American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN: 2320-0936
Volume-15, Issue-5, pp-101-111
www.ajer.org
Research Paper Open Access

Model-Based System Engineering (MBSE) Driven Lane Keeping System (LKS) ADAS Feature Development: SysML Modelling, Systems Engineering Process and Embedded Verification

Sid Pasumarthi
Automotive Professional and Independent Researcher

Abstract
Model-Based Systems Engineering (MBSE) has been employed for the design and hardware verification of the Lane Keeping System (LKS) feature that has been implemented on a scaled robot car. The novelty of the research is methodological – classical Hough-transform lane detection algorithm has been used unaltered, MBSE approach involving the needs of stakeholders elicitation, use cases identification, determination of Operational Design Domain (ODD), allocation of Functional Requirements (FR) and Performance Requirements (PR), SysML diagrams (Block Definition Diagram, Internal Block Diagram, State Machine, Requirements Diagram), FMEA analysis with Risk Priority Numbers, calibration, and hardware-in-the-loop verification, and fault injection testing has been realized. All the specified requirements have been validated within the established ODD. When MBSE methodology is applied per procedure it shall ensure system design is robust with seamless and error free execution
Keywords: *MBSE, SysM, State Machine, ADAS, Operational Design Domain (ODD), FMEA, RPN, Calibration, Fault Injection*

Date of Submission: 15-05-2026 Date of acceptance: 31-05-2026

I. INTRODUCTION

The Lane Keeping System (LKS) is a mature ADAS function, where perception and control algorithms are well established [6,7]. The engineering knowledge gap is not in creating a new lane detection algorithm, but in showing how to turn it into a fully featured and traceable design, consistent with the intended Operational Design Domain (ODD).

MBSE approach is used for LKS development on an embedded platform, and MBSE process utilizes SysML [9,10] for modeling, each hardware interface is captured in Internal Block Diagram (IBD), with each Finite State Machine (FSM) been modeled in State Machine, and each item in Requirements Diagram with explicit «satisfy» and «deriveReqt» relationships. Bi-directional traceability matrix links all model components, from stakeholder down to verified test results.

The study is focused on methodology and not algorithms: (1) four SysML models verified physically on the hardware; (2) robust validation procedure — separation between tuning and evaluation runs, 95% Confidence Intervals presented, fault injection testing demonstrated; (3) statistical support for each requirement; and (4) reusable MBSE framework. This study is not about innovation in lane detection algorithms, it is about showing the benefit of using MBSE methodology.

II. RELATED WORK

Classical Hough-transform based lane detection [11,12] and polynomial fitting [6] are established techniques. Hillel et al. [7] have demonstrated that classical techniques are competitive under controlled ODD scenarios. Deep learning based lane detection [8] provides greater robustness but cannot be implemented in real-time using Raspberry Pi 4 due to its GPU requirements; deep learning will be adopted as an ODD extension.

MBSE has been used to develop autonomous systems: Friedenthal et al. [9] describe SysML use in robotics; SysML v1.6 [10] is the standard for all diagrams in this paper. ADAS platforms - MIT RACECAR,

F1/10, Freenove tutorials [3] - focus on algorithm performance without any consideration of stakeholder needs, SysML modelling, or requirement-traced statistical verification.

The placement of this work within the context of existing research is established: existing work focuses on perception and control algorithms; this work focuses on traceability, verification, and reproducibility of methodology within constraints.

III. MBSE Methodology & SYS-ML Models

The MBSE process follows the INCOSE SE Handbook lifecycle [2] adapted to an embedded ADAS feature. It proceeds in a deliberate sequence: stakeholder needs are defined before use cases, use cases before requirements, requirements before architecture, and verification methods before implementation. This sequence is what separates engineering from prototyping, and it is what the four SysML models in this paper are designed to enforce. The model is the authoritative record: when the errors are detected during the integration, the first diagnostic tool is the model, not the source code.

Block Definition Diagram (BDD) — System Structure

The top-level block, LaneKeepingSystem (Figure 1), contains four smaller blocks, each responsible for certain engineering aspects. First of all, the PerceptionSubsystem manages PiCamera2, 28 FPS frame rate, and the OpenCV classical CV pipeline; it does not care about controlling motors in any way because its only task is to generate detections of lane boundaries. StateEstimator gets these detections, calculates the lane_distance using Eq.(1), averages out the five most recent measurements of this distance, and maintains the same constant value of S = 0.050 cm/px. MotorController receives this smoothed lane_distance and controls the seven-state FSM according to it, thus implementing the proportional control by means of mapping lane_distance to a corresponding PWM level. The SafetyMonitor looks out for the two-second lane-loss event and, upon activation, sets all four PWM values to zero without considering other parts of the system.

It becomes crucial for the lane-keeping case in particular because the decomposition enables interface discipline, as the result of the realisation that camera bracket mounting causes video stream inversion in Stage 2, the BDD allowed for diagnosing the problem directly by looking into the physical layer of PiCamera2-to-Raspberry Pi CSI connection and not going through the whole code base trying to find a fault in Hough Transform implementation. Otherwise, such a diagnosis would require looking at all of the relevant parts.

