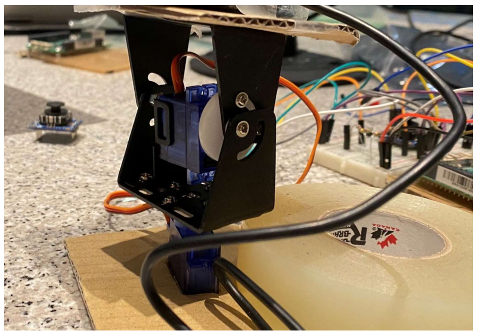
Servo motor guide for pan/tilt kit

By Amaan Khan, Jason Hong, Zakrye Osmond, Sammy Kaspar, Fall 2022 ENSC 351



Parts required:

- Beaglebone Green (BBG)
- Pan/tilt (THING the case)
- 9g Micro Servo(x2)
- 1kΩ Resistor(x2)
- Wires

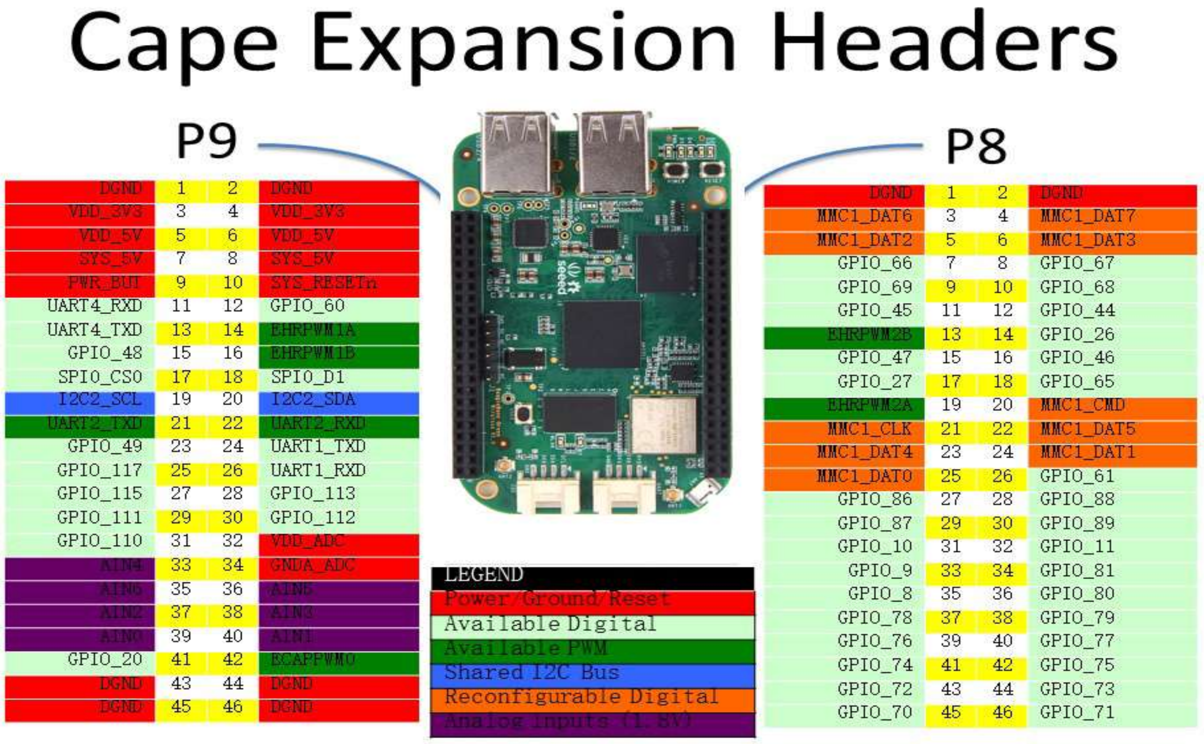
Table of contents:

1. PWM usage in servo

In order to control the motors we will use pwm (pulse width modulation)

To summarize, the way pwm works with motors is by using electrical pulses of variable length in order to move the motor to a specific position.
https://www.jameco.com/Jameco/workshop/Howitworks/how-servo-motors-work.html

For our motors we will be using a **period** of 20ms for the PWM. The result of this is that at a **duty cycle** of 0.5ms the motor will be in its minimum position and at a duty cycle of 2.5ms the motor will be at its maximum position.



