


申请人	刘浩洋	部门	成都研发部
项目编码/名称	一种面向嵌入式软件的自动化插桩测试工具		
概述、技术要点	<p>一种面向嵌入式软件的自动化插桩测试工具是一项针对 C 语言项目的创新性自动化插桩技术。该发明解决了传统手动插桩效率低下、测试代码管理困难、跨文件复用性差等技术问题，通过双模式插桩机制、三级索引配置管理和时间戳日志追溯等核心技术，实现了桩代码与源代码的分离管理，显著提高了嵌入式软件测试的自动化水平和维护效率。</p> <p>该工具支持传统注释模式 and 创新的锚点分离模式，能够自动识别源代码中的插桩位置，从 YAML 配置文件中获取对应桩代码并精确插入，同时提供完整的备份机制和详细的执行追溯功能，适用于 C/C++嵌入式软件开发、单元测试、系统集成测试等多种应用场景。</p>		
检索词 (关键词)	嵌入式软件测试；自动化测试；C 语言插桩		
检索方式：	自行检索 <input checked="" type="checkbox"/> 委外检索 <input type="checkbox"/>		
相关检索结果	TA:(嵌入式软件测试) and TA:(自动化测试)  专利检索表.XLSX		

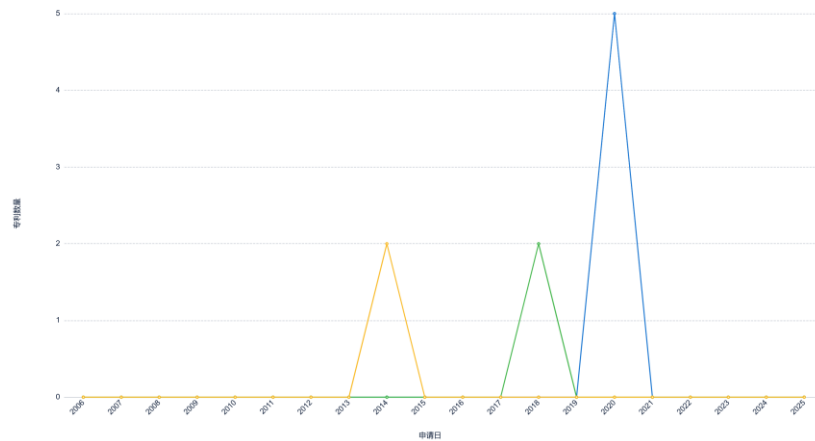
检索专
利与本
项目技
术的对
比分析
及结论

一、专利检索

检索式 1: TA:(嵌入式软件测试) and TA:(自动化测试)

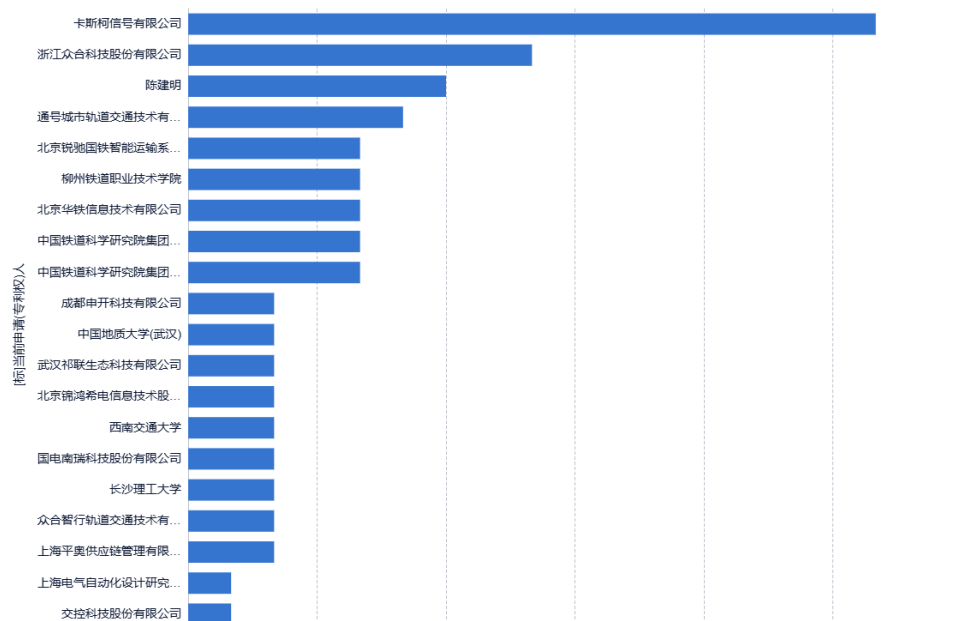
通过检索式 1, 检索出 23 项专利申请。分析情况如下:

