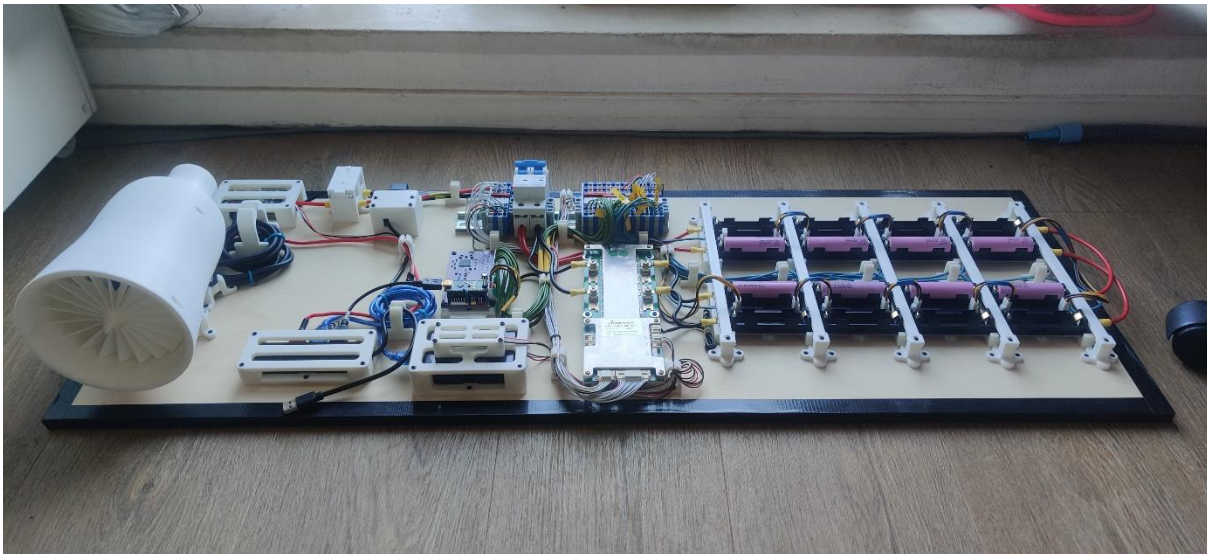
# Lithium Battery Research Rig

User Guide



Aleksandar Zdravkov

Eindhoven

16/07/2024

# Table of Contents

# Definitions

| | |
|---|---|
| AC | Alternating Current |
| ADC | Analog to Digital Converter |
| BMS | Battery Management System |
| BLDC | Brushless DC (Motor) |
| CAD | Computer Aided Design |
| CSA | Curriculum Supporting Activity |
| DC | Direct Current |
| DIY | Do-It-Yourself |
| EV | Electric Vehicle |
| ESC | Electronic Speed Controller |
| FET | Field Effect Transistor |
| GND | Ground |
| LFP | Lithium Iron Phosphate ($LiFePO_4$) |
| R&D | Research and Development |
| RPM | Revolutions Per Minute |
| SOC | State of Charge |
| UART | Universal Asynchronous Receiver-Transmitter |
| USB | Universal Serial Bus |

# SAFETY INFORMATION

This battery test and research rig is intended for use with 18650 size Lithium-based batteries. Most common of these are Li-Ion and Li-Poly, commonly found in smart devices (phones/laptops/tablets/etc.) and EV battery packs. These chemistries are incredibly dangerous as mishandling of the battery or subjecting it to abnormal operating conditions could cause it to enter a thermal runaway state. This is a terminal fault as Li-Ion and Li-Poly cells produce pure oxygen as a byproduct of the lithium in the cell reacting with the air and the other compounds in the cell structure as it's disintegrating on fire. Therefore, a Li-Ion/Poly cell fire CANNOT be extinguished as the fire is essentially producing its own fuel. This generally does not apply to LiFePO4 batteries as their chemical composition makes them much more stable. When in thermal runaway state, LFP cells do not produce their own oxygen fuel. Instead, a failing LFP cell simply vents its electrolyte compound (PO4) as a gas/vapor through a one-way relief valve. The temperature reached during thermal runaway of LFP cells is also typically not high enough to cause an uncontrolled fire.

For the safety of the students and Fontys Engineering, it is important that students given access to this rig are well informed about the operational principles and hazards of lithium-based batteries. When the rig is in use, it MUST ALWAYS be accompanied by a container full of sand and/or dirt that can be used to dump a failing cell into to burn up in safety (as mentioned above, once a cell is in thermal runaway, it cannot be saved). If a cell fails (starts smoking/swelling/burning) while installed on the rig, pour the sand/dirt on top of it to try to limit the spread to the other cells in the rig. If the failing cell was successfully contained, wait for it to cool off and carefully remove it from the rig. The cell-holder PCB can be easily replaced if it suffers damage. In case covering a failing cell with sand/dirt does not contain the fire, you can attempt to save the rest of the setup by cutting all the wires between the batteries section of the rig and the electronics. This will allow you to break-off the batteries section of the plywood board and let it burn, saving the expensive monitoring electronics, BMS, loads, etc. ONLY DO THIS IF YOU ARE CONFIDENT YOU HAVE ENOUGH TIME UNTIL FIRE IS COMPLETELY OUT OF CONTROL TO EXECUTE THE PROCEDURE. IF UNSURE, JUST RUN AWAY AND CALL FOR HELP.

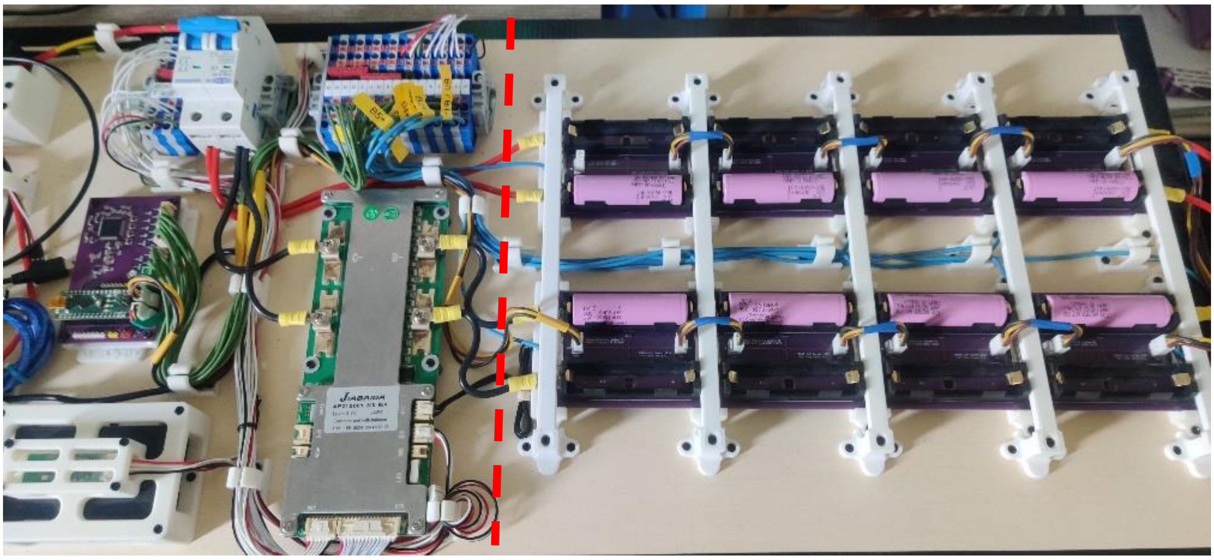


*Figure 1: Battery cell section of rig. Line of red dashes shows where to cut cables and break board in case of terminal battery fault.*

# IMPORTANT INFORMATION

Never leave battery cells in the rig unattended, especially while in storage! The BMS and Battery Monitoring board will draw power from any cells left in the holders and gradually drain them to 0V. Dropping below minimum spec sheet voltage is very bad for lithium cells so it should never be allowed to happen. Removing just the first or last cell in the stack does NOT prevent the drain from occurring in the other cells! ALL CELL HOLDERS MUST BE EMPTY WHEN THE RIG IS NOT IN USE!

