NiMH battery charger logger 2.0

Project

(*To make the links work, download this file and use a PDF-viewer on your PC*)

This is an experimental charger and data logger for NiMH batteries and packs, to get data and graphics for the charging process.
The ultimate NiMH charger can be so defined (https://www.powerstream.com/NiMH.htm):

1) *Soft start. If the temperature is above 40 degrees or below zero degrees start with a C/10 charge. If the discharged battery voltage is less than 1.0 Volts/cell start with a C/10 charge. If the discharged battery voltage is above 1.29 V/cell start with a C/10 charge.*
2) *Option: if the discharged battery voltage is above 1.0 Volts/cell, discharge the battery to 1.0 V/cell then proceed to fast charge.*
3) ***Fast charge** at 1 C until the temperature reaches 45° C, or the ΔT/Δt indicates full charge (1..2°C/min).*
4) *After terminating the fast charge, **slow charge** at C/10 for 4 hours to ensure a full charge.*
5) *If the voltage climbs to 1.78 V/cell without otherwise terminating, terminate.*
6) *If the time on fast charge exceeds 1.5 hours without otherwise terminating, terminate the fast charge.*
7) *If the battery never reaches a condition where the fast charge starts time out the slow charge after 15 hours.*
8) *Fuel gauge, communication to the device being powered, LED indicators all possible.*

Using 1.5 as charge/discharge factor:

Tickle charge (C/20...C/40) is safe for any time.

Constant tension, a special tickle charge, see https://github.com/msillano/e3DHW-PMS/blob/master/e3dhw-pms-intro_en.pdf .

Slow charge (C/10: 15h overnight charger) is safe, and time can be the unique termination method.

Fast charge (C/3.3...C: 5h...1.5h charger) requires at least a temperature termination test.

The NDV end charge (-ΔV <= -0.5 mV) is not very reliable according to Powerstream, so they don't use it.

Version 1.0 disconnection issue (slow and fast charge)

The NiMH charger version 1.0 as a problem, potentially very dangerous: if the USB link goes down, the power supply is without any control (contribution of Cliff Matthews).
The *RD6006* cut-off the output only for:

- $V_{out} > V_{OVP}$ (0...60V)
- $I_{out} > I_{OCP}$ (0...6A)
- $I_{out} < 10mA$ (in battery mode).

Not enough to protege a battery in the normal or abnormal end charge. Unattended charges are not allowed.

To have a more robust protection, this version 2.0 contemplates the optional use of a *UD18* (see https://github.com/msillano/UD18-protocol-and-node-red) because it can cut-off the

battery on its own when one of the following events occurs (the values *indicated* are user-configurable):

- Power < W_{min} (1...9W) for *XX* (01...99 min.)
- Time > *T-C* (countdown hh:mm, 0...24h)
- Current > *Over-C* (0...9A)
- Tension > *Over-V* (0...36V)
- Tension < *Low-V* (0...29V)

So this project uses the *RD6006-W* (not in battery mode) as Constant Current or Tension generator and as V/I/Ah/Temperature meter, while a *UD18* can be used only as autonomous HW protection in the *fast charge* mode.

The **node-red** flow must still verify *T, ΔT/Δt, ΔV* (point 3).

But now, if communications goes down, the battery is same enough protected by *UD18* via Time trigger (point 6) and *Over-V* (point 5).

Without UD18 a fast charge unattended is not safe.

Battery discharge

To discharge a battery (point 2), see the project ***NiMH discharger*** (https://github.com/msillano/e3DHW-PMS/blob/master/applications/Battery_discharger_en.pdf).

WIFI use

The WiFi is not standard, it requires heavy HW and SW modifications (see https://community.home-assistant.io/t/riden-rd6006-dc-power-supply-ha-support-wifi/163849). Not used.

Version 2.0 functions

Base schematics and definitions

I use here a single battery, but you can extend it to battery packs.

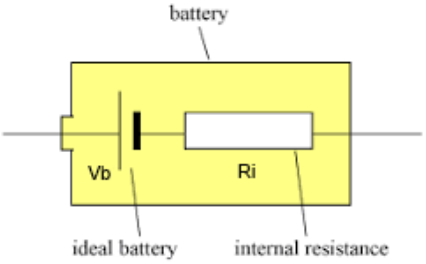
$\mathbf{V_{RD}}$ Tension from RD6006, as measured via USB

$\mathbf{V_{batt}}$ Battery tension under charge, not directly measurable (see later)

$\mathbf{V_{I0}}$ Battery tension *@I=0*, to get the internal tension $\mathbf{V_b}$

V_b measurement

A simple equivalent battery circuit contemplates an internal resistance, $\mathbf{R_i}$. So any external battery voltage measurement is function of the charge current $\mathbf{I_c}$



Really R_i can change by many factors: charge, temperature etc.

