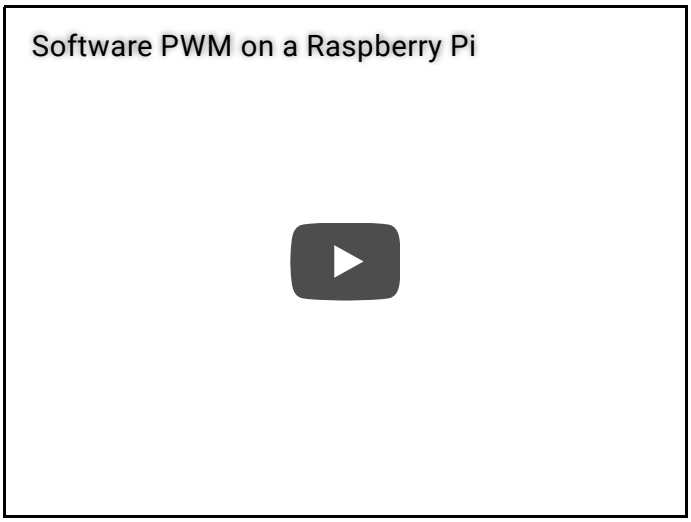
ozzmaker.com

Cisco, Linux, Raspberry Pi

Software PWM on a Raspberry Pi

| Mark Williams | 20 Comments



If you want to control the brightness of a LED, the speed of a DC motor or the direction of a servo, you will need PWM.

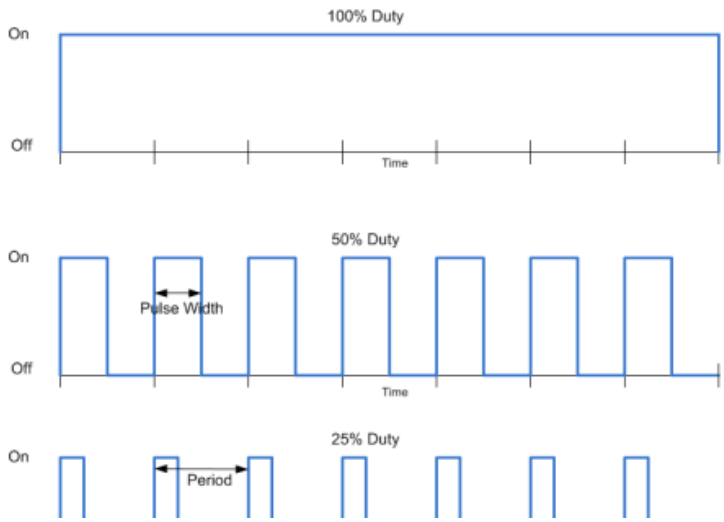
The video shows PWM being used to control the brightness of some LEDs.

Pulse-width modulation (PWM) is used to control the amount of power supplied to electrical devices, especially to DC motors, servos and LEDs.

PWM is able to achieve this by quickly turning off and on the power to the device. The measurement for this is duty cycle.

Duty cycles describes the proportion of 'on'; a low duty cycle corresponds to low power, because the power is off for most of the time. A high duty cycle corresponds to high power, because the power is on most of the time.

Duty cycle is expressed in percent, 50% is when the power is on half the time and 100% being fully on.



PWM can be performed in a number of ways on the Raspberry Pi.

Inbuilt hardware;
The Pi can perform PWM in hardware, but this can only be done on one pin (GPIO18) and when enabled it interferes with the audio jack. It is also hard to get working. Last time I looked, you had to recompile the kernel.

Software;
PWM can be performed in software. This is a very easy option. It isn't as precise as hardware PWM however in most instances software PWM will do.

External hardware;
There are external components that can be used to perform PWM. Eg Adafruit 16-Channel 12-bit PWM/Servo Driver – I2C interface – PCA96855

Software PWM with pi-blaster

In this post I will demonstrate how to use a modified version of Pi-Blaster, a software based PWM to control the brightness of some LEDs. I modified it to allow the pins for PWM to be specified at startup.

