electronics

MDPI

Article

Open and Flexible Li-ion Battery Tester Based on Python Language and Raspberry Pi

Andrea Carloni, Federico Baronti, Roberto Di Rienzo, Roberto Roncella and Roberto Saletti *

Dipartimento di Ingegneria dell'Informazione (DII), University of Pisa, Via Caruso 16, 56122 Pisa, Italy; andrea.carloni@ing.unipi.it (A.C.); f.baronti@iet.unipi.it (F.B.); dirienzo.r@gmail.com (R.D.R.); r.roncella@iet.unipi.it (R.R.)

* Correspondence: roberto.saletti@unipi.it; Tel.: +39-050-2217648

Received: 29 October 2018; Accepted: 14 December 2018; Published: 19 December 2018

Abstract: Technology improvements and cost reduction allow electrochemical energy storage systems based on Lithium-ion cells to massively be used in medium-power applications, where the low system cost is the major constraint. Battery pack maintenance services are expected to be required more often in the future. For this reason, a low-cost instrumentation able to characterize the cells of a battery pack is needed. Several works use low-cost programmable units as Li-ion cell tester, but they are generally based on proprietary-software running on a personal computer. This work introduces an open-source software architecture based on Python language to control common low-cost commercial laboratory instruments. The Python software application is executed on a Raspberry Pi board, which represents the control block of the hardware architecture, instead of a personal computer. The good results obtained during the validation process demonstrate that the proposed cell station tester features measurement accuracy and precision suitable for the characterization of Li-ion cells. Finally, as a simple example of application, the state of health of twenty 40 Ah $LiFePO_4$ cells belonging to a battery pack used in an E-scooter was successfully determined.

Keywords: open-source; python; raspberry Pi; cell station tester; state-of-health; lithium-ion cells; Programmable test

1. Introduction

In the last years, Lithium-Ion Batteries (LIBs) are increasingly penetrating the high-power mobility applications, such as the Electric Vehicles (EV) and Hybrid Electric Vehicles (HEV), because of the higher power and energy densities compared to other electrochemical Energy Storage System (ESS) technologies such as lead-acid and Nickel-Cadmium ones [1]. Nevertheless, LIB cost still limits its utilization in medium-power electric mobility field, such as in E-scooters, where the cost of the ESS still remains a large part of the total cost of the system and it is, therefore, one of the major issues for the full exploitation of the LIB technology in this field. However, the growing interest in the electrification of two-wheel vehicles [2] and the research efforts in LIB manufacturing that are leading to improved energy density and reduced cost of the batteries [3] will soon bring a large penetration of the LIB technology in these kinds of vehicles also.

Generally, an LIB pack configuration is fixed and is chosen by EV manufacturers at design time. As the performance of a battery is limited by the weakest cell, the entire battery pack must be replaced when that cell reaches its end-of-life (EOL) point. To enhance the configuration flexibility of an LIB pack and to extend its lifetime, the idea of adopting reconfigurable battery packs has gained attention in recent years. In [4] the authors propose an LIB maintenance service based on a refitting procedure. The refitting procedure consists of replacing the most degraded cells with new ones, without replacing

the entire pack, in order to maintain the battery State of Health (SoH) above a defined threshold. The weak aspect of this approach is that battery assembly is more complex, as it must allow the individual cell replacement.

In this refitting scenario, test equipment capable of characterizing the cells of an LIB, in particular, to measure their SoH, is needed. The health of LIBs changes over time due to the irreversible physical and chemical processes occurring inside the cells that result in a loss of capacity and in an increment of the internal resistance. Therefore, the SoH can be defined using either one of these two metrics. In the first case, the actual deliverable capacity of the cell is compared to the value when the battery is fresh, whereas the actual internal resistance is compared to its nominal one in the second case [5]. However, in a fully electric vehicle, such as an E-scooter, the capacity is the most interesting parameter, since it determines the operating range of the vehicle [5]. As indicated in [6], LIBs used in electric traction applications are considered at end of their life when they have lost 20% of the nominal capacity, i.e., when SoH reaches 80%.

As indicated in [7] and shown in Table 1, the commercially available test equipment for Li-ion cells can be divided into three groups: electrochemical workstations that use high frequency and advanced analysis techniques on a single cell, such as Electrochemical Impedance Spectroscopy (EIS), but with a limited current and power range; high-current cell testers that cycle the single cell with a limited support for high frequency analysis, and finally high-current and high-voltage battery tester that can cycle a complete battery pack. Considering that a typical E-scooter engine power ranges from 200 W [2] up to some kilowatts [8], and considering that the measurement of the capacity of a cell usually consists of a current integration operation [9,10] that does not require complex high frequency analysis, the two last instrument categories would be the preferred choice for extracting the SoH of an LIB in this kind of application. However, the cost of such instruments seems too high to be affordable to mechanical workshops.

Table 1. Performance data of typical test system [7].

Parameter	Electrochemical Workstations	High-Current Cell Tester	Battery Test Systems
Current range	±2.5 A	±250 A	±1000 A
Voltage range	0 V–5 V	0 V–5 V	0 V–600 V
Max. sampling frequency	2 kHz–4 MHz	100 Hz	50 Hz
High frequency analysis	Yes	No	No
Realistic load profiles	No	Yes	Yes
Individual aging-analysis on large number of cells	No	Without high frequency analysis	No

Some works found in the literature carry out specific studies on LIB by using low-cost test equipment composed by the assembly of common lab instrumentations and a Personal Computer that controls all of them by means of a specific software. The authors use an experimental setup to test a 8 Ah–24 V battery pack in [11]. However, such an interesting solution is not directly applicable to the test of a typical battery pack used in E-scooters [8,12], because of the intrinsic power limits of the lab instrumentation employed. To overcome these power limits, an electronic load and a charger with improved power range are used to characterize a single Li-ion cell with higher capacity, 25 Ah and 40 Ah, respectively in [10] and [13].

However, this kind of equipment is often controlled by means of proprietary-software packages, with which the test operations are managed, or by commercial software suites that allow the user to create a specific and customized application to control the test operations. This proprietary software is usually licensed to the end user under clauses that limit the user freedom and strictly fix how the software can be used. As an example, the application software package BPChecker2000 limits the user to use it only with a specifically owned lab instrumentation [11], being the modification and the redistribution of the code to third-parties denied [14]. The National Instruments LabView software

suite development platforms adopted in [10] and [13] allow the user to modify the source code of the LabVIEW application software or to realize a new one after the purchase of a temporary license [15].

Instead, Open-Source-Software (OSS) packages are usually licensed to guarantee four essential freedoms to the end user [16,17]: the freedom to run the program for any purpose; the freedom to study how the program works; the freedom to redistribute copies of the software; the freedom to improve the program and freely release the improvements to the community. Usually, an OSS license requires that the modified copies have to be protected by the same license of the original release, in order to propagate the OSS benefits to the user community and to avoid the possibility that an OSS would become proprietary [18]. An empirical research work [19] shows that OSS approach fosters improved creativity, since the user community can modify the source code and adapt it to further purposes. Moreover, the large number of users that become testers accelerates the process of defect finding and fixing.

For the above reasons, we developed a low-cost, open-source test equipment able to characterize the cells composing a battery pack. The hardware architecture proposed in this work consists of common low-cost programmable instruments managed by an application software, as in other works. However, our innovative contribution consists of a control application software based on the open-source Python language that runs on a Raspberry Pi instead of a PC. The Raspberry Pi board represents the basic building block of the new hardware architecture proposed in this paper and is an appealing low-cost alternative to a personal computer.

The rest of the paper is organized as follows: Section 2 describes the instrument specification and the hardware and software frameworks, Section 3 describes the experimental results that first show the timing reliability and the achieved voltage and current measurement accuracy of the platform. Then, an example of how the platform can be used to characterize a single Li-ion cell is shown. In particular, the SoH analysis of a second-hand LIB pack is performed. Finally, Section 4 draws the conclusions and possible future works.

2. Materials and Methods

2.1. Instrument Specification

A cell station tester able to characterize a single cell of an LIB for medium-power mobility applications such as E-scooter, E-bike and E-forklift should be characterized by the parameters shown in Table 2. The maximum cell input voltage applicable to the instrument should be at least 4.5 V, in order to be compatible to all the different lithium-ion cell chemistries [1]. Furthermore, even if the cell capacity in medium-power LIB varies from manufacturer to manufacturer and from application to application, a reasonable choice is to tailor the instrument to battery packs with cells the nominal capacity of which ranges from 10 Ah [20] to 60 Ah [21]. Finally, the order of magnitude of the discharging and charging currents that the instrument should sustain is 100 A and 50 A, respectively, according to the typical discharge and recharge maximum current rates of the applications. The above-mentioned values are the design parameters of the proposed tester.

Table 2. Proposed cell station tester design specification.

