by Jack Buffington

Getting Up to Speed With PWM

If you have been around the hobby robotics world, you probably noticed that almost every robot that you see uses hobby servos for just about every sort of movement. Whether it is to drive a wheel on a mobile robot, change the angle of something, or position a grabber arm, it will likely use a hobby servo. Hobby servos are useful devices, but they are not the only way to get things done in robotics. Along with hobby servos come a list of drawbacks, such as slow speed, little variability in speed, and the need to constantly send pulses to them so that they maintain speed or position. Last month's column talked about using Pulse Width Modulation (PWM) for audio purposes. This month, we'll go over how to use PWM to control DC motors.

Before discussing PWM, let's look at another way to control the speed of a DC motor. This would be to vary the current that is passing through the motor. While this is a valid way of controlling the speed of a motor, it is fairly inefficient and not very robust. Take a look at Figure 1; the current going through the motor can be adjusted by varying the position of the potentiometer. Let's pretend that the motor is just a resistor with a value of 30 Ω. In reality, the effective resistance of a motor will be lowest when the motor is stalled and highest when it is running at its top speed with no load placed on it. If you are dealing with a low enough voltage, then you can get away with this strategy.

For example, if you are running your motor at 5 volts and you have the potentiometer adjusted to 20 Ω, you will be dissipating 0.2 watts through the potentiometer. As the voltage goes up, the wattage through the potentiometer goes up with the square of voltage increase. This means that, if you increase the voltage to 10 volts, the wattage dissipated through the potentiometer jumps to 0.8 watts. If you happen to be using an inexpensive potentiometer from RadioShack, then you are just 0.2 watts shy of its rated limit. The potentiometer that you are using will start to become warm with this much wattage. Limiting the current through a motor is a quick and dirty method to vary the speed of a motor. It will work for small motors at low voltages.

In a similar vein, you can vary the speed of a motor by varying the voltage that you drive it with. One way to do this would be to use a variable voltage regulator. This method also suffers from overheating issues if your current draw is sufficiently high. This can be a slightly better method of varying the motor's speed because — to some extent — the overheating issue can be dealt with by using heatsinks on the voltage regulators.

While both of the previously mentioned methods of varying a motor's speed work, they have big problems with overheating and, because of that, they are also inefficient. This is where PWM comes in. With PWM, you are rapidly giving the motor power and then shutting off the power over and

Figure 1. Controlling a motor's speed by using a potentiometer.

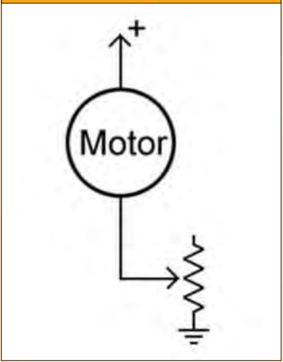


Figure 2. Controlling a motor's speed using PWM.

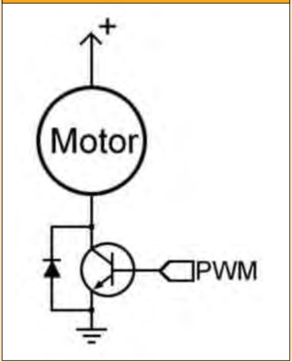
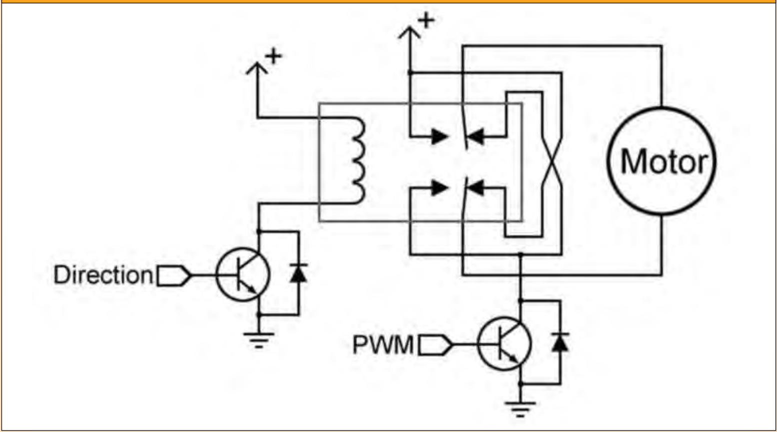


Figure 3. Reversible speed controller using a relay.



over again. The amount of time that you have the motor on verses off determines the average speed of the motor.

