

# DIY: Battery tester-discharger

*A circuit to discharge the batteries?!*


*A problem often occurs when rechargeable batteries are used in series/parallel: assess the state of charge of the individual element. Think of two rechargeable NiMH batteries, very charged in series with a third almost dead: when empty they can have about the same voltage, but when they discharge it can happen that the weakest battery goes into overdischarge before the total voltage drops below the fateful  $2,7\text{ V} = 3 \times 0,9\text{ V}$ . In the example we can found  $1,2 + 1,2 + 0,8 = 3,2\text{ V}$ .*

*In other words, 3 batteries in series must be not only equal (same brand, type, and capacity) but also in the same state of charge!*

*The problem is very real for Lilon-type batteries, which can ignite and even burst in extreme cases. There are so-called 'protected' Lilon batteries which have electronic circuitry inside the battery to prevent overcharge and overdischarge. [1]*

*The greatest risks occur when using series/parallel packs of Lilon batteries: either 'protected' batteries are used, or there are circuits to manage unprotected battery packs. [2]*

*Example:*

	<u>4A 3S Lilon Lithium Battery Protection Circuit Board Three Cell PCB</u>	2,11 €
--	--	--------

*NiMH batteries are more robust and less dangerous. The typical damage is the reduction of the charge capacity, which is almost always irreversible and which can lead to making the battery unusable.*

*The general NiMH batteries must not exceed the limits of overcharge and overdischarge and must be kept charged (even at 50%), refreshing the charge once a month.*

In the circuits used in e3DHW-PMS for the supply of electronic equipment, in cases of low power, 3 NiMH batteries or one Lilon battery are used. NiMH batteries, unlike Lilon batteries, can always be kept in charge (**slow charge**) and can, therefore, be used as a UPS..

To create a pack of 3 NiMH batteries correctly, which guarantees long battery life, the following strategy can be used:





1. use 3 identical batteries: same brand, type, capacity
2. discharge the batteries individually, up to the safety value (0.9 V)
3. put them in series and recharge the whole package up to 4.4 V
4. check that each battery has the required charge (about 1.47 V).

This tester allows you to assess the health of the individual battery and to perform step 2 reliably.

In other cases you may want to charge a battery in a 'blind' mode, for example for 1h to 1.2 C. This strategy is valid only starting from definitely discharged batteries: another use case for this tester.

# Features

- AA NiMH microcontroller battery tester.
- Automatic operation: insert the NiMH AA battery and check the LEDs.
- Safe discharge of charged batteries, a Beep signals the achievement of 0.9 V, avoiding the risk of overdischarge.
- Passive circuit, without power supply: uses the remaining battery charge.

Battery voltage	LED Test	Action
$V < 0,6^1$	off 	irrecoverable Battery to be eliminated
$0,6^1 < V < 0,9^2$	Red steady 	Overdischarge Possibly usable at reduced capacity
	Flashing red 	Discharge Wait for the discharge to end
$V > 0,9^2$ (max 5 V)	Verde + beep 	End of discharge (0.9 V) Remove the battery immediately