It should go without saying that nothing “dumb” shall be done with any lithium cells at any point, especially when a cell is under load (DUT), that risks the safety of people and/or the research rig. Hitting, puncturing, burning/heating/freezing, overloading, shorting, or otherwise subjecting lithium cells to abnormal conditions can be deadly to you and your colleagues. Exercise great care and be on high alert when working with this rig and with lithium batteries in general!

# Introduction

This user guide goes over all the information related to the use of the lithium battery research rig. The rig was created over a span of 2 academic years, starting as a semester 7 project in the 2022-2023 academic year. The S7 project ended with only a general schematic of the intended final product and the PCB design of the battery monitoring board. As part of CSA 1 and 2 over the following 1.5 years, the project was worked on by 2 students in their free time in-between internships and work. The rig gradually took shape with additional functionality added to make it more feature complete. As of writing this guide, the rig is only missing a controllable mechanical load in order to be a fully functional EV simulation device. The rig is intended to be used by Fontys students to study health and performance of lithium batteries and by extension, EV battery packs. There are many topics of interest for research and development, but they will be discussed in a separate guide. The GitHub repository hosting this project includes all source code, CAD files, schematics, data sheets, etc. that were used in the execution of the project.

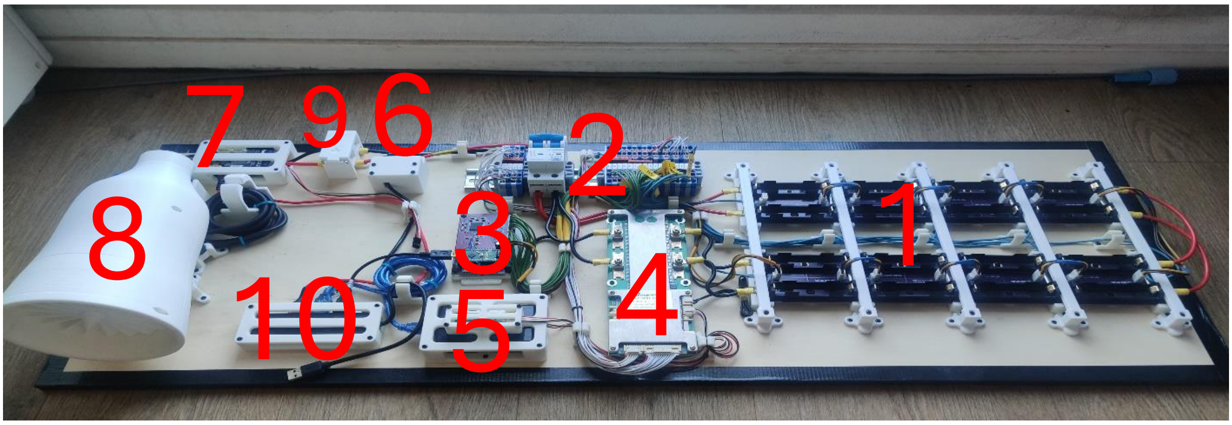



*Figure 2: Battery Research Rig Setup.*

Based on the figure above, a short description of each main component of the rig can be found below:

1. **Battery cell stack**
   The rig is designed around an 8 series cells with 2 parallel for each (8s2p) configuration of 18650 Li-Ion/Poly/FePO4 batteries. This configuration and slight variations of it (+-1/2

cells in series or parallel) are commonly used for smaller personal EVs like E-bikes, E-scooters, E-skateboards, etc. The cells are laid out in quick-swappable cell holders on custom PCBs in 2 sections of 4 series PCBs. Pay attention to the direction of the sections – they are mirrored symmetrically, meaning the positive and negative side invert between the sections. Pay close attention to the polarity of the cells when installing them in the holders. Each battery holder PCB also has a digital temperature sensor installed underneath each cell. These are connected to the battery monitoring board via their dedicated 3-wire cables in daisy-chain configuration.

2. **Distribution "box"**
   All cell taps (wires to each individual cell) are wired to a dedicated (for each cell) DIN-rail 4-port wire socket. The battery monitoring board and BMS connect to the box with the appropriate wiring scheme for correct operation of each device. Each wire socket has 1 port left uncopied, for future use (see R&D Guide). The box also includes a main DC-rated circuit breaker between the battery stack and any loads.
3. **Battery Monitoring Board**
   Customed designed PCB housing a MAX14921 16-cell battery monitoring IC, a MAX11161/11163 16-bit ADC, and a socketed Arduino Nano MCU. The Arduino Nano interfaces with the 2 MAX chips via SPI to get battery and cell voltage data. It also fetches temperature data for each cell from the daisy chain and the current sensor data for total battery pack current. The data is transferred to PC in raw format via Serial for processing in real time with the help of a dedicated MATLAB Application.
4. **JBD BMS**
   This is a commercial smart BMS that allows manual tuning of a wide variety of lithium battery protections. The BMS is required to keep the battery from unintentionally entering abnormal conditions. It's configured for Li-Ion cells currently but needs to be reconfigured when different cell chemistries are used.
5. **JBD Bluetooth and UART modules**
   These are the interfaces available for connecting with the JBD BMS. Only one of them can be used at a time since they share the same communication port on the BMS. By default, the BT module is connected, as it's easier to configure the BMS with the smartphone app. Alternatively, the UART box can be connected instead and used in conjunction with an open-source application, BMS Tools JBD, which allows for monitoring and control of the BMS from a PC.
6. **Current Transformer (sensor)**
   A split-core current transformer used to measure the total current going in or out of the battery pack. The sensor is bi-directional and rated for DC measurement. In the rig, the positive wire of the battery pack goes through the sensor. The sensor is rated for 5V power and signaling, providing an analog signal for measurements, which is read by the Arduino Nano.
7. **Flipsky VESC 4.12**
   The electronic speed controller used to convert the battery pack's DC power to 3-phase AC for driving the BLDC motor load. The VESC is extensively customizable, allowing for great load control when developing EV simulation runs.
8. **150W Electric Skateboard BLDC**
   A relatively big BLDC motor rated for up to 36V and up to 8A current draw to be used as an electrical load for the battery pack. The motor does not consume much power to reach its max RPM when no mechanical load is placed on it, which means it needs a

controllable mechanical load to make the rig fully operational. Until such a solution is created or found, a turbine-type engine was built around the motor to keep it cool (it heats up when running for a while with no load).

