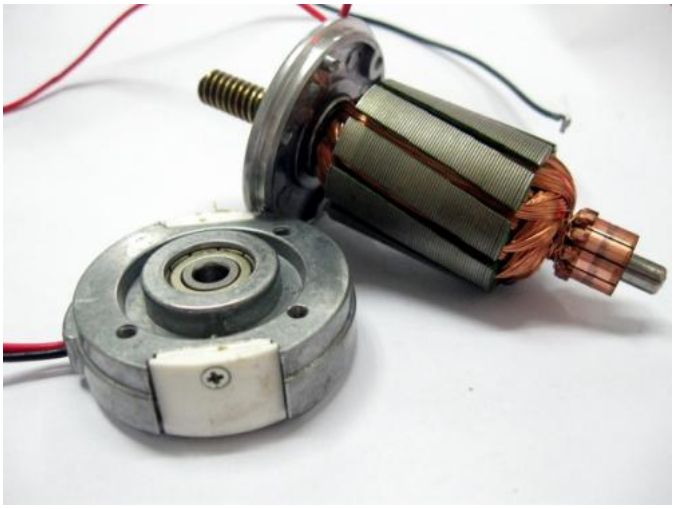
Changing PWM frequencies on Arduino controllers

 by OddBot Collected by 5 users
Tip/walkthrough

By OddBot @ January 29, 2014
Increase torque or reduce noise in DC brushed motors



By default, most Arduino pwm outputs are set to about 490Hz or 980Hz. These frequencies work quite well with small dc brushed motors but can cause unwanted noise or reduce motor torque in some cases.

The frequencies are generated as a result of the clock frequency (this tutorial assumes 16MHz) being divided by a pre-scaler. For 8Mhz clocks, divide the frequency by 2. Before you start changing frequencies there are a few things you must be careful of.

1. Many motor controllers cannot work at high frequencies and can short circuit when you change to high frequencies.
2. High frequencies increase inductive reactance of the motor windings. This can increase heat and decrease motor torque.
3. Timers may be used by other functions and libraries. Changing PWM frequencies can have unexpected side affects.

So before you start, check what is the maximum PWM frequency your motor controller can handle, check what timers are currently used (eg. Timer 0 used for Arduino timing functions, Timer 1 or 5 used for servos, Timer 2 used by tone() command.

Below are 2 list of timer settings. 1 for Arduinos using ATmega 8, 168, 328 and the other for Arduino's using ATmega1280 and ATmega2560 processors. Cut and paste the lines you need directly into your setup() function.

You should find that lower frequencies provide better torque for most small DC hobby motors. A frequency of 122.55Hz provides good torque without too much noise. A frequency of 30.64Hz give the best torque at low speeds but can cause noticable vibration in your robot.

Frequencies from 490.2Hz to 7812.5Hz are noisy and cause the motors to loose torque. Frequencies above 20000Hz cannot be heard by most humans although the pets might complain. Unless your motor and motor driver are designed for these frequencies then you risk damaging them. I found that the most small DC motors lost a lot of torque and tended to overheat.

Note: Divide the PWM frequency by 2 for an 8MHz clock, multiply by 1.25 for a 20MHz clock.

```
For Arduino Uno, Nano, Micro Magician, Mini Driver, Lilly Pad and any other board using ATmega 8, 168 or 328
//---------------------------------------------- Set PWM frequency for D5 & D6 -------------------------------

//TCCR0B = TCCR0B & B11111000 | B00000001;    // set timer 0 divisor to     1 for PWM frequency of 62500.00 Hz
//TCCR0B = TCCR0B & B11111000 | B00000010;    // set timer 0 divisor to     8 for PWM frequency of  7812.50 Hz
  TCCR0B = TCCR0B & B11111000 | B00000011;    // set timer 0 divisor to    64 for PWM frequency of   976.56 Hz
//TCCR0B = TCCR0B & B11111000 | B00000100;    // set timer 0 divisor to   256 for PWM frequency of   244.14 Hz
```

Search

Search
Shout Box Pop Out

User login

Username: *

Password: *

Log in

- Create new account
- Request new password

Log in using OpenID

Recent blog posts

- Miscellaneous items updates....
- AuMac, a multitask robot
- Good bye for a while.
- Wall-e MkII/Project 2: The master plans
- Toys for kids with disabilities
- The batteries are coming!
- Rocker-Bogies.
- I2C IO expander/servo controller using a picaxe 20x2
- Emotions and Motivations as behaviour architecture
- Wall-e MkII - the tinkering begins...

more

Latest weblinks

- SoCal MakerCon - Nov 8th 2014
- Linear travel motor
- Oh dear Jesus this is Terrifying
- Safety
- Octaworm
- Forth
- Humanoid Robot Pianist On Your PC
- The PiBot Website
- Hackaday judge and embedded guru: Jack Ganssle
- Engineering 101: It's all in the datasheet

Who's online

There are currently *21 users* and *21 guests* online.

