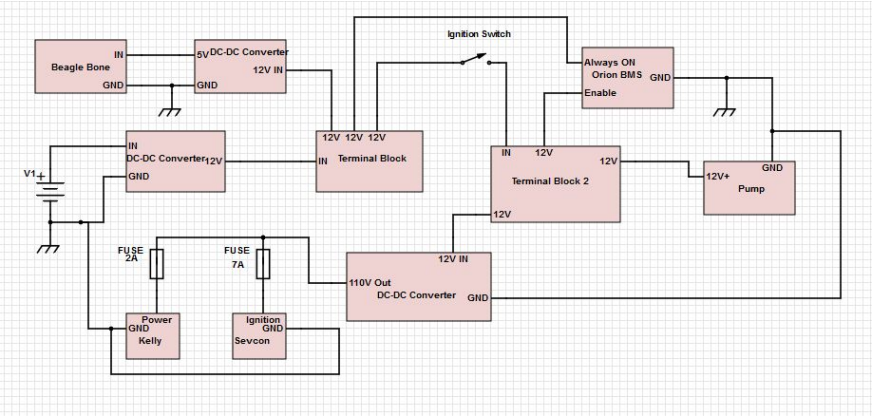
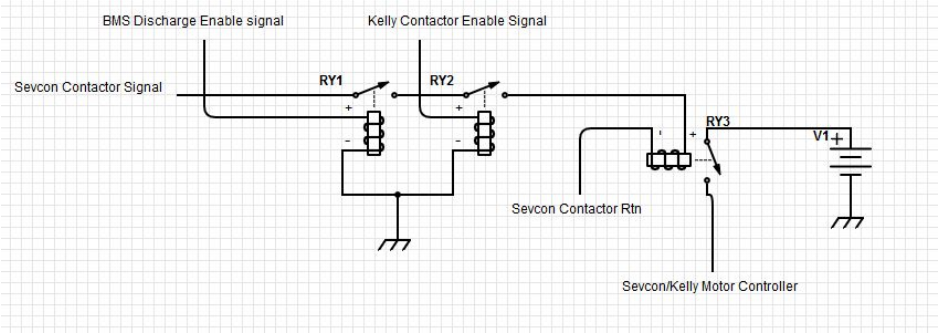
Design analysis - Electronic Systems

- 2 Major systems embedded
  - Requires key switch to be on
  - Always running
- BMS and BeagleBone Black required an uninterrupted 12V supply because they have features that continually monitor the system state



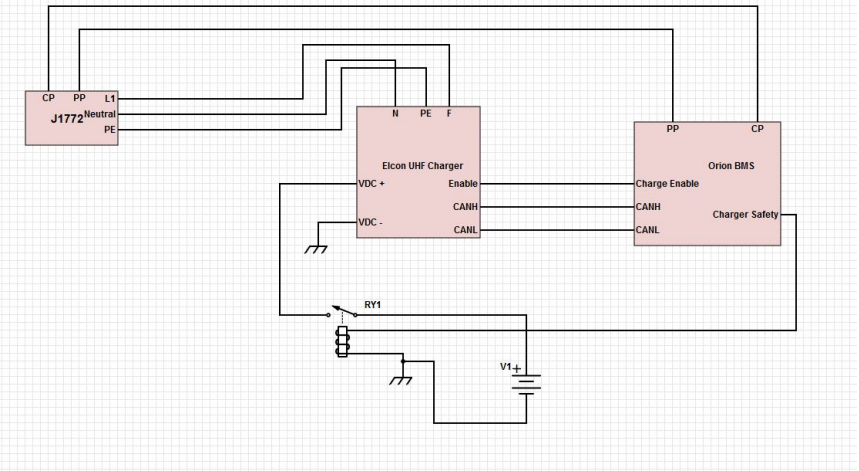
Design analysis - Electronic Systems

- Multiple devices want to control the when it is safe to discharge power.
- Relays in series was a safety design that required all systems to indicate it was okay for the contactor to be enabled. (Allow for discharge)



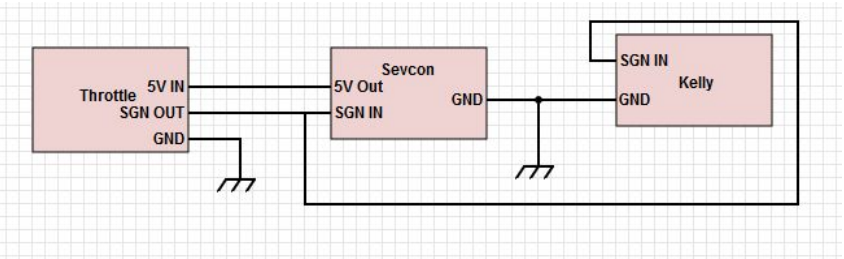
Design analysis - Electronic Systems

- BMS drives charging functionality.
- Elcon Charger is controlled by BMS over CAN bus.
- 2 Layers of safety
  - Charge Enable line
  - Charge Safety line
- Most common US charging standard used: SAE J1772