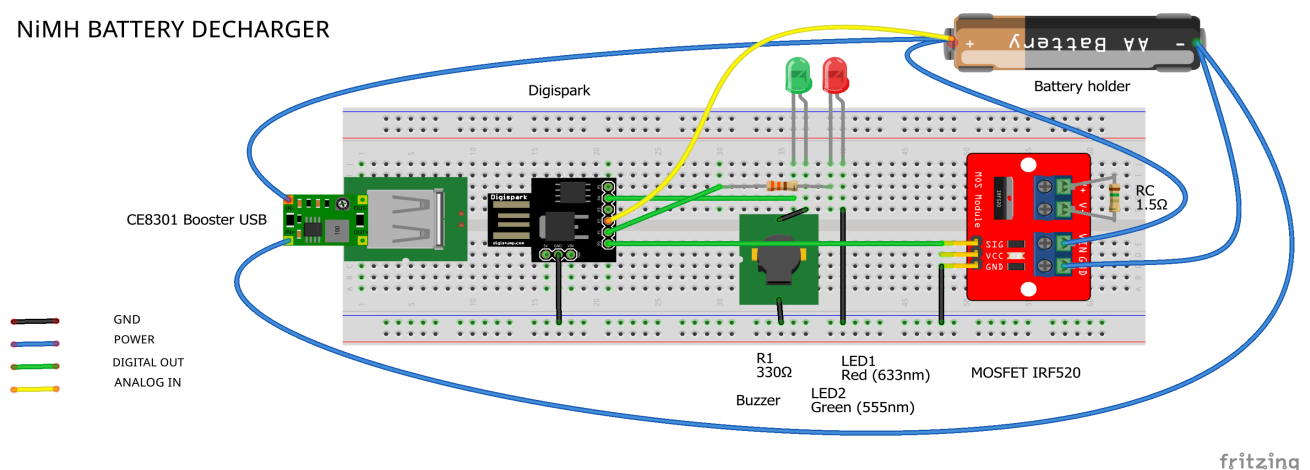
(1) indicative, it is the lower operating limit for the CE8301 Booster circuit

(2) depends on the internal  $V_{ref}$  of the ATtiny85. This value can be set by changing the constant ADC\_ADJ in the sketch (check with a digital voltmeter).

Note: the LEDs light up 5 seconds after the battery is inserted, due to the ATtiny85 start-up procedure.

## Circuit


### NIMH BATTERY DECHARGER

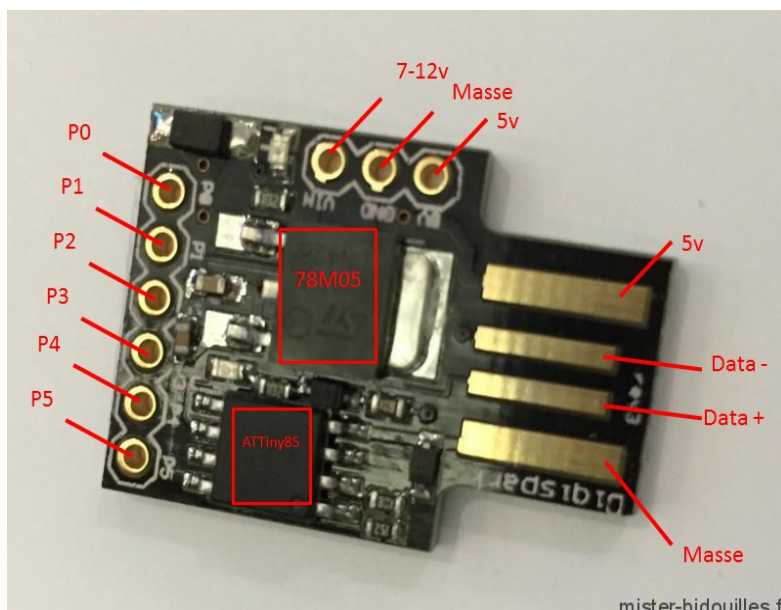
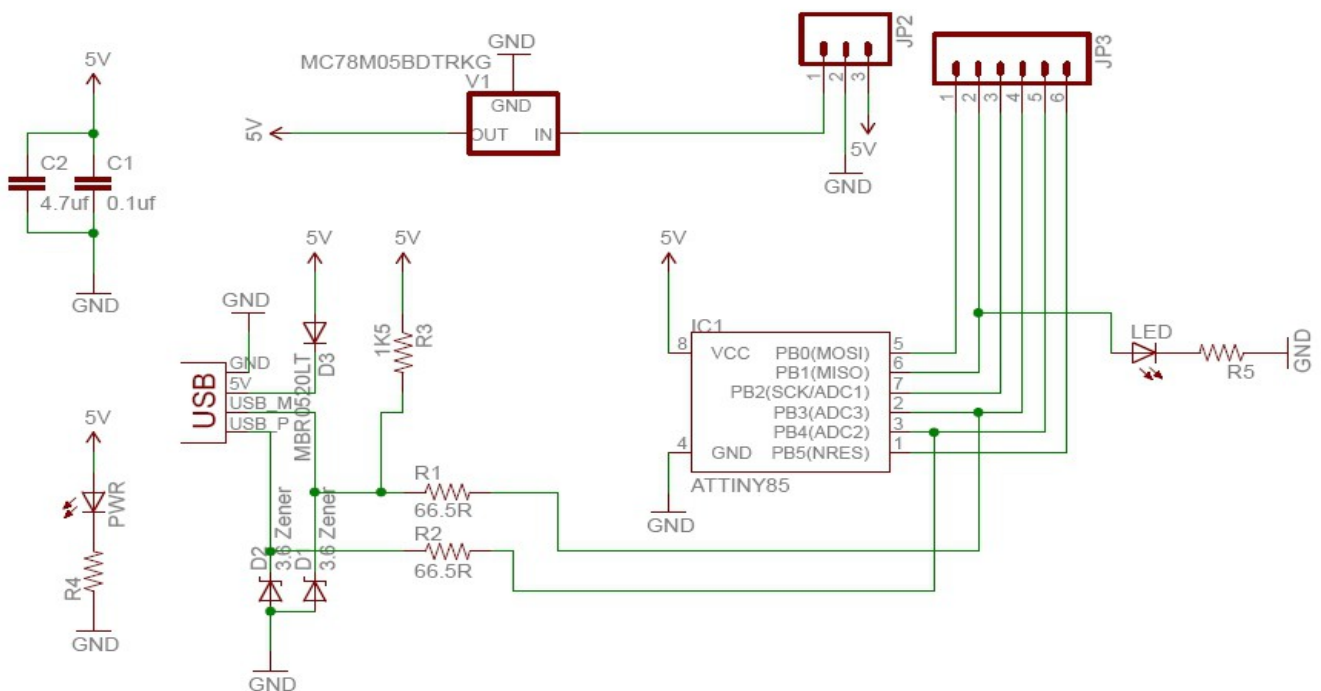


