Smart Car with Smart Infrastructure Sprint Plan

AISE4020 – Design Project

Academic Weapons

Sprint 1 — Open-Source Stack + Compatibility + Environment Prep (10 Tasks)

1) Apply supervisor feedback and freeze the sprint roadmap.
After approval, the team will review feedback and identify any required scope adjustments. The minimum viable demo will be locked, so the team always knows what must work first. Any stretch goals will be marked clearly as optional. This ensures the next tasks focus on execution rather than debating scope.

2) Translate the proposal into a short "requirements + success tests" list.
The approved features will be rewritten as simple requirements such as lane keeping, traffic light stop, and obstacle safety stop. For each, the team will define how it will be tested in the environment. These checks will be used at the end of every sprint to prove progress. This creates measurable goals instead of vague "it works" statements.

3) Select the open-source software baseline and record sources.
The team will decide on the open-source libraries to use for camera input, vision processing, and motor control integration. Each library will be verified for documentation quality and install stability. Links and short usage notes will be saved for the final submission. This prevents switching tools mid-project under pressure.

4) Confirm the final sensor and actuator list for redundancy planning.
A final list of sensors and actuators will be chosen with redundancy in mind. The camera will be treated as the main perception sensor, with distance sensing used as the safety fallback. Motors and motor driver control will form the core actuation layer, with indicators added for system state visibility. This locks the physical capability of the car before coding begins.

5) Run "hello world" tests for every selected hardware component.
Each component will be tested independently to confirm it works on the chosen platform. The camera must provide stable frames, the distance sensor must return consistent readings, and the motor driver must respond to PWM correctly. Any component that fails basic testing will be replaced early. This avoids losing time later to hardware uncertainty.

6) Define the feature-to-sensor mapping for initial and redundant sensing.
The team will map each feature to its primary sensing source and a fallback option. Lane keeping will begin as vision-first, with a fallback route available if needed. Obstacle safety will start as a distance-sensor first to keep testing safe. This mapping guides how features get added sprint by sprint.

7) Prepare a minimal environment setup that supports early perception testing.
Instead of building the entire smart infrastructure immediately, the team will create only what is needed for Sprint 2. This will include lane markings with consistent contrast and a controlled obstacle zone. The focus is repeatability, not realism. Later infrastructure elements will only be added once the pipeline is stable.

8) Set up the team development workflow and integration rules.
A shared repository will be structured into perception, planning, control, and hardware of I/O modules. Naming conventions and simple interface standards will be agreed early, so merge conflicts are minimized. Basic test scripts will be created to validate modules without full system runs. This lets multiple members contribute without breaking integration.

9) Create the "pivot rules" that trigger simplified alternatives.
The team will define clear signs that the project is becoming too complex, such as unstable detections or non-repeatable driving. For each risk, a fallback plan will be pre-approved, such as IR-based lane tracking or marker-based signs. These rules prevent last-minute feature chaos. The result is an agile but controlled development process.

10) Complete Sprint 1 review and confirm readiness for implementation work.
Sprint 1 will end with a checklist showing that tools install, hardware responds, and the baseline environment exists. Any remaining risks will be written as Sprint 2 priorities. The team will confirm which feature will be implemented first using a single-sensor approach. This sets a clean starting point for Sprint 2.

Sprint 2 — One Sensor + One Feature + Placement Planning (10 Tasks)

11) Implement the first end-to-end feature using one primary sensor.
Sprint 2 begins by proving a complete autonomy loop is possible. The team will implement lane sensing using the camera as the only main input for the feature. The output will be a usable lane error signal for driving decisions. This establishes the perception-to-control pipeline in a simple scope.

12) Validate the perception output before connecting full control.
Lane detection results will be tested on recorded or live frames in the prepared environment. The goal is stability across time, not perfect detection on one frame. The team will confirm the output signal changes logically as the car shifts position. This prevents unstable perception from creating unstable driving later.

13) Convert perception output into a control-ready signal.
The lane output will be turned into a lane-center offset value with a consistent sign convention. This is the "bridge" between perception and actuation. The team will verify that small errors lead to small corrections and large errors produce stronger corrections. Once this mapping is correct, driving behavior becomes predictable.

14) Implement lane-keeping control with conservative tuning.
Differential motor control will be used to steer toward the lane center. Tuning will begin at low speed to reduce crash risk and support debugging. The goal is repeatable lane following, not maximum speed performance. This creates the baseline autonomy feature for later expansion.