Figure 2 shows a simple circuit that you can use to vary the speed of a motor using a transistor to switch the motor on and off. Notice that the transistor has a diode across its emitter and collector. When you are running a motor, you are giving power to its coils in sequence. As you give power to a coil, a magnetic field appears around that coil. When you remove power from the coil, the magnetic field collapses and this sends a pulse of power in the opposite direction that it originally came into the coil. The diode is there to route that pulse of power back to the battery, where it won't damage anything. The relay also has a diode for the same reason.

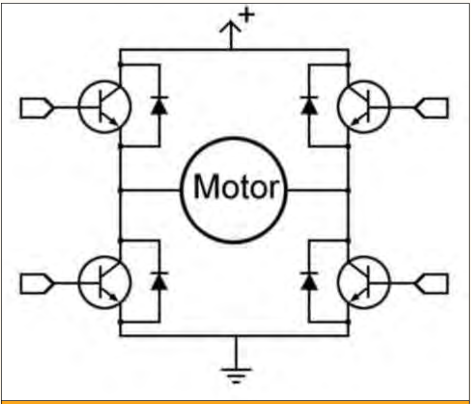


Figure 4. An H-bridge.

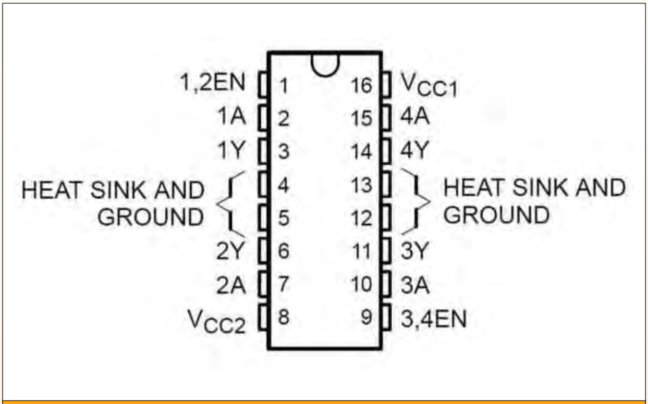


Figure 5. Pinout for the L293D.

One thing to pay attention to here is that you need a "fast" diode. The pulses of power coming out of the relay or motor are incredibly short in duration, but can be pretty destructive. Schottky diodes are usually the type used to prevent these pulses from damaging the transistor or FET in speed control circuits.

Varying the speed of a motor in one direction is useful for some applications, such as driving a fan or a water pump. For most robotic applications, you will want to be able to reverse the direction of the motor, as well as vary the speed. One way to accomplish that is to use a transistor and relay, as shown in Figure 3. In this circuit, the relay switches the motor's direction.

Switching direction with a relay is a fairly robust method of controlling a motor's speed and direction. It is easy to build and is inexpensive. The down side of using this method is that it is not solid-state, so the relay will have to be replaced from time to time if it switches often enough. Relays do not switch instantaneously. They can take anywhere from 0.5 to 30 ms to switch. If you need your motor to be able to switch direction rapidly, you will have to choose a relay that can switch quickly.

