МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА МЕТОДА АВТОМАТИЗАЦИИ РАЗВЁРТЫВАНИЯ СПЕЦИАЛИЗИРОВАННОГО ОКРУЖЕНИЯ РАЗРАБОТЧИКА ОСРВ С ОСОБЫМИ ТРЕБОВАНИЯМИ К БЕЗОПАСНОСТИ В УСЛОВИЯХ ИЗОЛИРОВАННОЙ СРЕДЫ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Селезнёва Александра Дмитриевича

Научный руководитель
старший преподаватель _____ П. О. Дмитриев

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2026

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ОСРВ — Операционная система реального времени.

Air Gap — Изолированная среда, не имеющая физического или логического соединения с открытыми сетями.

IaC — Инфраструктура как код (Infrastructure as Code) — подход к управлению инфраструктурой, при котором конфигурация описывается декларативно в виде манифестов.

YAML — Язык разметки для конфигурационных файлов (Yet Another Markup Language).

CLI — Интерфейс командной строки (Command Line Interface).

Git — Распределённая система управления версиями.

CI/CD — Непрерывная интеграция и непрерывное развёртывание (Continuous Integration / Continuous Deployment).

JTAG/SWD — Аппаратные интерфейсы для отладки и программирования микроконтроллеров.

SHA-256 — Криптографическая хэш-функция для проверки целостности файлов.

GCC — Набор компиляторов GNU (GNU Compiler Collection).

GDB — Отладчик GNU (GNU Debugger).

QEMU — Эмулятор аппаратных платформ (Quick EMUlator).

CMake — Кроссплатформенная система автоматизации сборки.

Python — Высокоуровневый язык программирования общего назначения.

Ansible — Система управления конфигурациями и автоматизации развёртывания.

Keyring — Библиотека Python для безопасного хранения учётных данных в системном хранилище.

Docker — Платформа для контейнеризации приложений.

VM — Виртуальная машина (Virtual Machine).

APT — Пакетный менеджер дистрибутивов Debian и Ubuntu (Advanced Package Tool).

ВВЕДЕНИЕ

Актуальность темы. Современная разработка программного обеспечения для операционных систем реального времени (ОСРВ) предъявляет повышенные требования к инструментальному окружению разработчика. В отличие от универсальных пользовательских систем, ОСРВ требуют использования кросс-компиляторов, специфических цепочек сборки, эмуляторов аппаратных платформ и строгой согласованности версий всех компонентов. Даже незначительное расхождение в версии компилятора может привести к трудноуловимым ошибкам в финальном бинарном коде.

Особая сложность возникает при работе в изолированных средах (Air Gap) — сегментах сети без доступа к интернету. В таких условиях внешние репозитории (GitHub, PyPI, APT) недоступны, а традиционные методы развёртывания (виртуальные машины, контейнеры, пакетные менеджеры) либо избыточно тяжеловесны, либо не обеспечивают прямой работы с аппаратными отладчиками (JTAG/SWD), что критично для системной разработки.

Таким образом, существует потребность в формализованном алгоритме и инструменте, который позволяет описывать желаемое состояние среды декларативно и автономно развёртывать его в изолированной среде.

Цель работы — разработка алгоритма автоматизации развёртывания инструментального окружения разработчика ОСРВ в изолированных средах, его программная реализация и сравнительный анализ с существующими решениями.

Задачи работы:

1. проанализировать существующие методы автоматизации развёртывания и выявить их ограничения для Air Gap-сред;
2. определить требования к алгоритму и утилите для задач разработки ОСРВ в закрытом контуре;
3. разработать архитектуру и алгоритм работы специализированной утилиты;
4. реализовать программный прототип утилиты и провести его тестирование;
5. провести сравнительный анализ разработанного решения с альтернативными подходами.

Объект исследования — процессы развёртывания инструментального окружения разработчика ОСРВ.

