# B#: a Battery Emulator and Power Profiling Instrument

Pai H. Chou, Chulsung Park, Jae Park, Kien Pham, and Jinfeng Liu
Center for Embedded Computer Systems
University of California, Irvine, CA 92697-2625 USA
{phchou,chulsung,jaep,phamkk,jinfengl}@uci.edu

## ABSTRACT

This paper describes B# (B-sharp), a programmable power supply that emulates the behavior of a battery. It measures the current load, calls a battery simulation program to compute the voltage in real time, and controls a linear regulator to mimic the voltage output of a battery. This instrument enables validation of battery-aware power-optimization techniques with accurate, controllable, reproducible results. This instrument also supports training mode with actual batteries, and it can even be used for recording and playback of a solar power source. This design has been prototyped and tested on hand-held devices with high accuracy and fast response time.

## Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Real-time and embedded systems

## General Terms

Measurement, Design, Experimentation

## Keywords

Battery emulation, Power profiling instrument

## 1. INTRODUCTION

Recently, researchers started developing *battery aware* power management techniques that exploit non-ideal features of batteries in order to maximize their effective lifetime. It has been shown that different discharge patterns can make a dramatic difference in the battery's life. To validate these battery-aware power management techniques, researchers can use either real batteries or battery simulation.

### 1.1 Measurement with Real Batteries

The most obvious way to validate the results is to measure the power with actual batteries. Real batteries are the most accurate, and they run at full speed. However, this approach has several drawbacks. If non-rechargeable batteries are used, then they must be replaced and disposed of after each experiment, making this approach expensive and unfriendly to the environment. Rechargeable batteries produce less waste and may even contain a smart battery interface [1] to offer more controllability. However, the results may not be very reproducible, because the discharge and recharge history of the battery will affect the total charge level each time. After each full recharge or as the battery ages, the battery will not necessarily hold the same level of charge each time. Results obtained from such experimental setups may be misleading.

### 1.2 Battery Simulation

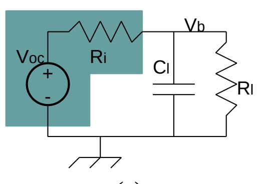
To achieve full reproducibility, researchers have turned to simulation. Several simulators have been proposed and implemented [2]. Dualfoil [3] is an electro-chemical model that solves partial differential equations in Fortran. It outputs the voltage and temperature response of the battery based on the discharge current load by the power consumer. It models *rate-capacity* and *rate-recovery* effects, but it is computationally intensive and does not capture *aging* effects. Models based on SPICE [4] and discrete-time VHDL [5] have also been proposed, and they are faster though less accurate. Researchers also proposed a stochastic model [2] to capture these effects, though they are designed to estimate battery lifetime rather than the actual voltage response.

To incorporate a battery simulator, researchers currently must simulate the system, too, so that the simulated power manager can close the feedback loop. However, the drawback with this approach is that, even if the battery simulator were infinitely fast and accurate, detailed, accurate, fast simulation of the entire system would still remain a challenge.

### 1.3 Our Approach: Battery Emulator

We propose a battery emulator, called B# ("B-sharp"), for experiments with battery-aware designs. B# is an intelligent power supply that mimics the behavior of a battery by running a battery simulator program in real-time. It senses the current load and responds by controlling the voltage in the same way an actual battery would. This enables researchers to conduct *in-situ* experiments on battery-aware designs without having to use actual batteries. Our approach combines the speed and accuracy advantages of measurement-based approaches with the flexibility and reproducibility of simulation-based approaches. The only other battery emulator we are aware of is the unpublished work on the Penn State University Battery Simulator (PSU-BS) [6]. Developed for testing electric vehicles, the PSU-BS can output up to 600W with a response time of 170ms. However, very limited information is publicly available on this project.

