Feasibility:

For a limited engineering staff, Starting from scratch, developing an ISO 26262-compliant automotive system is genuinely feasible by targeting ASIL B/C (not D), using the TMS570LS3137 microcontroller with its free development tools (Code Composer Studio and HALCoGen), skipping the expensive and complex AUTOSAR framework in favor of bare-metal code with FreeRTOS, leveraging PCBWay for affordable PCB assembly, and using off-the-shelf IP67 enclosures with standard automotive connectors. The entire process—from requirements through ready-to-install hardware—takes **6-9 months** with costs under **$7500** for prototypes, made possible by TI's free production-ready toolchain, pre-certified safety MCU features (lockstep cores, BIST, ECC), and the availability of open-source safety-critical software libraries. The key is maintaining rigorous documentation throughout, following MISRA C coding standards. Commercial deployment will likely require a third-party safety assessment, if that is desired, but the technical work is absolutely achievable solo with modern tools and properly scoped safety requirements.

Using Claud AI, it may be able to cut that time by 30%

Recommended Development Path

Phase 1: Requirements & Safety Analysis (2-4 weeks)

Document Requirements

Use free or affordable tools like Visure Requirements ALM (offers trial) or even structured Excel/Word templates

Perform Hazard Analysis and Risk Assessment (HARA) to determine ASIL level based on Severity, Exposure, and Controllability

Target ASIL B or C maximum for single-engineer feasibility (ASIL D requires extensive organizational infrastructure)

Safety Concept

Define safety goals and derive safety requirements allocated to architectural components

Consider ASIL decomposition to reduce requirements where possible

Phase 2: Hardware Selection (1-2 weeks)

Recommended MCU: TMS570LS3137 (the sweet spot for single engineers)

Why this choice:

 ISO 26262 ASIL D certified and IEC 61508 SIL 3 certified

Available on LaunchPad development kit for £18-20

Code Composer Studio (CCS) is free to use all the way through to production

Electronic Products

 Built-in safety features (lockstep cores, ECC, BIST)

 Good availability through distributors for PCBway assembly

Enclosure Selection:

 IP67 enclosures provide complete dust protection and can handle temporary submersion in water up to 1 meter deep for 30 minutes

Phase 3: Software Architecture (2-3 weeks)

Skip AUTOSAR. Here's why:

 AUTOSAR vendors license their software for specific model years, and using it for different models incurs additional expenses, with some of the highest off-the-shelf software costs available

Tailored bare-metal software is less complex, which is crucial during verification for Functional Safety, and is easier to verify, reducing associated costs

While AUTOSAR specification is open for in-house development, in terms of time and money it is often more cost-effective to buy ready-made modules and customize them

Instead, use:

 HALCoGen (free TI tool) for hardware initialization

 FreeRTOS (free, open-source) with Safety Critical version available

 Simple layered architecture: Application → HAL → Drivers

 Follow V-model methodology for software development with early testing at each step to find bugs and faults at very early stages

Phase 4: Development Environment Setup (1 week)

Core Tools (All Free):

Code Composer Studio - TI's Eclipse-based IDE

HALCoGen - Hardware initialization code generator

Git/GitHub - Version control

Static Analysis: Use free MISRA C checkers or trial versions of commercial tools

Coding Standards:

Follow MISRA C:2012 guidelines

Use static analysis tools to enforce compliance

Enforce coding standards like MISRA and AUTOSAR with static code analyzers to ensure safety and reliability, confirming compliance quickly and efficiently

Phase 5: Implementation (8-12 weeks)

Develop in iterations following V-model

Unit test each module with coverage tracking

Use TI's diagnostic libraries for self-test features

Document everything - ISO 26262 requires extensive documentation

Phase 6: PCB Design & Manufacturing (3-4 weeks)

Design:

Use KiCad (free) or Altium (if budget allows)

Include automotive-grade connectors (TE Connectivity, Molex)

Add protection circuits (TVS diodes, EMI filtering)

Design for EMC compliance (ground planes, proper routing)

Manufacturing:

Submit to PCBway with complete BOM

Confirm TMS570LS3137 availability before ordering

Request assembly service with inspection

Order 5-10 prototypes initially

Phase 7: Testing & Verification (4-6 weeks)

Required Testing:

Use static analysis, data and control flow analysis, unit testing, and application monitoring together with configurable test reports containing high levels of detail

Functional testing on hardware

Environmental testing (temperature, vibration if needed)

EMC testing (may need third-party lab)

Documentation:

Test reports with traceability to requirements

Verification matrices

Safety analysis (FMEA, FTA)

Phase 8: Enclosure Integration (1-2 weeks)

Mount PCB in enclosure

A VCU enclosure identical to the one used for Gtake systems is available on Ali express for less than $100.

Phase 9: Compliance Documentation (2-3 weeks)

ISO 26262 requires confirmation measures including Confirmation Review for work products, Audits for implemented processes, and Assessment for items or element characteristics

Critical Success Factors:

Start with ASIL B - manageable for single engineer

Leverage free tools - CCS, HALCoGen, FreeRTOS Safety Critical

Skip AUTOSAR - too complex and expensive

Use proven reference designs - TI provides LaunchPad schematics

Document continuously - don't leave it to the end

Plan for third-party assessment - budget for external safety audit if selling commercially

This approach balances cost-effectiveness with standards compliance while remaining feasible.

Time Savings with Claude AI: 2-3 months (30% reduction)

Where Claude Helps Significantly (High Confidence):

1. **Initial Code Scaffolding (70% faster)** - Claude can generate driver templates, HAL interfaces, and peripheral initialization code from HALCoGen outputs in minutes vs. hours
2. **MISRA C Compliance (50% faster)** - Claude can write MISRA-compliant code from the start and refactor existing code to meet standards
3. **Unit Test Generation (60% faster)** - Claude excels at generating comprehensive unit tests with edge cases
4. **Documentation (80% faster)** - Claude can generate function headers, API documentation, and safety manual content
5. **Code Review & Refactoring (40% faster)** - Claude can identify potential issues and suggest improvements

Where Claude Has Limited Impact (Be Realistic):

1. **Requirements Analysis** - Still needs human expertise; Claude can help format but not replace domain knowledge

2. **Safety Architecture** - Requires deep automotive safety knowledge; Claude can assist but not lead
3. **Hardware Testing & Debugging** - Physical testing cannot be accelerated by AI
4. **Formal Verification** - While research shows LLMs can generate formally correct code when combined with verification tools, this is still emerging technology **Tool**
5. **Qualification** - ISO 26262 requires tool qualification to ensure development tools support safety goals, and AI-generated code adds complexity here

The Risks:

Recent research on bugs in large language model generated code shows reliability concerns and **ISO 26262 explicitly requires traceability from requirements through implementation**. Every line of code must be:

- Traceable to requirements
- Verified through defined processes
- Part of documented V-model development

AI-generated code introduces risks:

- You must **thoroughly review and understand** every line Claude writes
- You're legally responsible for all code in safety-critical systems
- Current ISO 26262 frameworks have gaps in addressing AI-specific safety requirements, though enhancements for ML-based systems are being developed for ASIL A/B
- Using AI for code generation may require additional justification during safety audits

Breakdown of savings:

- Requirements & Safety: 3 weeks → 2.5 weeks (minor help)
- Software Architecture: 2.5 weeks → 2 weeks (documentation help)
- Implementation and iterations: 10 weeks → **6 weeks** (major savings here)
- Testing & Verification: 5 weeks → 4 weeks (test generation)
    - Documentation: 2.5 weeks → **1 week** (major savings)

- Everything else: Same (no AI acceleration)

Best Practices for Using Claude:

1. **Use Claude for boilerplate**, not critical safety logic
2. **Always manually verify** AI-generated code line-by-line
3. **Document that you reviewed** AI output (for audit trail)
4. **Use Claude for MISRA compliance checking** - this is highly effective
5. **Let Claude generate test cases** - excellent for coverage
6. **Never blindly trust** safety-critical functions from AI

Bottom line: Claude can realistically save you **2-3 months** of a 6-9 month project, but you must remain the safety engineer who understands, validates, and takes responsibility for every design decision and line of code.

Deliverables:

Safety Manual

Technical Safety Concept

Software Safety Requirements

Verification Reports

Traceability matrices

Total Timeline: 6-9 months

Cost Breakdown (USD):

Development kit: $20-50

PCB prototypes (10 units): $500-1,000

Components for assembly: $200-500

Enclosures (10 units): $200-500

Testing equipment: $500-1,000 (if not already available)

Static analysis tools (annual): $0-2,000 (use free trials initially)

Total: $1,500-5,000 (excluding labor)