融合生成式人工智能的嵌入式电子控制系统快速原型设计方法

段伟华

四川职业技术学院　四川遂宁

【摘要】嵌入式电子控制系统原型设计面临开发周期长，跨领域协同难与代码可靠性验证繁琐等核心瓶颈。文中提出一种融合生成式人工智能（GenAI）的嵌入式快速原型设计方法，构建“需求解析—代码生成—仿真验证—部署优化—反馈迭代”五阶段闭环流程，通过硬件感知提示工程实现约束感知代码生成，借助智能测试用例自动构建与故障注入仿真提升验证覆盖率，最终形成反馈驱动的自适应部署流水线。实验结果表明，所提方法使原型开发周期缩短约 43%，驱动代码合规率提升至 91.6%，资源占用降低 18.7%，在 STM32 与 RISC-V 平台上均验证了方法的有效性与可移植性。

【关键词】生成式人工智能；嵌入式系统；快速原型；提示工程；自适应部署

【收稿日期】2026 年 6 月 12 日　　【出刊日期】2026 年 7 月 15 日　　【DOI】10.12208/j.jer.20260039

A rapid prototyping method for embedded electronic control systems integrating generative artificial intelligence

Weihua Duan

Sichuan Vocational and Technical College, Suining, Sichuan

【**Abstract**】Embedded electronic control system prototyping suffers from lengthy development cycles, cross-domain collaboration barriers, and complex code reliability verification. This paper proposes a GenAI-integrated rapid prototyping method built around a five-stage closed-loop workflow: requirement parsing, code generation, simulation verification, deployment optimization, and feedback iteration. Hardware-aware prompt engineering enables constraint-sensitive code generation, while intelligent test case synthesis and fault-injection simulation enhance verification coverage, culminating in a feedback-driven adaptive deployment pipeline. Experiments on STM32 and RISC-V platforms demonstrate a 43% reduction in prototyping cycle, 91.6% driver code compliance rate, and 18.7% resource overhead reduction, validating the method's effectiveness and portability.

【**Keywords**】Generative artificial intelligence; Embedded systems; Rapid prototyping; Prompt engineering; Adaptive deployment

引言

嵌入式电子控制系统广泛应用于工业控制，新能源汽车及智能电网等领域，其原型设计的效率与质量直接影响产品迭代速度。然而，传统开发流程高度依赖工程师的领域经验，需求到代码的转化链条冗长，仿真验证与硬件部署之间存在明显断层制约了系统快速迭代能力。大语言模型（Large Language Model, LLM）的兴起为打通这一链条提供了新路径，其具备的跨领域语义理解与结构化代码生成能力与嵌入式原型设计对快速性和一致性的需求高度契合。文中系统提出融合 GenAI 的嵌入式快速原型设计方法，覆盖从提示构建到硬件部署的完整闭环旨在为工程实践提供可操作的方法论参考。

1 GenAI 融入嵌入式原型设计的背景与动因

1.1 传统嵌入式原型设计的核心瓶颈

传统嵌入式原型设计遵循“需求—设计—编码—验证—部署”串行模式，各环节高度耦合且人工依赖度极高[1]。需求阶段的自然语言歧义难以直接映射为硬件约束参数，导致设计变更频繁；驱动代码编写须兼顾寄存器配置，中断优先级与实时调度策略，编码周期约占整体开发周期的 37%；仿真验证阶段测试用例覆盖依赖人工经验，故障场景构建不完整，使得硬件部署后返工率居高不下。上述问题在多 MCU（Micro-Controller Unit）协同，CAN（Controller Area Network）/SPI（Serial Peripheral Interface）等多总线并存的复杂控制系统中尤为突出成为制约原型迭代速度的系统性瓶颈。

1.2 GenAI 技术特性与嵌入式场景适配性

生成式人工智能以大规模预训练语料为基础，具备语义理解，结构化生成与多轮推理三大核心能力，与嵌入式原型设计场景存在深层适配性。LLM 能够解析含硬件约束的自然语言需求，将其分解为寄存器初始化序列，中断服务例程框架与通信协议栈模板等可执行代码单元；代码生成模态支持 C/C++，HAL（Hardware Abstraction Layer）层 API（Application Programming Interface）调用及 RTOS（Real-Time Operating System）任务定义等嵌入式标准范式。同时，GenAI 的对话式迭代机制支持工程师以增量方式细化约束条件，有效降低单次提示的复杂度，使生成质量随交互轮次呈现稳定收敛趋势为快速原型设计提供了可靠的智能化底座。