B# is potentially much more useful than emulating batteries. In training mode, the user can connect an actual battery for calibrating a simulation model. B# can also play back a recorded stream



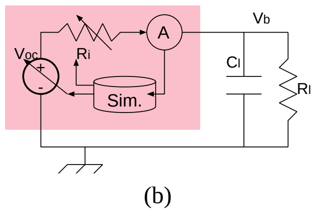


**Figure 1: Circuit models for (a) a battery + load. (b) a battery emulator + load.**

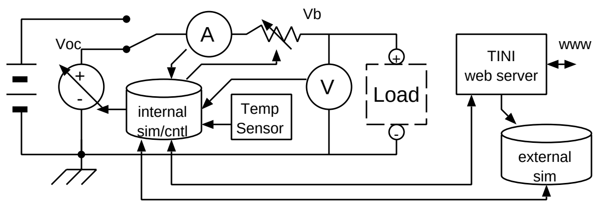


**Figure 2: Block diagram for the Battery Emulator.**

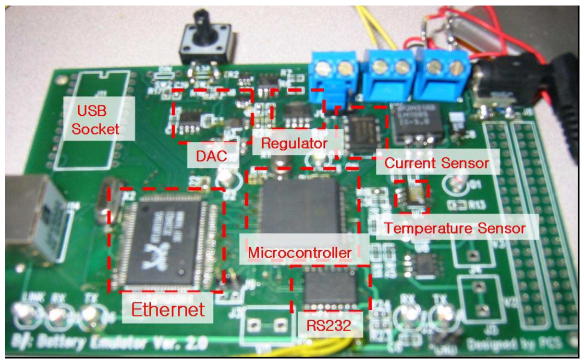


**Figure 3: Battery emulator board.**

of voltage values as collected from other sources, including a solar panel over the course of a day under different weather conditions. This paper first describes our design of the B# board. Then, we evaluate our B# implementation through a series of experiments with PDAs. We show that our emulation results closely match those with actual batteries. We conclude by outlining some technical challenges for the next version.

## 2. PROBLEM STATEMENT

Our goal is to integrate an analog subsystem for power control with a digital subsystem for simulation and access. Although power circuits and battery simulation have been studied extensively, their integration is posing new challenging problems. This section divides the problem statement into power circuitry, timing, and digital interface.

### 2.1 Power Circuitry

A battery-powered system can be modeled with an equivalent circuit shown in Fig. 1(a). The shaded region corresponds to the battery, where $V_{oc}$ and $R_i$ are the open-circuit voltage and internal resistance. The unshaded region corresponds to the load, with capacitance $C_l$ and resistance $R_l$. The load may vary over time based on the activities and power management policies. If the circuit sources current $I$, then the observed voltage of the battery, $V_b$, is

$$V_b = V_{oc} - I \times R_i \tag{1}$$

As the battery discharges, $V_{oc}$ decreases while $R_i$ increases, and both are based on the state of the battery and its internal temperature. The state of the battery is maintained by the simulation model, while the ambient temperature and current can be measured. Based on this circuit model, one possible circuit for a battery emulator is shown in Fig. 1(b). It would

1. measure the current ($I$) and temperature ($T$),
2. call the simulator to compute $V_{oc}$ and $R_i$, and
3. set the $V_{oc}$ and $R_i$ values.

The ambient temperature $T$ is used for setting the simulator's initial condition and can affect the battery's internal temperature. Most existing simulators including dualfoil compute $V_b$ directly without explicitly computing $R_i$ or $V_{oc}$. We will address this problem in the experimental results section.

### 2.2 Timing

The timing constraints on the battery emulator include the *sampling period* and the *response time*. The sampling period $p$ is defined to be the time between successive samplings of the current load by the emulator. The response time $\delta$ is defined to be the latency from the time the current is sampled to the time the output voltage is observed. We assume $0 < \delta \leq p$. Both $p$ and $\delta$ are limited by a number of factors:

- the load capacitance $C_l$, which has the effect of a low-pass filter,
- the data acquisition time, which is limited by the sampling speed and communication time to the computer, and
- the computation and communication time of running one iteration of the simulator.