Online users

- birdmun
- Mr.What
- bdk6

```
//TCCR0B = TCCR0B & B11111000 | B00000101;    // set timer 0 divisor to  1024 for PWM frequency of

//---------------------------------------- Set PWM frequency for D9 & D10 ------------------------------

//TCCR1B = TCCR1B & B11111000 | B00000001;    // set timer 1 divisor to     1 for PWM frequency of 31
//TCCR1B = TCCR1B & B11111000 | B00000010;    // set timer 1 divisor to     8 for PWM frequency of 3
  TCCR1B = TCCR1B & B11111000 | B00000011;    // set timer 1 divisor to    64 for PWM frequency of
//TCCR1B = TCCR1B & B11111000 | B00000100;    // set timer 1 divisor to   256 for PWM frequency of
//TCCR1B = TCCR1B & B11111000 | B00000101;    // set timer 1 divisor to  1024 for PWM frequency of

//---------------------------------------- Set PWM frequency for D3 & D11 ------------------------------

//TCCR2B = TCCR2B & B11111000 | B00000001;    // set timer 2 divisor to     1 for PWM frequency of 31
//TCCR2B = TCCR2B & B11111000 | B00000010;    // set timer 2 divisor to     8 for PWM frequency of 3
//TCCR2B = TCCR2B & B11111000 | B00000011;    // set timer 2 divisor to    32 for PWM frequency of
  TCCR2B = TCCR2B & B11111000 | B00000100;    // set timer 2 divisor to    64 for PWM frequency of
//TCCR2B = TCCR2B & B11111000 | B00000101;    // set timer 2 divisor to   128 for PWM frequency of
//TCCR2B = TCCR2B & B11111000 | B00000110;    // set timer 2 divisor to   256 for PWM frequency of
//TCCR2B = TCCR2B & B11111000 | B00000111;    // set timer 2 divisor to  1024 for PWM frequency of
```

For Arduino Mega1280, Mega2560, MegaADK, Spider or any other board using ATmega1280 or ATmega2560

```
//---------------------------------------- Set PWM frequency for D4 & D13 ------------------------------

//TCCR0B = TCCR0B & B11111000 | B00000001;    // set timer 0 divisor to     1 for PWM frequency of
//TCCR0B = TCCR0B & B11111000 | B00000010;    // set timer 0 divisor to     8 for PWM frequency of
  TCCR0B = TCCR0B & B11111000 | B00000011;    // set timer 0 divisor to    64 for PWM frequency of
//TCCR0B = TCCR0B & B11111000 | B00000100;    // set timer 0 divisor to   256 for PWM frequency of
//TCCR0B = TCCR0B & B11111000 | B00000101;    // set timer 0 divisor to  1024 for PWM frequency of


//---------------------------------------- Set PWM frequency for D11 & D12 -----------------------------

//TCCR1B = TCCR1B & B11111000 | B00000001;    // set timer 1 divisor to     1 for PWM frequency of 3
//TCCR1B = TCCR1B & B11111000 | B00000010;    // set timer 1 divisor to     8 for PWM frequency of
  TCCR1B = TCCR1B & B11111000 | B00000011;    // set timer 1 divisor to    64 for PWM frequency of
//TCCR1B = TCCR1B & B11111000 | B00000100;    // set timer 1 divisor to   256 for PWM frequency of
//TCCR1B = TCCR1B & B11111000 | B00000101;    // set timer 1 divisor to  1024 for PWM frequency of

//---------------------------------------- Set PWM frequency for D9 & D10 ------------------------------

//TCCR2B = TCCR2B & B11111000 | B00000001;    // set timer 2 divisor to     1 for PWM frequency of 31372.55 Hz
//TCCR2B = TCCR2B & B11111000 | B00000010;    // set timer 2 divisor to     8 for PWM frequency of  3921.16 Hz
//TCCR2B = TCCR2B & B11111000 | B00000011;    // set timer 2 divisor to    32 for PWM frequency of   980.39 Hz
  TCCR2B = TCCR2B & B11111000 | B00000100;    // set timer 2 divisor to    64 for PWM frequency of   490.20 Hz
//TCCR2B = TCCR2B & B11111000 | B00000101;    // set timer 2 divisor to   128 for PWM frequency of   245.10 Hz
//TCCR2B = TCCR2B & B11111000 | B00000110;    // set timer 2 divisor to   256 for PWM frequency of   122.55 Hz
//TCCR2B = TCCR2B & B11111000 | B00000111;    // set timer 2 divisor to  1024 for PWM frequency of    30.64 Hz


//---------------------------------------- Set PWM frequency for D2, D3 & D5 ---------------------------

//TCCR3B = TCCR3B & B11111000 | B00000001;    // set timer 3 divisor to     1 for PWM frequency of 31372.55 Hz
//TCCR3B = TCCR3B & B11111000 | B00000010;    // set timer 3 divisor to     8 for PWM frequency of  3921.16 Hz
  TCCR3B = TCCR3B & B11111000 | B00000011;    // set timer 3 divisor to    64 for PWM frequency of   490.20 Hz
//TCCR3B = TCCR3B & B11111000 | B00000100;    // set timer 3 divisor to   256 for PWM frequency of   122.55 Hz
//TCCR3B = TCCR3B & B11111000 | B00000101;    // set timer 3 divisor to  1024 for PWM frequency of    30.64 Hz


//---------------------------------------- Set PWM frequency for D6, D7 & D8 ---------------------------

//TCCR4B = TCCR4B & B11111000 | B00000001;    // set timer 4 divisor to     1 for PWM frequency of 31372.55 Hz
//TCCR4B = TCCR4B & B11111000 | B00000010;    // set timer 4 divisor to     8 for PWM frequency of  3921.16 Hz
  TCCR4B = TCCR4B & B11111000 | B00000011;    // set timer 4 divisor to    64 for PWM frequency of   490.20 Hz
//TCCR4B = TCCR4B & B11111000 | B00000100;    // set timer 4 divisor to   256 for PWM frequency of   122.55 Hz
//TCCR4B = TCCR4B & B11111000 | B00000101;    // set timer 4 divisor to  1024 for PWM frequency of    30.64 Hz


//---------------------------------------- Set PWM frequency for D44, D45 & D46 ------------------------

//TCCR5B = TCCR5B & B11111000 | B00000001;    // set timer 5 divisor to     1 for PWM frequency of 31372.55 Hz
//TCCR5B = TCCR5B & B11111000 | B00000010;    // set timer 5 divisor to     8 for PWM frequency of  3921.16 Hz
  TCCR5B = TCCR5B & B11111000 | B00000011;    // set timer 5 divisor to    64 for PWM frequency of   490.20 Hz
//TCCR5B = TCCR5B & B11111000 | B00000100;    // set timer 5 divisor to   256 for PWM frequency of   122.55 Hz
//TCCR5B = TCCR5B & B11111000 | B00000101;    // set timer 5 divisor to  1024 for PWM frequency of    30.64 Hz
```

