Draw a complete **Wiring Harness Diagram** for an e-bike, illustrating the interconnections between key components: the battery, motor, controller, throttle, display, and brake sensors.

1. **Introduction:**

   Objective: To connect and integrate all components of the BLDC electric vehicle system — including the motor, controller, battery, throttle, brakes, display, and sensors — through proper wiring harness connections to ensure safe and efficient operation of the electric drive system.

2. **Assembly:**

Step 1 – Motor → Controller

- Connect the 3 thick phase wires (Yellow / Green / Blue).
- Plug in the 5-pin Hall-sensor plug (Red = +5 V, Black = GND, Yellow/Green/Blue = signals).

Step 2 – Battery Pack → Controller

- Battery (+) to Controller (+).
- Battery (–) to Controller (–).
- Confirm correct voltage (e.g., 36 V or 48 V).

Step 3 – Throttle → Controller

- Red → +5 V
- Black → Ground
- Green (or Blue) → Signal

Step 4 – Left & Right Brakes → Controller

- Each brake lever has 2 wires.
- Plug both into the controller's brake inputs (they cut power when braking).

Step 5 – LCD Display → Controller

- Connect the 5- or 6-pin display cable to the port marked "Display."

Step 6 – Horn / Light Switch → Controller

- Connect the switch to the controller's horn/light output.
- From the switch, run wires to the horn and headlight.

Step 7 – Pedal Assist Sensor (PAS) → Controller

- Mount PAS near the crank.
- Red → +5 V
- Black → GND
- White → Signal

Step 8 – CAN Interface → USB → Laptop

- Plug CAN connector from controller into Type-C cable.
- Connect to laptop for tuning or diagnostics.

Step 9 – Battery Charger → Battery Pack

- Plug charger into the battery's charge port (never directly to controller).

Step 10 – Final Check

- Re-check polarity and connector seating.
- Make sure no wires are loose or exposed.
- Then connect the main battery and power ON.

Components Involved:

BLDC Hub Motor, Motor Controller, Battery Pack, Throttle, LCD Display, Left Brake Lever, Right Brake Lever, Horn, Headlight, Horn and Light Switch, Pedal Assist Sensor (PAS), CAN Interface, USB Type-C Cable, Charger, Laptop**.**

Conclusion

The wiring harness connections for the BLDC electric vehicle system were successfully completed. All components, including the motor, controller, battery, throttle, brakes, display, and sensors, were properly integrated to ensure smooth and safe operation. The system is now ready for testing and performance verification**.**

Model and simulate active and passive cell balancing of a lithium-ion battery using a resonant switched capacitor topology in MATLAB/Simulink, and compare their performance.

Introduction:

- **Objective**: To model and simulate active and passive cell balancing of a lithium-ion battery using a resonant switched capacitor topology in MATLAB/Simulink, and to compare their efficiency and balancing performance.
- **Tools Used**: Simulink is used to model to create active & passive cell balancing.

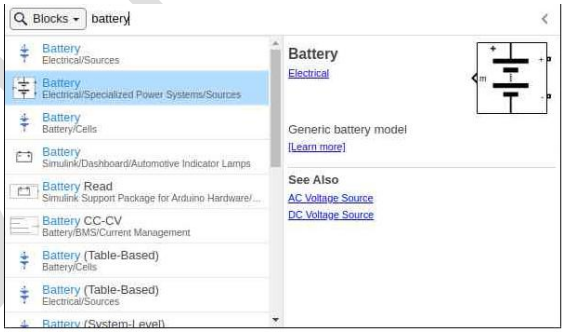
1. Model setup:

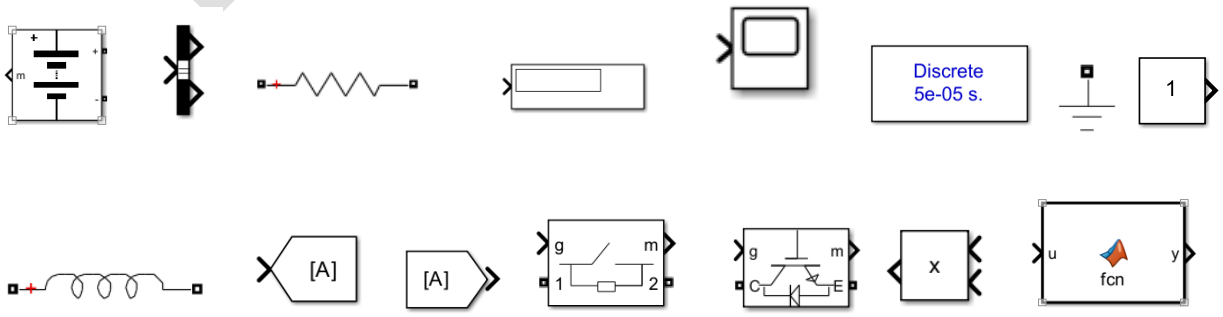
Open Simulink

- **Step 1**: Launch MATLAB and type Simulink in the command window to open Simulink.
- **Step 2**: Create a new model by selecting **File → New → Model**

2. Component selection:

Add the required component blocks by double-clicking on the Simulink canvas or by navigating through the Library Browser to select the necessary blocks

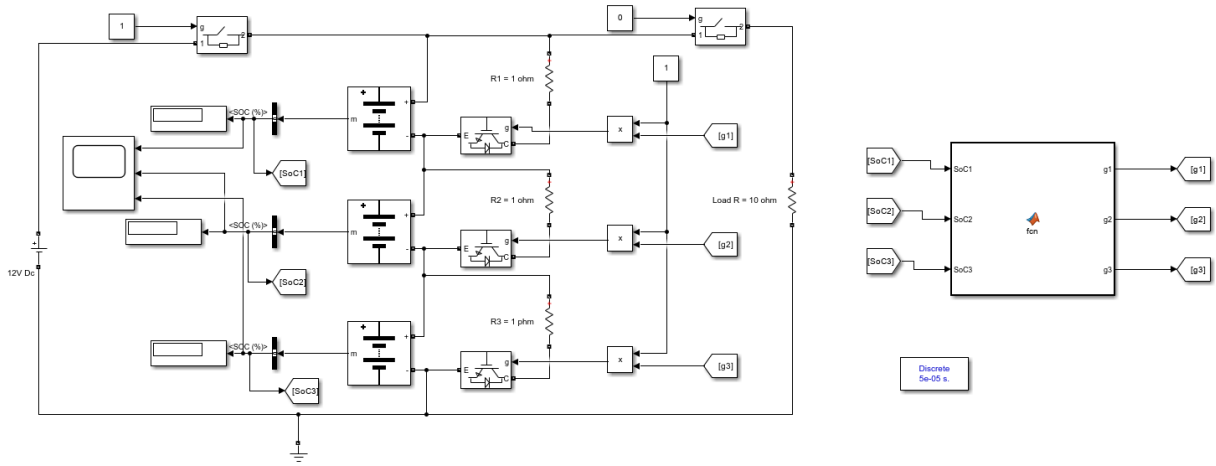




3. Connecting the blocks:

Connect all the components as per the following circuit

PASSIVE CELL BALANCING



4. MATLAB Function coding:

Passive_Cell_Balancing1 ▸ MATLAB Function

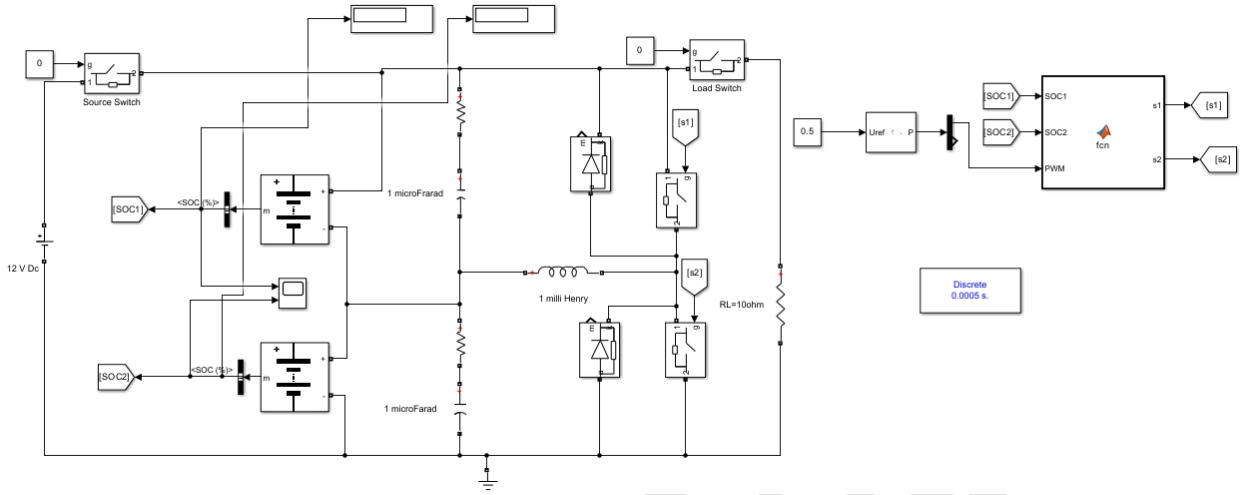
```
function [g1,g2,g3] = fcn(SoC1,SoC2,SoC3)

SoC1=int32(SoC1);
SoC2=int32(SoC2);
SoC3=int32(SoC3);

if SoC1>SoC2||SoC1>SoC3
    g1=1;
else
    g1=0;
end
if SoC2>SoC1||SoC2>SoC3
    g2=1;
else
    g2=0;
end
if SoC3>SoC1||SoC3>SoC2
g3=1;
else
    g3=0;
end

```

ACTIVE CELL BALANCING



5. MATLAB Function coding:

Active_Cell_Balancing1 ▸ MATLAB Function1

```
function [s1,s2] = fcn(SOC1,SOC2,PWM)
if SOC2<SOC1
    s1=PWM;
    s2=0;
elseif SOC1<SOC2
    s1=0;
    s2=PWM;
else
    s1=0;
    s2=0;
end

```

Double click on each block and change the required parameters

6. Working principle and control logic

In a lithium-ion battery pack, differences in cell voltages can lead to imbalance, reducing performance and lifespan. Cell balancing ensures uniform charge distribution among cells.

- **Passive balancing** dissipates excess charge from higher-voltage cells as heat through resistors until all cells reach the same voltage.

- **Active balancing** uses energy transfer circuits—such as the **resonant switched capacitor topology**—to move charge from higher-voltage cells to lower-voltage ones through capacitors and switches.

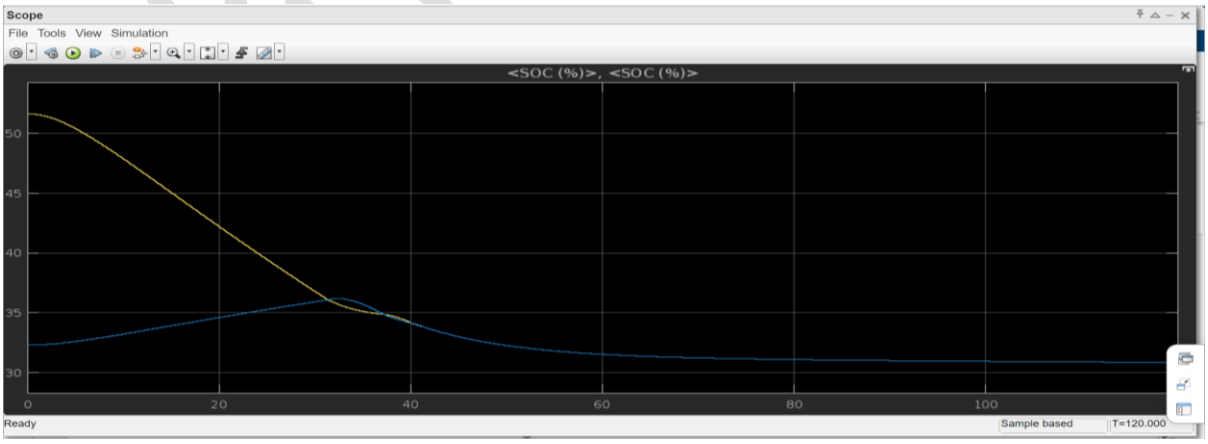
In the resonant switched capacitor method, a resonant circuit (inductor and capacitor) enables efficient energy transfer with reduced losses and faster balancing compared to passive methods.

7.

Simulation Results for ACTIVE CELL BALANCING - Charging Mode (CASE 1)

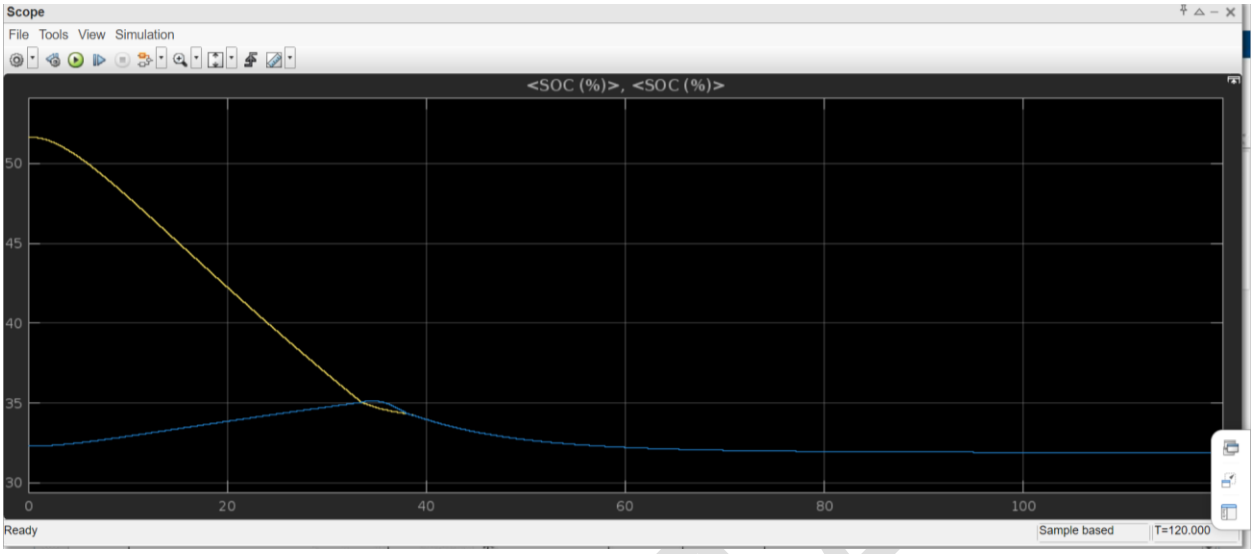
Trigger the source switch to connect the DC source to battery units, which charges cells to 100 %, after balancing to same SOC level (Assume Initial SoC of B1 = 50% and SoC of B2 = 30%)

Simulation Results - Discharging Mode (CASE 2)



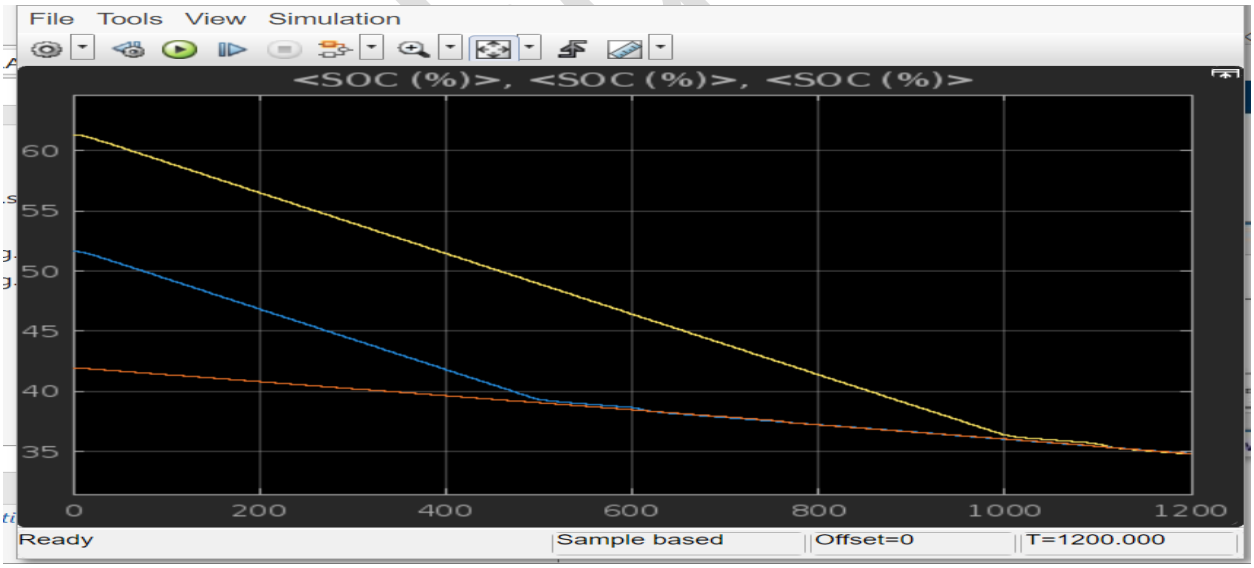
Trigger the load switch to connect the load to battery units, which discharges the cells based on connected load requirement, after balancing to same SOC level (Assume Initial SoC of B1 = 50% and SoC of B2 = 30%)

Simulation Results - Cell Balancing Mode (CASE 3)



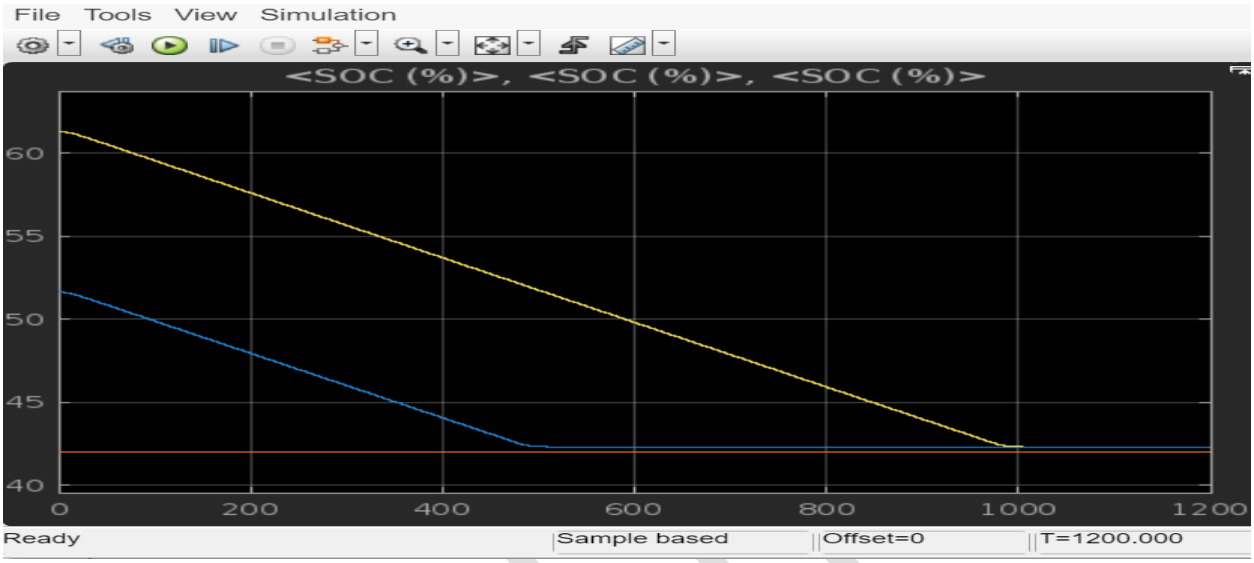
Disconnect both load and source switch, run simulation to see the cells balancing each other based on SOC of each cell (Assume Initial SoC of B1 = 50% and SoC of B2 = 30%)

Simulation Results PASSIVE CELL BALANCING - Discharging Mode (CASE 2)



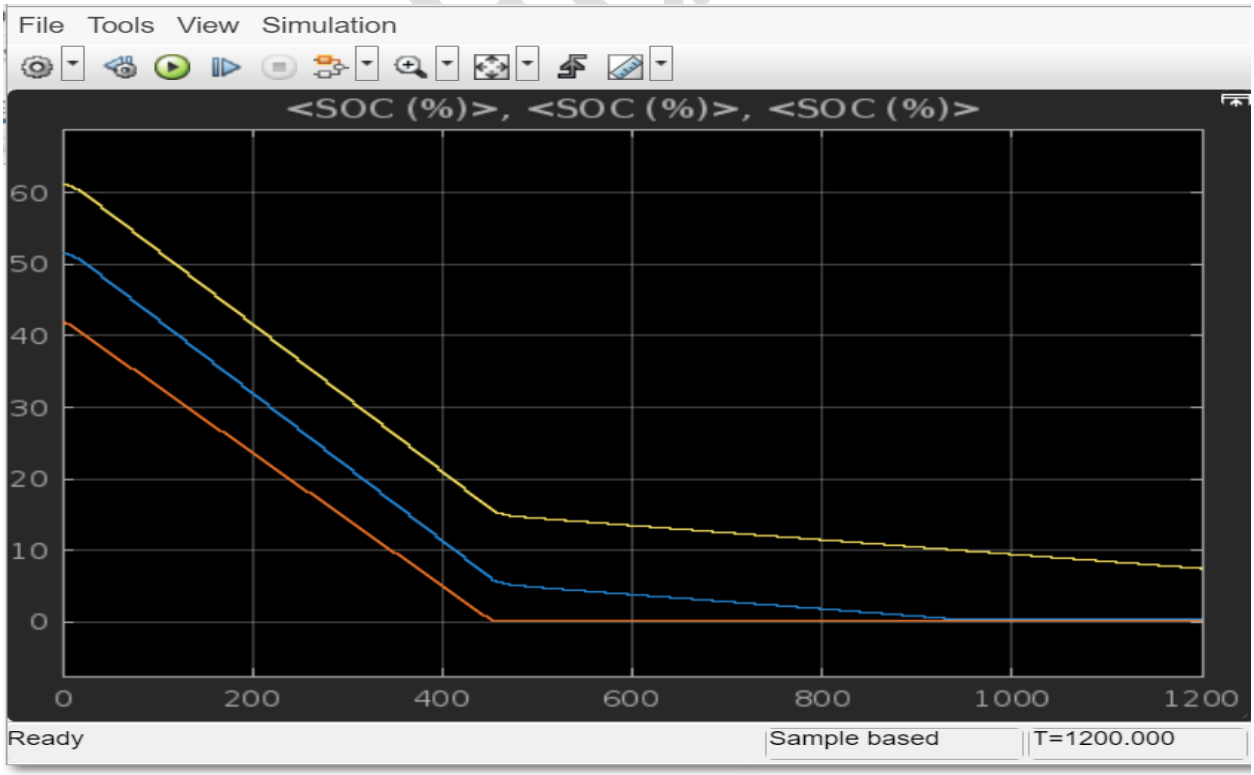
Trigger the load switch to connect the load to battery units, which discharges the cells based on connected load requirement, after balancing to same SOC level (Assume Initial SoC of B1 = 60%, B2 = 50% and B3 = 40%)

Simulation Results - Cell Balancing Mode



Disconnect both load and source switch, run simulation to see the cells balancing each other based on SOC of each cell (Assume Initial SoC of B1 = 60%, B2 = 50% and B3 = 40%)

Simulation Results - Both Charging and Discharging Mode



8.Conclusion:

The simulation results show that both active and passive balancing improve voltage uniformity among lithium-ion cells. However, the **active resonant switched capacitor method** provides **faster balancing, higher efficiency, and reduced energy loss** compared to passive balancing, making it more suitable for high-performance and large-capacity battery applications.