[标]当前申请（专利权）人申请时间趋势

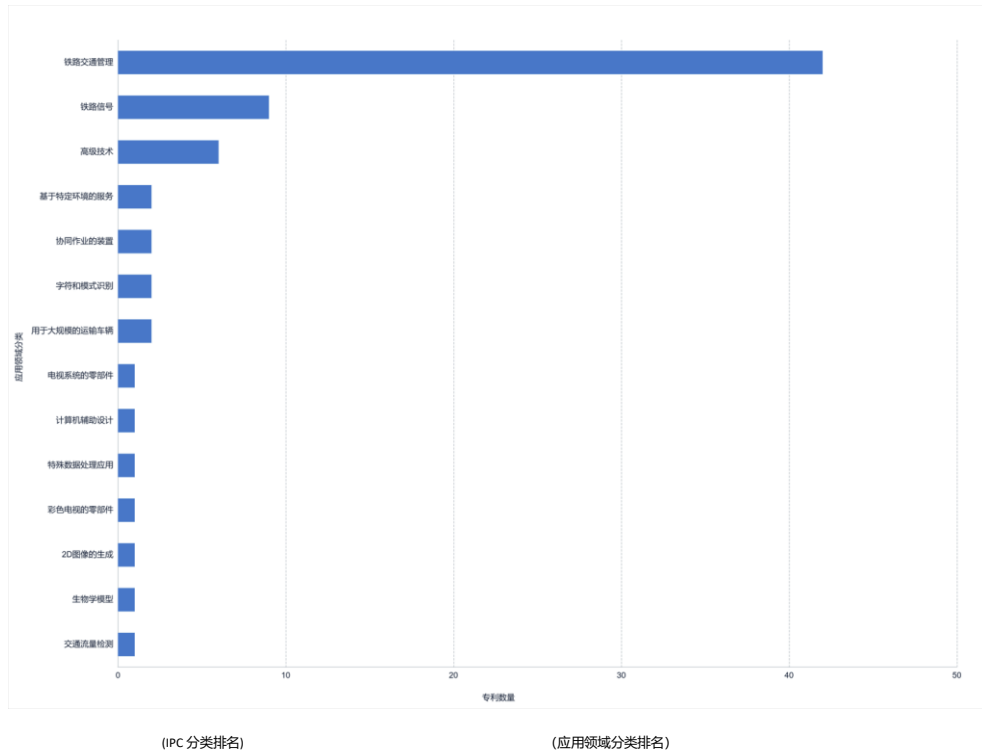


[标]当前申请（专利权）人专利数量排名

给您创建的图表提供一段描述...



[标]当前申请（专利权）人专利数量排名



通过检索式 1 检索到的专利见附件:



专利检索表.XLSX

所检测出的 23 项专利，均不直接涉及到本申请描述的嵌入式软件的插桩和自动化测试。现选取搜索结果中相关性最高的一篇进行如下详细分析：

1、《一种基于 TFTP 协议的嵌入式软件测试装置及方法》

该发明公开了一种基于 TFTP 协议的嵌入式软件测试装置及方法。该装置包括上位机和下位机，其中，上位机与下位机之间通过 TFTP 协议实现通信和消息传输，动态管理测试所需的桩代码。具体而言，上位机通过 TFTP 协议向下位机发送桩代码，下位机在接收到桩代码后进行解析，并在软件运行期间动态激活这些代码，以执行指定的测试任务。与传统技术相比，该发明通过动态加载与激活桩代码，

显著提升了测试的效率与自动化程度，具有结构简洁、操作简单和灵活性强等特点，适用于多种嵌入式软件测试场景。

本发明（YAMLWeave 自动化插桩工具） 专注于编译前的静态插桩管理，主要解决桩代码的自动化插入、配置管理和版本控制问题。其工作阶段处于开发测试准备期，通过智能锚点识别和 YAML 配置管理，实现桩代码的高效插入和统一管理。

该发明（自动开关桩工具） 则专注于运行时的动态桩控制，主要解决已插桩代码在实际执行过程中的智能开关和远程控制问题。其工作阶段处于运行时测试执行期，通过 UDP 协议等网络通信手段，实现对桩代码执行状态的实时动态控制。

