Design, Build and Validation of a Low-Cost Programmable Battery Cycler

Propp, Karsten; Fotouhi, Abbas; Knap, Vaclav; Auger, Daniel J.

Published in:
ECS Transactions

DOI (link to publication from Publisher):
10.1149/07401.0101ecst

Publication date:
2016

Document Version
Accepted author manuscript, peer reviewed version

Design, Build and Validation of a Low-Cost Programmable Battery Cycler

K. Propp[a], A. Fotouhi[a], V. Knap[b] and D. J. Auger[a]

[a] School of Aerospace, Transport and Manufacturing, Cranfield University College Road, Cranfield, Bedford, MK43 0AL, UK
[b] Department of Energy Technology, Aalborg University, 9000 Aalborg, Denmark
Email: d.j.auger@cranfield.ac.uk

The availability of laboratory grade equipment for battery tests is usually limited due to high costs of the hardware. Especially for lithium-sulfur (Li-S) batteries these experiments can be time intensive since the cells need to be precycled and are usually cycled with relatively low loads. To improve the availability of test hardware, this paper conducts a study to design and test a low cost solution for cycling and testing batteries for tasks that do not necessarily need the high precision of professional hardware. While the described solution is in principle independent of the cell chemistry, here it is specifically optimized to fit to Li-S batteries. To evaluate the accuracy of the presented battery cycler, the hardware is tested and compared with a professional Kepco bipolar power source. The results indicate the usefulness for application oriented battery tests with real life cycles, although inaccuracies occur in the current measurements.

Introduction

A promising way to increase the energy density of battery-based technologies is the lithium-sulfur (Li-S) chemistry due to its high theoretical specific energy density. Due to recent improvements of the understanding of the cell reactions (1) (2), the first commercially available cells appeared on the marked. This opens the opportunity for application-oriented research, which mainly predicts the suitability, lifetime and size of cells or battery-packs for varying user scenarios. The involved tests for this usually involve battery cycling under different conditions so as current profiles, power demands, temperatures and age in a controlled environment. The test equipment for these experiments generally consists of a physical discharge facility, a data acquisition system and host computer for setting the current profiles and logging the data (3) (4). These experimental layouts share the similarity that they rely on precise, complex and also expensive components, which are therefore usually limited in their availability. Li-S battery reduces the availability further since the actual performance of the Li-S cells is highly influenced by the short term discharge history (5) (6), commonly referred as 'history' effect. In order to exclude this 'history' effect it is necessary to 'reset' the cell, which is done by pre-condition cycling, as shown for example in (7), where the cell is cycled with low currents (discharge $C/5$ and charge $C/10$) before executing the target test procedure. The needed accuracy for these tests, and some other tests, is significantly lower. Therefore in this work a low cost, easy to use simulation and cycling tool for battery cells or small packs is developed on the base of a previously presented programmable discharger (8). The parts and functionality of the cycler are described in

section “Test bench hardware”, whereas the “test bench software” section contains a description of the software as well as the MATLAB code for execution. To evaluate the performance of the presented cycler, the “Experiments” section contains the experimental layout of tests, comparing the presented cycler with professional laboratory grade equipment. The results of these tests are presented and discussed in section “Results and discussion”.

Test bench hardware

As well as the hardware for standard test rigs, the presented cycler contains a physical charge/discharge circuit and a data acquisition device. For the latter an Arduino Uno microcontroller is used, which sends the demanded currents, the information whether the battery should be charged or discharged and the voltage measurements via the USB port to the host computer. The basic physical charge/discharge device itself consists of three main parts, which are explained in detail in the following.

Current controlled circuit

The main functionality of the current controlled charge or discharge is described in (9) and is adjusted in order to create a programmable battery discharger in (8). It consists of a MOSFET (IRF630) and a shunt power resistor (1 ohm; 50 watts). The MOSFET is controlled by two parallel operational amplifiers (LT1001) and acts as a variable resistor. Since the shunt resistor has a value of one ohm, the control voltage V_{in} also is equal to the current in the circuit (Fig. 1). Therefore, a control voltage of 0−5 V will generate equivalent currents of 0−5 A (see (10) for more details). To switch between charge and discharge current, without disturbing the functionality of the controlled current circuit, three relays are used due to their simple usage and their mechanical short circuit protection. As shown in figure 1, they guide the current through the battery in charge or discharge condition. To limit the amount of code and communication time to send and receive commands from MATLAB to the Arduino, all three relays are switched simultaneously with one Arduino I/O port.

