Arduino-based Smart Battery Cell Research Platform

Christian Siegel, *Aarhus University*

(Engineering Research and Development Project)

***Abstract*—This paper introduces an accessible smart battery cell research platform based on the Arduino environment. Smart battery cells are a recent research direction in the scope of Battery Management Systems (BMS) for large lithium-ion battery packs, where each battery cell is equipped with an individual Cell Management Unit (CMU). The wireless System on Chip (SOC) solution ESP8266 forms the core of the research platform. It is deployed on an off-the-shelf development board and extended with a custom shield. A single-board computer provides the wireless network for the smart battery cells and acts as gateway to a web service for interaction purposes. As a proof-of-concept, a web client is created that communicates with the battery pack via the MQTT protocol.**

I. Introduction

LITHIUM Ion (Li-Ion) batteries are utilized in most emerging electrically powered technologies such as electric vehicles, power tools or stationary energy storages. Large battery packs are made of battery cells connected in series and parallel to achieve a desired voltage and capacity. Due to their chemical properties, Li-Ion cells must be operated in strictly specified voltage, current, and temperature ranges to avoid critical damages to the battery cells. State-of-the-art battery packs use a central Battery Management Systems (BMS) to monitor cell parameters and control the charge and discharge of cells in a battery pack. Since these BMS are tied to a specific battery pack architecture they need to be customly developed. A recent proposal is to move the battery pack management from a centralized BMS towards individual Cell Management Units (CMU) at the cell level [1]. This leads to more flexibility in composing the battery pack architecture and reduces development time. Furthermore, the reduced battery pack complexity simplifies efficient charge transfer between individual cells, compared to traditional passive balancing, in order to equalize the State of Charge (SoC) of all battery cells.

Contributions of this paper. Recent research on smart battery cells was conducted on pricey custom development platforms incorporating powerful microcontrollers and real-time operating systems. Contribution of this paper is to introduce a more accessible smart battery cell research platform based on the Arduino platform, using mainly off-the-shelf hardware components. In section II, related work and its proposed solutions are investigated. Section III discusses and describes the used hardware and modifications made. The software is depicted in section IV. Finally, a conclusion is drawn in section V.

II. Related Work

Previous research was conducted on rather powerful and pricey hardware like a STM32F407 Cortex-M4 microcontroller board [1][2]. The microcontroller was connected to the battery cell through a custom designed sensor and control board, which could be interfaced using a Serial Periphal Interface (SPI) bus. The invidual smart cells communicated via a Controller Area Network (CAN) bus connected to the microcontroller boards [1].

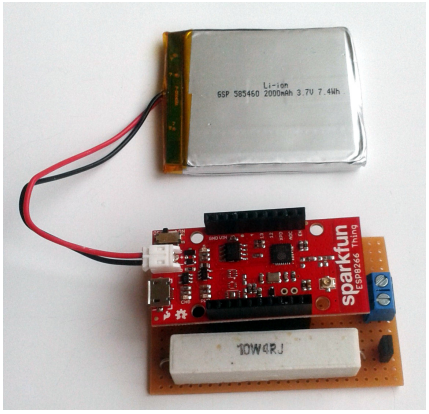

Fig. 1. Smart battery cell comprised of an ESP8266 development board, an extension shield, and a battery cell.

Especially communication poses a significant challenge, since wired busses like CAN, Ethernet, Inter-Integrated Circuit (I^2C), or Universal Asynchronous Receiver/Transmitter (UART) require a common ground potential. If the smart cells are connected in series, there is no common ground and the bus has to be galvanically separated using e.g. optocouplers.

[3] and [4] propose and demonstrate the use of contactless communication via capacitive coupling for distributed monitoring and controlling of battery cells. This technology can also be applied for communication between smart battery cells. Main disadvantage of this approach is the need for additional communication hardware which is not available off-the-shelf.

[5] proposes wireless communication for monitoring lead acid vehicle batteries. The emerging trend of an Internet of Things (IoT) leads to a range of hardware manufacturers offering low-cost wireless System on Chip (SOC) solutions. A full TCP/IP stack provides a well known, and convenient to use protocol suite for distributed systems.