9. **Battery DC I/O Port**
   Underneath the plastic cover are 2 banana plug receptacles coming from the battery positive and negative wires. The port can be used for a variety of things, such as charging the whole battery stack, attaching external loads, or just probing for measurements.
10. **USB Hub**
   Since the BMS, the battery monitoring board and the VESC all have USB interfaces for PC communication, a small USB hub is incorporated to simplify the connection to a PC for use with the rig.

# Preparation for use

This chapter covers the steps to take from "cold and dark" to configure the rig for use. When you begin work, the rig should look like the figure below, with no cells installed and DC circuit breaker in the OFF position.

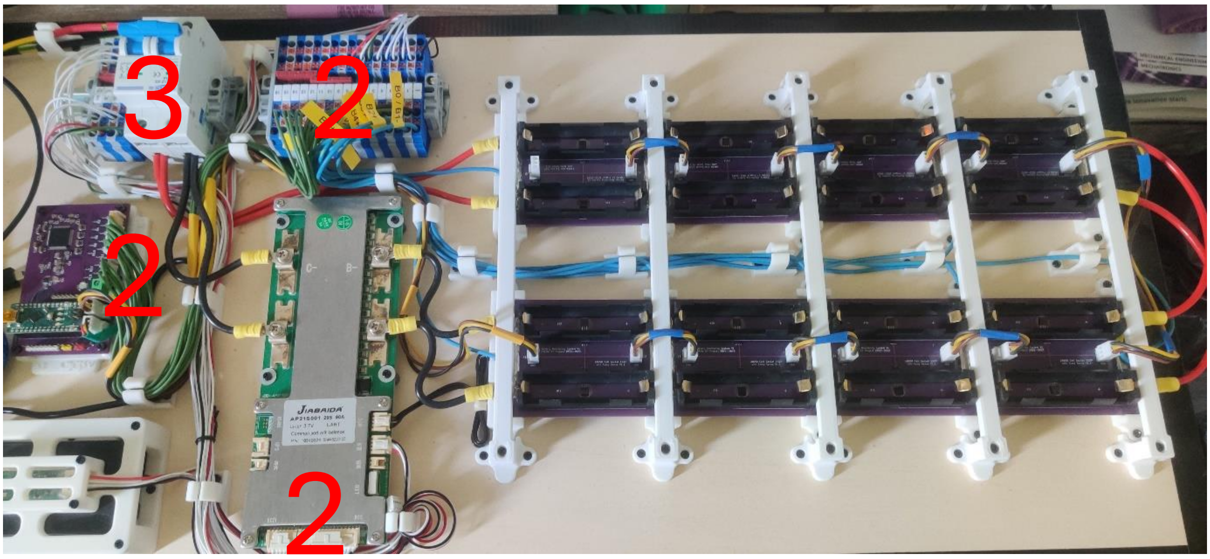



*Figure 3: Cold and dark battery rig.*

1. First, inspect the whole rig for any abnormalities, disconnected wires, etc. Once you make sure everything is looking proper, proceed with cable check.
2. Check all cables – All cell taps should be connected to their proper position in the cell stack and the distribution box. Check the BMS and battery monitoring board cables harnesses are connected.
3. With the DC breaker OFF, battery cells may now be inserted into the slots.
4. Insert cell into each holder, starting with cell 0 (closest to negative wires). If only 1 cell per segment in the series stack is used (so, 8s1p configuration), populate either only the BT1 or BT2 slots for each cell holder in the series. Pay attention to cell polarity!