CST Parameter	Target Specification
Instrument typology	Single Cell station tester
Nominal cell capacity range	From 10 to 60 Ah
Input voltage range	Up to 4.5 V
Maximum discharge current	100 A
Maximum charge current	50 A

2.2. Hardware Framework

Figure 1 shows the hardware framework of the proposed Cell Station Tester (CST). The overall system consists of two low-cost and programmable lab instruments, i.e., an electronic load and a charger (Force Block), a Data Acquisition device (Measure Block) and a Single Board Computer (SBC) that controls the system (Control Block).

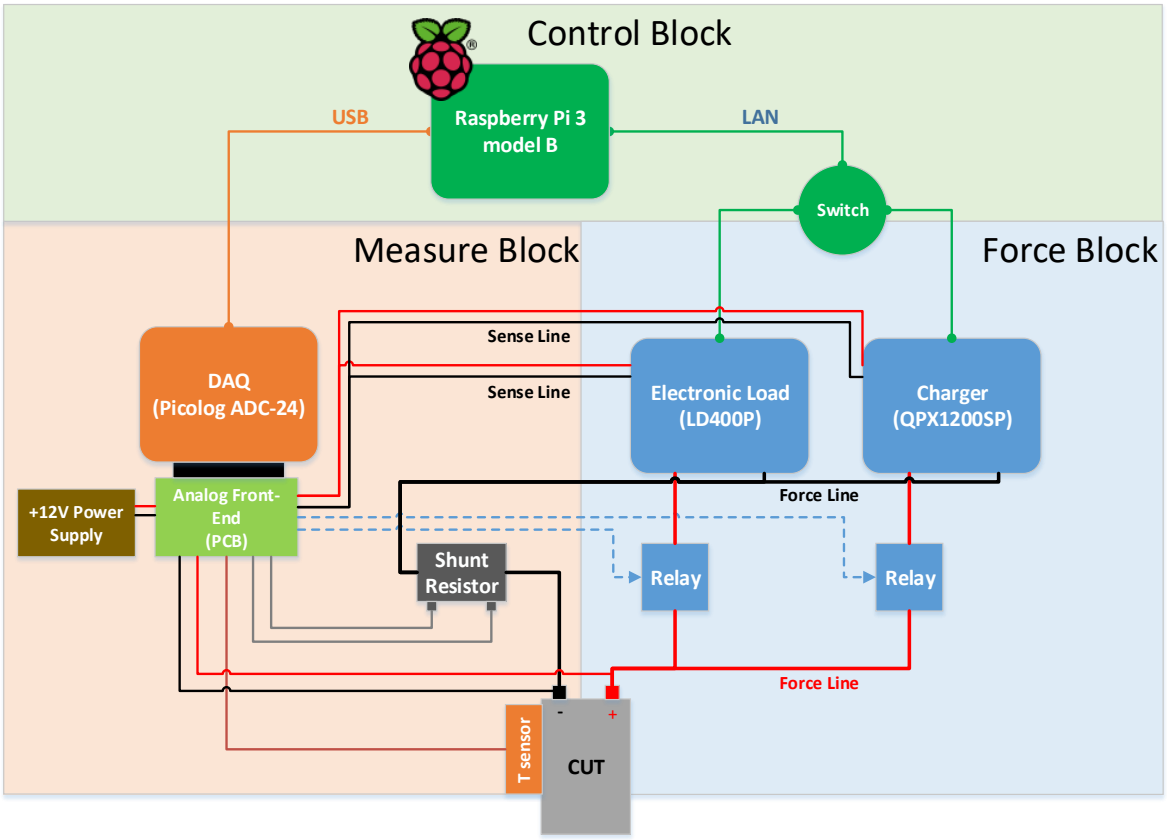


Figure 1. Hardware framework of the Cell Station Tester.

The electronic load LD400P and the charger QPX1200SP constitute the "force block" of the system. These two low-cost lab devices support charging and discharging currents comparable to the values reported in Table 2. They are programmable devices, so the user can decide how to discharge or charge the Cell Under Test (CUT), by defining the current magnitude and dropout voltages with specific commands. However, these instruments are characterized by power limits. In particular, the electronic load sustains a continuous maximum power of 400 W and the charger provides a continuous maximum power of 1200 W. Therefore, the user is asked to define the appropriate programming command parameters in each experiment, according to the power curves of each lab device, as shown in Figure 2.

Generally, the force block units work in two different modes. First, the units are set in Constant Current (CC) mode and force a constant current to the CUT. The current flowing in the CUT produces a terminals voltage increment or decrement according to the kind of test carried out. This operating mode is maintained until the CUT terminal voltage reaches the dropout voltage value. At this point, the lab units enter in Constant Voltage (CV) mode and fixes the voltage at the CUT terminals. Then, the current flowing in the CUT gradually decreases and the test finishes when the current value falls under a given threshold. However, fully customized test procedures can be carried out, as it will be shown afterword.

The two lab units are connected to the CUT through two separate paths. The first is a "force path" that sustains the high currents flowing in the cell and it is composed by 6AWG power line cables and

two power relays LEV100A4ANG. In particular, the relays withstand a maximum continuous current of 100 A and a maximum open contact voltage of 900 V. The second is a "sense path" that starts from the rear panel of the two lab units and directly contacts the CUT terminals, to implement a 4-wire voltage measurement. In this way, the voltage drops over the cables and the contact resistances of the power path do not impair the cell voltage measurement. The two lab units also make available the voltage values internally measured, but we considered the resolution provided by the electronic load insufficient for a reliable characterization of a battery cell.

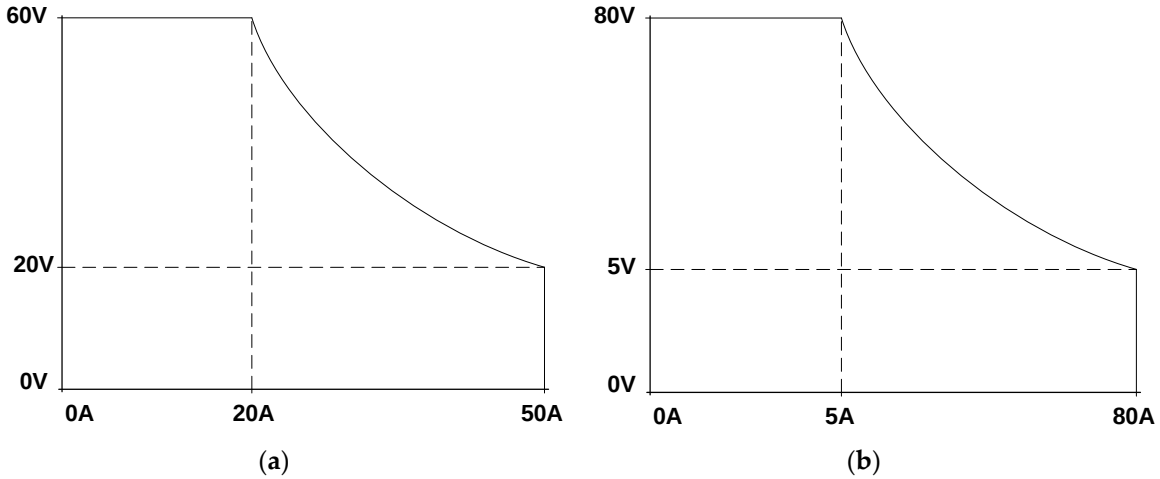


Figure 2. V-I power flex plot for: (**a**) QPX1200SP and (**b**) LD400P.

Therefore, we decided to add to the platform a dedicated acquisition device, the programmable DAQ Picolog ADC-24. Picolog represents the main unit of the "measure block". The DAQ constantly acquires the CUT terminal voltage, the current flowing in the force path through a high precision shunt resistor, and the temperature of the CUT by means of a LM35 sensor attached to the cell case that yields a 0.5 °C degree accuracy. In particular, the 4-wire high precision shunt resistor has 100 A/100 mV sensitivity, ±0.25% resistance accuracy and 15 ppm/°C temperature sensitivity. Thanks to the presence of some GPIO pins, the Picolog ADC-24 can set and read the state of the two power contactors on the force path. Should the read and set states of the relays not match, a sign of relay fault, the test execution would be safely aborted.

A custom Printed Circuit Board (PCB) was designed by means of the open-source electronic design automation environment KiCAD [22] to successfully insert the Picolog ADC-24 in the CST framework. First, the PCB extends the power capability of the GPIO pins to control the power contactors through a 12 V external power supply and a digital control circuit. Second, the cell's terminal voltage is reduced by a voltage divider to adapt it to the input range of the Picolog ADC-24 analog channel. Finally, the PCB allows the Picolog ADC-24 to read the state of the contactors and to send it to the control unit. As above-mentioned, the test execution can be aborted if the state of the contactors does not match with its expected setting.

Finally, the "control block" properly manages the units of the CST according to the test phase to be executed. The hardware framework explained so far is akin to a basic Li-ion cell experimental characterization equipment [10,23], except for the control unit. The control unit consists of a Personal Computer in all the testers described in the above-cited literature. The PC Operating System (OS) goal is to make the computer easy to use, by providing services to the user. For instance, a Graphical User Interface (GUI), a multiple user program execution, I/O operations, file-system manipulation, network and inter-process communication are services typically provided by the OS. The application software that controls the test thus allows the operator to customize the test protocol according to the different Li-ion technologies, to follow the test progresses over a GUI and to process the log-file generated during the test phases by means of the preferred software application.

