Servo & PWM Guide: RS002B Micro Servo

For Linux kernel BeagleBone Green, and Zen Cape Red

by Sen Wang.

This document guides the user through:

1. Wiring up the RS002B servo motor.
2. Set up PWM.
3. Turn the direction of servo motor from terminal.

Table of Contents

Formatting:

1. Host(computer) commands starting with (host)$ are Linux console commands:

    (host)$ echo “Hello world”

2. Target (BeagleBone) commands start with (bbg)$:

    (bbg)$ echo “On embedded board”

3. Almost all commands are case sensitive.

1. Servo Motor Basics

The direction of the servo can be controlled by PWM. This is because internally, the servo motor has mechanism that provides the feedback to its knowledge of position.

1.1 PWM Basics

First, we need to know PWM. There are two concepts, “period” and “duty cycle”. In the context of signals, “period” refers to the time it takes to complete a full cycle of its repetitive pattern. Generally, in the figure below, “period” is the time between the first vertical bar and the third vertical bar, or the time between third vertical bar and the fifth vertical bar.

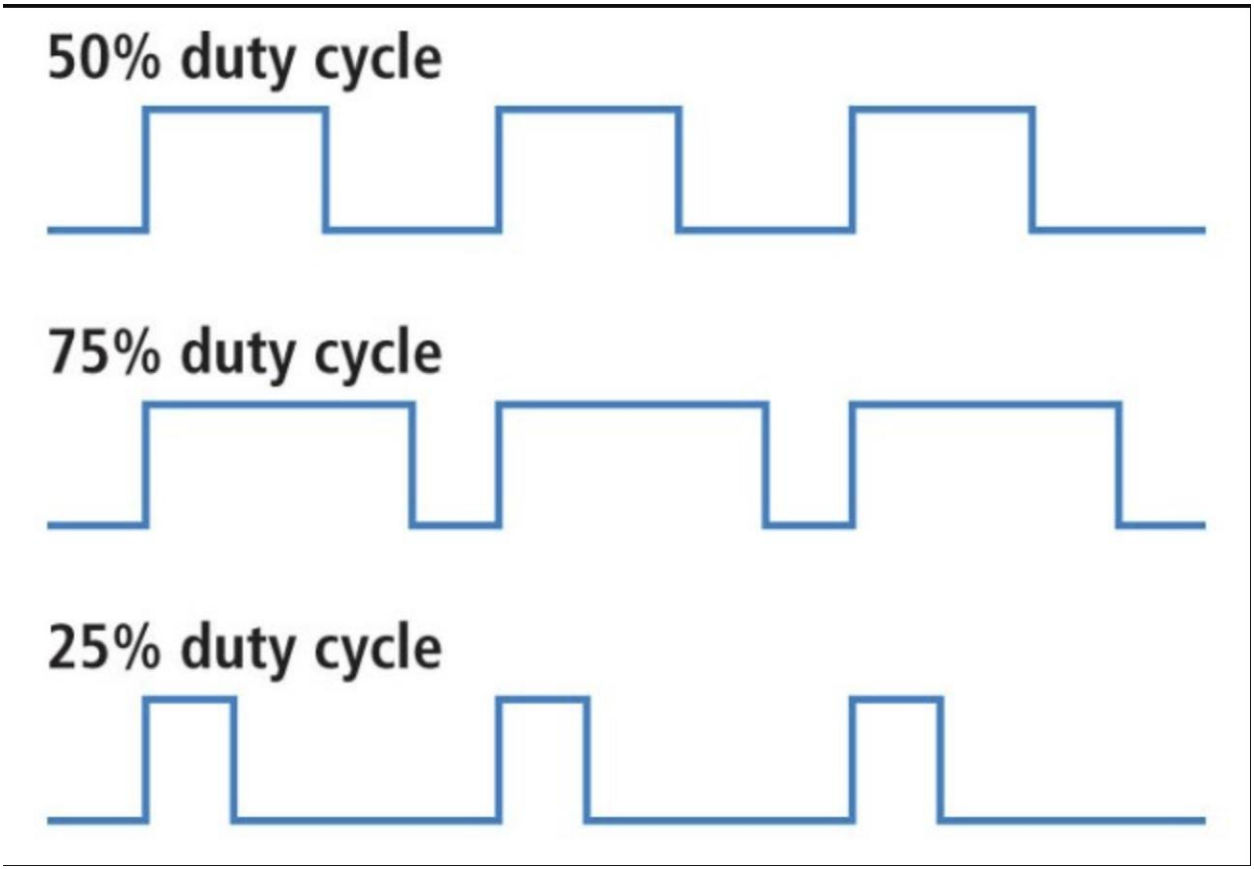


Figure 1 PWM with different duty cycle, from https://learn.sparkfun.com/tutorials/pulse-width-modulation/all

