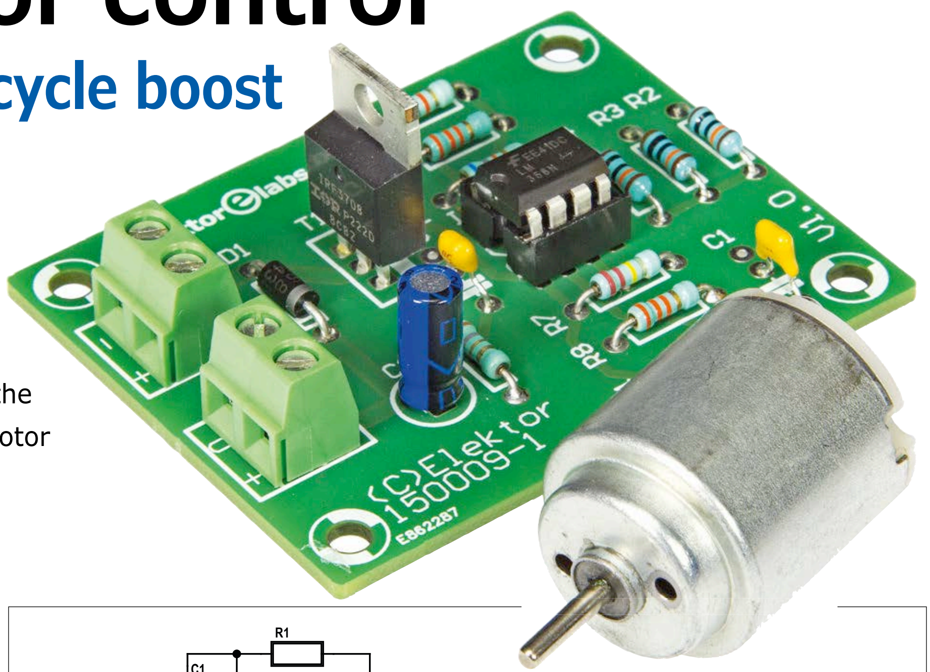
LABS PROJECT

PWM motor control

With added duty cycle boost

By **Rolf Blijleven (Netherlands)**

This circuit is designed to allow a small DC motor to run at low rotational speeds. The motor is driven using a PWM signal, in which the duty cycle increases whenever the motor is forced to work harder.



It sometimes happens that you need to run a DC motor at slow speeds. The standard solution is to drive the motor with a squarewave signal having a low duty cycle. But if the motor is overloaded mechanically, the squarewave may deliver inadequate power, causing the motor to stall. A solution to this problem lies in a PWM (pulse width modulation) circuit with a variable duty cycle. Circuits offering a manually adjustable duty cycle are of course plentiful, but I wanted a duty cycle that varied automatically. That should not be difficult, I thought. And it should also work without using a microcontroller! You take a PWM generator, use this to control the motor, measure the current through the motor and feed this back to the PWM generator as a control signal. With a couple of opamps, a MOSFET for the motor control, some resistors and capacitors plus a simple, single 5 V USB power supply it ought to succeed. I started out with the classic circuit in **Figure 1**, taken from the datasheet for the LM358 [1]. (The component numbers have been modified to accord with those in the definitive schematic in **Figure 2**.)

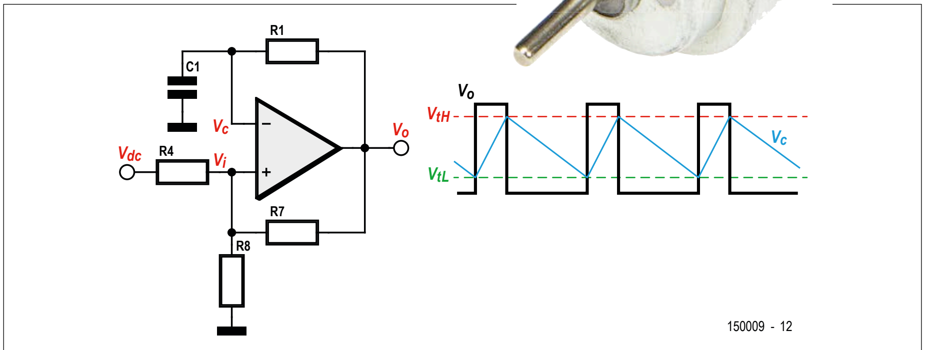


Figure 1. Functional diagram of a pulse generator using a single opamp.

PROJECT-INFORMATION

motor | PWM | Duty cycle

→ entry level
intermediate level
expert level

30 minutes approx.

Soldering iron
Old motor from a CD/DVD drive

€15 / £12.50 / $16 approx.

