CAN-Based Digital Twin of Electric Vehicle ECU System

Platform: MATLAB/Simulink
Subsystems: ECU Stateflow Controller, PID Speed Control, CAN Communication, Vehicle Dynamics
Method: Stateflow Logic and CAN-Based System Integration
Writer: Mehmet Karagülle

Contents

.

1. Introduction and Objectives

This project extends the previously developed Electric Vehicle Powertrain Model by introducing a CAN-based digital twin of the Electronic Control Unit (ECU)**.**
The aim is to simulate a real-time communication and control environment where the ECU manages vehicle modes and torque commands through the CAN protocol.

Main objectives:

- Develop a Stateflow-based ECU control structure with operational modes (Idle, Accel, Cruise, Decel).
- Integrate a PID controller to generate torque demand (Torque_Req) through CAN messages.
- Establish CAN Pack/Unpack communication between control and physical layers.
- Analyze the effect of communication scaling and parameter tuning on system stability.

2. System Architecture

The overall model consists of two main layers:

Layer	Description
Physical Layer	Simulates the EV Powertrain including dynamics and torque response.
Control Layer (ECU)	Stateflow manages driving modes and PID-based throttle control. Communication is with CAN messages.

The electric vehicle ECU system integrated with EV powertrain system is given below(Fig. 1):

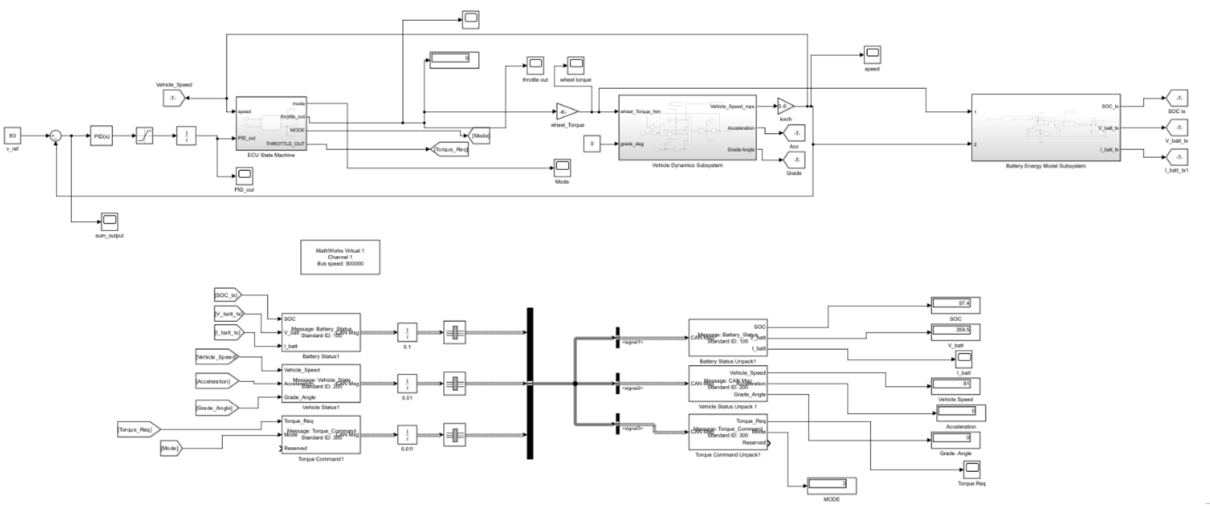


Figure 1: The electric vehicle ECU system with EV powertrain

3. ECU Stateflow Design (System Core)

The core of this project is the ECU State Machine (Stateflow) architecture. The ECU determines the vehicle's driving mode and generates throttle commands (Throttle_out) based on speed feedback, PID control and CAN communication.

3.1 Core Logic

The ECU is implemented as a finite state machine (FSM) with four operational modes, representing real vehicle behavior(Table 1):

State	Description	Output (Throttle_out)	CAN Signal
Idle	Vehicle is stationary, throttle disabled	0	Mode = 1
Acceleration	PID controller active, vehicle accelerates	PID_out	Mode = 2
Cruise	Speed maintained near target value	PID_out (steady)	Mode = 3
Deceleration	Throttle decreases smoothly	Decreasing throttle	Mode = 4

Table 1: Operational modes in Statefow

The state transitions emulate a driver's natural input behavior under throttle control, mapped directly into ECU logic.