On the other hand, “duty cycle” is the ratio of the duration of the active(on) state of a signal to the total period of signal. For example, in the above figure, 50% duty cycle means that half of the time is signal is high during one period. 75% of duty cycle means 75% of the time the signal is high during one period.

Overall, these two parameters would allow us to generate waves shown in the above figure.

In our context, we are going to set period 20,000,000 which represents 20ms, or 20,000,000 nano seconds. In this way, we can then turn the direction of our servo from 0 to 180 degrees by setting duty cycle from 500,000 to 2,500,000. In addition, 1,500,000 of duty cycle under 20,000,000 period is 90 degrees.

1.2 Wiring Up Servo

Figure 2 shows the basic wiring up for servo to connect to BeagleBone Green. The servo has three wires, red for power, yellow for signal (waves), and brown for ground. In order to power up the servo, the voltage we need is between 4.8V and 6V, so we need to connect the red wire in the servo with either P9_07 or P9_08 pin on the Zen cape which generates 5V. Then connect the brown wire on the servo with either P9_01 or P9_02 on the Zen cape which corresponds to the ground. Finally connect the yellow wire on the servo with either P9_21 or P8_13 on the Zen cape which generates signal(waves). Also, we need to wire up two resistors on the yellow wire, each has 470 ohms. Wire up the other one in the same way as well. Then we can control two servos by generating waves through pin P9_21 and P8_13.

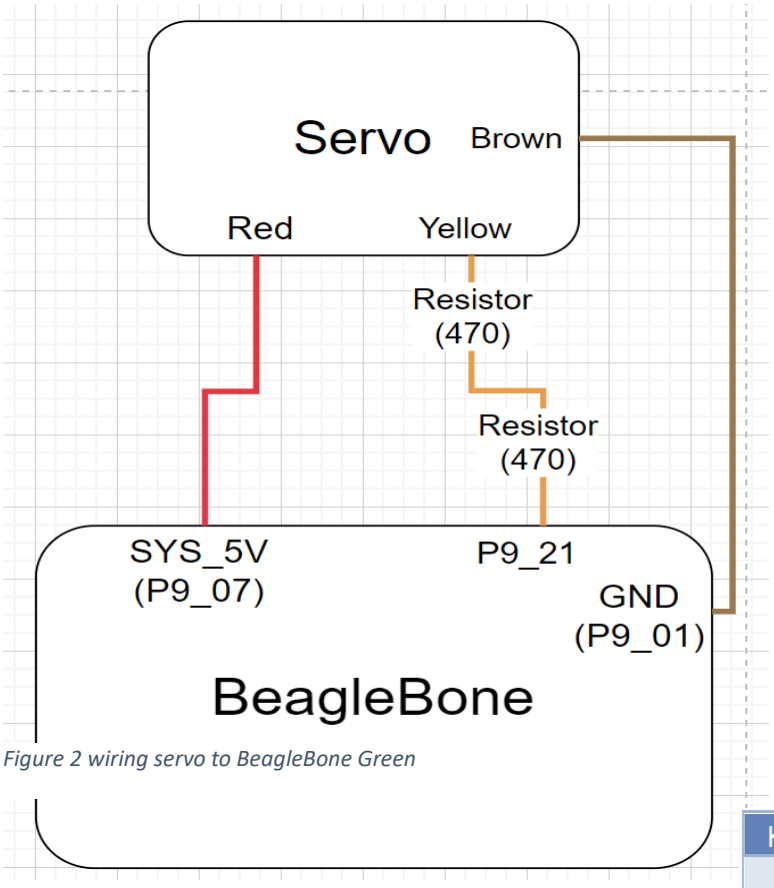


Figure 2 wiring servo to BeagleBone Green

Head_pin	$PINS	ADDR/OFFSET	Name
P9_01			GND
P9_02			GND
P9_03			DC_3.3V
P9_04			DC_3.3V
P9_05			VDD_5V
P9_06			VDD_5V
P9_07			SYS_5V
P9_08			SYS_5V

We can also calculate the current we have in one servo. The function we can use V = I * R, where I is current, and R is resistor.