The easiest way to get $\mathbf{V_b}$ is to make a measurement of V_{RD} with $I_c = 0$ ($\mathbf{V_{I0}}$). This method is used in version 2.0.

In version 1.0 instead 2 measurements are used, with different I_c, since $I_c = 0$ is not allowed by RD6006 in battery mode.

On my RD6006, also with I_{SET}=0.000, generates a residual current of 5 mA.

The measure of V_{RD}, also @I=0 ($\boldsymbol{V_{i0}}$), is not the measure of V_b, because we must calculate now the UD18 voltage drop. This value is variable with I_C (0.310..0.75 V) but at *Ic* = 5 mA it is stable (0.310..0.330 V) so we can use:

$$V_b = V_{i0} - UD18_{drop}$$

If the UD18 is not used, simply set *UD18DROP* to *0* in `Config` node.

R_i internal resistance

R_i is re-calculated in every measure cycle using V_{i0} and the last value of V_{batt} measured with current i_c

$$R_i = \frac{V_{batt}(i_c) - V_{i0}}{i_c}$$

deltaT temperature gradient

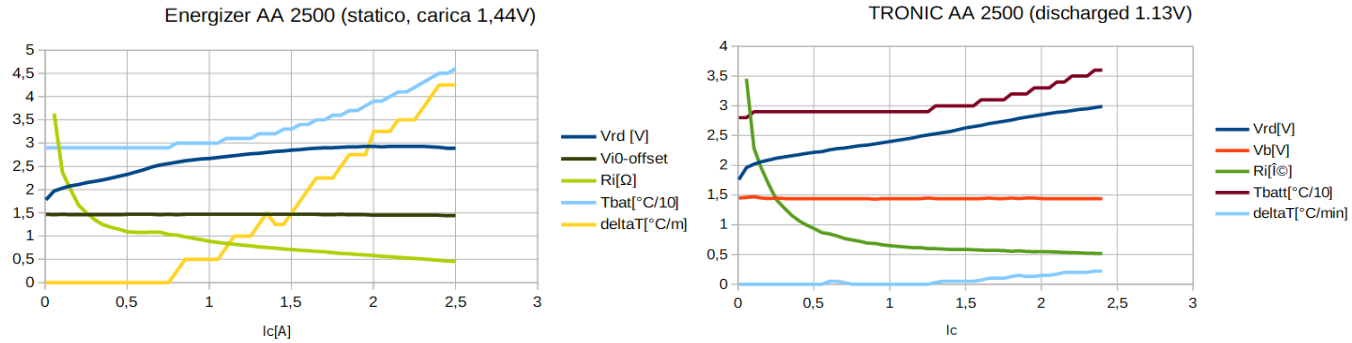
A circular buffer stores last 6+6+2 values of Tbatt: I use a 2 minute interval (12 values) plus 2 extra values to get smooth values (Tb_{13} is last value, Tb_0 is the oldest):

$$deltaT = \frac{Tb_{13} + Tb_{12} - Tb_1 - Tb_0}{4}$$

deltaV tension gradient

Same as *deltaT* but the first 30 values are skipped. Especially for testing purposes, to esclude it as end condition set a large negative value (e.g. -2) for *dVlimit* in `Config` node.

STEST mode: static test



This **charger-logger ver. 2** can work in various modes, user defined in `Config` node.

In the *static test mode* (STEST) the program can produce graphs as show. They are not a charge graphic, because every 10 s the I_C is augmented by 50mA. This allows to appreciate V_{RD}, V_b, R_i, *Tbatt*, *deltaT* as function of I_c only.