We have observed that the dualfoil simulator is not sensitive to load frequencies higher than 1KHz. We measured an actual Li-ion battery whose voltage response is plotted in Fig. 9 (to be discussed further in the results section). For this battery type, we choose the value $p = \delta = 100$ms.

### 2.3 Digital Interface

The digital interface of the instrument can be divided into data acquisition, simulation, and instrument control. The data acquisition interface defines a format for the current and temperature data to be streamed to another computer. The receiving computer can save the data to a file or feed it to a simulator. The simulator can potentially reside in two places: it can run locally inside the instrument itself, or it can run on a separate computer externally. The instrument control interface defines a format for the end user to control and configure the instrument.

## 3. EMULATOR DESIGN

### 3.1 Battery Emulator Board

The picture of our B# board is shown in Fig. 3. It consists of the power circuitry, the measurement circuitry and the digital subsystem. The firmware handles control and communication with a host computer.

### 3.1.1 *Power Circuitry*

The power circuitry essentially implements the equivalent circuit in Fig. 1(b). It is implemented with a 10-bit DAC and an adjustable linear regulator. The DAC is controlled by the microcontroller over the SPI(3-wire) interface and controls an adjustable linear regulator. For this implementation, the output voltage range is 1.0V-4.5V. The lower-bound voltage is actually variable, and it can certainly be set to 0V. The reason we can choose a higher lower-bound voltage is that most battery-powered devices have a cut-off voltage that is actually higher than 1V. By setting a higher lower-bound voltage, we are able to control the output voltage with a higher resolution.

Our choice of linear regulator has a maximum current rating of 800mA. If this is not sufficient, we could choose another component with higher current or compose several regulators in parallel. Note that we do not include an explicit resistor to model $R_i$ because the linear regulator already has its own internal resistance $R_i'$ that is actually higher than the battery's $R_i$. Therefore, adding a passive resistor will not be able to correctly model it; instead, we accomplish this through voltage adjustment to be discussed in the measurement section.

### 3.1.2 *Measurement Circuits*

The emulator contains three measurement circuits for the current ($I$), temperature ($T$), and voltage ($V_b$). These are implemented by sampling the voltage values of the three respective measurement circuits. We convert current into voltage by measuring the voltage drop across a very small resistor (35m$\Omega$) in series with the power supply, and then use an ADC to digitize the voltage. We convert temperature into voltage by using a temperature sensor IC. The PIC16F877 microcontroller already has eight built-in channels of 10-bit ADCs with an acquisition time of 19.72$\mu$s. We use three channels for measuring $I$, $T$, and $V_b$.

### 3.1.3 *Digital Subsystem*

The digital subsystem consists of a PIC microcontroller and the battery simulator. The PIC is responsible for controlling the power circuitry and configuration. The simulator can run either locally on the PIC or externally on a workstation.

The B# board uses the PIC16F877 microcontroller, which has 256K words of flash programming memory. This PIC also contains eight built-in ADC channels and a UART for serial communication. The B# board has an RS-232 serial port, an Ethernet port, and an optional USB port to communicate with a host computer. The baud rate of the serial port is set to 19,200bps for firmware download, but it is not fast enough for communicating with the simulation program. Instead, the Ethernet interface, with a sustained bandwidth of 2.5Mbps, is used to transfer the measured or simulated data between the B# board and the host computer.

The PIC software runs a command interpreter, which parses instrumental commands and responds to query or configuration commands, including software calibration of the ADCs. Each command is acknowledged according to a protocol. It also supports simple scripting for conditional and iterative sampling and control without having to rely on explicit communication with the host each time.

## 3.2 Simulation Software

B# can simulate batteries two ways: external simulation and local simulation. With external simulation, B#'s microcontroller sends the load current level to another computer running the battery simulation software to compute the voltage response; with local simulation, B# computes it itself.