Given, V = 5, R = 470 + 470 = 940. Then the current would equal to I = V/R = 0.00532A.

2. Setting PWM

The PWM channels that are sharing the same timers, cannot change their period

Zen Cape Use	PWM Channel	BBB Pin	Linux Path[1]	Notes
Buzzer	PWM-0A	P9-22	/dev/bone/pwm/**0/a/**	These two share a PWM timer: Period must be the same; duty cycle is independently.
unused	PWM-0B	P9-21	/dev/bone/pwm/**0/b**	
Blue LED	PWM-1A	P9-14	/dev/bone/pwm/**1/a**	These two share a PWM timer.
Red LED	PWM-1B	P9-16	/dev/bone/pwm/**1/b/**	
Green LED	PWM-2A	P8-19	/dev/bone/pwm/**2/a**	These two share a PWM timer.
unused	PWM-2B	P8-13	/dev/bone/pwm/**2/b**	

Figure 3 BeagleBone's PWM Channel

independently, but the duty cycles can be changed independently. In our context, we are going to use different timers, so wo will not encounter sharing the same timer's problem. The PWM Channels we are going to use are PWM-0B and PWM-2B. These two channels correspond to pin P9.21 and pin P8.13

P9-21 and P8-13 are the pins on Zen Cape, and each of them has different modes. One of the modes is pwm. So, at the beginning, we need to set both pins as pwm. Run the following command on the target to set pins.

(bbg)$ config-pin P9.21 pwm

(bbg)$ config-pin P8.13 pwm

Check which mode are these two pins are currently at, run the following command:

(bbg)$ config-pin -q P9.21

(bbg)$ config-pin -q P8.13

3. Turn the Direction of Servo from Terminal

After the two pins are set to pwm mode, then we can start to turn the directions from terminal.

First, we need to go to the corresponding directory that contains the period of the first servo, assume the first servo is connected to P9.21. Run the following command:

(bbg)$ cd /dev/bone/pwm/0/b

(bbg)$ ls

See if there is a file named "period"

The Initial value in "period" should be 0.

(bbg)$ cat period

Then we are going to set period to 20,000,000 by running the following command:

(bbg)$ echo 20000000 > period

(bbg)$ cat period

Check the value of period, see if it is equal to 20,000,000.

Then run the following command to set duty cycle to 1500,000 and check the value.

(bbg)$ echo 1500000 > duty_cycle

(bbg)$ cat duty_cycle

Then turn the pin on by running the following:

(bbg)$ echo 1 > enable

After we put a '1' into enable, then the servo that is connected to P9.21 should be turned roughly to point to the middle (90 degrees).

We can also turn the pin off by the following:

(bbg)$ echo 0 > enable

The other servo can also be turned in the same way, except we need to change to the corresponding folder that mapped to P8.13 by running the following:

(bbg)$ cd /dev/bone/pwm/2/b

(bbg)$ ls

(bbg)$ echo 20000000 > period

(bbg)$ cat period

(bbg)$ echo 950000 > duty_cycle

(bbg)$ cat duty_cycle

(bbg)$ echo 1 > enable

4. Trouble Shooting:

If there is a "Permission denied" error occurs, then probably need to do a "sudo tee", or need to exported PWM for that pin.

"Invalid argument" what trying to echo 1 > enable. This is because you need to set the value for period and duty_cycle first.

Unable to change period. This is because by design, the PWM Channels are shared. For example, P8_13 share a PWM Channel with P8_19, P9_21 shares a PWM Channel with P9_22. So, firstly set the period to the desired period, and then change duty_cycle independently.

When trying to change the period, and get the following error:

“bash: echo: write error: invalid argument”. This might because that the period cannot be lees than duty_cycle. So, change duty_cycle to 0, then change period, then change duty_cycle to the desired value.