III. Hardware

This section discusses the hardware choices made in order to realise the Arduino-based smart battery cell research platform, which is shown in Fig. 1.

TABLE I
COMPARISON OF WIRELESS SOC MCU PROPERTIES [6][7]

	ESP8266	CC3200
Core	Tensilica L106	ARM Cortex-M4
Clock	80 – 160 MHz	80 MHz
RAM	∼50 kB usable in station mode	≤256 kB
Flash	external	external
I/Os	16	27
ADC	1x 10-bit	4x 12-bit
I²C	1	1
SPI	1	1
UART	1	2

TABLE II
COMPARISON OF WIRELESS SOC POWER CONSUMPTIONS [6][7]

	ESP8266	CC3200
Tx	∼145 mA	∼215 mA
Rx	56 mA	59 mA
Modem idle/sleep	15 mA	15.3 mA
Deep-Sleep	10 µA	4 µA

A. Development Board

To prevent the need for a pricey, customly designed circuit board, the research platform should be based on an off-the-shelf Microcontroller Unit (MCU) development board. In order to further reduce the need of additional hardware, this development board should also comprise a communication interface. As already mentioned in section II, there are wireless SOCs available that meet these requirements, like the *Espressif Systems ESP8266*[1] or the *Texas Instruments CC3200*[2]. The MCU properties of the named SOCs are outlined in Table I. Both MCUs have a 32-bit core. While the CC3200's core can be clocked at maximum 80 MHz, the ESP8266's core clock can be increased up to 160 MHz. With around 50 kB of usable RAM in station mode, the ESP8266 has significantly less RAM than the CC3200, which provides 256 kB. Both the ESP8266 and the CC3200 don't come with internal flash storage. The CC3200 has almost twice the amount of general purpose I/Os than the ESP8266, and four ADCs instead of one. The CC3200's ADCs also have a higher resolution of 12 bits, compared to the 10-bit ADC of the ESP8266.

Since the CMU is powered by a battery cell, its power consumption is an important figure to maximize the smart battery cell's capacity. Table II shows the power consumption of both wireless SOCs. The ESP8266 typically draws less current while sending than the CC3200, with ∼145 mA and ∼215 mA respectively. However, the transmission current heavily depends on the WLAN protocol and the transmission power, which can be configured. The receiving current difference is mimimal, with typically 56 mA for the ESP8266 and 59 mA for the CC3200. When the wireless module in sleeping mode, both wireless SOCs also draw almost the same current, with 15 mA and 15.3 mA respectively. In the deepest sleeping mode both devices use ≤10 µA.

Although the ESP8266 is inferior to the CC3200 regarding hardware features, it is chosen as the core of the research platform. This is due to a significant lower price while still meeting the requirements for the platform. The SOC is deployed on the *SparkFun ESP8266 Thing*[3] development board. It comprises an ESP8266, flash storage, an antenna, an on-board Li-Po charging circuit, and a JST socket to connect a battery cell. Hence, only the passive cell balancing, and the voltage measurement circuits need to be added externally.

B. Battery

A prismatic, 2000 mAh Li-Po battery with a JST connector is used. It can be directly plugged into the development board. Its capacity is similar to industrial 18650 cylindrical cells, which typically provide around 2500 mAh. This enables it to power the CMU for several hours without recharging. Furthermore, the battery has a built-in protection circuit to prevent damage to the cell if used improperly.

C. Extension Shield

The remaining additional hardware components are combined to an extension shield for the development board.

The *SparkFun ESP8266 Thing* board doesn't provide a breakout pin for V_{BAT}. However, there is a *not connected* (NC) pin in the FTDI pin header (`FTDI_BASICPTH`) [8]. A jumper wire from the JST battery connector (`JP2`) to the FTDI pin header maps V_{BAT} to the NC breakout pin.

The voltage of the Li-Po battery cannot directly be measured with the ESP8266's ADC. To map the maximum voltage of the battery (4.305 V) to the maximum ADC voltage (1 V) a voltage divider, consinsting of a 22 kΩ and a 100 kΩ resistor, is used. This results in a resolution of 5.42 mV. To prevent the discharge of the battery through the resistors while the ESP8226 is switched of, the voltage divider is not wired to ground but to `GPIO4`. This pin can be configured to output *low* (ground) while powered on. If the ESP8266 is powered off, `GPIO4` has a high impedance and the cell is not discharged.