The circuit is very simple: 4 modules and 3 components. The use of cheap pre-assembled modules makes the use of ad hoc printed circuits superfluous and this greatly simplifies both the tests and the final assembly.

## Microcontroller

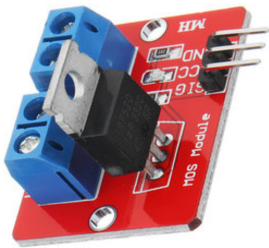
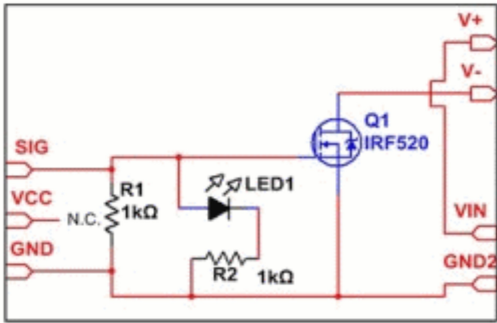
The main component is the ATtiny85 Microcontroller (AVR, 8 bit, 8K) contained in the Digispark board [3] [3]

	Digispark Micro development board for ATTINY85	1,21 €
---	---	--------



note: The I/O pins, indicated on the previous wiring diagram as JP3 [1..6] on the printed circuit, correspond to **P0..P5** and are thus used:

**P0: DIGITAL OUTPUT**, connected to the SIG input of a MOSFET card (IRF520), which controls the load for the battery under test (GND2 and VIN). When P0 is high RC is connected to the battery.

		<p><b>IRF520 MOS Driver Module</b></p> <p>1.26 €</p>
---	--	--

The load resistor can be easily changed (it is connected to the two terminals V + and V-) if necessary to speed up the discharge.