Model and simulate a single lithium-ion cell in MATLAB/Simulink, incorporating its electrical characteristics such as open-circuit voltage, internal resistance, and dynamic behavior.

1. **Introduction:**

- **Objective**: Extend the model to simulate a battery pack with multiple lithium-ion cells connected in series and series-parallel configurations.
- **Tools Used**: Simulink is used to model the battery used to create battery.

2. **Model setup:**

Open Simulink

- **Step 1**: Launch MATLAB and type Simulink in the command window to open Simulink.
- **Step 2**: Create a new model by selecting **File → New → Model**

3. **Component selection:**

Battery

Current measurement

Voltage measurement

Series RLC Branch

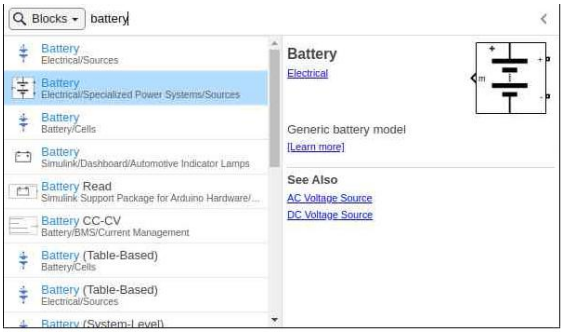
Bus Selector

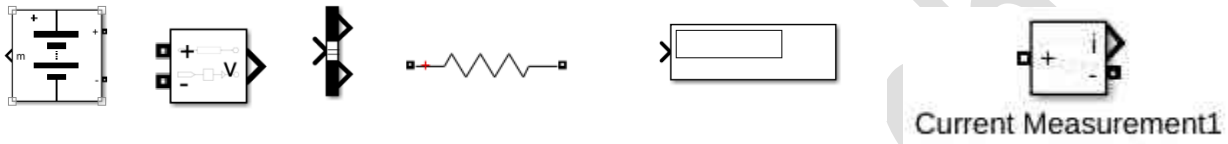
Display

Powergui

Scope

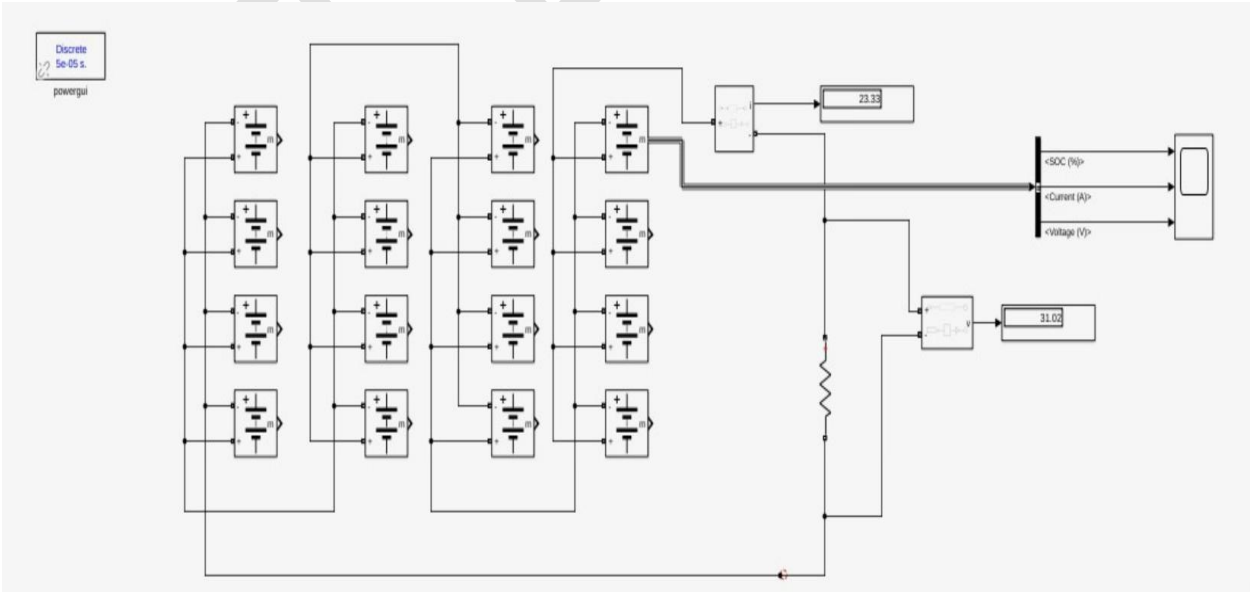
Add the required component blocks by double-clicking on the Simulink canvas or by navigating through the Library Browser to select the necessary blocks



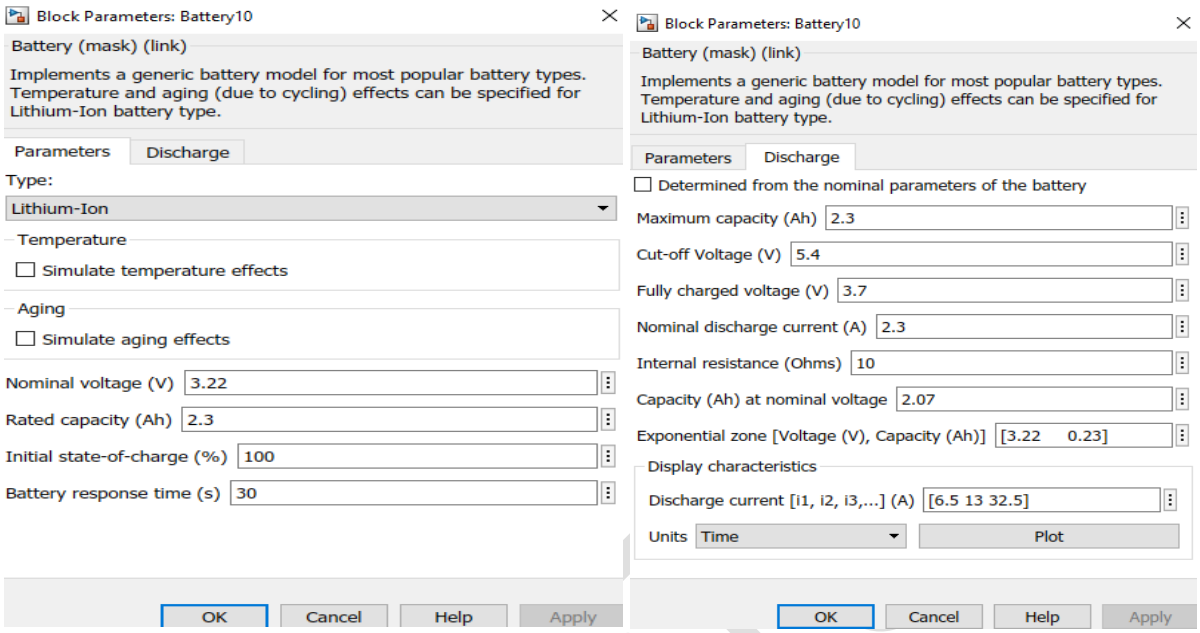


4. Connecting the blocks:

Connect all the components as per the following circuit



5. Parameter setting:



Double click on each block and change the required parameters

6. Working principle and control logic

We represent a single Li-ion cell by a reduced electrical/electrochemical equivalent that captures:

- **Open-circuit voltage (OCV)** as a function of state-of-charge (SOC). OCV(SOC) is the battery's equilibrium voltage (no current).
- **Internal (ohmic) resistance R0** — instantaneous voltage drop when current flows.
- **Dynamic polarization (slow voltage dynamics)** — usually modeled with one or more RC branches (Thevenin / R-C networks) to capture transient voltage relaxation (diffusion, charge transfer).
- **SOC dynamics (charge conservation / coulomb counting)** — how the SOC evolves from charge/discharge current.
  Optional additions: thermal model, capacity fade, multiple RC branches for high-fidelity dynamics.

Core equations (single-RC Thevenin style)

Use these variables and sign convention: discharge current I>0 (A), charge I<0. Nominal capacity Q_nom in Ah (convert to coulombs where needed).

1. SOC dynamics (coulomb counting):

$$\frac{d\,SOC}{dt} = -\frac{I}{3600 \cdot Q_{nom}}$$

(3600 converts Ah → Coulombs)

2. RC dynamics (one RC branch; V_{rc} is voltage across the RC element that subtracts from terminal voltage):

$$\frac{dV_{rc}}{dt} = -\frac{1}{R_1 C_1} V_{rc} + \frac{1}{C_1} I$$

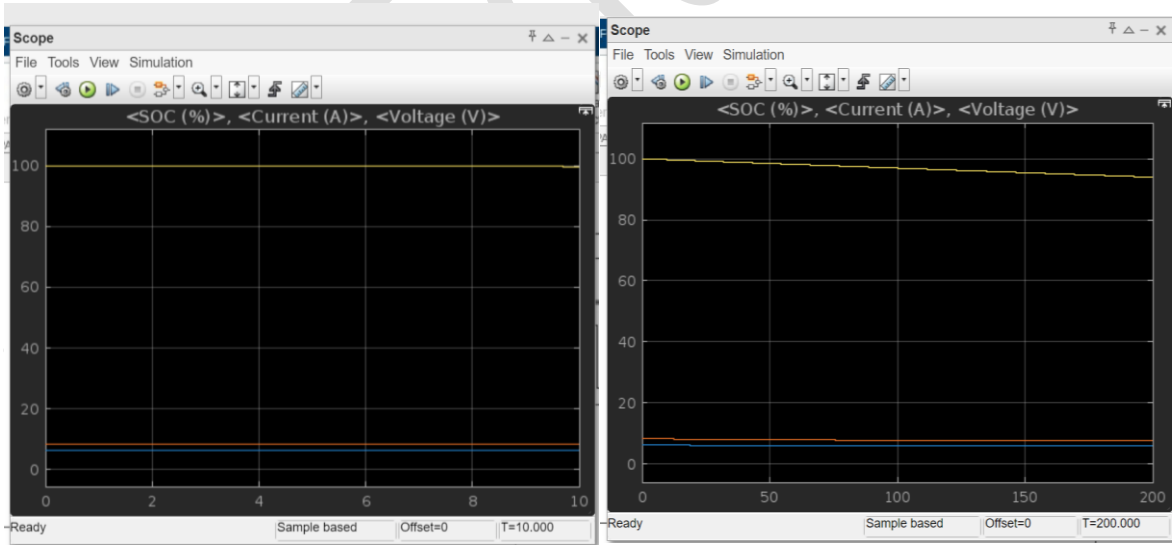
3. Terminal voltage:

$$V_t = OCV(SOC) - IR_0 - V_{rc}$$

Notes:

- OCV(SOC) is modeled with a lookup table (measured curve) or fitted polynomial.
- To increase accuracy, add more RC branches: $V_t = OCV - IR_0 - \sum V_{rc,i}$ with each $\dot{V}_{rc,i} = -\frac{1}{R_i C_i} V_{rc,i} + \frac{1}{C_i} I$.
- Ensure SOC is clamped to $[0,1]$.

7.Results:



At the end of 10 Sec Discharge Time, SOC = 99% **At the end of 200 Sec Discharge Time, SOC=93%**

8.Conclusion:

The experiment demonstrates that the A single lithium-ion cell can be effectively modeled in MATLAB/Simulink, which includes open-circuit voltage (OCV), internal resistance (R_0), and an RC network to capture dynamic behavior. This model accurately simulates the cell's voltage response, SOC variation, and transient effects during charge and discharge, making it suitable for battery performance analysis, control design, and system integration studies.

Model and simulate a three-phase inverter circuit in MATLAB/Simulink to drive a BLDC motor

Introduction:

- **Objective**: To model and simulate a three-phase inverter circuit in MATLAB/Simulink for driving a BLDC motor, and to analyze its performance in terms of speed control, torque response, and commutation characteristics.
- **Tools Used**: Simulink & Simscape is used to model the three-phase inverter circuit.

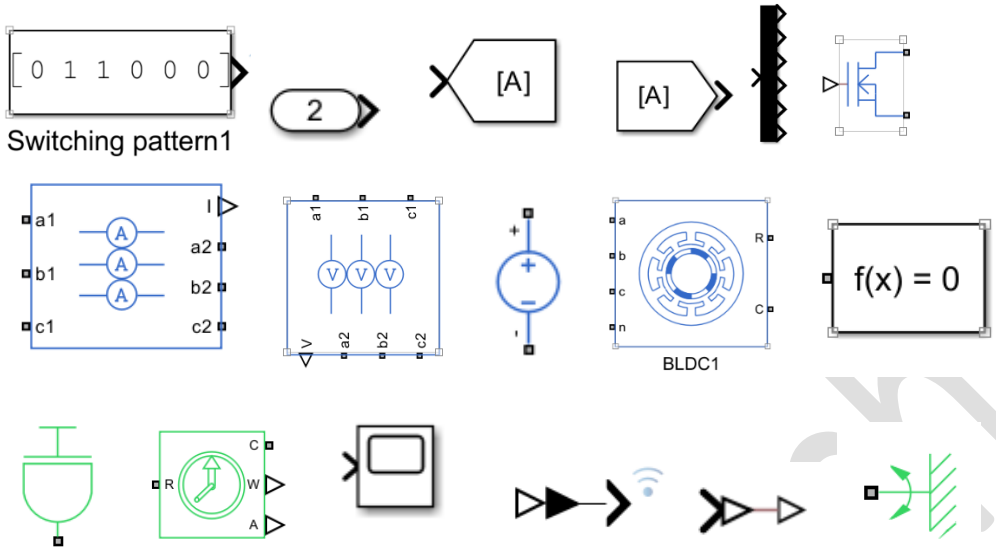
1. **Model setup:**

   Open Simulink

   - **Step 1**: Launch MATLAB and type Simulink in the command window to open Simulink.
   - **Step 2**: Create a new model by selecting **File → New → Model**

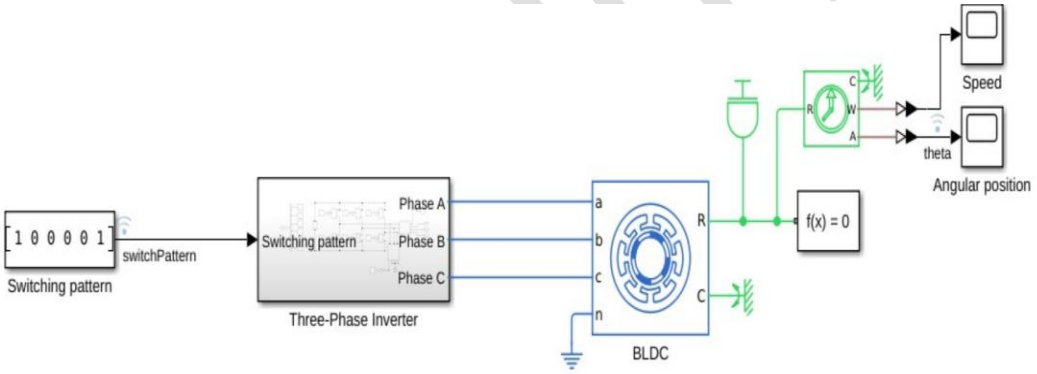
2. **Components used:**
   - Constant
   - Subsystem
   - BLDC Motor
   - Electrical reference
   - Solver configuration
   - Inertia
   - Mechanical Rotational Reference
   - Ideal Rotational Motion Sensor
   - Inport
   - Demux
   - Goto
   - Voltage source
   - From
   - Simulink-PS converter
   - MOSFET (Ideal switching)
   - Three phase voltage sensor
   - Three phase current sensor
   - Connection port
   - PS – Simulink converter
   - scope

   Add the required component blocks by double-clicking on the Simulink canvas or by navigating through the Library Browser to select the necessary blocks

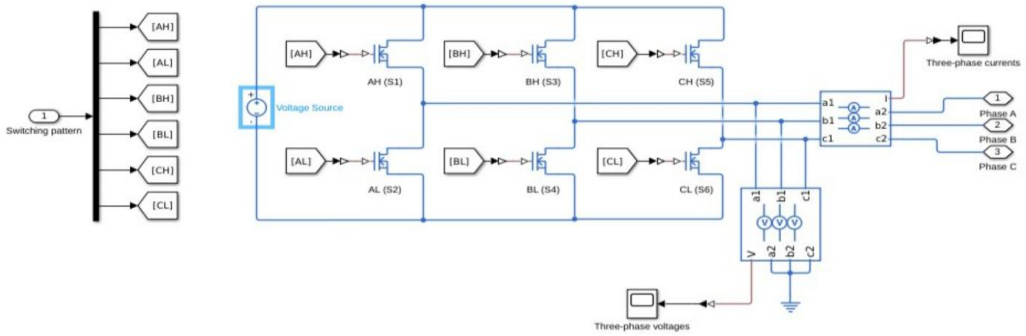


3. Connecting the blocks:

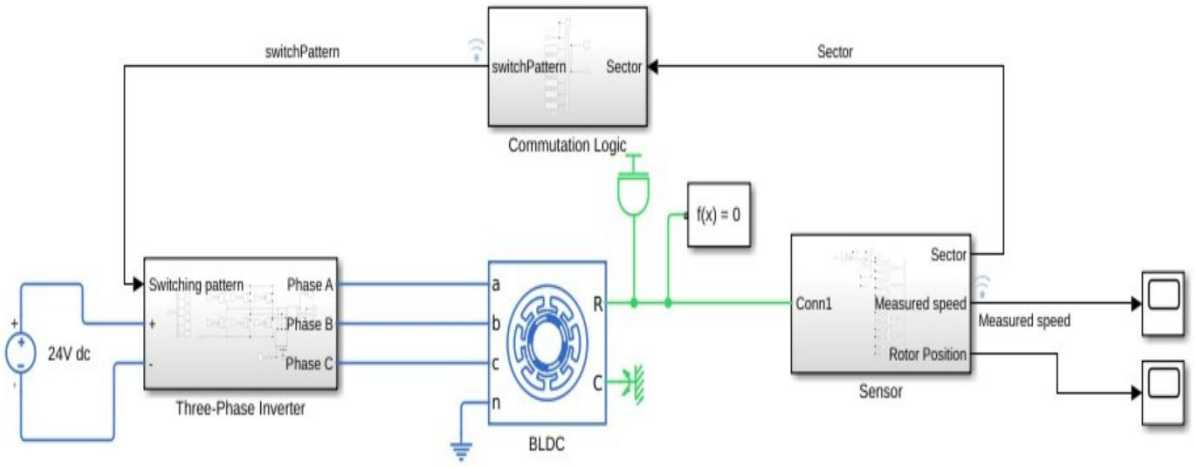
Connect all the components as per the following circuit



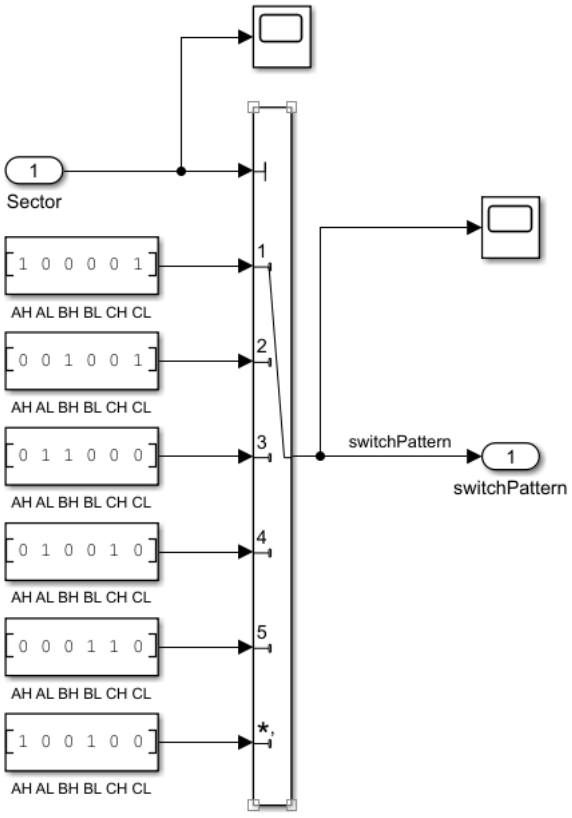
4. Subsystem:

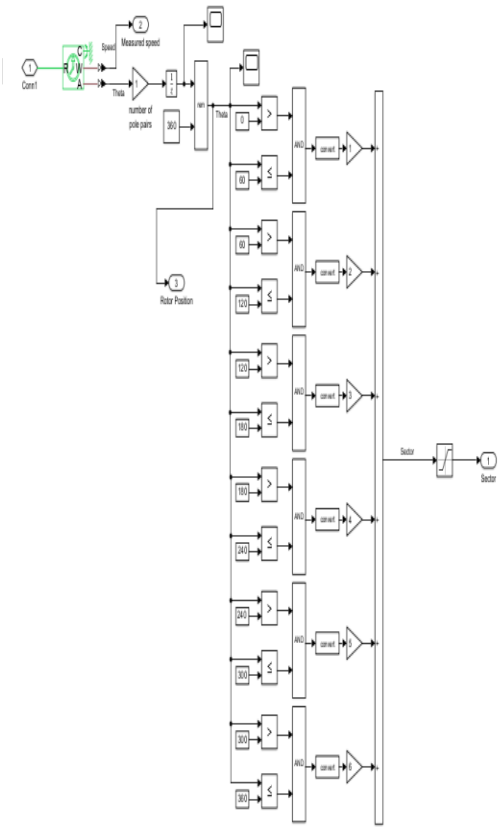


Trapezoidal Commutation of BLDC Motor in MATLAB



Switching Pattern – Implementation in MATLAB





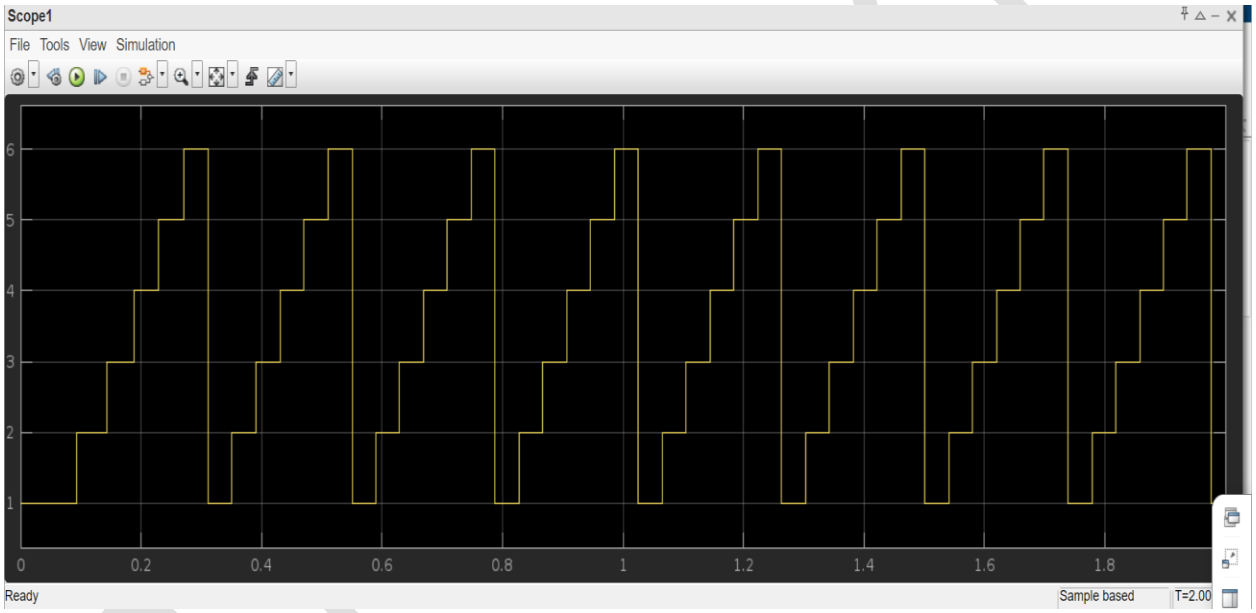
5. Working principle and control logic

A **three-phase inverter** converts DC power into three-phase AC power to drive a **Brushless DC (BLDC) motor**. The inverter uses six power switches (typically MOSFETs or IGBTs) arranged in a three-leg configuration to generate a three-phase output.