Предмет исследования — методы и алгоритмы автоматизации развёртывания программных компонентов в изолированных средах (Air Gap) с особыми требованиями к безопасности.

Методы исследования. В работе использованы методы системного анализа, декларативного описания инфраструктуры (IaC), модульного тестирования и сравнительного анализа.

Практическая значимость. Разработанные утилиты `isp-up` и `isp-up-ansible` позволяют сократить время подготовки рабочего места разработчика в 3–4 раза по сравнению с ручной установкой, обеспечивают воспроизводимость окружения и могут быть внедрены в процесс сопровождения разработки систем реального времени.

Структура работы. Бакалаврская работа состоит из введения, четырнадцати разделов, заключения, списка источников и приложения. Разделы 1–5 посвящены анализу предметной области, особенностям развёртывания и существующим подходам. Разделы 6–7 формулируют требования и описывают подход к реализации. Разделы 8–9 представляют архитектуру разработанных утилит. Разделы 10–12 описывают тестирование, критерии и сравнительный анализ. Разделы 13–14 содержат ограничения, перспективы и заключение. В приложении приведены структуры проектов и листинги кода.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первом разделе дано определение инструментального окружения разработчика ОСРВ. Показано, что оно включает кросс-компиляторы, средства линковки, отладчики, эмуляторы аппаратных платформ и инструменты сборки. Наличие согласованного набора версий особенно важно, поскольку несовместимость элементов может приводить к сбоям при сборке.

Во втором разделе рассмотрены особенности развёртывания окружений для ОСРВ. Развёртывание окружения разработчика — это процесс установки, настройки и приведения программной среды к состоянию, необходимому для написания, сборки, отладки и тестирования целевого программного обеспечения. Описан типовой сценарий ручной установки из 20–30 шагов, который занимает 1–2 часа и содержит высокий риск ошибок.

Третий раздел посвящён особенностям развёртывания в изолированной среде (Air Gap). Сформулированы два ключевых требования: идемпотентность (повторный запуск не ломает уже настроенную среду) и воспроизводимость (одинаковое состояние на разных машинах).

В четвёртом и пятом разделах проведён анализ существующих подходов: виртуализации, контейнеризации, конфигурационного управления и ручной сборки образов. Выявлены их недостатки: виртуальные машины требуют передачи тяжёлых образов и усложняют проброс USB/JTAG; контейнеры вносят дополнительный уровень абстракции; пакетные менеджеры не работают без доступа к внешним репозиториям.

Шестой раздел формулирует требования к разрабатываемому решению. Функциональные требования включают поддержку различных источников компонентов (HTTP/HTTPS, file://, Git), автоматическое разрешение зависимостей и декларативность описания через YAML-манифесты. Требования к безопасности включают контроль целостности артефактов через SHA-256 и безопасное хранение учётных данных через `keyring`. Нефункциональные требования включают идемпотентность, переносимость, модульность и расширяемость.

Седьмой раздел описывает подход к реализации. Разработаны две версии утилиты: `isp-up` (автономное Python-приложение) и `isp-up-ansible` (с интеграцией Ansible для масштабирования на парк машин).

Восьмой раздел детально описывает архитектуру `isp-up`. Утилита

построена на модульном принципе. Описаны модули: `ManifestLoader` (загрузка YAML-манифестов), `DependencyResolver` (разрешение зависимостей), `DownloadManager` («ленивая» загрузка с кэшированием), `AuthManager` (безопасное хранение паролей через keyring). Приведены примеры использования и структура каталога после установки.

Девятый раздел описывает архитектуру `isp-up-ansible`. В отличие от первой версии, развёртывание выполняется через Ansible, что обеспечивает декларативность и идемпотентность. Описаны принципы работы, система манифестов, алгоритм развёртывания, управление аутентификацией, режимы работы (онлайн, офлайн, DMZ, USB) и приведены примеры использования.

