



Государственное бюджетное образовательное учреждение высшего образования
Московской области

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

имени дважды Героя Советского Союза, летчика-космонавта А. А. Леонова

Колледж космического машиностроения и технологий

КУРСОВОЙ ПРОЕКТ

по дисциплине МДК.01.02. Программирование в мехатронных системах

ПРОЕКТИРОВАНИЕ И ПРОГРАММИРОВАНИЕ СИСТЕМЫ СБОРКИ И ДОСТАВКИ ОТПРАВЛЯЕМЫХ ПРОГРАММНЫХ ПАКЕТОВ ДЛЯ ПЛК НА БАЗЕ ВСТРАИВАЕМЫХ ОПЕРАЦИОННЫХ СИСТЕМ

Пояснительная записка

КП.15.02.10.22.09.ПЗ

Обучающийся группы МР–19:

Краснов А. С.

Руководитель курсового проекта:

Фатеев Д. И.

Результат защиты: _____

г. Королёв, 2022 г

СОДЕРЖАНИЕ

ГЛОССАРИЙ.....	4
ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	6
ВВЕДЕНИЕ.....	7
1 АНАЛИТИЧЕСКАЯ ЧАСТЬ	10
1.1 Анализ существующих решений.....	10
1.1.1 Организационная структура предприятия	10
1.1.2 Существующие средства разработки.....	11
1.2 Исследование работы CoDeSyS	11
1.3 Устройство CoDeSyS.....	12
1.3.1 Средства разработки	12
1.3.2 Система исполнения «CoDeSyS»	12
1.3.3 Система загрузки программы на ПЛК.....	13
1.3.4 Сравнение интерфейсов сред программирования ПЛК.....	13
1.4 Поиск оптимального решения	13
1.4.1 Кроссплатформенность	15
1.4.2 Единый интерфейс	15
1.4.3 Выполнение программы.....	15
2 ПРАКТИЧЕСКИЙ РАЗДЕЛ	17
2.1 Разработка собственного программного комплекса	17
2.1.1 Архитектура.....	17

Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Краснов А. С.			ПРОЕКТИРОВАНИЕ И ПРОГРАММИРОВАНИЕ СИСТЕМЫ СБОРКИ И ДОСТАВКИ ОТПРАВЛЯЕ- МЫХ ПРОГРАММНЫХ ПАКЕТОВ ДЛЯ ПЛК НА БАЗЕ ВСТРАИВАЕМЫХ ОПЕРАЦИОННЫХ СИСТЕМ	Лит.	Лист
Провер.		Фатеев Д. И.					2
Реценз.		Фатеев Д. И.					33
Н. Контр.						ККМТ гр. МР-19	
Утверд.							

2.2 Особенности реализации программного комплекса	18
2.2.1 Универсальность	18
2.2.2 Реализация кроссплатформенности	18
2.2.1 Выбор протокола подключения к ПЛК	19
2.3 Особенности реализации «BuilderPLC»	20
2.3.1 Интерфейс и использование	20
2.3.2 Работа с файлами	22
2.4 Особенности реализации сервера.....	22
2.4.1 Удобство развертывания	23
2.4.2 Конфигурирование сервера	23
2.4.3 Особенности использования веб-фреймворка	23
3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ	27
3.1 Экономическое обоснование	27
3.2 Лицензирование разработанного программного обеспечения	27
ЗАКЛЮЧЕНИЕ	28
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	29
Приложение 1	30
Приложение 2	32

Изм.	Лист	№ докум.	Подпись	Дата	ПРОЕКТИРОВАНИЕ И ПРОГРАММИРОВАНИЕ СИСТЕМЫ СБОРКИ И ДОСТАВКИ ОТПРАВЛЯ- ЕМЫХ ПРОГРАММНЫХ ПАКЕТОВ ДЛЯ ПЛК НА БАЗЕ ВСТРАИВАЕМЫХ ОПЕРАЦИОННЫХ СИСТЕМ				Лит.	Лист	Листов	
Разраб.		Краснов А. С.										
Провер.		Фатеев Д. И.								3	33	
Реценз.		Фатеев Д. И.							ККМТ гр. МР-19			
Н. Контр.												
Утверд.												