1. Download and compile pi-blaster.

```
pi@raspberrypi ~ $ git clone https://github.com/mwilliams03/pi-blaster.git
pi@raspberrypi ~ $ cd pi-blaster
pi@raspberrypi ~ $ make pi-blaster
```

2. Start Pi-Blaster.

We are going to use pi-blaster in the user space. Once started, it runs as a background process.
If you start pi-blaster without any parameters, it will enable PWM on the default pins.

```
pi@raspberrypi ~ $ sudo ./pi-blaster
```

The default pins are;

```
Channel number    GPIO number   Pin in P1 header
```

```
0               4               P1-7
1               17              P1-11
2               18              P1-12
3               21              P1-13
4               22              P1-15
5               23              P1-16
6               24              P1-18
7               25              P1-22
```

You can also specify the pins at start that you only want to use for PWM. To enable PWM only on pins 22, 24, 17 and 16;

```
pi@raspberrypi ~ $ sudo ./pi-blaster 22 24 17 16
```

3. Configure PWM on a pin.

Pi-blaster creates a special file (FIFO) in /dev/pi-blaster. Any application on your Raspberry Pi can write to it (this means that only pi-blaster needs to be root, your application can run as a normal user).

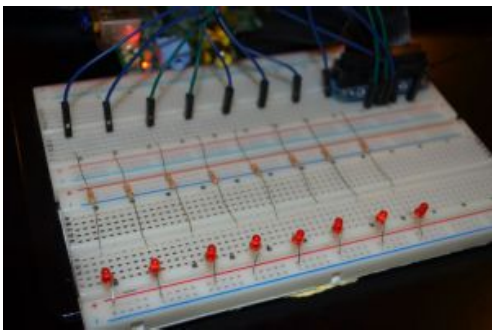
- To set pin1 to a PWM of 20%
  `echo "1=0.2" > /dev/pi-blaster`
- To completely turn off pin0:
  `echo "0=0" > /dev/pi-blaster`
- To completely turn on pin1:
  `echo "1=1" > /dev/pi-blaster`

4. Stop pi-blaster

Use killall to kill the pi-blaster process

```
pi@raspberrypi ~ $ sudo killall pi-blaster
```

5. Sample program used in the above video



You can write your own programs to control PWM just by writing to /dev/pi-blaster.
The code below was used to control the LEDs in the above video. And which is also shown to the right

Copy the code below into a file. E.g. pwd-led.c. And then compile;

```
pi@raspberrypi ~ $ gcc -o pwd-led pwd-led.c
```

```c
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>


#define DELAY 70000
#define LEVELS 7
#define INCREMENT  0.15
    FILE *fp;

void  INThandler(int sig)
{
    int pin;
        signal(sig, SIG_IGN);
    //turn all pins off
        for (pin = 0; pin < 8; pin++){
        fprintf(fp, "%i=%f\n", pin,0.0);
        fflush(fp);
    }
    fclose(fp);
        exit(0);
}

int main(void)
{
    signal(SIGINT, INThandler);

    // arrays used to control brightness of a row of LEDs. left then righ
    float left[] = {0,0,0,0,0,0,0,0,1,0.75,0.375,0.18,0.09,0.04,0.005,0,0
    float right[] = {0,0,0,0,0,0,0,0,0.005,0.04,0.09,0.18,0.375,0.75,1,0,(

    int pin = 0;
    float pwm = 0.0;

    // open the pi-blaster device file. If it fails, confirm that pi-blas
    fp = fopen("/dev/pi-blaster", "w");
    if (fp == NULL) {
        printf("Error opening file\n");
        exit(0);
     }

    // create the PWM values which are used for the animation going in on
    float pwdPower[LEVELS*4];
    float i;
    int j;
    int count = 7;

    for (j = 0; j < LEVELS ;  j++){
```

```
49          pwdPower[j] = 0.0;
50      }
51
52      for (i = 0.0; i < 1 ; i +=INCREMENT){        if (i>1.0)i=1.0;
53          pwdPower[count] = i;
54              count++;
55      }
56
57      for (i = 1; i >0.00001 ; i -=INCREMENT){
58          if (i<0.0)i=0.0;
59          pwdPower[count] = i;
60              count++;
61      }
62
63          for (j = 0; j < (LEVELS) ;  j++){
64                  pwdPower[count] = 0.0;
65          count++;
66          }
67
68  ////////////////////////////////////////////////////////////////////
69
70      //This section illustrates different brightness levels
71      // using PWM with a 2 second puase between levels.
72
73      // Enable all pins using the value 0.002 for PWM
74      for (pin = 0; pin < 8; pin++){
75                  fprintf(fp, "%i=%f\n", pin,0.002);
76                         fflush(fp);
77      }
78      sleep(2);
79
80      // Enable all pins using the value 0.03 for PWM
81      for (pin = 0; pin < 8; pin++){
82                      fprintf(fp, "%i=%f\n", pin,0.03);
83                          fflush(fp);
84      }
85      sleep(2);
86
87      // Enable all pins using the value 0.05 for PWM
88      for (pin = 0; pin < 8; pin++){
89                      fprintf(fp, "%i=%f\n", pin,0.5);
90                          fflush(fp);
91      }
92      sleep(2);
93  ////////////////////////////////////////////////////////////////////
94
95      // This section will cycle through the power levels 5 times.
96      for (j = 0; j < 5 ; j++){
97          for (i = 0.0; i < 1 ; i +=INCREMENT){
98              for (pin = 0; pin < 8; pin++){                          fprintf(
99              for (pin = 0; pin < 8; pin++){
100                         fprintf(fp, "%i=%f\n", pin,i );
101                             fflush(fp);
102             }
103         usleep(DELAY);
104
105         }
106     }
107 ////////////////////////////////////////////////////////////////////
108
109     // The below section performs the animations
110     int c = 0;
111     int g;
112     for(g = 0; g < 2; g++){       for(c = 16; c>= 0; c--){
113             for (pin = 0; pin < 8  ; pin++){
114                             fprintf(fp, "%i=%f\n", pin, right[pin+c]);
```

```
                                fflush(fp);
                }
            usleep(DELAY);
            }
            for(c = 0; c < 16; c++){
                for (pin = 0; pin < 8; pin++){
                            fprintf(fp, "%i=%f\n", pin, left[pin+c]);
                                fflush(fp);
                }
            usleep(DELAY);
            }
        }

        for(g = 0; g < 4; g++){
            for(c = 0; c < (LEVELS*3); c++){
                for (pin = 0; pin < 8; pin++){
                                fprintf(fp, "%i=%f\n", pin, pwdPower[pin+c]);
                            fflush(fp);
                }
                usleep(DELAY);
            }
        }

        //turn all pins off
            for (pin = 0; pin < 8; pin++){
            fprintf(fp, "%i=%f\n", pin,0.0);
            fflush(fp);
        }
        fclose(fp);
        return 0;
    }
```