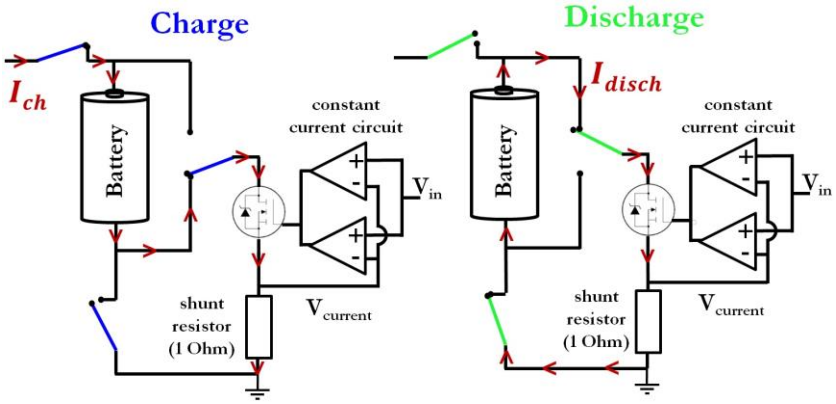

Figure 1 Battery simulation circuit

Power Source

As a cheap and easily available power source with the right specifications a computer power supply is used. With its stable output voltage of ±12 V it cannot only be used to charge the battery, but also to provide power to the operational amplifiers, controlling the current.

Arduino Uno

An Arduino Uno microcontroller (11) is used, due to its simple MATLAB/Simulink connectivity. For the measurement of the terminal cell voltage another operational amplifier is added, used as a voltage subtractor, to the circuit because the battery potential changes between charge and discharge in relation to the ground. Furthermore the batteries terminal voltage is multiplied times two, since the maximum charging voltage, recommended by OXIS Energy, for the tested Li-S cell is with 2.45 V about half as much as the A/D converter ports of the Arduino are able to measure. Hereby the range of the A/D converter ports of 5 V is optimally used and the resolution of the measurement is doubled (Fig. 2).

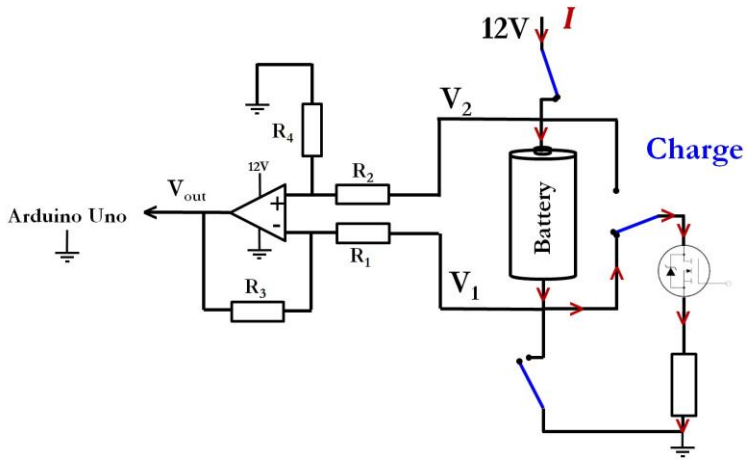

Figure 2 Voltage subtractor for cell voltage measurement

The output signals from the host computer to the Arduino are the charge/discharge signal to one I/O port and the control voltage V_{in}, generating the currents in combination with the current controlled circuit. One PWM signal (8 bit) of the Arduino and a low pass filter (Fig. 3) are used to create the control voltage physically. The input signals, measured with two 10 bit A/D converter ports, are the actual terminal voltage of the battery and the current. As stated before, the current measurements are gathered by measuring the voltage over the shunt resistor. For the sake of completeness also the external precision voltage reference (LT1019) for the Arduino shall be mentioned, since it improved the accuracy and reduced the measurement noise significantly. To summarize the functionality of the battery cycler, the whole concept of the circuit is described stepwise as illustrated in figure 3.

- The host computer sets the I/O port of the Arduino to charge- or discharge condition (all relays are connected to one port).
- The host computer sends a current demand (integer from 0 to 255) to the Arduino.

- The Arduino changes its PWM output ratio according to that and via the low pass filter the control voltage V_{in} between 0 and 5 V is generated.
- The current controlled circuit sets the current according to the control voltage V_{in}.
- The host computer asks for the voltage measurements of the shunt resistor (current) and the operational amplifier output (battery terminal voltage).