15) Add the safety sensor layer as a separate, higher-priority override.
Even though Sprint 2 is "one sensor + one feature," the system will include safety redundancy early. The distance sensor will trigger an emergency stop that overrides lane-keeping commands. This keeps testing safe and mirrors real autonomy safety layering. It also becomes the first example of multi-sensor interaction.

16) Run repeatability tests and measure baseline performance.
The team will run multiple short trials using the same track and record outcomes. Failures will be categorized into perception issues, control issues, or environment issues. Small changes will be applied one at a time to confirm what improves performance. This creates real evidence to guide Sprint 3 expansion.

17) Trial one fallback alternative to estimate "complexity cost."
A reduced-scope backup method will be tested, such as IR lane sensing or simplified lane

geometry. The point is not to replace the main design, but to confirm the team has a working escape route if needed. Performance and setup effort will be compared against vision-only lane tracking. This gives the group a realistic complexity model.

18) Use Sprint 2 results to finalize sensor placement on the vehicle body.
Based on what worked and what struggled, the team will decide where sensors must be mounted for best visibility and reliability. Camera height and angle will be fixed based on lane visibility requirements. Distance sensor placement will be chosen to avoid false echoes and blind zones. This planning step is what enables the chassis build in Sprint 3.

19) Perform a Sprint 2 "scope health check" before scaling features.
The team will decide whether to keep full vision lane tracking as the primary method or activate a hybrid fallback. Any unstable elements will be simplified now, not later. The outcome will be a realistic plan for multi-feature integration rather than optimistic guessing. This protects Sprint 3 from overload.

20) Deliver the Sprint 2 demo and document what is now proven.
Sprint 2 will end with a demonstration of lane keeping plus emergency stop safety behavior. The team will collect short logs or video evidence of repeatable operation. What is proven stable will become the baseline moving forward. This is the handoff into Sprint 3 system expansion.

Sprint 3 — Chassis Integration + Redundancy Expansion (10 Tasks)

21) Build the full chassis with sensor mount locations finalized.
Sprint 3 begins by turning the sensor placement plan into a solid physical build. Sensors will be mounted securely so their calibration does not shift during driving. Wiring will be fixed with strain relief to prevent intermittent faults. This creates a reliable hardware platform for multi-feature operation.

22) Add redundancy for one feature using two sensor sources.
The team will implement redundancy for one key feature, typically lane tracking. Vision remains the primary method, while a secondary sensor method supports fallback operation. The switching logic will use confidence rules so the system degrades safely instead of failing abruptly. This demonstrates the automotive redundancy principle directly.

23) Expand the camera's role into a second feature.
After lane keeping is stable, the vision pipeline will be extended to detect one infrastructure element such as a traffic light state. This tests whether the compute setup can handle multi-feature vision processing. If performance drops, the feature scope will be simplified rather than forcing heavy algorithms. This step validates scalability.

24) Integrate traffic light detection into driving decisions.
Traffic light detection will feed the planning layer as a stop/go input. The vehicle must remain stable in-lane while transitioning between cruise and stop. The goal is smooth, predictable stops that show rule compliance. This adds realistic infrastructure response beyond lane following.

25) Add one sign-based behavior using a controlled sign set.
A simple sign behavior such as stop sign detection will be integrated next. Sign placement will be standardized in the environment to support repeatability. If classification is unstable, a fallback such as marker-based identification will be used. This keeps the feature achievable while still demonstrating vision-driven interpretation.

26) Strengthen the planning module to handle multi-feature inputs.
The planning module will be upgraded into a clearer finite state machine with priorities. Emergency stop will always override traffic compliance, and traffic compliance will override normal cruising. This prevents conflicting decisions when multiple detections occur close together. It also makes the system easier to explain in the final demo.

27) Improve control stability under planning transitions.
Control tuning will focus on smooth acceleration changes and stable stopping behavior. Steering corrections must remain stable when the vehicle slows down or resumes motion. The system should avoid oscillation and aggressive corrections that break perception. This makes multi-feature autonomy look intentional, not chaotic.

28) Expand the environment gradually to support new features.
Traffic light placement and signs will be added to the existing track without changing lane quality. The team will introduce only one new infrastructure element at a time to keep testing controlled. This prevents environmental changes from hiding software issues. The environment becomes a consistent validation platform.