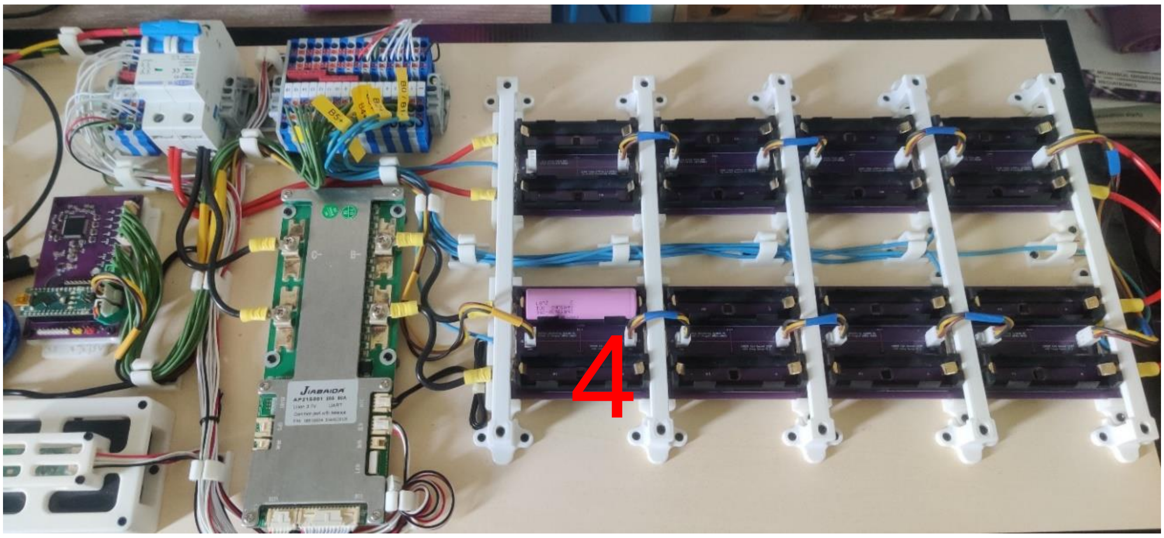



*Figure 4: First cell insert.*

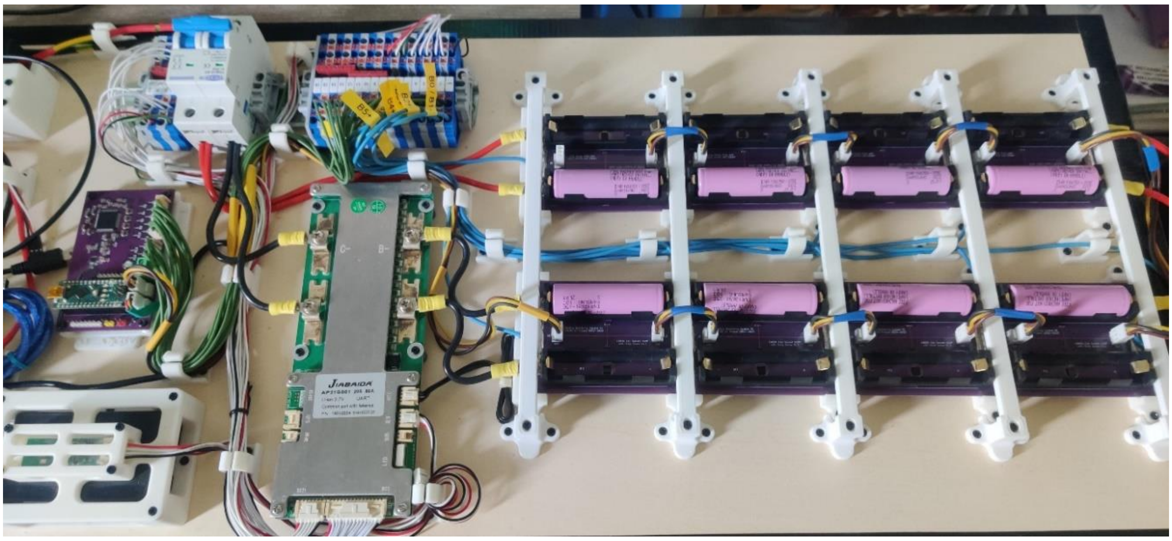


*Figure 5: All cell holders populated for 8s1p configuration.*

5. When all cells have been installed, connect to the JBD BMS to check if all cells are correctly detected.


*Figure 6: JDB BMS Homescreen after power up. All 8 cells detected.*

6. Once the pack is correctly installed, the USB hub can be connected to a PC to power up the Arduino Nano on the battery monitoring board and provide an interface to the JBD UART box and the VESC.
7. The MATLAB Battery Monitor PC application can now be started and a connection with the Arduino Nano on the monitoring board can be established. The instructions for using the application can be found in the next chapter.
8. With the PC application configured and connected to the battery monitoring board, testing runs can executed.
9. Check that an adequate load is correctly attached to the DC circuit breaker. If all is good, flip the circuit breaker to the ON position. The VESC should light up and appear as a new COM port on the PC connected to the USB hub.
   NOTE: If the load/VESC does not power up, check that the JBD BMS has switched on its Discharge FETs.
10. With the load configured and online, the battery rig is ready for science. Students shall prepare a test plan or protocol to conduct and commence its execution at this point.

# Battery Monitor PC Application

In order to process the incoming battery data from the monitoring board, a special MATLAB App was developed. The application polls the battery monitoring board at a user defined polling rate (0.5s default, 0.2s minimum) for all battery data that it collects (voltage, current, temperature) in raw format. The data is converted to usable values and displayed on 3 MATLAB graph windows. The application is packaged both as a MATLAB App module and as standalone application installer that includes a MATLAB runtime necessary to run. So MATLAB is technically not required to run the app. The main functions of the app are described below. Most fields and widgets in the application also show a helpful tooltip when hovered upon with the mouse pointer to guide the user.

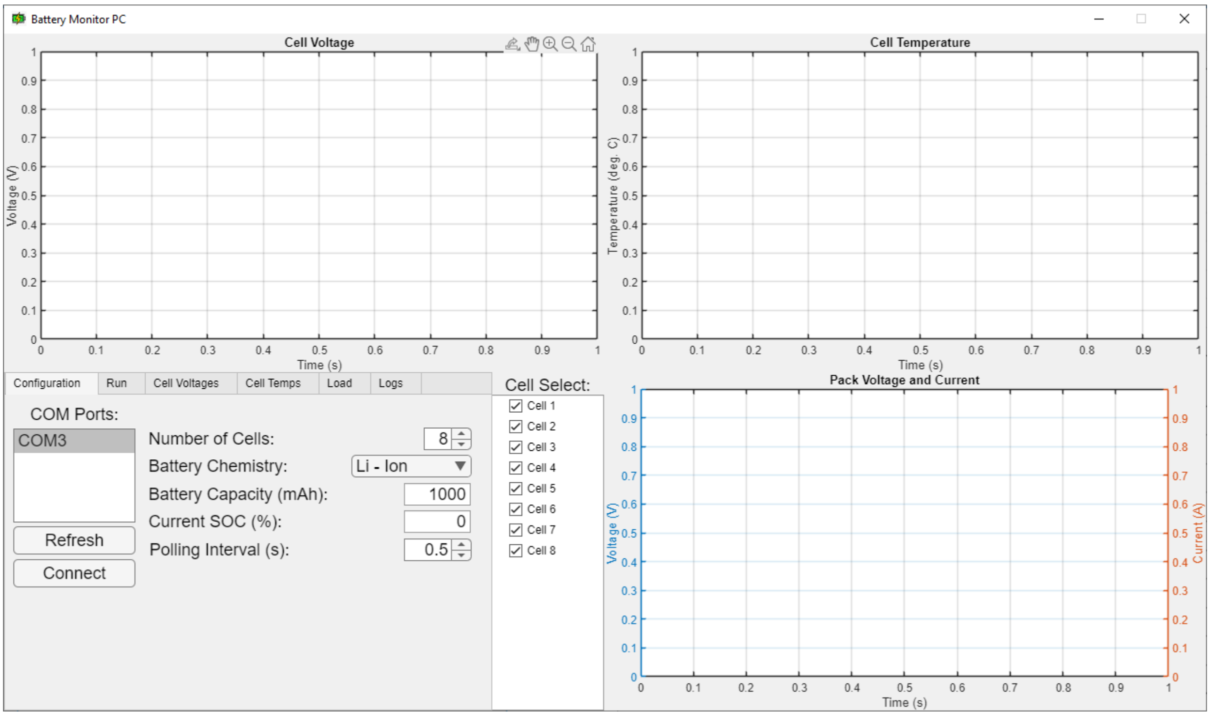



*Figure 7: Battery Monitor PC Configuration tab.*

The figure above shows the interface of the Battery Monitor PC application when it's launched. The top left and the two right quadrants of the application window are permanently occupied by the voltage, temperature and voltage/current graphs. The bottom left quadrant is where all the control UI is located, separated into tabs that group related functionality and/or data displays together. If the battery monitoring board is plugged in, it will show up as a COM port in the list at the left of the Configuration page. To the right of it are the parameters for the cell stack currently installed in the rig. All values except current SOC and rated battery capacity MUST be configured before connecting to the Arduino on the monitoring board, as they are required for configuration of the Arduino and will become locked after hitting “Connect”. Once connected to the monitoring board Arduino, cyclic data polling will start at the specified polling rate, but the graphs will not work at this time. To go easy on the PC running the app, the graphs are only going to start updating once a test run is initiated via the Run tab. If it's needed to check battery data before starting a run, the fields in the Cell Voltages and Cell Temps tabs will always update any time there is new data available. The Cell Select window scales automatically with the number of cells selected and only controls which cell data is displayed on the graphs. The data for all cells up to the selected number of cells is always retrieved from the Arduino and written to the file log, if enabled. Once configured and connected to the Arduino, the Run tab becomes the main window for the bottom left quadrant of the app.

