Shilpa Rao
Work @ Arc Boats

Arc Boats

Automated
Battery Tests

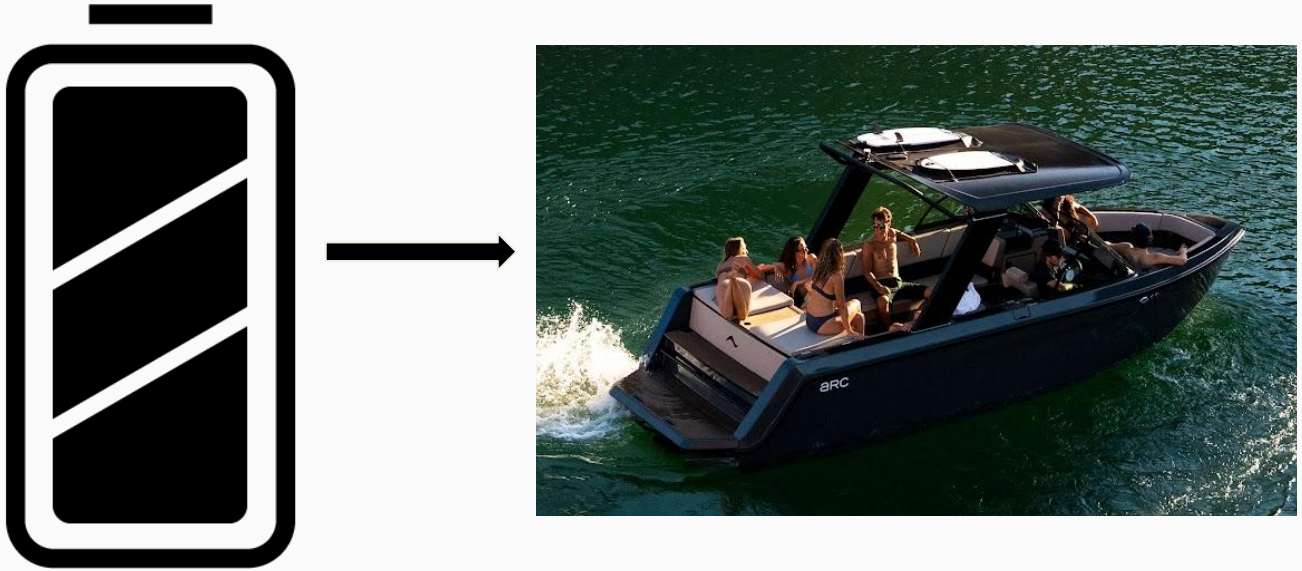
800v
226 kWh

Arc Boats: Scope

Verify no electrical or mechanical errors before battery is sealed and ready for the water.

Context: end of Q3 production sprint, first customer boat

Sr EE responsibilities



Arc Boats:
Automated Battery Tests

Deliverables

01

Hipot:

Check for leakage or shorts between insulated components using high transient current

02

Precharge:

Precharge contactor closes and BMS returns pass/fail condition

03

Pack charge:

Charge pack until 1% SOC increase is achieved

04

BMS check:

Verify accurate voltages / leak boolean / temperatures etc over CAN

Arc Boats:
Automated Battery Tests

Results

01 50% automated

Hipot:

Check for leakage or shorts between insulated components using high transient current

02 Automated

Precharge:

Precharge contactor closes and BMS returns pass/fail condition

03 Automated

Pack charge:

Charge pack until 1% SOC increase is achieved

04 Automated

BMS check:

Verify accurate voltages / leak boolean / temperatures etc over CAN

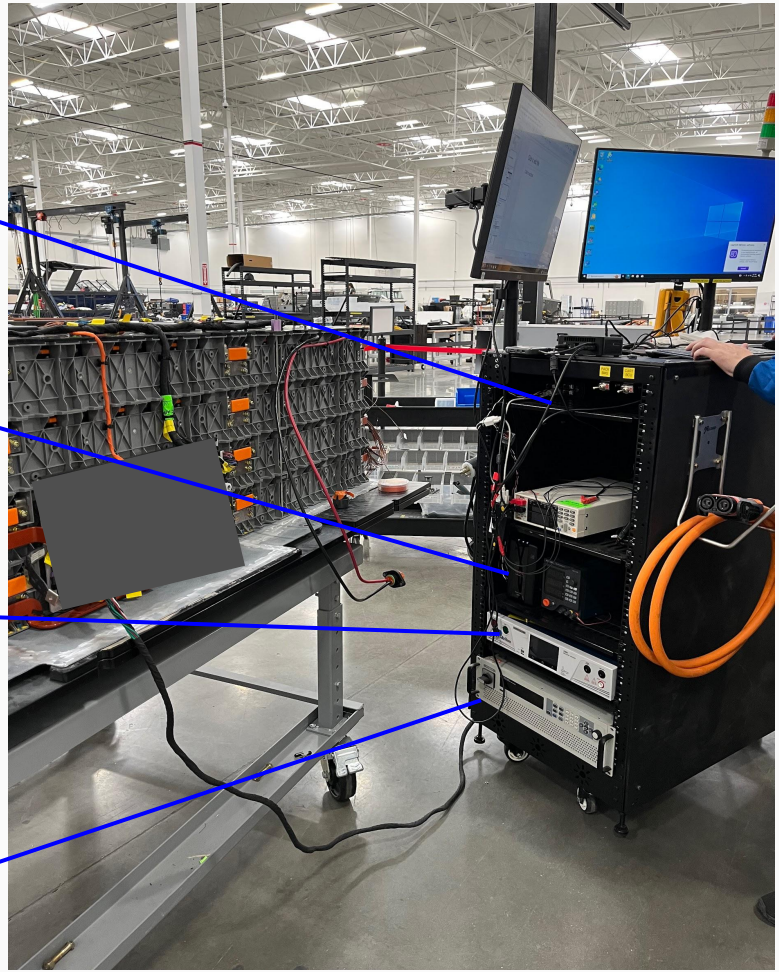
Test Rack

CAN Communications Rack
Linux PC
Control VCU

APC + Rockseed power supplies

HypotULTRA 7850

ITech IT6000C



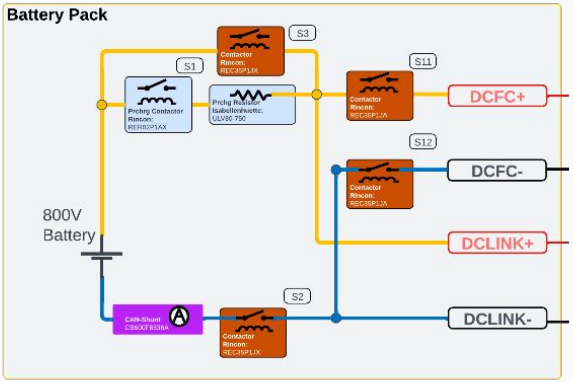
Test Rack:
simulating a boat

- Tricked BMS into thinking it's in a boat
  - Allowed precharge, pack charge
- Used extra Ecotron VCU to command BMS states not otherwise allowed
- Used VCU GPIO to control HypotUltra 7850
- Customized proprietary Python and OpenHTF framework
  - Primary product from controls team = linux .exe that runs entire boat operations from VCU
- Delivered Linux executable using VCU for custom battery tests
- Laid groundwork for better testing collaboration between teams (control, mech, EE)
  - VCU kept in loop means more upkeep by controls team
  - VCU firmware has to match BMS revisions
  - Supports modularity - test BMS as it evolves

Hipot Test

Why hipot?