However, a new concept of computers has recently emerged, as indicated in [24]. This new concept is known as Single Board Computer (SBC). An SBC is smaller and cheaper than a classic computer. The cost and dimension of an SBC are comparable to a Micro Controller based evaluation board, but the support of an OS, the presence of USB, Ethernet and HDMI peripherals and the rather good computational power make SBCs appealing in many applications. One of the most popular SBC devices is the Raspberry Pi [25]. Indeed, it is used in many research works, as in a distributed system for pollution monitoring on a smart city [24], in image processing [26,27] and educational [28] fields.

For these reasons, we decided to use a Raspberry Pi 3 model B as the control unit of our tester. It supports the Raspbian open OS, in which the user application controlling the entire tester runs. The features of the developed user application will be explained in the next sub-section. An appropriate USB-driver was installed on the Raspbian OS to manage the Picolog ADC-24, whereas a LAN network infrastructure was set to control the LD400P and QPX1200SP. The LAN framework consists of a star architecture where the peripheral nodes are the Raspberry PI, LD400P and QPX1200SP and the central node is a Netgear GS605v5 ethernet switch.

2.3. Software Framework

Figure 3 shows that the user application is a multi-process program: one is written in Python language and the other is written in C/C++. Python is developed under an OSI (Open Source Initiative) approved open-source license [29], and it is particularly suitable to develop software applications on a Raspberry Pi board [28]. Therefore, it fits particularly well to our development scope.

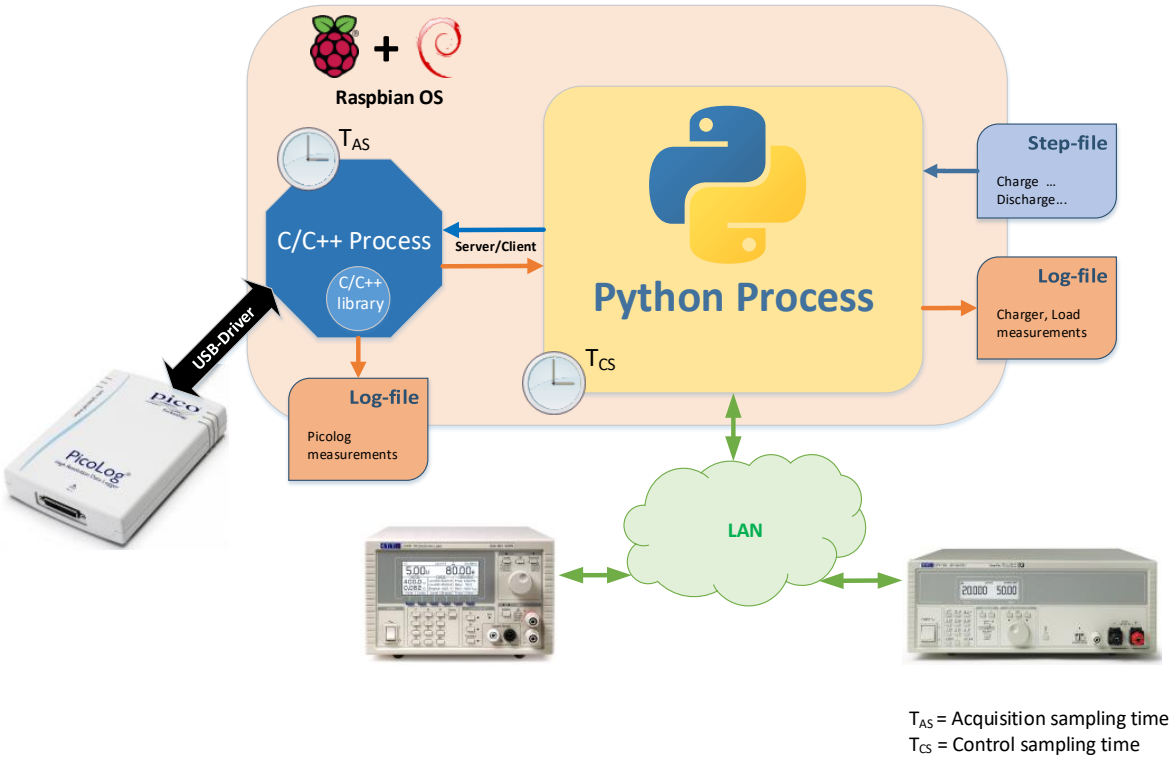


Figure 3. Software framework of the Cell Station Tester.

The Python process is the main software process of the application. It controls the lab instrumentation (i.e., load, charger and DAQ) according to the commands defined by the operator in the "Step-file", manages the system safety, controls the execution of the test phases and saves the data acquired from the electronic load or the charger in a log-file. Instead, the secondary C/C++ process allows the Python one to control the data acquisition device by means of a specific Application Programming Interface (API) released by PicoTech, which produces actions on the Picolog device by using the USB interface. Moreover,

the C/C++ process creates a file that logs the Picolog acquired data with a specific and independent acquisition rate.

The Python process sends specific commands to the programmable "force-block" units and the secondary C/C++ process to control the entire environment, according to the server/client paradigm. Here, the Raspberry Pi represents the host client, while the Electronic load, the charger and the secondary process are the host servers. The host client sends the requests to the host servers, and the host servers respond with valid data or with an error flag. In the last case, the Python process enters the error state and aborts the test execution.

It is important to point out the relationship between the system clock and the sampling times used by the Picolog device and the Raspberry Pi, which are different to each other. Let us define the *acquisition sampling time* as the time that elapses between two consecutive data acquired by the Picolog, and *control sampling time* the time that elapses between two data internally acquired by the electronic load and the charger. The *acquisition sampling time* is fixed at 0.5 s in the actual release of the software application and is referred to the Picolog internal clock. The *control sampling time* can be set from 0.5 s to 2 s by the operator and is referred to the Raspberry Pi internal clock. The latter is also the rate with which the Python process executes periodic tasks. A separate acquisition and control rate may be a useful feature when the application requires different control and acquisition sampling times. For example, a high control sampling time would be useful when the cell status needs to frequently be checked for safety reasons, while the data would be acquired at a lower rate to reduce the dimension of the test data.

After the explanation of how the inter-process communication and lab instrument management are done and before showing the main process actions, it is worth to explain how the user customizes the tests according to the Li-ion technology of the cells to be characterized and the desired test protocol. The operator describes the test protocol to be executed in a specific text file named "Step-file". This file adopts a structure very similar to that used in [30]. The file consists of lines of comments or commands. Comments are preceded by the character # and are ignored during the Python process execution. Besides, command lines produce actions controlling the lab instrumentation.

Figure 4 shows an instance of a "Step-file". Here, the structure of the command lines is discussed. Each record contains seven fields that are used to define a single step-test operation:

```
#full charge
charge          1       1       -1      20      3.6     0.1

#stop
measure         1       1       7200    0       0       0

#discharge 2.5%
discharge       1       1       90      37      2.8     0.1
```

Figure 4. Example of Step-file syntax.

Operation type that assumes the values: *charge*, *discharge* or *measure*, to define a charging, a discharging or a rest phase;

- *log enable* that assumes 0 or 1 to disable or enable the log of any available quantity during the test;
- *control sampling time* in seconds (from 0.5 s to 2 s);
- *test length* in seconds, where -1 value means that the elapsed time will not be considered to stop the actual step;
- *constant current amplitude* in ampere, with which the cell under test is charged or discharged;
- *dropout voltage value* at which the CV mode of the lab instruments starts;
- *stop current amplitude* that defines the threshold value under which the CV mode is ended.

Therefore, a fully customizable test process can be described and executed by listing the proper sequence of commands, each one with the appropriate parameters. For example, a pure CC test is

programmed by posing the *stop current amplitude* equal to the *constant current amplitude*. In any case, any action or experiment performed on the CUT must maintain the cell inside its Safe Operating Area (SOA) to avoid the occurrence of critical or dangerous issues [31]. The SOA is defined in term of voltage, current and temperature. The first and the second can be set by means of the *dropout voltage* and *constant current amplitude* in the command lines of the Step-test file, whereas the last one is stored in another configuration file. The configuration file is the "picologinfo.txt" in which the map of the analog and digital pins of the Picolog ADC-24, the sensitivity of the temperature sensor, the conversion information about the ADC-channels, as well as the allowed temperature range of the CUT are specified. Should the measured data exceed the SOA, the test is immediately interrupted, and a safe state is reached.

The Python process execution can be divided in three major phases: Initialization, Acquisition, and Termination, as described in the flowchart of Figure 5.