RC has no critical value. The discharge current (xC) must be between 0.1C and 2C (where C is the capacity of the battery in Ah) and the resistor must be of adequate power (2W - 4W). There should be no overheating problems if air can circulate on the resistor. Calculate that the circuit, with the various LEDs, consumes about 80 mA.

To be precise, in formulas:

$$RC \approx \frac{1.2}{xC - 0,08} \quad [\Omega]$$

$$P > 1,4 \cdot xC \quad [W]$$


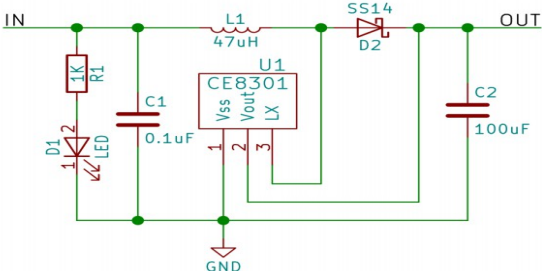
**P1: DIGITAL OUTPUT.** Check the red LED (LED1), with a current limiting resistor of 330 Ω (R1) in series. When P1 is high the LED is on.

**P2: ANALOGUE INPUT.** Connected to the battery positive to measure the voltage with the internal Attiny85 ADC (10 bit). Since the Microcontroller has a low and variable power supply, the ADC must use the internal reference voltage of 1.1 V.

**P4: OUTPUT DIGITALE.** Check the green LED (LED2), grounded via an electromagnetic Buzzer (or Beeper). By sending a square wave signal, the LED lights up and the Buzzer sounds

**P3, P5: NC**

## Alimentazione

		<p><b>PFM Control DC DC 0 9V 5V To USB Step-up</b></p> <p>1,30 €</p>
---	--	--

A booster with CE8301 (step-up) is used to increase the voltage from the battery (0.6 ... 1.4 V).