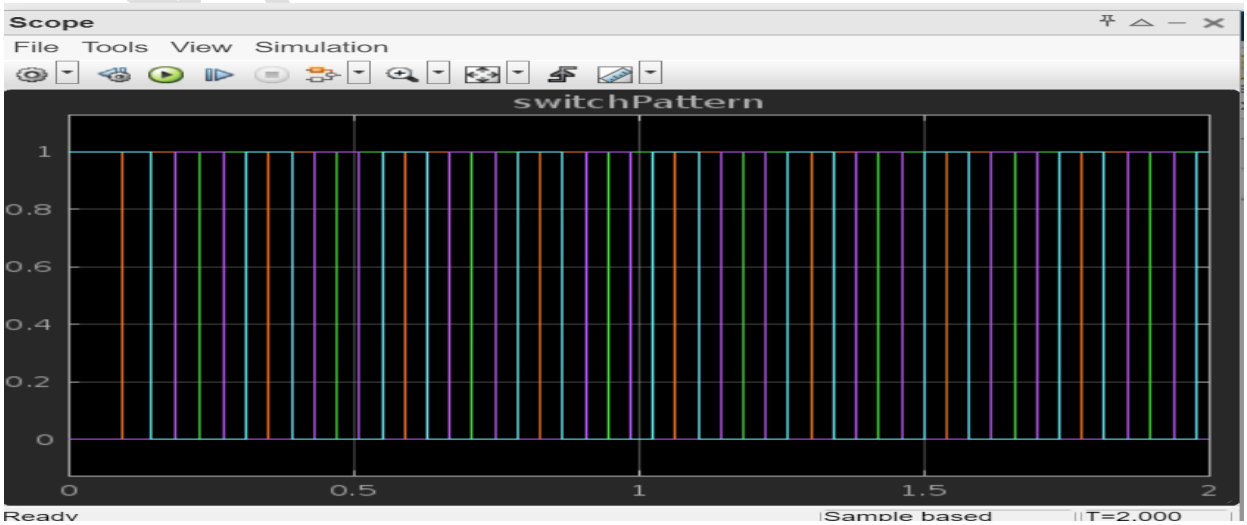
By sequentially switching these devices according to rotor position signals from Hall sensors, the inverter produces a **trapezoidal back-EMF** waveform that drives the motor. This controlled commutation ensures continuous torque production and efficient speed control of the BLDC motor.

7.Results:

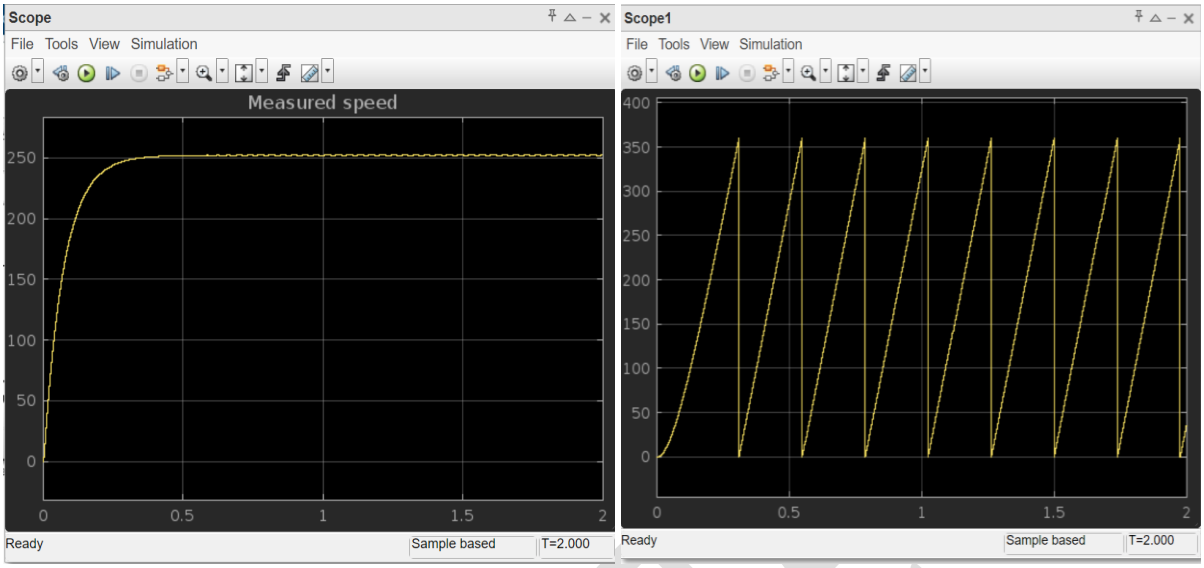
Simulation Results - Sector Selection



Simulation Results - Switching Pattern



Simulation Results - Speed and Rotor Position



8.Conclusion:

The simulation demonstrates that the three-phase inverter effectively drives the BLDC motor by converting DC to controlled three-phase AC supply. Proper switching sequence and commutation ensure smooth torque, efficient speed control, and reliable motor operation suitable for various industrial and electric vehicle applications.

PWM Controlled Buck Converter for Speed Control of BLDC Motor for EV Applications

Introduction:

- **Objective**: To design and simulate a PWM-controlled buck converter for regulating the DC voltage supplied to a BLDC motor, enabling efficient speed control suitable for electric vehicle (EV) applications.
- **Tools Used**: Simulink & Simscape is used to model the PWM Controlled Buck Converter for Speed Control of BLDC Motor.

1. **Model setup:**

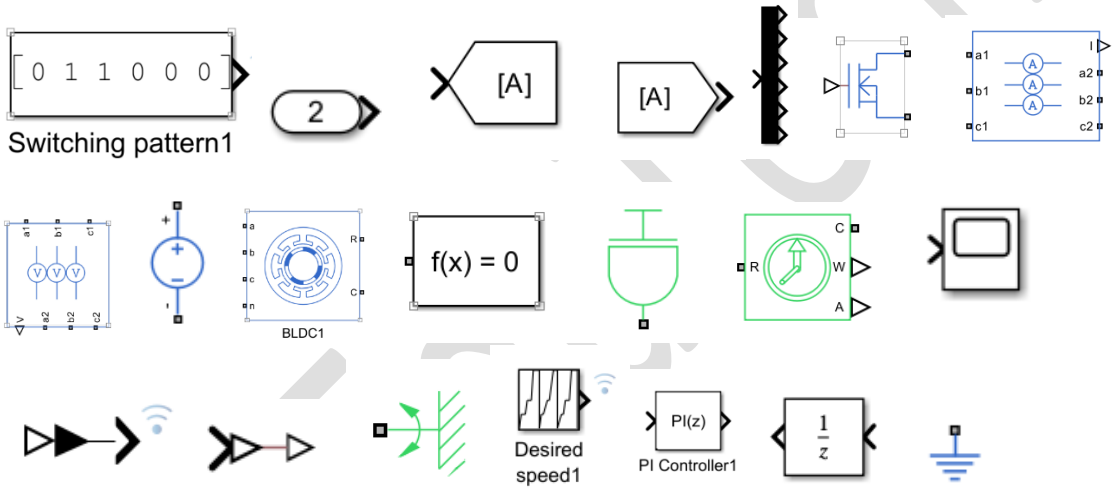
   Open Simulink

   - **Step 1**: Launch MATLAB and type Simulink in the command window to open Simulink.
   - **Step 2**: Create a new model by selecting **File → New → Model**

2. **Components used:**
   - Repeating Table
   - Subsystem
   - PID Controller
   - Sum
   - Solver configuration
   - Unit delay
   - powergui
   - Mechanical Rotational Reference
   - Ideal Rotational Motion Sensor
   - Inport
   - PWM Generator
   - Logical Operator
   - Data type conversion
   - Inductor
   - Capacitor
   - Simulink – PS Converter
   - DC voltage source
   - Voltage sensor
   - Terminator
   - Multiport switch
   - Gain
   - Relational operator
   - Outport
   - Demux

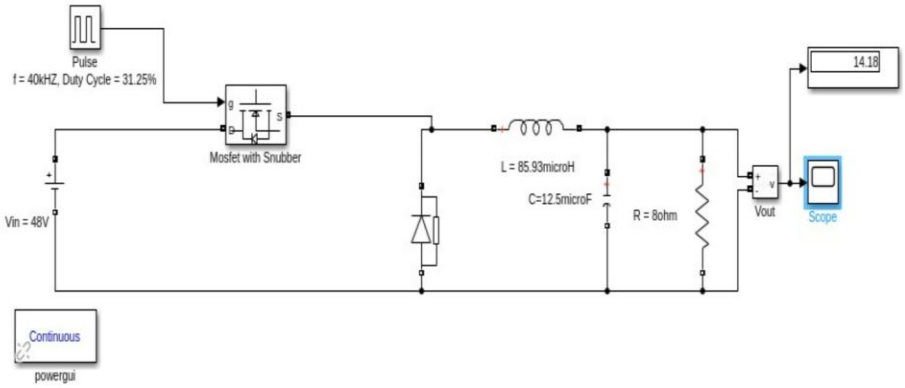
- Goto
- Voltage source
- From
- MOSFET (Ideal switching)
- Three phase voltage sensor
- Three phase current sensor
- Connection port
- PS – Simulink converter
- scope

Add the required component blocks by double-clicking on the Simulink canvas or by navigating through the Library Browser to select the necessary blocks

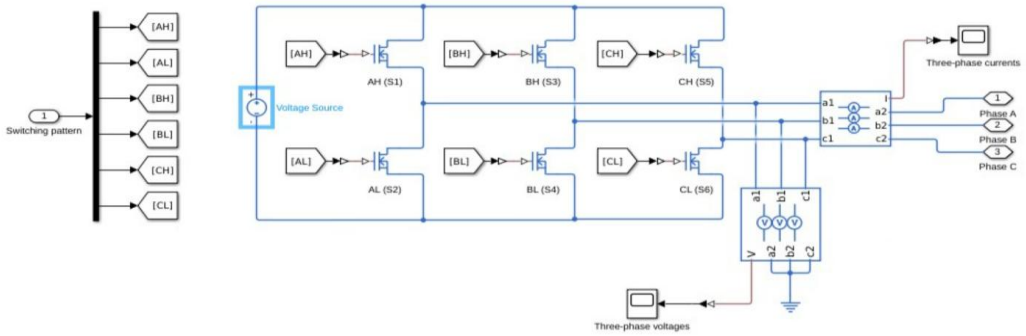


3. Connecting the blocks:

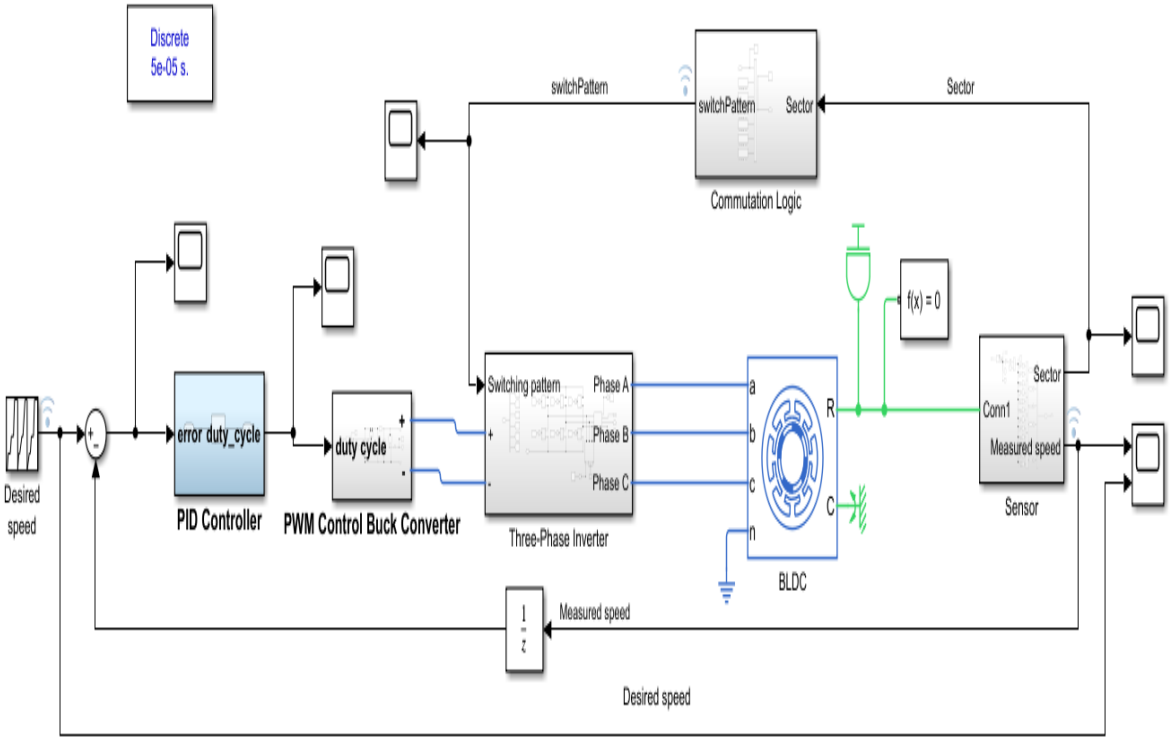
Connect all the components as per the following circuit



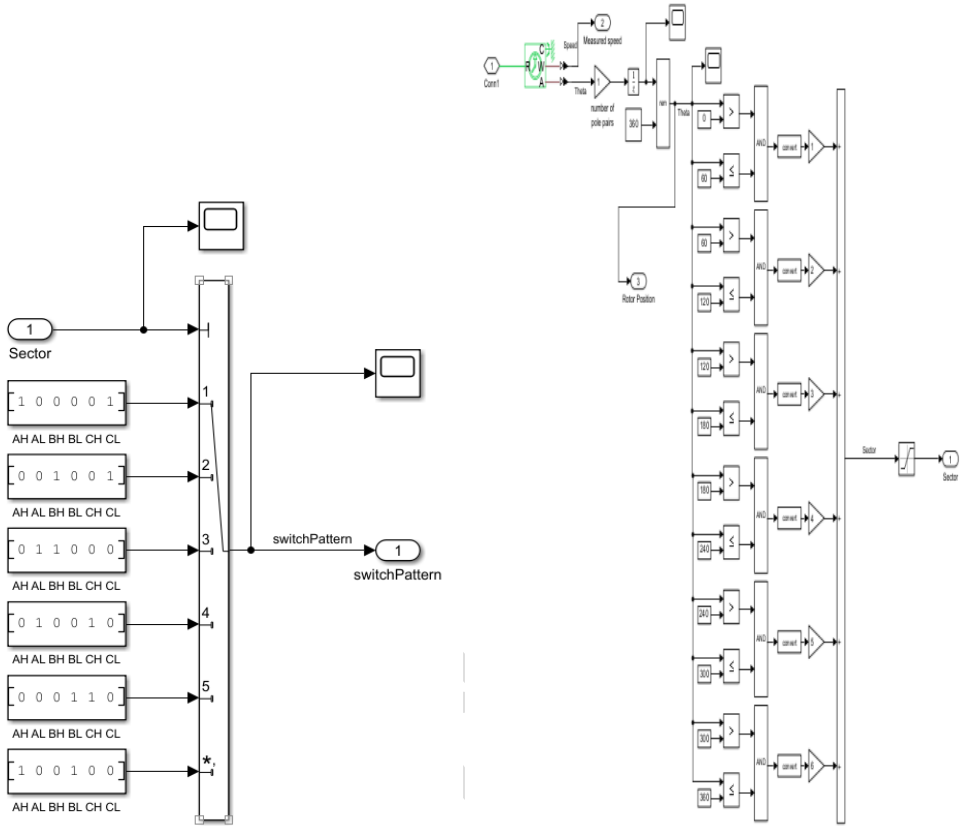
4. Subsystem:



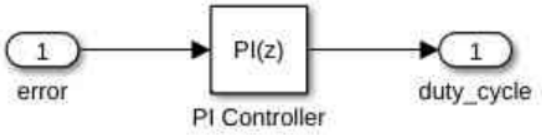
MATLAB Simulation Model – PWM Controlled Buck Converter for Speed Control of BLDC Motor

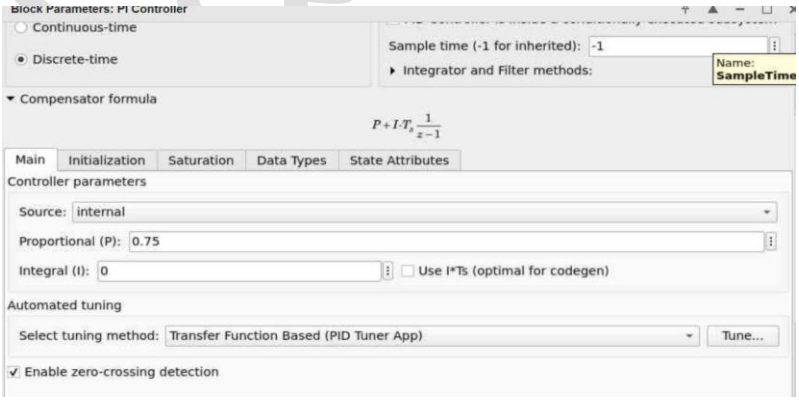


Trapezoidal Commutation of BLDC Motor in MATLAB Switching Pattern – Implementation in MATLAB



Subsystem – PI Controller





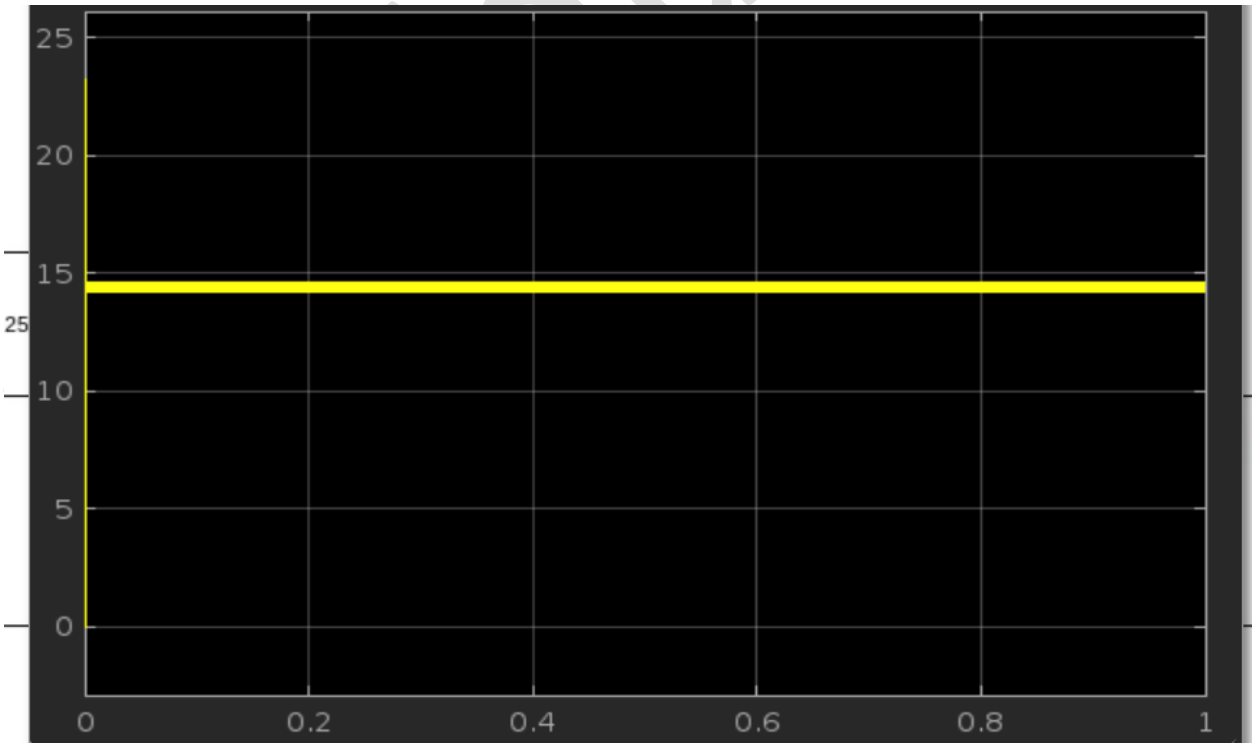
5. Working principle and control logic

A PWM-controlled buck converter reduces the input DC voltage to a lower level suitable for controlling the speed of a BLDC motor. By varying the duty cycle of the PWM signal applied to the converter's switching device (usually a MOSFET), the output voltage—and hence the motor speed—is adjusted.

When the switch is ON, energy is stored in the inductor; when it is OFF, the stored energy is released to the motor through the freewheeling diode. This continuous switching maintains a smooth DC output, providing efficient and precise speed control for EV applications.