Related

PiBBot - Robotic Project - Phase 1 - Testing Components
February 1, 2013
In "Balancing Robot"

Success with a Balancing Robot using a Raspberry Pi
April 18, 2013
In "Balancing Robot"

Camera Pan and Tilt control with TFT and Touchscreen
March 24, 2015
In "PiScreen"

LED LINUX PWM RASPBERRY PI

20 THOUGHTS ON “SOFTWARE PWM ON A RASPBERRY PI”

Pingback: Software PWM on a #RaspberryPi | Raspberry PiPod

Impossible
SEPTEMBER 29, 2013 AT 2:42 PM
Thanks for this post, the code at the bottom of the page. What language is it and how would I make my Pi parse it?

Impossible
SEPTEMBER 29, 2013 AT 8:23 PM
I think I have found out what my issue was. After guessing it was C, I found out that the "#include" were empty. I filled this in then performed the following.

gcc -o program markspwmdemo.c
chmod +x program
./program

Daniel
NOVEMBER 17, 2013 AT 6:58 PM
Hi,
can you tell me which includes are missing?
Kind regards
Daniel

★ Mark Williams
NOVEMBER 18, 2013 AT 2:15 AM
Done… sorry about that.

Mark

jas
DECEMBER 4, 2013 AT 6:37 PM
What is the frequency of the PWM output and can it be adjusted?

★ Mark Williams
DECEMBER 4, 2013 AT 9:57 PM
Use ./pi-blaster to view the current frequency;
sudo ./pi-blaster
Using hardware: PWM
Number of channels: 8
PWM frequency: 100 Hz
PWM steps: 1000
Maximum period (100 %): 10000us

Minimum period (0.100%): 10us

You can adjust these by changing a few defines at the top of the source code:

NUM_SAMPLES: The number of steps
SAMPLE_US: The time of one step (minimum period)

If you do not neet a resolution of 1000 steps (approximately equivalent to a 10 bit DAC), then you can reduce the number of samples or increase the duration of the steps.

---

vierbergenwout
DECEMBER 17, 2013 AT 2:23 PM
Is is possible that I can't make/install pi-blaster?
Or do I miss some logical stepps you hadn't mentioned?

Wout

★ Mark Williams
DECEMBER 17, 2013 AT 10:24 PM
What issue are you having? Can you show me the error message?