### 3.2.1 *External Simulation*

Currently B# is set up with an external host computer running a modified version of dualfoil [3] . The Fortran source code for dualfoil is freely available. It performs simulation in batch rather than interactively. We modified dualfoil to enable the simulator to run for a period of time before reading the new inputs for the next period. The new simulation program is also customized to exchange data with the B# board by making Ethernet communication calls.

To simulate the chemical reactions inside a battery in detail, dualfoil requires significant amount of computation resources. In general, it is challenging to provide the B# board with the software simulation results in real-time. We define a simulation period to be 80ms. Ethernet adds 1–2ms to the total latency. This latency is rather negligible when the entire simulation is to be performed for a much longer period (e.g., a few minutes). We observe that the current variance during this period is insignificant and has little impact on the overall battery life. If necessary, the length of the simulation period can be shortened by using a faster host computer.

### 3.2.2 *Local Simulation*

Local simulation means executing the battery simulator on the instrument itself. If simplified simulators are acceptable, and they are simple enough, then they can run locally on B#'s microcontroller, which runs at 20MHz. We implemented a version that resembles a Mealy machine with 256 charge states, except the state transitions are probabilistic. The ideas are borrowed from [2]. Each state has an output function that maps the 256 (8-bit) $I$ levels to $V_b$. In the most general case we need $256 \times 256 = 64K$ entries, depending on the battery type and capabilities. We filled the entries of the table using B#'s data acquisition mode to collect many samples of real battery data, so that we could process the data in batch. Once the table is built, the simulator can start from any arbitrary state. It uses the measured current ($I$) to index into the table for the next charge state and the voltage output, by interpolation if necessary. State transition probabilities are updated over time to prevent the battery from looping back to the same charge state indefinitely.

## 4. EVALUATIONS

This section evaluates the current implementation of B# by actual measurements with handheld devices and real batteries. Fig. 4 shows an experimental setup with the B# board measuring the current and voltage values of a Compaq iPaq's battery, and a host computer collecting the measured data and running dualfoil simulation. The issues are to validate

1. $R_i$ as shown in Fig. 1(b);
2. response time of the battery; and
3. accuracy of emulated battery life

## 4.1 Data Acquisition on PDA Platforms

We tried B# on a number of devices, including a Palm IIIe organizer, a Compaq iPaq, a Casiopeia PDA, and several others. The Palm takes two AAA batteries for a 3.0V power source. The Palm PDA has a relatively small range of current draw, which varies between 20mA to 40mA in its heaviest use. The Palm has a cut-off voltage around $V_b = 1.5V$, at or below which the system shuts down. The rest of this paper reports results with the iPaq and the Casio PDA, which consume about two orders of magnitude more power than the Palm. To vary the load, we controlled combinations of backlight settings and video/audio activities.

Fig. 5 shows the voltage and current data collected on the Casio PDA over 3 minutes. Data for additional voltage/current profiles were collected but not shown due to paper length limitations.