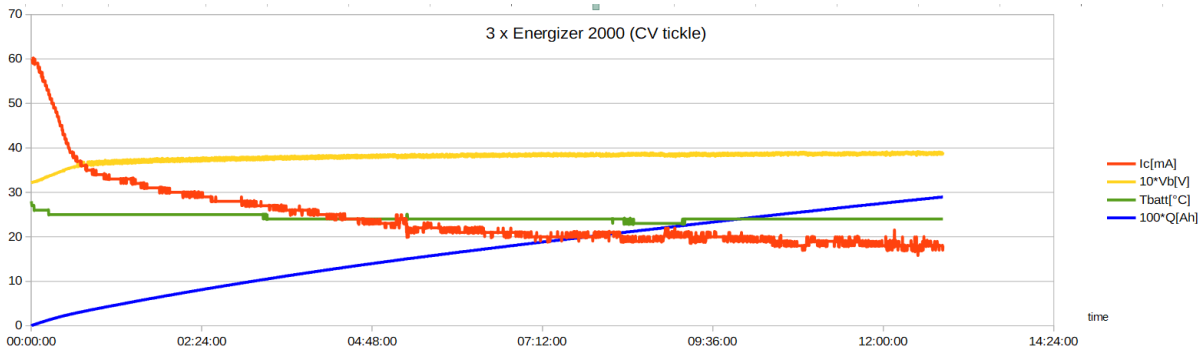
Looking at graphs, we can see that V_b is almost stable, the process is too fast to recharge the battery.

In the charged battery V_{RD} start quasi linear, then (@0,6 A ≈ C/4) change the slope. Also R_i change slope (@0,6 A).

An interpretation is that the termination of the chemical process of 0_2 recombination starts the temperature variations. ***deltaT*** shows that well.

In the discharged battery, the thermal variations are less strong.

CV mode: tickle charge at constant tension

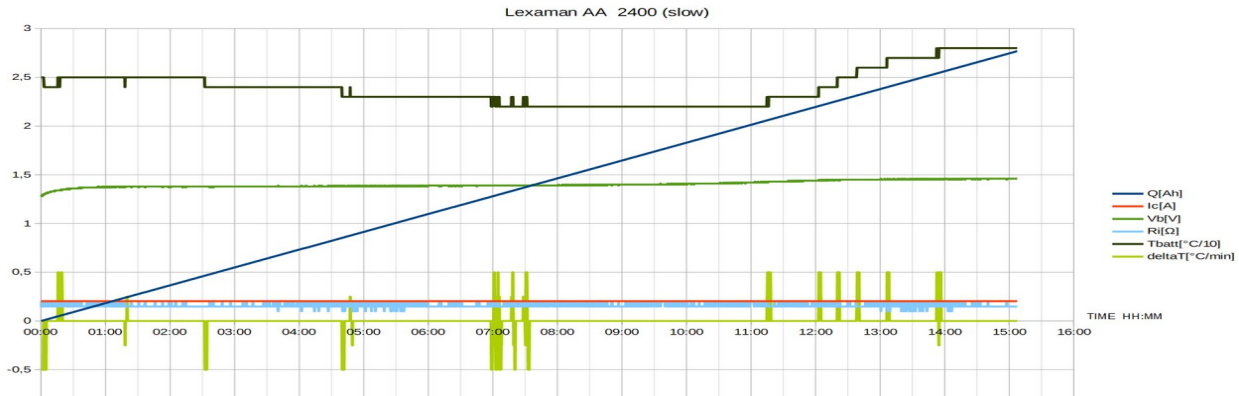


The test device is an Ikea Blåvik lamp (see https://github.com/msillano/e3DHW-PMS/blob/master/applications/ikea_blavik_usb_en.pdf) The tension is 5V, the start batteries charge is 3.3 V (≈1.06 each). After 8h the current drop down to 20mA and the temperature is low.

Note: In this test the *UD18DROP* in `Config` node is set to 0.910 V to compensate the diode drop.

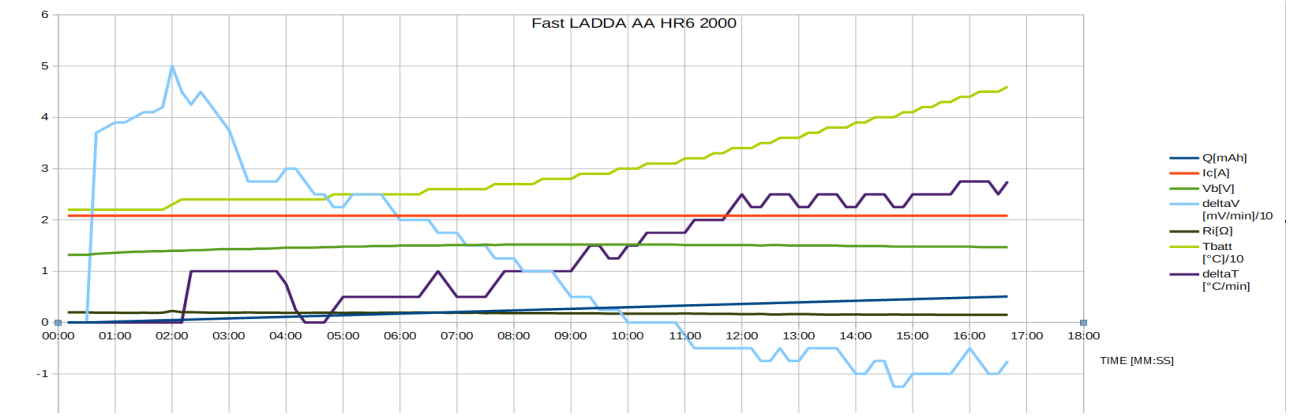
The graph confirms the measurements done in the lamp project.