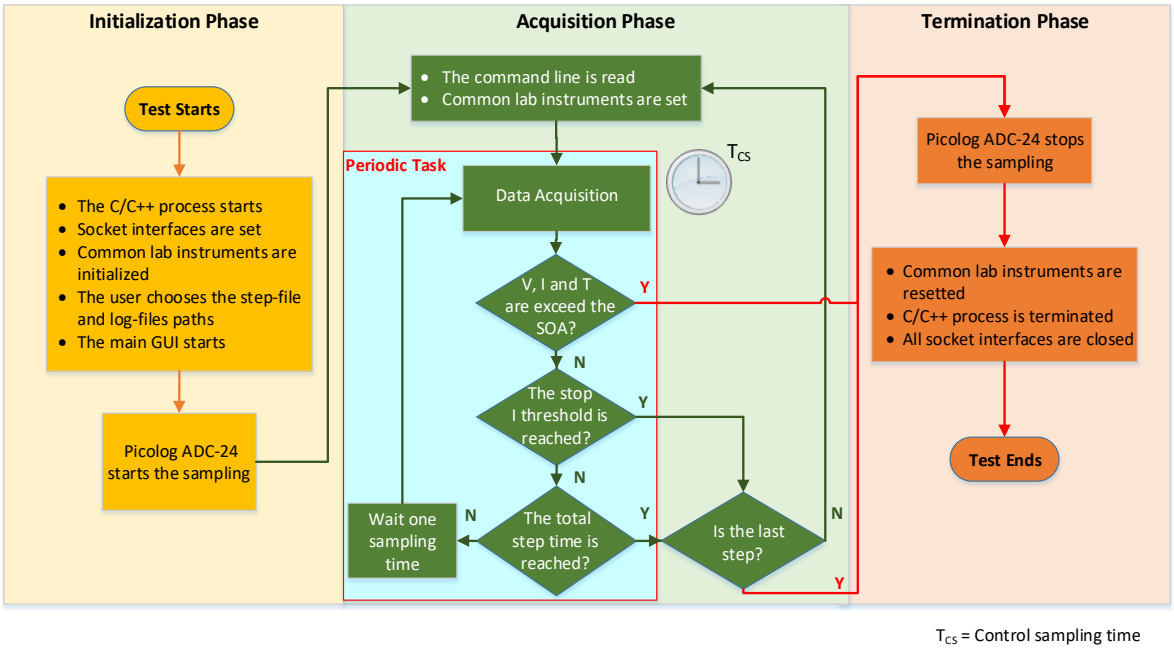


Figure 5. Flowchart of the software process written in Python.

In the Initialization phase, the process creates the socket interfaces with the LD400P and QPX1200SP, starts the C/C++ process and opens the inter-process socket interface, resets the lab instruments and brings them to a safe initial state, reads the step-file, holds the decoded commands in a software data structure and creates a Graphical User Interface (GUI). Finally, the Python process starts the Picolog ADC-24 sampling operations.

The Acquisition phase represents the main phase of the test process. The test commands saved in the data structure are processed and transformed in actions on the CUT. Here, the Python process loads the parameters of the current command, properly configures the instrumentation and the state of the relays and starts a periodic task. This task is implemented using the "apscheduler" Python module. It periodically acquires the voltage and current measurements from the "force-block" units, together with the CUT temperature and the relay states from the Picolog each *control sampling time*. The voltage, current and temperature values are then verified to be inside the cell SOA limits. The task controls if the stop conditions due to either the elapsed time or the current threshold are met. If neither condition is satisfied, the task waits the next acquisition time (max 2 s in the present release), according to the *control sampling time* parameter value. If either one of the two stop conditions is recognized, the process closes the log-files created during the execution of the step and proceeds to the next command, if any. Otherwise, the process enters the Termination phase. Instead, the test procedure is immediately

terminated if one of the voltage, current or temperature values exceed the SOA limits, to avoid any safety hazard.

During the Termination phase, the tester reaches a safe state and properly ends the test execution. For this purpose, both power contactors are switched off, the two instruments units are reset, and the Picolog ADC-24 sampling is ended. Then, all the communication socket interfaces are closed, and finally, both the software processes terminate their execution. In any case, if the Python process terminates because of a hardware or software error or the onset of a SOA exceeded alert, an error message identified by a code is shown on the Raspian OS LXTerminal which will be described in the next subsection.

2.4. Graphical User Interface

The interaction between the user and the application is managed by an appropriate GUI.

Several windows are open on the GUI during the tests to show information and to acquire the user actions. First, an LXTerminal is opened when the software starts. LXTerminal is a terminal emulator component inside Lightweight X11 Desktop Environment (LXDE), which is an open-source desktop environment for Unix-like system. The LXTerminal window informs the user on the software application status and requires the user to carry out some actions when needed, as shown in Figure 6. Any time the path to a file is required (e.g., the Step-file that describes the test sequence, the log-file where the data coming from the DAQ are stored, or the log-file where the data coming from the charger and load devices are stored), the software application opens an appropriate window.

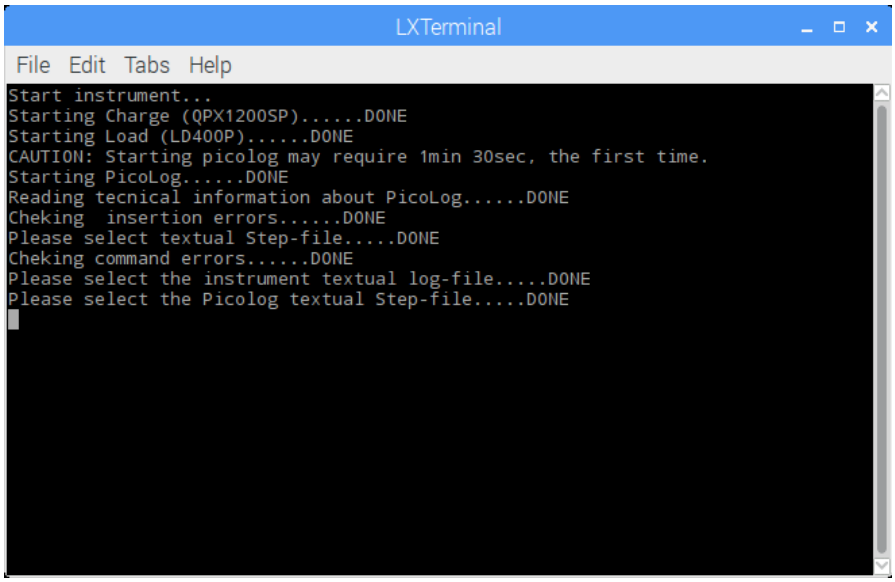


Figure 6. The LXTerminal window that acts as console of the application software during its execution.

When the tester is properly initialized, the software application periodic task described above starts, and another window opens. As shown in Figure 7, the user can monitor the command line that is now in execution, the battery cell data in term of voltage, current and temperature, the quantity of charge extracted or inserted in the CUT and the step progression. Furthermore, two light indicators show the test status. In particular, the "running?" light indicates if the software application is running or not with green or red color, respectively. The "allright?" light indicator shows if an error occurred in the same way.

If an error occurs durign the test execution, the software application enters in the termination phase, turns in red both the light indicators, and shows on the LXTerminal console the code of the error that triggered the abnormal test termination. If all the steps of the test are executed without errors, the program terminates normally, by switching to red the "running?" light only and showing

a positive conclusion message on the LXTerminal console. Finally, it is worth to mention that all the windows beside the LXTerminal console are designed using the "Tkinter" Python module.

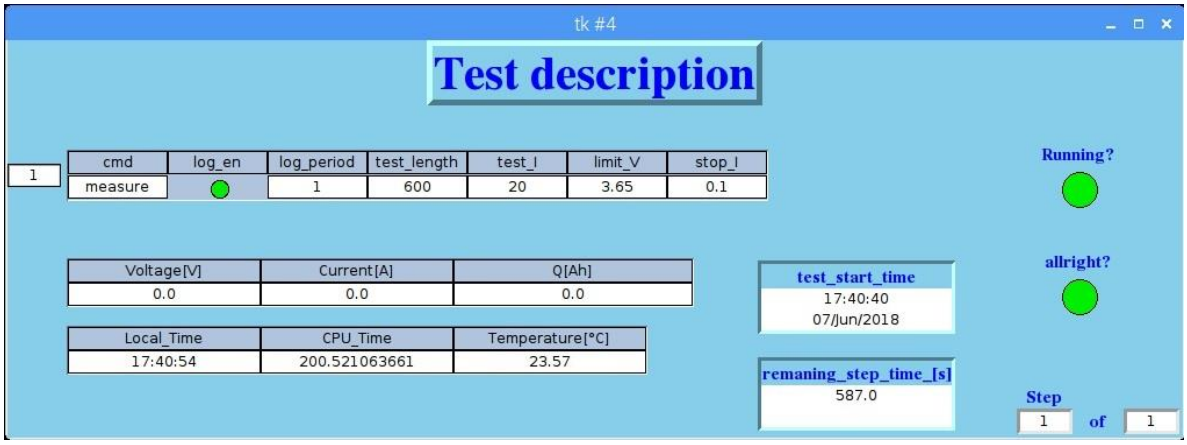


Figure 7. The window opened while the software application is executing the periodic task.