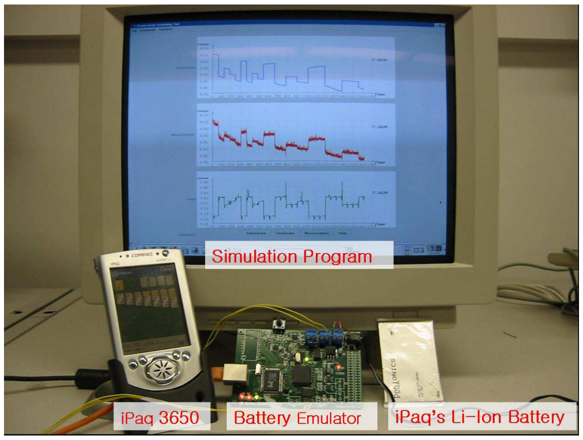


**Figure 4: Experimental setup.**

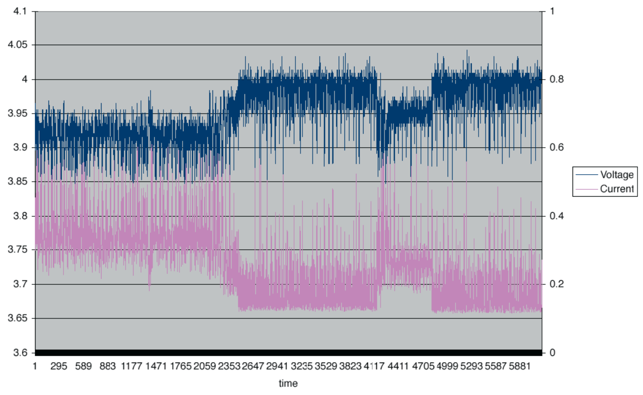


**Figure 5: 3 minutes of the Voltage (navy) and current (magenta) data collected from the Casio PDA.**

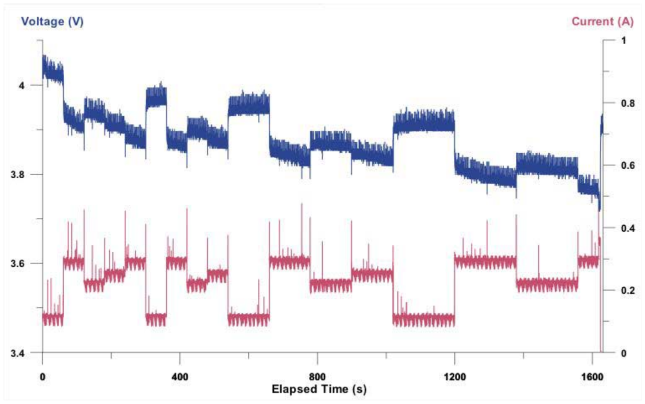


**Figure 6: 27 minutes of voltage (blue) and current (red) data collected from an iPaq running on its own battery. On battery depletion, the voltage goes back up but the current goes to 0.**

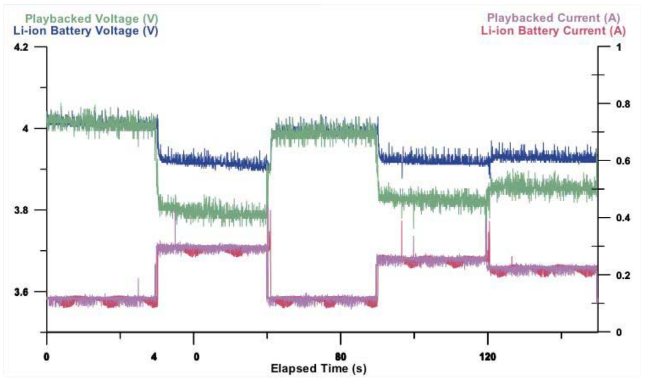


**Figure 7: Voltage and current profile of a measured Li-ion battery ($V_b$ in navy, $I$ in red) and the measured playback version ($V'_b$ in green, $I'$ in magenta) by setting $V'_{oc}(t) = V_b(t) + 0.1V$.**

The Compaq iPaq is another hand-held device we evaluated with our battery emulator. We chose it to experiment with a larger dynamic range of load. Its processor consumes a minimum of 150mW alone, and the backlit color display consumes even more power. Unlike the Palm IIIe, this particular model of iPaq uses a rechargeable battery and is sealed without any externally replaceable batteries. We tested the load by varying the brightness of the LCD backlight. The voltage data we collected from the iPaq running on its own battery is shown in Fig. 6. It lasts 27 minutes, and on depletion, the voltage shoots back to the fully charged level but the current is nearly 0A.

## 4.2 Internal Resistance

In this experiment, we use the playback feature to test B# ability to emulate the internal resistance of the battery. In our circuit, the linear regulator can correctly produce the proper *open-circuit* voltage value. However, the measured $V'_b$ is actually slightly lower than $V_{oc}$ for $I > 0$. This is because the linear regulator has its own internal resistance $R'_i$, which is actually a function of $I$.