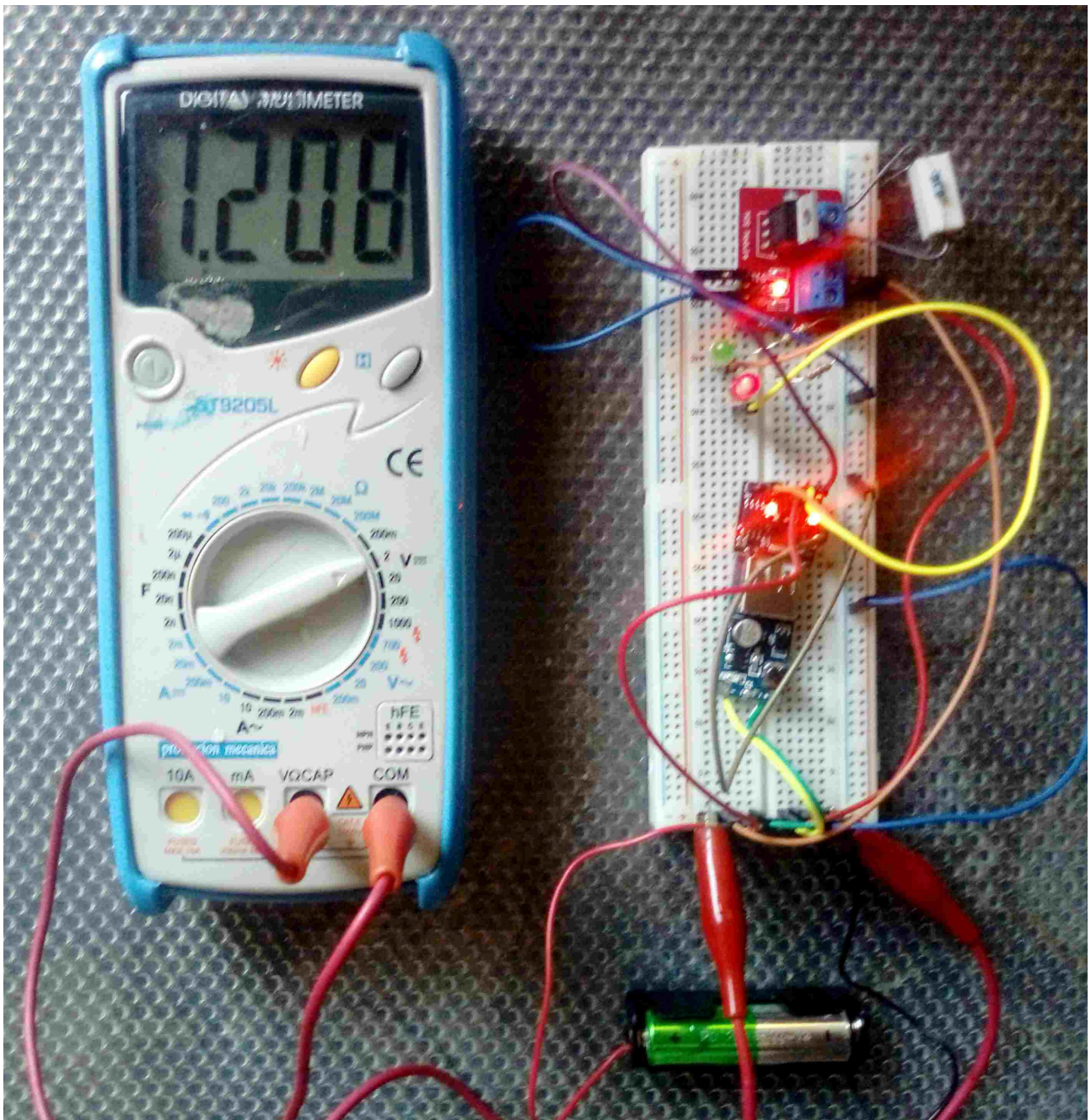
At very low input (0.6V-1V) this circuit under load does not supply 5V, but always carries out its step-up function providing about 3V, sufficient for ATtiny85 [5]

The Digispark card is directly inserted into the USB socket of the booster.

By simply extracting the card from the booster and using a USB cable connected to the PC, the Attiny85 microcontroller can be programmed in-circuit, a very convenient option for firmware updates.



# Prototype



The prototype, where measurements and modifications are made, I built it on a breadboard with the use of 20 cm cables. with connectors (3 types: M-M, M-F, F-F).



## Realization

The final realization was done in a small Teko plastic container: 11 x 7 x 5.5 cm.

Compared to the scheme, two bushings connected to the battery have been added:

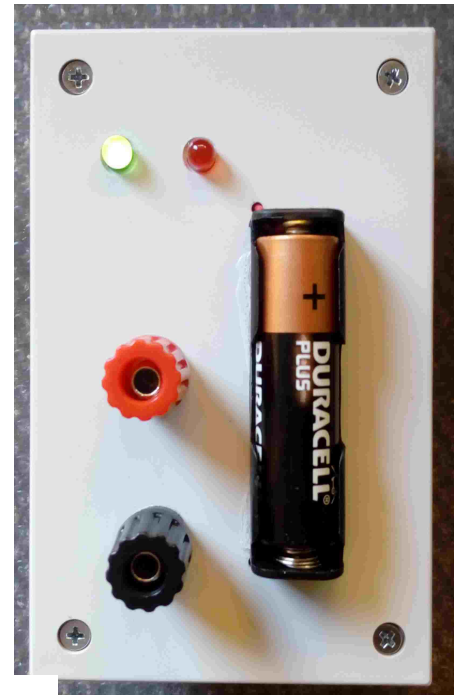
1. make it easy to connect a digital voltmeter to measure battery voltage
2. allow you to use other batteries, not type AA

First step is the drilling of the upper cover: to avoid heating the plastic I use a cordless screwdriver, with a chuck for the tips.