ГЛОССАРИЙ

Программируемый логический контроллер (ПЛК) — специальная разновидность электронной вычислительной машины. Чаще всего ПЛК используют для автоматизации технологических процессов. В качестве основного режима работы ПЛК выступает его длительное автономное использование, зачастую в неблагоприятных условиях окружающей среды, без серьёзного обслуживания и практически без вмешательства человека.

Встраиваемая операционная система — любая компьютерная операционная система, имеющая специализированное назначение или рассчитанное на использование вместе с конкретным встраиваемым приложением. Конечный пользователь, как правило, не может изменять такие системы.

Кроссплатформенность — способность программного обеспечения работать с несколькими аппаратными платформами или операционными системами, в данном случае с различными ПЛК.

Машинный код — система команд (набор кодов операций) конкретной вычислительной машины, которая интерпретируется непосредственно процессором или микропрограммами этой вычислительной машины, в том числе и виртуальной машиной.

Виртуальная машина — программная и/или аппаратная система, эмулирующая аппаратное обеспечение некоторой платформы и исполняющая программы для guest-платформы на host-платформе (host — хост-платформа, платформа-хозяин) или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы, также спецификация некоторой вычислительной среды (например: «виртуальная машина языка программирования Си»). Виртуальная машина исполняет некоторый машинно-независимый код (например, байт-код, шитый код) или машинный код реального процессора. Помимо процессора, ВМ может эмулировать работу как отдельных компонентов аппаратного обеспечения, так и целого реального компьютера (включая BIOS, оперативную память, жёсткий диск и другие периферийные устройства).

Сетевая модель OSI — сетевая модель стека сетевых протоколов OSI/ISO. Посредством данной модели различные сетевые устройства могут взаимодействовать друг с другом. Модель определяет различные уровни взаимодействия систем. Каждый уровень выполняет определённые функции при таком взаимодействии.

Транспилиция — преобразование программы, при котором используется исходный код программы, написанной на одном языке программирования в качестве исходных данных, и производится эквивалентный исходный код на другом языке программирования.

Компилируемый язык программирования — язык программирования, исходный код которого преобразуется компилятором в машинный код и записывается в файл с особым заголовком и/или расширением для последующей идентификации этого файла, как исполняемого операционной системой (в отличие от интерпретируемых языков программирования, чьи программы выполняются программой-интерпретатором).

Фреймворк — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. Можно также говорить о каркасном подходе как о подходе к построению программ, где любая конфигурация программы строится из двух частей: постоянная часть — каркас, не меняющийся от конфигурации к конфигурации и несущий в себе гнезда, в которых размещается вторая, переменная часть; сменные модули (или точки расширения).

Ядро Linux — ядро операционной системы, соответствующее стандартам POSIX, составляющее основу операционных систем семейства Linux, а также ряда операционных систем для мобильных устройств, в том числе Android, Tizen, KaiOS.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Управляющая программа ПЛК — программа, реализующая основной алгоритм работы ПЛК.

Платформа ПЛК — совокупность характеристик, параметров, особенностей аппаратного и программного обеспечения, правил работы, парадигм и способов взаимодействия, использующихся при работе с конкретным ПЛК, а также других составляющих производственной экосистемы.

ПО — программное обеспечение (программа или множество программ, используемых для управления компьютером).

ВВЕДЕНИЕ

В данной работе рассмотрена проблема отсутствия стандартизации интерфейсов и способов загрузки управляющих программ в ПЛК различных производителей и рассматривается способ решения данной проблемы путем создания кроссплатформенного программного обеспечения, предоставляющего единый интерфейс и способ загрузки для ПЛК различных производителей.

Программируемые логические контроллеры широко применяются в сфере промышленной автоматизации разнообразных технологических процессов на больших и малых предприятиях. Популярность контроллеров легко объяснить. Их применение значительно упрощает создание и эксплуатацию как сложных автоматизированных систем, так и отдельных устройств, в том числе — бытового назначения. ПЛК позволяет сократить этап разработки, упрощает процесс монтажа и отладки за счет стандартизации отдельных аппаратных и программных компонентов, а также обеспечивает повышенную надежность в процессе эксплуатации, удобный ремонт и модернизацию при необходимости. [3]

Существует множество различных видов ПЛК, в том числе и свободно программируемые, в которые программа загружается через специальный интерфейс с персонального компьютера используя специальное ПО производителя. Такие ПЛК, как правило, работают на базе встраиваемых операционных систем. Каждый производитель подобных ПЛК разрабатывает собственный интерфейс и программное обеспечение для загрузки программного обеспечения на ПЛК, и подобные решения от разных производителей, как правило не совместимы. Отсутствие стандартизации и унификации также создает множество проблем и сложностей при построения автоматизированных промышленных систем.