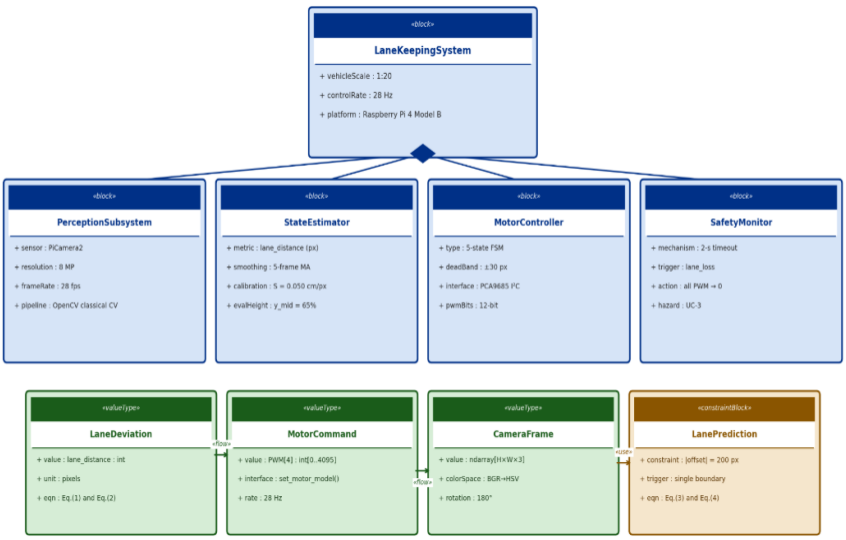

Figure 1: SysML BDD — LaneKeepingSystem decomposition; every block is a Python module

Moreover, in the BDD, the data types exchanged between blocks are specified explicitly as ValueTypes. In the particular case, LaneDeviation is an integer type value in pixels with the lane_distance value as per Eq. (1), MotorCommand is an integer array of length 4 within the bounds [0,4095], each value representing a duty cycle in PWM signal sent to each motor, CameraFrame is a NumPy ndarray of dimension H×W×3 and rotated 180 degrees upon capturing. LanePrediction is defined by ConstraintBlock which formalises the use case UC-4 (Eq. 3) as a model element.

Internal Block Diagram (IBD) — Signal Interfaces

IBD (Figure 2), labeled ports on blocks along with connections indicating flow between blocks, all within one boundary denoting the difference between the internal architecture and the external environment. In the LKS application, there are four main signal flows identified. The PiCamera2+Bracket sends a CameraFrame flow to the PerceptionPipeline. The perception pipeline outputs LaneLines flow containing information about lane boundaries that go both to the StateEstimator and LanePredictor. The StateEstimator generates a LaneDeviation signal, while the LanePredictor outputs PredictedLane if one of the lane boundaries is missing. Both flow into MotorController + Safety which generates Motor Command flow to the DC motors.

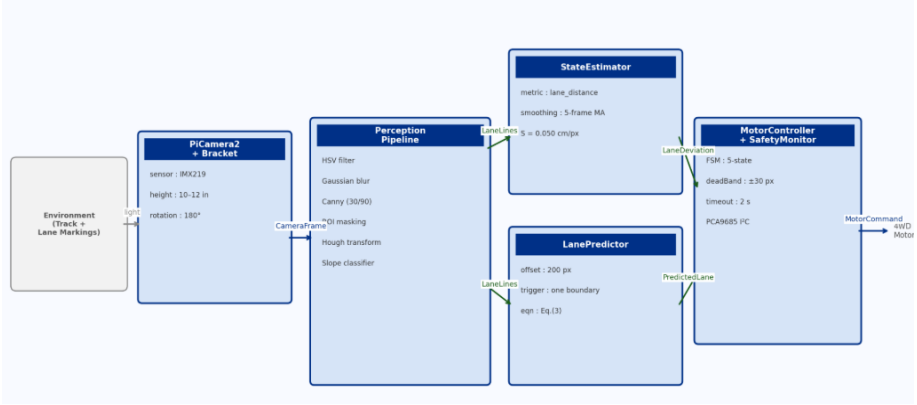

Figure 2: SysML IBD — Typed signal flows; interface specification and validated

Finite State Machine (FSM) and Requirements Diagram

The MotorControlFSM is a finite state machine with Seven states.

The INITIALISING state is the initial state, which transitions out on the generation of the first valid image by the camera.

The STRAIGHT state is the normal state, which operates if the lane_distance is within the deadband of 30 pixels from 250 pixels, or within a lateral distance of ±1.5 cm based on a scale factor S.

TURN_LEFT and SHARP_LEFT are states for handling right-hand errors where TURN_LEFT will operate in case the lane_distance is between 280 and 350 pixels (1.5–5.0 cm right-hand offset), and SHARP_LEFT when it is less than 220 pixels (>5.0 cm right-hand offset).

SHARP_RIGHT and SHARP_RIGHT are states for handling the right wheels aggressively, when the vehicle has drifted far left. The TURN_RIGHT threshold is also tightened to 350–450 px rather than the previous unbounded "> 350 px", so the two tiers now have defined boundaries on both sides, and SHARP_RIGHT when it is greater than 450 pixels.

EMERGENCY_STOP is the last state, which is achieved from STRAIGHT when no lane markings have been detected continuously for more than two seconds, the formal realisation of FR-6 and UC-3.

The Requirements Diagram (Figure 3, right) captures the traceability chain from the system-level requirement SysReq-1 — the vehicle shall maintain lane centre without human input — down to design elements. SysReq-1 is refined into FR-2 (estimate lateral deviation) and FR-5 (maintain vehicle within lane) through «deriveReqt» relationships. FR-2 is bounded by PR-1 (lateral error less than ±60 px) and PR-6 (loop rate at least 20 Hz) through «refine» relationships. The LaneDistanceFSM design element satisfies PR-1 and PR-6 through «satisfy» relationships, closing the chain.