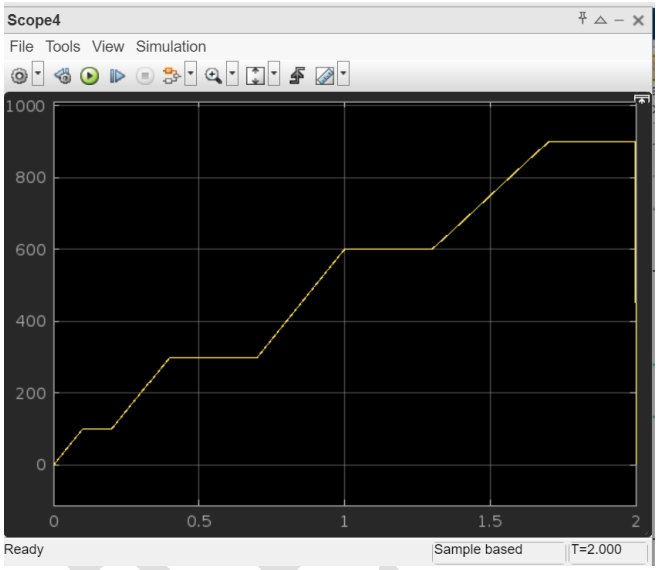
7.Results:

MATLAB Simulation Results of Buck Converter

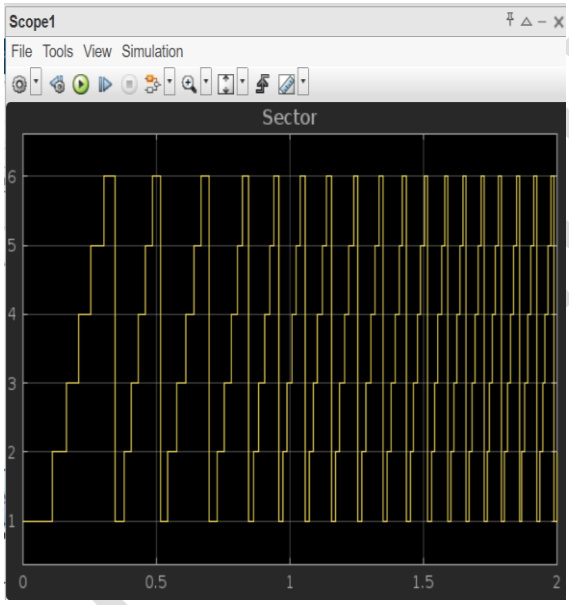


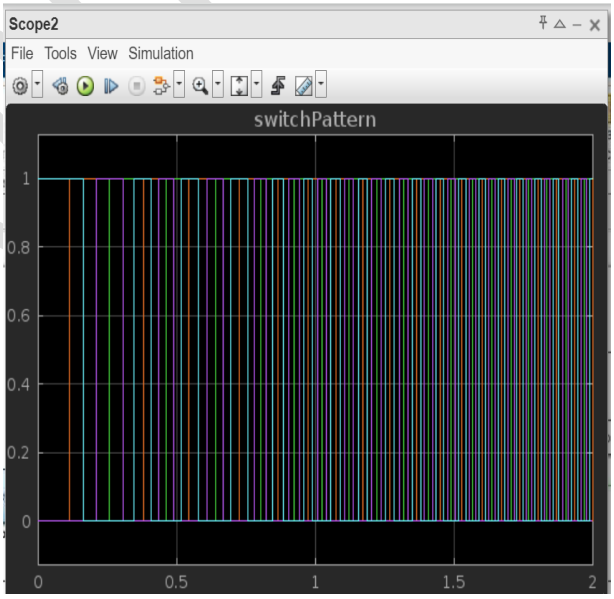
Simulation Results - CASE I - Desired Speed Characteristics - Input

S. No.	Time (s)	Speed (rpm)
1	0	0
2	0.1	100
3	0.2	100
4	0.4	300
5	0.7	300
6	1.0	600
7	1.3	600
8	1.7	900
9	2.0	900

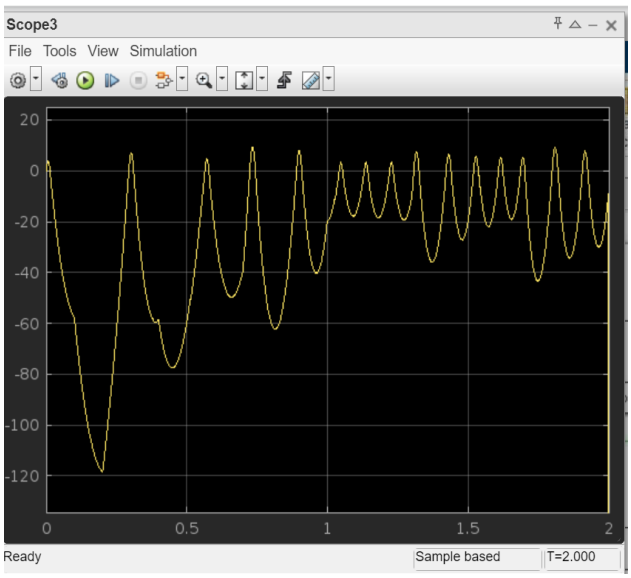


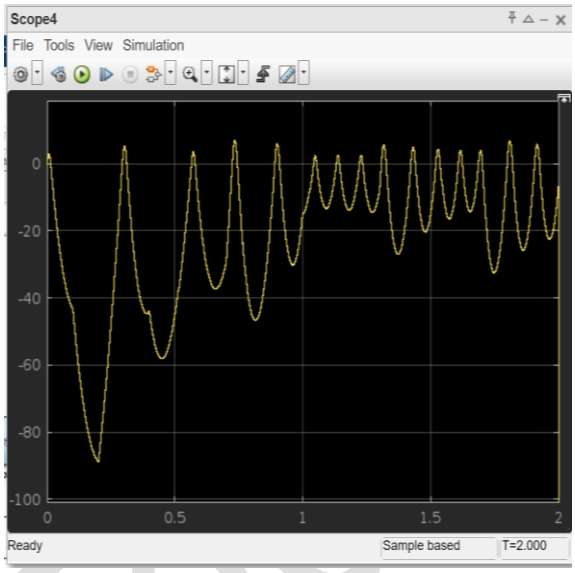
Simulation Results - Sector Selection and Switching Pattern



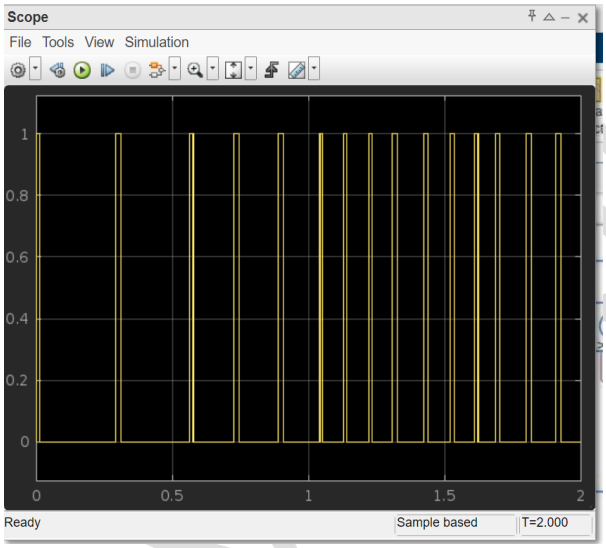


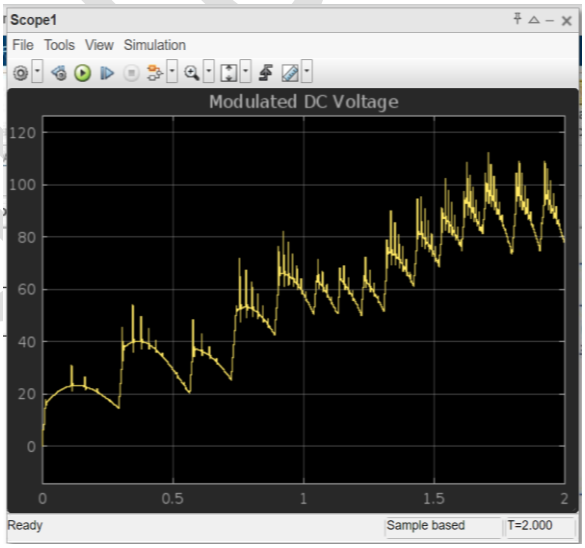
Simulation Results - Error Signal and Duty Cycle



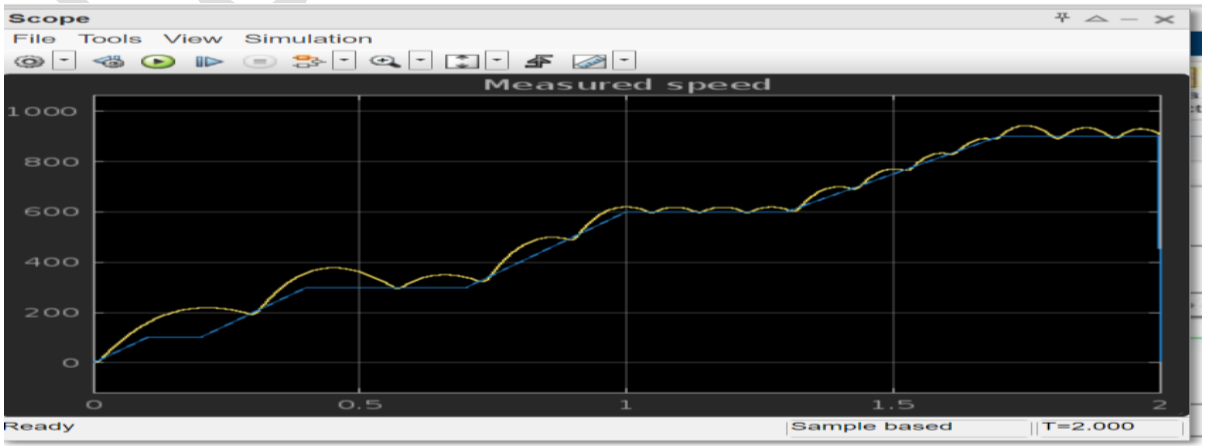


Simulation Results - Pulse Generation and Modulated DC Voltage of Buck Converter

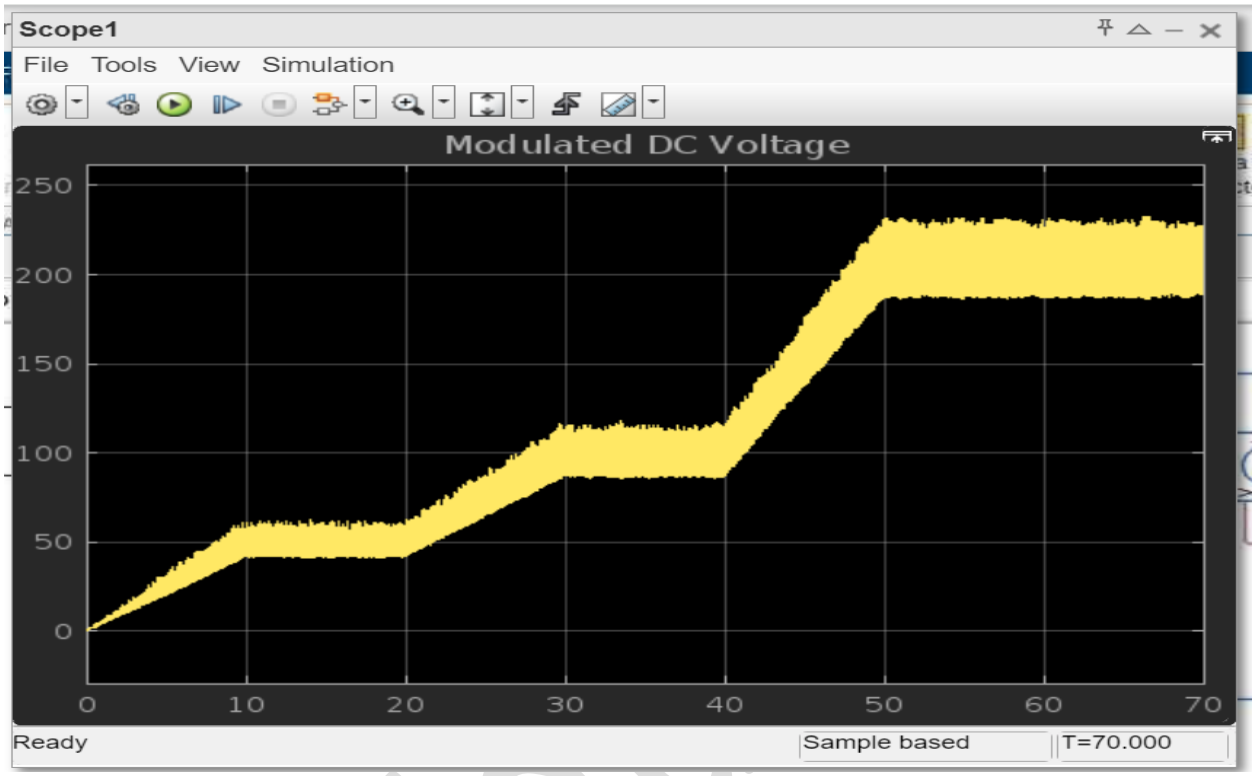




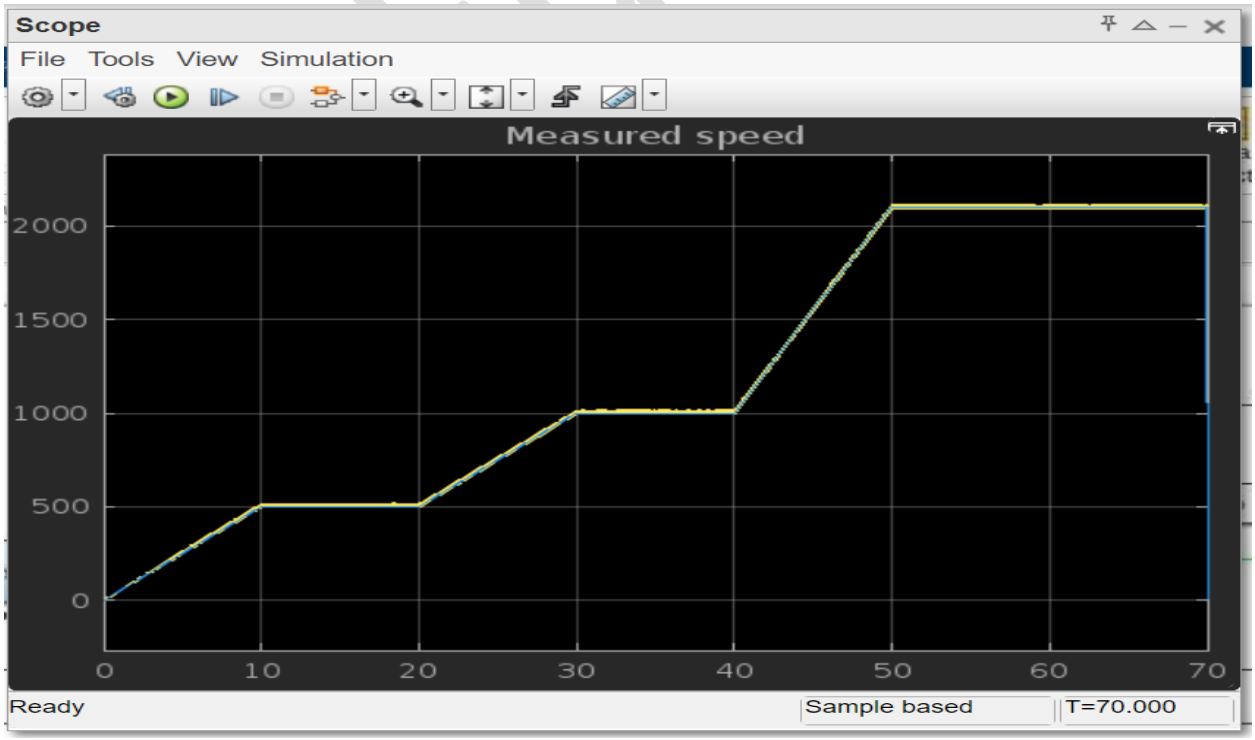
Simulation Results - Measured Speed Vs Desired Speed



CASE II - Desired Speed Characteristics - Input



Simulation Results - Measured Speed Vs Desired Speed



8.Conclusion:

The PWM-controlled buck converter effectively regulates the voltage supplied to the BLDC motor, enabling precise and efficient speed control. This approach improves energy efficiency, reduces losses, and provides reliable motor performance, making it well-suited for electric vehicle applications.

Demonstrate the use of the Battery Pack Building App in MATLAB/Simulink to design and visualize battery pack layouts, and analyze how configuration choices impact energy storage and power delivery in different applications.

Introduction:

- **Objective:**
  To design and visualize battery pack layouts using the Battery Pack Building App in MATLAB/Simulink, and to analyze the effect of different cell configurations on the energy capacity and power performance for various applications.
- **Tools Used**: Simulink is used to model the battery and Battery Builder app used to create battery packs.

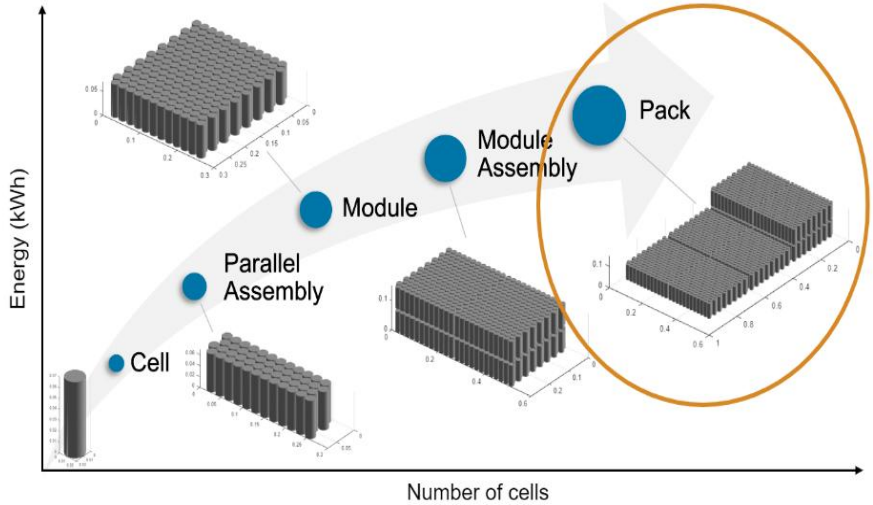
1. **Model setup:**

   Open Simulink

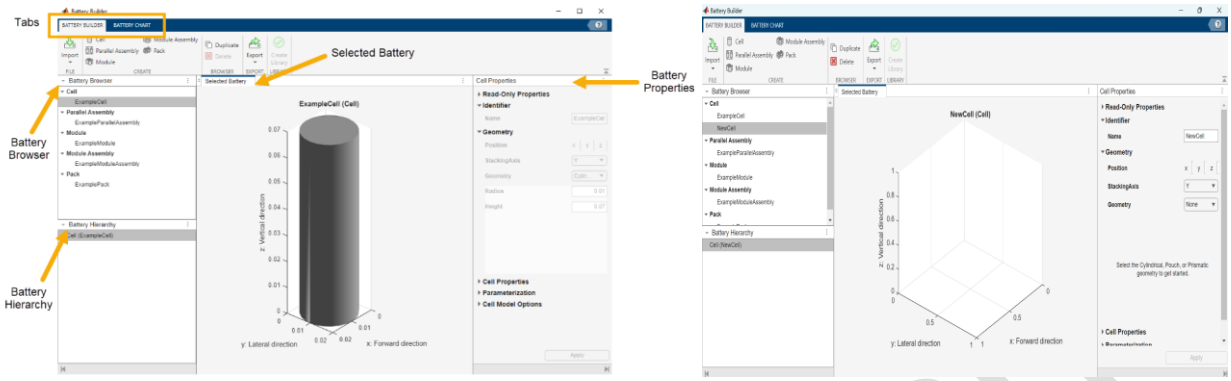
   - **Step 1**: Launch MATLAB and type Simulink in the command window to open Simulink.
   - **Step 2**: Create a new model by selecting **File → New → Model**
   - Open Battery Builder App in Simscape top ribbon
   - openExample('simscapebattery/BuildABatteryPackModelInSimscapeExample')

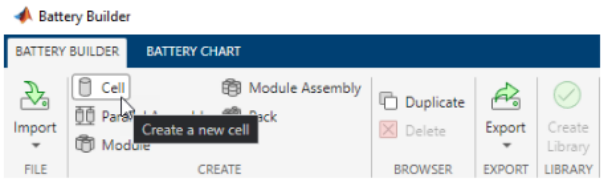
2. **Component selection:**

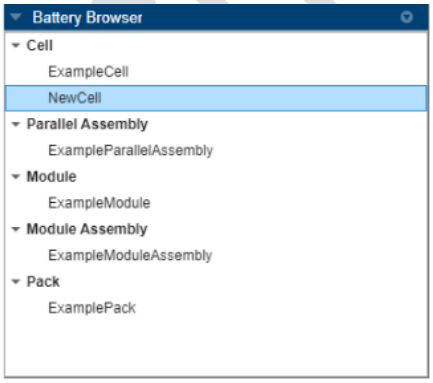
   Add the required component blocks by double-clicking on the Simulink canvas or by navigating through the Library Browser to select the necessary blocks