- Dielectric withstand test ensures no current will flow between two disconnected points
- ISO 6469 standard
- Toshiba resource on EV withstand test
  - Test Voltage = 1.41 * (2U + 1000 Vrms) where U = 800v (operating voltage)
  - Test Voltage = 3,666 volts DC for 1 minute
- Captures transient spikes / switching events at 2-3x nominal voltage

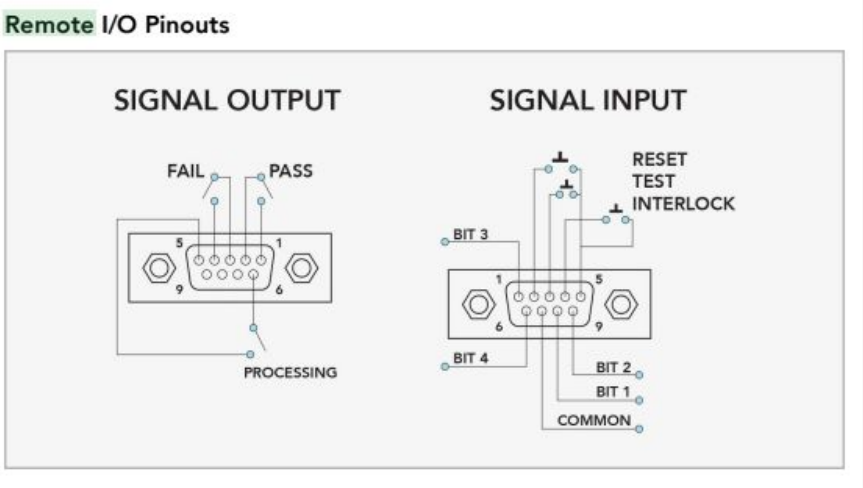


TEST CASE	RETURNING INFO
DCLink + to Case GND [Ω]	Pass/Fail/Processing
DCLink - to Case GND [Ω]	Pass/Fail/Processing
DCLink + to DCLink - [Ω]	Pass/Fail/Processing
DCFC + to Case GND [Ω]	Pass/Fail/Processing
DCFC - to Case GND [Ω]	Pass/Fail/Processing
DCFC + to DCFC - [Ω]	Pass/Fail/Processing

Hipot Test

1. **Interface with battery = manual**
2. **Voltage output trigger = automated (human-in-loop)**
3. **Recording results= automated**

- Initial vision: contactors auto switching between battery ports
- Reason for halfway automation
  - **Risk of high voltage short if contactor failure; no time to test**
  - Result: technician places HV alligator clips on battery
- Linux PC runs pre-set test. Records results from DB9 connector on Hipot
  - Uses vehicle control unit (VCU) high side outputs (HSO) to command switches
  - VCU 12v digital inputs



Hipot Test

Final product

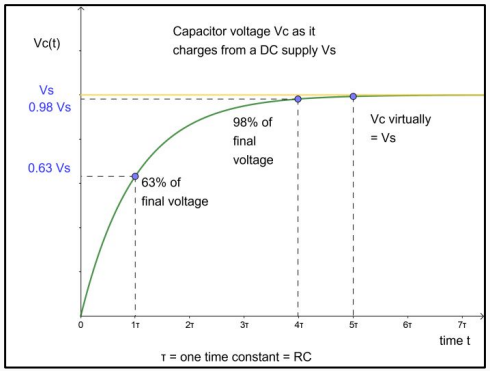
Assumes you have an operational Linux PC and access to the Arc Boats automated-atp git repository.