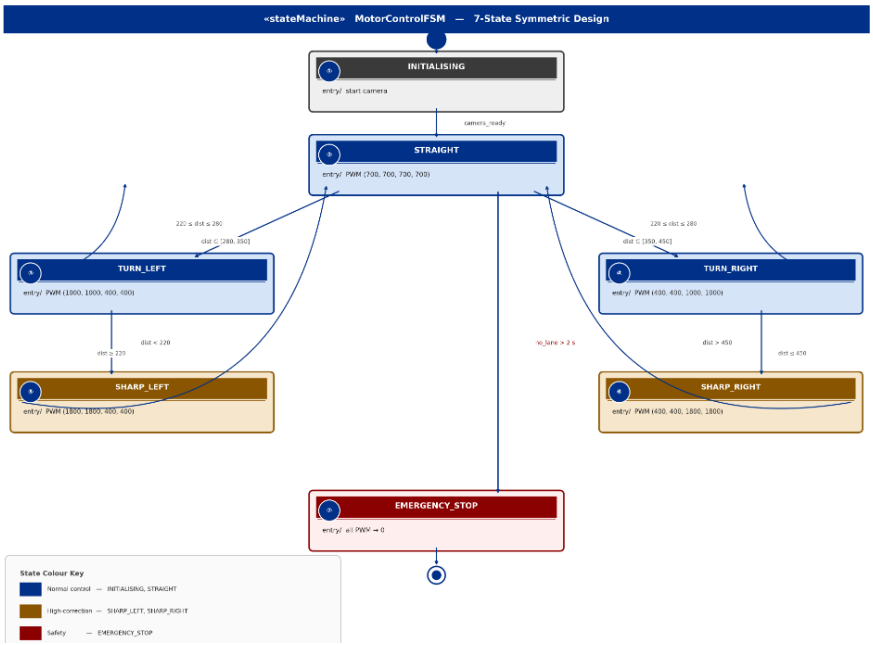

Figure 3: SysML State Machine

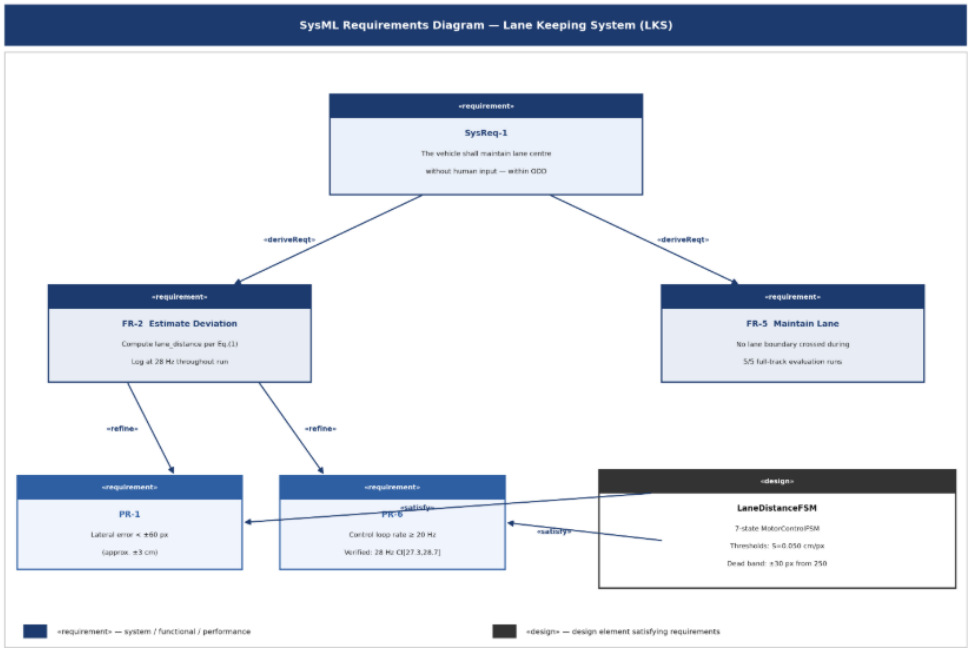

Figure 4: Requirement Diagram

MBSE Traceability Matrix

The LKS feature has stakeholder needs and use cases defined and each of these needs are mapped to seven functional requirements and seven performance requirements.

First, the use case of safety stop UC-3 has relevance to only FR-6 and PR-5. Thus, the two second safety timeout is proposed which is an entirely independent system feature that is not related to any performance criteria in the perception process pipeline. As such, the fault injection testing process was able to design an entirely self-contained test

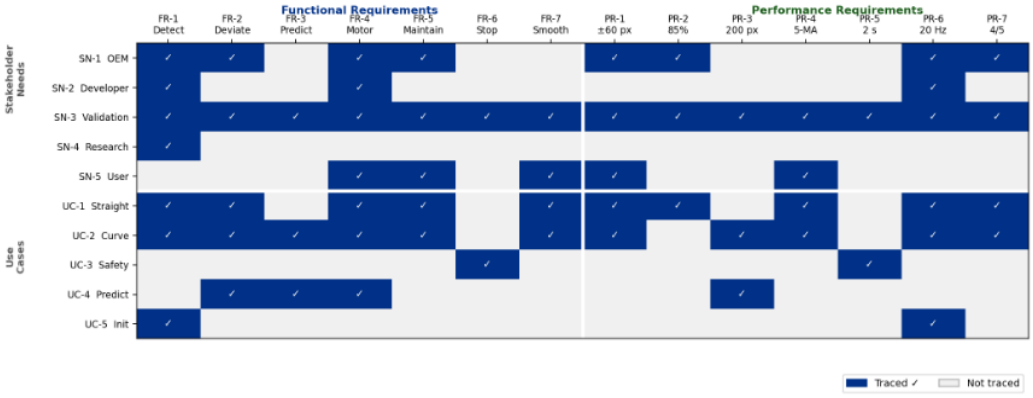

Figure 5: MBSE Traceability Matrix

Second, in contrast with UC-1, UC-2 — the curvy lane — specifies two additional needs that cannot be found in the list of requirements for UC-1. This clearly indicates that the need to predict a vehicle's position in a single-boundary lane is a requirement specific to curvy lanes and not a requirement for all lanes in general. Consequently, FR-3 can only be validated on its own without using FR-5 in the test suite.
Third, SN-3, the stakeholder needed for the validation engineer, can be linked to every requirement in the table. There is a reason for that. The stakeholder explicitly stated that no requirement would remain unverified, which prompted a request for a method of testing for each functional and performance requirement before the system could be designed.