If cells are connected in series, the charging or discharging must be stopped when any of the cells reaches its upper or lower voltage range limits. This requires the cells to be balanced externally. An easy to implement balancing method is *passive balancing*. It uses switchable resistors to bypass cells which already reached their maximum voltage during charging [9]. Passive balancing is realized using a 4 Ω, 10 W power resistor and a FQP30N06L n-channel MOSFET. The MOSFET gate is conncted to `GPIO5` which also controls a LED on the development board [8]. Thus, the state of the passive balancing can easily be observed. A pull-down resistor ensures $V_{GS} < V_{GS,off}$ while the ESP8266 is switched off.

D. Gateway

The ESP8266 can use an existing wireless network (station mode) or provide a network itself (access point mode). The latter allows to operate the smart battery cells in a mesh network. Among other things, this requires routing strategies between the single nodes, which adds complexity to the

[1] https://espressif.com/en/products/hardware/esp8266ex/overview
[2] http://www.ti.com/product/CC3200
[3] https://www.sparkfun.com/products/13231

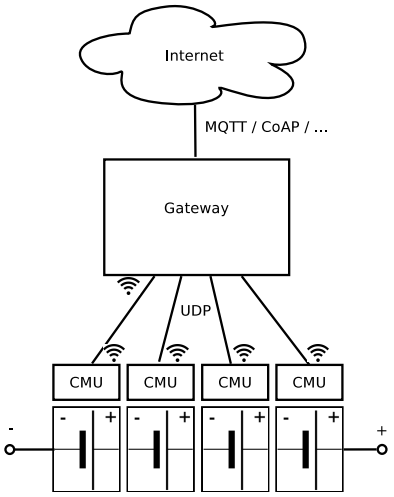

Fig. 2. Deployment diagram of a battery pack comprising four smart battery cells connected in series, and a common gateway.

distributed communication of the battery pack. For this reason, a single wireless access point is used for all cells. This access point also acts as gateway between the battery pack and its environment. Figure 2 schematically depicts a battery pack consisting of four smart battery cells connected in series. The common gateway provides the WiFi network for the CMU communication and relays communication to some sort of web service.

A *Raspberry Pi 3 Model B*[4] single-board computer is used as the gateway. Using the *Raspbian GNU/Linux* operating system it can be configured to act as wireless router with Dynamic Host Configuration Protocol (DHCP), and run the gateway software.

IV. Software

A. Cell Management Unit (CMU)

The CMU software uses the Arduino application programming interface (API). The ESP8266 Arduino core can be added to a standard Arduino installation as a third-party platform package.[5]

The CMU connects to the gateway network with preconfigured Service Set Identifier (SSID) and passphrase. After establishing a connection to the network, the state of the smart battery cell is periodically broadcasted via UDP messages. The state comprises the current cell voltage in mV, the balancing state (on = 1 / off = 0), and the balancing mode (manual = 0 / automatic = 1). The state message has a size of five bytes. Table III illustrates the message structure.

TABLE III
Cell state message structure

Byte	1	2	3	4	5
	Message ID = 0	Voltage [mV]		Balancing state	Balancing mode

[4]https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[5]https://github.com/esp8266/Arduino

All CMUs continously check for incoming messages and handle them. If the state message of another CMU is received, this smart battery cell's state is cached in order to calculate the battery pack's average cell voltage. Since no topology information is available, the single identifier of a CMU is its IP address, which is assigned by the gateway. In a class C network, only the last octet of the address varies. Hence, this octet can be used as unique ID, as shown in the following example.

$$192 \,.\, 168 \,.\, 0 \,.\, \underbrace{42}_{\substack{CMU \\ ID}}$$

By saving the state of each smart battery cell in the network, the number of cells in a network is bounded by the CMU memory. Furthermore, in a class C network the number of assignable addresses is limited to 254, which should be sufficient in this research context. To overcome memory limits in large battery packs, CMUs could be clustered in multicast groups. These groupy can then just share aggregated information.