Из-за отсутствия стандартизации и индивидуальной, несовместимой с другими производителями реализации способа загрузки управляющей программы на ПЛК, перед внедрением подобной системы производственной автоматизации необходимо тратить множество времени и средств для ознакомления именно с внедряемой системой, узнать все параметры и особенности аппаратной части ПЛК и освоить программное обеспечение для его программирования.

Подобная несовместимость между различными ПЛК может помешать обновлению инфраструктуры предприятия. Еще одна проблема, которую влечет за собой отсутствие обратной совместимости и сильные различия в платформах, на базе которых построены ПЛК – это необходимость переобучения инженеров производственной автоматизации при переходе или использование незнакомой им платформы, а так как платформы между собой не совместимы это процесс может быть довольно долгим. Также при поддержке уже имеющийся автоматизированной инфраструктуры какого-либо предприятия требуются специалисты, хорошо знакомые именно и используемой на предприятии платформой, что уменьшает количество возможно нанятых специалистов на рынке труда, и возможно, требует их дальнейшего переобучения. Подобные проблемы сильно усложняют внедрение современных технологий на производство и создают проблему при поддержке устаревших технологий.

Актуальность работы обусловлена отсутствием стандартизации способов загрузки управляющих программ на ПЛК между различными производителями, что облегчит переход и модернизацию производства, а также предоставит инженерам и разработчикам единый интерфейс и способ взаимодействия для большей части ПЛК на рынке.

Цель работы заключается в разработке программного обеспечения для загрузки управляющих программ на ПЛК различных производителей предоставляя для этого единый и стандартизированный интерфейс.

Предметом данной работы является технология и интерфейс доставки файлов на удаленный ПЛК.

Задачи проекта:

1. Провести анализ существующих на данный момент решений и выявить их основную проблему
2. Выбрать существующее решение, провести исследование его устройства и работы
3. Найти оптимальный способ решения выявленной проблемы
4. Разработать собственное решение на основе ранее найденного оптимального способа
5. Произвести анализ результатов работы и тестирование программного обеспечения

Курсовая работа состоит из 33 страниц, 13 рисунков, 1 таблицы и 2 приложений. Данная работа включает в себя глоссарий, обозначения и сокращения, заключение, список используемых источников и 2 приложения.

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ существующих решений

Современный ПЛК имеет развитый комплекс программных средств, состоящих из системного программного обеспечения ПЛК, предоставляемого производителем контроллера и сторонних или собственных программных средств, предназначенных для разработки, отладки и записи в контроллер пользовательских программ.

Системное программное обеспечение ПЛК, состоящее из операционной системы с интегрированным в нее набором драйверов, отвечает за выполнение контроллером пользовательского приложения, обслуживает низкоуровневую систему ввода-вывода контроллера, интерфейсы передачи данных, управляет распределением памяти, режимами энергопотребления, таймерами, осуществляет обработку ошибок, позволяя пользователю, разрабатывающему приложение, полностью сосредоточиться на алгоритмической части решения прикладной задачи. [5]

Средства разработки и отладки пользовательских программ позволяют создавать и корректировать программы, реализующие алгоритмы работы контроллера, моделировать на ПК процесс выполнения программы контроллером, наблюдать за промежуточными результатами вычислений, а также записывать программное обеспечение в контроллер.

1.1.1 Организационная структура предприятия

ПЛК может использоваться повсеместно там, где есть производство – любая задача, которая требует использования электрических устройств управления, имеет потребность в ПЛК. А для внедрения ПЛК на любое производство требуется его программирование и администрирование, в том числе и загрузка управляющей программы. Особенно остро данная проблема проявляется при работе и обслуживании множества ПЛК, при том, что данные контроллеры, а вместе с тем и программное обеспечение для работы с ними, может довольно сильно отличаться, что создает дополнительные трудности при их эксплуатации.