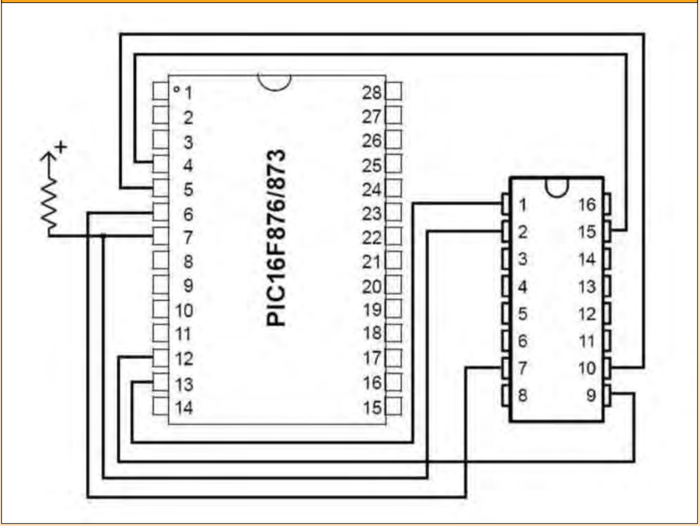
One further drawback to using relays is that they have different specifications for how much current they can carry as opposed to how much current they can switch. Often, you will find that a relay can only switch 10% of the current that it can handle continuously. This is due to arcing between the contacts, which can slowly erode them or destroy them in one big flash if you try to switch them while they are carrying a decent amount of current!

A solid-state circuit able to switch a motor on and off — as well as reverse its speed — is what is known as an H-bridge. An H-bridge is four transistors or FETs arranged as shown in Figure 4. By switching on the transistors or FETs in opposite corners of the H-bridge, you can cause the motor to run. If you turn off those transistors or FETs and turn on the other two, the motor will start running in the opposite direction.

One additional thing that you can do is turn on the two top or two bottom transistors or FETs at the same time. This causes the motor's leads to be shorted together and will cause the motor to resist movement without your circuit having to provide any power to the motor. You can really see this with a geared motor. Try turning the output shaft with your fingers and then shorting the two motor leads together and turning it again. You should see a noticeable difference.

H-bridges are great little circuits, but they are not something that is easily designed. While this column treats them as digital circuits, they are definitely NOT digital and sit squarely in the realm of analog electronics. It can be a very frustrating experience to try to design your own H-bridge unless you have a solid foundation in analog electronics. Luckily for the robotics hobbyist and professional engineers alike, there are pre-made H-bridges on single chips or on circuit boards that you can buy. Some examples are the L293,

Figure 6. Connecting an L293D to a PIC16F873 microcontroller.



TECH TIDBIT

The term Hi-Z is just a fancy way of saying that an I/O pin is for all practical purposes connected to neither the positive voltage nor ground. It is as if there is no connection at all.

L293D, L298, and LMD18200. This column will discuss the L293D. It is the least expensive of the three chips and can carry the least amount of current (600 mA). It is sufficient though for small gear motors.

The L293D is a dual H-bridge that has built-in diodes to catch the reverse voltage spikes that were mentioned previously. Figure 5 shows the pinout for the L293D.

The L293D has three input lines for each H-bridge. There are two inputs that directly correspond to the two outputs. If one of these pins is set high, the corresponding pin will be set to your motor drive voltage. If you input a low signal, then the output pin will be set to ground. The third input line is the enable line. If this line is high, then the outputs will be as described above. If the enable pin is driven low, then the output pins will go high. In Figure 6, the L293D's enable lines are connected to the PIC16F873's PWM output lines. By using these peripherals, you will be able to control the speed of motors connected to the L293D chip.