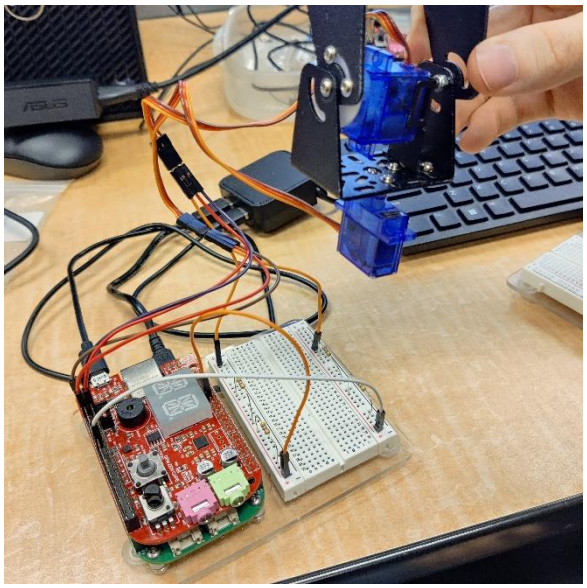
Also, check if the corresponding pin is configured to the right mode. This can be done by the following commands:

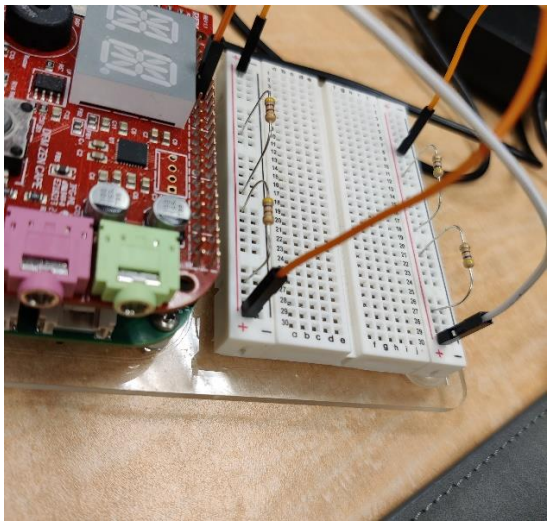
(bbg)$ config-pin -q P9.21

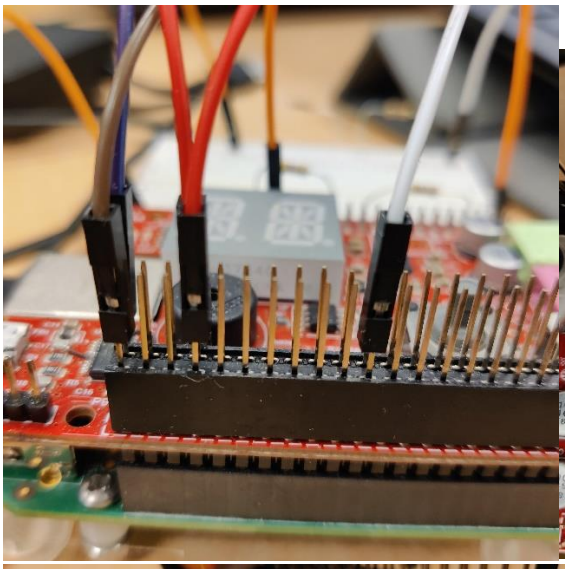
(bbg)$ config-pin -q P8.13

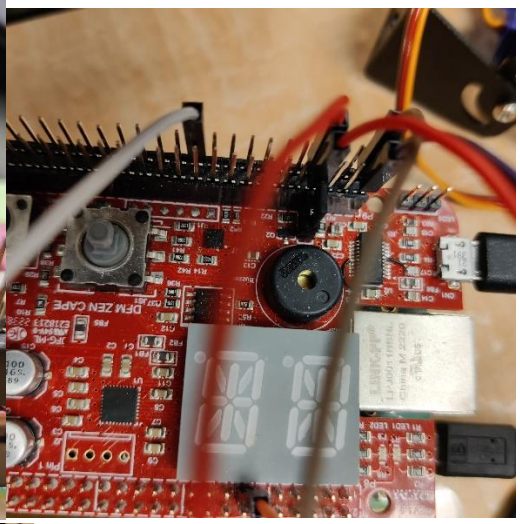
If the mode is not pwm, then set thses two pins to pwm mode.

The following pictures show how the wire up looks like when all wire up finished.








DEM ZEN CAPE