3. Cell Station Tester Validation Results

This section shows the results obtained from the experiments carried out on the proposed CST to validate its functionality. The validation of the experimental setup shown in Figure 8 was divided in three parts. First, we analyzed the repeatability of the *control sampling time* to characterize the time reliability of the system, as Raspbian is not a real-time operating system. Second, we calculated the CST voltage and current measurement accuracy, in order to demonstrate the functionality of the tester and to evaluate its performance. Then, we performed a characterization experiment on a second-hand battery pack used in an electric scooter to extract the SoH of the cells for a possible battery pack refitting. The pack was composed by twenty prismatic CALB CA40 $LiFePO_4$ cells. Each cell has a nominal capacity of 40 Ah, 2.5 V and 3.65 V cut-off voltages, a charging and discharging maximum temperature of 45 °C and 55 °C and a maximum charging and discharging current of 40 A and 80 A, respectively. The aim of this last experiment is just to show a simple example of test that our open-source cell tester can carry out. More complex experiments, such as Pulse Current Tests (PCT) used to extract the relationship between Open Circuit Voltage (OCV) and State of Charge (SoC), could easily be performed by customizing the Step-test file.

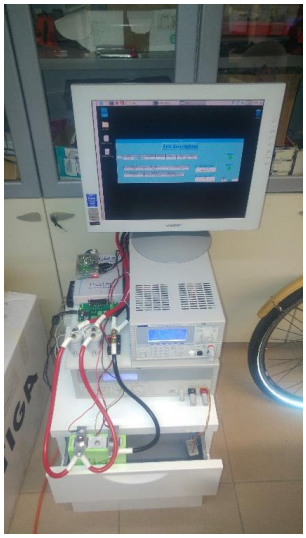
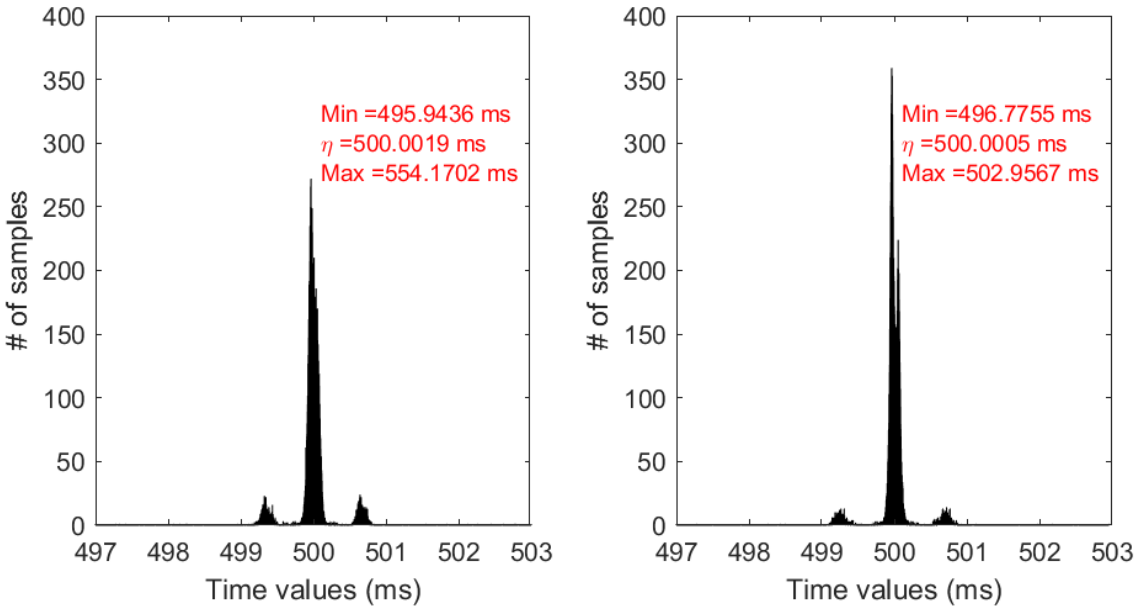


Figure 8. Cell Station Tester experimental setup.

3.1. Timing Reliability

The first test aim is the evaluation of the time reliability of the periodic acquisition phase. The duration of each occurrence of the periodic task was measured with the Python-module function *perf_counter()*, which returns the system clock of the Raspberry Pi any time the function is called. The function was called at the beginning of the task, so that the difference between two consecutive values gives the exact task duration. The test consists of five charging and five discharging steps, in which the charger and the electronic load are used, respectively. The *test length* and the *control sampling time* of each step are set to 3600 s and 0.5 s, respectively.

Figure 9a,b shows the distributions of the control sampling time as measured during the discharge and charge experiments, respectively. The sampling time is not constant, as Raspbian is not a real-time OS and it does not guarantee the exact periodicity of a given task, which is scheduled together with other background processes. However, the values are strongly distributed around an average duration of 500 ms, as set in the Step-test file, with two tiny side lobes. Furthermore, the experiment demonstrates that the variability of the control sampling time does not depend on the kind of instrument used (i.e., charger or electronic load). Indeed, the distributions obtained are rather uniform from step to step and a longer interval is balanced on average by a shorter one. The lobes do not exceed 1 ms from the mean value.



(a) (b)

Figure 9. Control sampling time rate distributions during (**a**) discharging and (**b**) charging phases.

3.2. Voltage and Current Measurement Accuracy

According to the definition of accuracy in ISO-5725-1, we extracted the terms "trueness" and "precision" for both voltage and current measurements. Trueness is the distance between the mean value of a large number of measurement results and the reference value to be measured. Precision represents the deviation of the large number of measurement results around its mean value.

To extract the precision and trueness of our CST, at least in one given point, we measured a fixed voltage 600 times, with a 0.5 s sampling time, at room temperature. The voltage source was a completely relaxed CA40 cell. This source can reasonably be considered reliable and stable, at least during the measurement time, so that the uncertainties can be ascribed to the CST only. The probability density

function (pdf) of the results was then evaluated. The standard deviation σ of the pdf distribution defines the measurement precision according to equation (1).

$$\text{Precision} = 6\sigma \tag{1}$$

Instead, the mean value of the pdf gives the averaged result, from which the trueness of the instrument can be calculated, provided that a more accurate measurement of the same source is available for comparison. An Agilent 34401A digital meter was used as reference instrument, so that it was connected to the same voltage source to simultaneously carry out the same set of measurements. The Agilent meter averaged output was finally compared to the mean value measured by the CST. Figure 10 shows the pdfs of the 600 voltage measurements with a bin dimension of 1 µV, carried out by the CST and Agilent meter, respectively. The shape of both pdfs resembles very well a Gaussian one. The red lines represent the Gaussian fitting of the distributions, from which the standard deviation and mean values are extracted. First of all, it should be noted that the precision (standard deviation) of the instrument we chose as reference (Agilent meter) is one order of magnitude better than the CST, so it can be considered as a good reference. Then, the precision achieved by the CST is $6\sigma = 8.94 \times 10^{-5}$ V, a rather good value, fully suitable to Li-ion cell characterization. Finally, the CST measured voltage value is 3.303 V, to be compared to the reference value of 3.295 V, for a difference of only 8 mV (i.e., 0.24%).

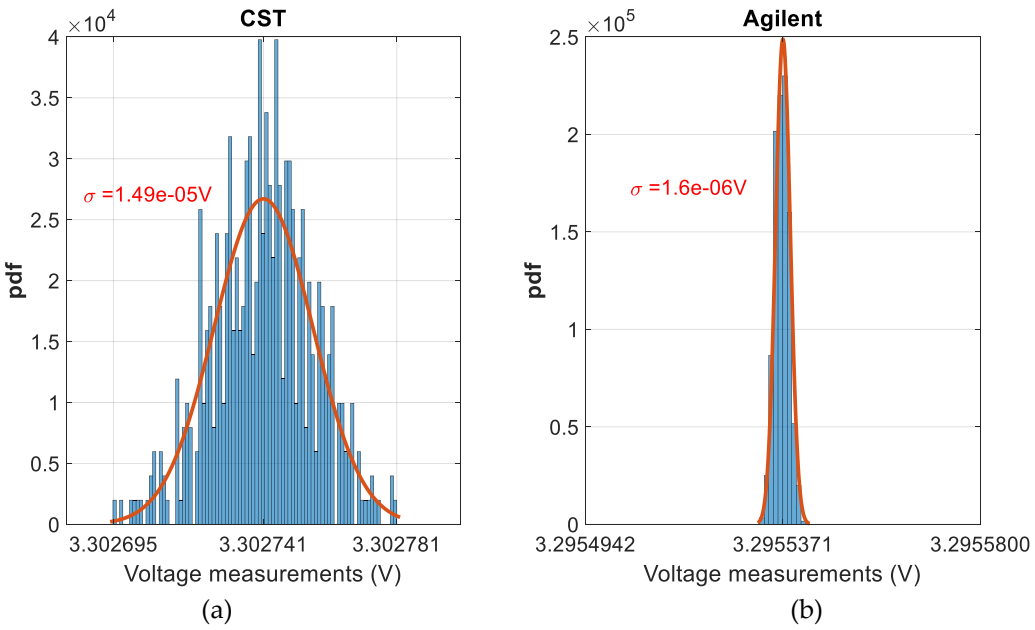


Figure 10. Pdf of 600 measurements carried out over a relaxed CA40 cell by: (**a**) our Cell Station Tester (CST); (**b**) an Agilent 34401A digital meter used as reference instrument. The mean values and standard deviations of the approximating Gaussian functions are shown on the diagrams.