3.2 Input and Outputs

The following signals and symbols are defined at the Stateflow interface to connect the ECU with the vehicle dynamics and control system(Table 2)(Table 3):

Signal	Type	Source / Destination	Description
speed	Input	Vehicle Dynamics	Actual vehicle speed (km/h)
V_ref	Input	Step / Constant block	Target reference speed
PID_out	Input	PID Controller	PID-calculated throttle value
Throttle_out	Output	Powertrain subsystem	ECU throttle command
Mode	Output	CAN Pack block	ECU operating mode

Table 2: Signals at the Stateflow

This structure allows the ECU to function independently while communicating with CAN.

Symbol	Meaning	Typical Value	Purpose
V_CRUISE	Reference (target) speed	80 km/h	Defines steady-state target
H	Hysteresis margin	0.5–1.0 km/h	Smooths state transitions and prevents chatter

Table 2: Symbols at the Stateflow

3.3 Transition Conditions (Mode Logic)

State transitions are driven by vehicle speed comparisons and PID behavior(Fig. 2):

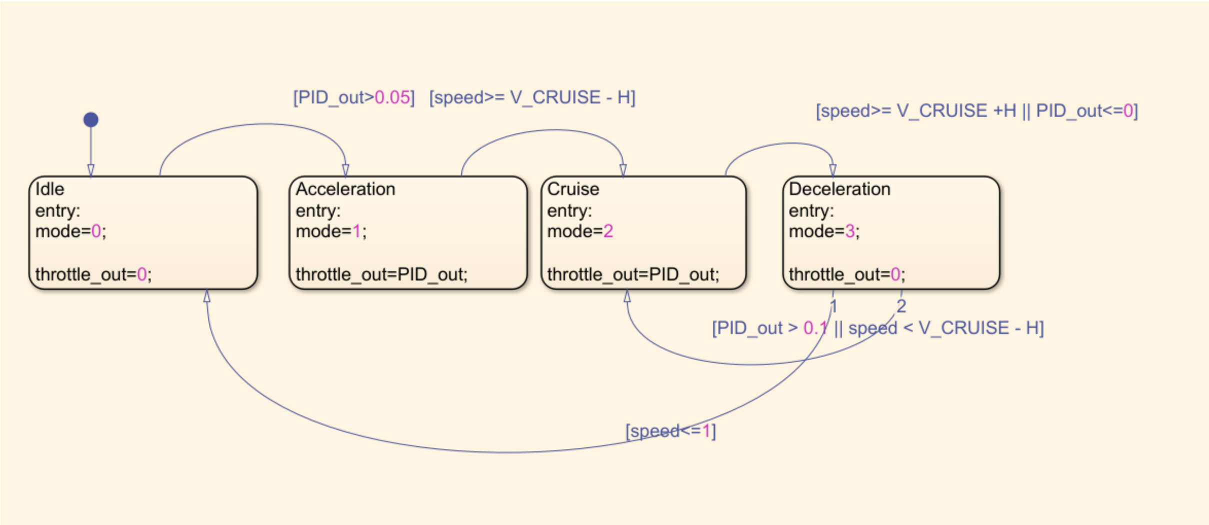


Figure 2: Stateflow chart at the ECU

3.4 PID Integration

The PID controller operates externally from the Stateflow chart to preserve timing accuracy. Its output (PID_out) is used as the ECU's throttle reference, processed based on mode logic(Fig. 3):

- Acceleration / Cruise: Throttle_out = PID_out
- Deceleration / Idle: Throttle_out = 0 or a decreasing value

This separation of control and state logic improves modularity — allowing PID tuning without altering the ECU logic itself.

The Throttle_out signal is then sent to the CAN communication block.

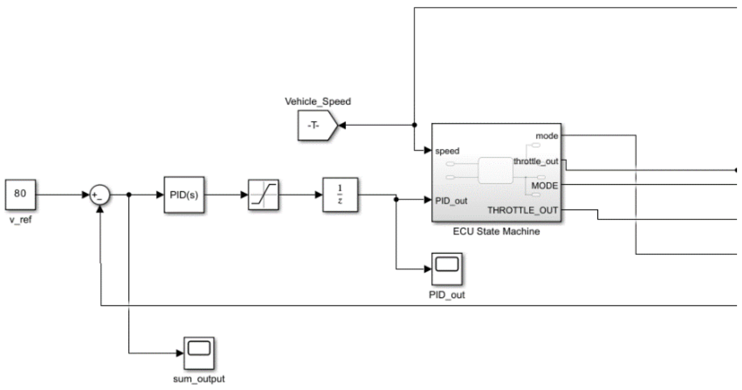


Figure 3: PID integration on ECU state machine

3.5 Delay Synchronization (Time Alignment)

During early testing, an algebraic loop occurs because the ECU computes outputs and feedback simultaneously.
To solve this, a Unit Delay (sample time = 0.01 s) or Transport Delay is added before the CAN transmission stage(Fig. 4).