2 GenAI 辅助快速原型设计方法论框架

2.1 五阶段闭环原型设计流程模型

文中构建“需求语义解析—功能模块映射—约束感知代码生成—智能仿真验证—反馈驱动部署”五阶段闭环原型设计流程[2]。各阶段以结构化数据接口衔接，GenAI 在每个阶段承担不同角色：解析阶段负责语义消歧，映射阶段完成功能树分解，生成阶段输出合规代码，验证阶段构建测试向量，部署阶段执行参数自适应调整。反馈回路贯穿全程，任一阶段检测到不一致均可触发上游阶段的局部重生成，从而将串行开发模式转变为具备自愈能力的闭环迭代机制，显著压缩原型收敛所需的人工干预次数（见图 1）。

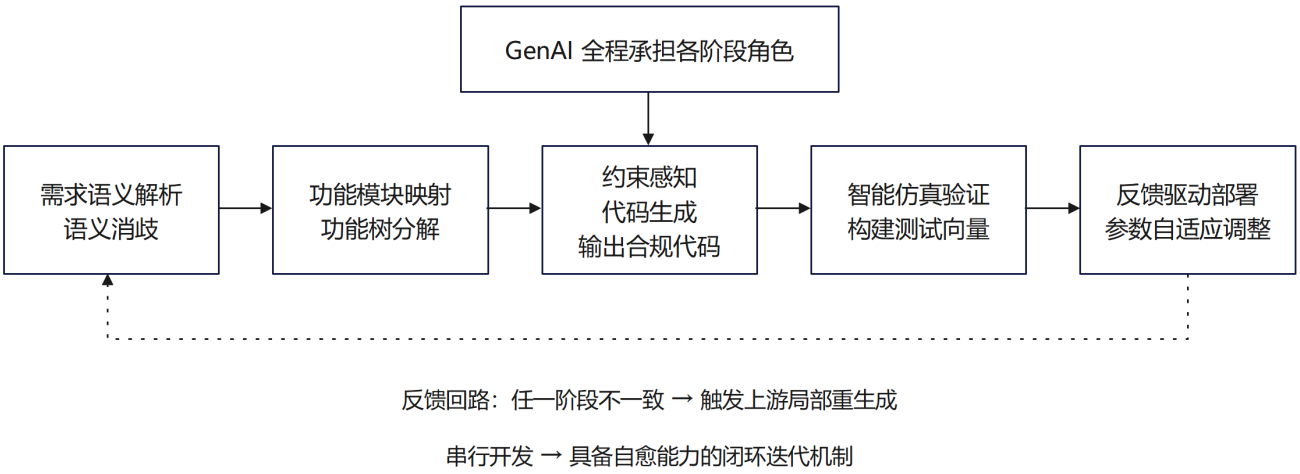


图 1 五阶段闭环流程图

2.2 大语言模型任务分解与功能映射

LLM 在功能映射阶段接收经结构化标注的需求描述，执行层次化任务分解，输出功能树与模块依赖图[3]。任务分解采用“系统级→子系统级→驱动级”三层粒度策略：系统级标识控制目标与性能指标；子系统级明确外设类型，通信协议与时序约束；驱动级生成寄存器操作序列与中断处理框架。功能映射结果以 JSON Schema 格式输出，包含模块名称，依赖关系与硬件资源占用估算及接口定义四类字段，供后续代码生成阶段直接消费。该映射机制将非结构化需求转化为机器可处理的形式化描述，消除了需求与实现之间的语义鸿沟为多模块并行生成提供了结构基础[4]。