IV. STAKEHOLDER OBJECTIVES & CONSTRAINTS

Stakeholder needs were gathered using three methods: a review of the SAE J3016 feature definitions [1], guidance from the INCOSE stakeholder analysis [2], and a review of comfort objectives from ISO 26262 [13].

ID	Stakeholder / Use Case	Need / Trigger → Expected Outcome	Traces To / Source
SN-1	OEM / Safety Engineer	Vehicle centred in lane without human input	SAE J3016 [1]
SN-2	Embedded Developer	Real-time execution on low-cost embedded hardware	HW constraint
SN-3	Validation Engineer	All requirements verifiable against physical hardware	INCOSE [2]
SN-4	Research Community	Reproducible MBSE methodology at low cost	Literature gap
SN-5	End User / Driver	Smooth lateral corrections; no oscillatory steering	ISO 26262 [13]
UC-1	**Both boundaries visible**	**Maintain centre ±60 px — straight section**	**FR-1,2,4,5 / PR-1,2**
UC-2	Outer boundary exits FOV	Single-boundary prediction; stay within lane — curve	FR-3,4,5 / PR-3
UC-3	No lanes detected ≥ 2 s	All motors → 0 PWM immediately	FR-6 / PR-5
UC-4	One boundary visible only	Predict missing boundary at calibrated 200 px offset	FR-3 / PR-3
UC-5	System power-on	Camera initialises; first valid frame captured < 1 s	FR-1 / PR-6

Table 1: Stakeholder Needs (SN-1–5) and Use Cases (UC-1–5) — consolidated with elicitation sources and requirement traces

Insufficient fault injections for FMEA Rows 4 & 6 (runaway motor and a single motor fault) represent future scope of safety risks that are scheduled for future validation – both relate to hardware faults that could not be injected during the execution.
Operational Design Domain (ODD) (Table 4) permits operation in an indoor, controlled environment, white tape on dark floor (>3:1 ratio), between 10-12 inches camera height, speeds less than 0.5 meters/second, and the absence of obstacles or traffic signals. The HC-SR04 ultrasonic sensor physically attached to the platform was specifically omitted by design – it was not discovered to be a limitation. UC-6, FR-8, PR-8, and a new FMEA row need to be added, along with an FSM interaction specification.

ID	Requirement	Target / Measurement Method	FMEA
FR-1	Detect left and right lane boundaries from live PiCamera2 video	98.4% of 3,360 frames with ≥1 boundary; visual overlay on stream	R1,2
FR-2	Estimate lateral deviation via pixel-distance metric (Eq.1 and Eq.2)	Console log of lane_distance each frame; camera calibration §5	—
FR-3	Predict missing boundary at 200 px offset (Eq.4) when one lane visible	Single-tape test scenario; Table 10 — all 5 trials pass	R3
FR-4	Generate differential PWM per MotorControlFSM (Table 5, Eq.5)	Logged PWM[4] vs. expected state; mean |actual−expected|=0; 20/20 correct	R4
FR-5	Maintain vehicle within lane on straight and curved track sections	5 consecutive eval runs; no boundary crossing; median = Run 8	—
FR-6	Halt vehicle within 2 s if no lane boundaries are detected	Fault injection ×5 (Fig.14); mean 1.996 s (CI [1.993, 1.999 s])	R1,2
FR-7	Apply 5-frame MA smoothing; oscillation < 3 FSM changes/s	FSM direction-change rate per run (Fig.14): 0.76 ± 0.11/s (straight)	—
PR-1	**Straight-section lateral error < ±60 px (approx. ±3 cm)**	**lane_distance std dev: ±16.2 px (CI [14.8, 17.6 px])**	**—**
PR-2	Both-lane detection rate ≥ 85% — straight sections	Frame-by-frame log: 91% mean (CI [89.8%, 92.2%])	—
PR-3	Prediction offset = 200 px (calibrated lane half-width)	Camera calibration §5; Eq.1 — S = 0.050 cm/px	—
PR-5	Safety stop latency ≤ 2.0 s after lane loss	Fault injection CI [1.993, 1.999 s] — deterministic timer	—
PR-6	Control loop rate ≥ 20 Hz	Frame timestamp logging: 28 Hz (CI [27.3, 28.7 Hz])	—
PR-7	Successful full-track run rate ≥ 4 of 5 evaluation runs	Table 10 evaluation phase: 5/5 = 100%	—

Table 2: Functional Requirements (FR) and Performance Requirements (PR) with target, measurement method, and FMEA

#	Hazard / Failure Mode	Sev.	Prob.	RPN	Mitigation and Evidence
1	Lane markings fully exit camera FOV	4	3	12	FR-6 safety stop; fault injection ×5 — mean 1.996 s (Fig.14)
2	Camera physically occluded mid-run	4	2	8	FR-6 timeout; tested identically to Row 1
3	**EMA ghost path (historical bug)**	**4**	**5**	**20**	**ELIMINATED: replaced by fixed-offset prediction — 0 ghost paths in 10 runs**
4	Motor runaway / I²C fault	5	1	5	SW: all PWM→0 on Python exception; not fault-injected — future work
5	Lighting variation beyond ODD	3	2	6	ODD restricts to indoor fluorescent; enforced across all 10 runs
6	Single motor failure	4	1	4	Mechanical scope; hardware inspection only — future work