29) Conduct a Sprint 3 complexity review and lock Sprint 4 scope.
Sprint 3 ends with a decision: which features are stable enough for final integration and which must remain simplified. Any feature that causes instability will be reduced to its fallback version now. The team will lock Sprint 4 as “integration + reliability,” not “new features.” This ensures the final sprint stays achievable.

30) Deliver the Sprint 3 integrated demo showcasing redundancy.
The Sprint 3 demo will show lane keeping with safety redundancy and at least one infrastructure response behavior. The emphasis will be repeatable operation, not maximum feature count. Logs and evidence will be saved for documentation. This demo proves the system is now multi-sensor and multi-feature capable.

Sprint 4 — Full Integration + Final Build + Final Performance (10 Tasks)

31) Run all sensors simultaneously and verify real-time stability.
Sprint 4 begins with full-system runtime testing. The team will confirm that sensors can run together without slowing perception or breaking control timing. Any unstable sensor stream will be isolated and corrected. This establishes the starting point for final performance tuning.

32) Implement sensor fusion logic for graceful degradation.
The system will be designed to keep autonomy working when one sensor becomes unreliable. Confidence rules will decide when to trust vision and when to rely on safety sensing. The goal is stable behavior under imperfect conditions, not perfect detection. This directly reflects the redundancy principle described in your approach.

33) Finalize the smart infrastructure track for the final scenario.
The final environment layout will be locked with consistent lanes, signs, traffic light visibility, and obstacle zones. The team will remove unnecessary complexity that increases failure risk. The track must support a clear "story" for the final demo. This makes testing and demonstrations consistent.

34) Validate a complete end-to-end run repeatedly.
The vehicle will complete full runs that include lane keeping, stopping for lights/signs, and obstacle safety response. Testing will be repeated enough times to show reliability, not luck. Any failure will be traced to its source and fixed with minimal changes. This builds confidence in final performance.

35) Tune the system for smoothness and clean behavior.
Control tuning will focus on smooth steering, stable speed changes, and predictable stops. Perception thresholds will be adjusted to reduce false triggers. Planning transitions will be refined to avoid rapid state switching. The result should look like a controlled autonomous system, not a prototype.

36) Lock the final chassis build and verify mechanical reliability.
All mounts, wiring, and battery placement will be secured for the final version. The vehicle will be inspected after test runs to confirm no loosening or shifting occurs. Mechanical reliability will be treated as essential because it directly affects perception stability. This prevents last-minute hardware failures.

37) Confirm safety behavior in failure scenarios.
The team will test how the vehicle reacts when vision confidence drops or sensor readings

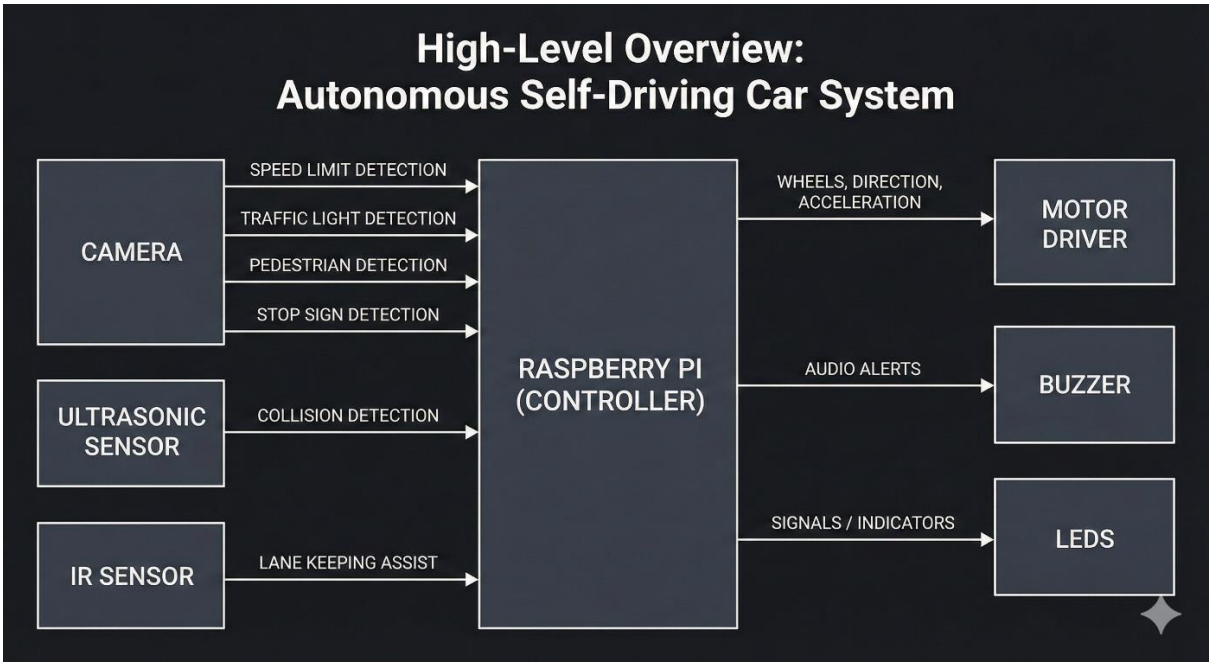
stop updating. The correct response should be a safe stop rather than unpredictable driving. Emergency stop must always be available and immediate. This provides a strong safety argument in final evaluation.

