# Live Fast, Die Old
## Simple Battery Lifetime Extender

**By Volker Schmidt**
(Germany)

Although tons of home appliances, kitchen-type utensils and consumer electronics run off batteries, manufacturers do not seem to care much about battery health. Electrically, most of the designs consist of no more than a battery pack, a switch and a motor. As a consequence, batteries get fully drained sometimes, degrading their lifetime. This .POST project shows how to extend battery life by means of an Atmel's ATtiny45V microcontroller, a power MOSFET and a few discrete components. The 'patient' in this case is an old electric cheese mill. After the 'surgery', the mill works way better, and a much longer battery lifetime can be expected. Needless to say that the circuit can be used in similar devices, such as a hand-held vacuum cleaner.

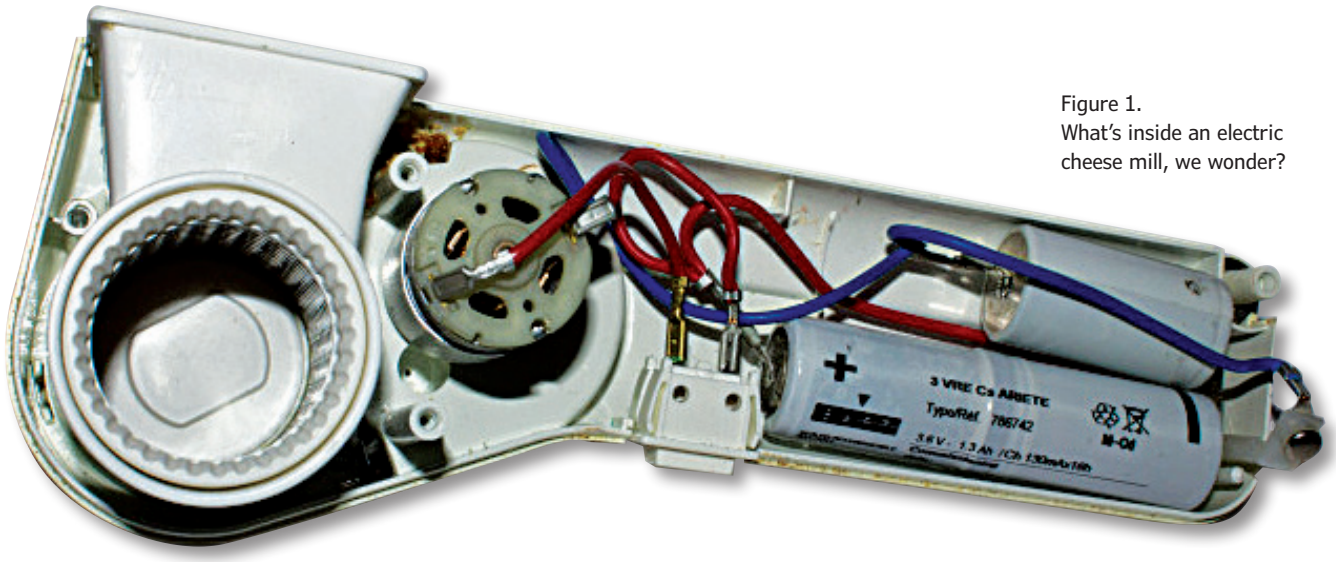


Figure 1.
What's inside an electric cheese mill, we wonder?

**Objective set: a pasta dish with shredded Parmesan**

After opening the case of the cheese mill (**Figure 1**) it was easy to locate the source of the problem. The battery pack consisting of three NiCd 1300 mAh cells was ruined. The cause was inherent to the design: the battery was connected directly to the motor, via a switch (see **Figure 2**), leaving it at risk of being totally discharged, something that should be avoided at all costs with this kind of battery.

First of all, brand new batteries had to be purchased. The original battery pack was still available as a spare part from an Internet supplier, but today there are better alternatives. Indeed, the author used 2400 mAh Sub-C NiMH batteries that fit perfectly in the mill case of Figure 1, and can be found in some online distributors such as Pollin, Germany. However, for our prototype at Elektor we went for a Li-Ion cell that supplies a nominal voltage of 3.6 V ($V_{CC}$). In any case, replacing the batteries

would never have solved the deep discharging issue. A battery monitor was required, and an analog design would have been too complex with such a low voltage. Besides, the circuit board couldn't be too large if we wanted to fit it inside all kinds of (handheld) devices, so the perfect solution was clear: a microcontroller.

