**RobotMoose BMS operation**

**Overview:**

The BMS is designed to charge and manage any 3-cell lithium battery. However, the firmware it comes loaded with is specific to Li-Po chemistry voltages, and using it with other chemistries could cause over charging or a fire.

The BMS acts as a shield atop an Arduino Uno platform, and sends battery health and charge data to the main robot controller via I2C protocol.

**Connection:**

The BMS must be connected correctly before it will allow the robot to power up. The battery positive and negative are connected, as well as the balance leads plugged into the BMS board. The start/estop pin must be connected through a momentary push button to ground. Additionally, “e-stop2” and start/e-stop pins must be connected to each other through a toggle switch. Connected gives normal operation, disconnected or open switch cuts power to the robot.

The charger (12V dc supply) is connected at the CHGR terminal. There is also a battery OUT terminal, which along with the second large GND pin provides power to the rest of the robot.

Finally, SDA, SCL, and GND must be connected to the I2C port of the main control board in order to allow communication between the robot and the BMS.

**Operation:**

**The BMS is connected to the balance plug of the battery, as well as the** main power leads of the battery and the power leads of the battery charging supply. During operation, the BMS continually monitors the voltage of each of the 3 cells to ensure that they are within the specified voltage range. Any cell too high or too low will cause the BMS to cut power to the rest of the robot, as well as to itself, in order to keep from discharging the battery into an irreparable state.

To begin operation, the emergency stop toggle must be in the connected position, and the momentary start button must be depressed for about 5 seconds. This pulls the gate of the main power PFET low, providing power to the BMS and the rest of the robot. As the BMS wakes up and performs a battery voltage check, if everything is correct, it will provide power to the gate of an Nfet, which latches the main PFET and allows the momentary button to be released.

If the robot doesn’t stay powered upon releasing the start button, check that the stop switch is in the run position, the balance leads of the battery are connected, and that the battery is not over-charged or over-discharged.

**Communication:**

For more advanced users, communication with the BMS can be accomplished without RobotMoose. The I2C address of the BMS is 0x02 by default, though this can be easily change by modifying the value of the “ADDRESS” parameter in the BMS firmware. Upon receiving an I2C request from the master, the BMS transmits a three byte packet. The first byte is header with the value 0xCC; this is only used as an error detection and synching mechanism. The next byte contains the charge of the battery in percent, from 0 to 100. The last byte contains the charging and discharging flags, and these are interpreted as shown in **Table 1**:

Table 1: Bit Field Interpretation for Last Byte of BMS Packet

|  |  |  |
| --- | --- | --- |
| **Bit** | **Value** | **Description** |
| 0 (LSB) | 0 | Cell #1 is not discharging. |
| 1 | Cell #1 is discharging. |
| 1 | 0 | Cell #2 is not discharging. |
| 1 | Cell #2 is discharging. |
| 2 | 0 | Cell #3 is not discharging. |
| 1 | Cell #3 is discharging. |
| 3 | 0 | Battery is not charging. |
| 1 | Battery is charging. |
| 4 | 0 | System is not operating normally. Caused by either the battery being  disconnected or by any of the cell voltages being below 3.2V. |
| 1 | System is operating normally. |
| 5-7 | 0 | Unused |