Theory

The basic principle will be familiar to many readers: V_{DC} is a DC voltage (DC for *direct current*, not to be confused with DC for *Duty Cycle* [2]). At switch-on the positive opamp input V_i is always greater than zero, whereas the capacitor voltage V_c at the negative input is still zero. Opamp output V_o goes into saturation at its maximum output voltage (V_{omax}). The output voltage charges capacitor C1 via R1, until V_c becomes greater than V_i. We'll call that state V_{tH}. When this point is reached the output flips back to zero, whereupon the capacitor is discharged via R1 to the point V_{tL} where V_c drops below V_i. Now V_o goes high again and the cycle repeats. The circuit is a combination of a saw-tooth generator and a Schmitt Trigger.

The great thing now is that you can drop these components into a couple of formulae, with which you can calculate exactly which values you need for a particular cycle. We can also approach this a bit more intuitively. Using any 741-type opamp that you choose, V_{omax} will

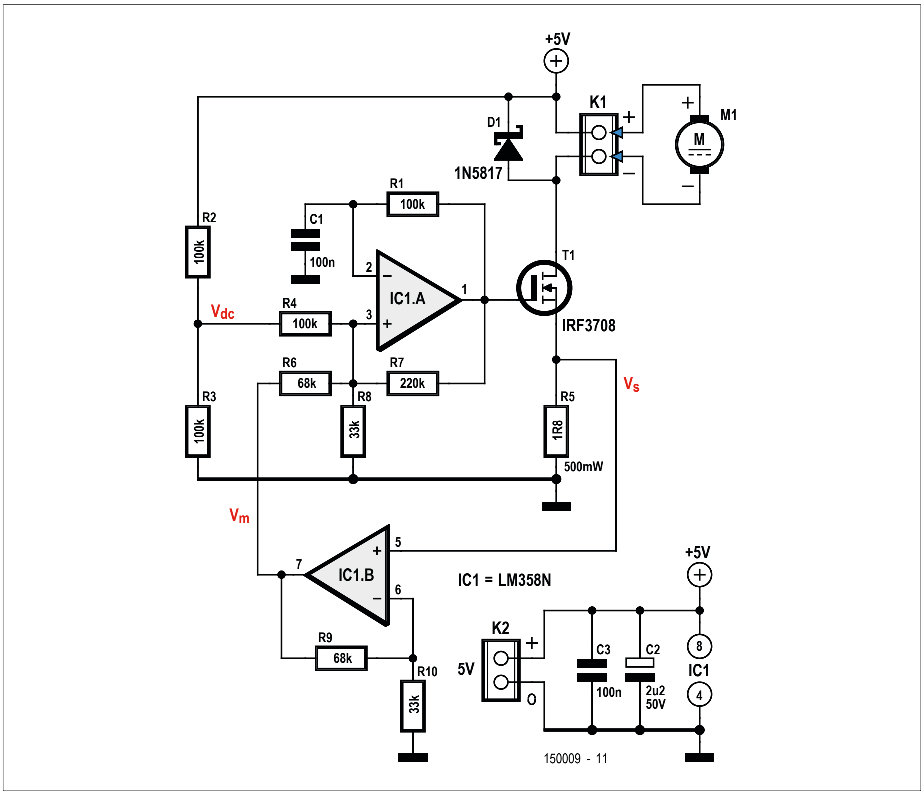


Figure 2. Generator IC1A drives the motor via MOSFET T1, whilst IC1B in conjunction with current sensor R5 assure a degree of positive feedback.

go up to 1.5 V below the supply voltage. If you wish to go all the way up to the supply rail, then take an opamp with an FET output stage, such as the CA3130 or CA3140.

The higher V_{DC} is, the higher V_i will be — and the higher the duty cycle. If V_i is always greater than V_c, then the duty cycle is 100% (in other words the full DC voltage). V_i is the determining variable in the whole affair.

To make this more comprehensible, it may be better to think not in terms of resistance but of conductance. The higher the value of R4 (the lower the conductance of V_{DC} through R4 to V_i), the less V_{DC} contributes to V_i, thus the lower the duty cycle.

The lower R8 is, the more the conductance of V_i to ground (GND) and consequently the lower the duty cycle.

The lower R7 is, the greater the influence of V_{omax} on V_{tH} and hence the greater the difference between V_{tH} and V_{tL}. But what matters in duty cycles is the *sum* of V_{tH} and V_{tL} and this does not change much with R7, meaning that R7 has less influence on the duty cycle than R4 and R8.

At this point you would choose R4, R7 and R8 to have a duty cycle that is always high enough to run the motor. But I wanted a duty cycle that was as low as possible and would automatically increase a little when the engine had to work harder.