Holes: LED:  $\varnothing$  5 mm, bushings:  $\varnothing$  8 mm, screws:  $\varnothing$  3 mm.

Once the mechanical accessories (bushings and battery holder) are fitted, the LEDs and pre-assembled boards are fixed to the back of the panel, using hot glue.

The Digispark microcontroller is not fixed but only inserted into the USB socket of the booster.

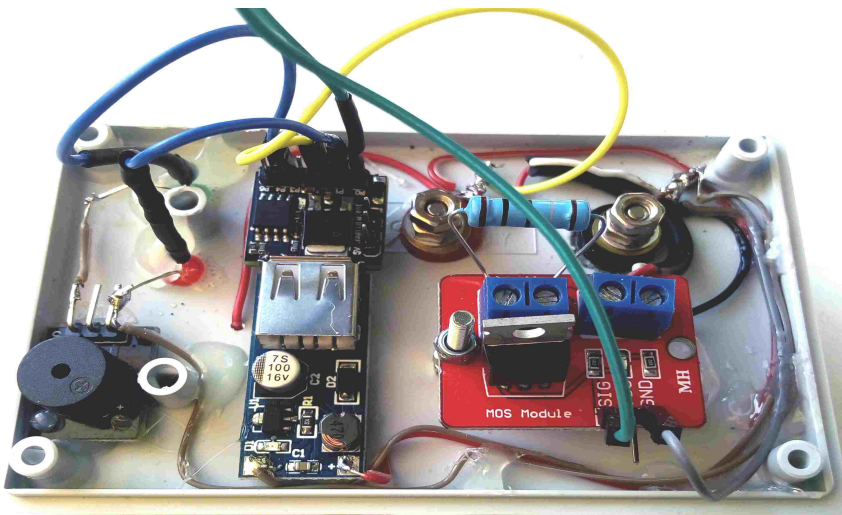


I started by wiring the battery holder and the bushings, using welds or cables with F connectors.

The wires and connectors are made stable with hot glue.

A single component is flying: R1, mounted on the wire and covered by a heat-shrink sheath.

The four I/O connections to the microcontroller are left a bit long to easily extract the Digispark card.



# Firmware

The circuit is very simple: all the logic is in the program that is run on ATtiny85. A great convenience is that this Microcontroller can be developed with Arduino IDE.

The necessary steps are simple and perfectly illustrated in [3]. Needless to repeat them here. With the Arduino IDE version 1.8.9 I had no problem.

The sketch to use is the `NiMH_discharger.ino` file

In principle, no modifications to the sketch are necessary.

It is however easy to redefine the logic because the firmware is structured as a simple and intuitive Moore automaton.

The 0.9 V value identifying the end of the discharge can be calibrated with good accuracy. Check with a digital voltmeter at which voltage your circuit switch and change the constant `ADC_ADJ` at the beginning of the sketch: a couple of tests and the calibration is done.

Good job!

# References

1. *Matt Mielke, Simple Undervoltage and Overcurrent Protection for Lithium-Ion and Lithium Polymer Batteries*,  
<https://www.digikey.com/eewiki/display/Motley/Simple+Undervoltage+and+Overcurrent+Protection+for+Lithium-Ion+and+Lithium+Polymer+Batteries>
2. *Texas Instruments*, Protect your battery, <http://www.ti.com/power-management/battery-management/protectors/overview.html>
3. *Adriarobot*, Digispark, <https://www.adriarobot.it/arduino/digispark/digispark.htm>
4. *Matt's Electronics* CE8301 Booster Converter <https://www.matts-electronics.com/wp-content/uploads/2018/06/CE8301.pdf>
5. *MPJA.com*, CE8301 datasheet, <https://www.mpja.com/download/ce830.pdf>
6. *Atmel*, Attiny85 datasheet, [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85\\_Datasheet.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf)