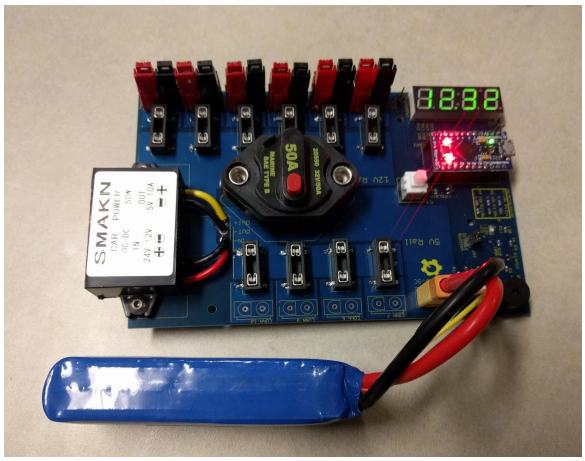
PiE PDB

The Life, Care, and Feeding of the PiE Custom PDB



Features

- Overall Maximum Current: 55 Amps
- Per-Load Current: 15 Amps
- Effective Resistance: ???
- 5V on-board Regulator, 10 Amps
- Overvoltage Undervoltage Protection
- Cell Imbalance Protection
- Overcurrent Protection
- USB Communication
- 7-Segment Display Array
- Audible Warnings
- Reverse - Polarity Tolerant
- Low Leakage Current

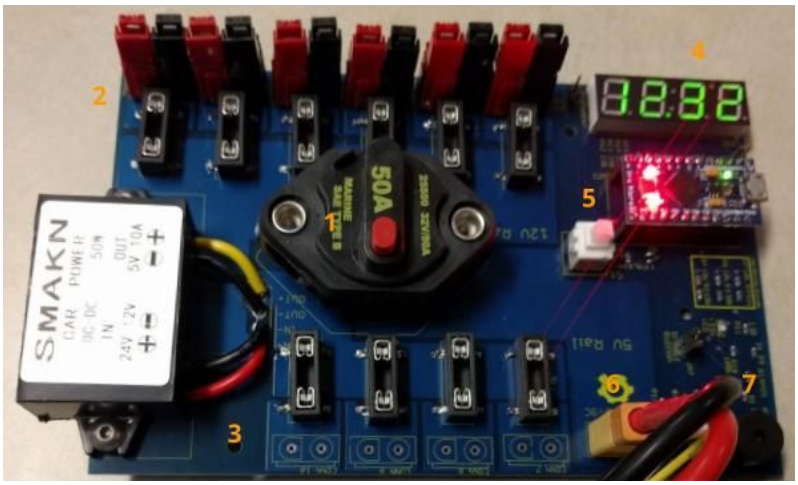
Overview

The PiE PDB is PiE's custom-designed Power Distribution Board designed to meet key criteria laid out by PiE Staff. The PDB interfaces a Three-Cell Lithium Polymer Battery with power consumers, an on-board protection system, and a master circuit breaker, up to a 50 Amp limit.

The PDB provides two separate power rails: an unregulated direct connection to the battery (12 V), and a regulated 5V rail. All loads have current protection in the form of interchangeable automotive fast-blow fuses. All rails are designed to support up to 15 Amp Currents on any load, but the practical power limit is set by other criteria.

The battery protection is based around an on-board Arduino Pro Micro (ATmega 32u4) and custom-designed circuits. The battery protection system protects against undervoltage, overvoltage, and imbalance conditions on the battery. The protection system provides a visual and audible indications of the battery status. The ATmega can also report the battery safety condition over USB communication.

Layout & Functions



1. Circuit Breaker
2. Battery Rail
3. 5V Regulated Rail
4. Voltage Display
5. Arduino Pro Micro
6. Battery Connection
7. Balance Cell Connector