Figure 7. Code to drive a two-wheeled robot using the L293 and a PIC.

```
// PWM01.c
// This program is a demonstration program for a two-wheeled robot. It slowly
// accelerates the robot forward and then slows back down to a stop. It then
// turns and backs up.
// This program compiles with the CCS C compiler and is meant to be run on a
// PIC16F873 with a 20 Mhz oscillator.

#include <16F873.h>
#device adc=8
#use delay(clock=20000000)
#fuses NOWDT,HS, PUT, NOPROTECT, BROWNOUT, LVP, NOCPD, NOWRT, NODEBUG
const int8 forward =    0b00100100;
const int8 backward =   0b00011000;
const int8 right =      0b00101000;
const int8 left =       0b00010100;
#byte portA = 5

void main()
{
int i, PWMvalue;

    setup_adc_ports(NO_ANALOGS);
    setup_adc(ADC_OFF);
    setup_spi(FALSE);
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DIV_BY_16,63,1);
    setup_ccp1(CCP_PWM);
    setup_ccp2(CCP_PWM);

    portA = forward;
    // slowly ramp up over five seconds
    for(i = 0; i < 255; i++)
        {
        set_pwm1_duty(i);
        set_pwm2_duty(i);
        delay_ms(20);
        }
    // slow back down over five seconds
    for(i = 255; i > 0; i–)
        {
        set_pwm1_duty(i);
        set_pwm2_duty(i);
        delay_ms(20);
        }
    // turn
    portA = right;
    set_pwm1_duty(255);
    set_pwm2_duty(255);
    delay_ms(700);
    // go backwards
    portA = backward;
    delay_ms(2000);
    // stop
    set_pwm1_duty(0);
    set_pwm2_duty(0);
}
```

Figure 7 shows a simple program that can control two motors for a robot that drives using tank-style steering. The program will cause the robot to slowly speed up until it reaches top speed, then slow back down to a stop, turn at full speed, and then go full speed backward before stopping. This program is made to compile in the CCS C compiler for a PIC16F873 with a 20 MHz crystal.

The way that the microprocessor is connected to the L293D in Figure 6 allows the motor to go in either direction or coast. This type of PWM is called "sign magnitude." This is not the only way to do PWM, though. There are two other types. One type is called "locked antiphase." This type of PWM keeps the enable line high and rapidly switches the direction that the circuit is trying to drive the motor. If you do this fast enough, the amount of time that you are driving it clockwise versus counter clockwise determines both the speed and direction of the motor.

There is one more way to control a motor through PWM. This method alternates between driving the motor and shorting the leads of the motor together to act as a brake.

Figures 8 and 9 show how the L293D could be set up to use these other two types of PWM. To drive the circuit in Figure 8, just send a PWM signal out to the H-bridge on the line corresponding to the motor that you want to control. To drive the circuit in Figure 9, you would have to do your PWM in software, since you would need to be able to hold one input for the H-bridge low and send a PWM signal to the other input. This would drive the motor in one direction. If you wanted the motor to go the other direction, then you would want to hold the second pin low and send a PWM signal to the first pin.

You may be asking yourself right now why you would want to use one type of driving an H-bridge over another. The first method — where the motor is either driven or is coasting — is easy to set up a circuit for if you are driving a motor in only one direction. It also requires no extra circuitry when connecting to an H-bridge that has an enable line. The down side of driving a motor this way is that it is not that good at controlling the actual speed of a motor. This method primarily controls the amount of torque that the motor puts out. If there is no load on the motor, then it will achieve top speed with a fairly low PWM value.

The second method where the motor is rapidly driven in opposite directions turns out to be a pretty good way of controlling a motor's speed. The speed versus PWM value ends up being pretty linear. There are certain situations where this method of driving an H-bridge fits in nicely with the type of math that you are using on your microcontroller.

The third method also produces a fairly linear speed versus PWM, but requires that you manually change the direction of the motor. This method gives you double the number of distinct speeds that you can command your motor to go, but — in reality — you are unlikely to notice much of a difference.

One question that arises when working with PWM is how fast should you do your PWM. The simple answer is that it depends on your application. You can successfully do PWM at rates of 1,000 Hz or lower, but you may find that the constant whine caused by the PWM quickly becomes annoying. On the other hand, you don't want to do your PWM too fast because the transistors or FETs used in H-bridges are not digital devices. They do not transition from fully off to fully on instantly. There is some amount of time during switching where they are not quite off and not quite on. During this time, a larger than normal amount of voltage will be dropped through these devices, causing them to dissipate more wattage as heat.