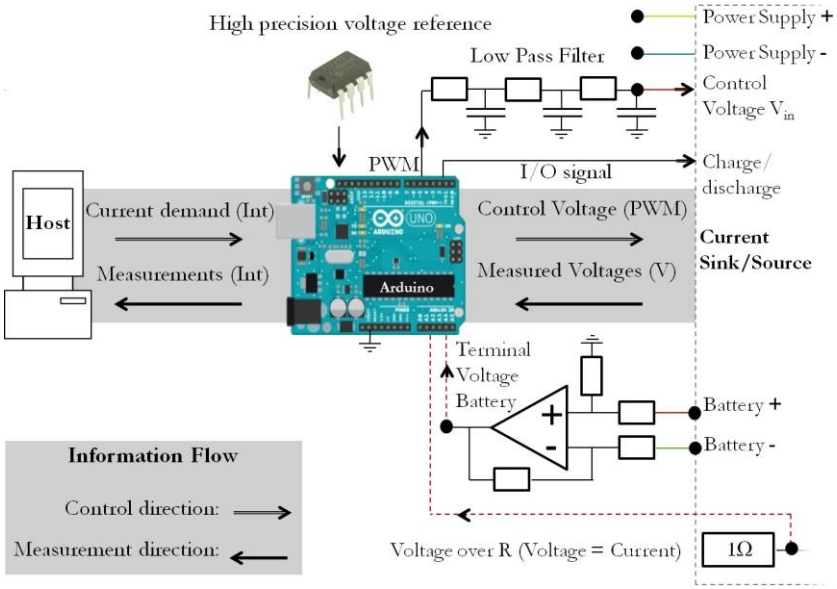

Figure 3 Communication Computer − Arduino − Current Sink

Due to the hardware is only capable of constant current charging; the battery cycler is mostly suited for Li-S and Ni-MH batteries. Since Li-ion batteries are charged in constant voltage condition as well and are generally sensitive to overcharging, it is recommended not to be used with the presented set-up. However, the addition of hardware or software based constant voltage charging is possible without much effort.

Test bench software

The main two software components of the battery cycler are the program running on the Arduino itself and the MATLAB code for generating the current demands and storing the measurements on the host computer. The former one is provided by the Arduino-IO Package (12) from MATLAB, enabling the user to execute Arduino likewise commands from the host computer and use the information directly within the MATLAB workspace environment (13). The MATLAB code for the cycling consists mainly of one vector containing the information (integers) of the current demand for every second (signal vector) and two vectors storing the measurements of current and terminal voltage. An easy and quick way to generate the signal vector is to use the Simulink (14) environment. As shown in figure 4, the signal vector contains three rows: The first row includes the current demand for the discharge, the second one the information for the charge and the third one contains only zeros for stopping the current flow without stopping the loop. Thereby the information whether the battery is charged or discharged is included by the sign of the integer. Positive numbers represents charging and negative discharging. Therefore in principle the charge vector can also contain discharge parts. The different rows are mainly to change the pattern of the current demand between the maximum and minimum battery voltage. An example for this can be seen in figure 5 where the charge

vector is constant and the discharge vector represents a drive cycle. Here the user has all the flexibility of the MATLAB environment for individual changes.

The code for the actual cycling is mainly a 'while' loop, executed once per second until a defined end time (see Appendix). The included functions of the loop are to define charge or discharge condition, send the current demand, store the measurements and switch between charge and discharge row within a defined voltage window. When the measured battery terminal voltage reaches the allocated maximum for the tested cell in charging condition, the source of the signal vector is changed to the discharge row and vice versa. Furthermore each time the maximum battery voltage is exceeded, a counter (cycle) is added to count the number of full battery cycles. Specifically for Li-S batteries, a timer is included that limits the maximal charge time to ten hours, because under some conditions (age, temperature, shuttle) the maximal voltage is not sufficient to prevent battery overcharges.

The presented code in the appendix is a simple example for the application of the presented battery test bench. The interested user is obliged to experiment and improve the program since MATLAB offers a large variety of functions usable in this scenario.