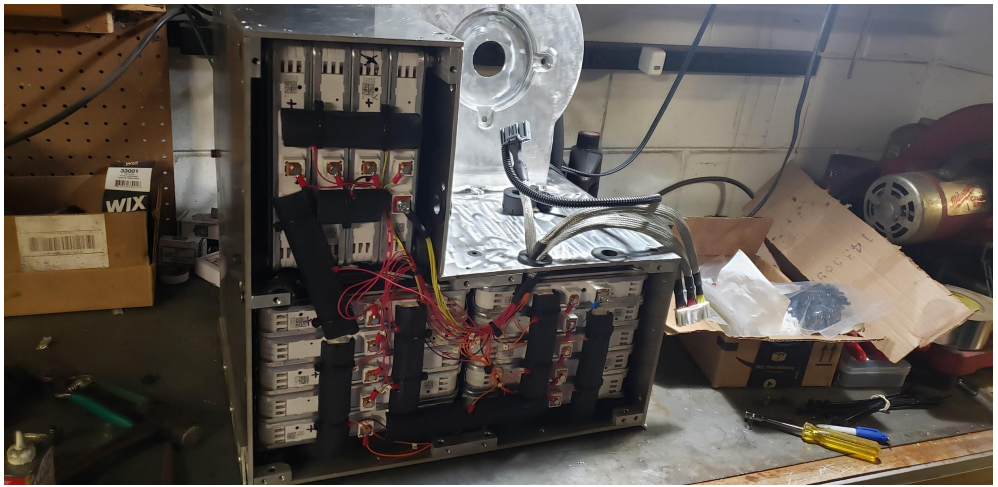
Design analysis - Electronic Systems

- Both chargers required 0-5V throttle input.
- Sevcon provided more control for creating a throttle map. Kelly was extremely limited
- Another microcontroller should have been dedicated for a closed feedback loop to maintain same wheel speed. (Traction control)



Design analysis - Electronic Systems

- BMS uses taps on each cell in series to monitor cell voltage.
- BMS uses thermistors to monitor cell temperature.
- Shielding was used on the wires to minimize potential interference from High Voltage lines running close by.



Design analysis - Electronic Systems

- BeagleBone Black running Debian 9.5
- Services added in the system to execute scripts on boot
- Python scripts execute bash commands on console to poll hexadecimal values from I2C or CAN ports

```
def bashExec(self, cmd):
    cmdList = cmd.split()
    res = subprocess.check_output(cmdList).decode('utf-8').strip('\n')
    return res

def setI2C(self, reg1, reg2, val):
    cmd = self.bashSet + " " + reg1 + " " + reg2 + " " + val
    execute = self.bashExec(cmd)

def getI2C(self, reg1, reg2):
    cmd = self.bashGet + " " + reg1 + " " + reg2
    res = self.bashExec(cmd)
    return res
```

Design analysis - Electronic Systems

- Frontend GUI written in HTML/css/Javascript
- Cefpython module used to create Javascript bindings, which call back into the data-collecting Python scripts
- Response time was very slow; a faster microprocessor should have been used, and the frontend should be written in C++

```
<!DOCTYPE html>
<html>
<head>
  <link href="./style.css" type="text/css" rel="stylesheet">
</head>
<body>
  <div class="header">
    <img src="./tachometer.png" class="tachometer">
    <battery-stats left="4.25" fill="0"></battery-stats>
    <h1 class="speed" id="speed">24</h1>
    <rpm-dial progress="0" class="rpmdial" id="rpmdial"></rpm-dial>
  </div>
  <script src="./gui.js"></script>
</body>
</html>
```

```
  setProgress(percent) {
    if(percent < 0 || percent > 65){
      return;
    }
    const offset = this._circumference - (percent / 100 * this._circumference);
    const circle = this._root.querySelector('circle');
    circle.style.strokeDashoffset = offset;
  }

  static get observedAttributes() {
    return ['progress'];
  }

  attributeChangedCallback(name, oldValue, newValue) {
    if (name === 'progress') {
      this.setProgress(newValue);
    }
  }
}
```

Design analysis - Electronic Systems

- Bench test of Front hub motor
- Used simplified system
  - Contactor
  - Throttle
  - Battery Pack
- No modifications from the previous wiring schematics were needed to run the motor successfully

Design analysis - Electronic Systems

- Bench test of Mid-Drive motor
- Used same simplified system as front hub motor
  - Contactor
  - Throttle
  - Battery Pack
- No modifications from the previous wiring schematics were needed to run the motor successfully