Ratings & Specifications

Specifications

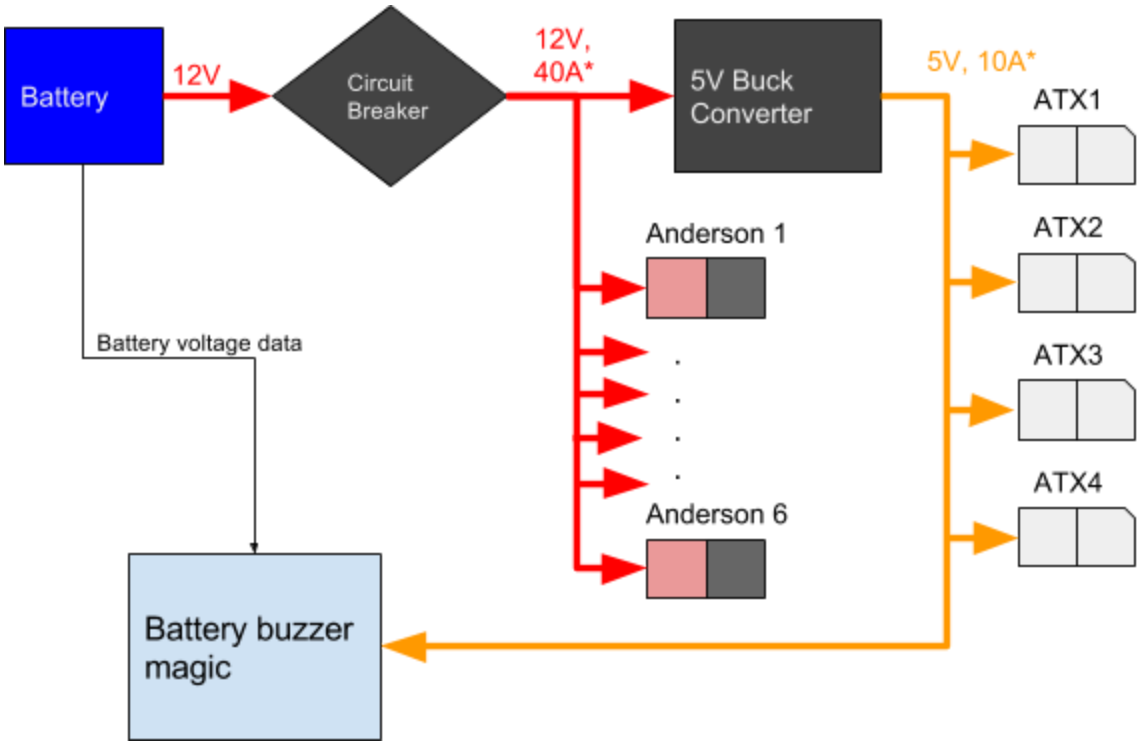
Battery	Typically, Turnigy 2.2mAh 25-35 C
Nominal Battery Voltage	11.1V (12V)
Battery Rail Connections	6
Regulated Voltage	5V
Regulated Rail Connections	4
Voltage Sensing Accuracy	+/- 15mV
Buzzer Sound Level	90 dB
Fuse Type	ATC automotive Fuses
Size	7.17” x 4.00“
Copper Thickness	2 oz
Mounting	3 #4-40 Screw Holes

Ratings

Maximum Battery Voltage	??
Maximum Battery Current, Sustained	55 Amps
Maximum Battery Current, Instantaneous	?? Amps
Maximum Current on any Battery Rail Connection	15 Amps
Maximum Current of 5V Regulated Rail	10 Amps
Leakage Current when turned off	< .1 uA
Power draw when turned on	

Thermal Information

Block Diagram



Power Rails

12V Rail

The so-called 12V rail is directly connected to the Battery through the Circuit Breaker, and is not regulated. The 12V rail is designed to be the primary source of power for large loads, such as motor controllers.

The 12V rail has 6 available connections. Connections are made through the use of 15 Amp rated Anderson Connectors. Fuse holders are available for ATC Automotive fuses. These fuse holders cannot be bypassed, so the use of fuses is required. These connectors, when properly assembled, are mechanically prevented from connecting in a reverse polarity situation. The polarity convention was chosen by PiE staff to mesh with pre-existing hardware.

5V Rail

The 5V rail is powered via a COTS switching regulator, which draws power from the battery. Mounting is provided for the switching regulator on the PDB itself.

The 5V rail has 4 available connections. Connections are made through the use of Molex Connectors. Fuse holders are available for ATC Automotive fuses. These fuse holders cannot be bypassed, so the use of fuses is required. Molex connectors have a polarity, and are mechanically prevented from being connected backwards.

Due to the capacitance of the regulator, a discharge resistor is provided to bleed all voltage when the PDB is turned off.

Circuit Breaker

Power control and overcurrent protection is provided via an integrated Circuit Breaker. This circuit breaker is mounted to the PCB for all-in-one access.

Battery Connectors

Power connection from the battery is made via a standard XT60 connector. This connector mechanically prevents the battery from being connected backwards.

Balance cell connection is made via a .1” JST Connector. This connector discourages, but does not absolutely prevent, reverse polarity situations. The battery protection system’s circuits have been specifically designed to avoid damage in a reverse polarity situation.

Battery Protection System

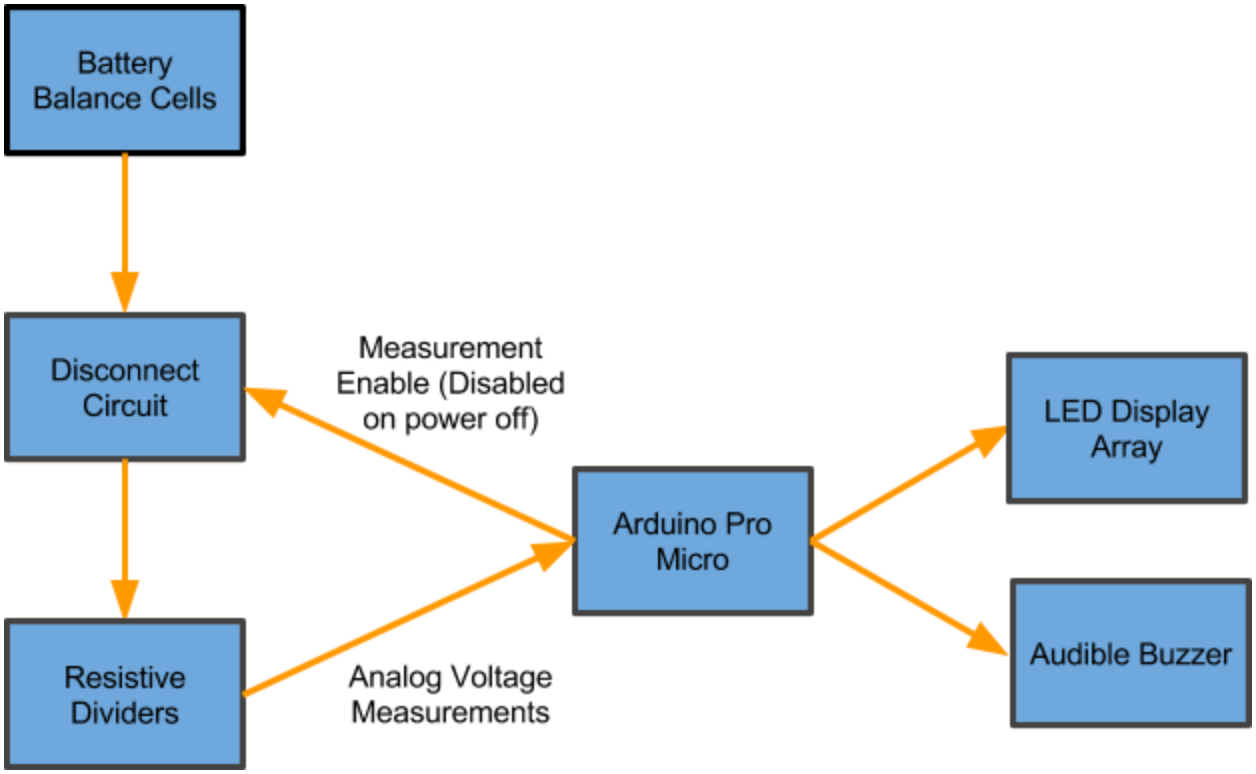
Overview

The battery protection system (or “Battery Buzzer”) provides undervoltage, overvoltage, and cell imbalance protection for the Lipo Battery. The system also has transient protection to guard against loading effects rapidly changing the safety state. The system can only warn the user, it cannot automatically disconnect the battery from use.

This system has been specifically engineered to draw minimal amounts of power when the PDB is turned off, to avoid discharging the battery to an unsafe level. The system also has visual and audible feedback to inform users to the state of the battery.

To use this system, the main power connector of the battery must also be connected. For flexibility and future proofing, the system is built around an Arduino Pro Micro (ATmega32u4). For reasons explained in the Appendix, both high-precision resistors and a calibration procedure are required for accurate voltage measurements.

