# Start-up Electrical Requirements

## Key should be connected b/w 36V battery and DC-DC Converter.

## As soon as the DC/DC converter powers up, it should power up the Arduino (VCU).

# Start-up Software Requirements

## Measure current from the current sensor to make sure it’s below a certain threshold.

### If at any point the current overshoots the limit, switch off the solid-state relay.

## Switch on the solid-state relay once safety check is done and enter accessory mode.

### Switch on the red LED to indicate the VCU is online.

## Check the current mode which includes forward, reverse, pedal-assist or cruise control. Make sure vehicle is in forward mode and all the others are off before waking up the motor controller.

## Check the throttle value. Make sure it is at 0.

## Detect a change from the push button. Once the change is detected, close the relay to wake the motor controller.

## When the Arduino starts, initialize all the outputs to default states and this has to be done before switching on the motor controller.

# Mode Selection

## By default, Forward mode is on.

## In case of digital high to any of the mode other than throttle, switch to that mode as long as that is the only digital high.

## In case of failure in the mode enabling go back to the forward mode.

## To switch to any mode other than forward vehicle must remain in forward mode prior to that.

## In case of multiple triggers, sustain the previous mode.

## Digital pins will be considered high when they are grounded.

## MCU wake feedback will be provided to input to determine push mode.

# Forward Electrical Requirements

## The throttle output should be connected to the analog input of the Arduino.

## Arduino will generate a frequency signal to control the throttle and motor controller.

## Detect the speed limit based on digital input.

### Two different digital pins should be assigned for high and low pins.

## Detect the push button change to turn on and off the pedal assist mode.

### Assign a digital input.

## Assign the analog input for the pedal assist sensor on Arduino.

## Take feedback from Hall sensors in the digital inputs.

# Forward Software Requirements

## Forward Mode is on by default and switched off if you are in cruise control or reverse.

## Convert the analog ADC input of throttle to percentage value.

## Calculate the speed of the vehicle based on the hall sensors or at least detect vehicle is moving or still.

## Map the percentage value according to Three Speed Mode.

### Based on speed switch, determine mode.

### Map the throttle linearly.

### Provide a digital low to the low speed in the motor controller if low or ground is detected on the low-speed pin of Arduino.

### Provide a digital low to the high speed in the motor controller if low or ground is detected on the high-speed pin of Arduino.

## When forward is enabled, give feedback as 1.

## 0 – 100% throttle will be converted to 0 – 255 PWM duty cycle.

# Pedal Assist Mode

## Detect a digital low or ground from the push button.

### The digital ground should not persist for a longer period and a consecutive digital high should be detected within a threshold period of time.

## Activate the pedal assist mode.

## Detect the pedal assist motion on an analog input from the PAS.

### Detect the driver effort based on PAS input.

## Detect the level based on the low-speed and high-speed pins.

## Map the output throttle based on level detected and driver effort.

### In level 1 assist the driver with 25% of driver effort.

### In level 2 assist the driver with 50% of driver effort.

### In level 3 assist the driver with 75% of driver effort.

## Based on activated mode, pass the pedal assist throttle value.

## For switching off pedal assist, the same routine should be followed as switching in the pedal assist mode.

## When pedal assist is enabled, give feedback as 2.

# Cruise Control Mode

## Detect a ground on the digital pin connected to the press button.

## Store the throttle percentage value at the instance of change in the state of the digital pin.

## Make sure throttle percentage is less than a safe value for cruise control.

## Generate some diagnostic from red LED indicating cruise control is on.

### 

## Keep the throttle value constant at this value.

## If at the time of press of the “press button” the throttle value was 0, then set the cruise control to a pre-calibrated value.

## If a vehicle speed is above a certain threshold, don’t activate CC.

## When press button is unpressed, deactivate cruise control.

## When cruise control is enabled, give feedback as 3.

1. **Reverse Mode**
   1. **Electrical Requirements**
      1. Connect the push button to a digital input.
      2. Create a digital output on the Arduino. This will be connected to the reverse input of the motor controller.
   2. **Software Requirements**
      1. Detect a long press of the push button.
      2. Detect a constant press of the push button.
      3. Detect whether vehicle is moving or still.
      4. If the current vehicle speed is less than a very small threshold value and initial two conditions are true then go to reverse.
      5. As soon as the push button is no longer pressed go back to the forward mode.
      6. Turn on the red LED light when entering the reverse mode.
      7. When reverse is enabled, give feedback as 4.
      8. Throttle should be mapped and the maximum value should be very small.
      9. Enable Reverse Throttle when Reverse is enabled.
2. **Battery (Phase 2)**
   1. **Electrical Requirements**
      1. Add one voltage sensor to the battery.
      2. Add one current sensor to the battery.
   2. **Software Requirements**
      1. Based on the current calculate the approximate state of charge using the Coulomb counting method.
      2. Use the voltage as feedback to improve the state of charge estimate.
3. **Diagnostic (Phase 2)**
   1. **Electrical Requirements**
      1. Add a relay channel to control the diagnostic LED.
      2. Connect the diagnostic LED to the relay.
   2. **Software Requirements**
      1. Detect an issue in the start-up.
      2. Detect an issue in the normal drive mode.
      3. Detect an issue in the pedal assist mode.
      4. Detect an issue in the cruise control mode.
      5. Detect an issue in the reverse mode.