SLOW charge



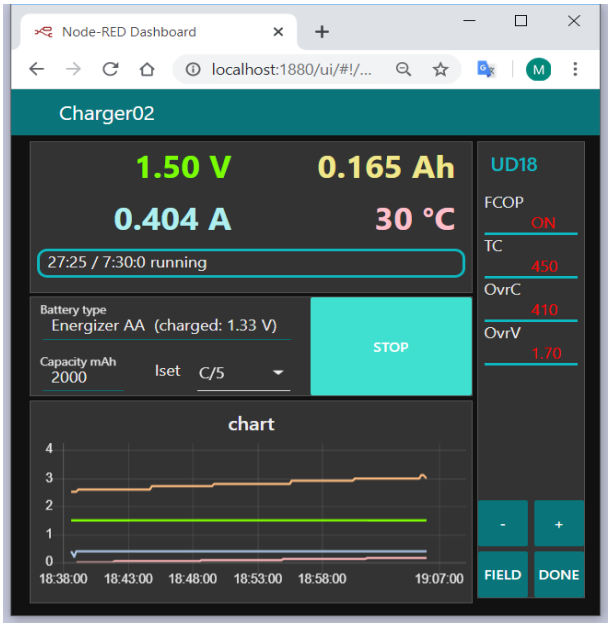
This charge (15 h at 200 mA) shows that the battery is charged after 11h: V_b rise a little, T_{batt} goes from 22°C to 28°C, but very slowly: *deltaT* max is only 0.5 °C/min. The 0_2 recombination process works well, the slow charge is safe.

FAST charge



In this charge (17 min at 2000 mA) we can see how *deltaT* grows until 2.75 °C/min (see point 3) while *deltaV* is low as -1.25 mV/min. The battery final temperature is 45°C (point 3).