Block Diagram



Safety Thresholds

Overvoltage Threshold, any cell	4.4 V
Undervoltage Threshold, any cell	3.3 V
Imbalance Threshold, between cells	0.3 V

Calibration

To enhance the accuracy of the battery voltage measurement system, it is necessary to calibrate the device. Consult the “Hardware Configuration & Setup” section for calibration instructions.

It is possible to use the system without calibration, but cell voltages may be inaccurate by 10%. The calibration must be done with a single arduino pro micro; if a different arduino is used, calibration must be redone. The calibration value is saved in nonvolatile EEPROM, so it need not be done on every power cycle.

There are three main sources of error in calculating the voltage of the battery cells: Value errors in the resistive divider, error in the ADC caused by reference voltage error, and ADC precision error. This calibration guards against the second kind of error. Consult the appendix for an error propagation analysis of the entire system.

Hardware Configuration & Setup

Flashing Procedure

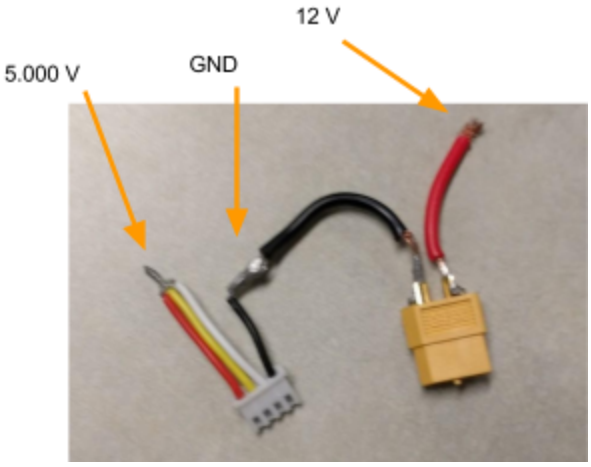
To flash the Arduino Pro Micro that runs the Battery Buzzer, use ./flash.sh, found in the hibike folder of PieCentral. The most recent code is stored here.

1. The flashing procedure can be done at any
2.], but it is preferred to be done after the PDB is assembled.
3. Plug the soldered arduino into the PDB.
4. From the hibike folder of PiE Central, using an ubuntu computer,
    a. Run “sudo ./flash.sh BatteryBuzzer”
5. Should see “AVR Done” if the upload was successful.
6. Then, it would be ideal to run the Calibration or Test procedures.

Calibration Procedure

1. Connect the Calibration Jig to the PDB.
2. Power the Calibration Jig with 5.000 Volts and 12V, from a DC power Supply.
3. Turn on the PDB by closing the main breaker.
4. Short CALIB_JUMPER with a jumper or a female-female wire.
5. Remove the jumper on CALIB_JUMPER.
6. Confirm that calibration information is displayed on the LED display.

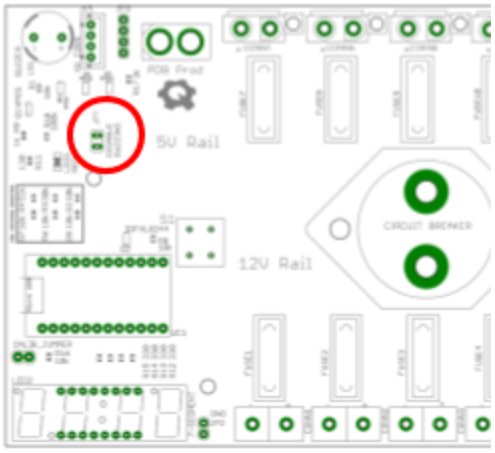
Calibration Jig



Temporarily Disable Buzzer

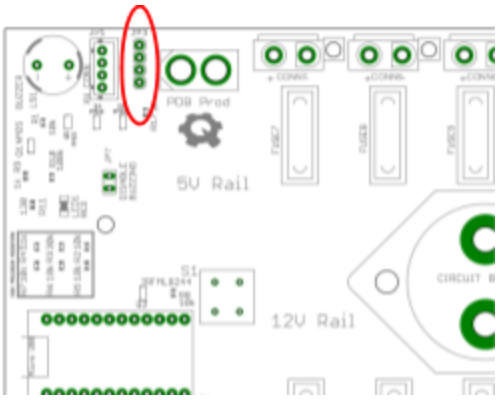
Developing software for the PDB can often be frustrating, because the buzzer on the PDB will continue to buzz loudly.