Then, a Keithley 2460 sourcemeter was programmed to generate ten voltage values between 0 and 4.5 V with 0.5 V steps and was connected to the input voltage channel of both the reference instrument (Agilent meter) and the CST, in order to characterize the CST trueness on the entire input range. As the sourcemeter precision in generating a fixed voltage is comparable to the CST precision in measuring it, the experiment was carried out to evaluate the trueness only. The voltage value generated by the sourcemeter was simultaneously sampled 600 times with a 0.5 s sampling. Finally, the mean values of the readings coming from the two instruments were calculated for each input step and compared to each other, to evaluate the CST trueness errors on the full input scale. Let us define V_{CSTi} the mean value measured by the CST and V_{REFi} the reference value measured by the Agilent meter for each *i*-th

step. The voltage absolute error $E_{V\mathrm{i}}$ is the difference between the two values and is expressed by (2) for each step.

$$E_{V\mathrm{i}} = V_{\mathrm{CST}i} - V_{\mathrm{REF}i} \tag{2}$$

Let us also calculate the linear regression best fit $E_{\mathrm{V}}(V_{\mathrm{REF}})$ of the error values, as expressed in (3), which is characterized by an offset Off_V and a gain G_V.

$$E_V(V_{\mathrm{REF}}) = Off_V + G_V V_{\mathrm{REF}} \tag{3}$$

Figure 11 shows the linear regression fit of the $E_{V\mathrm{i}}$ points. It is noticeable that the introduced error has an offset of about 2.15×10^{-5} V and a gain of about 2.08×10^{-3}.

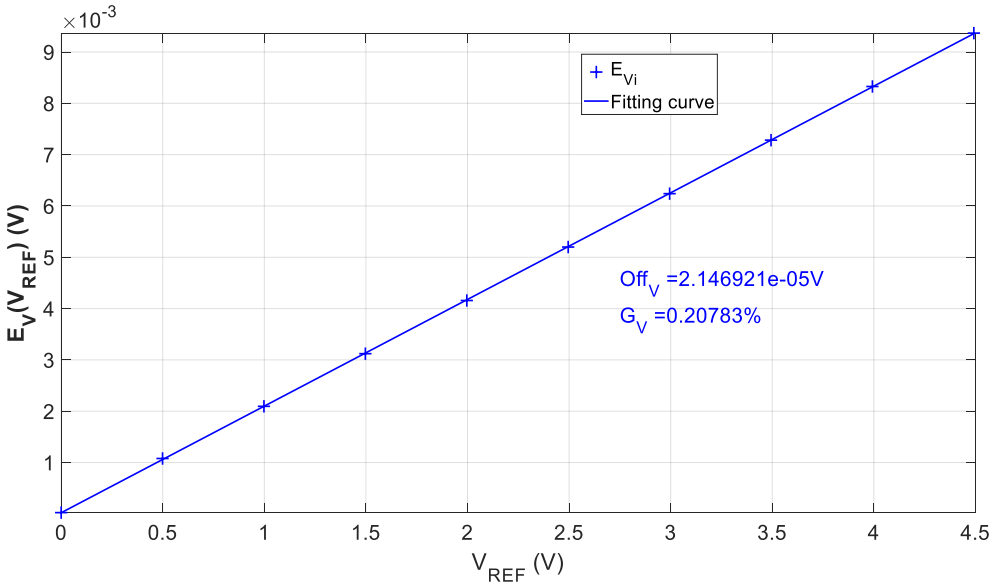


Figure 11. Linear regression function that best fits the voltage absolute error over ten values to determine the CST "trueness" on the entire input range.

The same procedure that allowed us to determine the precision and trueness of the CST voltage measurements was repeated for the current input channel. The first experiment was carried out by shorting the current input terminals, to measure a zero-current value, and thus to determine the CST offset. Other 600 experimental points were collected, from the Gaussian fitting of which the CST precision is derived, as shown in Figure 12a, where the pdf with a bin size of 0.15 mA is reported. The precision value results to be $6\sigma = 8.64 \times 10^{-3}$ A. Instead, the current mean value is -3.158 mA, that is the offset value introduced by the CST when the input current is zero.

As the CST measures the current as a voltage drop over a 1 mΩ shunt resistor, the shunt resistor was removed from the current inputs and the sourcemeter was used to generate fourteen voltage values between -80 mV to 50 mV with 10 mV steps. This configuration emulates a current ranging from -80 A to 50 A. The same procedure described above to evaluate the voltage trueness was repeated for the current. The mean values of the readings coming from the two instruments $I_{\mathrm{CST}i}$ and $I_{\mathrm{REF}i}$ for each i-th step were evaluated and compared. The current absolute error $E_{I\mathrm{i}}$ is the difference between the two values expressed by equation (4), whereas the linear regression function is described in (5), where Off_I and G_I are the offset and gain parameters of the function.

$$E_{I\mathrm{i}} = I_{\mathrm{CST}i} - I_{\mathrm{REF}i} \tag{4}$$

$$E_I(I_{REF}) = Off_I + G_I I_{REF} \tag{5}$$

Figure 12b shows the linear regression fit of the E_{Ii} points. It is worth noting that the introduced error has an offset of about -1.387×10^{-3} A and a gain of about 7.54×10^{-4}.

Finally, Table 3 summarizes the CST characteristics by reporting the maximum values of the precision and the trueness achieved in the input voltage and current ranges. If we compare the precision and trueness values with the voltage and current dynamic ranges supported by the proposed CST, we end up with very satisfactory results, suited to the characterization of Li-ion cells. The observation of the experimental results suggests that the trueness error is mainly due to a gain error. Therefore, an appropriate calibration procedure that corrects the gain error would make it possible to further reduce the absolute error and improve the accuracy of the tester. The calibration procedure is enclosed as Supplemental Material of the paper.

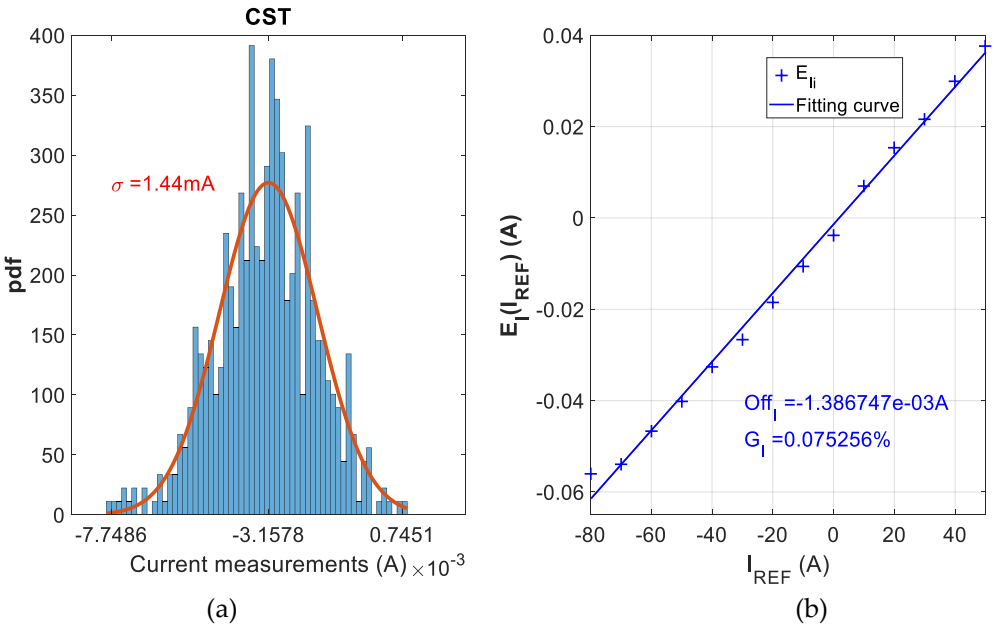


Figure 12. (**a**) pdf of a set of values measured on the current input channel of the CST when shorted; (**b**) absolute error linear fit over 14 values to determine the CST "trueness" on the entire input range.

Table 3. CST voltage and current measurement accuracy.