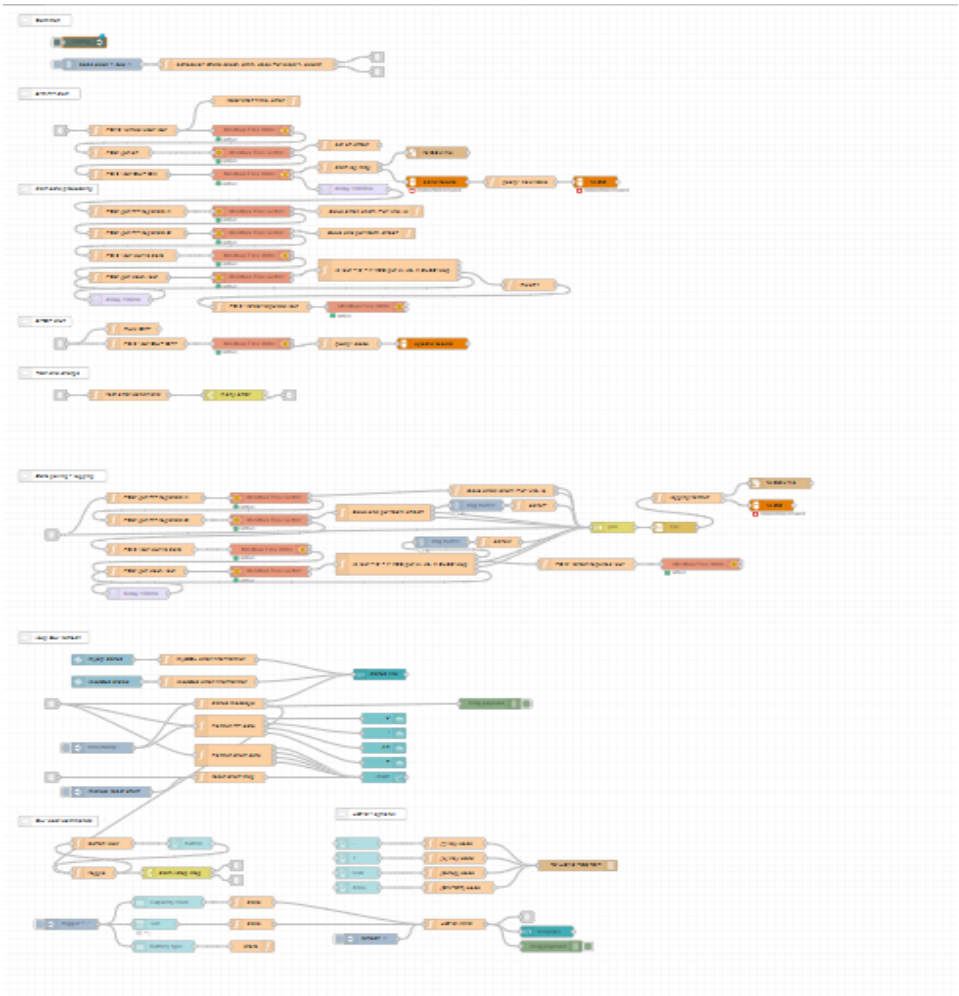
node-red implementation



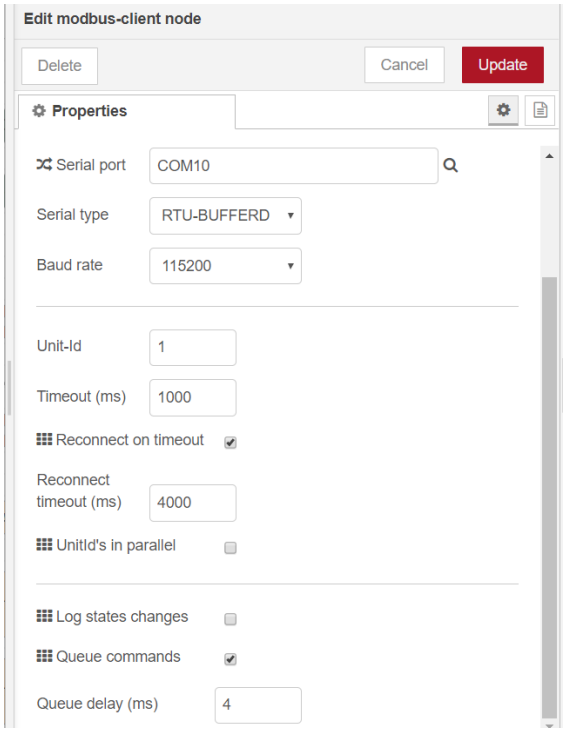
- The UD18 is totally independent: the required values are show in the right pane (after start) and the UD18 can be set from UI via Bluetooth (see https://github.com/msillano/UD18-protocol-and-node-red for details) or directly using the HW buttons. It is never read.
- Data are stored every 10 seconds, as CSV file or in MySQL tables.
- Some default values are defined in `Config` node and the user can change it.
- Todo: this implementation is a work in progress: now it is usable, but requires more development in the areas:

1. Startup and config
2. Errors management
3. Workflow, point 4 not implemented (new mode AUTO).
4. WEB application for MySQL

- The `node-red-contrib-circularbuffer` creates connection problems to `modbus-flex-write`. I can't use it. The more simple `node-red-contrib-ring-buffer` works ok.
- The USB connection PC-RD6006 is enough stable. But if it don't auto-restart:
  1. disconnect and reconnect the USB cable
  2. do *Deploy/Restart Flows*
  3. In one case I had to reinstall the driver CH341SER.EXE (warning: this can change the COM port)



Installation



1. A **WAMP** (or LAMP, or ...) is required to have *mySQL* and *phpMyAdmin*.
2. Add to the ***node-red*** *'palette'* the packages: ***`node-red-node-serialport, node-red-node-mysql, node-red-contrib-modbus, node-red-contrib-ring-buffer.`***
3. Copy the contents of the file `charger2.json` to the clipboard and import it into a new flow in **node-red**.
4. Update 'configuration nodes': '`batteries`' for mySQL and '`RD6006`' for MODBUS serial. Here my configuration.
5. The file `charger2.sql` contains the mySQL db definitions. Import it using *phpMyAdmin*.