1. Turn Hipot meter on (bottom left button). Press "OK".
2. On Linux PC, run main.py by running:
    a. `shilpa@shilpa-ThinkPad-T14-Gen-3:~/Desktop/automated-atp/test-runner$ poetry run python3 ./test_runner/battery_checkout/main.py`
3. Check work instructions. Move alligator clips, then proceed through each phase of test accordingly.


```
shilpa@shilpa-ThinkPad-T14-Gen-3:~/Desktop/automated-atp/test-runner$ poetry run python3 ./test_runner/battery_checkout/main.py
This code will run the Hipot test until you enter N. Press enter to continue.
```

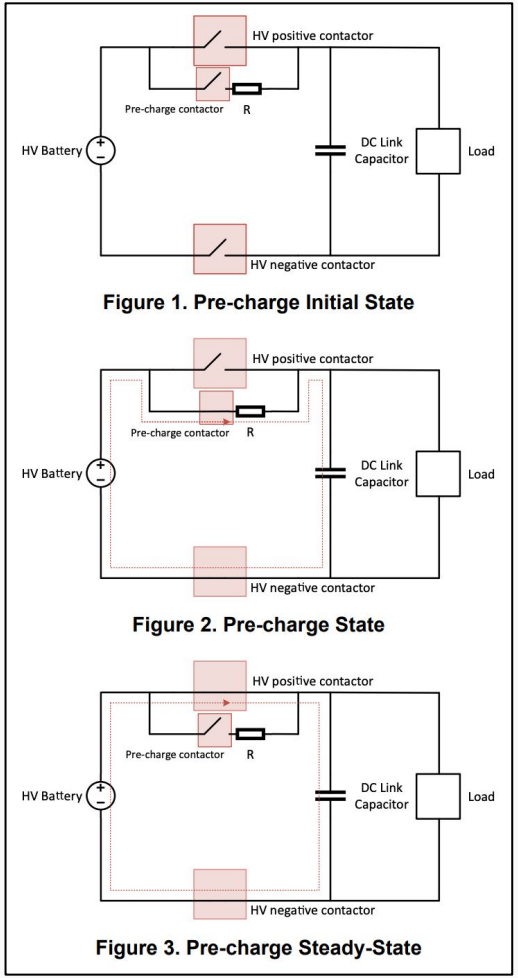
Precharge Test

- Battery charged via AC→DC rectifier. Load = inverter/motor.
- Precharging controls the current increase across DCLink cap.
  - DCLink cap: "for filtering the switching ripple on the DC bus"
- Requirement by BMS. Charging not allowed otherwise.
- Code already completed by controls team.
- My scope: command precharge test over CAN
  - Repurposed complex controls code
  - Hacked VCU to fake "VCU heartbeat" and allow pre-charge to continue



$$v_C = E\,(1 - e^{-t/RC})$$

Precharge RC circuit controls charge buildup on capacitor



Source: Texas Instruments

BMS check

- Implemented prior by controls team
- I used hacked VCU to access those signals
  - Recall: BMS expected some preprogrammed VCU logic to enable precharge and deliver signals
- My API checked each signal against expected values