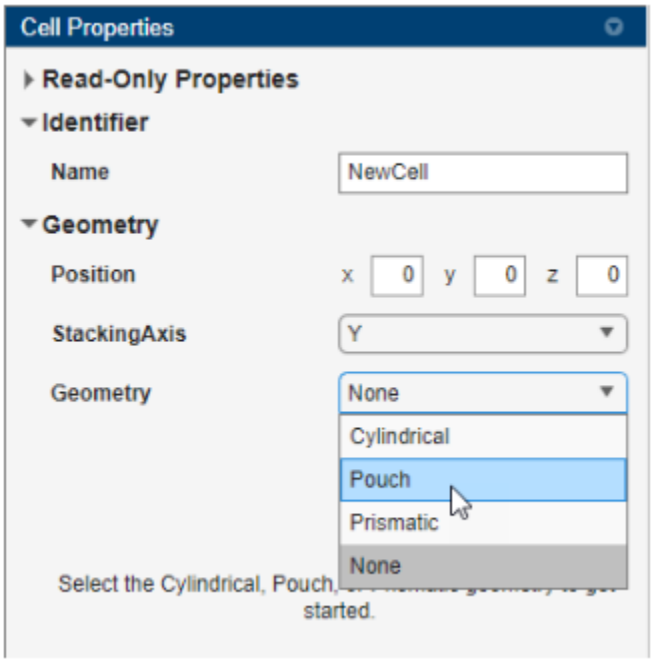


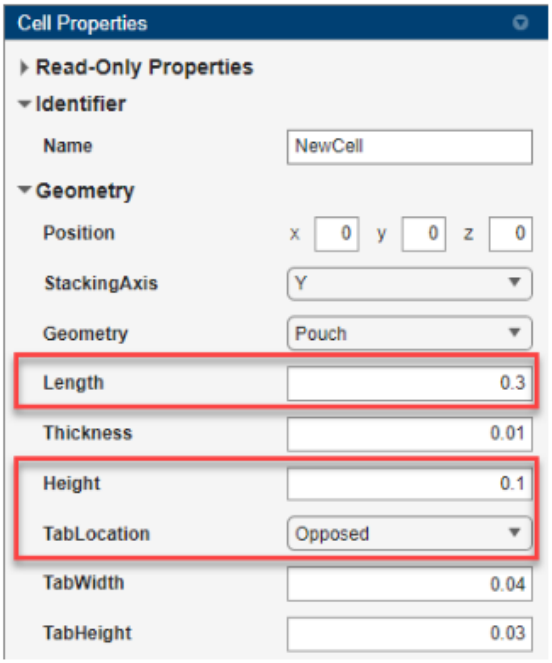
Create Cell



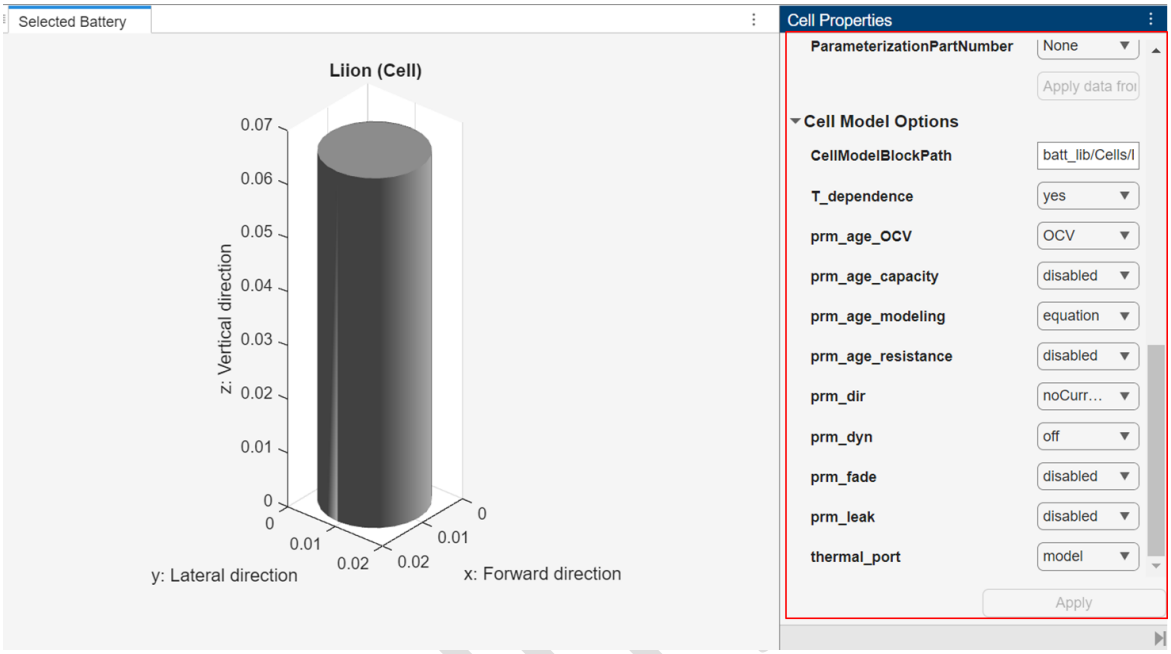


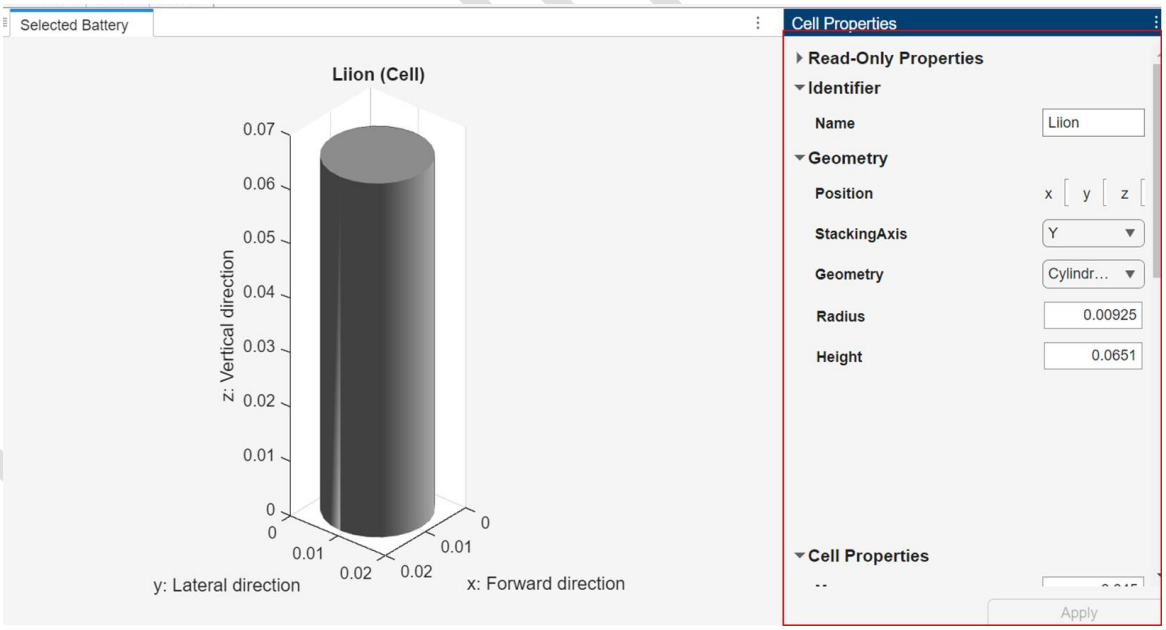




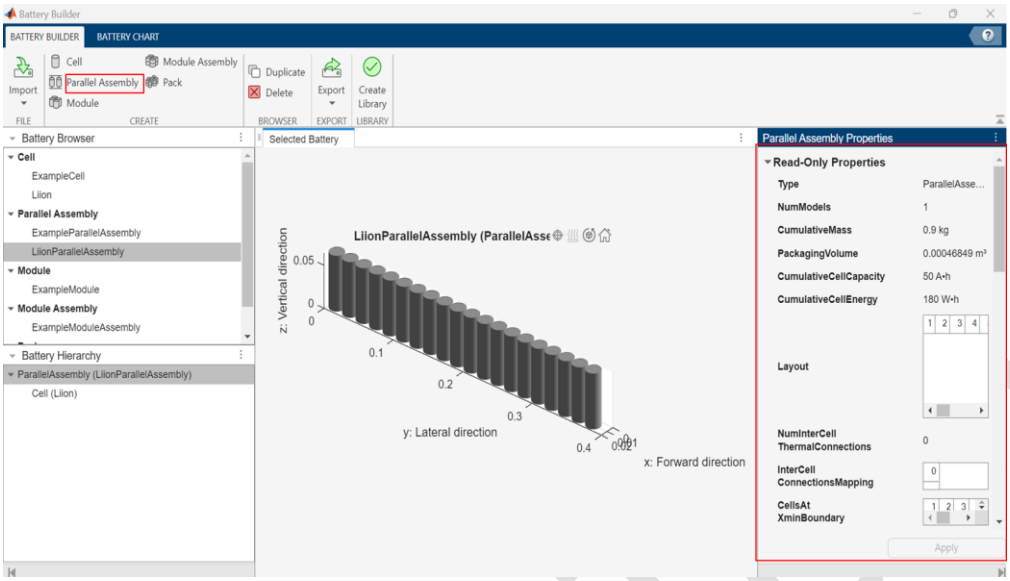


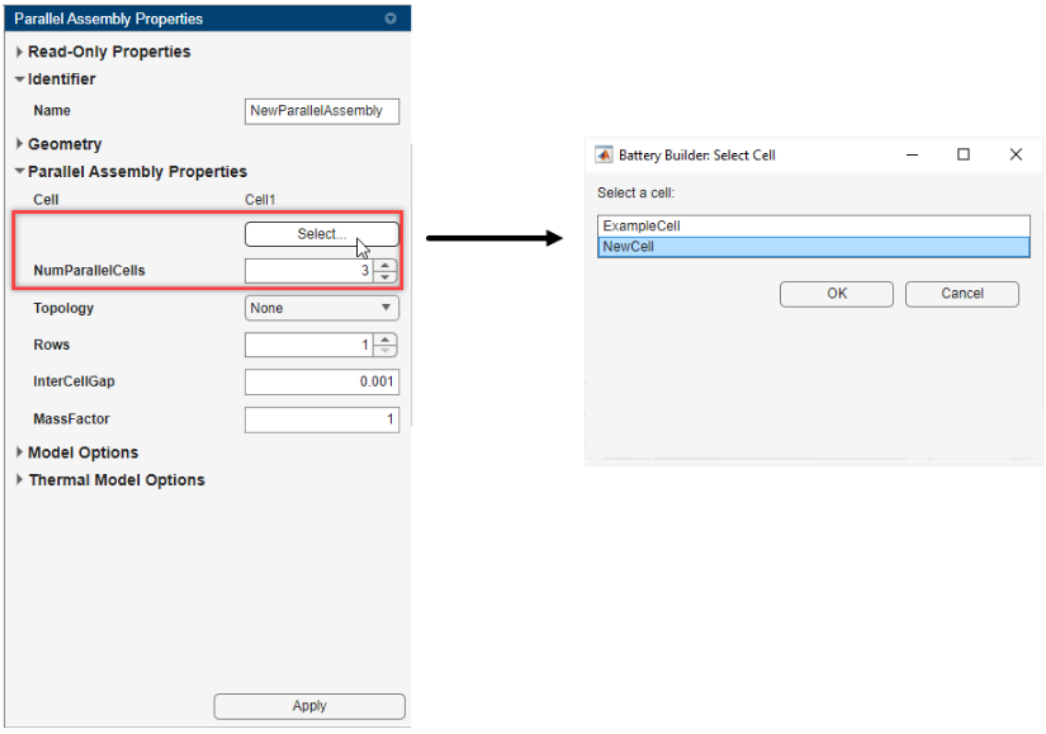
Keying in Geometrical and Electrical Specification of Cell

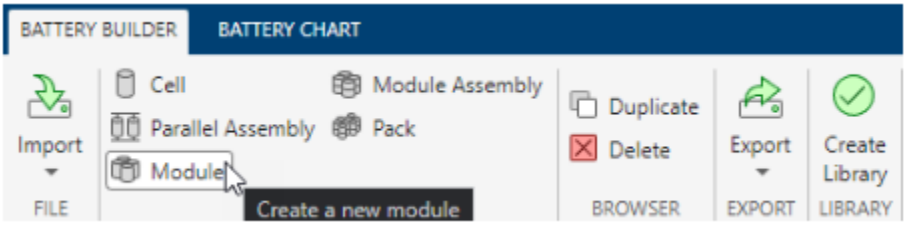


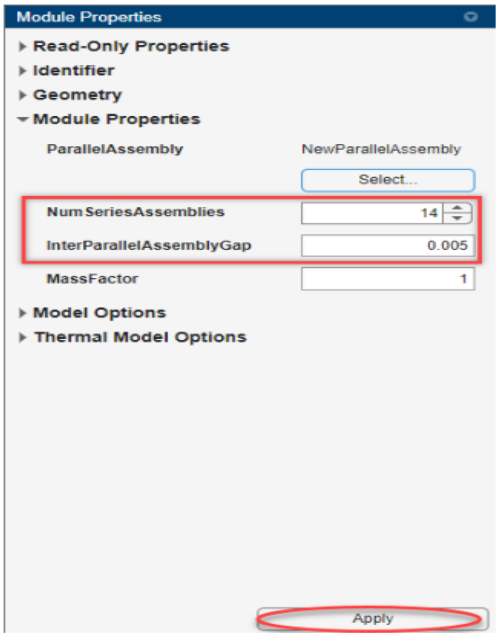


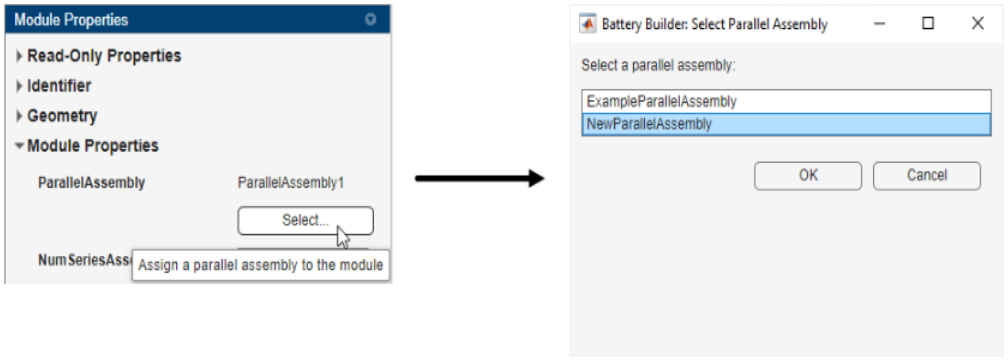
Create Parallel Assembly



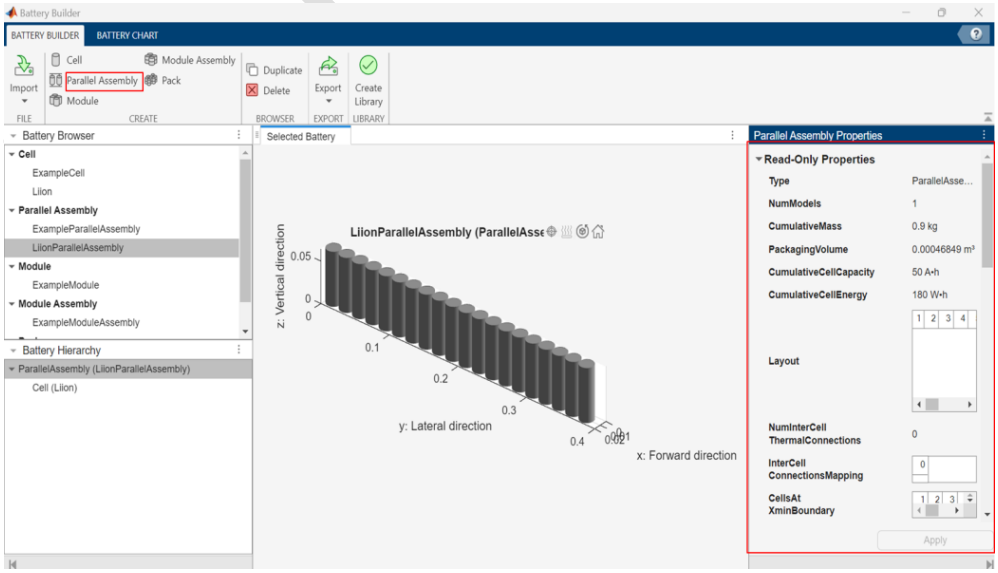




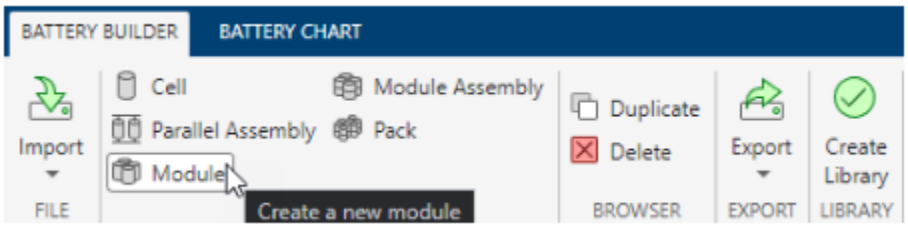


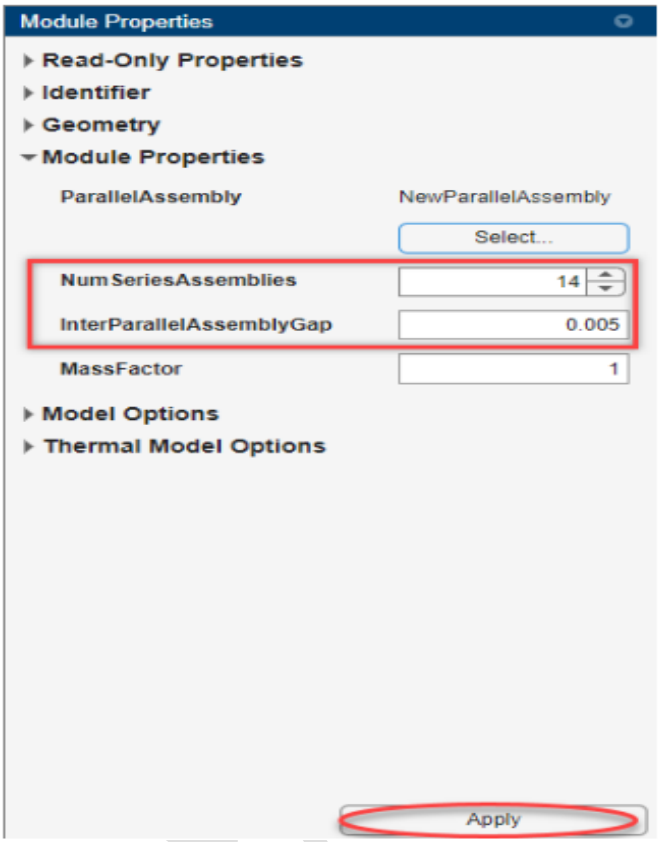


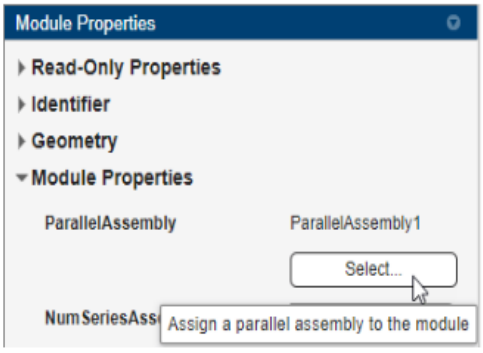
Keying in Geometrical and Electrical Specification of Parallel Assembly

