

InstrumentedFactory



Battery

rated_amp_hours # given
batt_r_ohms # given
initial_soc_per # given
battery_profile # given csv file (ocv vrs soc)
start_time

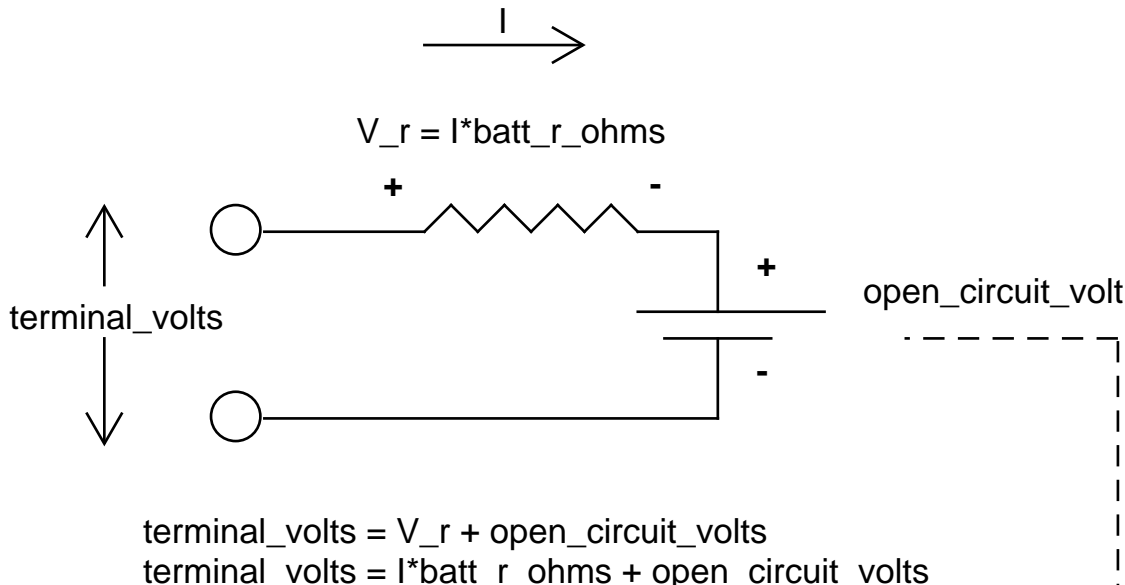
open_circuit_volts
soc_per
amp_hours
last_current_amps
last_terminal_volts
last_sample_time

_amp_hours_given_amps(amps, time)
_amps_given_terminal_volts(terminal_volts)
_time_to_seconds(time)
fn_soc_to_ocv(soc)

build_ocv_soc_profile

```
def _amps_given_terminal_volts(terminal_volts):  
    voc = self.fn_soc_to_ocv(self.soc_per)  
    v_r_volts = terminal_volts - voc  
    amps = v_r_volts / self.batter_r_ohms  
    return amps
```

```
def _amp_hours_given_amps(amps, time):  
    delta_t_sec = time - self.last_sample_time  
    amp_hours = amps * delta_t_sec / 3600.0  
    return amp_hours
```

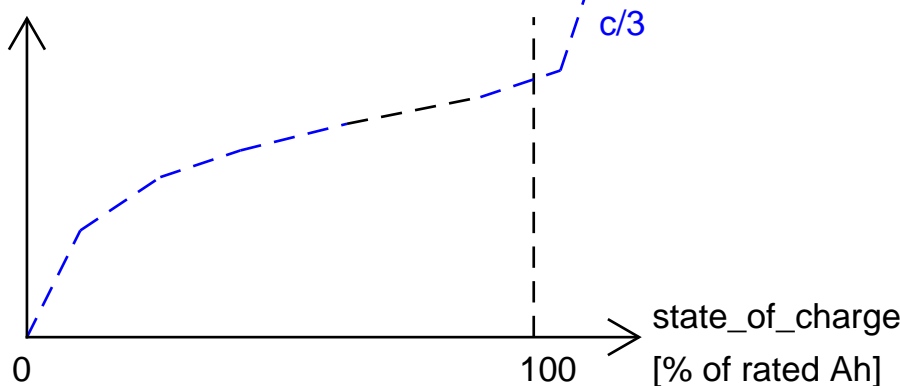


build_ocv_soc_profile

```
entry /  
    self.start_time = time.now()  
    self.rate_amp_hours = (given_value)  
    data_ocv_soc = np.getfromtxt(  
        self.battery_profile,  
        delimiter=',',  
        skip_header=1,  
        names=['state_of_charge', 'open_circuit_voltage'],  
        dtype='float, float')  
    self.fn_soc_to_ocv = \  
        interp1d(  
            data_ocv_soc['state_of_charge'],  
            data_ocv_soc['open_circuit_voltage']  
        )
```

battery_profile (csv file):

open_circuit_voltage [V]



volts_to_amps

```
volts as e /  
    self.post_lifo(  
        Event(signal=signals.Volts,  
            payload=Volts(volts=e.payload.volts, time=time.now()))
```

Volts = namedtuple(
 'Volts', ['volts', 'time'])

```
Volts as e /  
    amps = self._amps_given_terminal_volts(  
        terminal_volts=e.payload.volts,  
        time=e.payload.time)  
    self.last_terminal_voltage = e.payload.volts  
    self.post_lifo(  
        Event(  
            signal=signal.Amps(payload=amps,  
                time=time))
```

amps_to_amp_hours

```
amps as e /  
    self.post_lifo(  
        Event(signal=signals.Amps,  
            payload=Amps(amps=e.payload.amps, time=time.now()))
```

Amps = namedtuple(
 'Amps', ['amps', 'time'])

AmpHours = namedtuple(
 'AmpsHours', ['amp_hours'])

```
Amps as e /  
    amps = e.payload.amps  
    last_time = e.payload.amps  
    terminal_voltage = amps*self.batt_r_ohms + self.open_circuit_volts  
    amp_hours = self._amp_hours_given_amps(amps=amps, time=last_time)  
    self.last_current_amps = amps  
    self.last_sample_time = e.payload.time  
    self.last_terminal_voltage = terminal_voltage  
    self.post_lifo(  
        Event(signal=signal.Amp_Hours,  
            payload=AmpHours(amp_hours=amp_hours))
```

update_charge_state

```
amp_hours as e /  
    self.amp_hours = self.soc_per/100.0 * self.rated_amp_hours + e.payload.amp_hours  
    self.soc_per = self.amp_hours / self.rated_amp_hours * 100.0  
    self.open_circuit_volts = self.fn_soc_vrs_voc(soc=self.soc_per)
```