Every CMU's state is marked with the millisecond timestamp of the last received update. If an entry is older then two seconds, this entry is ignored when calculating aggregated values. This allows to dynamically reconfigure the battery pack by adding or removing smart battery cells from the network.

To control the passive balancing of the smart battery cells, another message type is defined. The balancing control message has a size of three bytes. The structure of the message is shown in Table IV.

TABLE IV
Balancing control message structure

Byte	1	2	3
	Message ID = 1	Balancing state	Balancing mode

If the balancing mode is set to *manual*, the balancing circuit can be turned on and off by setting the balancing state. On the other hand, if the balancing mode is set to *automatic*, the cell continously compares its cell voltage to the aggregated battery pack average voltage. If the own voltage is higher than the average, the passive balancing is automatically activated.

B. Gateway

The gateway software, written in Java, is structured in four main parts consisting of configuration, cell communication, web communication, and the battery pack domain model (see Fig. 4).

The cells client listens for broadcasted CMU state messages on the network and updates the model of the battery pack accordingly. It also forwards change requests of the balancing state and mode to the individual CMUs.

The web client gets the state of the battery pack and its single cells from the model. This information can then made accessible via an arbitrary protocol. As a proof of concept, a MQTT client is implemented. It periodically publishes every single cell state, as well as the minimum, maxmimum, and

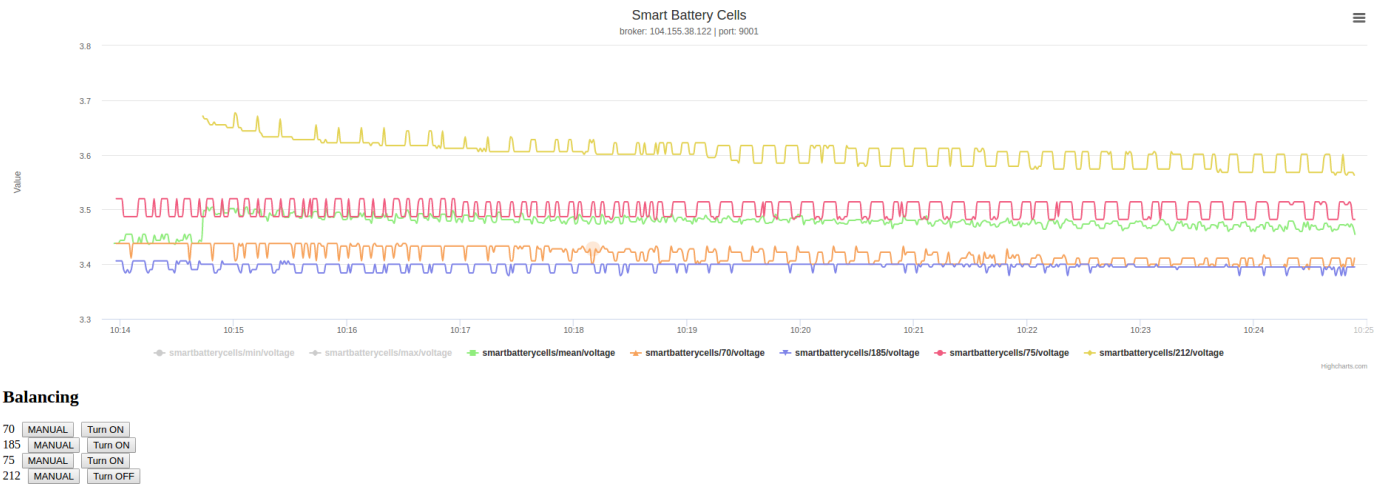

Fig. 3. Screenshot of the web based user interface. The cell with ID 212 (yellow) was added to the battery pack during runtime. This caused the average battery pack voltage (green) to raise. The passive balancing of the newly added smart battery cell was manually activated, what caused a faster voltage drop over time.