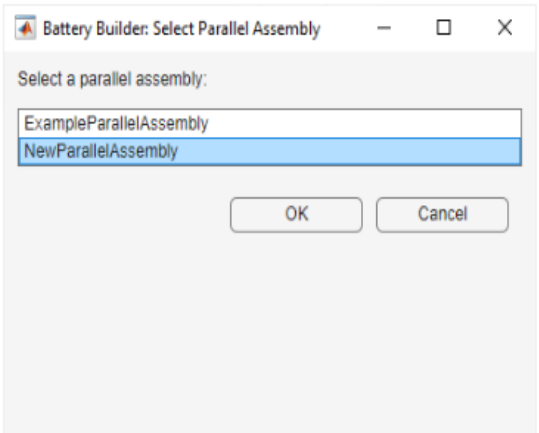


Create Module

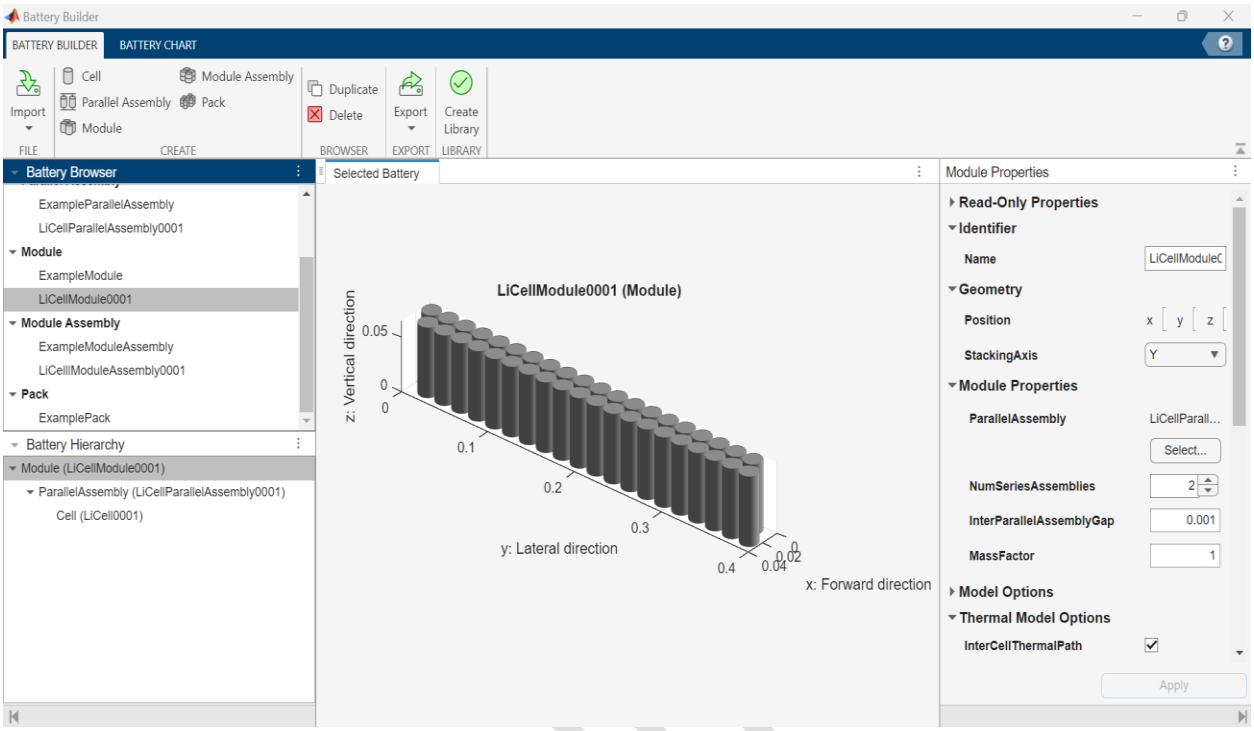




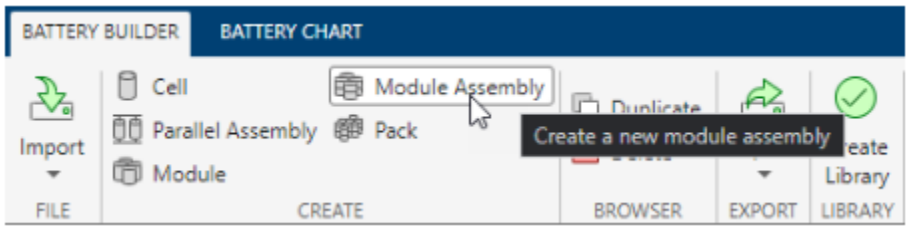


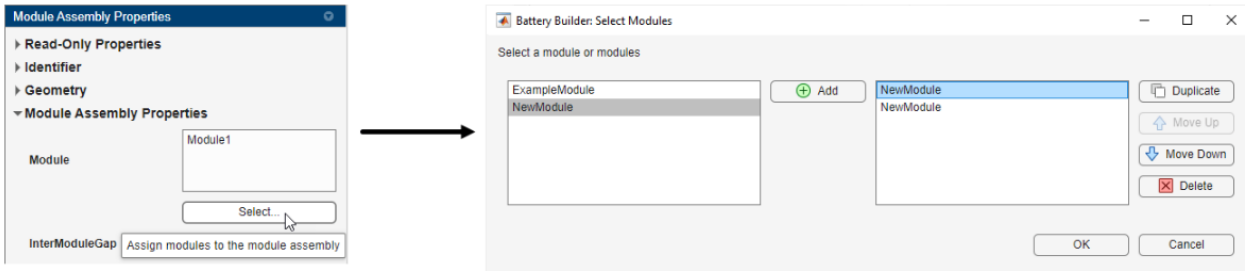


Keying in Geometrical and Electrical Specification of Module

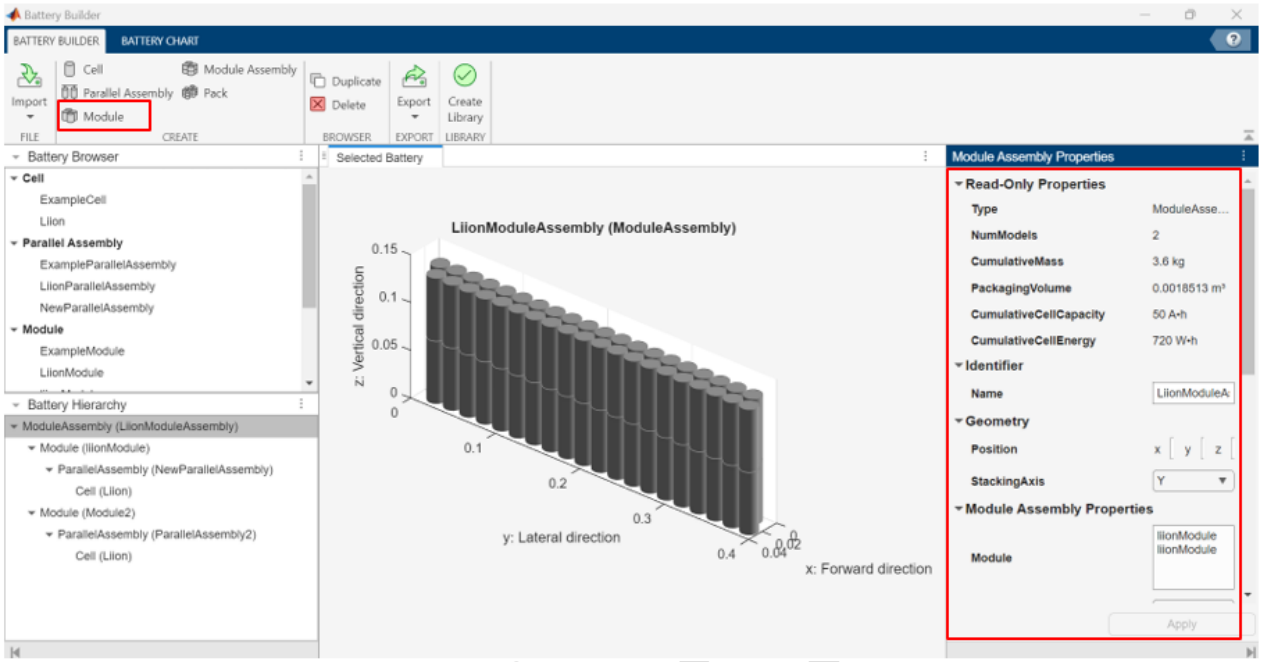


Create Module Assembly

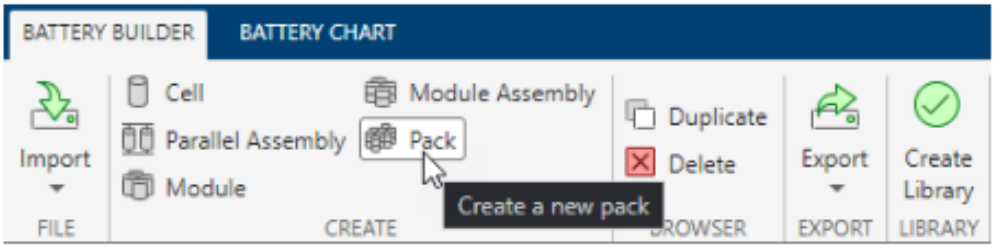


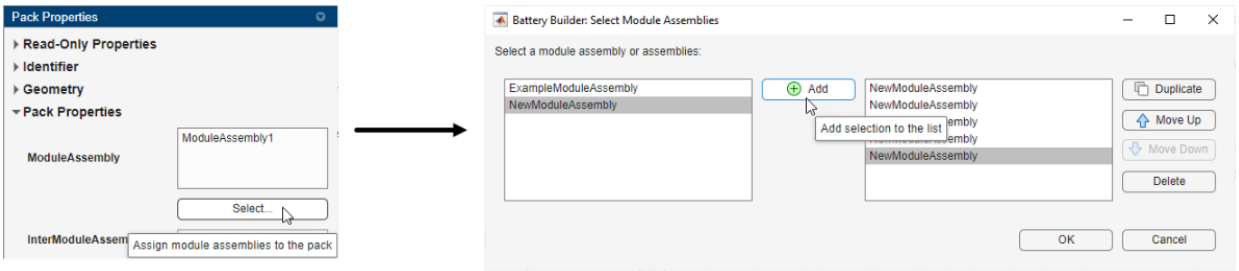


Keying in Geometrical and Electrical Specification of Module Assembly

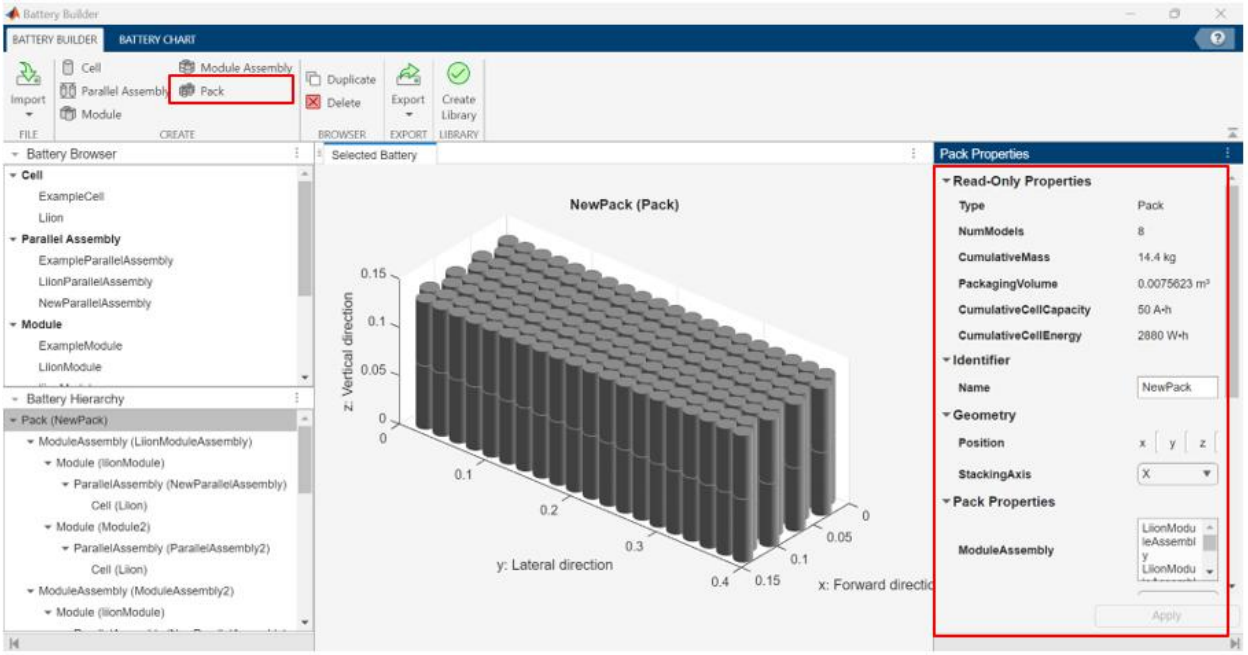


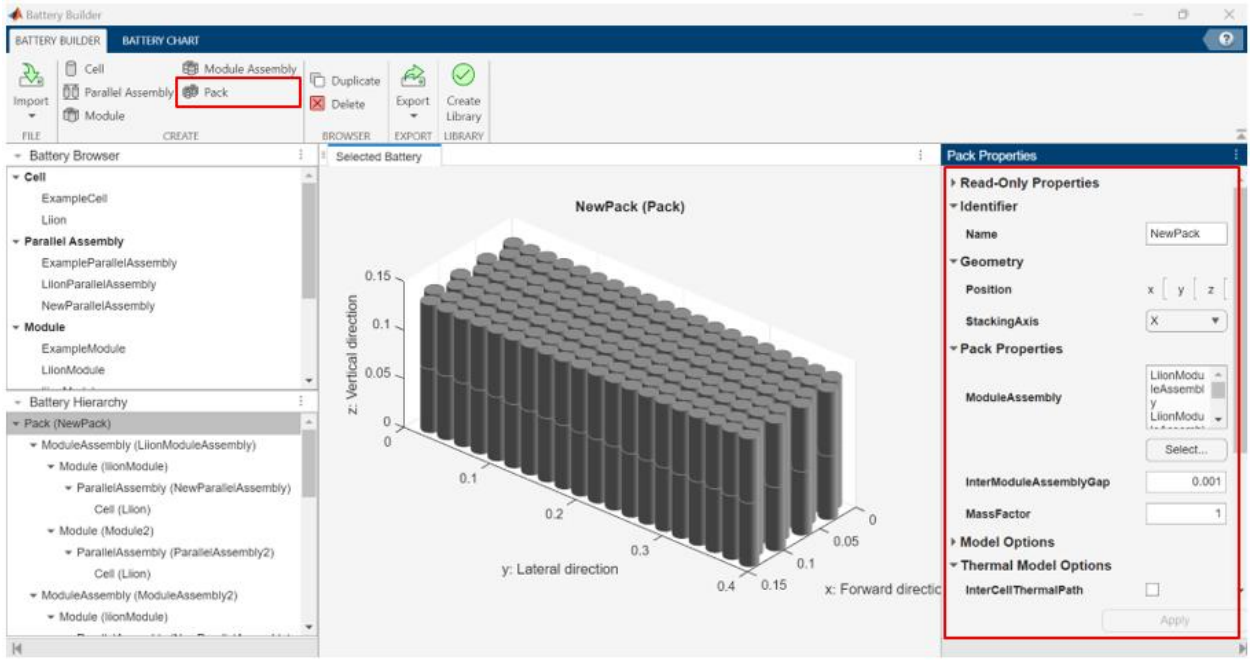
Create Battery Pack

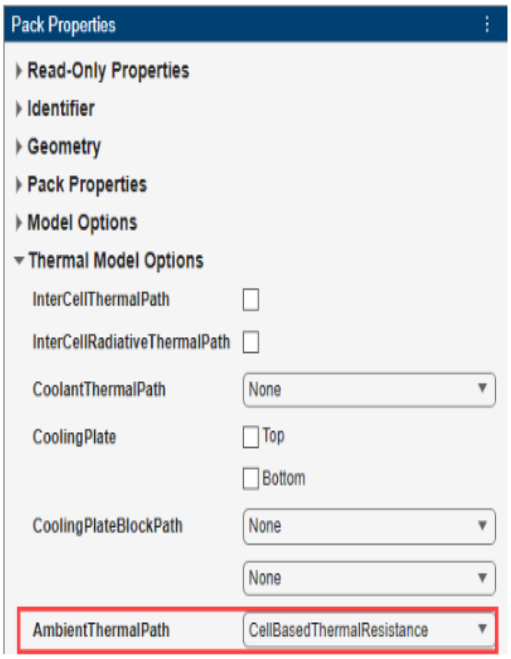


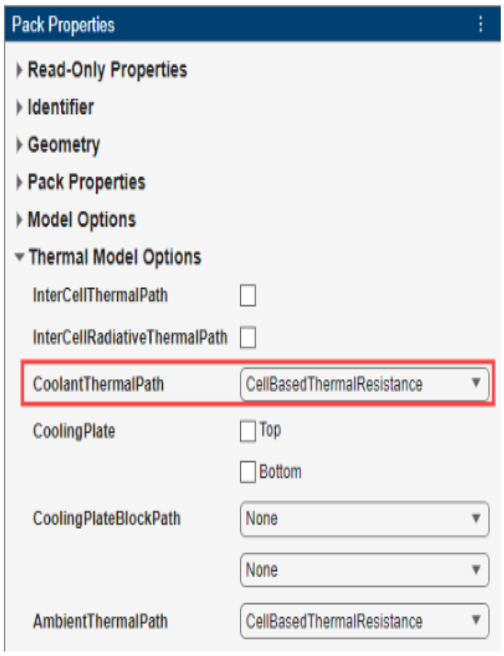


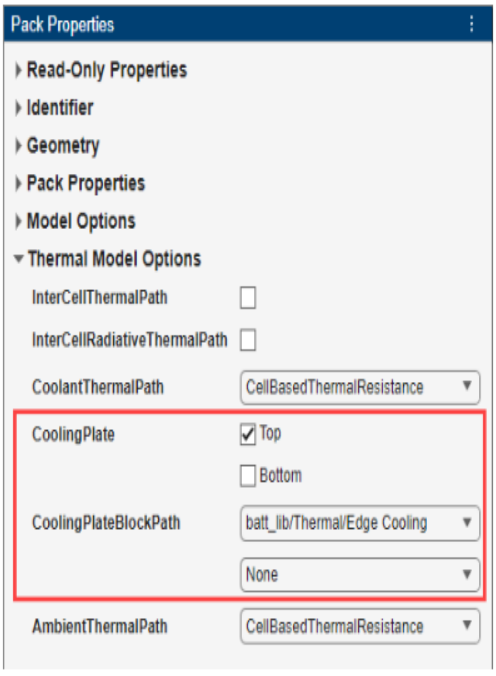
Keying in Geometrical and Electrical Specification of Battery Pack

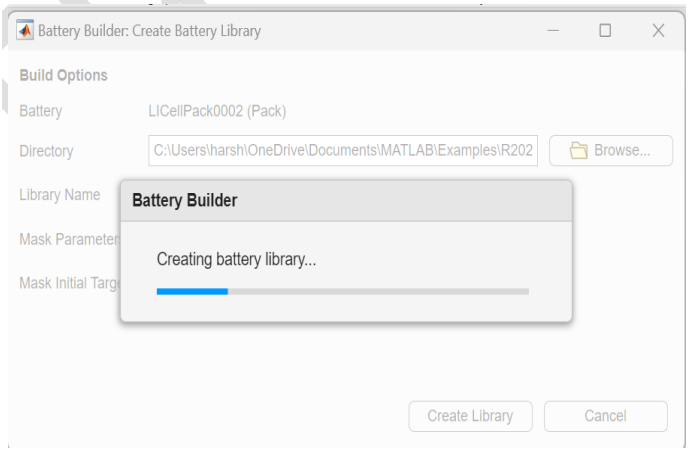




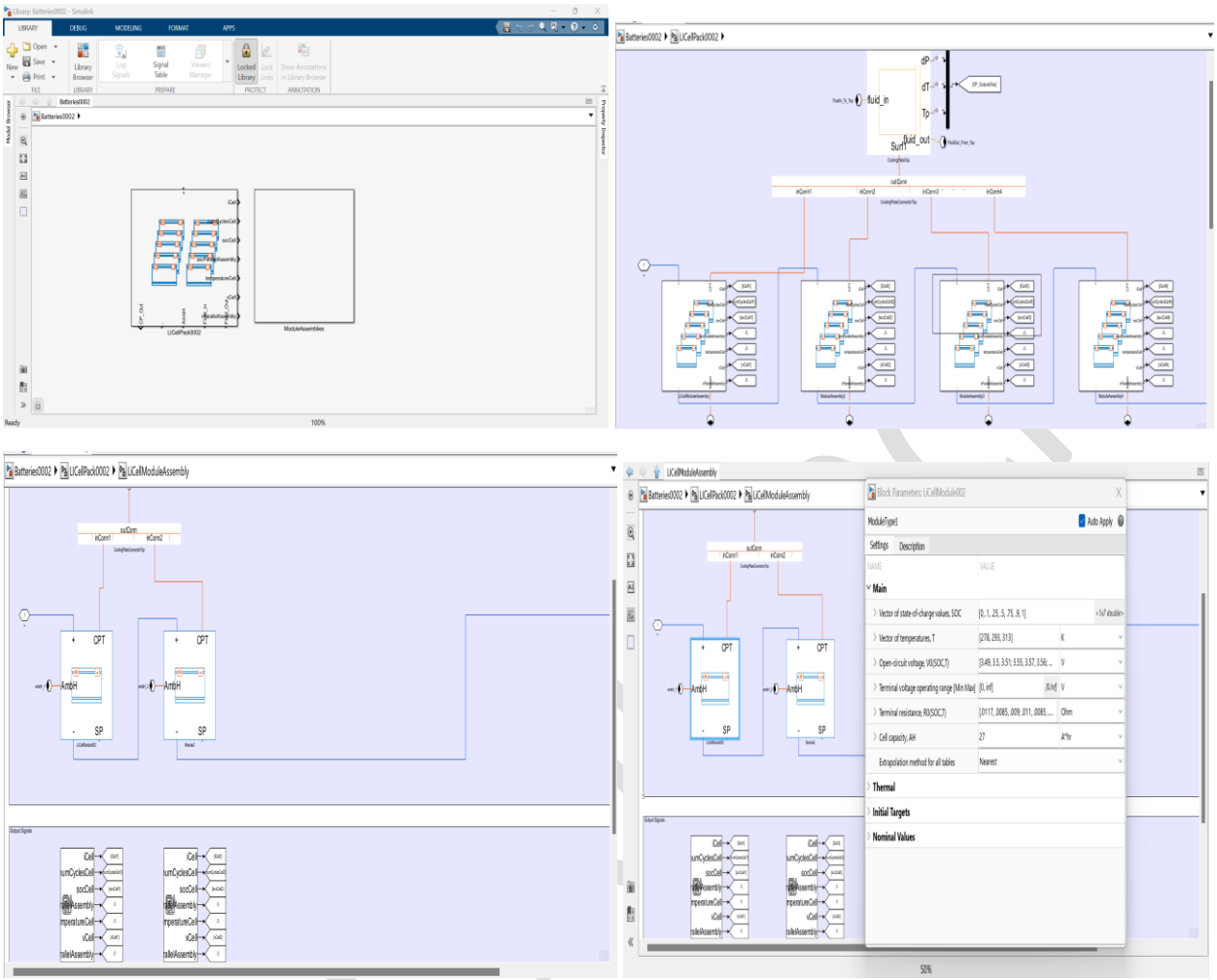








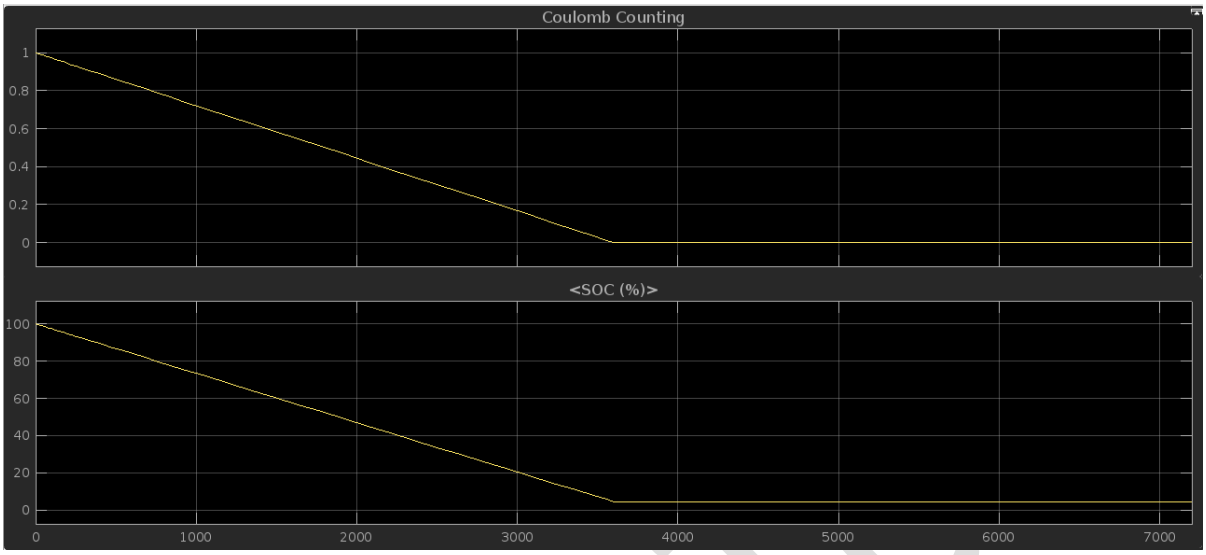
Pack and Assembly in Library



Creating Battery Pack Library in MATLAB

Once the pack has been created click on create library to have a customized pack ready to be used in the E-Bike Path for saving new pack can left default. Once it is being created, can be verified for capacity, Open circuit voltage, SOC, Thermal settings.

7.Results:



8.Conclusion:

The experiment demonstrates that the Coulomb Counting method effectively estimates the SoC of lithium-ion batteries, provided accurate initial conditions and current measurements are maintained.

Objective: For the Lithium-Ion battery pack with 24 volts, 7 Ah, interface the **Battery Management System with a CAN board and tabulate the following:**

- State of charge of the battery
- State of health of the battery
- Battery pack voltage Charging current
- Cell temperature

Specification:

- Lithium-Ion battery
- BMS and CAN bus
- Power supply

Software Required:

1. MATLAB R2020a and above

2. Windows 7/8/10/11

Features of Batbot Software

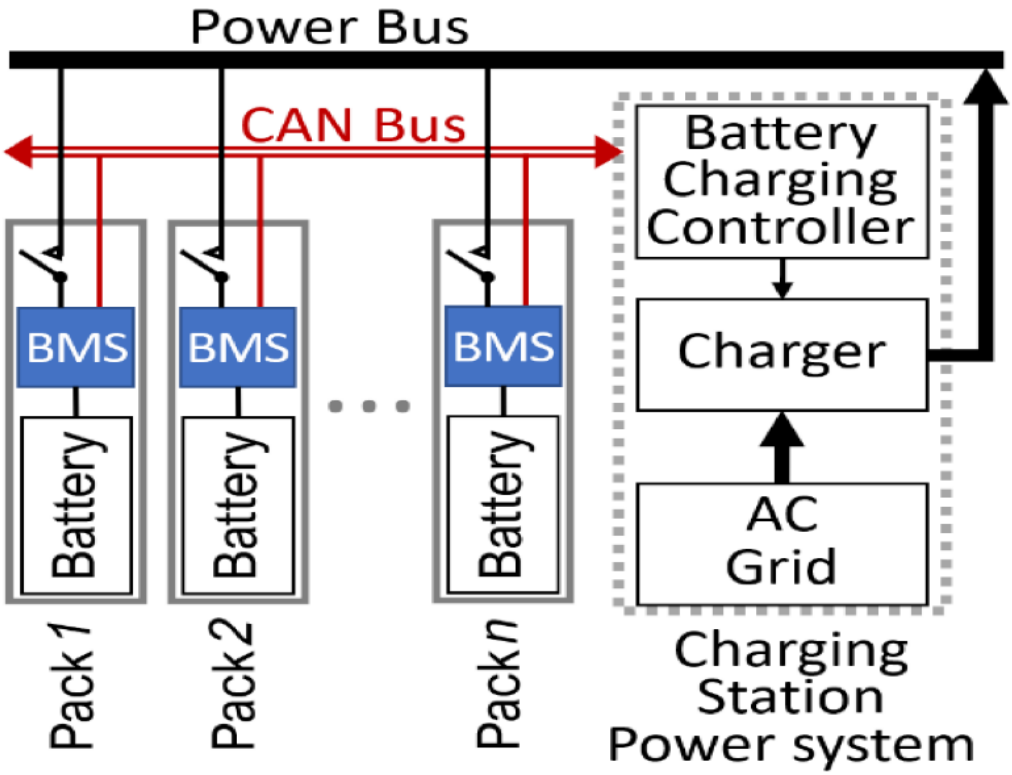
- Indicates the battery pack temperature, voltage and charging current
- Indicates Individual cell voltage (minimum and maximum)
- Monitoring State of Charge and State of Health
- Helps to analyse the battery cell balancing during charging mode / discharging mode

CAN Bus for BMS

The battery management system (BMS) commonly incorporates a CAN bus to monitor and control the battery's charging, discharging, and temperature.