To temporarily disable the buzzer, connect a jumper across the JP7 Pin Header. This disables buzzing by grounding the Gate of the N-MOSFET that drives the Buzzer.



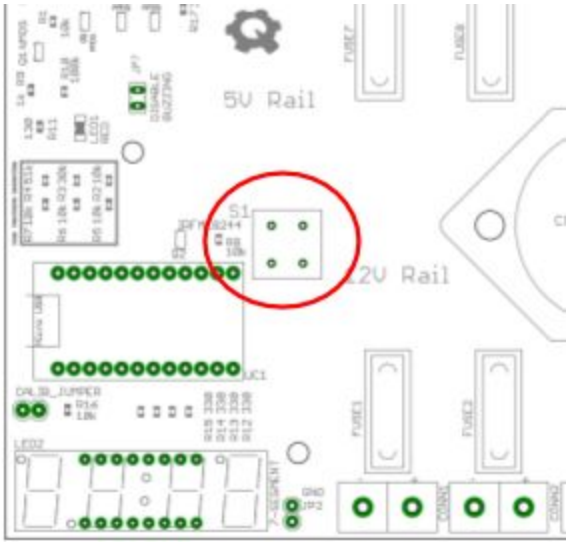
Mount for COTS Battery Buzzer

For backup purposes, there is a .1” header installed in the PDB directly connected to the battery cells. Solder in a COTS Battery Buzzer here if desired.



Secondary Power Switch

To check the battery voltage without turning on the whole PDB (and thus Robot), firmly press and hold S1. This action will turn only the battery buzzer portion of the PDB on. Release S1 when done.



Testing Procedure

1. Program the Arduino Pro Micro.
2. Confirm that the LED display reads “8.8.8.8.” on boot.
3. Confirm that the Red LED is lit, and the Buzzer is buzzes temporarily, when PDB is powered.
4. Calibrate the PDB
5. Turn off the PDB using the main breaker. Confirm 0 current draw from power supply.
6. Plug a real Lipo Battery into the PDB. Using hibike_process.py, confirm that
    a. the PDB is calibrated
    b. Vcell1, vcell2, vcell3 are all approximately the values as measured directly from the battery by a DMM.
7. Visually inspect the board to see that all power connections are soldered.

Interfacing with Software

The battery protection system uses an Arduino Pro Micro to perform measurement and display functions. This arduino can be communicated with over USB using PiE's Hibike communication framework. There are also versions of the firmware that communicate using ASCII print statements.

The hibike code that runs the PDB will eventually be found in the hibike folder of Pie Central, PiE's monolithic github repo.

Additionally, there is an older version of the software that uses an Arduino IDE. This version may be helpful if the PDB is to be used in a situation where hibike is not available, or for development work. This code can be found here.

Mechanical Enclosure

Assembly Notes

- Through hole soldering notes:
    - It's best to solder these in from shortest to tallest:
    - Shorter: battery connector (XH CONN/JP1), seven segment display (LED2), jumpers (JP7 and CALIB_JUMPER only!)
    - Taller: buzzer, Arduino headers*
- Buzzer:
    - Remove the yellow sticker / covering from the top of the buzzer after soldering. The yellow covering quiets the breaker.
- Procedure for mounting the Circuit Breaker:
    - Using dikes, clip off the black tab between the lugs on the back of the Circuit Breaker. The remainder should be filed flush with a fine file.
    - Use soldering irons to solder the Circuit Breaker's lugs into place. This may require two soldering irons to do effectively.
    - Use a wrench and locknuts to then secure the lug in place further. This is for safety reasons
- 5V regulator
    - Bolt the 5V regulator to the board first, before soldering the wires in. This will ensure that the wires are the right length.
- DNP
    - Do not populate S1, since students may accidentally use this functionality at inappropriate and possibly damaging times
    - Do not populate JP2, this is a debugging feature.
    - Do not populate the COTS battery buzzer, this is a backup feature.
    - Do not populate JP7, this is a development feature.