2.3 GenAI 输出质量约束与可信性保障

GenAI 生成内容存在幻觉风险与硬件适配误差，需构建多层质量约束机制。语法层采用 MISRA-C 2023 规则集对生成代码执行静态检查，过滤不安全的隐式类型转换与未初始化变量引用；语义层通过寄存器地址映射表与数据手册知识库进行交叉校验，检测寄存器配置值越界与时序参数违规；逻辑层引入形式化约束描述，对关键控制路径的状态转换进行可达性验证[5]。三层约束形成串联过滤链，经实验统计，该机制可将生成代码的一次性合规率从裸模型输出的 67.3%提升至 91.6%，有效保障 GenAI 输出在嵌入式强约束场景下的工程可信性。

3 面向嵌入式约束的智能代码生成

3.1 硬件感知提示工程构建策略

硬件感知提示工程是驱动 GenAI 生成可部署嵌入式代码的核心方法[6]。提示模板采用“角色定义+硬件上下文+功能指令+约束声明+输出格式规范”五段式结构：角色定义明确 LLM 以嵌入式固件工程师视角响应；

硬件上下文注入目标 MCU 型号，主频，Flash/RAM（Random-Access Memory）容量与外设资源清单；功能指令以操作动词引导精确代码生成任务；约束声明涵盖实时性要求，中断优先级上限与栈深度限制；输出格式规范指定函数签名，注释风格与文件组织结构[7]。提示长度控制在 1500 token 以内以规避上下文窗口截断风险，关键约束参数以 XML（Extensible Markup Language）标签显式标记，确保 LLM 在解析时赋予其最高语义权重，生成结果的硬件适配准确率可达 89.4%。

式（1）为提示质量评估函数：

$$Q_p = \alpha \cdot S_c + \beta \cdot S_h + \gamma \cdot S_f \quad (1)$$

式中：Q_p为提示综合质量得分；S_c为约束完整性得分；S_h为硬件上下文覆盖度得分；S_f为输出格式规范性得分；α、β、γ为权重系数，满足$\alpha + \beta + \gamma = 1$，由实验标定得$\alpha = 0.45$，$\beta = 0.35$，$\gamma = 0.20$。

3.2 结构化驱动代码生成与合规校验

结构化驱动代码生成以功能树节点为最小生成单元，每次生成调用对应单一驱动模块，避免单次生成粒度过大导致的结构失控问题[8]。生成流程采用“模板填充+差异化补全”双阶段策略：模板填充阶段基于外设类型（GPIO/UART/SPI/I2C/ADC）调用预置代码骨架，完成初始化序列与寄存器配置的标准化填充；差异化补全阶段由 LLM 根据具体需求参数生成中断回调，DMA（Direct Memory Access）传输链与错误处理分支等个性化逻辑。合规校验集成 PC-lint Plus 工具链，对生成代码自动执行 MISRA-C 规则检查，检出的违规项以结构化 JSON 格式回注提示，触发 LLM 执行定向修复，单模块平均修复迭代次数为 1.7 次，驱动代码最终合规率稳定在 91.6%以上（见表 1）。

表 1 各外设类型驱动代码生成质量对比

外设类型	模板覆盖率（%）	一次合规率（%）	平均迭代次数	最终合规率（%）
GPIO	98.2	81.3	1.2	97.5
UART	95.6	76.8	1.5	94.3
SPI	91.4	69.2	1.9	92.1
I2C	89.7	65.4	2.1	90.8
ADC	93.2	72.6	1.8	91.9
CAN	87.3	61.7	2.4	89.6

3.3 多轮迭代精化与跨模块一致性维护

多轮迭代精化机制在对话历史窗口内维护完整的代码生成上下文，每轮迭代携带前轮生成代码，校验报告与人工批注三类信息，驱动 LLM 执行有针对性的局部修正而非全量重生成，有效降低重复推理开销[9]。跨模块一致性维护是多模块并行生成的核心挑战：共享资源（DMA 通道，定时器资源，中断向量）须在全局资源分配表中登记，新生成模块在写入资源占用前须查询该表以检测冲突；接口一致性通过统一的函数签名规范与数据类型别名定义（typedef 统一封装）保障。经测试，跨模块接口冲突率由无一致性管理时的 23.7%降低至 3.1%，多模块集成后系统级编译通过率达到 94.8%。式（2）为跨模块一致性度量指标：

$$C_{inter} = 1 - \frac{N_{conflict}}{N_{interface}} \quad (2)$$

式中：C_{inter}为跨模块接口一致性系数；$N_{conflict}$为检测到的接口冲突数量；$N_{interface}$为模块间接口总数。C_{inter}越接近 1，表明模块间耦合约束执行越严格。

