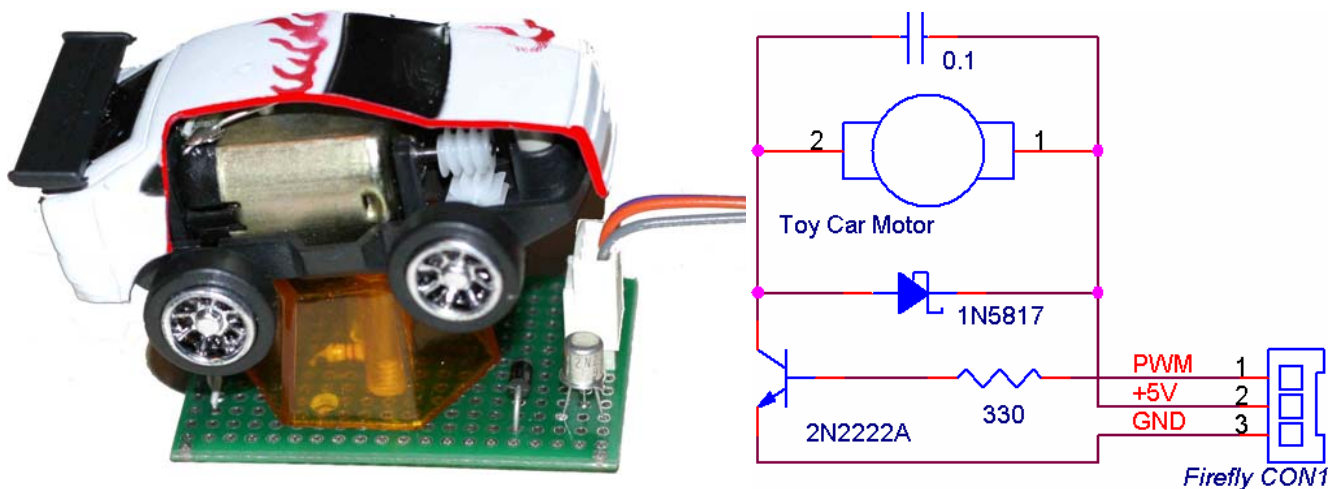


## Two Page Project: PWM motor control using a \$1 toy car

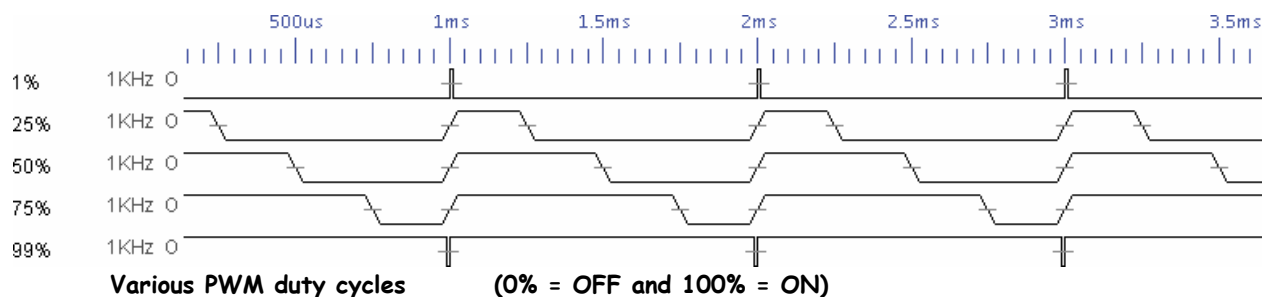


**WARNING** the transistor gets very hot; do not touch it while the motor is running.

This simple project can be assembled in an evening. It demonstrated using PWM (Pulse Width Modulation) to vary the speed on a simple 3V motor found in a toy car. The car was cut open to show off the simple worm & pinion 12:1 gear motor drive; typically 3V toy motors run at about 10,000 RPM.

The toy car was bought at a dollar store for *are you ready* \$1. Assembled on a small prototyping PCB, the layout is not critical except to make the small motor less electrically noisy; a 0.1uF capacitor was placed across the motor terminals as close to the motor as possible (*located on top of the motor, barely visible in the above photo*). A 1N5817 Schottky diode acts as an inductive spike suppressor and an NPN 2N2222A transistor acts as a high current ~300ma switch.

A PWM speed of 1 KHz was chosen (oscillator @ 1MHz, PR2 = 250) as it was close enough for the A/D conversion of VR1 (0-255). Keep in mind any value entered into CCPR1L greater than the PR2 value will set the duty cycle to 100%; so A/D values > 250 will be at 100% duty cycle. Only when the PWM signal is high is power applied to the motor.



The circuit was designed to connect directly to [blueroomelectronics](http://blueroomelectronics.com) Firefly's PWM / Servo connector CON4 and a simple demo program for the 16F88 can be seen [here](#). Turning VR1 will vary motor speed. Digital PWM is commonly used for motor control as it is very efficient compared to analog methods.

## Two Page Project: PWM motor control using a \$1 toy car

### Program listing ToyCar.asm



```
;*** ToyCar.asm      Firefly DIP switch UDDUDD
;*** uses VR1 (RA1) to set PWM (RB3) to control a toy motors speed
    list      p=16F88
    include <p16F88.inc>
    __CONFIG __CONFIG1, 0x377C & _CCP1_RB3

    org      0x000
    movlw    b'00000100'
    movwf    T2CON          ;B0 TMR2 on, no prescaler
    movlw    b'00001100'
    movwf    CCP1CON        ;B0 CCP1 PWM mode
    movlw    b'11001001'
    movwf    ADCON0          ;B0 A/D on VR1
    bsf      STATUS,RP0      ;B1 change to bank 1
    movlw    b'01000010'
    movwf    OSCCON          ;B1 set the oscillator to 1 MHz
    movlw    b'00000010'
    movwf    ANSEL           ;B1 PORTA,1 (VR1) analog input
    movlw    b'11110111'
    movwf    TRISB           ;B1 RB3 Output (PWM)
    movlw    .250            ;1KHz (osc /4 /250)
    movwf    PR2             ;B1 set PWM frequency to 1 KHz
;*** read VR1 and moves result to PWM duty register
main    bcf      STATUS,RP0    ;B0 change to bank 0
        bsf      ADCON0,G0      ;B0 begin A/D conversion (VR1)
addone   btfss   ADCON0,G0      ;B0 A/D done yet?
        goto    addone          ; loop till A/D done
        movfw   ADRESH          ;B0 W = A/D high byte
        movwf   CCPR1L          ;B0 set duty cycle to A/D result
        goto    main           ; * any A/D result > 250 will = 100% PWM *
End
```

### Further Thoughts

- What effect does the PWM frequency have on the motor?
  - try setting OSCCON to a different frequency and see what happens
- Notice the motor may not start from a low duty cycle
  - *but* if you start it at a higher duty cycle you can slow it down to a crawl
- Why does the transistor run hot?
  - try substituting an N-Channel Power FET (an IRF510 will work like a charm)
  - *the IRF510 has a very low on value 0.54Ω, it probably won't even get warm*
  - *you also don't need the 330Ω resistor if you use a Power FET*
- Use the Inchworm in debugger mode and put CCPR1L in a watch window
  - you can pause the debugger and enter new CCPR1L values directly
  - the PWM hardware will continue to run in pause mode
- Want to control the motors direction? *you'll need an H-bridge*
  - research “[H-bridge](#)” or “[L293D](#)” on Google and you'll see plenty of examples
  - more sophisticated motor controllers will generally use an H-Bridge
  - the L293D and SN754410 have the clamp diodes built in see our [Mongoose](#) kit