Known Issues

- The battery buzzer momentarily beeps when the PDB is turned on, even if the battery connected is safe.
- Placement of the COTS battery buzzer header is poor.
- The LED display is hard to read in bright light. Use of a scattering cover or diffuser built into the enclosure may ameliorate this problem.
- The Circuit Breaker footprint is not exactly correct - the current circuit breaker used is slightly offset.
- The wires for the buck converter are rather hard to solder.
- There are naming convention issues throughout the parts.
- The PDB cannot be made bigger due to Eagle Licensing issues.
    - At current size, PDB cannot be edited with the free version of EAGLE (requires an educational license to allow large board area).

- 6, 12V Power Connections may be insufficient for high-end robot designs.
- The voltage on IO2 (A2) varies wildly when CALIB (A3) is connected to 5V. The source of this error is unknown, but suspected to be in the Arduino Pro Micro itself. This error requires that calibration be done once Calib has been un-shorted, hence the “Press, and then remove” step in the calibration procedure.
- The correction for the 5V regulator is hard to solder.
- Consult the issues tracker for other, mostly hardware related, issues.

Design Philosophy

The PDB is the crux of the PiE Robotic Control System. The PDB was designed to be as reliable as possible, since damage to the power system can be very expensive and time consuming to fix.

One observation is that our intended users - High School Students - tend to connect all wires of the same type, whether they are intended to be connected or not. Therefore, all incompatible electrical rails use geometrically different connectors.

Future Plans

The current version of the PiE Custom PDB represents PiE's first foray into the design of a Power Distribution System. The functionality implemented on the PDB does not represent a huge change from PiE's previous PDBs. This scope was intentionally chosen, to maximize the chances of success when creating PiE's first custom PDB. The rest of this section will assume that there is no heretofore undiscovered design flaw in this PDB.

Future options for creating the PDB mostly center around making a more intelligent and user friendly PDB. There are a few specific avenues of advance: Better Fuses / Current Protection, Better Circuit Breaker, Soft-boot capability, More Intelligent Protection Systems, More Connections.

Circuit Breaker

Currently, the Circuit Breaker in use on the PiE PDB is the same circuit breaker used by PiE for years. It costs approximately thirty dollars, and is extremely bulky. Additionally, the current protection feature of the circuit breaker is rarely used. Needless to say, it is not ideal.

There are commercially available push-button & power mosfet packages, such as this one. It is unknown is there is a commercially available alternative in the PDB's power range. Additionally, relays may present an alternative which has better safety implications than a MOSFET - based system. Either way, a push-button controlled system may reduce cost and increase ease of use. It may also be leveraged to enable other systems, such as soft booting.

Soft Boot

In the current situation, there is no intelligence applied to the robot being turned on or off. If the battery is in an unsafe situation, the robot can still be turned on. This could further damage the battery, creating a further unsafe situation.

The issues are more numerous when looking at the robot's shutdown procedure. Robot has no control or warning that it will be turned off. The user just pulls the breaker, which promptly disconnects battery voltage from the system. For the linux-based processor of the PiE Kit, this instant power off has drastic implications. More than once, there have been files lost during updates because the Beagle Bone Black did not have time to write files to disk in a graceful shutdown process.

If there was a more complicated push-button system, PiE could implement a graceful shutdown process. When the Power off Button is pressed, the PDB's Arduino could inform the Beagle Bone Black, which would begin an orderly shutdown of the robot. The final step of the shutdown would be accomplished when the Beagle Bone Black informs the Arduino to pull power.

Fuses

The PDB uses automotive fuses to accomplish current protection of loads. PiE has previous experience with ATC automotive fuses, which caused them to be used in this PDB. Automotive fuses are known to not react fast enough to protect motor controllers and other complex electronics from hardware damage.

In future versions, it may be wise to replace the hardware fuses with other, more advanced options, such as current monitoring FETS, etc. Such a process would also give us instant across-the-board power consumption information, which would be useful to guide future power electronics efforts.

Intelligent Systems

The Battery protection circuit used in the PDB is completely home-grown. While it is successful, we now know that there are more flexible options. Manufacturers such as TI make specialty Li-Ion / Lipo battery protection chips which have high response rate overcurrent protection, coulomb counter, etc, built in. It would significantly reduce part count, and increase future flexibility, to use a chip such as TI BQ76920.

Connections

Finally, the PDB could use more battery connections. Game analysis teams suggest that 8 or more battery connections are needed, more than the 6 currently available.

Additionally, no analysis has been performed on the layout of the PCB. Currently, for minimal operation, connections need to be made on three sides of the PDB. This layout result in large wiring problems for students. Observations should be made to determine if a different layout of the PDB would be more optimal.