Данное программное обеспечение можно применять на всех предприятиях использующих ПЛК для упрощения их эксплуатации и ускорения обслуживания множества ПЛК, а также использовать данное ПО (Программное обеспечение) как универсальную систему взаимодействия с ПЛК.

1.1.2 Существующие средства разработки

На рынке ПЛК существует множество различных производителей. Одними из таких производителей, представленных на Российском рынке, являются компании: «Овен», «Siemens», «Mitsubishi», «Beckhoff».

Большая часть производителей ПЛК имеет собственное программное обеспечение, обычно базирующиеся на программном комплексе «CoDeSyS». Но интерфейс и способ взаимодействия различного программного обеспечения базирующегося на «CoDeSyS» довольно сильно отличается и, как ранее упоминалось, модифицированные версии различных производителей несовместимы между собой.

1.2 Исследование работы CoDeSyS

Название CoDeSyS сегодня созвучно с профессиональным проектированием приложений в сфере промышленной автоматизации. Более 100 известных компаний во всем мире избрали CoDeSyS своим инструментом программирования. CoDeSyS применяется изготовителями средств автоматизации, системными интеграторами, а также инженерами и техниками на производстве. Ежедневно тысячи пользователей во всем мире с успехом используют его. Сегодня CoDeSyS является самым распространенным инструментом МЭК 61131-3 программирования в Европе. [1] Фактически он сам уже давно служит стандартом систем МЭК программирования, именно по этой причине он и был взят за основу при проектировании собственного программного обеспечения не только большинством производителей ПЛК, но и мной.

1.3 Устройство CoDeSyS

CoDeSyS является ведущим программным инструментом разработки проектов в соответствии с МЭК 61131–3. Он сочетает в себе классическое программирование контроллеров с возможностью профессиональной разработки ПО для устройств автоматизации.

CoDeSyS (сокращение от слов Controller Development System) это инструмент программирования промышленных компьютеров и контроллеров опирающийся на международный стандарт МЭК 61131-3. Изящный функциональный набор и простота применения, обеспечили CoDeSyS первое место на Европейском рынке программных инструментов. [2]

CoDeSyS предоставляет программисту удобную среду для программирования контроллеров на языках стандарта МЭК 61131-3. Используемые редакторы и отладочные средства базируются на широко известных и хорошо себя зарекомендовавших принципах, знакомых по другим популярным средам профессионального программирования.

1.3.1 Средства разработки

Основой комплекса CoDeSyS является среда разработки прикладных программ для программируемых логических контроллеров (ПЛК). Она распространяется бесплатно и, может быть, без ограничений установлена на нескольких рабочих местах.

1.3.2 Система исполнения «CoDeSyS»

Встроенные компиляторы CoDeSyS генерируют IL код, компилируемый в машинный код (двоичный код), выполняющийся на аппаратной платформе ПЛК. Но на ПЛК с операционной системой подобный код лучше выполнять на виртуальной машине, что обеспечивает совместимость между различными платформами и предсказуемость результатов выполнения программы. Виртуальная машина для выполнения IL кода называется «Control Runtime System», она устанавливается в контроллер в процессе его изготовления его производителем. Существует специальный инструмент (Software development kit), позволяющий адаптировать её к различным аппаратным и программным платформам. [4]

Благодаря выполнению программы на виртуальной машине можно сосредоточить все внимание на создании исполняемой программы, а разработчик ПЛК позаботиться о ее выполнении в используемом контроллере.

1.3.3 Система загрузки программы на ПЛК

Загрузить программу в ПЛК можно из интерфейса среды разработки CoDeSyS, но производители, как правило предлагают модифицированный интерфейс, а использование оригинальной, не модифицированной версии не всегда представляется возможным из-за наличия различных программных модулей производителей ПЛК, которые отсутствуют в оригинальной версии программы.

1.3.4 Сравнение интерфейсов сред программирования ПЛК

Сравнивая интерфейс различных сред разработки от производителей ПЛК, таких как «Овен» (рисунок 1.2), «Siemens» (рисунок 1.1) и «Mitsubishi» (рисунок 1.4) с оригинальной версией CoDeSyS (рисунок 1.1) можно заметить множество различий.