For the L293D H-bridge, it takes an average of 600 nanoseconds to transition from fully on to fully off and then fully on again. If you did your PWM at a rate of 1,666,666 Hz, you would be forcing the H-bridge to be in transition 100% of the time and would cause it to burn up almost immediately even with a small load attached to it. Even at much lower rates, the transition times can still cause the part to heat up significantly.

A general rule of thumb for setting the PWM rate for a motor is to put the frequency just high enough that the sound coming out of the motor is not loud enough to be

Science and Engineering Tools for Schools - GEARS-IDS™

DESIGN, BUILD, TEST & LEARN with GEARS-IDS™ Invention & Design System

Invest in the best trainer for your Robotics program!

THE GEARS-IDS™ ADVANTAGE

- **Hands-on projects** with curriculum resources help students learn math, science and engineering principles.
- **High-torque all metal gear head motors.**
- **World-class pneumatics.**
- **Electronics** include speed controllers, switches, wire, and connectors.
- **Interfaces with Radio Controllers** or Parallax BASIC® Stamp programmable micro-controller.
- **Gears 3D solid models.**
- **Precision-machined Steel and Aluminum** drive components.
- **All components organized** in a storage container for classroom management.
- **Used in high schools** and colleges across the country.

Using GEARS-IDS™ every school can participate in engineering design competitions similar to those played at universities and national robotics events!

GEARS-IDS™ is more than robotics. Use it to illustrate physical science principles, pre-engineering or as the basis for a year-round technology education.

GEARS-IDS™ is rugged enough to customize your designs with additional manipulator, actuators or cameras.

Find out how your school can get the GEARS-IDS™ advantage. Contact Mark Newby at mnewby@gearseds.com.

*Note: Licensed **SolidWorks** users are eligible to receive a free copy of Prof. Marie Planchard's **SolidWorks** 2004 Tutorial with their first Gears Kit.*

Gears Educational Systems, LLC

Gears Educational Systems, LLC
105 Webster Street, Hanover, MA 02339
781.878.1512 • www.gearseds.com

Circle #25 on the Reader Service Card.

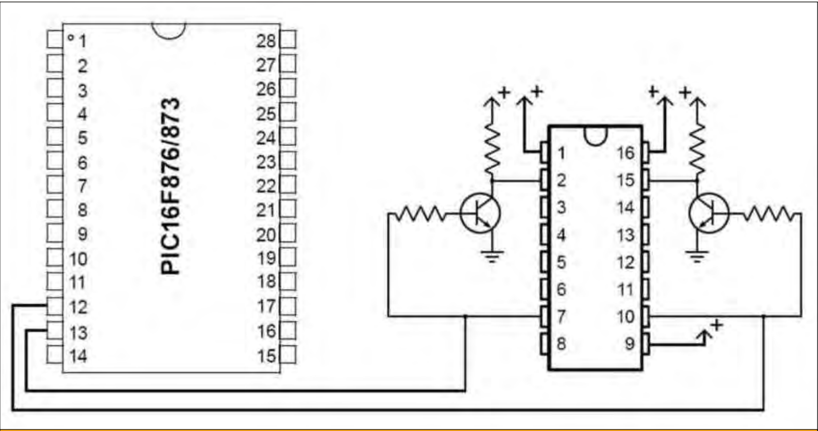


Figure 8. Controlling a motor using Locked Anti-Phase PWM.

TECH TIDBIT

www.ccsinfo.com
Sells the C compiler for PIC processors used in this column

www.microchip.com
Manufacturer of the PIC microcontroller

www.mouser.com
Source for electronic parts

www.solarbotics.com
Sells motors/gearbox combinations and other useful robot parts.

annoying. The volume of the sound will decrease as you raise the PWM frequency. There are other things to consider, but — for most hobby applications — this is enough.