Quantity	Precision	Max. Absolute Trueness
Voltage	8.94 µV	9.37 mV
Current	8.64 mA	61.59 mA

3.3. SoH Measurements of a Used Battery Pack

Let us now detail the experiment that shows, as a simple example, how the CST described in this paper was used to measure the SoH of the cells belonging to an E-scooter second-hand battery pack. First, let us illustrate the test protocol adopted to calculate the deliverable capacity of a brand-new CA40 cell. Figure 13 shows the Step-file described in Section II that implements the test sequence, which consists in a full charge and discharge cycle. The sequence is constituted of three different phases:

- a charging phase, where the cell is fully charged with a 20 A current;
- a discharging phase, where the cell is completely discharged with a 40 A current;
- a charging phase, where the cell is returned to the full-charge state with a current of 20 A.

```
#Full charge
charge          1       0.5     -1      20      3.65    0.01

#Full discharge
discharge       1       0.5     -1      40      2.8     0.01

#Full charge
charge          1       0.5     -1      20      3.65    0.01
```

Figure 13. The test sequence adopted to measure the available charge in a cell, as described in the Step-file controlling the CST.

The first and last steps were only used to bring the cell to a known initial state, whereas the second step was used to calculate the capacity of the CUT. The total cell capacity was calculated by integrating the discharging current during the test (i.e., Coulomb counting). The lower and upper cut-off voltages and the end-current threshold were set to 2.8 V, 3.65 V and 10 mA, respectively. The sampling time was fixed to 0.5 s.

Figure 14 shows the cell voltage and current during the three phases, as measured by the CST. Each step ends when the cell voltage reaches the upper or lower cut-off value and the current falls under the established threshold. The available CUT charge, as measured by the CST, results to be 37.012 Ah. This value is lower than the manufacturer rated one. However, the test by which the cell manufacturer measures the nominal capacity of a new cell is slightly different than the test adopted here [32], so finding a difference is reasonable. In fact, we used a higher discharging current to reduce the test duration and a higher low cut-off voltage to satisfy the safety indication of the cell manufacturer [32]. Both the higher current and cut-off voltage produce the effect of reducing the deliverable capacity of the cell because of the Peukert's law [33] and the more limited cell voltage dynamics.

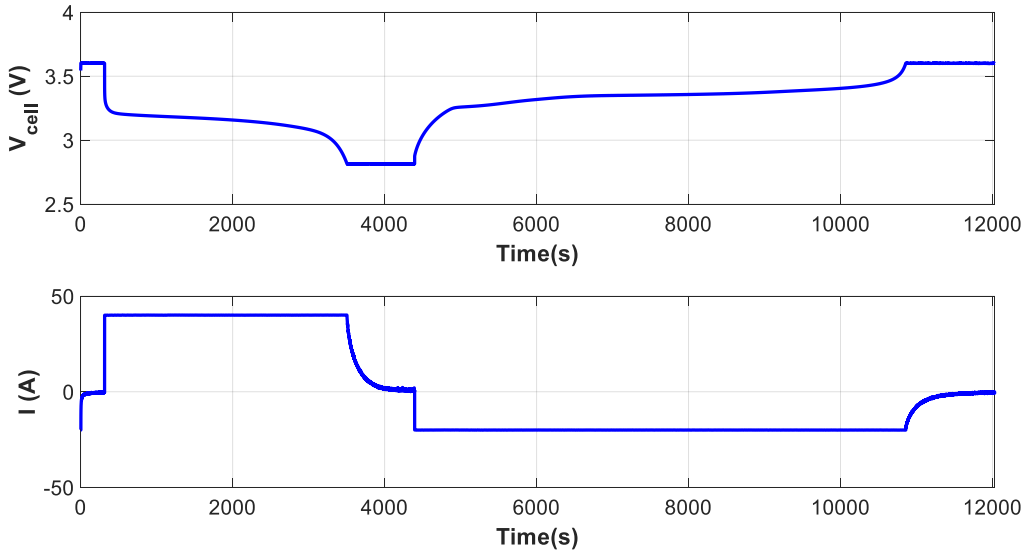


Figure 14. Cell voltage and current during the three phases experiment to measure the cell capacity.

The same test procedure was then applied to extract the deliverable capacity of twenty CALB CA40 cells coming from a second-hand LIB pack used in an E-scooter. Bearing in mind the reference value measured on the brand-new cell, we finally measured the SoH of each cell according to Equation (6).

$$SoH = \frac{\text{Residual Capacity of sec ond} - \text{hand cell}}{\text{Capacity of new cell}} \cdot 100 \quad (6)$$

Figure 15 shows a bar plot where each column represents the SoH of each cell of the battery pack. It is worth noting that the health degradation was not the same for each cell of the pack. This result can be ascribed to cell-to-cell variations in term of internal resistance, different working temperatures inside the pack, and other mismatch factors that are characteristic of the manufacturing process and pack configuration [34]. Nevertheless, the results show that most of the cells are around 90% of SoH. Instead, the cell number 8 reaches 80% of SoH, approaching the end-of-life point usually considered for mobility applications. The characterization test suggests the substitution of this weak cell with another one to prolong the use of the battery pack.

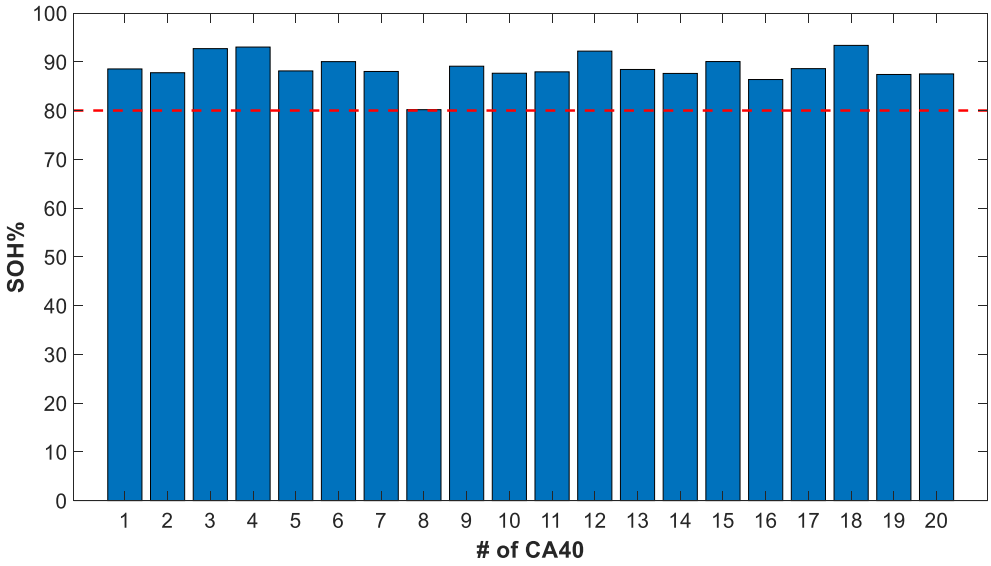


Figure 15. SoH of twenty CALB CA40 cells used in an E-scooter.

4. Discussion and Conclusions

The experimental results described in the above Section demonstrate that the proposed CST is capable of characterizing Li-ion cells with reliable sampling timing rate and good current and voltage measurement accuracy. A simple example (SoH determination) of how the developed tester can be used was also described. In particular, the SoH measurement results showed the capability of our CST to evaluate the degradation of each cell within a battery pack used in E-scooter application. As the cells were not aged at the same rate, the CST was useful to identify the most degraded cells of the battery pack during a maintenance service event. However, other more complex test protocols, for instance, PCT tests, could easily be programmed and performed. Indeed, the flexibility of the tester allows an operator to customize the test to any kind of Li-ion cell, according to the chemistry used and the manufacturer specifications.

Another valuable feature of the proposed tester is that the rate at which the test is controlled is different from the data acquisition rate. It is thus possible to control the test execution and the cell status at a rate different than the data collection rate. This feature could be useful for long tests at low discharge rates when the data acquisition time is large to avoid an excessive size of the log files but, at the same time, a high time resolution is needed to detect the exact time when a threshold is reached. This feature will fully be exploited in a future release of the application software.

Besides the advantages due to the use of common laboratory instruments, the tester proposed in this work introduces two valuable novelties, if compared to other works. The first is using the Python language to create the application-software that manages the programmable lab instruments. The seconds is the adoption of a Raspberry Pi board as main control unit in place of a PC. The Raspberry Pi 3 reduces the cost of the tester, without losing any features that a PC usually provides, for instance, a proper GUI, as well as communications and file system services.

Generally, the application software is designed using the LabVIEW graphical language running on a PC. In fact, many lab instruments are released with specific LabVIEW drivers for their integration. The software designer task is simplified, and the development time is reduced, but the use is permitted only after buying a proper license. Furthermore, the cost of the license can change according to the type of OS where the application is developed. Instead, Python is open and portable, so the software developer can use it in different OS without any license restrictions, and it is also compatible with some SBC devices. On the other hand, Python is a text-based programming language, so the creation of a new Python application is more time-consuming with respect to using LabVIEW. However, the Python source code of the application can freely be shared in a public domain, fostering increased creativity in the user community that can modify and adapt the basic code to the individual purpose and accelerate bug finding and fixing.