---

vierbergenwout
DECEMBER 18, 2013 AT 9:30 PM
After retrying and deleting everything again the "nothing to make" error dissepeared. So it's working! I did all the time the same, so I still don't know what went wrong…

---

promet
JANUARY 31, 2014 AT 1:31 AM
Hi I've been playing with motor control in Python with my pi. I was formerly using GPIO direct pin addressing, but wanted to try pi-blaster to reduce CPU usage.

I have pi-blaster installed, and I can see that it's running, as it reports its details, as per mwilliams03's post above.

Also, when I execute my Python code, I can "cat /dev/pi-blaster" and see the appropriate values written to the file that I expect from my code. The values don't seem to generate output on the designated pins though.

That is, the output voltages for my motor control pins, according to my multimeter, appear to be zero, where I would expect them to be corresponding voltage values.

So, pi-blaster seems to be writing to it's file correctly, but no corresponding outputs. Anyone have any thoughts?

I'd post the code here, but I'm not sure if WordPress supports "code" tags. 😉

promet
JANUARY 31, 2014 AT 1:38 AM
In case anyone would care to have a look though:

http://paste.ubuntu.com/6847008/

wrock
APRIL 1, 2014 AT 1:06 PM
Do oyu have some test result, on how exact the timings are? Are there any timing varities if you've got more clients on it? e.g 20 servos?
greetings

prometxet
APRIL 29, 2014 AT 10:52 AM
Hi wrock,

I don't have any test result. I would expect though, that some voltage would be visible upon writing to "/dev/pi-blaster" though, no?

Pingback: Software PWM – Raspberry PI | Robofun Blog

Chris
JUNE 18, 2015 AT 7:54 AM
Ok complete noob here. Trying to test this out. So I wish to try the above code. I'm guessing that I need something in front of the incules lines at the top. And we're are you writing this in the Python she'll or the terminal? Thanks

★ Mark Williams
JUNE 18, 2015 AT 10:55 AM
copy the code above into a file E.g. pwm-led.c and then compile;

gcc -o pwd-led pwd-led.c

Jim Julian
MAY 17, 2016 AT 3:38 AM
Hello,

Ran across your fork while investigating linear actuator control options for a solar array using my new Raspberry Pi.
Everything went well until I tried the Make Install command :

pi@raspberrypi:~/pi-blaster-master $ sudo make install
make[1]: Entering directory '/home/pi/pi-blaster-master'
/bin/mkdir -p '/usr/sbin'
/usr/bin/install -c pi-blaster '/usr/sbin'
/bin/mkdir -p '/lib/systemd/system'
/usr/bin/install -c -m 644 pi-blaster.service '/lib/systemd/system'
make install-data-hook
make[2]: Entering directory '/home/pi/pi-blaster-master'
systemctl enable pi-blaster
Failed to execute operation: Bad message
Makefile:846: recipe for target 'install-data-hook' failed
make[2]: *** [install-data-hook] Error 1
make[2]: Leaving directory '/home/pi/pi-blaster-master'
Makefile:771: recipe for target 'install-data-am' failed
make[1]: *** [install-data-am] Error 2
make[1]: Leaving directory '/home/pi/pi-blaster-master'
Makefile:719: recipe for target 'install-am' failed
make: *** [install-am] Error 2
pi@raspberrypi:~/pi-blaster-master $

If you have any ideas on what I can do or should do differently, please let me know.
Thank you,
Jim Julian

Rick
SEPTEMBER 10, 2017 AT 12:45 PM
Invalid channel number 25
echo "25=0.2" > /dev/pi-blaster
I just wanted one pin pwm gpio 25
before I start digging in to source code, anyone know why am i getting is error?

Rick
SEPTEMBER 10, 2017 AT 11:26 PM
OK, I figure it out.
sudo pi-blaster/pi-blaster
Using hardware: PWM
Number of channels: 8
PWM frequency: 100 Hz
PWM steps: 1000
Maximum period (100 %): 10000us
Minimum period (0.100%): 10us
By using default it shows 'Number of channels: 8', so I want to control gpio 25 as above table shows channel 7, pin 22, gpio 25,
echo "7=0.01" > /dev/pi-blaster write out channel number instead pin number it seems works. But default controls eight gpio pins and I just need one pwm gpio25 and other gpio s needed for other purpose, so little more fiddling, "pi-blaster 25" returns 'Number of channels 1' so writing file echo "0=0.01" > /dev/pi-

blaster channel index 0 as single gpio or the first gpio number, so I can use other gpio's for other purpose and have one pwm channel.
Thanks.