This ensures:

- Deterministic (stable) simulation,
- Throttle values updated every 0.01 s,
- CAN transmission executed at 0.1 s intervals.

The result is synchronized real-time control and message timing consistency.

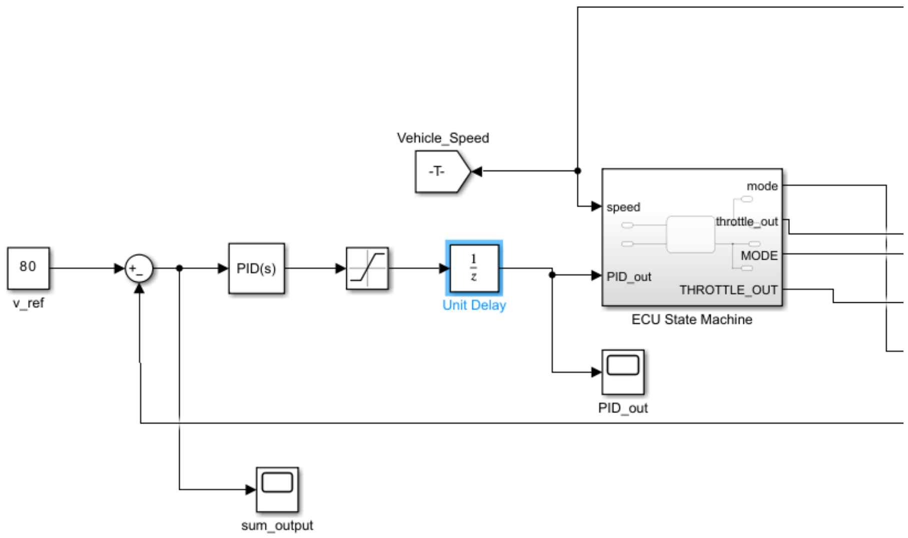


Figure 4: Unit delay block

3.6 CAN Transmission (Torque Command Frame)

The ECU transmits the throttle command and operating mode through the CAN network. The message structure follows Message ID 300 – “Torque_Command”(Table 4)(Table 5)**:**

Signal	Start Bit	Length (bit)	Data Type	Factor	Description
Torque_Req	0	32	single	1000	PID-generated throttle demand
Mode	32	8	uint8	1	Active ECU mode
Reserved	40	8	uint8	1	Reserved for future signals

Table 4: Torque Command Pack parameters

Signal	Start Bit	Length (bit)	Data Type	Factor	Description
Torque_Req	0	32	single	0.001	PID-generated throttle demand
Mode	32	8	uint8	1	Active ECU mode
Reserved	40	8	uint8	1	Reserved for future signals

Table 5: Torque Command Unpack parameters

This ensures that both PID control output and state **i**nformation are transmitted simultaneously, mirroring how a real ECU communicates with the powertrain controller.

3.7 Why It’s the heart of the system

This ECU Stateflow structure is the control core of the entire model because it integrates:

1. Dynamic decision logic — determines what mode the vehicle is in.
2. PID-based control — computes throttle and torque commands.
3. CAN communication — transmits real-time data between controller and plant.

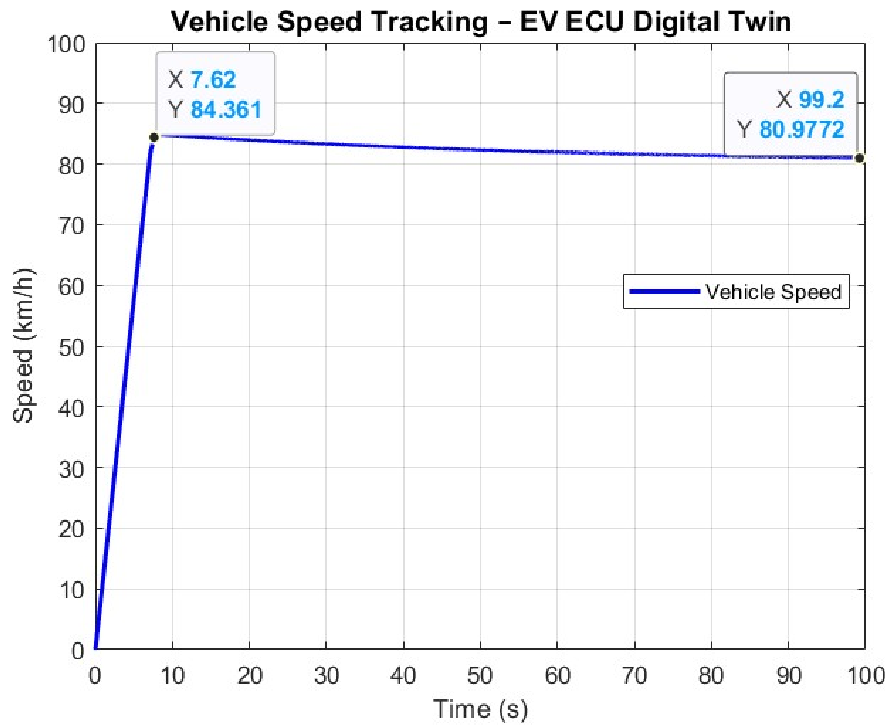
Together, these make the ECU both the “brain” (control intelligence) and the “nervous system” (communication pathway) of the digital twin model.