P9			
DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPI0_CS0	17	18	SPI0_D1
I2C2_SCL	19	20	I2C2_SDA
UART2_TXD	21	22	UART2_RXD
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPPWM0
DGND	43	44	DGND
DGND	45	46	DGND

P8			
DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DAT0	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

LEGEND
🟥 Power/Ground/Reset
Available Digital
🟩 Available PWM
🟦 Shared I2C Bus
🟧 Reconfigurable Digital
🟪 Analog Inputs (1.8V)

Figure 1

As we can see from the headers, on the BBG the following pins can be used for PWM: P9_21,P9_22, P9_14,P9_16, P8_13,P8_19.

2. Setup to enable PWM

2024 Note from Dr. Brian
Changing uEnv.txt unnecessary; consult PWM guide.

1. In order to enable PWM we must change our bootfile uEnv.txt in /boot on the beaglebone
   BBG: cd /boot

2. Before we do this we should back it up
   BBG: cp uEnv.txt uEnv.bak

3. Edit the file
   BBG: sudo nano uEnv.txt

4. Replace the following lines

   ```
   ###Additional custom capes
   #uboot_overlay_addr4=<file4>.dtbo
   #uboot_overlay_addr5=<file5>.dtbo
   #uboot_overlay_addr6=<file6>.dtbo
   #uboot_overlay_addr7=<file7>.dtbo
   ```

   With

   ```
   ###Additional custom capes
   #uboot_overlay_addr4=<file4>.dtbo
   #uboot_overlay_addr5=<file5>.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
   ```

5. Save this file and reboot the BBG

3. Finding which pwmchip to use

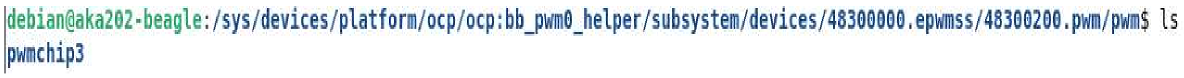
1. Depending on which pins you would like to use for PWM you will need to find which pwmchip is in control of those pins.

   P9_21 and P9_22 correspond to ePWM0 (48300200.pwm)
   P9_14 and P9_16 correspond to ePWM1 (48302200.pwm)
   P8_13 and P8_19 correspond to ePWM2 (48304200.pwm)

2. In order to find the pwmchip in control of those pins we need to go the epwm and see which pwmchip is shown (for this example we will find the pwmchip being used for P9_21 and P9_22)

   BBG: cd /sys/devices/platform/ocp/ocp:bb_pwm0_helper/subsystem/devices/48300000.epwmss/48300200.pwm/pwm

   Using ls in this directory should should which pwmchip is in use

   ```
   debian@aka202-beagle:/sys/devices/platform/ocp/ocp:bb_pwm0_helper/subsystem/devices/48300000.epwmss/48300200.pwm/pwm$ ls
   pwmchip3
   ```

   Figure 2

   Above we can see in figure 1, the pwmchip being used for P9_21 and P9_22 is pwmchip3.

3. After this export the pwmchip

   BBG: cd /sys/class/pwm/pwmchip3
   BBG: echo 1 > export

4. Servo circuit wiring

Wiring for servos where red is the power, brown is ground, and orange is PWM.