### Hardware

Atmel's ATtiny45V microcontroller unit (MCU) [1] was chosen to perform the control tasks, due to its 10-bit analog-to-digital converter (ADC), its reduced 8-pin package as well as a wide voltage range, starting at a minimum of 1.8 V. In practice though it needs a minimum of 2.1 V to carry out successful analog-to-digital conversions (using an internal reference of 1.1 V). In addition, the ATtiny45V keeps the overall power consumption pretty low, adding a current of about 0.3 mA—and it's available in both through-hole and SMD packages. It's important to note that the design can also be implemented using an ATtiny25V or an ATtiny85V.

The micro is powered directly by the Li-Ion battery at $V_{CC}$. It ticks at 4 MHz using its internal RC oscillator, obviating the need for an external quartz crystal.

Let's have a look at the schematic shown in **Figure 3**. MOSFET T1 is effectively inserted between the negative pole of the battery (GND) and the load (a motor in this case). It is controlled by MCU port line PB3 via R1. Voltage divider R2/R3 provides a $V_{CC}/4$ voltage level to an ADC input on PB4. This division is needed because the ADC works with an internal reference voltage of 1.1 V. The microcontroller employs this level for its monitoring tasks, as will be explained below. Capacitor C2 shunting the battery prevents any noise from disrupting the ATtiny's operation. The Reset input is pulled to $V_{CC}$ via R4 and D1.

### Software

The firmware is easily explained. Please refer to the source code found at [2]. Basically, in the Main function, with every loop iteration the voltage in PB4 gets measured and stored in a 10-value array. As explained in the hardware section, these values are proportional to $V_{CC}$ (battery voltage). A simple conversion makes easy to guesstimate the real voltage of the battery pack. Once the array is full, the program compares its values with the minimum voltage. If all the values are below the minimum, output PB3 remains Low causing the MOSFET to be held Off, opening the high-current circuit and preventing the battery from experiencing a case of deep discharging. If on the contrary, not all the values are below the minimum, PB3 will go High, switching the MOSFET On (and the whole circuit closed and working). The variable "`MinVolt`" in de code stores the minimum voltage in mV, and it should be set depending on the battery you're using.

The program was written in C. It is well commented and available for downloading from the project page on the Elektor.

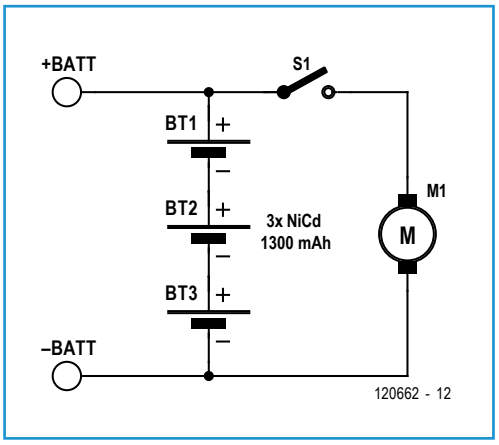



Figure 2.
Original circuit (not much of a circuit) of the mill.