The Python and C/C++ process codes have utilized the proprietary API functions to control the external instruments (i.e., the Picolog ADC-24, LD400P and QPX1200SP), in this software release. A software developer that wants to modify our source code and adapt it to different instrument units should replace the proprietary APIs used in the codes with new ones pertaining to the other set of instruments used. To simplify this process, we are working on a second release of the software. Here, we will add a driver infrastructure that uses appropriate "driver-files", where general purpose APIs are associated with the specific APIs for the instruments used. In this way, the developer is not forced to change all the APIs in the code of the application software, but instead, he will only change the "driver-file".

The source code of the current version of the CST software application, together with Supplemental Material, is released to the public domain in accordance with the open-access philosophy.

In conclusion, this work has shown the architecture and the experimental characterization of a low-cost tester for battery cells. It consists of common lab instruments connected together and managed by an open-source software application written in Python and running on a Raspberry Pi board. The tester has been experimentally characterized to derive its precision and accuracy that are suitable for Li-ion cell characterization. An application example of a cell screening test performed on an E-scooter used battery pack has been described. The example has shown the capability of the tester to easily determine the SoH of the cells and to suggest the appropriate maintenance procedure. However, the tester is flexible and can be programmed to perform more complex experiments.

Supplementary Materials: The following are available online at http://www.mdpi.com/2079-9292/7/12/454/s1. The user manual of the CST, the calibration procedure, the source code of the software applications and the hardware architecture described in this paper are available on the web-site: https://github.com/batterylabunipi/TWIST.

Author Contributions: Instrument conception and design, A.C., R.R.; Software, A.C., R.D.R.; Validation, R.D.R., F.B.; Accuracy analysis, A.C., F.B., R.S.; Writing–First Draft Preparation, A.C.; Writing–Revision, R.R., F.B., R.D.R., R.S.

Funding: This work was partly supported by the Project "Urban Districts with zero environmental and energetic impact" financed by the University of Pisa under grant PRA_2017_33.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tarascon, J.M.; Armand, M. Issues and challenges facing rechargeable lithium batteries. *Nature* **2001**, *414*, 359–367. [CrossRef] [PubMed]
2. Rechkemmer, S.K.; Zhang, W.; Sawodny, O. Modeling of a Permanent Magnet Synchronous Motor of an E-Scooter for Simulation with Battery Aging Model. *IFAC-Pap. Online* **2017**, *50*, 4769–4774. [CrossRef]
3. Nitta, N.; Wu, F.; Lee, J.T.; Yushin, G. Li-ion battery materials: Present and future. *Mater. Today* **2015**, *18*, 252–264. [CrossRef]
4. Mathew, M.; Kong, Q.H.; McGrory, J.; Fowler, M. Simulation of lithium ion battery replacement in a battery pack for application in electric vehicles. *J. Power Sources* **2017**, *349*, 94–104. [CrossRef]
5. Mathew, M.; Janhunen, S.; Rashid, M.; Long, F.; Fowler, M. Comparative Analysis of Lithium-Ion Battery Management Systems. *Energies* **2018**, *11*, 1490. [CrossRef]

6. Podias, A.; Pfrang, A.; Di Persio, F.; Kriston, A.; Bobba, S.; Mathieux, F.; Messagie, M.; Boon-Brett, L. Sustainability assessment of second use applications of automotive batteries: Ageing of Li-ion battery cells in automotive and grid-scale applications. *World Electron. Veh. J.* **2018**, *9*, 24. [CrossRef]
7. Weßkamp, P.; Haußmann, P.; Melbert, J. 600-A Test System for Aging Analysis of Automotive Li-Ion Cells with High Resolution and Wide Bandwidth. *IEEE Trans. Instrum. Meas.* **2016**, *65*, 1651–1660. [CrossRef]
8. Honda, "EV-Neo.". Available online: http://world.honda.com/EV-neo/spec/index.html (accessed on 28 October 2018).
9. Dahmane, Z.E.; Malek, A.; Bouhali, M.; Bounabi, M.; Kaced, K.; Cheikh, M.S.A. A proposed pulses current method to extract the batteries parameters. In Proceedings of the 2017 6th International Conference on Systems and Control (ICSC), Batna, Algeria, 7–9 May 2017; pp. 555–560.
10. Vergori, E.; Mocera, F.; Somà, A. Battery Modelling and Simulation Using a Programmable Testing Equipment. *Computers* **2018**, *7*, 20. [CrossRef]
11. Chen, Z.; Wang, L.Y.; Yin, G.; Lin, F.; Wang, C. Accurate probabilistic characterization of battery estimates by using large deviation principles for real-time battery diagnosis. *IEEE Trans. Energy Convers.* **2013**, *28*, 860–870. [CrossRef]
12. Jiangsu Niu Electric Technology. Niu Home Page. Available online: https://www.niu.com/en (accessed on 28 October 2018).
13. Jespersen, J.L.; Tønnesen, A.E.; Nørregaard, K.; Overgaard, L.; Elefsen, F. Capacity Measurements of Li-Ion Batteries using AC impedance Spectroscopy. *World Electr. Veh. J.* **2009**, *3*, 127–133. [CrossRef]
14. KIKUSUI, BPChecker2000 License. Available online: https://www.kikusui.co.jp/en/download/en/licence.html (accessed on 28 October 2018).
15. NATIONAL INSTRUMENTS, LabVIEW General Purpose Software License Agreement. Available online: https://www.ni.com/legal/license/ (accessed on 28 October 2018).
16. Ciolli, M.; Federici, B.; Ferrando, I.; Marzocchi, R.; Sguerso, D.; Tattoni, C.; Vitti, A.; Zatelli, P. FOSS Tools and Applications for Education in Geospatial Sciences. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 225. [CrossRef]
17. GNU Operating System Home Page. Available online: https://www.gnu.org (accessed on 28 October 2018).
18. GNU General Public License. Available online: https://www.gnu.org/licenses/gpl.html (accessed on 28 October 2018).
19. Paulson, J.W.; Succi, G.; Eberlein, A. An empirical study of open-source. *Electronics* **2016**, *5*, 74.
20. TOSHIBA, TOSHIBA SCiB Rechargeable Battery. Available online: http://www.scib.jp/en/product/cell.htm (accessed on 28 October 2018).
21. Designboom, BMW C-evolution. Available online: https://www.designboom.com/technology/bmw-c-evolution-electric-motorcycle-04-30-2014/ (accessed on 28 October 2018).
22. KiCad, KiCAD EDA Home Page. Available online: http://kicad-pcb.org/ (accessed on 28 October 2018).
23. Kim, J.; Lee, S.; Cho, B.H. Complementary cooperation algorithm based on DEKF combined with pattern recognition for SOC/capacity estimation and SOH prediction. *IEEE Trans. Power Electron.* **2012**, *27*, 436–451. [CrossRef]
24. Noriega-Linares, J.; Ruiz, J.N. On the Application of the Raspberry Pi as an Advanced Acoustic Sensor Network for Noise Monitoring. *Electronics* **2016**, *5*, 74. [CrossRef]
25. Raspberry Pi. Raspberry Pi 3 Model B. Available online: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (accessed on 28 October 2018).
26. Kong, Y.; Lee, S.; Lee, J.; Nam, Y. A head-mounted goggle-type video-oculography system for vestibular function testing. *EURASIP J. Image Video Proc.* **2018**, *28*. [CrossRef]
27. Paul, N.; Chung, C.J. Application of HDR algorithms to solve direct sunlight problems when autonomous vehicles using machine vision systems are driving into sun. *Comput. Ind.* **2018**, *98*, 192–196. [CrossRef]
28. Kölling, M. Educational Programming on the Raspberry Pi. *Electronics* **2016**, *5*, 33. [CrossRef]
29. Python, "Python.". Available online: https://www.python.org/about/ (accessed on 28 October 2018).
30. Baronti, F.; Zamboni, W.; Femia, N.; Roncella, R.; Saletti, R. Experimental analysis of open-circuit voltage hysteresis in lithium-iron-phosphate batteries. In Proceedings of the IECON 2013—39th Annual Conference of the IEEE Industrial Electronics Society, Vienna, Austria, 10–13 November 2013; pp. 6728–6733.
31. Lu, L.; Han, X.; Li, J.; Hua, J.; Ouyang, M. A review on the key issues for lithium-ion battery management in electric vehicles. *J. Power Sources* **2013**, *226*, 272–288. [CrossRef]

32. C. USA, CALB Additional Information. Available online: http://www.calbusainc.com/additional-information/ (accessed on 28 October 2018).
33. Xie, J.; Ma, J.; Chen, J. Peukert-Equation-Based State-of-Charge Estimation for LiFePO4 Batteries Considering the Battery Thermal Evolution Effect. *Energies* **2018**, *11*, 1112. [CrossRef]
34. Paul, S.; Diegelmann, C.; Kabza, H.; Tillmetz, W. Analysis of ageing inhomogeneities in lithium-ion battery systems. *J. Power Sources* **2013**, *239*, 642–650. [CrossRef]