38) Prepare the final demo and a reduced-scope backup demo.
The final demo route will be planned to show key behaviors clearly. A fallback demo will be prepared that still demonstrates full integration if one feature becomes unstable. The team will rehearse both paths to ensure timing and reliability. This guarantees a successful demonstration day.

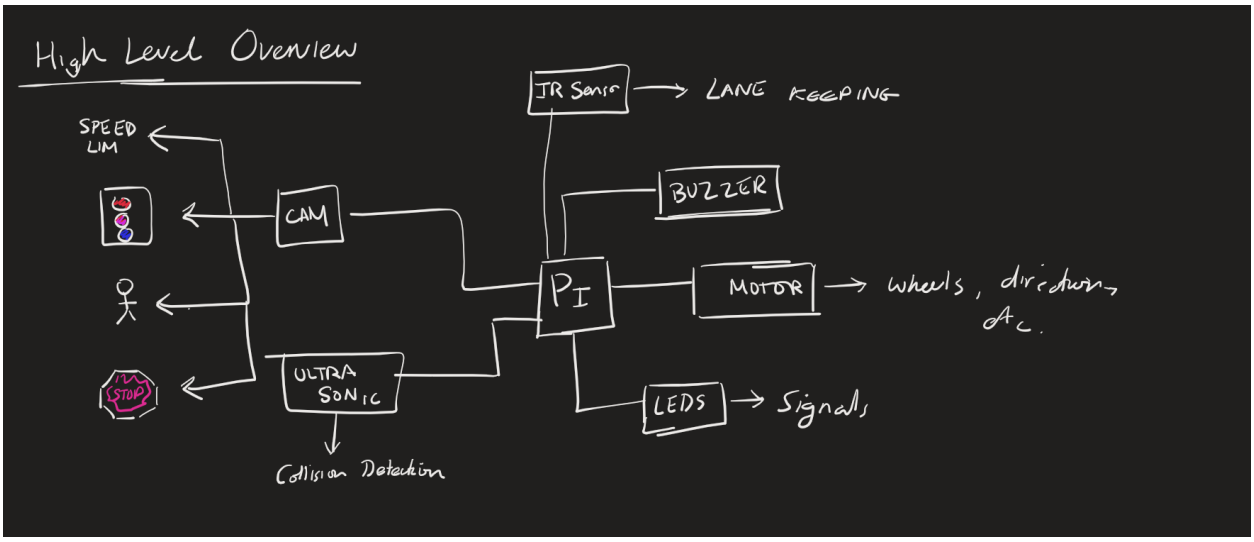
39) Finalize documentation and open-source resource reporting.
All open-source resources will be cited with clear descriptions of what each contributed. The documentation will focus on implemented and validated results rather than future claims. Sprint outcomes will be summarized to show progression and engineering decisions. This supports grading requirements and professional presentation.

40) Complete final readiness review and submission packaging.
A final system checklist will confirm power stability, sensor operation, safety override behavior, and repeatable autonomy. The team will complete a full rehearsal run and lock the system configuration. Any last changes will be limited to low-risk adjustments. This closes the project with a stable integrated deliverable.

Block Diagram — Rough Draft



The diagram above is generated by Google Gemini from a written draft. This written draft is attached below:



GitHub

View 1 · Academic Weapons Kanbaan View

Note that the GitHub repository is only initialized and can be visited from the link above. The tasks within the Kanban View are not allocated due to the uncertainty of the timeline. Tasks will be added as the group meets and delegates different elements of the project to group members.