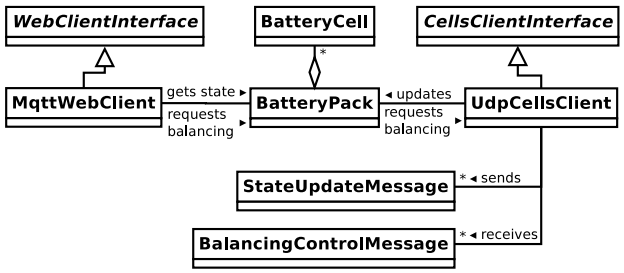

Fig. 4. Gateway classes for web and cells communication and battery pack domain model.

average battery pack voltages. It also subscribes to a balancing topic for every smart battery cell and forwards these requests to the model.

In order to enable convenient changes to configuration parameters, a YAML file is used. It is loaded at startup to create a configuration object, which subsequently is used by the rest of the software.

C. User Interface

The most basic user interface can be a class that implements the `WebClientInterface` and periodically outputs the battery pack state to the Java console.

As mentioned in section IV-B, a MQTT client is implemented to publish the cell state to a MQTT broker. A simple web based user interface is used to display this information. The balancing mode can be controlled via buttons that trigger the publishing of a control message to the MQTT broker. Fig. 3 shows a screenshot of the user interface with four battery cells.

V. Conclusion

This paper introduced an Arduino-based smart battery cell research platform. The platform comprises an ESP8266 WiFi SOC on the inexpensive *Sparkfun ESP8266 Thing* development board. Additional hardware for cell voltage measurement and passive balancing is combined on a simple extension shield. The CMUs communicate via UDP/IP on a WiFi network created by a Raspberry Pi 3 Model B, which also acts as a gateway. The gateway runs a software to relay cell state information and balancing control messages between the cell network and a web service. As a proof-of-concept the communication between the battery pack and a graphical web interface was implemented via the MQTT protocol.

The smart battery cell research platform software currently only supports basic features which are to be researched and extended using this platform in the future. Furthermore, the ADC showed jitter which has to be reduced by additional hardware and/or software filtering.

References

[1] S. Steinhorst, M. Lukasiewycz, S. Narayanaswamy, M. Kauer, and S. Chakraborty, "Smart cells for embedded battery management," in *Proceedings of the 2nd International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA 2014)*, pp. 59–64, 9 2014.

[2] S. Steinhorst and M. Lukasiewycz, "Topology identification for smart cells in modular batteries," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2015)*, pp. 1–4, 3 2015.

[3] N. Martiny, T. Mhlbauer, S. Steinhorst, M. Lukasiewycz, and A. Jossen, "Digital data transmission system with capacitive coupling for in-situ temperature sensing in lithium ion cells," *Journal of Energy Storage*, vol. 4, pp. 128–134, 2015.

[4] V. Lorentz, "Electrical energy storage: Distributed battery monitoring." EES 2012 Electrical Energy Storage Exhibition Munich, Nov. 2012. http://media.nmm.de/91/lorentz_fraunhofer-iisb_14.11.2012_10.30_26767091.pdf.

[5] M. Schneider, S. Ilgin, N. Jegenhorst, R. Kube, S. Pttjer, K. R. Riemschneider, and J. Vollmer, "Automotive battery monitoring by wireless cell sensors," in *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, pp. 816–820, May 2012.

[6] Espressif, "ESP8266EX Datasheet 2016 v4.9." https://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf, 2016. Accessed: 2016-08-30.

[7] Texas Instruments, "CC3200 SimpleLink™ Wi-Fi® and Internet-of-Things Solution, a Single-Chip Wireless MCU." http://www.ti.com/lit/gpn/cc3200, Feb. 2015. Accessed: 2016-08-30.

[8] Jim Lindblom, "SparkFun_ESP8266_Thing." Schematic, https://cdn.sparkfun.com/datasheets/Wireless/WiFi/SparkFun_ESP8266_Thing.pdf, May 2015. Accessed: 2016-10-16.

[9] M. Isaacson, R. Hollandsworth, P. Giampaoli, F. Linkowsky, A. Salim, and V. Teofilo, "Advanced lithium ion battery charger," in *Battery Conference on Applications and Advances, 2000. The Fifteenth Annual*, pp. 193–198, IEEE, 2000.