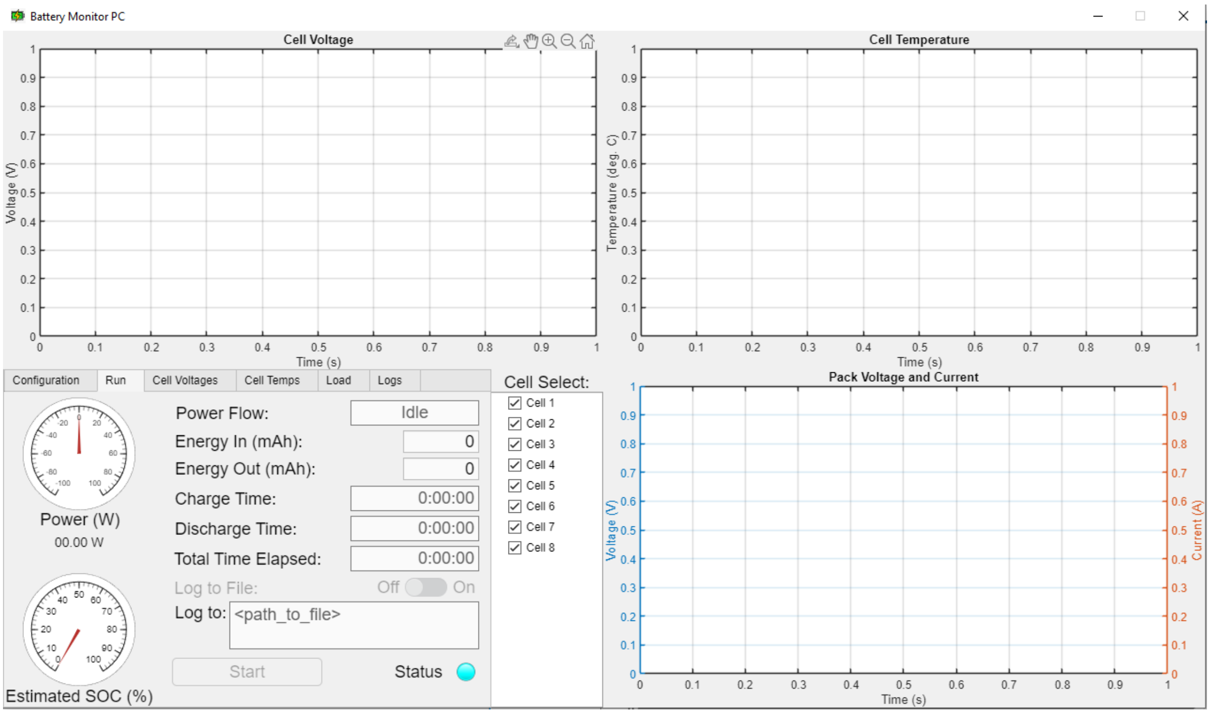



*Figure 8: Battery Monitor Run tab.*

The run tab shows the information relevant to the test run being conducted, such as timers, energy meters, a state-of-charge (SOC) estimate and power flow. The tab also contains the interface for configuring file logging. If the Log to File button is flipped to On, a prompt will pop-up to select file name and location for the log file. Once a valid file and location are successfully specified, the application will be configured to record all incoming battery data after the Start button is pressed to the specified file in a csv format. The log file can then be processed with MATLAB scripts to recover test run data. After pressing the Stop button (Start button turns to Stop button when pressed), a test run is concluded and the log file (if enabled) is closed. The Log to File switch is reset back to Off. A new log file must be specified by repeating the procedure for any subsequent test runs. During a test run, the 3 graphs will start updating at a rate close to real-time with the incoming voltage, current and temperature data.