- Output from Linux .exe →

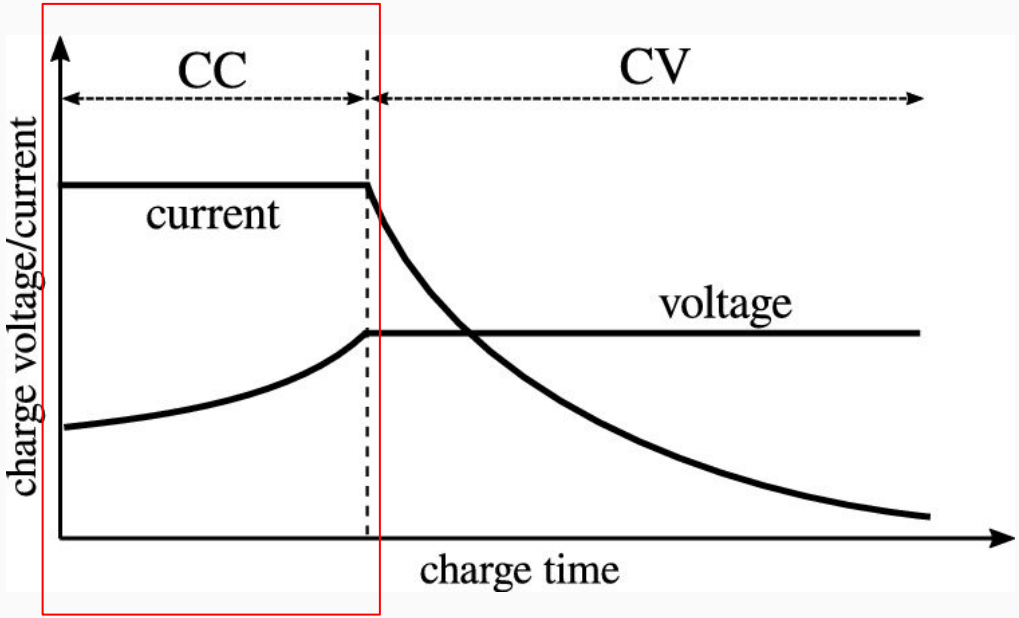
```
bms_prechrg_timeout:  INACTIVE
bms_dclink_neg_stuck_open:  INACTIVE
bms_dclink_neg_stuck_clsd:  INACTIVE
bms_dclink_pos_stuck_open:  INACTIVE
bms_dclink_pos_stuck_clsd:  INACTIVE
bms_dcfc_neg_stuck_open:  INACTIVE
bms_dcfc_neg_stuck_clsd:  INACTIVE
bms_dcfc_pos_stuck_open:  INACTIVE
bms_dcfc_pos_stuck_clsd:  INACTIVE
bms_bmu_mia:  INACTIVE
bms_host_mia:  ACTIVE
bms_curr_sensor_mia:  INACTIVE
bms_heartbeat_lost:  INACTIVE
bms_hvil_fault:  INACTIVE
bms_iso_res_warn:  INACTIVE
bms_iso_res_sev:  INACTIVE
bms_thermistor_fault:  INACTIVE
bms_cell_volt_fault:  INACTIVE
bms_hv1_volt_fault:  INACTIVE
bms_hv2_volt_fault:  INACTIVE
bms_hv3_volt_fault:  INACTIVE
bms_hv5_volt_fault:  INACTIVE
bms_pack_volt_fault:  ACTIVE
bms_dclink_pos_drv_fault:  INACTIVE
bms_dclink_neg_drv_fault:  INACTIVE
bms_dcfc_pos_drv_fault:  INACTIVE
bms_dcfc_neg_drv_fault:  INACTIVE
bms_balance_ckt_overtemp:  INACTIVE
bms_balance_ckt_open:  INACTIVE
bms_balance_ckt_short:  INACTIVE
bms_eeprom_fault:  INACTIVE
bms_lv_supply_fault:  INACTIVE
:PASS


========================= test: main  outcome: PASS =========================

Environment config: EnvConfig(USE_VIRTUAL_CANS=False, USE_UI=False)
Enter the ID of the boat you're testing to start the test.
--> l
bms_leak_aft_stbd_adc:  1473
bms_leak_aft_stbd_adc:  1471
bms_leak_aft_port_adc:  1474
bms_relative_humidity_adc:  1753
bms_ambient_temp_adc:  1789
:PASS
```

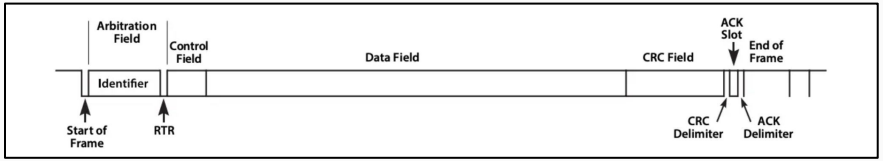
Pack Charge Test

- Trial and error:
  - Pack voltage determined from BMS readings
  - Charge voltage set to 1.2 x pack voltage
  - Pack charged at State of Charge (SOC) 0.5-1% increase per 20 minutes (slower = safer)
- Automated iTech IT6000C Power Supply over CAN - previously tried and failed by sr engineers
- Set power supply to CC (constant current) mode due to low SOC.