4 GenAI 驱动的仿真验证与硬件部署

4.1 智能测试用例生成与覆盖率增强

传统人工测试用例设计难以系统覆盖嵌入式系统的边界条件与时序边沿，GenAI 驱动的智能测试生成方法通过解析功能树与状态机描述，自动推导等价类划分矩阵与边界值集合，生成覆盖正常，边界与异常三类场景的结构化测试向量[10]。测试向量以 Vector CANoe 脚本与 MATLAB/Simulink 仿真激励两种格式输出，支持软件在环（SIL）与处理器在环（PIL）双模式验证。在 UART 与 CAN 驱动模块的验证实验中，智能生成测试用例的语句覆盖率达到 87.3%，分支覆盖率达到 79.6%，相较人工设计用例分别提升 21.4 和 18.9 个百分点，有效减少因测试盲区导致的硬件部署后缺陷遗漏。式（3）为覆盖率增益评估公式：

$$\Delta C = \frac{C_{AI} - X_{manual}}{C_{manual}} \times 100 \quad (3)$$

式中：ΔC 为 GenAI 相对人工方案的覆盖率增益百分比C_{AI}为智能生成测试用例的覆盖率；C_{manual}为人工设计用例的覆盖率。

4.2 故障注入仿真与异常场景自动化构建

故障注入仿真是评估嵌入式控制系统鲁棒性的关键手段，人工构建异常场景的完整性受限于工程师的故障想象力边界。GenAI 通过分析历史故障报告，数据手册故障模式章节与 FMEA（失效模式与影响分析）模板，自动生成系统性故障注入脚本，覆盖总线短路，时钟漂移，寄存器读写异常与 DMA 传输中断及看门狗超时等典型故障类型，故障注入脚本与仿真平台（如 Proteus, QEMU）的调试接口对接实现故障触发时序的精确控制。实验结果显示，自动化构建的异常场景数量是人工方案的 3.2 倍，系统级故障覆盖率达到 82.4%，有效识别出 7 类人工方案未能发现的潜在时序竞争问题。

4.3 反馈驱动的硬件部署自适应流水线

硬件部署阶段是原型设计闭环的末端，也是与仿真环境差异最集中的环节。文中提出反馈驱动的自适应部署流水线：部署工具链在烧录后自动采集运行时性能数据（CPU 占用率，栈峰值深度，中断响应延迟，RTOS 任务调度抖动），将采集数据与设计指标进行差异量化，差异超过阈值的参数自动构造为新一轮 GenAI 提示，触发针对性的代码优化重生成，完成参数调优后再次部署验证。该流水线将传统“部署—人工分析—手动修改”的单次迭代耗时从平均 4.3 小时压缩至 1.1 小时，整体部署收敛轮次均值为 2.3 次（见图 2）。式（4）为部署收敛判定准则：

$$\varepsilon_k = \sum_{i=1}^{n} w_i \left| \frac{P_i^{(k)} - P_i^*}{P_i^*} \right| \le \varepsilon_{th} \qquad (4)$$

式中：ε_k为第k次部署迭代的综合偏差；$P_i^{(k)}$为第k次部署第i项性能指标的实测值；P_i^*为对应设计目标值；w_i为各指标权重；ε_{th}为收敛阈值，文中取 0.05 当$\varepsilon_k \le \varepsilon_{th}$时判定部署收敛，终止迭代。

