Lafayette College | Electrical and Computer Engineering

Battery Pack Errata

ECE/ME 492 | Spring 2020

Last Revision: 4/15/2020

Prepared by: Jon Abel, Clement Hathaway, Simone Khalifa,
Tim LaGreca, and Connor Winarczyk

Full Pack

- Charging - There is a possibility that the MeanWell PB-1000-48 battery charger that was already purchased will not effectively charge the packs. On its datasheet, the charger is explicitly stated to support Li-Ion cell charging. However, it was shipped to the team with firmware for Lead-Acid battery charging. There was only one attempt to charge the pack made, and when connected and powered on the charger's LED illuminated green, meaning it thought the pack was fully charged, when it was not. MeanWell's support was not helpful at all in figuring this out, but try again and try to get it working. It's a great charger, it would be awesome to charge the packs at 17.4A.

PacMan

- First things first: build and test PacMan v1.3. PacMan v1.2 was as functional as possible at the end of the year. Changes that were fixed with fly-wire and repositioning components on v1.2 were fixed in v1.3. v1.3 also contains many adjustments to spacing and some changed/extra components.

- Testing for v1.2: not all of the sensors were calibrated. Each sensor was verified to work and transmit correct data, but none were calibrated properly. This could be worthwhile to complete before moving onto v1.3.

- Work out how to deal with regenerative braking. With the current design, If the voltage polarity switches, it will likely destroy the 48V-24V DC-DC converter on PacMan and also the 48V-24V DC-DC converter that's powering the TS Active LED indicator. It would also likely destroy the LTC4151 power monitor being used to measure pack voltage. Maybe diodes could be used to deal with that.

- Transient protection on any pack voltage inputs might be valuable to protect against inductive voltage spikes throughout the batteries. Like putting some tranzorbs between the PACK+/PACK- input. Idrk.

- The circuitry for charge detect can use fewer components and be made less complex. For example, voltage could be sent out of CHRG_DETECT+ and CHRG_DETECT- could connect to the gate of an N-channel MOSFET to drive the optoisolator's LED.

- Double check the shunt resistor R26's value. It was chosen with a maximum current draw from the cells of 100mA in mind, but that may not be correct. A PacMan was never connected to 16 functional CellMan all drawing power, so that maximum current draw may need to increase. Use the datasheet for the LTC4151 to help with this, it expects a maximum differential voltage across the shunt of 81.92mV.

- PLEASE consider switching from the e-Paper display to an OLED display module. Not only would the display be better, but it would probably shrink the already too big panel connector J11.
    - Sidenote from Clement (Firmware): I also agree with this, the slow response time of the e-ink display caused us a lot of issues in the firmware, especially in regards to input. This has required us to use more threads to handle input to deal with this slow update. An OLED would probably help simplify and solidify the code base in regards to input. The e-ink driver code is also incredibly large and verbose.

- It could be interesting trying to control which source powers PacMan, or how much of each source. If the GLV battery capacity is the car's limiting factor, maybe controlling/limiting PacMan's current draw from the GLV battery would be a necessary step to increase the car's running time.
- If the ESP32 crashes while the car is active, the AMS safety loop will likely open since it is controlled directly by an ESP32 GPIO pin. Even if the ESP32 resets and recovers (because of a watchdog or something), the car would still be finished since an AMS safety loop fault requires someone other than the driver to reset the safety loop. The rules allow 1 minute before the safety loop has to be opened after some sort of fault. Maybe the safety loop opening could be delayed so that a momentary glitch in the processor would not end an entire event run.
- The RTC's crystal and loading capacitors might need some tuning. I did not have enough time to calibrate the RTC and check that it tracks well over time.
- I wouldn't recommend this for next year, but eventually it might be a good idea to switch from the ESP32 DevKitC to an STM32 microcontroller. CellMan already uses an STM8, and this would help unify the AMS architecture. The processor could probably run faster without the overhead from Arduino and FreeRTOS (which is only partially being utilized anyway) and it would certainly take up less space, allowing the PacMan board to shrink.
    - Sidenote from Clement (Firmware): The same shrinking could be done by not utilising the Dev board. The actual ESP32 chip is very tiny. I think we use a good portion of FreeRTOS, but that is also available for the STM32F3 and there is a premade driver for CANopenNode but you would have to get a CAN controller (such as an MCP2515, depending on STM chosen). The ESP32 has this built in.

PacMan Firmware

- Continue to use the GitHub and use the comment section of your Git comments to detail everything you changed so it is easy to keep track of where things went. I tried to also include helpful debug information in them. Consider using branches if you want to make large changes to the codebase as well so that the Master hopefully remains stable.
- Upgrade to RaspberryPi 4's if you're going to use them for development and debugging, the firmware takes an incredibly long time to cross compile on the RaspberryPi compared to a modern x86 machine.
- I would recommend setting up a RaspberryPi with the CANopenStack for testing in addition to SCADA.
- Segment detection. This first has to be fixed on the hardware first, but then the software for PacMan as well. Currently we cannot differentiate between segments since we can only collect information about a CellMan's potential relative to its ground. Since we have two separate isolated grounds on each segment we have 2 CellMan reporting the roughly same relative potentials to ground. To fix this, it should be possible to reset segments individually so that we can scan and detect CellMen in a segment and then switch the reset to the other segment and repeat. We should then be able to assign a segment ID to the CellMen detected and run the current sorting algorithms with this in mind.
- The bug should be investigated where sometimes the data returned from CANopenNode is not complete. We believe it has to do with the Object Dictionary being accessed in that moment but are not sure. It will report "Remote access" in this case.
- Charging should be abstracted up slightly so there is a "driver" in the middle between the hardware and the algorithm implementation that could in future be changed out without affecting lower level code.
- I would implement a much better State of charge algorithm, this is detailed slightly in the code but I would at least implement the simple coulomb counting method using the current sensors. An improved method would be to use the EKF to estimate SOC using a model of our batteries. This is what is used in industry and you can find many research papers exploring how to implement this.
- A simple WiFi or Bluetooth interface (built-in to the ESP32 already) to modify configuration parameters would be incredibly useful. Nadovich has raised security concerns about this, but I'm doubtful that the people at the competition would be out to

mess with you or would have the knowledge/time to do so. It would be very convenient to be able to update things quickly on your phone/tablet/laptop.

- I would avoid any form of multi-master implementations for I2C or SPI. It becomes messing and unnecessarily complicated quickly. Keep the communication buses as simple as possible.
- I also wouldn't necessarily recommend this for next year, but seriously explore Micropython or Rust (work is being done to have Xtensia support in the LLVM). The amount of trouble we ran into with memory errors due to improperly accessed RAM caused us to lose a lot of time debugging. A ESP32 debugging tool would also probably be a great addition to help aid in debugging.
- Interrupts have been implemented to speed up button presses, but continue to investigate how to have them react even more quickly.

CellMan

CellMan Firmware