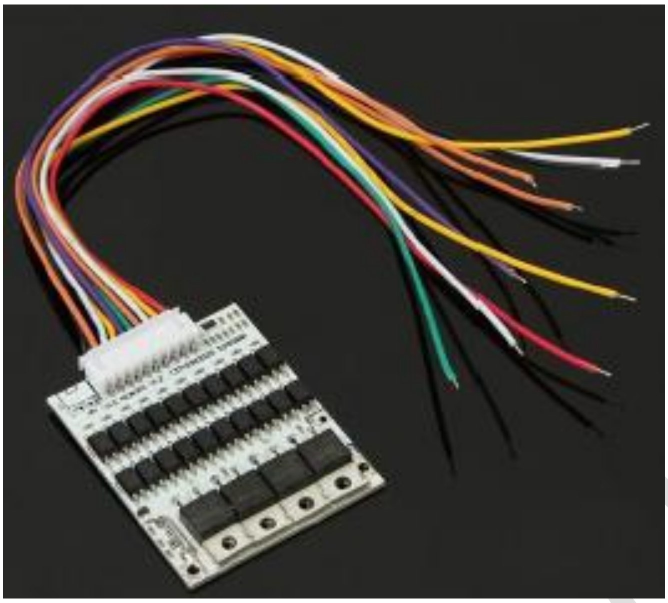
The BMS can facilitate communication among different components of the battery management system and with external devices, such as charging stations or inverters**.**

CAN bus Interface



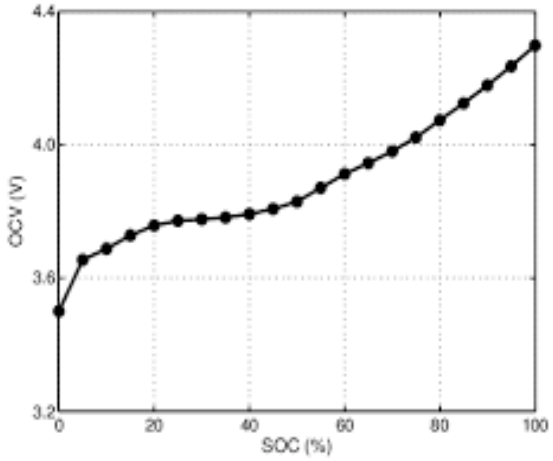
BMS Unit

A Battery Management System (BMS) controls the charging and discharging of rechargeable batteries, ensuring safe voltage levels and current rates. It helps protect the battery and extends its lifespan by preventing overcharging or excessive discharging.



Observation:

NM - Electric & Hybrid Vehicles							
Observation - Battery Management System (Task 10)							
Name							
Reg. No.							
College Name							
Sl. No.	State of Health (%)	State of Charge (%)	Pack Voltage (mV)	Charging current	Cell temperature		
					Zone 1	Zone 2	Zone 3



Result:

Design a powertrain system for an electric vehicle (EV) 2 focusing on key components such as the electric motor, battery, inverter, and transmission

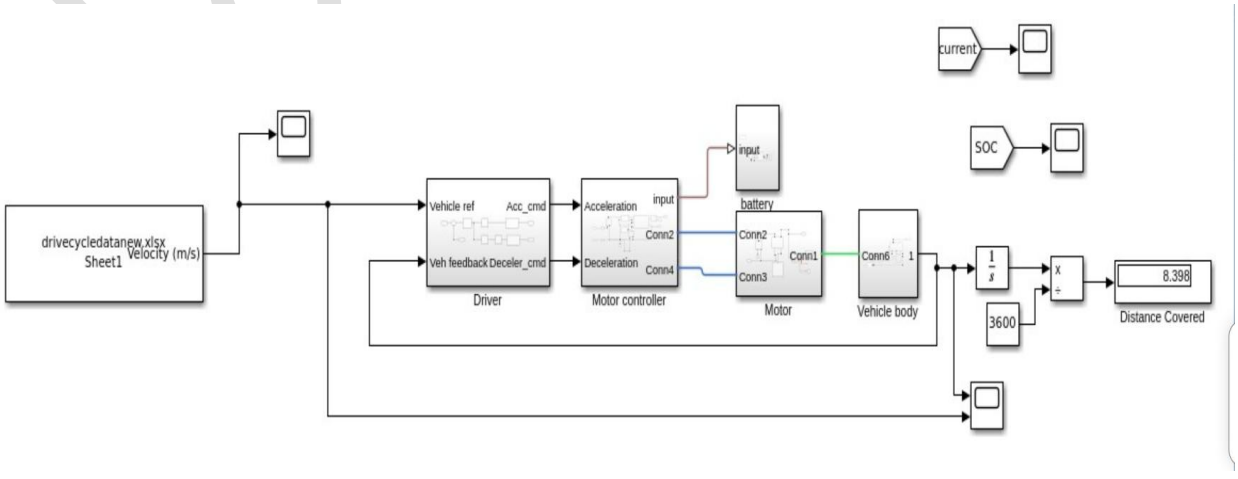
Objective:

To design and analyze the key components of a 2-wheel electric vehicle powertrain, including the electric motor, battery pack, inverter, and transmission system, and evaluate their performance characteristics.

Main Components

- Battery Pack
- Electric Motor
- Inverter
- Charging Port
- Regenerative Braking System
- Power Electronics Controller
- Thermal Management System
- Onboard Charger
- Vehicle Control Unit (VCU)
- Chassis and Body
- User Interface System

MATLAB Simulink Model



Drive Cycle Data

Drive cycle data refers to standardized patterns of vehicle operation used to evaluate fuel consumption, emissions, and performance of vehicles under various conditions. This data is essential for regulatory compliance, testing, and improving vehicle design.

Parameters Measured

a) Speed

b) Acceleration and Deceleration

c) Time

d) Distance

Types of Drive Cycles:

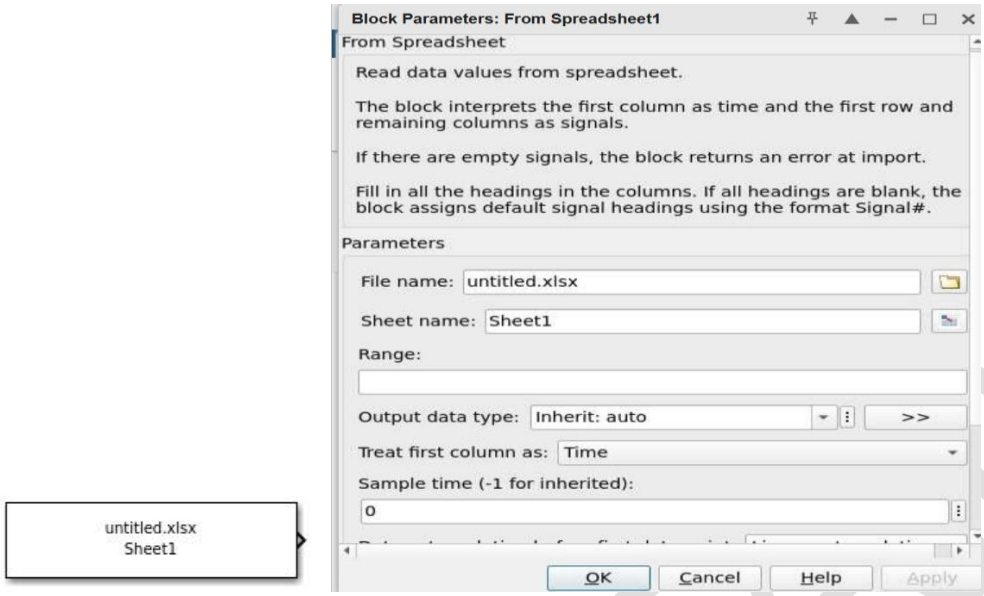
1. **Urban Drive Cycle:** Represents city driving conditions, characterized by frequent stops, starts, and lower speeds.
2. **Highway Drive Cycle**: Simulates driving on highways with higher speeds and fewer stops.
3. **Combined Cycle**: A mix of urban and highway driving, reflecting average use.

Drive Cycle Data in MATLAB - Block Name: “From Spreadsheet”

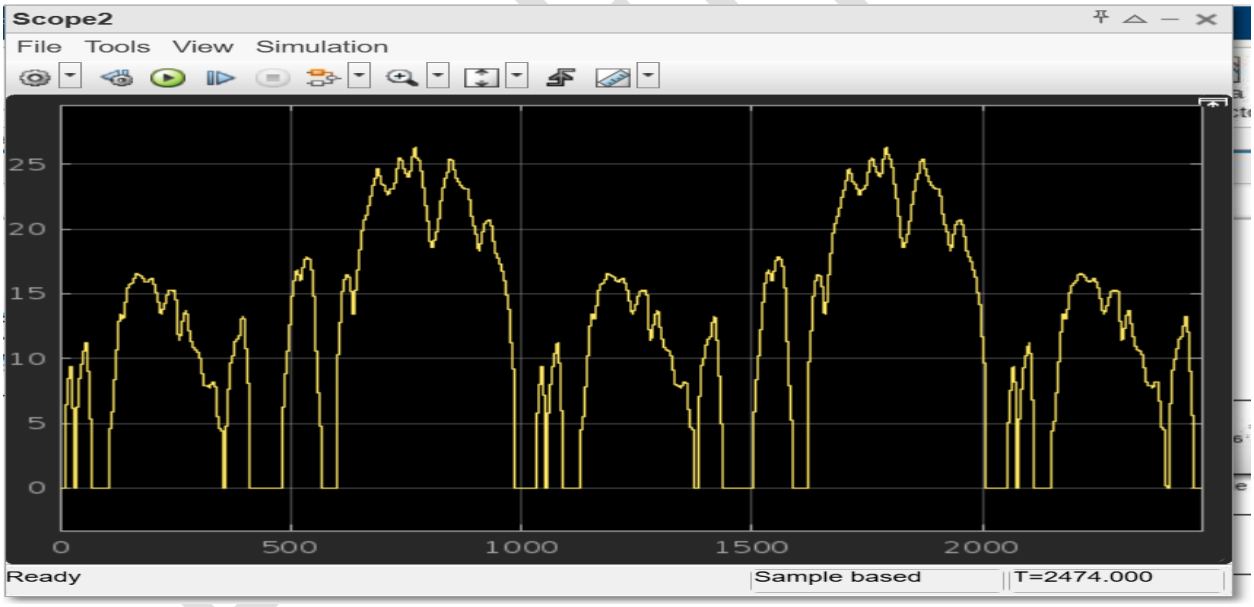
The FTP-75 drive cycle data is used in MATLAB. The FTP-75 drive cycle is a standard test procedure that measures a vehicle's emissions over a specific distance and time.

The FTP-75 cycle has the following parameters:

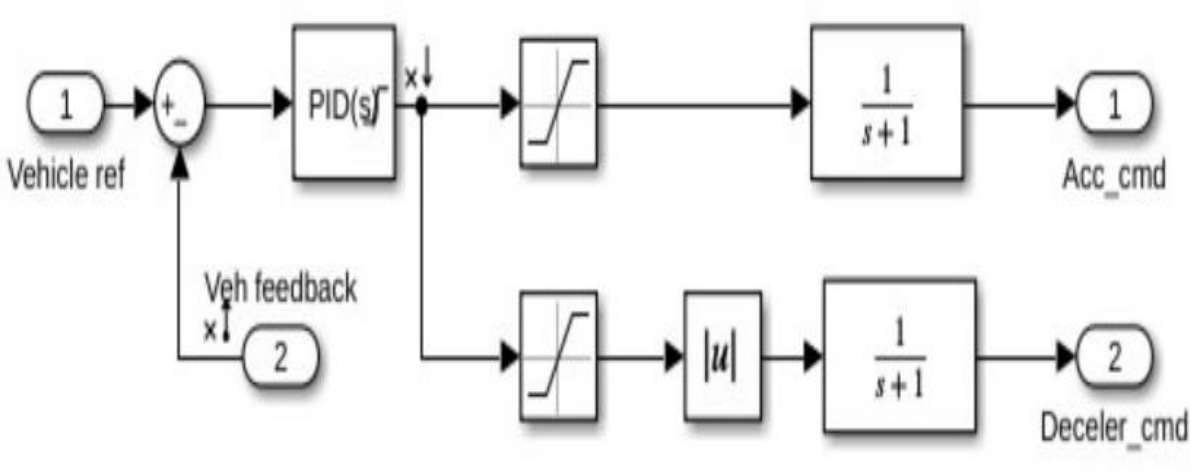
- Distance: 11.04 miles (17.77 km)
- Duration: 1874 seconds
- Average speed: 21.2 miles/hr (34.1 km/hr)
- Max. speed: 56.7 miles/hr (91.25 km/hr)



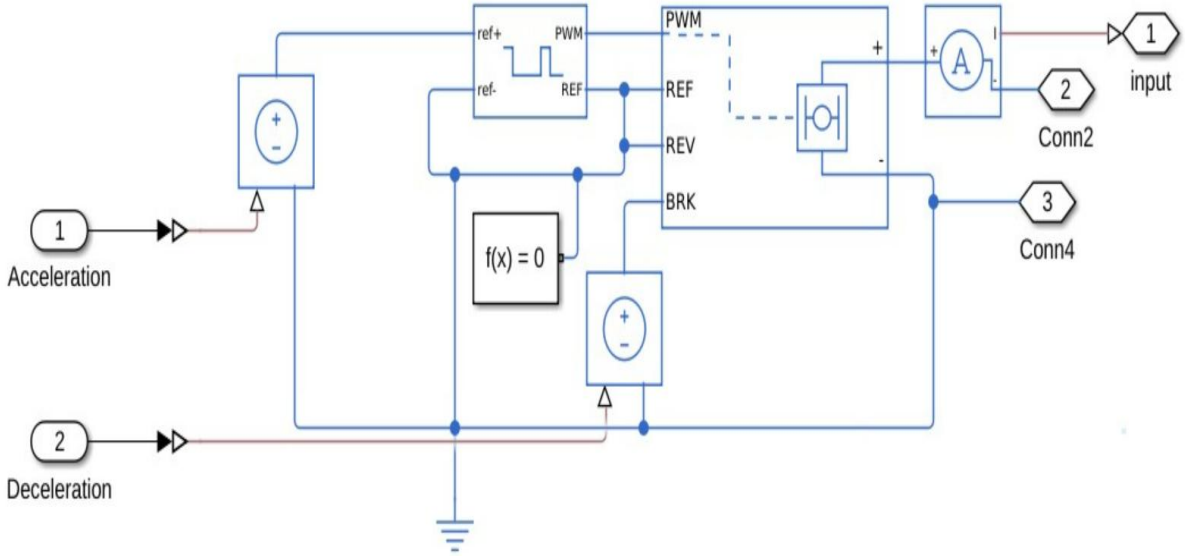
Drive Cycle Data Plot (Repeated Sequence)



Subsystem – Motor Driver Circuit



Subsystem – Motor Controller



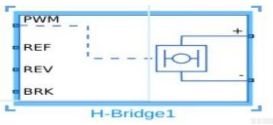
Blocks – Controlled PWM Voltage and H-Bridge

Block Parameters: Controlled PWM Voltage

Controlled PWM Voltage — Auto Apply

Settings | Description

NAME	VALUE	
Modeling option	Electrical input ports	
PWM		
PWM frequency	3000	Hz
Simulation mode	Averaged	
Input Scaling		
Input voltage for 0% duty cy...	0	V
Input voltage for 100% duty ...	1	V
Output Voltage		
Output voltage amplitude	5	V



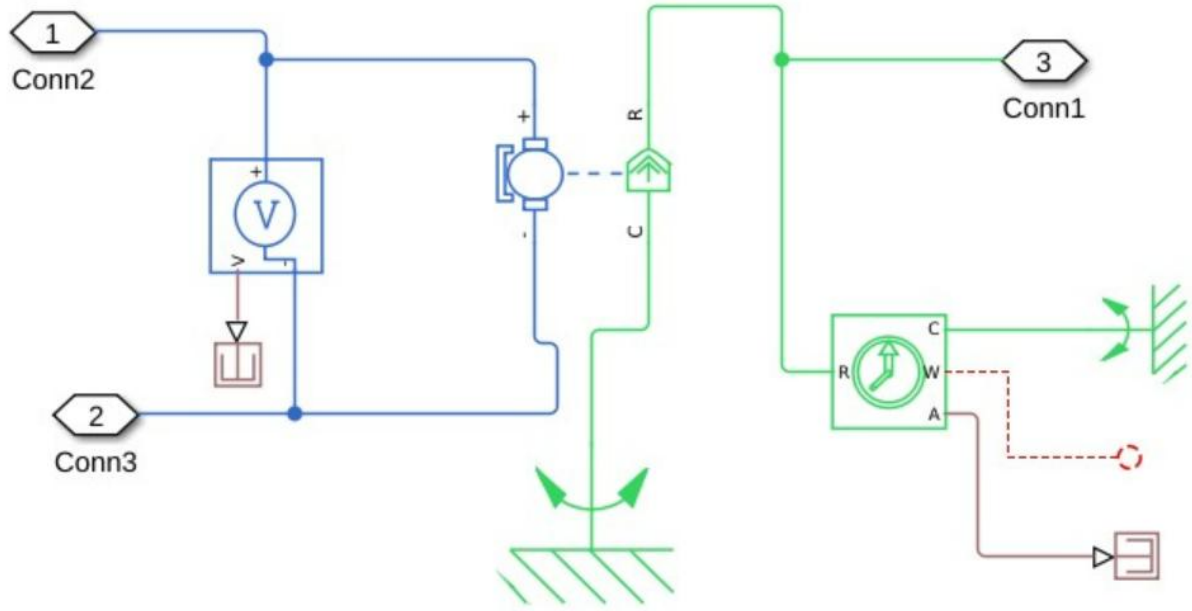
Block Parameters: H-Bridge

H-Bridge — Auto Apply

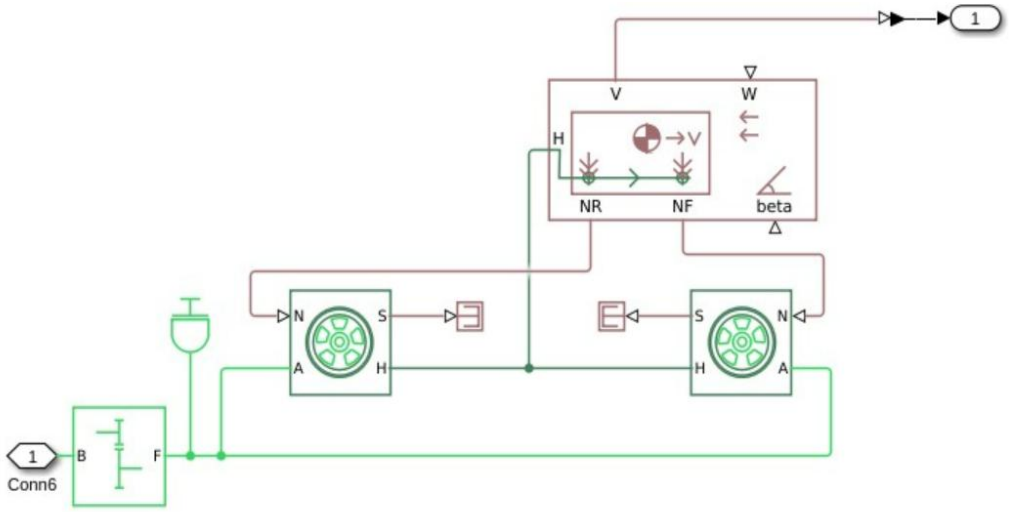
Settings | Description

NAME	VALUE	
Modeling option	No thermal port	
Simulation Mode & Load Assumptions		
Power supply	Internal	
Simulation mode	Averaged	
Regenerative braking	Always enabled (suitable for linearization)	
Load current characteristics	Smoothed	
Input Thresholds		
Enable threshold voltage	2.5	V
PWM signal amplitude	5.0	V
Reverse threshold voltage	2.5	V
Braking threshold voltage	2.5	V
Bridge Parameters		
Output voltage amplitude	48	V
Total bridge on resistance	0.1	Ohm
Freewheeling diode on resist...	0.05	Ohm

Subsystem – Motor (DC Motor)



Subsystem Vehicle Body



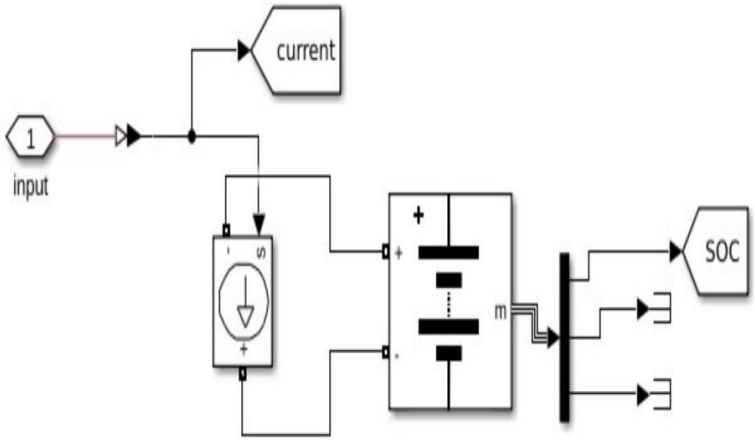
Block Parameters: Tire (Magic Formula)1

Tire (Magic Formula) ☑ Auto Apply

Settings | Description

NAME	VALUE	
Main		
Parameterize by	Peak longitudinal force and corresponding s	
Rated vertical load	3000	N
Peak longitudinal force at rat...	3500	N
Slip at peak force at rated lo...	10	
Geometry		
Rolling radius	0.255	m
Rolling Resistance		
☑ Model rolling resistance		
Resistance model	Constant coefficient	
Constant coefficient	0.015	
Velocity threshold for rolling ...	0.001	m/s
Dynamics		
Advanced		