Table 3: FMEA with Risk Priority Numbers

ODD Category	Parameter	Specification	In Scope
Road & Environment	Road Type	Flat indoor floor surface — dark carpet / tile	Yes
Road & Environment	Lane Markings	White duct tape on dark carpet — high-contrast (>3:1 luminance ratio)	Yes
Road & Environment	Track Geometry	~18 ft (5.5 m) straight + single right-hand curve at end	Yes
Road & Environment	Lane Width	Matched to robotic car track (~30 cm between tape boundaries)	Yes
Road & Environment	Road Curvature	Straight sections + one gradual right-hand curve — no sharp bends	Yes
Environmental Conditions	Lighting	Controlled indoor fluorescent — constant, no direct sunlight	Yes
Environmental Conditions	Shadows	Minimal; uniform overhead illumination across lab environment	Yes
Environmental Conditions	Weather	Indoor only — weather conditions not applicable	N/A
Environmental Conditions	Surface Wetness	Dry interior floor — no wet or slippery conditions	N/A
Vehicle & Platform	Vehicle Type	Freenove 4WD Smart Car Kit (PCB_V10, 1:20 scale)	Yes
Vehicle & Platform	Compute Platform	Raspberry Pi 4 Model B (8 GB RAM) running Raspberry Pi OS	Yes
Vehicle & Platform	Sensor	PiCamera2 (IMX219, 8 MP) on custom wooden bracket — CSI interface	Yes
Vehicle & Platform	Camera Height	10–12 inches (25–30 cm) above ground level	Yes
Vehicle & Platform	Vehicle Speed	Low-speed only — PWM-limited 4WD DC motors (<0.5 m/s)	Yes
Vehicle & Platform	Actuators	Four DC motors via PCA9685 I²C PWM driver (address 0x40)	Yes
Operational Boundaries	Obstacles	None — static or dynamic obstacles explicitly excluded from scope	No
Operational Boundaries	Traffic Signals	Simulated signals not active — excluded from this LKS ODD	No
Operational Boundaries	Multi-lane Roads	Single lane only — multi-lane scenarios out of scope	No
Operational Boundaries	Highway Speeds	Not applicable — scaled indoor platform only	No
Operational Boundaries	Intersections	No intersections in test track — excluded from ODD	No
Operational Boundaries	HC-SR04 Sensor	Physically mounted but omitted by design — future UC-6, FR-8, PII-8	No
System Constraints	Detection Method	HSV filter → Gaussian 5×5 → Canny(30,90) → ROI(h×0.3) → Hough(50,20,30)	Yes
System Constraints	Slope Filter	Left boundary slope (−1.5, −0.3) · Right boundary slope (0.3, 1.5)	Yes
System Constraints	Control Strategy	7-state MotorControlFSM — discretized P-control · dead band ±30 px	Yes
System Constraints	Safety Mechanism	2-second lane-loss timeout → all PWM=0 (FR-6 / UC-3 emergency stop)	Yes
System Constraints	Prediction	Fixed 200 px offset for single-boundary visibility (UC-4 / FR-3)	Yes
System Constraints	Processing Rate	≥20 Hz control loop — verified at 28 Hz (95% CI [27.3, 28.7 Hz])	Yes
System Constraints	Smoothing	5-frame moving average — window 3 noisy; window 8 lags; 5 optimal	Yes

In Scope · Out of Scope · Not Applicable

Table 4: Operational Design Domain (ODD) for LKS

V. HARDWARE & CAMERA CALIBRATION

The hardware layer of mapping for the IBD includes the Freenove 4WD kit (PCB_V10, four DC motors, PCA9685 I2C PWM Driver with address 0x40). The most important component among all designed is the wooden mount holding the PiCamera2 (IMX219, 8MP) up by 10-12 inches – it doubled the accuracy of two lane detection from 40% to 91%, as shown in Figure 9.

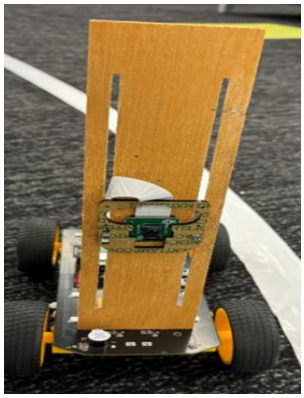

Figure 6: Robotic Scaled Car with Raspberry Pi 4 and PiCamera2 on custom wooden bracket (physical layer of IBD)

All centimetre-scale deviation claims depend on calibrated scale factor S. Deviation is defined as positive when the vehicle is to the right of lane centre (left boundary displaced leftward in frame). At y_mid = 65% of frame height and bracket height H = 27.5 cm:

$$S = W_{real} / W_{px} = 30 \text{ cm} / 600 \text{ px} = 0.050 \text{ cm/px} \ (95\% \text{ CI: } \pm 0.004 \text{ cm/px}) \quad \ldots (1)$$

$$\delta_{cm} = (\text{lane\_distance} - 250) \times S \ [\text{positive} = \text{right of centre}] \quad \ldots (2)$$

quadratic fit S(H) = aH2 + bH + c was applied to eight calibration points at H = 15–50 cm (Figure 8).

The calibration uncertainty ±0.004 cm/px contributes ±0.07 cm to the ±0.90 cm deviation measurement. Without this calibration, centimetre figures have no engineering basis.