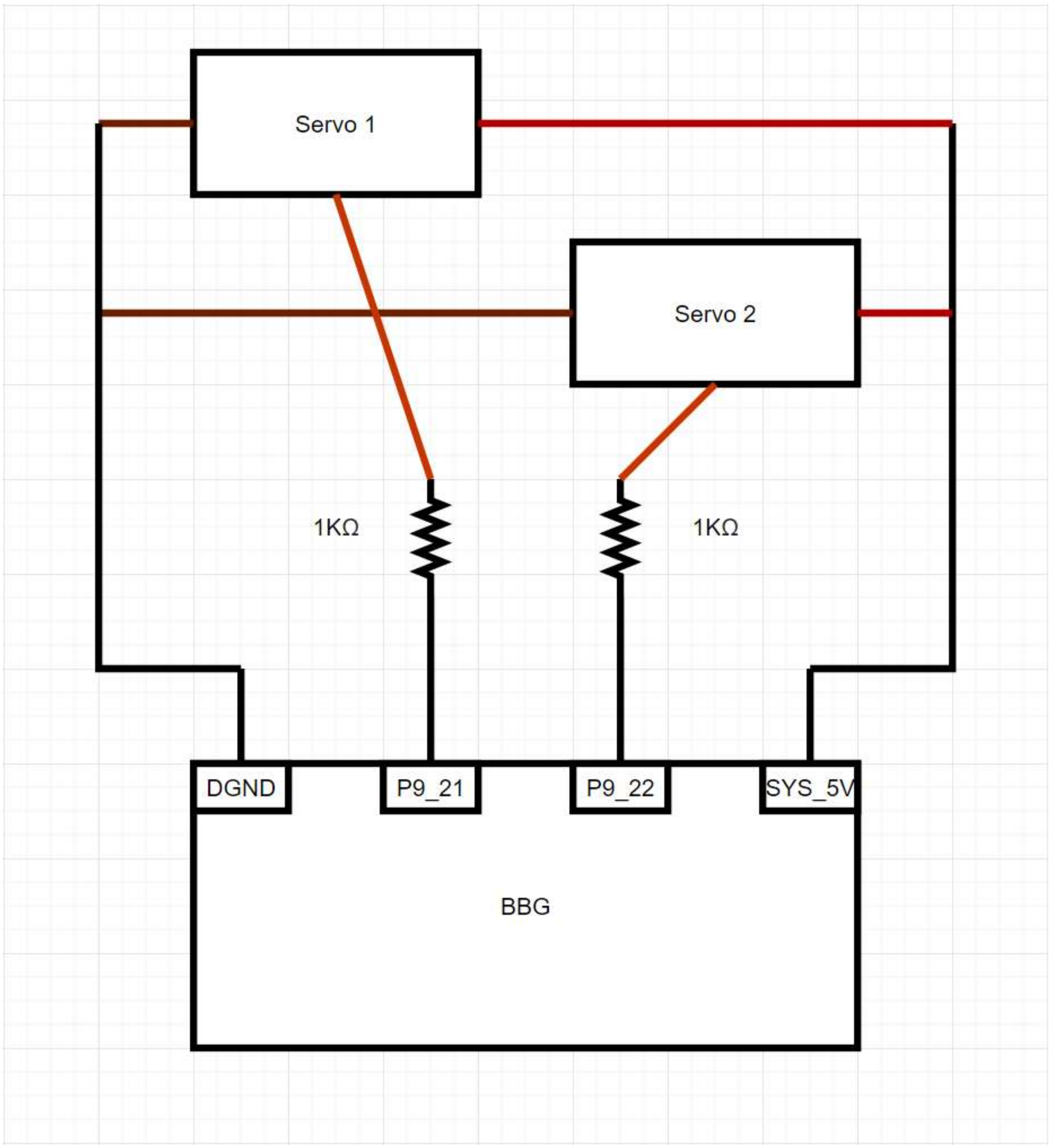


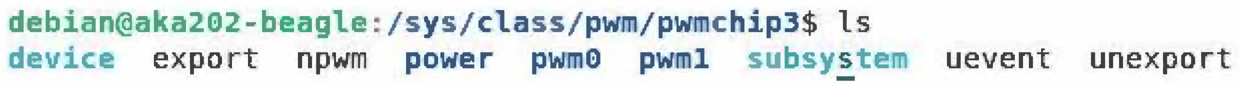
Figure 3

5. Terminal commands to control servos

After completing all the setup we can use the terminal to control the motors.

1. Since we set up P9_21 and P9_22 we can use their pwmchip and control their corresponding motors from there.

   BBG: cd /sys/class/pwm/pwmchip3

   ```
   debian@aka202-beagle:/sys/class/pwm/pwmchip3$ ls
   device   export   npwm   power   pwm0   pwm1   subsystem   uevent   unexport
   ```

   Figure 4

   In figure 3 we can see pwm0 and pwm1 which correspond to p9_21 and p9_22.

2. In order to control the motor which is connected to P9_21 switch to pwm0

   BBG: cd pwm0

3. First set the period to 20,000,000, and the duty cycle to 1,500,000, and enable it, the period corresponds to 20ms and the duty cycle corresponds to 1.5ms.

   BBG: echo 20000000 > period
   BBG: echo   1500000 > duty_cycle
   BBG: echo   1 > enable

   This set of commands should put the motor into its middle position, in order to do the same with the other motor, perform the same commands but in the pwm1 directory instead of pwm0.

4. In order to move the servo change the value of the duty cycle. Some values are: 500000 (0.5ms=100% CW), 1500000 (1.5ms=middle), 2500000 (2.5ms 100%CW)

   BBG: echo somevalue > duty_cycle

5. In order to turn off the servo

   BBG: echo 0 > enable

Notes:

With this set up the vertical turning is controlled by pwm1 and horizontal by pwm0.

In order to run the servos programmatically in C, something like fprintf could be used on period,duty_cycle and enable.

Troubleshooting

- If there are any issues with the uEnv.txt file it can be reverted back to what it was before using the uEnv.bak file.
- If the error “-bash: echo: write error: Device or resource busy” shows up when trying to export, the pwmchip has likely already been exported.
- If unable to enable a servo, make sure the values for period and duty cycle are set. If they are not set, enabling should not be possible.