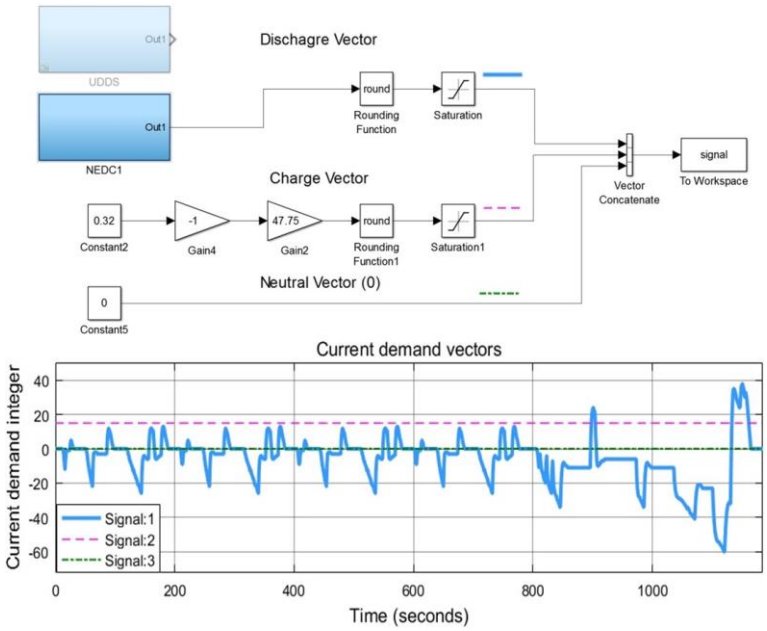

Figure 4 Simulink model for current demand vector

Experiments

The presented solution for battery cycler is in theory capable of not only cycling the cell with constant current conditions, but also to simulate realistic current profiles. Therefore it is of interest how accurate the measurements are for voltage limits and if the low cost solution can supplement the standard hardware in some cases. To test this, the battery cycler is compared with laboratory grade test equipment, a Kepco bipolar power source/sink (Kepco BOP100-10MG). As an initial test one cell is cycled with the same current profile and environmental conditions (thermal chamber at 20 °C) with the two devices, firstly with the Arduino based solution and secondly with the BOP100-10MG. Afterwards, the individually taken measurements are compared. Since often in

application oriented battery cell research the behavior of a cell in a practical application is tested, a New European Drive Cycle (NEDC) (15) related current profile (16) is used due to its good mix of realism and simplicity. For the test an OXIS Energy Li-S 3.4 Ah long life chemistry pouch cell was charged with a constant current until the maximum voltage of 2.45 V, followed by the NEDC discharge profile shown in (Fig. 5 A) to the depletion state at 1.5 V. Since the measurements are directly stored as input (current), output (terminal voltage) and time vector, the post-processing of the data in MATLAB (17) is straight forward and no data conversion is needed.

Since this test unveils some deviations between the two devices, a second test procedure with a pulse current profile (Fig. 5 B) is done to find methodological or hardware related differences. Here the measurements of cell voltage and current in the circuit are taken in parallel to the measurements of the cycle devices. For these a three month old Uni-T 71E multimeter was used that is capable to send its measurements via USB to the host computer with software provided by the manufacturer.

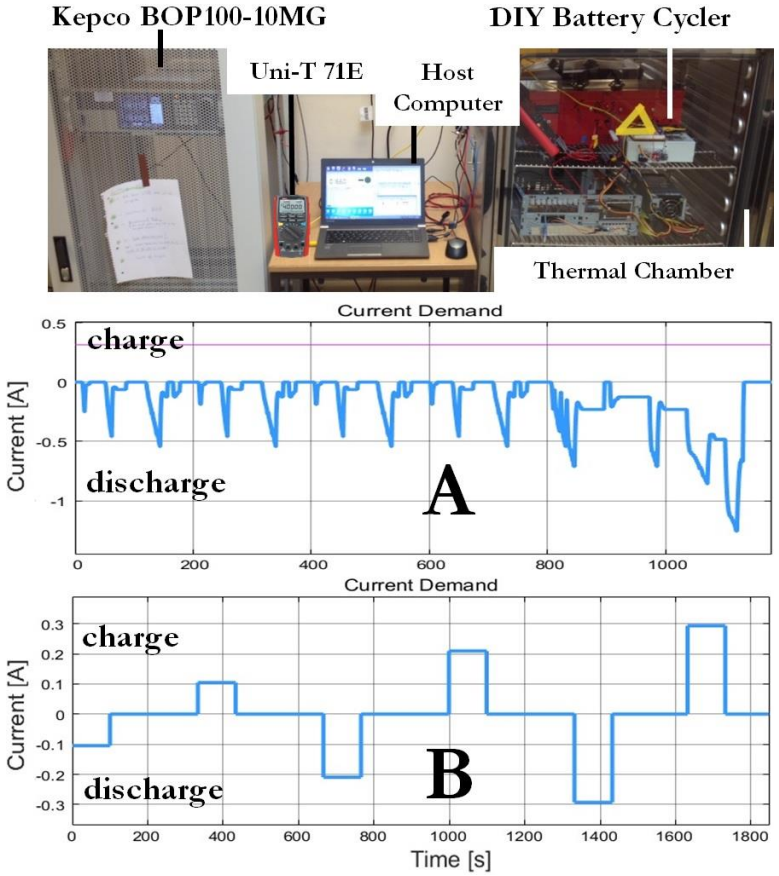

Figure 5 Test set-up and discharge current profiles

Results and discussion

Firstly the drive cycle data is evaluated, which shows that both measurements are sufficiently close at the beginning of the test (Fig. 6). Here the most obvious differences between both cycle devices are the current demand and measurement resolutions. While the Kepco is able to generate a smooth current profile with steps of 1−2 mA, the smallest