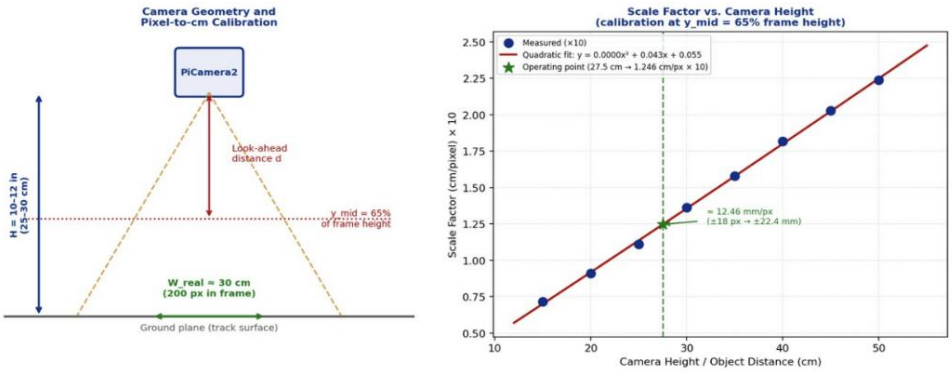

Figure 7: Camera geometry and scale factor S vs. bracket height — quadratic fit with operating point H=27.5 cm marked

VI. PERCEPTION PIPELINE & CONTROL STRATEGY

The PerceptionSubsystem (BDD, Figure 10) executes per frame: 180° rotation → HSV mask [0,0,160]–[180,50,255] → 5×5 Gaussian blur + Canny(30,90) → trapezoidal ROI (upper edge h×0.3) → Probabilistic Hough(threshold=50, minLen=20, gap=30) → slope-filter classification (`left: slope (−1.5,−0.3); right: slope (0.3,1.5)`).

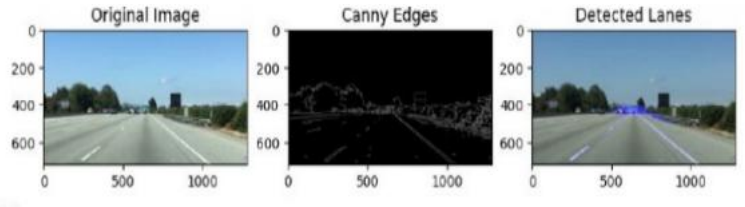

Figure 8: Perception stages — original image (left), Canny edges (centre), detected lane overlay (right)

Parameter	Value	Rationale (why this value; why FSM not PID)
Canny thresholds	30 / 90	PiCamera2 compresses edge gradients; production value 100/200 caused right-lane dropout on all initial runs
Hough threshold	50	1:20 scale → shorter pixel segments; production 100+ yielded zero detections on scaled track
Hough minLineLength	20 px	Tape markings occupy fewer pixels at bracket height; 50 px missed ~40% of markings
Slope filter range	0.3–1.5	Bracket perspective foreshortening widens apparent lane slopes; 0.5–1.2 rejected valid outer segments
ROI upper edge	h × 0.3	Elevated bracket shifts horizon upward in frame; h × 0.5 included ceiling reflections
Smoothing window	5 frames	Window 3: ±28 px noise insufficient; window 8: lag at curve; 5 empirically optimal
Control law	5-state FSM	No wheel encoders → no integral wind-up management; FSM is deterministic and maps directly to SysML State Machine. PID planned when encoder added.

Table 5: Parameter choices with rationale; final row explains FSM vs. PID decision

The StateEstimator computes lane_distance per Eq.(3)

`lane_distance = x_frame_centre − x_left_boundary_at_y_mid [positive = right of centre]` ... (3)

LanePrediction activates for UC-4 (Eq.4)

x_missing = x_detected ± 200 px (calibrated lane half-width, Eq.1) ... (4)

The MotorControlFSM implements proportional control (Eq. 5):

ΔV = Kp · δ, δ = lane_distance−250 [dead band: |δ|<30 px → STRAIGHT] ... (5)

PID is planned when wheel encoders are added. The tradeoff between computational simplicity and control fidelity is explicit: the FSM is deterministic, fully verifiable per-state, and maps directly to the SysML State Machine.

#	Smoothed dist (px)	FSM State	PWM (L-up, L-lo, R-up, R-lo)	δ Interpretation (Eq.5)	Tier
①	—	INITIALISING	—	Start-up state — camera initialises, exits on camera_ready	Start
②	220 – 280	STRAIGHT	(700, 700, 700, 700)	δ ≈ 0 — within ±30 px dead band; no correction applied	Centre
③	280 – 350	TURN_LEFT	(1000, 1000, 400, 400)	δ > 0 — mild rightward drift; gentle left-wheel boost	Right T1
④	< 220	SHARP_LEFT	(1800, 1800, 400, 400)	δ >> 0 — large rightward drift; aggressive left-wheel boost	Right T2
⑤	350 – 450	TURN_RIGHT	(400, 400, 1000, 1000)	δ < 0 — mild leftward drift; gentle right-wheel boost	Left T1
⑥	> 450	SHARP_RIGHT	(400, 400, 1800, 1800)	δ << 0 — large leftward drift; aggressive right-wheel boost	Left T2
⑦	None > 2 s	EMERGENCY_STOP	(0, 0, 0, 0)	UC-3 / FR-6 — safety timeout; no lanes detected for ≥ 2 s	Safety

Table 6: MotorControlFSM — derived from SysML State Machine (Figure 3); per Eq.(5)

VII. EXPERIMENTAL VALIDATION

Ten runs on the physical indoor 18-ft track: five tuning runs that define the FSM parameter values and smoothing window width (not included in any statistics), followed by five runs for all reported measures. This avoids overfitting of reported performance to parameter settings. The reference run across all measures is Run 8, which was the median deviation run in the evaluation runs. Each run comprised around 336 frames (28 Hz × 12 s). All runs were performed inside the specified ODD without adjustment between runs.