To compare the internal resistance, we first set B# to data acquisition mode. The measured $V'_b$ and $I$ are shown in navy and red lines in Fig. 7. Then, we playback the recorded voltage, and add a constant $\Delta$ voltage of 0.1V to drive the $V'_{oc}$ output. The measured $V'_b$ and $I'$ are shown in green and magenta, respectively [1]. We observe that $I$ (measured on battery) and $I'$ (measured on playback) match each other exactly, but $V_b \geq V'_b$. The reason is due to the internal resistance $R_r$ of the linear regulator. Furthermore, since the linear regulator is an active circuit, its internal resistance actually varies as a function of $I$. Real Li-ion batteries have an $R_i$ around 0.2Ω on a full charge, and rises to around 0.7Ω near depletion [7]. In contrast, the linear regulator has a variable internal resistance $R_r$ as shown in Fig. 8.

From this we can conclude that adding a programmable resistor to model $R_i$ would not be a feasible approach for several reasons. First, programmable resistors do not come in such small value increments. Second, since the linear regulator's internal resistance is already higher than that of the battery, adding a passive resistor in series will not reduce the resistance. As a result, $R_i$ must be implemented by means of adding a $\Delta(I)$ voltage to $V_{oc}$. The adjustment

[1]For monochrome copies of this paper, green and navy are the two top curves that are stacked on top of each other when high, and the green curve dips lower than navy when low. Magenta and pink are the two stacked curves below.

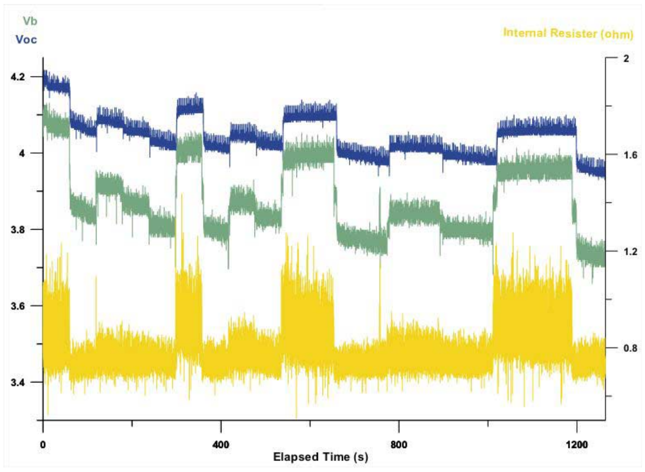


**Figure 8: Internal resistance $R_r$ (yellow) of B#'s linear regulator as computed from $(V'_{oc}(\textbf{set}) - V'_b(\textbf{measured}))/I$ over 21 minutes.**

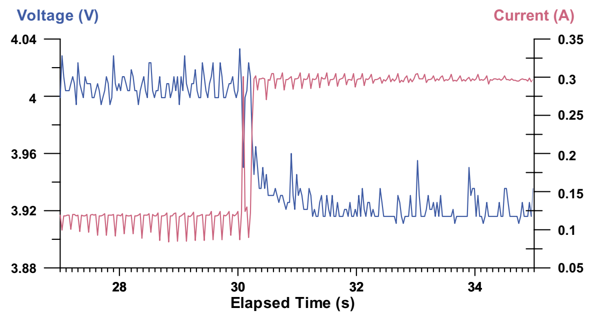


**Figure 9: Measured response time of Li-ion battery to changes in current load.**

value $\Delta(I) = V'_{oc} + R_r(V'_b, I) \times I$.

## 4.3 Response time

The response time of real batteries can be measured using our circuit. Fig. 9 shows a change in current load (red curve) causes a change in $V_b$ (blue curve). The current starts to increase at 30.18s and the voltage finishes decreasing at 31.26s. If we change the range of the current swing, the response time would be different, but in this experiment, the range of the current swing is the largest expected. Therefore, we can consider this response time as the shortest expected of the iPaq's DLP305590 battery.