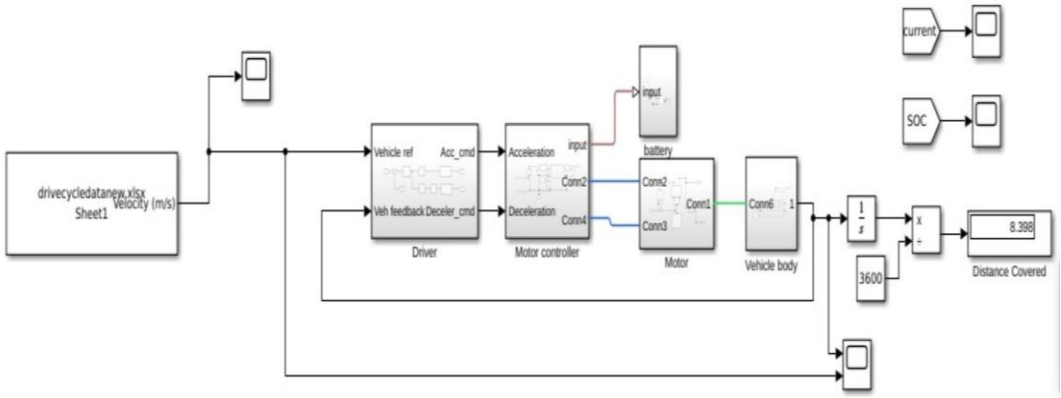
Subsystem – Battery Input



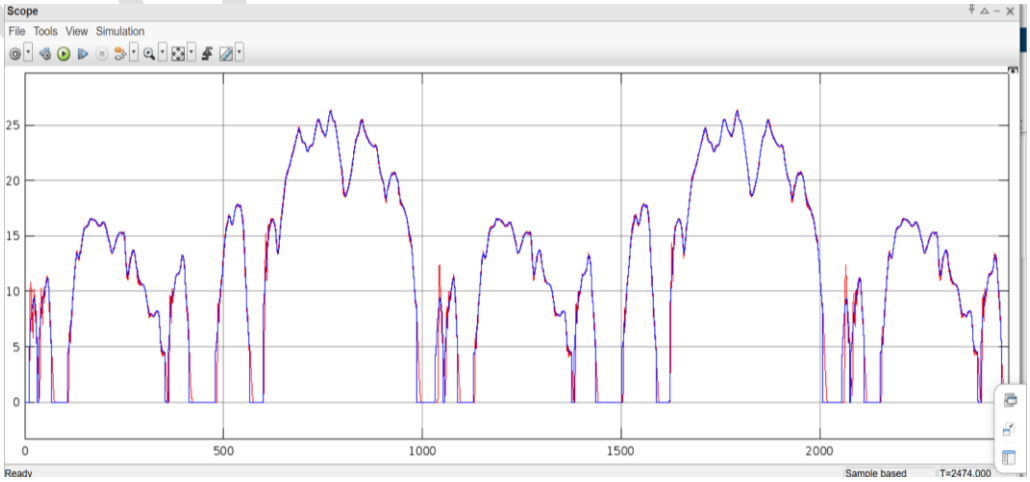
Simulation Results

Total Simulation Time (based on drive cycle time data) = 2474 sec

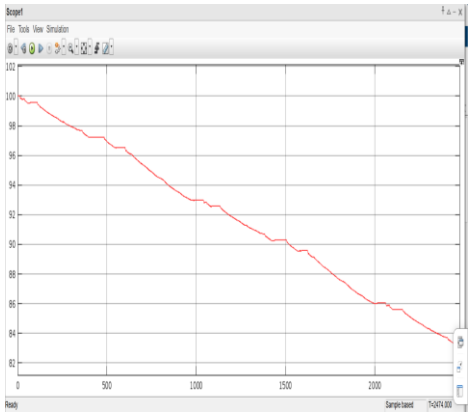
Range / Distance Travelled = 8.4 km

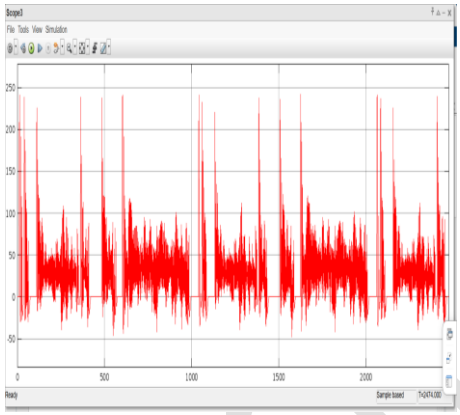


Simulation Results: Measured Speed Vs Desired Speed



Simulation Results - State of Charge and Battery Current





State of Charge estimation of lithium-ion battery using Coulomb Counting method

1. Introduction:

- **Objective**: Briefly explain the purpose of the Coulomb Counting method and its application in estimating the state of charge (SOC) of a lithium-ion battery.
- **Tools Used**: Simulink is used to model the battery and the Coulomb Counting method.

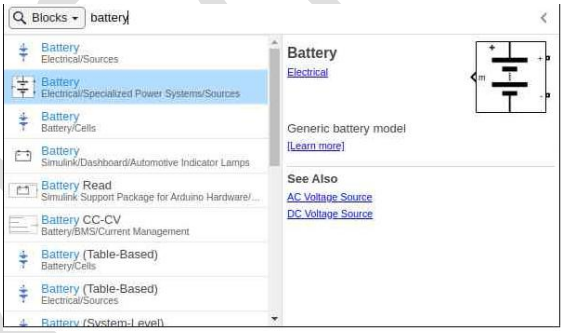
2. Model setup:

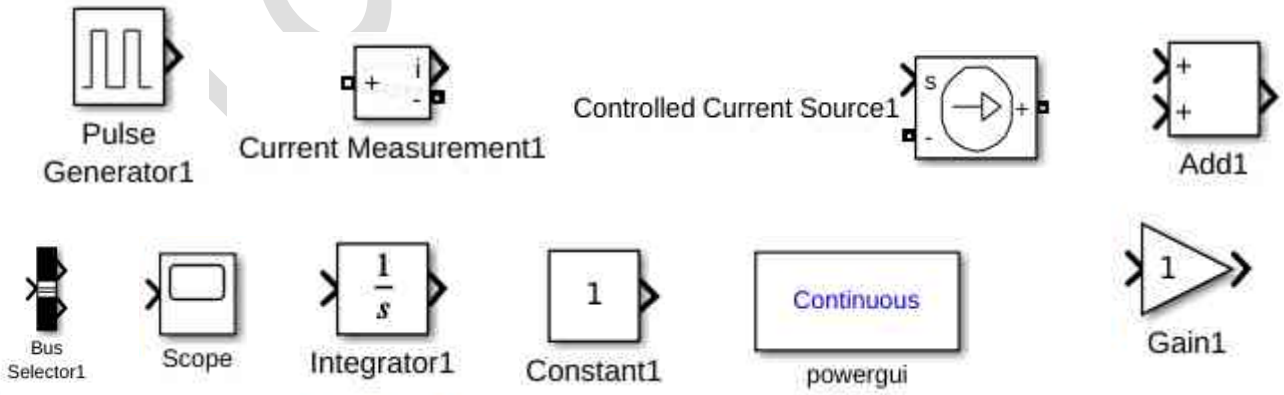
Open Simulink

- **Step 1**: Launch MATLAB and type Simulink in the command window to open Simulink.
- **Step 2**: Create a new model by selecting **File → New → Model**

3. Component selection:

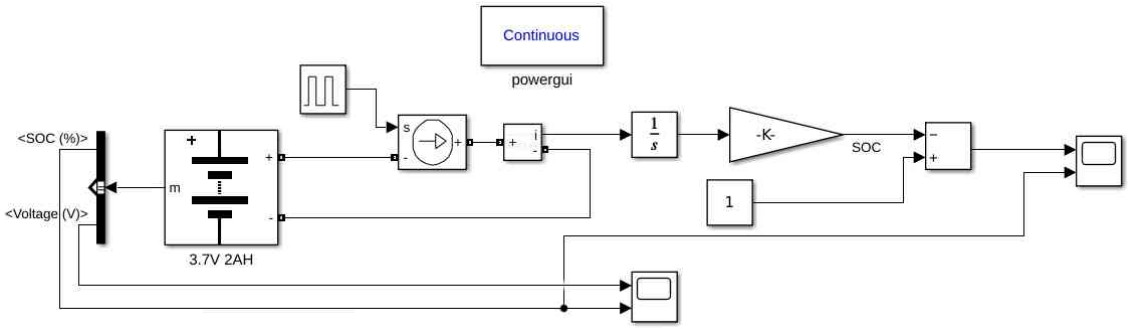
Add the required component blocks by double-clicking on the Simulink canvas or by navigating through the Library Browser to select the necessary blocks





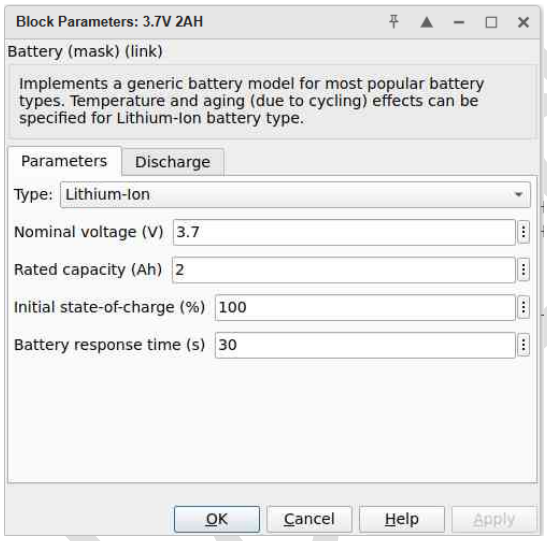
4. Connecting the blocks:

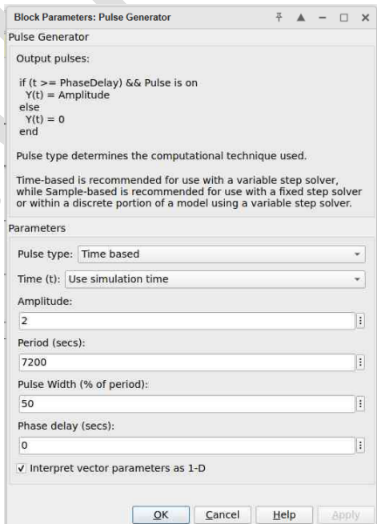
Connect all the components as per the following circuit

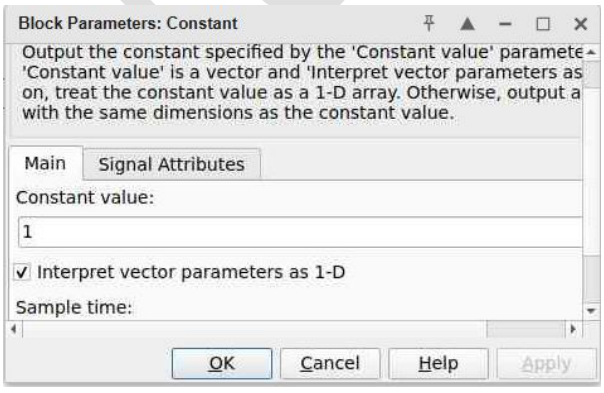


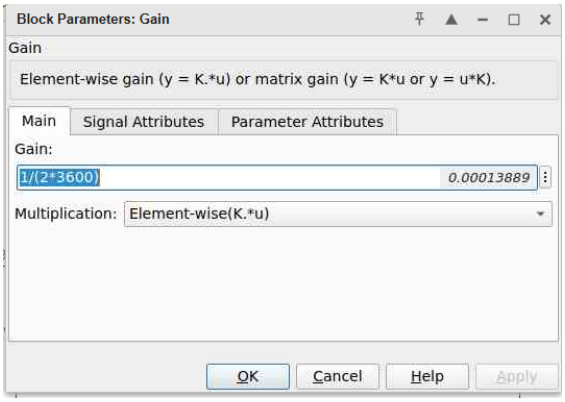
5. Parameter setting:

Double click on each block and change the required parameters







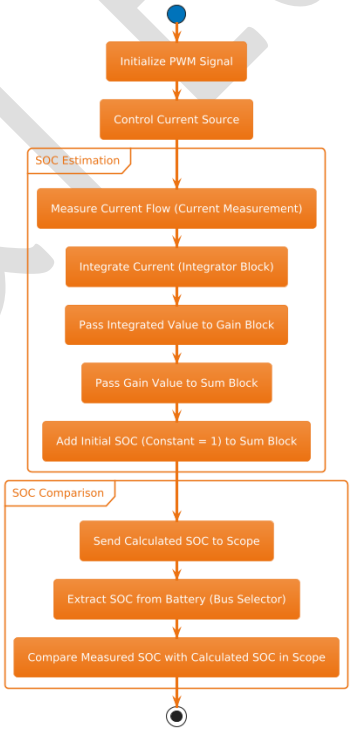


6. orking principle and control logic

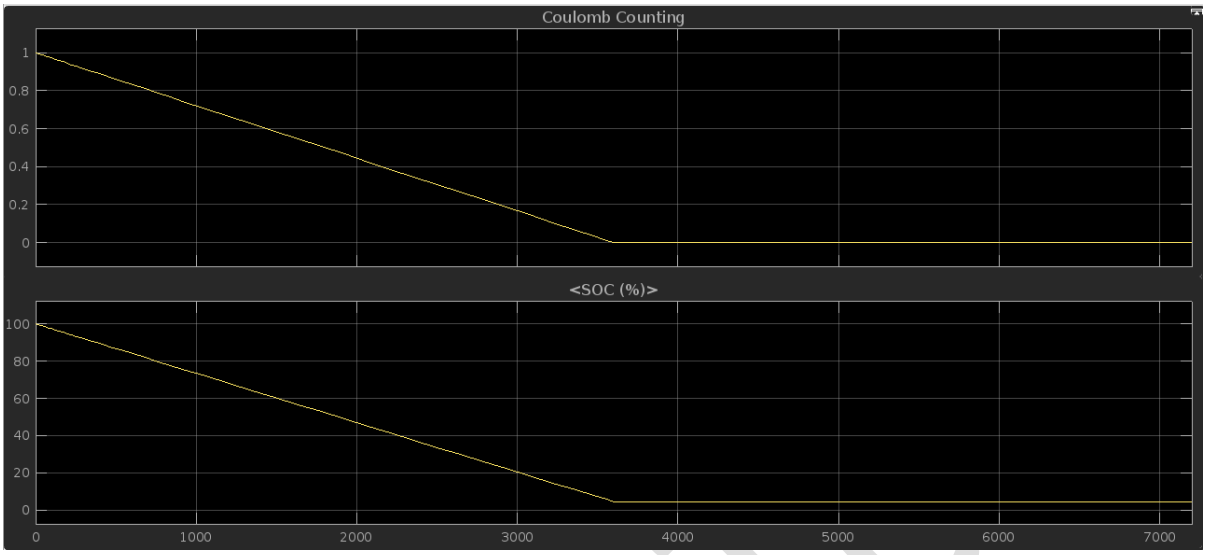
The **State of Charge (SOC)** of a battery represents the remaining charge relative to its capacity. The **Coulomb Counting Method** estimates the SOC by integrating the current flow over time, tracking the battery's charge consumption or gain.

1. **PWM Signal Initialization**: A PWM signal is generated to control the current source, regulating the battery's charging or discharging.
2. **Current Measurement**: The current flowing through the battery is measured, providing real-time data on the battery's charge and discharge rate.
3. **Current Integration**: The measured current is integrated over time using an **Integrator Block**, calculating the total charge transferred in or out of the battery.
4. **Gain Block**: The integrated current is passed through a **Gain Block**, scaling the value to correspond to the battery's SOC.
5. **Initial SOC Addition**: The SOC starts at an initial value (typically 100%) and is updated by adding the calculated charge from the integrator, reflecting the change in SOC.
6. **SOC Display**: The updated SOC is visualized in a scope, showing how the SOC evolves during discharge or charge cycles.
7. **SOC Comparison**: The calculated SOC is compared with the actual SOC from the battery to verify the accuracy of the estimation.

This method provides continuous, real-time monitoring of the battery's SOC, essential for battery management and optimization in various applications, such as electric vehicles and energy storage systems.



7.Results:



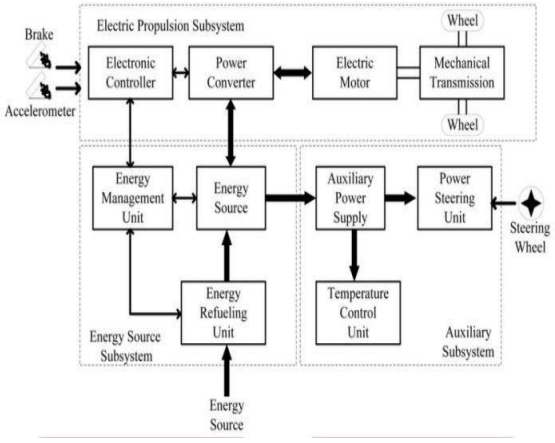
8.Conclusion:

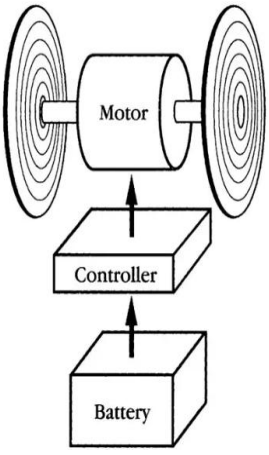
The experiment demonstrates that the Coulomb Counting method effectively estimates the SoC of lithium-ion batteries, provided accurate initial conditions and current measurements are maintained.

Objective: Assemble an electric bike using the components BLDC hub motor with hall sensor – 24 volts, 250 watts, Lithium-Ion battery pack – 7Ah, 24 volts, BMS), BLDC motor controller, Throttle, Brake levers (left and right), LCD display, Headlight, Pedal assist sensor, Horn CAN Interface Charger

Specification:

1. BLDC hub motor with hall sensor – 24 volt, 250 watts
2. Lithium-Ion battery pack – 7Ah, 24 volt
3. Battery Management System
4. BLDC motor controller
5. Throttle
6. Brake lever – Left & Right
7. LCD Display
8. Headlight
9. Pedal assist sensor
10. Horn
11. CAN interface cord
12. Charger
13. **Full EV Model for a Two-Wheeler (Block Diagram)**





Procedure:

Here's step-by-step assembly process for 4-motor electric bike

1. Install the BLDC Hub Motor

- Mount the hub motor in the front or rear wheel.
- Secure it with axle nuts and ensure the motor wires can reach the controller.

2. Mount the Lithium-Ion Battery

- Secure the battery to the frame (typically on the downtube or rear rack).
- Connect the battery to the motor controller.

3. Wire the Motor Controller

- Connect the motor's power wires (3-phase) and Hall sensor wires to the controller.
- Attach the battery wires to the controller's power input.

4. Install the Throttle

- Position the throttle on the right handlebar and secure it.
- Wire the throttle to the motor controller's throttle input.

5. Install Brake Levers

- Mount brake levers on the handlebars.
- Connect the brake lever switches to the controller for cut-off functionality.

6. Install the Pedal Assist Sensor (PAS)

- Mount the PAS sensor on the bottom bracket and secure it.
- Connect the PAS wire to the controller.

7. Install the LCD Display

- Attach the LCD display to the handlebars.
- Connect the display to the controller.

8. Install the Headlight and Horn

- Mount the headlight and horn on the bike.
- Connect them to the controller for power.

9. Wire Everything

- Ensure all components (throttle, brake levers, PAS, LCD, motor, etc.) are securely wired to the controller.
- Use zip ties to organize the wires.

10. Charge the Battery

- Plug in the charger and fully charge the battery before testing.

11. Test the Bike

- Power on the bike, check throttle, PAS, brakes, and headlight.
- Take a test ride to ensure everything works properly.

12. Final Checks

- Ensure all components are securely mounted and tightened.
- Perform any fine-tuning on the display settings if needed.

13.Enjoy Your Electric Bike

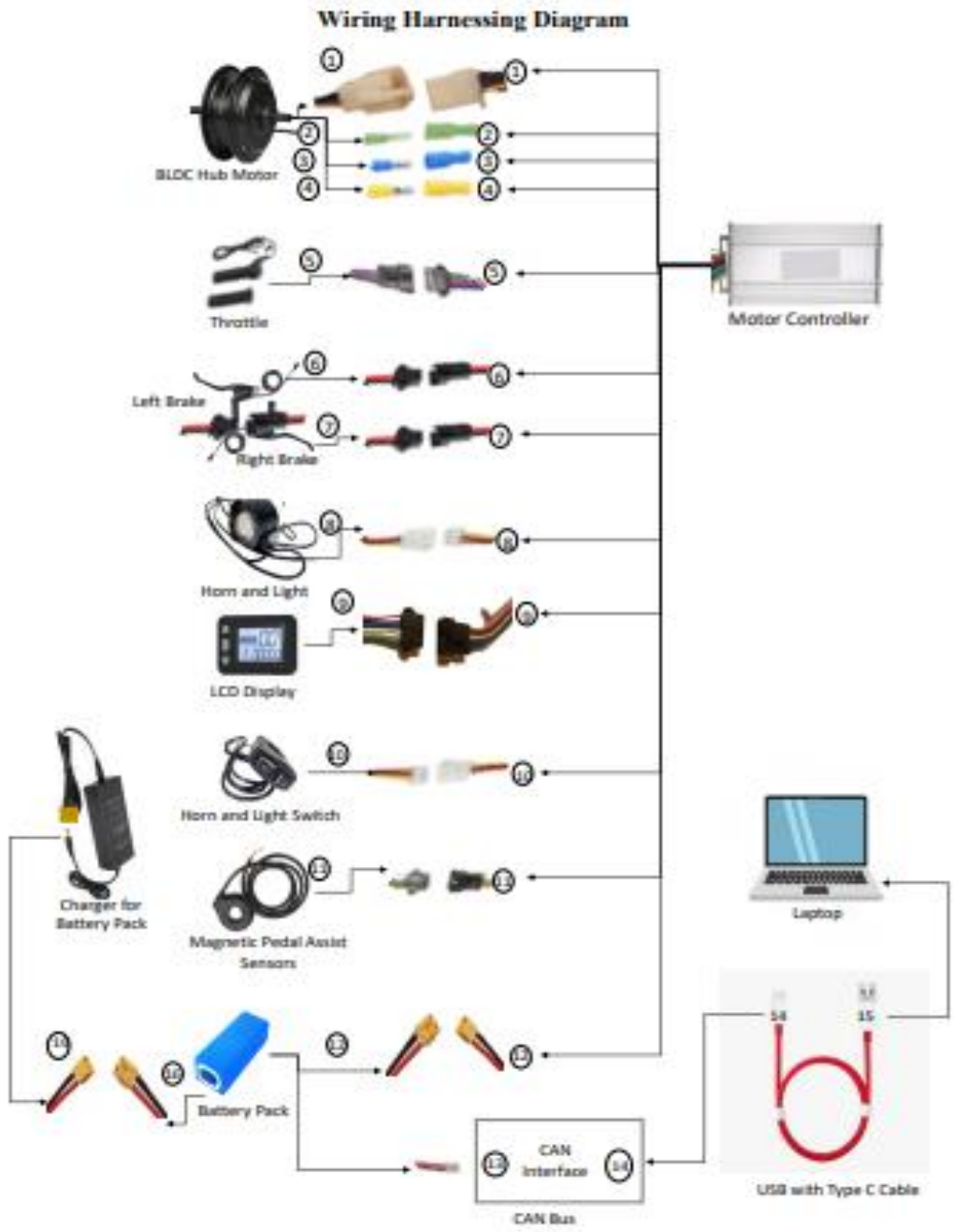
- Once everything is set up, your electric bike is ready to go!

Safety Notes:

- Ensure the battery is installed securely and that it is not in danger of coming loose while riding.
- Double-check all wiring for any exposed connections or short circuits before testing.
- Make sure your bike's frame and all components are rated to handle the additional weight and power of the electric components.

Connecting the blocks:

Connect all the components as shown in the circuit diagram below, ensuring that each block is linked according to the specified configuration.



Result:

This assembly process should be provided with a fully functional electric bike using your specified components