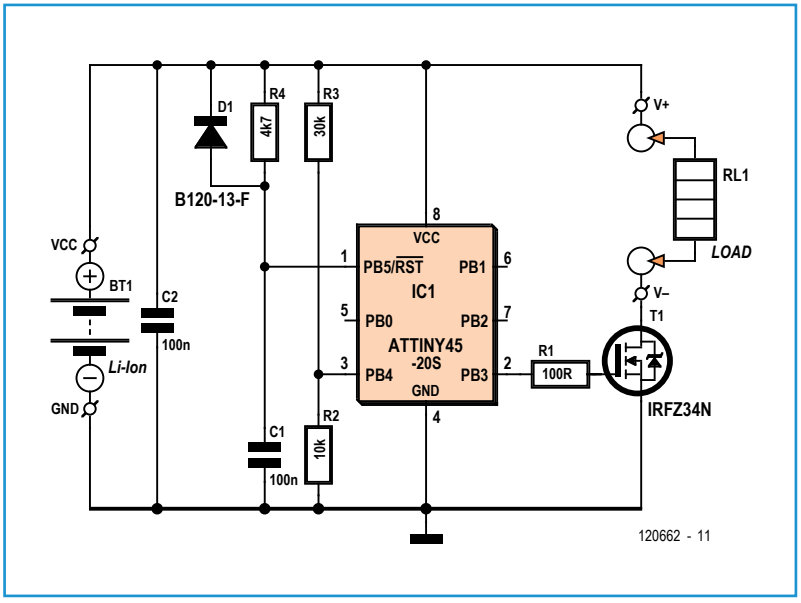
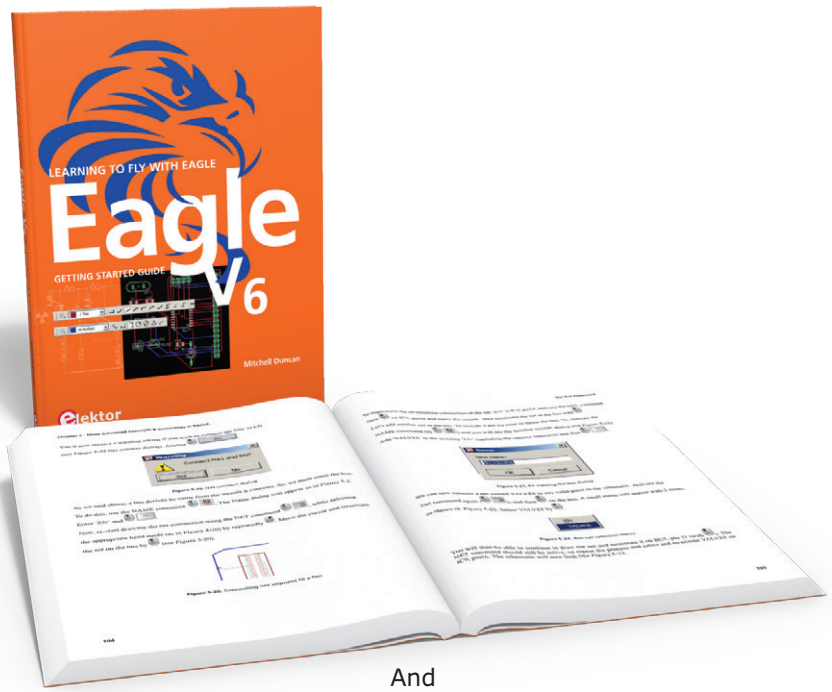



Figure 3.
Schematic of the Battery Lifetime Extender.

LABS website [2]. Also at that site is the project for Atmel Studio 5, and a brief explanation on how the ADC values get converted and compared. Regarding the notorious microcontroller fuse settings, be sure that CKDIV8 is set to True (i.e. clock set to 4 MHz).

**For the cheese, but not cheesy**

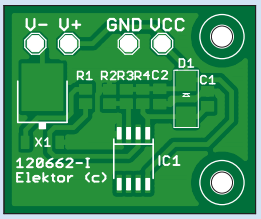
The author built his prototype on a small piece of breadboard, isolated it with heat shrink tubing, and installed it close to the upper Sub-C battery of the cheese mill (Figure 1 shows the original setup of the manufacturer). However, for convenience at Elektor we've designed a small PCB that may be downloaded for free at the Elektor.LABS' page associated with this article [2], including the PDF files, the Gerbers and the Eagle project files. If you are not familiar with Eagle, or would like to hone your skills with the program, we recommend reading and actively using Elektor's book called *Eagle V6 Getting Started Guide — Learning to fly with Eagle* [3].




And now, there's no excuse in bringing back to life all those long forgotten, battery-powered devices. Oh, and don't miss that well deserved pasta dish...

(120662)

## COMPONENT LIST




**Resistors**

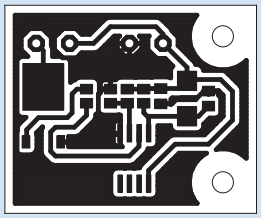
R1 = 100Ω, 0.25W, 5%
R2 = 10kΩ, 0.25W, 1%
R3 = 30kΩ, 0.25W, 1%
R4 = 4.7kΩ, 0.25W, 1%

**Capacitors**

C1,C2 = 100nF, 16V min., ceramic

**Semiconductors**

T1 = IRFZ34, MOSFET, N-channel, 60V
D1 = B120-13-F, Schottky diode, 1A, 20V
IC1 = ATtiny45V-10SU, 8-bit MCU, SOIC, programmed



## Internet Links

[1] www.atmel.com/devices/ATTINY45.aspx

[2] www.elektor-labs.com/120662

[3] www.elektor.com/products/books/electronics/eagle-v6-getting-started-guide.2453009.lynkx