possible step size with the Arduino is 20 mA. This is due to the 8 bit resolution of the PWM signal, which can only generate 256 different states. With the maximum voltage of 5 V, representing similar currents, the step size is limited. For an increased resolution a digitally controlled potentiometer could be a solution. With the voltage measurement however, the resolution is with 2 mV about twice as high as the Kepco ones. This is due the optimized measurement range for Li-S batteries of maximum 2.5 V. With other cells or small cell packs the resolution decreases proportionally to the maximum cell voltage.

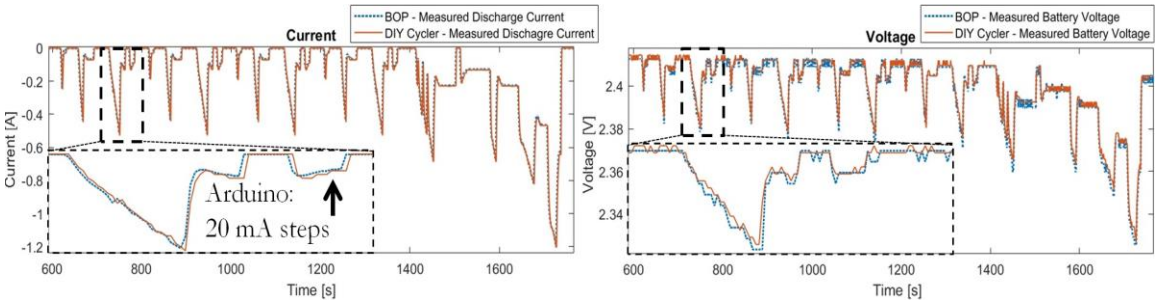

Figure 6 Voltage and current measurements for an extract of the drive cycle test (lines almost overlay)

Generally the measurements at the beginning of the cycle test can be evaluated as equally useful for application oriented battery research. However, when examining the whole discharge process (Fig. 7) the differences become clear.

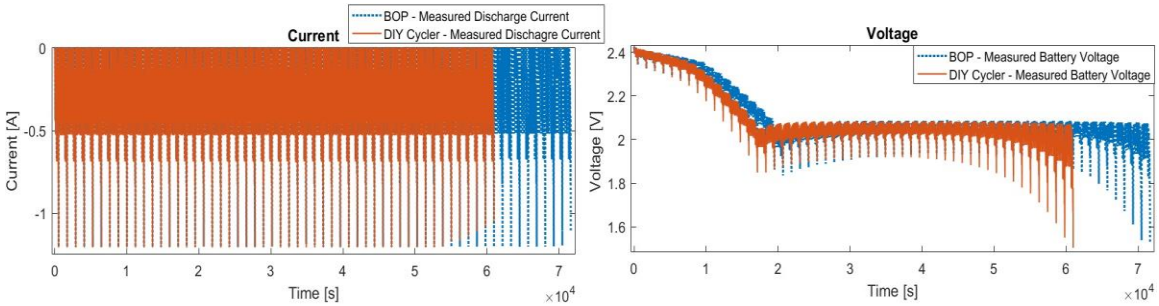

Figure 7 Voltage and current measurements for the whole of the drive cycle test

The discharge duration of the Kepco cycler is about three hours longer than the DIY solution, which also means a significantly difference in the usable capacity of the battery. The cumulated current varies between 10746 As measured with the Kepco and 9384 As with the Arduino.

To find the origins of the detected difference the pulse tests with the multimeter are used. Hereby the multimeter values, with its resolution 0.1 mV and accuracy of ±0.025 % + 5 in the voltage mode of 4 V and a resolution of 0.01 mA with a accuracy of ±0.01 % +15 in the current range of 400 mA (18), are seen as the reference for both cycle devices with one exception. When examining the measurements of all three devices (Fig. 8 and Fig. 9) it is visible that there is a difference in the time code of the measurements, leading to an increasing divergence between the multimeter and both of the cyclers. The origins of that are hard to determine since the communication between the Uni-T 71E and the host computer is done with software from the manufacturer, running in parallel to the MATLAB script. Here we assume the tic, toc command within the MATLAB environment to be accurate (19) and evaluate only the vertical difference between the measurements.