Pack Charge Test

- Two problems with CAN DBC
  - (1) Power supply address encoded in all signals.
    - Incorrect address assumed in DBC, meaning no information sent.
  - (2) Vendor-provided DBC missing SDO_CCS signal inside SDO_Server.
    - Required to enable remote control of machine
    - CAN messages from power supply contained 8 signals (bytes) carrying info, but Vendor DBC only mentioned 7 signals
- 2 other engineers had tried and failed to remotely control these power supplies.
- Replaced $10k/license vendor software



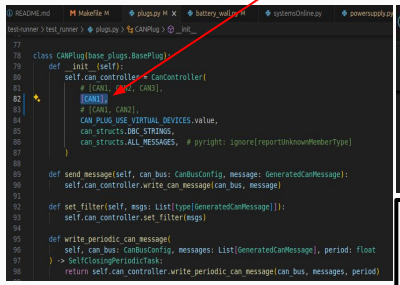
Source: Kvaser

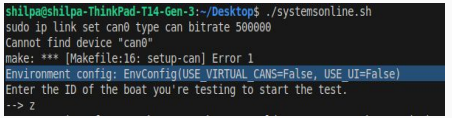
```python
@htf.plug(can=CANPlug)
def ps_setvoltage(test: htf.TestApi, can: CANPlug, voltage=2000):  # mV
    print("setting voltage")
    SDOServer = can_structs.SDOServer(
        # Range: 0 to 1, Scale: 1
        SDO_Size_Indicator=1,
        # Range: 0 to 1, Scale: 1
        SDO_Expedited=1,
        # Range: 0 to 3, Scale: 1
        SDO_Number=0,
        # Range: 0 to 1, Scale: 1
        SDO_Toggle=0,
        # SDO_CCS = 1,
        SDO_CCS=SDO_CCS_Enum.initiateDownload,
        # Range: 0 to 65535, Scale: 1
        SDO_Index=2,
        # Range: 0 to 255, Scale: 1
        SDO_SubIndex=2,
        # Range: 0 to 4294967295, Scale: 1
        SDO_Data=voltage,
    )
    can.send_message(CAN2, SDOServer)
```

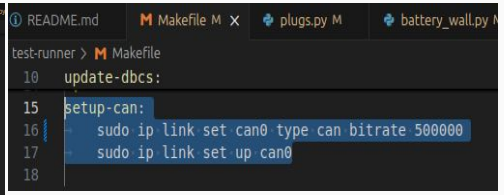
Testing final solution

- "Checked out" test battery ~4 times in last 2 weeks of internship
  - Identified common operation errors
  - Mostly Linux / CAN based
- Put together work instructions with troubleshooting guidance
- This slide shows 2 excerpts from work instructions with my learnings

If you get this error, make sure that the function **CANPLug** (in plugs.py) only declares **one** CAN line, labeled [CAN1].







Make sure that the command setup-can in the Makefile (/home/shilpa/Desktop/automated-atp/test-runner/Makefile) sets the baudrate to 500000. This differs from the boat - the boat requires the first CAN line to be 250000 as it talks to the VCU.

If you get this error, check to make sure that you've plugged in a PeakCAN device to a port on the laptop. This code will not work with Kvaser or Vector (etc).

Roadblocks: technical / behavioral

Senior EE responsibility as an intern →

- Mentor left midway through internship (7 EEs → 6 EEs)
- I was the only engineer left w/ up-to-date battery test knowledge
- First customer boat **production sprint**
- Handoffs to Manufacturing lead + lead EE

Incorrect vendor data →

- Learned to be rigorous with vendor information
- No question is dumb
- First principles are important

HVIL signal drop during production handoff →

- Collaborated with mechanical team to diagnose issue and unblock boat production
- Mechanical, production or electrical issue?
  - All 3: over-torqued PCB stud