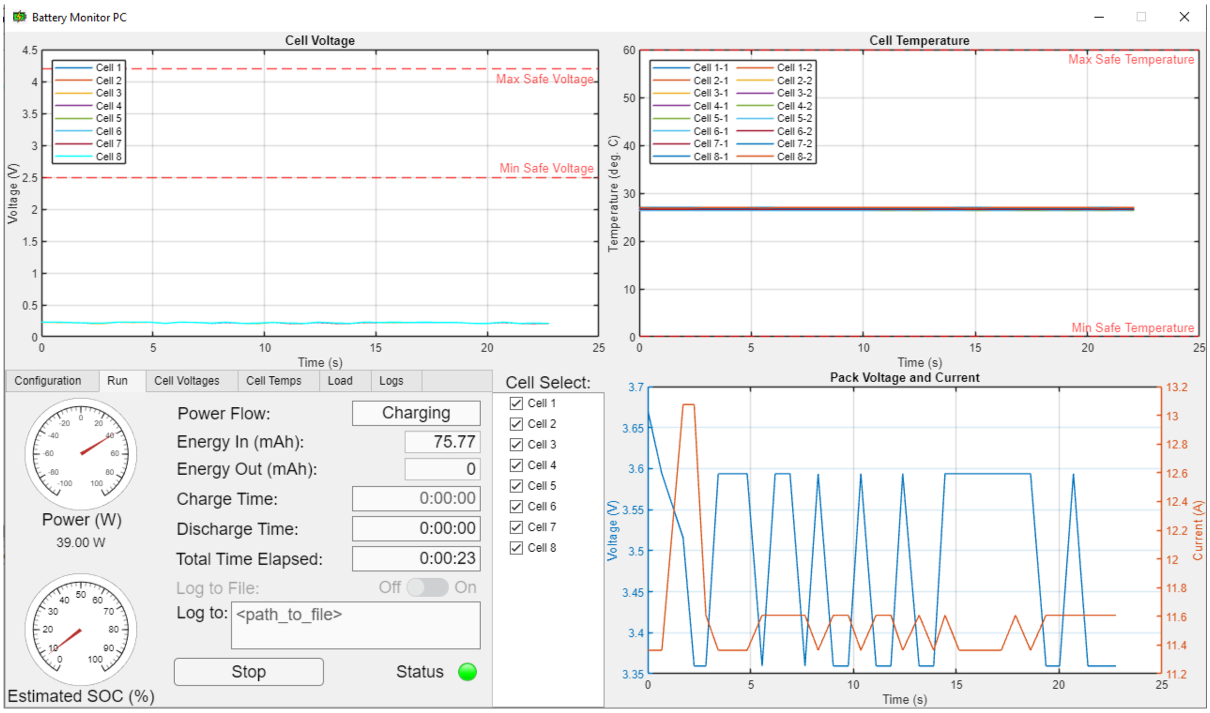


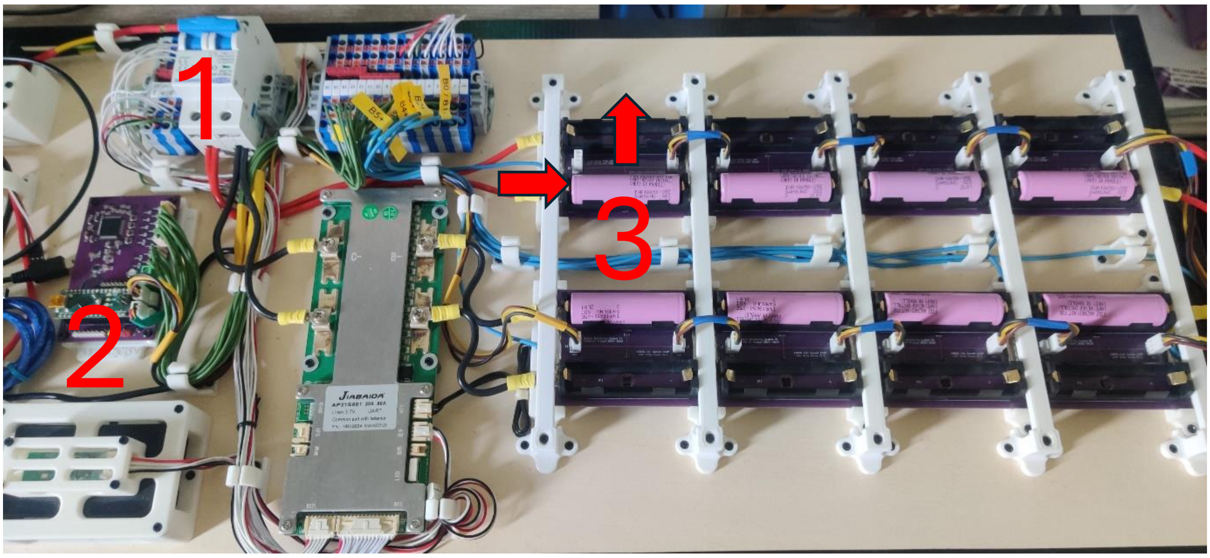

*Figure 9: Battery Monitor Run tab while running.*

The safe voltage and temperature lines on the top quadrant graphs are determined by the selected battery chemistry in the Configuration tab at the start. There are also a load control tab and logs tab in the control quadrant, but these were not implemented due to time. The visualization is intended to help students quickly observe the diverging behavior of cells in a battery pack that occurs due to a variety of factors such as aging, performance degradation with use over time, manufacturing tolerances, etc. More about the research use of this rig can be found in the R&D guide.

Note: It is recommended that an ongoing run is stopped and the Arduino disconnected from the Configuration tab before closing the application, but the app will try to handle that for the user if they forget.

# Preparation for storage

When a testing session has concluded, the rig must be prepared for storage. The sequence generally follows the setup for use sequence, just in reverse order.




1. Disconnect all loads by flipping the DC circuit breaker to the Off position.
2. Unplug the USB Hub and/or monitoring board Arduino from PC.
3. Start dismounting battery cells from the start, beginning with the top cell (8) and working back to cell 1. The easiest way to get a cell out of the tight holders is to push it back with a finger from the positive terminal side back towards the negative and grip the cell in the middle with the other hand, pulling/pushing upwards to release the cell.
4. Once all cells are out of the rig, make sure to tie any loose cables and wires and put the rig back in its nylon bags and put it away in safe location. No prolonged exposure to direct sunlight!

# Storage

When the rig is in storage, there must be no cells left in it in any cell holder! The DC circuit breaker must be in the Off position and the rig should be covered to avoid dust accumulation in tight spaces. Since there are a lot of 3D printed parts on the rig, prolonged exposure to direct sunlight, high temperatures or humidity are to be avoided. In general, the rig should be treated with care both when in use and when in storage.

Any battery cells dedicated for use with this rig should be stored separately, at 30% state-of-charge and periodically topped up with a charger back to 30% if the rig is dormant for a while.

# BMS

The rig utilizes a commercially available smart BMS for lithium batteries from Chinese manufacturer JBD. Smart BMS allows for pretty much all protection settings (trigger thresholds, protection trigger delays, protection recovery thresholds, etc.) to be manually configured to an appropriate value for the specific lithium battery chemistry used on the rig. The BMS is included in the rig to protect the battery cell stack and students in case something unplanned happens. However, for some interesting tests that can be performed, it would be desirable to put the battery cell stack in what would normally be considered abnormal condition, causing the BMS to trigger protections. Therefore, a smart BMS is required. The JBD BMS is widely used for personal DIY EVs and EV battery packs, as well as solar energy storage server rack LFP batteries. As a result, there is a free utility available on GitHub for connecting to JBD smart BMS boards from a PC (via the

UART box). The utility can read all measurements and settings from the BMS and even log battery data to csv files. This is very useful in case there are problems with the custom battery monitoring board. However, for most of the time, it would probably be easier to operate the BMS via its mobile app (available from JBD website). Below is a quick guide for it:

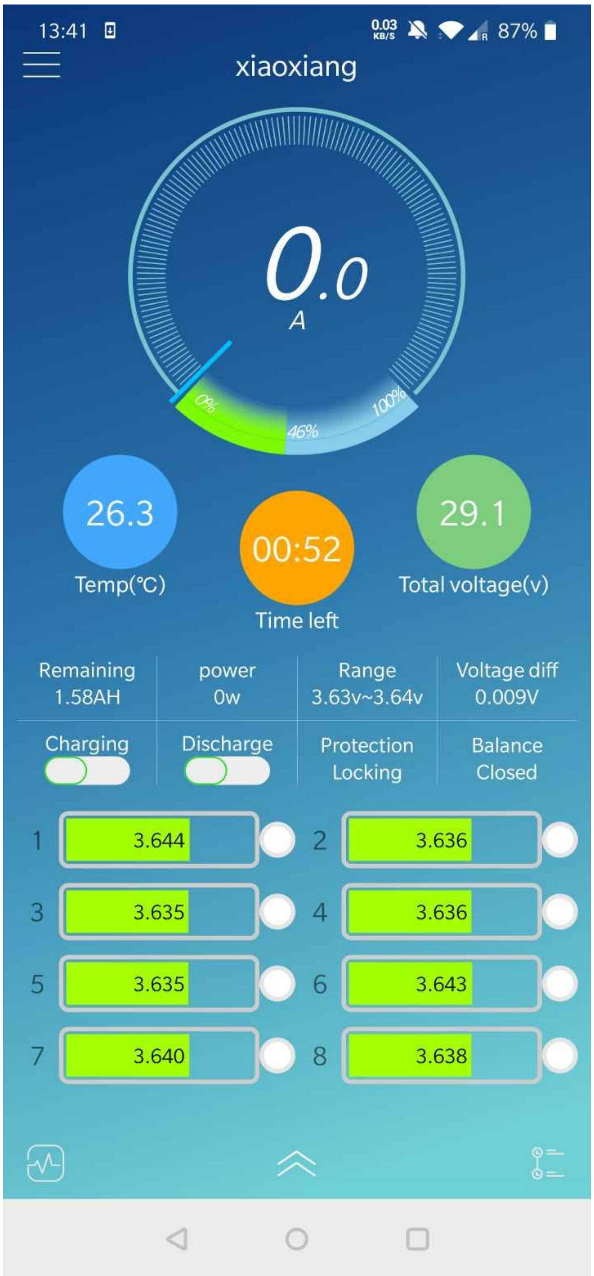



*Figure 10: JBD Smart BMS mobile app dashboard.*

On the figure above is the home page of the mobile JBD BMS app (Xiaoxiang). The page shows the most important battery data, such as cell voltages, pack voltage and current, temperature, and charge/discharge control. Pressing on the burger menu in the top left corner opens the app navigation pane:

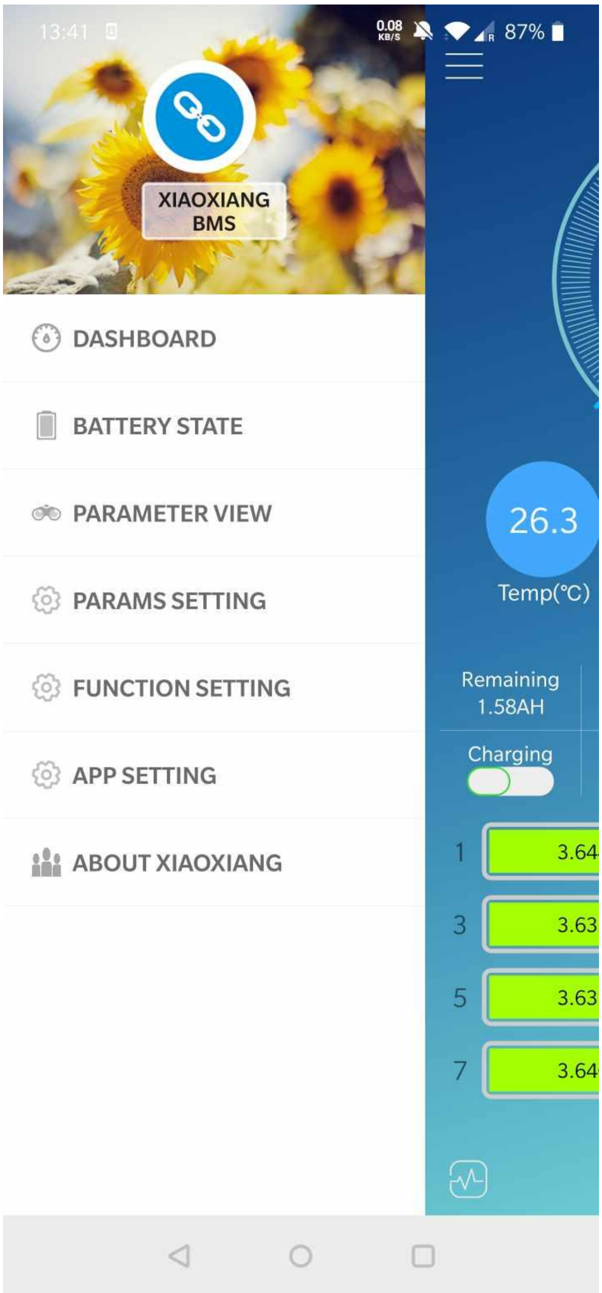



*Figure 11: JBD Mobile app navigation pane.*

From here, the most important menus are the parameter view/setting (they lead to the same page) and function setting. The parameter menu is self-explanatory, example figure below.

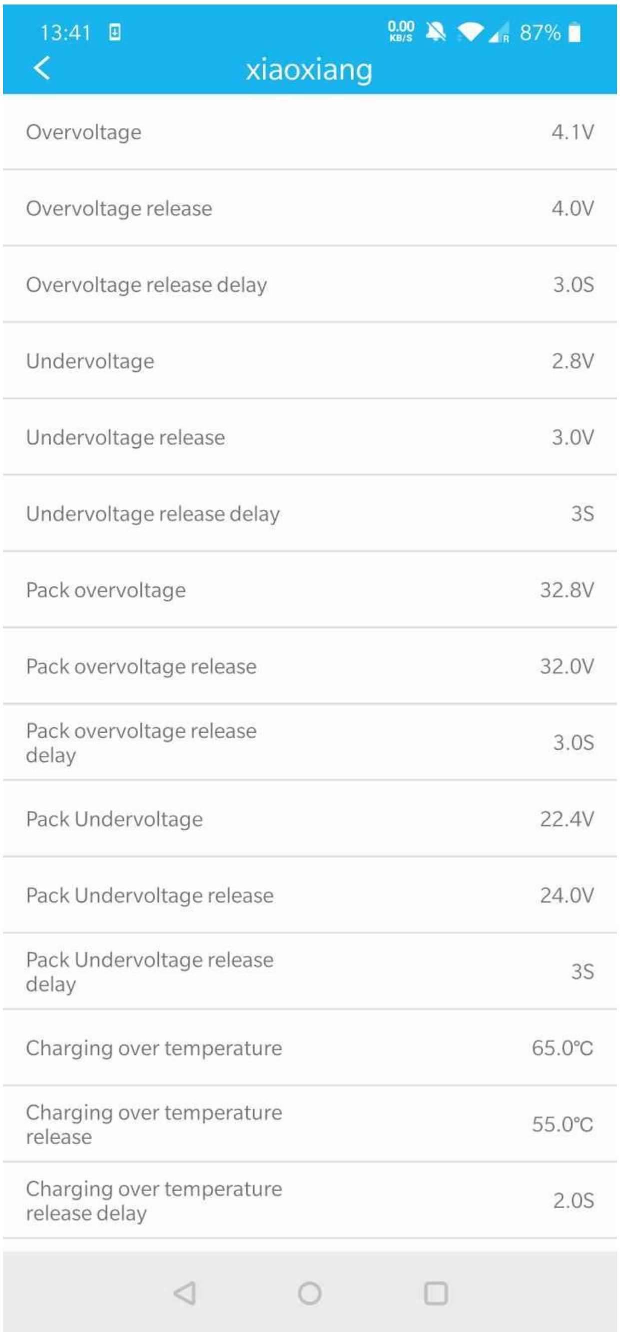



*Figure 12: JBD Mobile app parameter settings page.*

The settings on the figure above are for Li-Ion cells with an added safety margin for student use. For example, the BMS is configured to trip overvoltage protection at 4.1V, instead of the 4.2V standard for Li-Ion cells. This configuration is set on the BMS at the time of submission of the project so it may no longer be the case in the future. Always check the BMS parameter settings are set correctly for the target lithium chemistry variant before commencing test runs with loads/chargers!

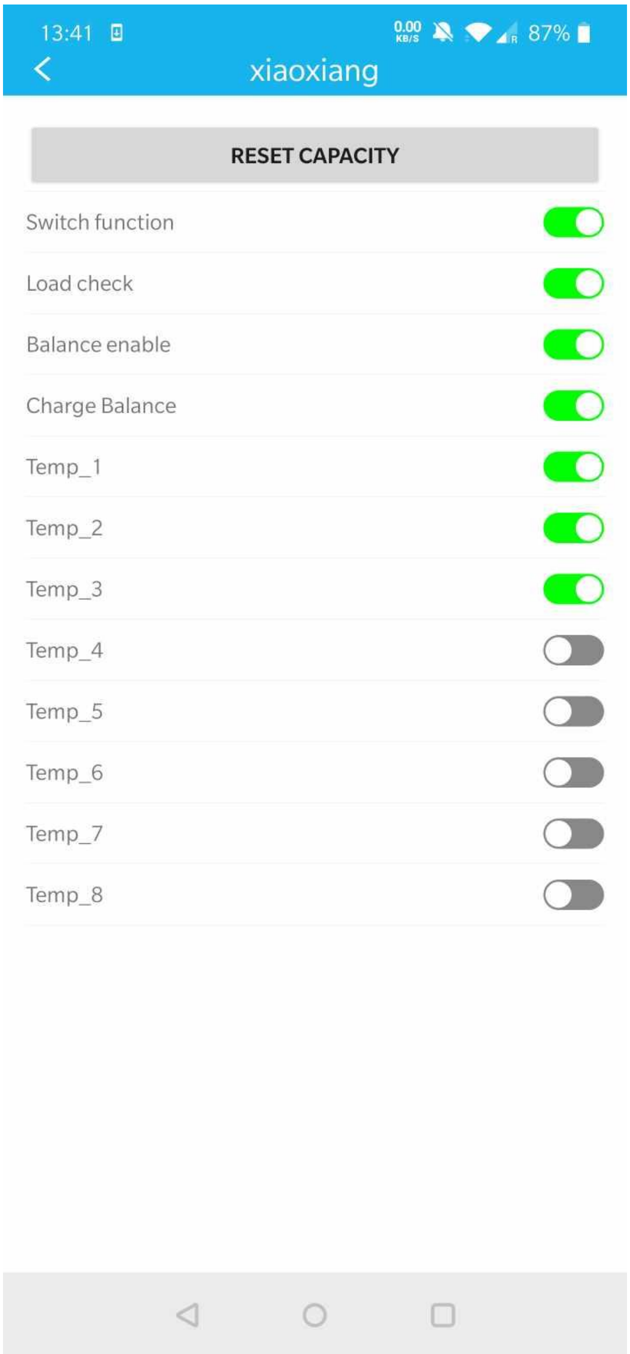



*Figure 13: JBD Mobile app function setting page.*

Finally, the figure above shows the function setting menu. Sometimes, if the battery is not connecting to the load, it's possible the "Switch function" at the top of this menu has toggled position. Toggle it and check if the load powers on.