Differences in timer frequencies:
You will see that timer 0 frequencies are double that of the other timers. This is because Timer 0 is used by Arduino for timing functions such as delay(), millis(), micros() and more. To get better accuracy, The Arduino IDE sets Timer 0 to fast PWM mode. It should be noted that for these timing functions,the Arduino software automatically compensates for 8MHz, 16MHz and 20MHz clocks. The default prescaler setting is always 64.

For all other timers, the Arduino IDE sets the timer mode to phase correct PWM. This causes the counter to count from 0 to 255 and then from 255 back to 0 again. This provides a more symetrical output but at half the frequency. The default timer settings are configured in the **wiring.c** file of your Arduino installation.

If you are looking for a timer library for generating timer interrupts or custom PWM outputs then check out this library: http://playground.arduino.cc/Code/Timer1. There are 2 versions, one for timer 1 and another for timer 3. It should be easy to modify these libraries to work with the other timers if required.

If you want more information about configuring timers and the different modes then check these tutorials out.
Ken Shirriff's secrets of Arduino PWMs: http://www.righto.com/2009/07/secrets-of-arduino-pwm.html
RobotFreaks's Arduino 101: Timers and interrupts: http://letsmakerobots.com/node/28278

- Ladvien
- cevinius
- bluesthue
- Dan M
- mark8sutton@gma...
- Hackgar
- K120189
- mtriplett
- ncoonrod14
- bdk6
- Maxhirez
- DannyJeffrey
- Duane Degn
- lukeyes
- Yahmez
- Cromz
- Kallie
- drewtoby

Navigation

- LMR on Google+
- LMR on Facebook
- LMR on Flicker
- LMR on Twitter
- LMR Scrapbook
- User list
- Unread posts
- RSS feeds
- Spam Control

Comment viewing options

Threaded list - expanded ▼ | Date - newest first ▼ | 10 comments per page ▼ | Save settings

Select your preferred way to display the comments and click "Save settings" to activate your changes.

By 42Bots @ Wed, 2014-01-29 17:34

delay() and millis()

Login or register to post comments

I read that changing these settings for timer 0 will affect delay() and millis() functions. Would changing the other timers (1 or 3 for example) have the same effect? Any other potential issues (possibly libraries that depend on those, I assume)? Thanks for the informative post!

> By OddBot @ Thu, 2014-01-30 04:53
>
> 
>
> **Yes timer 0 is used for the**
>
> Login or register to post comments
>
> Yes timer 0 is used for the timing functions so yes, adjusting timer 0 will affect those functions. The simplest workaround is to allow for the changes in your code. For example if you set the prescaler so it runs 4x slower then you know that when the Arduino thinks 1mS has passed, in reality 4mS have passed.
>
> Changing other timers will have no affect on the timing functions but they will affect any libraries that uses those timers.

By unix_guru @ Wed, 2014-01-29 14:52

Thanks OB1. Great tutorial.

Login or register to post comments

I struggled through all of this a few months ago. This will definitely help newcomers!

Great work as always.

> By OddBot @ Wed, 2014-01-29 15:19
>
> 
>
> **It's more of a cut an paste**
>
> Login or register to post comments
>
> It's more of a cut an paste solution than a tutorial.
>
> It is based on this tutorial: http://playground.arduino.cc/Main/TimerPWMCheatsheet but I've filled in some blanks for using the ATmega1280 and 2560.

ALL LMR ARE BELONG TO US!
Let's make robots!