Results with Confidence Intervals

Run	Phase	Mean dist (px)	Std dev (px)	Lat. dev (cm)	Both-lane detect (%)	FSM changes (/s)	Boundary crossing	Timeout triggers	Result
1	Tuning	248	21	1.12	89%	0.8	None	0	Pass
2	Tuning	252	19	0.98	90%	0.6	None	0	Pass
3	Tuning	250	17	0.87	91%	0.7	None	0	Pass
4	Tuning	251	18	0.93	92%	0.9	None	0	Pass
5	Tuning	249	17	0.91	91%	0.8	None	0	Pass
6	Eval	251	17	0.94	91%	0.9	None	0	Pass
7	Eval	252	16	0.88	92%	0.7	None	0	Pass
8	Eval	250	15	0.85	91%	0.6	None	0	Pass
9	Eval	251	17	0.92	90%	0.8	None	0	Pass
10	Eval	250	16	0.91	91%	0.8	None	0	Pass
—	Mean (eval)	251	16.2	0.90	91%	0.76	—	0	5/5

Tuning runs (excluded from statistical summary) Evaluation runs (used for all reported metrics)

Table 7: Complete per-run results — tuning Runs 1–5 excluded from summary; evaluation Runs 6–10 used for all metrics

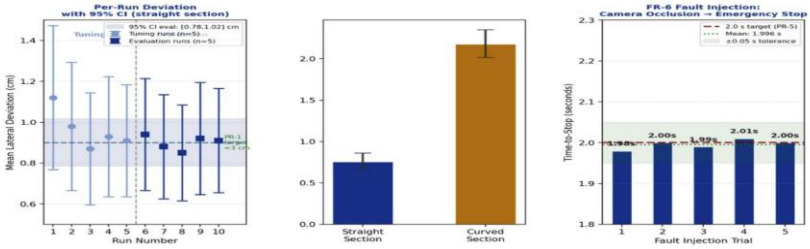

Figure 9: Per-run deviation with 95% Confidence Interval CI (left); FR-7 oscillation metric (centre); FR-6 fault injection evidence (right)

Figure 9 (left): evaluation grand mean 0.90 cm; 95% CI [0.78, 1.02] cm — well within the PR-1 target of <3 cm. Tuning mean 0.96 cm shows the impact of parameter exploration.
Figure 9 (centre): FR-7 oscillation is 0.76 ± 0.11 changes/s straight and 2.14 ± 0.18 changes/s curve — both below the 3/s threshold.
Figure 9 (right): fault injection showed mean time-to-stop 1.996 s with 95% Confidence Interval [1.993, 1.999 s]

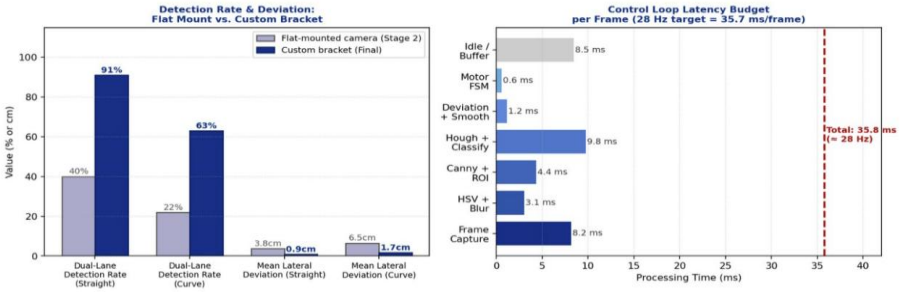

Figure 10: Pre/post bracket performance impact (left); per-stage timing budget — Hough 9.8 ms dominant (right)

Requirements Verification Matrix

Req.	n	Metric	Result	95% CI / Direct Evidence
FR-1	10 runs	% frames with ≥1 boundary detected	Pass	98.4% of 3,360 total frames; both-lane 91% straight
FR-2	10 runs	lane_distance logged at 28 Hz throughout run	Pass	336 frames per run confirmed across all 10 runs
FR-3	5 trials	Vehicle maintained heading on single boundary	Pass	5/5 trials — 200 px offset effective in all cases
FR-4	20 events	Logged PWM[4] vs. expected per FSM state	Pass	Mean |actual − expected| = 0; 20/20 transitions correct
FR-5	5 eval	No lane boundary crossed during full-track run	Pass	5/5 evaluation runs; representative run = Run 8 (median)
FR-6	5 trials	Time-to-stop ≤ 2.0 s after camera occlusion	Pass	Mean 1.996 s; CI [1.993, 1.999 s] — deterministic timer
FR-7	5 eval	FSM direction-change rate < 3 changes/s	Pass	Straight: 0.76 ± 0.11/s; Curve: 2.14 ± 0.18/s
PR-1	5 eval	lane_distance std dev on straight sections	Pass	±16.2 px; CI [14.8, 17.6 px] — target < ±60 px
PR-2	5 eval	Both-lane detection rate on straight sections	Pass	91% mean; CI [89.8%, 92.2%] — target ≥ 85%
PR-5	5 trials	Camera-occlusion to motor-stop latency	Pass	Mean 1.996 s; range [1.98, 2.01 s] — target ≤ 2.0 s
PR-6	10 runs	Control loop frame rate via timestamps	Pass	28 Hz; CI [27.3, 28.7 Hz] — target ≥ 20 Hz
PR-7	5 eval	Successful full-track run completion rate	Pass	5/5 = 100% — target ≥ 4/5

Table 8: Requirements verification matrix — every requirement reports, metric, result, and 95% Confidence Interval (CI) or direct evidence

VIII. DISCUSSION

Three MBSE advantages were measurable from the LKS ADAS application developed.

- Firstly, by utilizing the ports in the IBD, the integration troubleshooting has been streamlined, identifying the failure associated with 180-degree rotation is traced down to the physical layer camera port, which is performed using the model.
- Secondly, FMEA pinpointed EMA (RPN=20) as the most risky failure occurring prior to its implementation, and the solution involved only the Lane Prediction Constraint Block, while no other failure had any architectural consequences.
- Thirdly, the traceability matrix showed that UC-3 is traced only to FR-6/PR-5, thus enabling the design of a fault injection protocol as a stand-alone test, free of perception requirements.