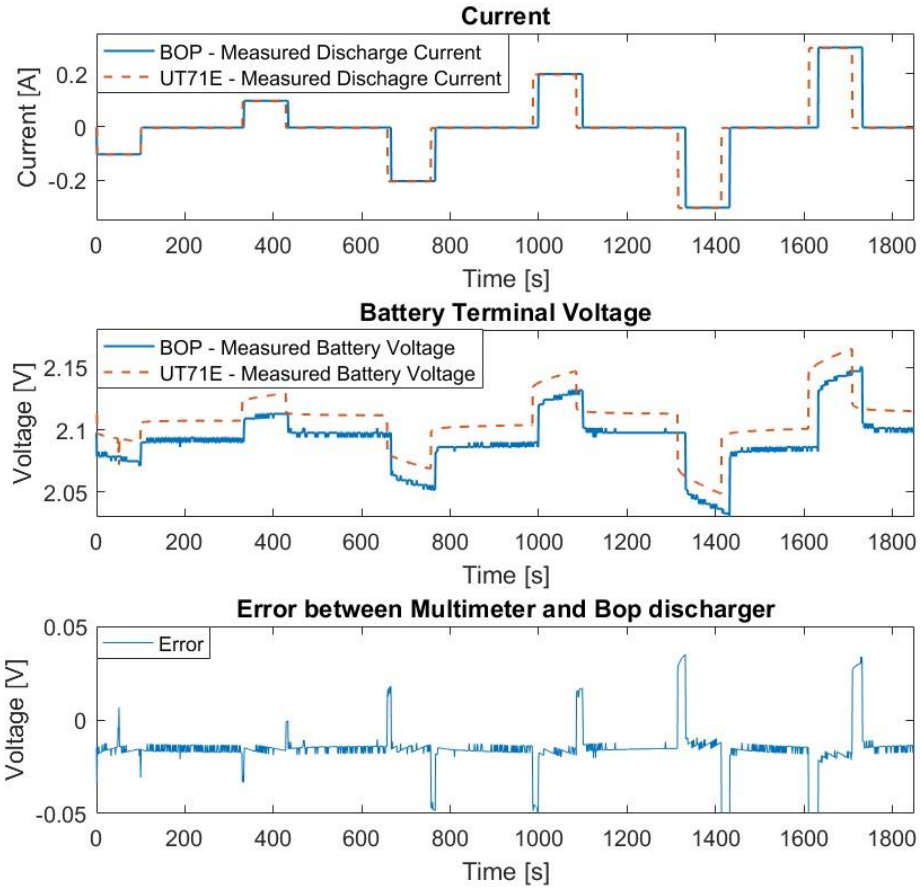

Figure 8 Voltage and current measurements of the Uni-T 71E and the Kepco cycler

As indicated by the drive cycle tests the voltage measurements of both devices are relatively similar. When comparing them with the multimeter measurements the Kepco is with an average offset of −15.5 mV slightly more imprecise than the Arduino (−7.6 mV), which is presumably due to the larger measurement range of the Kepco of 100 V. For the current however this is different. While the Kepco current data is within a 1 to 2 mA range to the Uni-T measurements, the differences of the Arduino are more pronounced. While discharging, the multimeter values show about 18 mA more current flow than the Arduino. Since the same effect is also visible during charging the reason for this deviation are likely tolerances of the 1 Ohm shunt resistor, governing the current control, and leakage currents in the circuit. Here lays presumably the reason for the difference in the usable capacity of the battery in the drive cycle test. When multiplying the 18 mA difference with the discharge time of the Kepco device (0.018 A × 72,000 s) the resulting calculated lost capacity is with 1296 As very close to the measured one of 1362 As. Therefore the presented hardware is capable of running real life tests with the limitation of precise results regarding the cell capacity.

