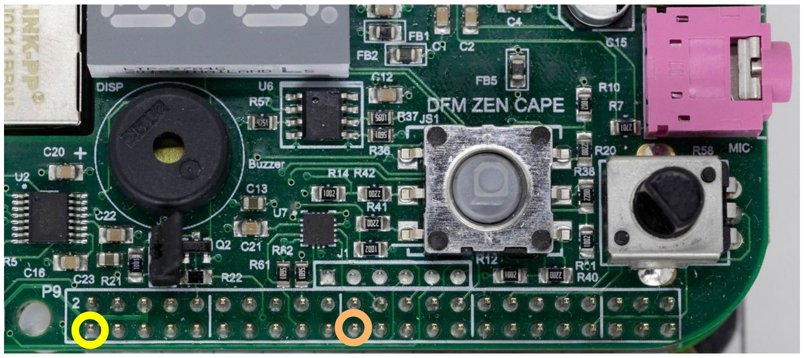
5. Here are some photos of the final wiring on an older board.
   - Photos taken from the "top" of the board, so P9 by the POT and joystick on the Zen cape.
   - Wires connect to breadboard in the order: P9.21 (PWM), P9.1 (Ground). Longer lead of LED has been bent.
   - Note the LED has been bent over onto its side so that it will flash down onto the breadboard, as needed if you are trying to have the LED light be sampled by a light sensor.







6. To the right is a closeup of how to count the P9 header.
   - Board in rotated so BBG is “down” and breadboard is “up.”
   - P9 pins are numbered from 1 on the left bottom.
   - Count from left, bottom row to top row. Bottom row is all odd numbers; top is even.



Location of P9.1 (GND) and P9.21 (PWM) 48-pin P9 header.