技术领域的精确细分

虽然两项发明均属于软件测试技术领域，但在具体的技术细分领域存在明显差异：

- **本发明技术领域：**静态代码分析与自动化插桩技术，侧重于**源码级别**的代码注入和管理
- **该发明技术领域：**运行时测试控制与动态桩管理技术，侧重于**执行期**的桩功能开关和远程控制

核心技术手段的本质区别

插桩管理实现方式的根本不同：

1. **技术实现层面：**
 - 本发明：基于静态分析的文本处理和代码生成技术
 - 该发明：基于网络通信的运行时控制技术
2. **控制粒度差异：**
 - 本发明：文件级和项目级的批量插桩管理

○ 该发明：单个桩点或桩组的精细化实时控制

3. 交互方式差异：

- 本发明：通过配置文件和用户界面进行预设置
- 该发明：通过 UDP 协议进行实时网络通信控制

解决问题的差异化定位

针对的具体技术痛点差异：

- 本发明解决的痛点：
 - 手动插桩效率低下，工作量大
 - 桩代码分散管理，维护困难
 - 缺乏统一的插桩标准和工具支持
 - 版本控制和回滚机制不完善
- 该发明解决的痛点：
 - 插桩后代码的桩功能无法灵活控制
 - 大规模测试场景下桩开关管理复杂

创新维度的互补关系

两项发明在嵌入式软件测试的完整工作流程中形成**有机互补**的技术生态：

1. 工作流程互补：

- 本发明提供"插桩部署"阶段的自动化解解决方案
- 该发明提供"测试执行"阶段的智能控制解决方案

2. 技术栈互补：

- 本发明专注于编译时技术栈（静态分析、代码生成、配置管理）
- 该发明专注于运行时技术栈（网络通信、实时控制、动态管理）

3. 应用场景互补：

- 本发明适用于开发阶段的测试准备和环境搭建

○ 该发明适用于执行阶段的测试控制和结果调优

4. 创新价值互补:

- 本发明通过提高插桩效率和管理水平, 降低测试准备成本
- 该发明通过提升测试执行的灵活性和可控性, 提高测试质量和效率

技术协同效应

当两项发明配合使用时, 能够形成从静态部署到动态控制的完整测试解决方案:

1. **无缝衔接:** 本发明生成的插桩代码可直接支持该发明的动态控制协议
2. **标准统一:** 两项发明可共享统一的桩标识和管理规范
3. **效率倍增:** 静态高效部署 + 动态精确控制 = 测试效率的指数级提升
4. **场景扩展:** 支持从单机开发测试到分布式集成测试的全场景覆盖

综上所述, 两项发明在技术定位、解决问题、创新维度等方面各有侧重, 形成了嵌入式软件测试领域中**"插桩管理"**与**"桩控制"**的完整技术链条, 具有明显的差异化价值和协同效应。

二、期刊检索

检索式 1: 嵌入式软件测试 AND 自动化测试 AND C 语言插桩

通过检索式 1, 检索出 17 篇期刊论文, 见附件:

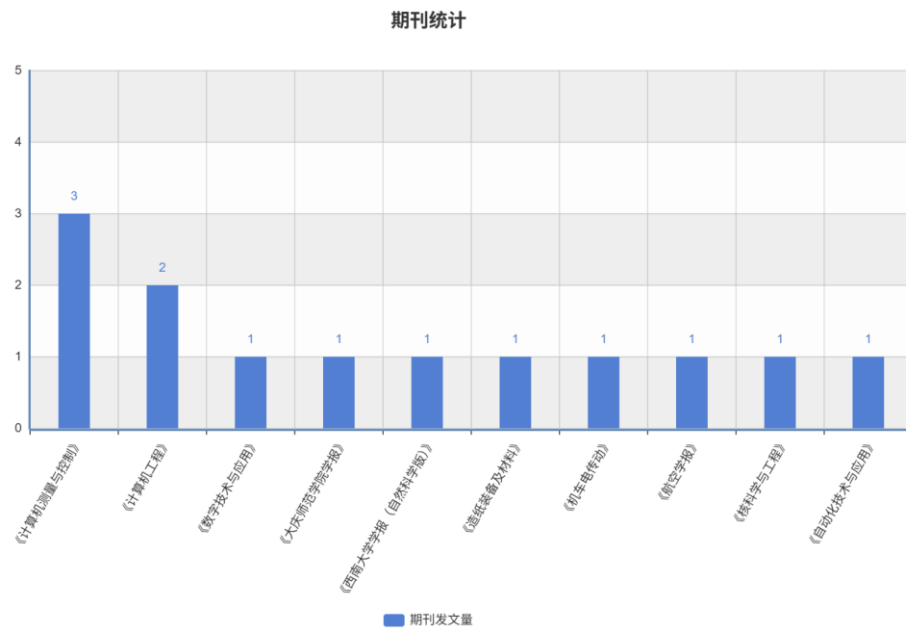


期刊检索表.xlsx

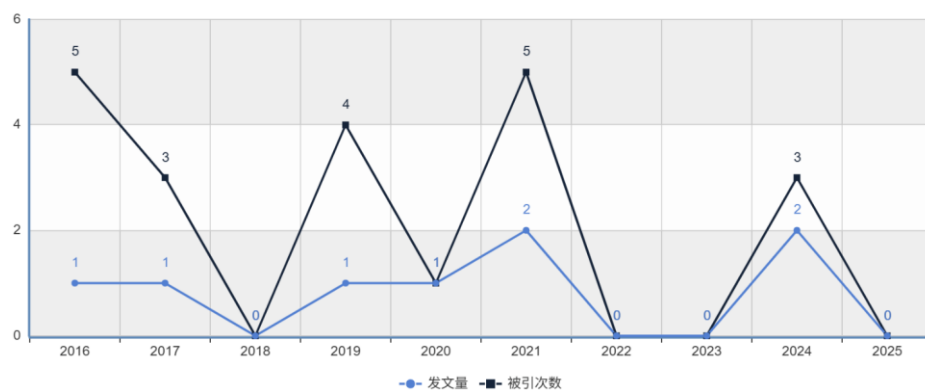
文章涉及主要学科统计:

序号	领域名称	发文量	主要研究主题
1	自动化与计算机技术	17	计算机网络 网络安全 神经网络 计算机网络
2	机械工程	1	汽车 发动机 故障诊断 轿车 电动汽车
3	交通运输工程	1	汽车 高速公路 施工技术 公路 船舶
4	航空宇航科学技术	1	飞机 无人机 航空发动机 航空 直升机

主要期刊统计分析:



近 10 年学术成果产出统计表



所检索的 17 篇期刊, 均没有本发明提及的 C 语言项目的在插桩部署阶段的自动
化解决方案。现选取一篇相关性最高的作详细对比分析:

1、《嵌入式航天软件汇编代码覆盖测试方案》

该发明从插桩策略、覆盖率收集方法 2 个方面设计实现了一种汇编代码覆盖测试方案,解决了嵌入式航天软件汇编代码覆盖测试没有工具支持的问题。针对嵌入式系统资源紧张、时序相关性强的特点,在研究汇编语言指令结构的基础上,制定了一套基于比特位表征方式的汇编插桩策略;提出了一种基于 JTAG 接口的通用覆盖率数据收集方案,解决了覆盖信息输出通道受限问题。实验结果及工程应用实例证明了方案的有效性和可行性

一、技术领域细分对比

该发明（汇编代码覆盖测试方案）

- 核心聚焦：汇编代码层面的覆盖测试
- 技术层次：底层汇编指令级别的代码分析
- 测试目标：验证汇编代码的执行覆盖率，确保测试完整性
- 应用场景：航天软件等对可靠性要求极高的关键系统

本发明 YAMLWeave（自动化插桩测试工具）

- 核心聚焦：C 语言源代码层面的功能测试插桩
- 技术层次：源代码级别的静态分析和代码生成
- 测试目标：在源码中自动插入调试、监控、验证等功能性桩代码
- 应用场景：嵌入式软件开发全生命周期的测试支持

细分差异：

- 该发明解决"测试是否充分"的问题
- 本发明解决"如何高效测试"的问题

- 两者在测试技术栈中处不同层次，形成垂直互补关系

二、技术手段区别

该发明的技术手段

汇编插桩策略 → 比特位标记 → JTAG 数据收集 → 覆盖率统计

- 插桩方式：比特位表征的汇编级插桩
- 数据收集：通过 JTAG 接口收集执行信息
- 分析目标：统计代码块执行次数和覆盖情况
- 资源优化：针对嵌入式资源限制的轻量级覆盖率方案

本发明 YAMLWeave 的技术手段

锚点识别 → YAML 配置查询 → 源码插入 → 功能验证

- 插桩方式：基于锚点标识的源码级插桩，支持双模式：

// 传统模式

// TC001 STEP1: 功能描述

// code: printf("调试信息");

// 分离模式

// TC001 STEP1 segment1 → 查询 YAML 获取桩代码

- 配置管理：三级索引 YAML 配置系统

TC001:

STEP1:

segment1: |

if (data < 0) {

```
printf("错误：数据异常 %d\n", data);
```

```
return ERROR_INVALID_DATA;
```

```
}
```

- 代码生成：智能识别插入点，保持代码格式和缩进
- 管理策略：时间戳目录管理，完整的执行历史追溯

核心区别：

- 插桩粒度：该发明在汇编指令级，本发明在 C 语言函数/语句级
- 插桩目的：该发明为覆盖率统计，本发明为功能测试辅助
- 数据处理：该发明重在数据收集分析，本发明重在代码管理维护

三、解决问题差异

该发明解决的核心问题

1. 测试充分性验证：确保汇编代码的执行路径被完全覆盖
2. 资源受限环境适配：在嵌入式系统资源紧张下实现覆盖测试
3. 时序敏感性处理：处理实时性要求严格的航天软件测试
4. 工具缺失问题：填补汇编代码覆盖测试工具的空白

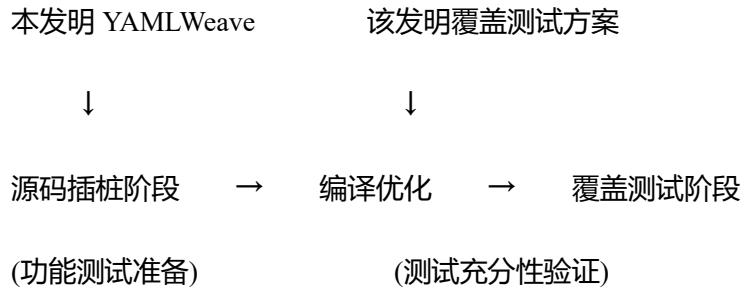
问题维度差异：

- 该发明关注测试质量保证（是否测试充分）
- 本发明关注测试效率提升（如何高效插桩）
- 该发明解决验证完整性问题
- 本发明解决开发生产力问题

四、创新点互补关系

技术互补性分析

1. 测试流程中的互补定位



2. 数据维度互补

- 本发明：提供功能正确性验证的桩代码

```
// 插入数据验证桩代码  
if (sensor_data < MIN_VALUE || sensor_data > MAX_VALUE) {  
    log_error("传感器数据异常: %d", sensor_data);  
    trigger_safe_mode();  
}
```

- 该发明：提供执行路径完整性验证

汇编指令块 A: 执行 3 次 ✓

汇编指令块 B: 执行 0 次 X (发现未覆盖路径)

分支覆盖率: 87.5%

3. 应用场景互补

联合应用示例：某航天飞控软件测试

1. 开发阶段：使用 YAMLWeave 插入功能验证桩代码

TC_FLIGHT_CONTROL:

STEP1:

```
altitude_check: |
```

```
  if (altitude < SAFE_ALTITUDE) {
```

```
    log_critical("危险高度: %fm", altitude);
```

```
    execute_emergency_protocol();
```

```
  }
```

2. 测试阶段: 使用该发明的覆盖测试验证测试充分性

关键控制算法覆盖率: 99.8% ✓

异常处理路径覆盖率: 95.2% ✓

中断服务程序覆盖率: 88.7% (需补充测试)

4. 技术创新维度互补

创新维度 本发明 YAMLWeave 该发明覆盖测试方案

工程效率 自动化插桩管理 自动化覆盖率收集

质量保证 功能正确性验证 测试完整性验证

资源优化 编译时优化 运行时轻量级监控

维护性 配置化管理 数据化分析

扩展性 插件式架构 多平台适配

协同价值最大化

两个发明的结合使用可以实现:

1. 完整的测试闭环:

o YAMLWeave 确保功能测试的高效实施

o 覆盖测试方案确保测试的充分性验证

	<p>总结</p> <p>这两个发明在嵌入式软件测试领域形成了技术栈垂直互补关系：</p> <ul style="list-style-type: none">• YAMLWeave：专注于"怎么测"的效率问题，通过自动化插桩管理提升测试开发效率• 该发明：专注于"测得够不够"的质量问题，通过覆盖率分析确保测试充分性 <p>两者结合可以构建完整的嵌入式软件测试解决方案，既保证了测试实施的高效性，又确保了测试验证的完整性，为嵌入式软件特别是航天等关键领域软件的质量保证提供了全面的技术支撑。</p>
项目 经理/检索 分析人 员 部门 负责人 审批：	<p>检索及分析人员签字：</p>