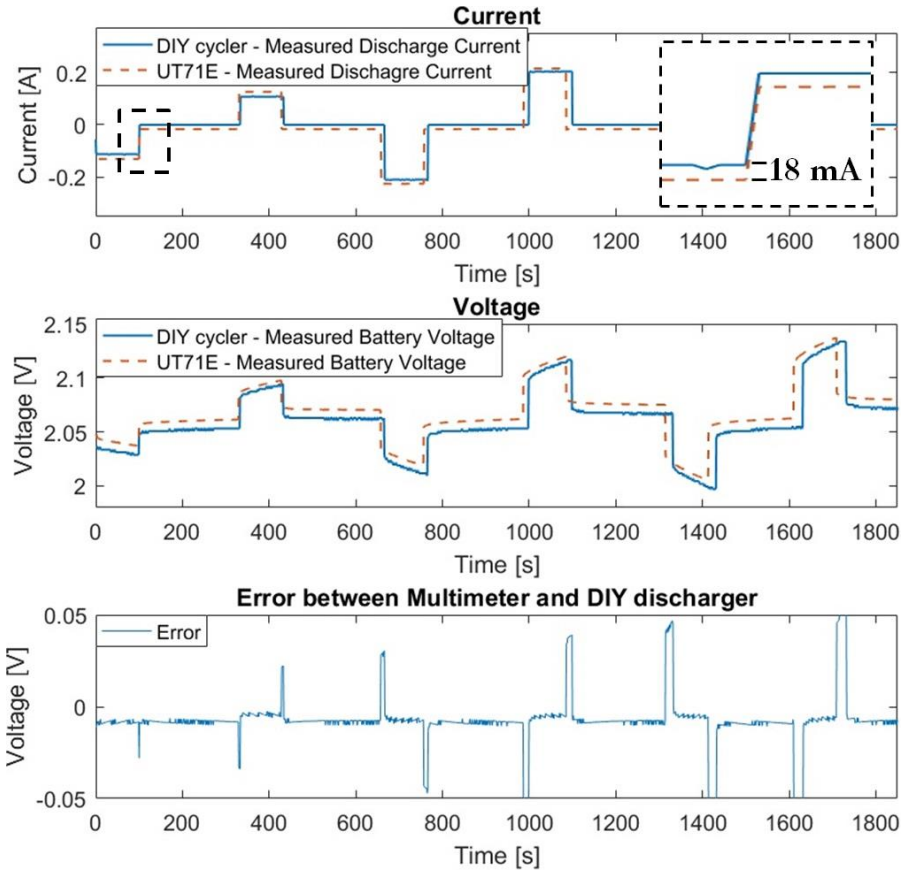

Figure 9 Voltage and current measurements of the Uni-T 71E and the Arduino cycler

Conclusion

A low cost battery cycle solution was presented with details for its three main parts, the physical discharge circuit, the Arduino Uno and the software within the MATLAB programming environment. The approach showed to be not only capable to be a simple and cheap way for increasing the availability of professional laboratory equipment for pre-cycling Li-S calls, but also enables the simulation of different user cases for batteries. However, due to limitations of the maximal discharge and charge currents and methodological errors in the current measurements, the determined values are not as accurate as professional hardware. This counts especially for the cumulated current which is usually used to determine the capacity of the battery. However, due to the flexibility of the MATLAB environment, the simplicity of the hardware and the possible improvements the proposed rig has the potential to supplement existing laboratory equipment. Our future goals lay in the improvement of the current control circuit and in the addition of a simple way for temperature control.

Acknowledgement

This research was undertaken as part of the Revolutionary Electric Vehicle Battery project, co-funded by Innovate UK. University funding for this project is administered by EPSRC under grant number EP/L505286/1. Enquiries for access to the data referred to in this article should be directed to researchdata@cranfield.ac.uk.