COMPONENT LIST

Resistors

(5%/0.25 W, unless indicated otherwise)
R1,R2,R3,R4 = 100 kΩ
R5 = 1.8 Ω 0.5 W
R6,R9 = 68 kΩ
R7 = 220 kΩ
R8,R10 = 33 kΩ

Capacitors

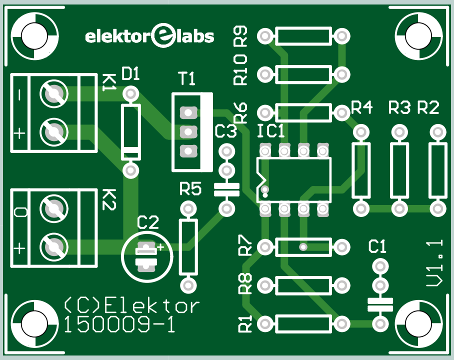
C1,C3 = 100 nF
C2 = 2.2µF/16 V, radial, 2 mm pitch

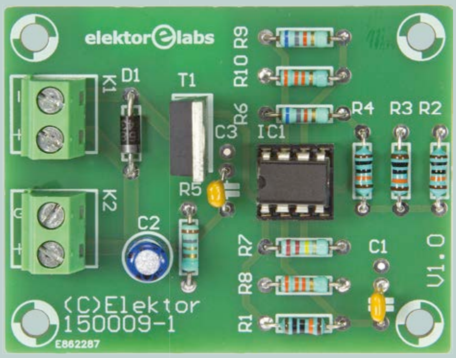
Semiconductors

D1 = 1N5817
T1 = IRF3708PBF
IC1 = LM358

Miscellaneous

K1,K2 = 2-way screw terminal block, 0.2" pitch
Small DC motor and gearbox (recycled from an old PC CD drive)
PCB # 150009-1 from Elektor Store





Circuitry

You can see the complete schematic in Figure 2 and let me say at the outset that the circuit does not allow itself to be domesticated willingly. More about that more later; the principles of the circuit come first.

R2 and R3 fix V_{DC} at half the supply voltage. The PWM signal at the output of IC1A goes to a MOSFET that switches the motor on and off. Using R5 (1.8 Ω) we measure the current through the motor (nominally around 450 mA). This yields a voltage V_s of 0.8 V at normal loading, rising to 1.2 V when the motor must work harder. In that instance around 700 mA flows through R5. But because the voltage and current are pulsed, the power ($I^2R = 0.7^2 \times 1.8 = 0.9$ W) in the resistor is reduced by the duty cycle to around 20%. A 0.5-W resistor for R5 will not take any notice of this.

V_s is amplified a little using IC1B and the result goes via R6 to V_i. V_i rises slightly in consequence and as the motor becomes still more heavily loaded, it draws more current as appropriate.

This all sounds very straightforward but selecting the correct resistor values is not so easy. There are still some snakes in the grass. To begin, the frequency of the PWM signal decreases as the duty cycle rises. You can see this in the formula. The charging time *t*1 and the discharging time *t*2 of C1 are decisive:

$$duty\,cycle = \frac{t1}{t1+t2} \times 100\%$$

The greater the charging period, the higher the duty cycle, but the frequency is 1/(*t*1+*t*2) and this then becomes lower. That gives us fewer pulses per second, making the motor speed fall (at a time when you actually want to keep it constant). If you increase the influence of V_m by using a lower value for R6, then you increase the duty cycle but lower the speed even further.

A shorter time constant (R1 × C1) was not a solution because the DC motor made an ominous sound at PWM frequencies above 75 Hz and above 120 Hz the screeching stopped.
Furthermore, the voltage across R5 is a DC voltage with commutation ripple superimposed, caused by switching from one carbon brush to the other while the motor is running. At higher loading, the voltage across R5 is at a maximum and the ripple frequency also increases. This makes sense, because the motor then turns more slowly.
Finally, the motor current (amplified and converted to voltage V_m) was added to the *positive* input of the PWM opamp, achieving positive feedback. According to textbook theory this is a bad idea, in part because the impedance of the voltage source influences the summation. In fact that does not play a major role here. The output impedance of IC1B is very low and much lower than R6. The source impedance of voltage divider R2/R3 here is 50 kΩ, half of R4. I did also experiment with an emitter follower at the voltage divider to reduce the impedance, but that turned out to make hardly any difference to the operation.

If the circuit in Figure 1 using rather simple formulae in a spreadsheet was predictable, then the schematic in Figure 2 displayed some troublesome phenomena trapped in the mathematics. I decided to simply experiment — that is, after all, what breadboards were invented for.
And in this way I got it working. With the values given in the schematic and in the list of components I had a duty cycle of around 17% at 55 Hz, increasing to 21% at 45 Hz under the toughest mechanical stress. This time I did not need to do anything more.

Elektor Labs have designed a small PCB for the circuit, which can be seen in **Figure 3**. This will always be handy in case you want to experiment but you will still have to adjust some resistor values for each type of motor. Next time, if the speed must be kept constant, I think I would design something with a microcontroller.

(150009)

Web Links

[1] LM358 data sheet: www.ti.com/lit/ds/symlink/lm158-n.pdf

[2] http://en.wikipedia.org/wiki/Duty_cycle

FROM THE STORE

→ **150009-1**
PCB