## 4.4 Simulation Accuracy

Given that the power circuitry can output the desired voltage with calibrated internal resistance, the question now is whether the chosen simulation granularity can accurately emulate a battery. To recall, we have a maximum latency of 100ms, and in practice this is due to 80ms simulation latency plus 1–2ms Ethernet. We noticed that the simulation latency of dualfoil is dependent on the amount of fluctuations in the load: it could produce the simulation results more quickly if the load current does not change for a certain period of time. On a Pentium-III 500MHz PC running Linux, if the load current remains constant within 100ms, the simulator is able to produce the results in less than 80ms. However, if the load current fluctuates a few times during this 100ms interval, the simulation software may need a longer time to compute the voltage response. On a 2GHz Athlon PC running Linux, the simulation latency is around 30ms.

Our metric is somewhat qualitative. We compare the power profiles as shown in Fig. 10. Given the same current load (bottom curve, green), the top curve (blue) shows the voltage response of dualfoil, and the middle (red) shows that of the real battery. Note that the simulated voltage is the open-circuit voltage, which will be higher than the measured voltage when internal resistance is considered. In terms of the simulated battery life, B# tracks the real battery almost exactly.

## 5. CONCLUSIONS

We present the B# battery emulator for batteries whose behavior can be modeled in software. B# can interface with state-of-the-art battery simulators running remotely as well as simple, table-driven battery simulation locally. Moreover, its data acquisition and replay features makes B# a versatile research tool. It can be used to collect power profiles on actual computers and replay the power profile on another device. It is also useful for training battery models. It enables in-situ experiments with various charge level and voltages. It not saves the cost of buying and disposing of batteries, but more importantly, it makes these experiments reproducible.

Future work includes model refinement and recharge emulation. Refinement will continue for modeling the internal resistance as well as high-current load and other exceptional conditions. Improving and more precisely controlling the response time of the voltage will also make B# applicable to devices that are ultra sensitive to the fluctuations in the supply voltage. To emulate recharge, we plan to design new hardware features that not only support reverse current flow but also incorporate smart battery chips for charge optimization.

## Acknowledgments

The authors would like to thank Dr. Bruce Tromberg for making available the laboratory space for prototyping this instrument. This work was sponsored by the National Science Foundation under grant number CCR-0205712.

## 6. REFERENCES

[1] Smart Battery System Implementers Forum. SBS IF Specifications. In *http://www.sbs-forum.org/specs/*, 2001.

[2] Debashis Panigrahi, Carla Chiasserini, Sujit Dey, Ramesh Rao, Anand Raghunathan, and Kanishka Lahiri. Battery life estimation for mobile embedded systems. In *Proc. Intl. Conf. on VLSI Design*, pages 55–63, January 2001.

[3] M. Doyle, T.F. Fuller, and J.S. Newman. Modeling of galvanostatic charge and discharge of lithium-ion insertion cells. *J. Electrochemical Society*, 141(1):982–990, April 1994.

[4] S. Gold. A PSPICE macromodel for lithium-ion batteries. In *Proceedings of the 12th Battery Conference*, pages 9–15, 1997.

[5] L. Benini, G. Castelli, A. Marcii, E. Macii, M. Poncino, and R. Scarsi. A discrete-time battery model for high-level power estimation. In *Proceedings of DATE*, pages 35–39, 2000.

[6] Matthew M. Mench. Penn State University Battery Simulator (PSU-BS). In *http://mtrl1.me.psu.edu/mtrl/BatSimFac.htm*, 2000.

[7] David Linden and Thomas Reddy. *Handbook of Batteries*. McGraw-Hill, 2001.