References

1. D. Bresser, S.Passerini, B. Scrosati, *Chemical Communications* 49 (2013) 10545–10562.
2. M. Wild, L. O'Neill, T. Zhang, R. Purkayastha, G. Minton, M. Marinescu, G. Offer, *Energy & Environmental Science* 8 (2015) 3477–3494.
3. H. He, R. Xiong, H. Guo, *Applied Energy* 89 (2012) 413–420.
4. R. Xiong, H. He, F. Sun, K. Zhao, *IEEE Transactions on Vehicular Technology* 62 (2013) 108–117.
5. A. Rosenman, R. Elazari, G. Salitra, E. Markevich, D. Aurbach, A. Garsuch, *Journal of The Electrochemical Society* 162 (2015) A470–A473.
6. A. Ferrese, J. Newman, *Journal of The Electrochemical Society* 161 (2014) A948–A954.
7. V. Knap, D. -I. Stroe, R. Teodorescu, M. Swierczynski, T. Stanciu, *IEEE Energy Conversion Congress and Exposition (ECCE)* (2015), pp.1375–1381.
8. K. Propp, A. Fotouhi, D. J. Auger, *Computer Science and Electronic Engineering Conference (CEEC)* (2015), pp.225–230.
9. E. Holland, Everything including the kitchen (current) sink!, http://embeddederic.blogspot.co.uk/2011/07/everything-including-kitchen-current.html ,2011, Accessed:2015-08-23.
10. D. L. Jones, Diy constant current dummy load for power supply and battery testing, http://www.eevblog.com/2010/08/01/eevblog-102-diy-constant-current-dummy-load-for-power-supply-and-battery-testing , 2009, Accessed: 2015-05-30.
11. Arduino website, http://www.arduino.cc , 2015 Accessed: 2015-05-16.
12. Math Works, Matlab support for Arduino (aka Arduino io package), http://www.mathworks.com/matlabcentral/fileexchange/32374-matlab-support-for-arduino--aka-arduinoio-package- ,2014, Accessed: 2015-04-06.
13. G. Campa, Arduino io package: Slides and examples, http://www.mathworks.com/matlabcentral/fileexchange/27843-arduino-io-package--slides-and-examples ,2014, Accessed: 2016-01-017.
14. Simulink version 8.5 (R2015a), The Mathworks, Inc., Natick, Massachusetts, 2015.
15. S. Samuel, L. Austin, D. Morrey, *Proceedings of the Institution of Mechanical Engineers, PartD: Journal of Automobile Engineering 216* (2002) 555–564.
16. R. Kötz, S. Müller, M. Bärtschi, B. Schnyder, P. Dietrich, F. Büchi, A. Tsukada, G. Scherer, P. Rodatz, O. Garcia, *ECS Electro Chemical Society*, 52ndMeeting, San Francisco.
17. MATLAB version 8.5.0.197613 (R2015a), The Mathworks, Inc., Natick, Massachusetts, 2015.
18. UNI-T, Intelligent digital multimeters ut71e, http://www.uni-trend.com/productsdetail2.aspx?ProductsID=1169&ProductsCateId=908&CateId=908, 2006, Accessed: 2016-08-06.
19. M. Knapp-Cordes, B. McKeeman, Improvements to tic and toc functions for measuring absolute elapsed time performance in matlab, http://uk.mathworks.com/company/newsletters/articles/improvements-to-tic-and-toc-functions-for-measuring-absolute-elapsed-time-performance-in-matlab.html , 2011, Accessed: 2016-07-04.

Appendix

```
while toc < maxtime   %maxtime: maximum testing time, i:iteration of the loop (initially = 1)
if (signal(i,cd) < 0) %signal: three column current demand vector, cd: charge (2)/discharge (1) column
a.digitalWrite(Relay123, 0); %Charge/discharge configuration of the relays to Arduino I/O output port
else a: Arduino on serial port
a.digitalWrite(Relay123, 1);
end
time(i) = toc; %time: measurement Time
a.analogWrite(PWM_out, abs(signal(i,cd))); %Write current demand on PWM output,
v1(i) = a.analogRead(AD1); %Read the voltage over the 1 Ohm resistor -> Current,
Curr_Volt1(i) = (v1(i)/196.6); %AD1: Arduino A/D converter port, Curr_Volt1: measured current
if (i == 1) %Plot charge current negative
Curr_Volt1(i) = Curr_Volt1(i);
else
if (signal((i-1),cd) < 0)
Curr_Volt1(i) = Curr_Volt1(i) * -1;
end
end
figure(3);
plot(time, Curr_Volt1, 'g');
legend('Actual Current Load on Battery [A]');
v2(i) = a.analogRead(AD2); %Read the Battery Terminal Voltage
Batt_Volt2(i) =  (v2(i)/409.6); %AD2: Arduino A/D converter port, Batt_Volt2: measured battery voltage
figure(4);
plot(time, Batt_Volt2, 'c' );
legend('Battery Terminal Voltage [V]');

if (Batt_Volt2(i) < Vmin) %Switch between charge/discharge row (signal vector)
cd =  2 ; %switch to charge row, Vmin: minimum allowed battery voltage
end

if (cd == 2) %count charge time  in seconds (if fs = 1)
chargeTimer = chargeTimer + 1;
end

if (Batt_Volt2(i) > Vmax) || (chargeTimer > maxChargeTime) %maxChargeTime: maximum allowed charge time
cd = 1; %switch to discharge row, Vmax: maximum allowed battery voltage
chargeTimer = 1; %set charge timer back to one
cycle = cycle + 1; %counter for charge/discharge cycles
end

if (cycle == cycle_max) %cycle_max: maximum allowed chare/discharge cycles
cd = 3; %switches to 0 Current
end
drawnow
display(cycle);
while(ni == i) %Wait for appropriate time for next measurement
ni = floor(toc*fs)+1; %sampling frequency in hz (here we use 1 Hz)
end
i = ni;
end
```