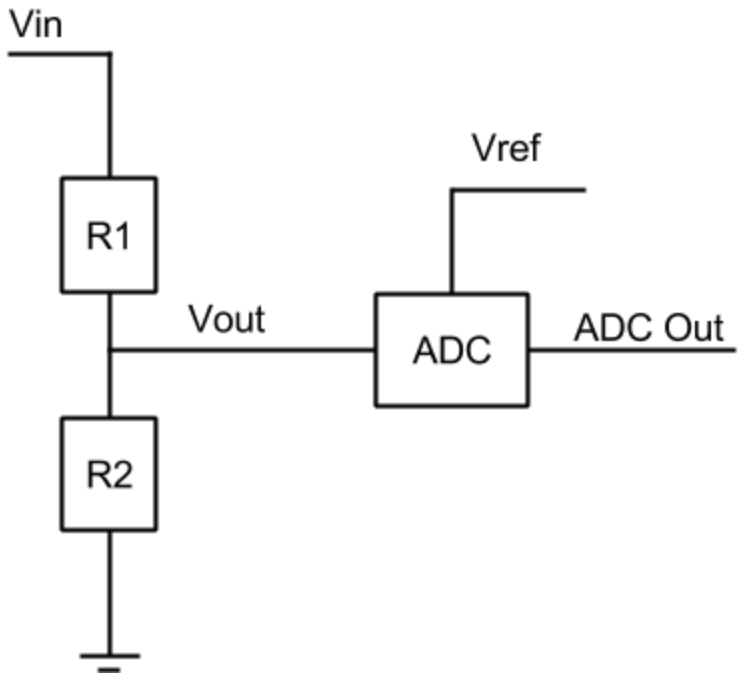
E-Prod Documentation Packet

See this link for PDF.

Appendices

Error Propagation Analysis

In order to validate the design of the battery voltage sensing circuit, an error propagation analysis is performed on the circuit design. Figure 1 shows a circuit diagram of the battery sensing circuit that is used for each cell.



As implemented on the PDB, Vin is connected to the battery cell voltage to be measured.

For a resistive divider, it is well known that,

$$V_{out} = V_{in} \frac{R_1}{R_1 + R_2},$$

Assuming that the current on Vout is negligible. Such an assumption is appropriate in this case, as the current into the ADC has been measured to be less than 0.003uA.

Rearranging equation 1, it can be shown that

$$V_{in} = V_{out} \frac{R_1 + R_2}{R_1} .$$

The input to the ADC will now be considered. An ADC works by comparing the input voltage to some known reference voltage, and then digitizing the result. Therefore,

$$ADC_{Out} = Round_{int}(N \frac{V_{out}}{V_{ref}}) ,$$

Where N is the number of ADC counts, and assuming that the Vout is between 0 Volts and Vref.

Combining equations, and neglecting the rounding error of the ADC, it can be shown that,

$$V_{in} = \frac{R_1 + R_2}{R_1} ADC_{Out} \frac{V_{ref}}{N} .$$

We will now conduct an error propagation analysis of equation XX, via the method shown on Page 76 of *Mechanical Measurements, 6th Edition.* Quoting the text:

> To estimate uy [The statistical uncertainty of a function f which has several input values x1… xn, all of which are also uncertain.], we assume that each uncertainty is small enough that a first-order Taylor Expansion of y(x1, x2, … xn) provides a reasonable approximation:

$$y(x_1 + u_1, x_2 + u_2, \ \dots \ x_n + u_n) = y(x_1, x_2, \dots , x_n) + \frac{\partial y}{\partial x_1} u_1 + \frac{\partial y}{\partial x_2} u_2 + \dots + \frac{\partial y}{\partial x_n} u_n$$

Therefore, the maximum expected error from the true value is as follows:

$$\varepsilon = y(x_1 + u_1, x_2 + u_2, \ \dots \ x_n + u_n) - y(x_1, x_2, \dots , x_n) = \frac{\partial y}{\partial x_1} u_1 + \frac{\partial y}{\partial x_2} u_2 + \dots + \frac{\partial y}{\partial x_n} u_n$$