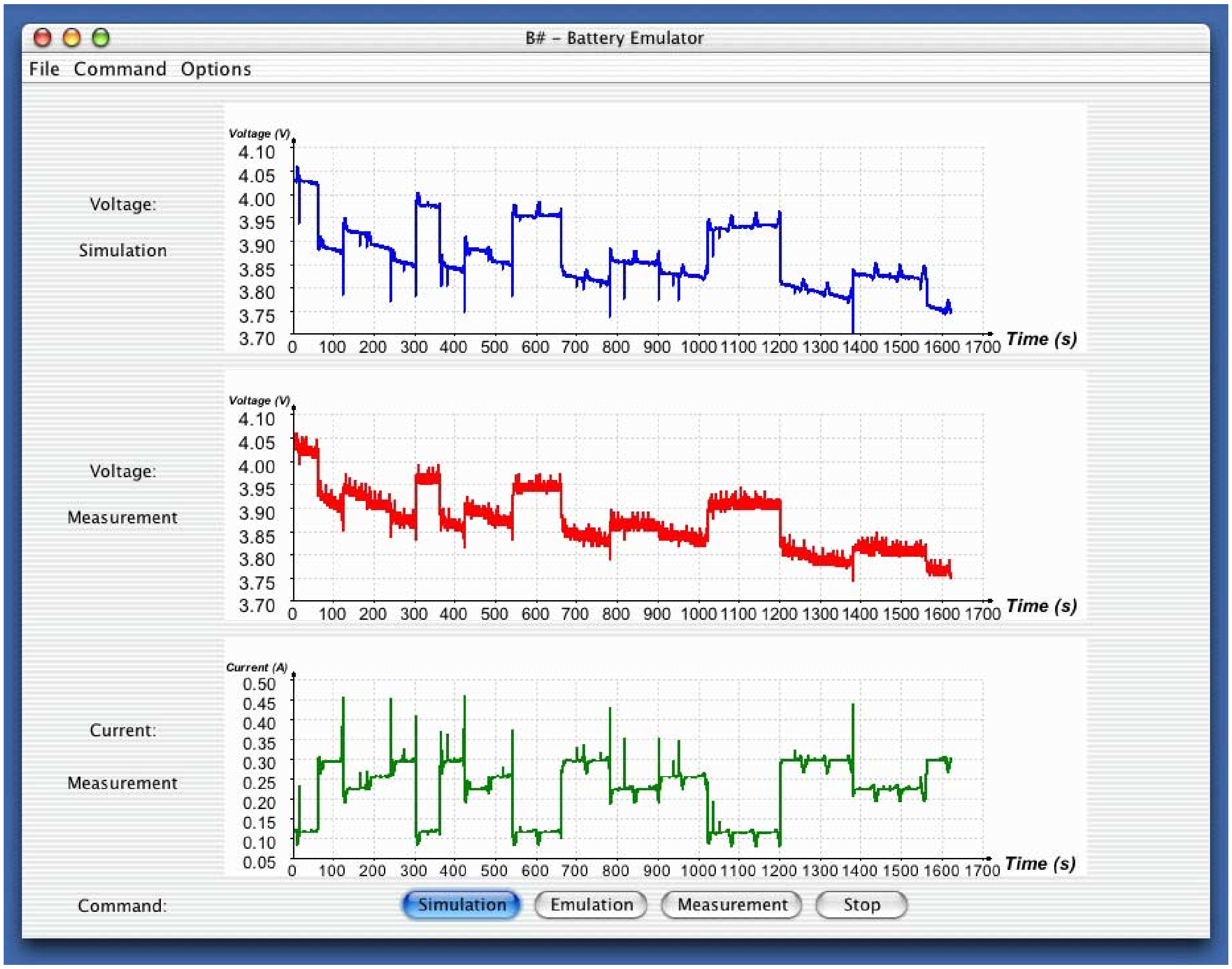


**Figure 10: A screen shot of the battery simulation graphical user interface showing the simulated voltage profile, measured voltage profile, and the current load to drive both.**