# Temperature sensors

*Figure 14: 18650 Lithium cell holder PCB with built-in DS18B20 digital temperature sensors.*

The rig incorporates a number of DS18B20 digital temperature sensors, placed in the battery cell holder PCBs underneath each cell slot. The sensors are daisy chained between battery holder boards via custom 3-pin cables, carrying +5VDC, GND, and a Data wire. All sensors are wired in parallel along the path, sharing the single data line. The DS18B20 implements the OneWire communication protocol, which uses unique device IDs to interface with the sensors in the chain. That means the sensor IDs need to be read from them and hardcoded into the monitoring board Arduino code to correctly match the data from each sensor to its corresponding cell. To do this, a little Arduino program is included, courtesy of ChatGPT 4.0, for detecting all temperature sensors connected to the monitoring board and printing to the Serial monitor their unique device IDs and current temperature readings. This program will come in handy if a sensor or entire cell holder board is damaged and needs to be replaced. Simply connect the board with the special 3-pin cable (yellow/brown/black) between the battery monitoring board and one end of the cell holder board and load the program on the Arduino. Open the Serial monitor at 9600 baud rate to monitor the ID and readings print outs and hold one sensor with your fingers for a few seconds to identify it by its changing temperature reading. Write down the unique device IDs and their cell position (BT1/BT2 on the cell holder board, cell position in stack when installing in the rig) and paste them into the BatteryMonitorMCU main program in the correct places (labeled with comments). Additionally, more sensors can be attached to the rig for flexible use if needed by connecting them to the open 3-pin port on the top cell of the stack (cell 8). Again, their ID(s) will have to be retrieved and coded in the main program to work. A modification of the MATLAB application will also be required. More on that in the R&D guide.

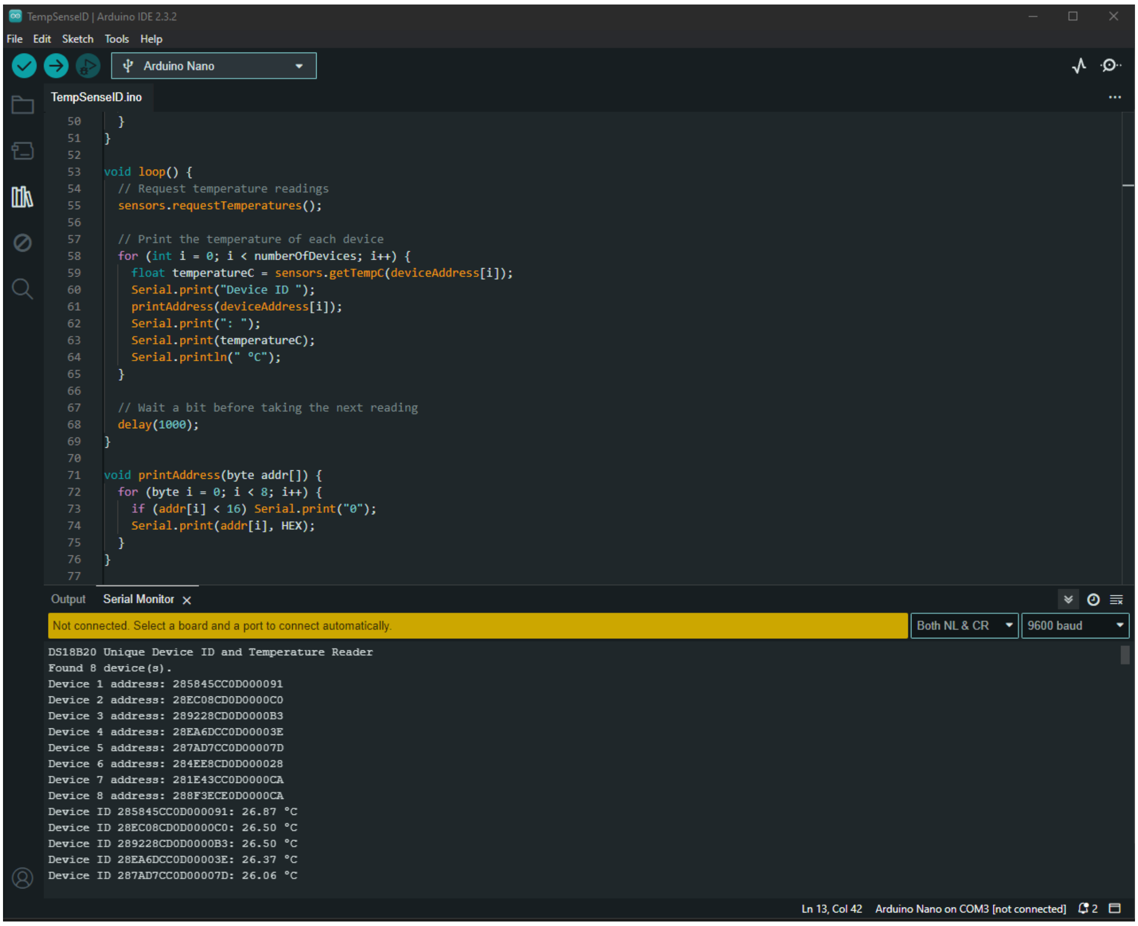



*Figure 15: Arduino program for detecing the unique device IDs of DS18B20 temperature sensors.*

# Load

To simulate the electrical load on a battery pack, a dynamic, preferably programmable load is necessary. To that end, a 150W BLDC motor typically used in electric skateboards is installed on the opposite side of the plywood board. The motor is run by a Flipsky VESC 4.12 electric speed controller (ESC) provided by Fontys. The VESC is wired with ring terminals on both its input and output side to allow easy reuse and change. Using the microUSB interface, the VESC can be configured and controlled from a PC with the VESC PC tool. The VESC tool can downloaded from the VESC website after registering a free account. There are many ways to setup the control of the motor, and this task will be done by future students assigned to this project. The end goal is to be able to create “load profiles” where the motor simulates the operating conditions of a real electric skateboard so that its electrical effects on the battery pack can be studied. More information on this can be found in the R&D document.

# Resources

All resources used in the creation of this project and/or required for its use and further development can be found in the following public GitHub repository:

https://github.com/sashkoBeats/Lithium-Battery-Research-Rig