This is not quite good enough for the locked anti-phase strategy, though. Since this strategy is constantly sending power through the motor, it is always consuming power. Motors are essentially coils. When you power a coil, it takes a certain amount of time for the magnetic field to build up and then collapse. If you set your PWM rate too low, then the field will have enough time to fully collapse as you change directions. This will cause your power consumption to skyrocket. By setting your PWM rate higher, you will see a drop in the amount of power consumed by the motor when it is being driven with a 50% duty cycle PWM wave. The ideal frequency where the least amount of current is drawn may be well above the range of frequencies that a person can hear if your motor has low inductance.

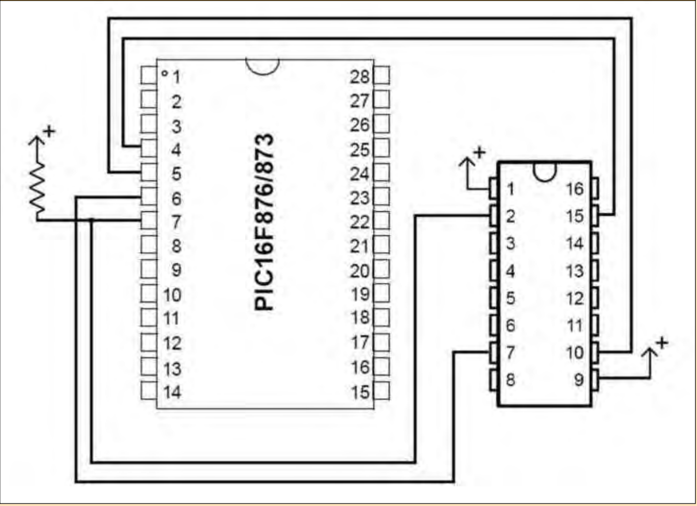
This article has talked a lot about how to drive motors, but maybe some time should be spent on where you can get your hands on motors to drive in the first place. Certainly, the easiest way to find motors to get started with is from a toy. Remote controlled toys tend to make good robot bases, since they already have space to put batteries and have a complete drivetrain ready to use.

If you are the more adventurous sort (good for you!), then you might prefer to buy your own motors and gearboxes so that you build your own robot parts the way you want them to be. **Solarbotics.com** sells some really nice motor/gearbox combinations for prices that are well within a hobbyist's budget.

If you really want to fully engineer things and get your robots to peak performance, take a look at **maxon motorusa.com** where you can buy pretty much any size DC motor that runs at whatever voltage you want and can be connected to many different gearboxes with other things such as rotary encoders or tachometers. Be prepared for sticker shock when you ask for prices, though. A motor/gearbox combination from Maxon will set you back anywhere from $80.00 to $200.00 and there is a $250.00 minimum for orders. Even still, Maxon makes some really excellent motors and is definitely worth a look if you want to take your robot to the next level.

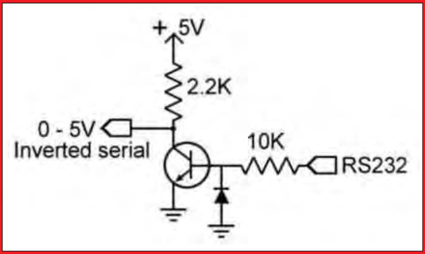
Hopefully, this month's column has opened your eyes to how you can get started using DC motors in your robots. Controlling DC motors using PWM is an easy thing to do with most microcontrollers and can open the doors to creating more exciting robots. **SV**

Figure 9. Controlling a motor so that it is either driven or is braking.



TECH TIDBIT

Would you like to be able to communicate with your PIC using your computer's serial port? Here is a handy little circuit that will take a RS232 signal and invert it while converting it to a 0-5 volt signal. If the microcontroller that you use doesn't need the inversion, switch the position of the transistor and 2.2K resistor.


+5V
2.2K
0 - 5V
Inverted serial
10K
RS232