4.Simulation Results

The plot illustrates the response of the vehicle speed (blue line) over a 100-second simulation period. The controller successfully tracks the desired speed profile generated by the ECU(Graph 1).

- The vehicle accelerates rapidly from rest, reaching approximately 84.8 km/h at around 7.8 seconds.
- After the initial acceleration, the system stabilizes near the reference value of 80 km/h, showing a small steady-state deviation of less than 5%.
- The rise time is short, indicating a fast response of the control system.
- The small overshoot (~5 km/h) demonstrates a slightly aggressive but acceptable tuning of the PID controller.
- The settling behavior confirms that the ECU control logic maintains speed stability with minimal oscillation.

This result verifies that the PID parameters, tuned using the Ziegler–Nichols method, achieve a well-balanced dynamic performance—fast transient response with stable steady-state behavior. The CAN-based ECU structure effectively transmits control commands and feedback signals, enabling real-time speed tracking consistent with digital twin validation objectives.



Graph 1: Vehicle speed tracking

5. Common Issues and Fixes

Issue	Cause	Solution
"Signal overlap in CAN Pack"	Bit range conflict	Reassigned bits (0–31, 32–39, 40–47)
Wrong scaling of Torque_Req	Mismatch between Pack/Unpack factor	Set consistent scaling (1000 ↔ 0.001)
CAN data not updating	Virtual bus mismatch	Added "To Nonvirtual Bus" before CAN Pack
Unstable PID torque curve	Unsynchronized sample times	Aligned CAN and simulation step size

Table 6: Common issues and fixes

After these corrections, the system achieved accurate data transfer and torque stability across all states.

6. Conclusion

This study successfully demonstrates a CAN-based digital twin of an electric vehicle ECU system.
The integrated approach allows real-time simulation of communication, control, and dynamics in MATLAB/Simulink.

7. References

[1]MathWorks,“Vehicle Network Toolbox – CAN Communication Interface,” Documentation, 2024.
[2] MathWorks, “Simulink Control Design – PID Tuning Methods,” Documentation, 2024.
[3] ISO 11898-1:2015 – Road Vehicles – Controller Area Network (CAN).
[4] R. Rajamani, Vehicle Dynamics and Control, Springer, 2012.
[5] Gao, D. W., “Modeling and Simulation of Electric and Hybrid Vehicles,” Proc. IEEE, 2008.

8.Appendix A – MATLAB Code for ECU System

```
clear; clc; close all;

% --- Get simulation output structure ---

out = sim('ECU_Durum_Makinesi');    % replace with model name

if evalin('base','exist(''out'',''var'')')

    out = evalin('base','out');
```

```
else

    error('Simulation output variable ''out'' not found in workspace. Run the
simulation first.');

end

% --- Extract signals from the Simulink.SimulationOutput object ---

try

    Vehicle_Speed = out.Vehicle_Speed;

catch

    error('Vehicle_Speed or Speed_ref not found inside the simulation output
(out).');

end

% --- Extract time and data vectors ---

tV = Vehicle_Speed.Time;

v  = Vehicle_Speed.Data;

% --- Plot ---

figure('Color','w');

hold on;



grid on; box on;

% Vehicle speed (blue solid)

plot(tV, v, 'b', 'LineWidth', 1.8);



xlabel('Time (s)', 'FontSize', 11);

ylabel('Speed (km/h)', 'FontSize', 11);

title('Vehicle Speed Tracking - EV ECU Digital Twin', ...
```

```
    'FontWeight', 'bold', 'FontSize', 12);



legend('Vehicle Speed', 'Location', 'best');

ylim([0 100]);
```