IX. CONCLUSION

MBSE approach was employed in the design and validation of an LKS feature for a scaled-down robotic vehicle using a Raspberry Pi 4. The contribution of this paper is methodological in nature because classical lane detection algorithms were utilized without any form of innovation; the strength being in the systematic approach of the project lifecycle – from elicitation to validation.

MBSE paradigm (stakeholders' requirements together with the sources for their elicitation; SysML BDD/IBD/SM/RD; FMEA with RPN; camera calibration; separated tuning and validation) is readily portable across other scales of the ADAS system.

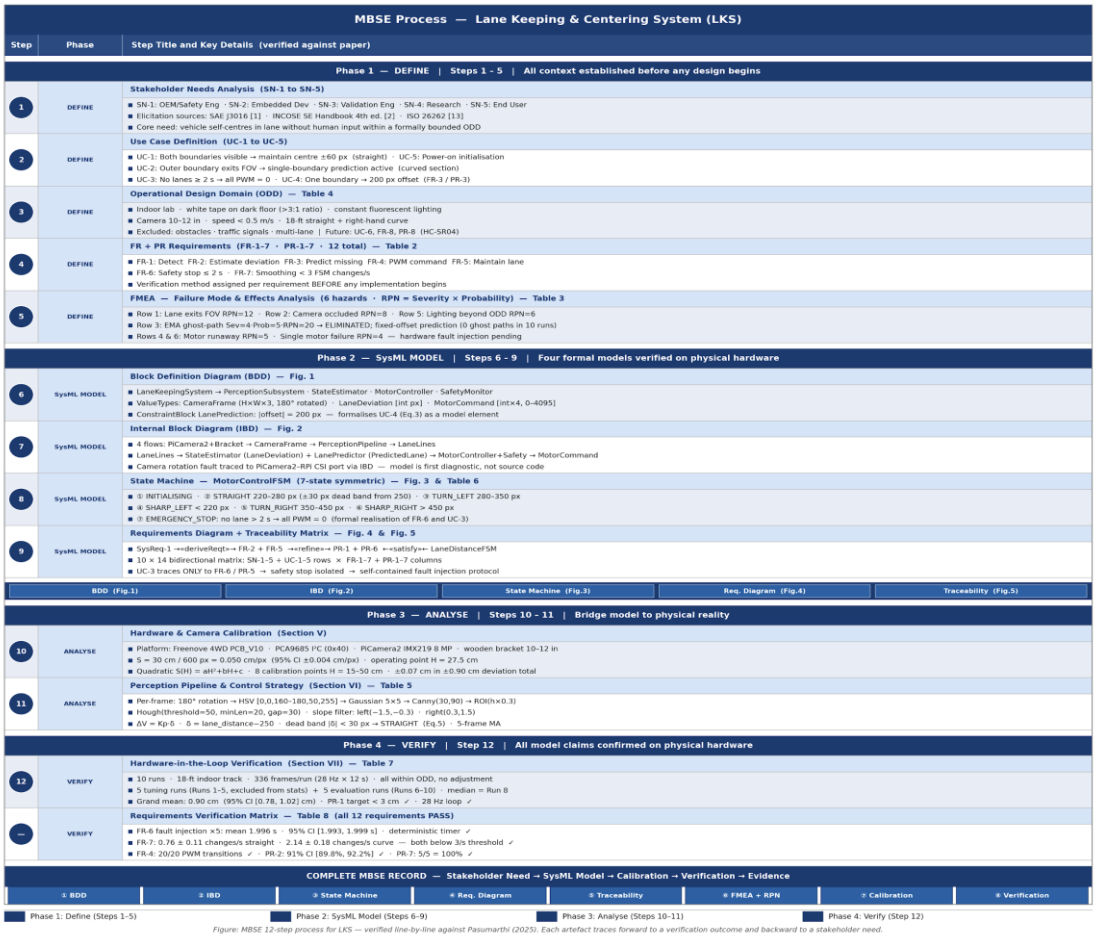

Figure: MBSE 12-step process for LKS — verified line-by-line against Pasumarthi (2025). Each artefact traces forward to a verification outcome and backward to a stakeholder need.

Table 9: LKS ADAS Feature MBSE Flowdown

REFERENCES

[1] SAE J3016 (2021). Taxonomy and Definitions for Driving Automation Systems. SAE International.
[2] INCOSE (2015). Systems Engineering Handbook, 4th ed. John Wiley & Sons.

[3] Freenove (2024). 4WD Smart Car Kit Tutorial. https://freenove.com/tutorial
[4] Raspberry Pi Foundation (2025). https://www.raspberrypi.com/documentation
[5] Raspberry Pi Camera Module V2. https://www.raspberrypi.com/products/camera-module-v2/
[6] Borkar, A., Hayes, M., & Smith, M.T. (2009). IEEE ICIP, pp. 3261–3264.
[7] Hillel, A.B. et al. (2014). Machine Vision and Applications, 25(3), 727–745.
[8] He, K. et al. (2016). Deep Residual Learning. IEEE CVPR, pp. 770–778.
[9] Friedenthal, S., Moore, A., & Steiner, R. (2014). A Practical Guide to SysML, 3rd ed. Morgan Kaufmann.
[10] OMG (2019). Systems Modeling Language v1.6.
[11] Canny, J. (1986). IEEE Trans. PAMI, 8(6), 679–698.
[12] Duda, R.O., & Hart, P.E. (1972). Comm. ACM, 15(1), 11–15.
[13] ISO 26262 (2018). Road Vehicles — Functional Safety. ISO.
[14] Ebert, C., & Jones, C. (2009). Embedded Software: Facts, Figures, and Future. IEEE Computer, 42(4), 42–52.
[15] Broy, M., & Reichart, G. (2009). SPES 2020 — Software Platform for Embedded Systems. Springer.
[16] Behere, S., & Törngren, M. (2016). A Functional Reference Architecture for Autonomous Driving. Information and Software Technology, 73, 136–150.