In this specific case,

$$y(x_1, x_2, \dots , x_n) = v_{in}(R_1, R_2, ADC_{out}, V_{ref}, N) = \frac{R_1 + R_2}{R_1} ADC_{Out} \frac{V_{ref}}{N} .$$

By equation YY, the maximum expected error is as follows:

$$\varepsilon = \frac{-ADC_{out} R_2 V_{ref}}{N R_1^2} u_{R_1} + \frac{ADC_{OUT} V_{Ref}}{N R_1} u_{R_2} + \frac{(R_1 + R_2) V_{ref}}{N R_1} u_{ADC_{out}} + \frac{(R_1 + R_2) ADC_{out}}{N R_1} u_{V_{ref}} + \frac{-ADC_{out}(R_1 + R_2) V_{ref}}{N^2 R_1} u_N .$$

However, several terms can be eliminated from the above equation. Specifically, $u_N = 0$ because the number of ADC steps is accurately known by design. Additionally, $u_{ADC_{out}} = 0$ since this value is read accurately in software steps. Additionally, the error caused by the previously neglected rounding function of the ADC must be taken into account. This error is characterized by the following equation,

$$\varepsilon_{round} = \frac{V_{ref}}{2N} .$$

Therefore, the total error expected in the measurement setup can be expressed as follows,

$$\varepsilon = \frac{-ADC_{out} R_2 V_{ref}}{N R_1^2} u_{R_1} + \frac{ADC_{OUT} V_{Ref}}{N R_1} u_{R_2} + \frac{(R_1 + R_2) ADC_{out}}{N R_1} u_{V_{ref}} + \frac{V_{ref}}{2N}$$

Since errors can be both positive or negative, the negative sign in the first term is removed. With that change, the final, simplified error form is as follows

$$\varepsilon = \frac{ADC_{out}(R_1 + R_2) u_{vref} + ADC_{out}(R_2 U_{R_1} + R_1 U_{R_2}) V_{ref}}{N R_1^2} + \frac{V_{ref}}{2N} .$$

Resistors are sold labeled with a certain tolerance. Therefore, U_{R_1} can be expressed as $T\,R_1$, where T is the tolerance of the resistor with a nominal value of R_1. Adding this simplification in for both resistors,

$$\varepsilon = \frac{ADC_{out}[\,(R_1+R_2)\,U_{V_{ref}} + 2\,R_2\,T\,V_{ref}]}{N\,R_1^{\,2}} + \frac{V_{ref}}{2N}\,.$$

We will further assume that the Values of R_1 and R_2 have been chosen so that, on average, ADC_{out} will be approximately one-half of N. With that further simplification,

$$\varepsilon = \frac{(R_1+R_2)\,U_{V_{ref}} + 2\,R_2\,T\,V_{ref}}{2\,R_1^{\,2}} + \frac{V_{ref}}{2N}\,.$$

In the circumstance of measuring the upper battery voltage cell,

$$R_1 = 51k,\ R_2 = 10k,\ T = 0.5\%, V_{ref} = 2.56,\ U_{V_{ref}} = .2\ V,\ N = 1024\,,$$

Values which are derived from datasheets or design specifications. In this case,

$$\varepsilon = .123\ V\ ,$$

Which justifies the need to calibrate for the reference voltage.

To determine the level of accuracy to which the calibration process calculated V_{ref}, a second error propagation analysis is performed. To calibrate, an accurately known voltage is input to V_{in}, and the value for the reference voltage is calculated. Specifically,

$$V_{ref} = \frac{V_{in}\,N\,R_1}{(R_1+R_2)\,ADC_{out}}$$

Performing a similar error propagation analysis to the one above,

$$\varepsilon_{V_{vref}} = \frac{2\,N\,R_1\,R_2\,V_{in}}{ADC_{out}\,(R_1+R_2)^2}.$$

Again, assuming that ADC_{out} will be approximately one-half of N,

$$\varepsilon_{V_{vref}} = \frac{4\,N\,R_1\,R_2\,V_{in}}{(R_1+R_2)^2}.$$

In the circumstance of our calibration setup,

$$R_1 = 51k,\ R_2 = 10k,\ T = 0.5\%, V_{in} = 5\ V,\ N = 1024\,.$$

The calculation shows that,

$$\varepsilon_{V_{vref}} = 0.013V$$

Therefore, with this calibrated value of the reference voltage, the overall measurement error of the cell will be

$$\varepsilon = .011\ V\ .$$