Десятый раздел посвящён тестированию и верификации. Тестирование проводилось на ОС Astra Linux Special Edition 1.7 и Ubuntu 22.04 LTS. Разработаны пять сценариев: чистая установка, повторный запуск (идемпотентность), обрыв сети (Air Gap), сбой в середине установки, неверные учётные данные. Все сценарии успешно пройдены.

Одиннадцатый раздел формулирует критерии сравнения альтернативных подходов: воспроизводимость, идемпотентность, автономность, совместимость с дистрибутивами, объём артефактов, скорость обновления, аудитопригодность.

Двенадцатый раздел содержит сравнительный анализ подходов (ручная установка, виртуальные машины, контейнеры, `isp-up`). Показано, что `isp-up` занимает промежуточное положение, объединяя преимущества декларативного подхода и нативной установки.

Тринадцатый раздел описывает ограничения и перспективы развития. Текущие ограничения: привязка к Debian-подобным системам и требования к дисковому пространству. Перспективы: поддержка других архитектур (arm64), гибридный режим (генерация Dockerfile), интеграция с CI/CD (GitLab CI).

ЗАКЛЮЧЕНИЕ

В результате выполнения работы достигнута поставленная цель: разработан алгоритм автоматизации развёртывания инструментального окружения разработчика ОСРВ в изолированных средах, выполнена его программная реализация и проведён сравнительный анализ с существующими решениями.

Основные результаты:

- выявлено, что традиционные методы (ручная установка, ВМ, контейнеры) неприменимы в изолированных средах для системной разработки из-за проблем с весом образов, работой с железом или зависимостью от внешних репозиториев;
- спроектирована и реализована утилита `isp-up`, основанная на декларативных YAML-манифестах, идемпотентной установке и верификации компонентов через SHA-256;
- разработана версия на базе Ansible (`isp-up-ansible`) для масштабирования процесса на парк машин;
- проведённое тестирование подтвердило работоспособность в условиях обрыва сети (Air Gap) и устойчивость к сбоям;
- разработанный инструмент позволил сократить время подготовки рабочего места разработчика в 3–4 раза по сравнению с ручной установкой.

Практическая значимость подтверждается тем, что разработанные утилиты могут быть внедрены в процессы сопровождения разработки систем реального времени. Предложенный подход позволяет минимизировать «человеческий фактор» при настройке критически важного ПО и обеспечить идентичность окружений у всех участников проектной группы.

Перспективы развития связаны с поддержкой других архитектур (arm64), гибридным режимом (генерация Dockerfile) и интеграцией с CI/CD (GitLab CI).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Моррис К. Infrastructure as Code. Управление серверами в облаке. — 1-е изд. — СПб.: Питер, 2019. — 336 с.

2 Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. — СПб.: Питер, 2024. — 1120 с.

3 Любанович Б. Python. Экспресс-курс. 3-е изд. — СПб.: Питер, 2021. — 592 с.

4 Столлман Р. Руководство по GNU GCC. — М.: Бином, 2015. — 240 с.

5 Барр М., Масса А. Программирование встраиваемых систем на C и C++. — М.: ДМК Пресс, 2020. — 352 с.

6 Зыль С. Н. Операционные системы реального времени: учебное пособие. — М.: ИНФРА-М, 2021. — 168 с.

7 Петров С. В., Кузнецова Е. А. Автоматизированное развёртывание инструментальных сред разработки в закрытых сегментах сети // Программные продукты и системы. — 2024. — № 1. — С. 55—63.

8 Документация Ansible. Работа с плейбуками [Электронный ресурс]. — URL: https://docs.ansible.com/ansible/latest/userguide/playbooks.html (дата обращения: 9.05.2026).

9 Документация библиотеки Keyring [Электронный ресурс]. — URL: https://pypi.org/project/keyring/ (дата обращения: 9.05.2026).

10 YAML — это не язык разметки. Версия 1.2 [Электронный ресурс]. — URL: https://yaml.org/spec/1.2.2/ (дата обращения: 9.05.2026).