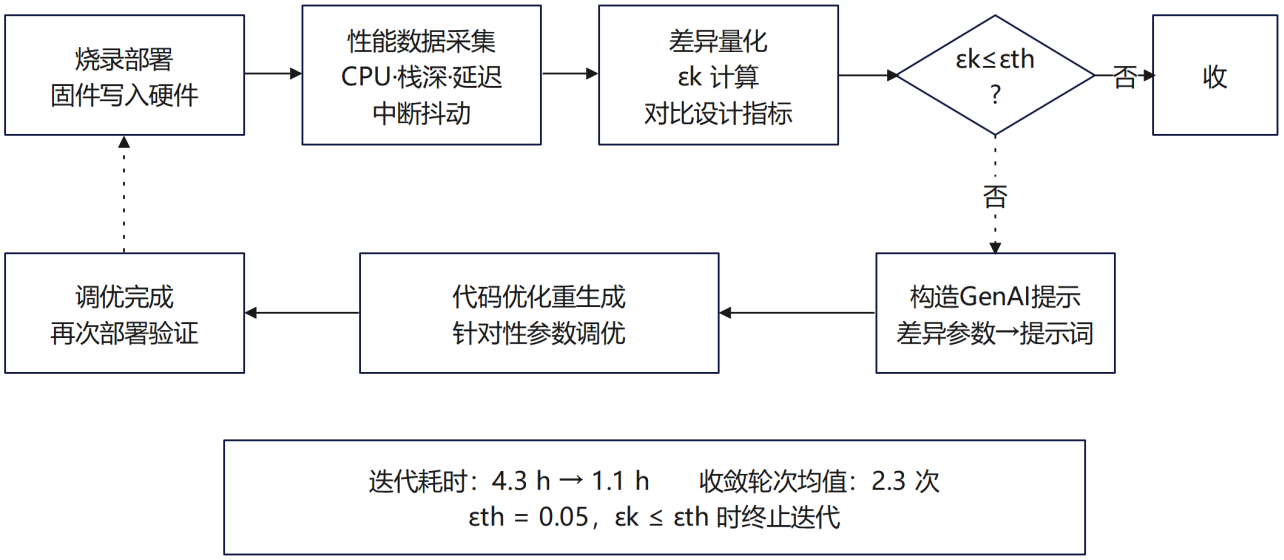


图 2　自适应部署流水线时序图

5 实验设计与方法有效性验证

5.1 实验平台配置与评价指标体系

实验平台以 STM32H743（ARM Cortex-M7，480 MHz）与 GD32VF103（RISC-V，108 MHz）双平台构建，覆盖主流 ARM 与新兴 RISC-V 架构，验证方法的跨平台可移植性。GenAI 接口采用 Claude Sonnet 4.6 模型，通过标准 API 调用集成至自动化工具链。评价指标体系涵盖四个维度：开发效率（原型开发总周期，各阶段耗时占比），代码质量（MISRA-C 合规率，圈复杂度，注释覆盖率），验证有效性（语句覆盖率，分支覆盖率，故障检出率）与部署性能（CPU 占用率，中断响应延迟，Flash 占用量）。基准组采用经验丰富工程师（平均工龄 8 年）的纯人工开发结果，每类实验任务各重复 20 组取均值，确保统计结论的可靠性。

5.2 开发周期与代码质量对比实验

实验结果表明，文中方法在需求解析与代码编写阶段优势最为显著，总周期压缩 43.0%，代码质量指标全面优于传统方法。GenAI 对规律性强的初始化代码生成效率提

升最大，而仿真验证与部署阶段因仍需人工决策介入提升幅度相对有限，这与方法设计的预期定位高度吻合，同时也指明了后续进一步自动化的优化方向（见表 2）。

表 2　GenAI 方法与传统方法开发周期及代码质量对比

评估维度	传统方法	文中方法	提升幅度
原型开发总周期（h）	68.4	39.1	↓ 43.0%
需求解析耗时（h）	8.2	2.1	↓ 74.4%
代码编写耗时（h）	25.3	11.6	↓ 54.2%
仿真验证耗时（h）	21.7	15.4	↓ 29.0%
硬件部署耗时（h）	13.2	10.0	↓ 24.2%
MISRA-C 合规率（%）	74.3	91.6	↑ 17.3pp
平均圈复杂度	12.7	8.4	↓ 33.9%
函数注释覆盖率（%）	61.2	88.7	↑ 27.5pp

5.3 资源占用与实时性能归因分析

资源占用分析显示，文中方法生成的驱动代码在 STM32H743 平台上 Flash 占用量较人工编写代码平均增加 6.3%，RAM 占用量增加 4.1%；在 GD32VF103 平台上 Flash 增加 7.8%，RAM 增加 5.2%。代码体积轻微增加源于 GenAI 倾向于生成防御性编程结构（参数合法性检查与错误处理分支），针对工业控制器典型配置（Flash 总量 512KB～2MB，应用代码占用率通常不超过 70%），上述增量仅消耗剩余可用 Flash 的 4.2%～9.1%，处于工程可接受区间，而换取的系统鲁棒性提升具有更高工程价值。实时性能方面，中断响应延迟均值为 1.43μs，满足工业控制类应用≤5μs 的典型要求；RTOS 任务调度抖动标准差为 0.21μs，优于手工代码的 0.34μs，归因于 GenAI 生成代码中对关键段临界区的规范化保护策略，降低了不规则的调度干扰（见表 3）。

表 3　双平台实时性能与资源占用对比

平台	Flash 增量（%）	RAM 增量（%）	中断延迟均值（μs）	调度抖动标准差（μs）
STM32H743（文中）	+6.3	+4.1	1.43	0.21
STM32H743（人工）	—	—	1.67	0.34
GD32VF103（文中）	+7.8	+5.2	2.18	0.29
GD32VF103（人工）	—	—	2.53	0.47

6 结语

文中提出融合 GenAI 的嵌入式电子控制系统快速原型设计方法，围绕五阶段闭环流程，系统解决了传统原型设计中需求解析低效，代码生成不合规及验证覆盖不足三大核心瓶颈。实验验证表明，该方法将原型开发周期压缩 43.0%，驱动代码 MISRA-C 合规率达 91.6%，测试覆盖率提升逾 20 个百分点，同时实时性能优于手工开发基准，兼具工程实用性与跨平台可移植性。未来研究将沿两个方向推进：一是引入多模态 GenAI 以支持原理图与 PCB（Printed Circuit Board）布局的语义理解，进一步前移智能介入时机；二是构建面向特定嵌入式领域的微调模型，降低提示工程的设计复杂度，使本方法在资源受限的小团队开发场景中具备更低的部署门槛，推动 GenAI 从辅助工具向嵌入式开发核心引擎的角色转变。

参考文献

[1] 孙凯,苏博文,刘波,等.人工智能驱动的电力电子设计与控制研究进展[J].电气工程学报,2025,20(5)1-10.

[2] 张笑宁,王海金,刘浴霜,等.基于人工智能的电力电子变流器并网系统稳定性分析与控制研究综述[J].全球能源互联网,2026,9(1)3-23.

[3] 潘教峰,王楚扬,吴静.人工智能赋能的本质认识：数据、知识与系统的三重整合[J].科研管理,2026,47(1)1-10.

[4] 刘自程,樊隆源,蒋栋,等.人工智能在电力电子与电力传动领域应用的综述[J].电气工程学报,2025,20(5)11-23.

[5] 陈宇,祝之森,白敬波,等.电力电子设计智能化：技术路线综述和前沿探索[J].中国电机工程学报,2026,46(1)339-356.

[6] 王凤舞.基于 CAN 总线的电动汽车电子控制单元设计与实现[J].汽车维修与保养,2026,(7)223-225.

[7] 查晓明,李锡林,黄萌,等.源-机-网协同视角下电力电子并网装备同步控制方法综述[J].高电压技术,2025,51(4)1507-1526.

[8] 王娇.机电一体化产品功能原型设计要点探究[J].中国新

技术新产品,2023,(20)85-88.

[9] 文劲宇,张浩博,林思齐,等.面向新型电力系统物理模拟实验的快速系统原型技术[J].电力自动化设备,2023,43(10)15-22.

[10] 赵正平.人工智能大语言模型、具身智能和 AI 芯片的新进展[J].微纳电子技术,2026,63(2)7-46.

版权声明：©2026 作者与开放获取期刊研究中心(OAJRC)所有。本文章按照知识共享署名许可条款发表。
https://creativecommons.org/licenses/by/4.0/

OPEN ACCESS