Подобные различия затрудняют переход операторов ПЛК между этими средами разработки, требуется время для освоения и привыкания к новым интерфейсам. Это влечет за собой простой производства и сложности при обновлении инфраструктуры предприятия. Решением подобных проблем является создание среды разработки с унифицированным интерфейсом.

1.4 Поиск оптимального решения

Для решения описанных выше проблем разрабатываемый программный комплекс должен обеспечивать единый интерфейс для взаимодействия с ПЛК вне зависимости от его платформы и способа запуска управляющей программы. Для этого ПО должно быть кроссплатформенным как со стороны клиента (диспетчерской), так и со стороны сервера (ПЛК).

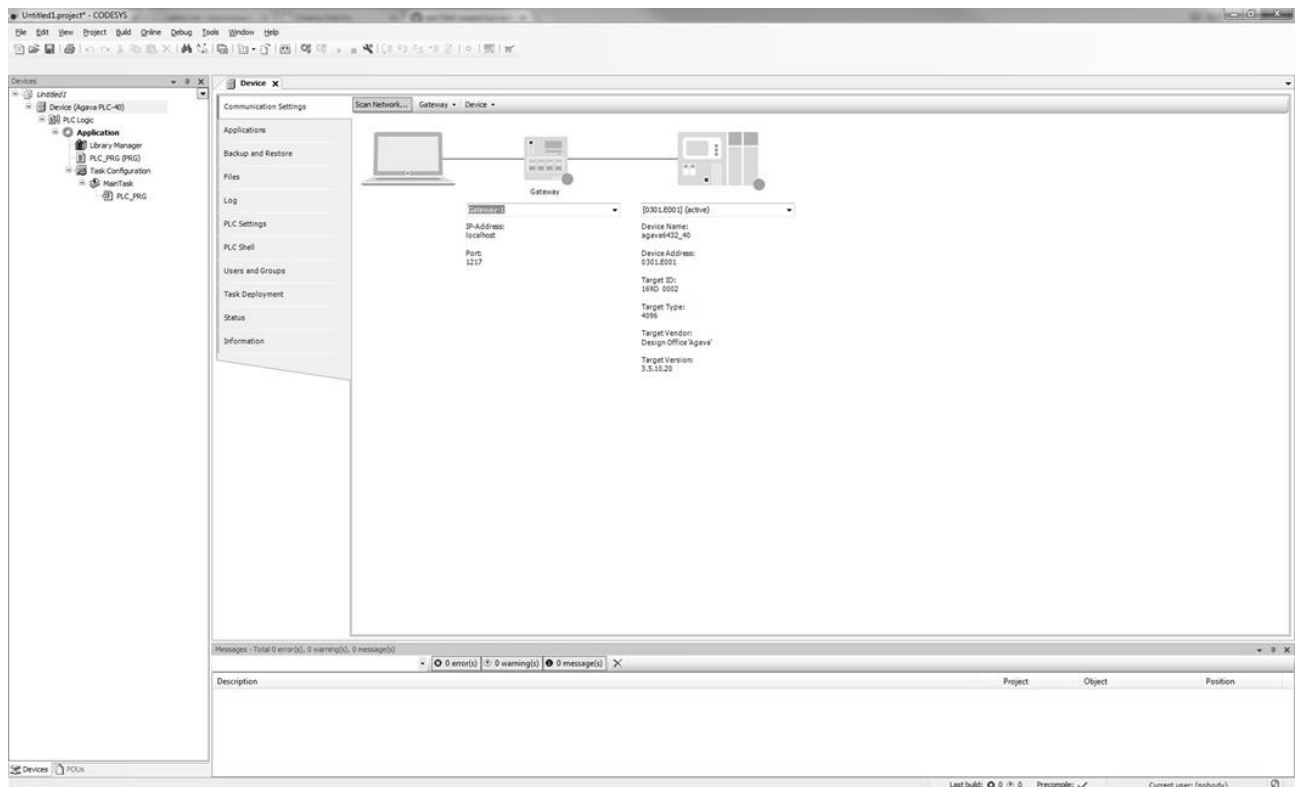


Рисунок 1.1. Загрузка программы в ПЛК в стандартной версии CoDeSys

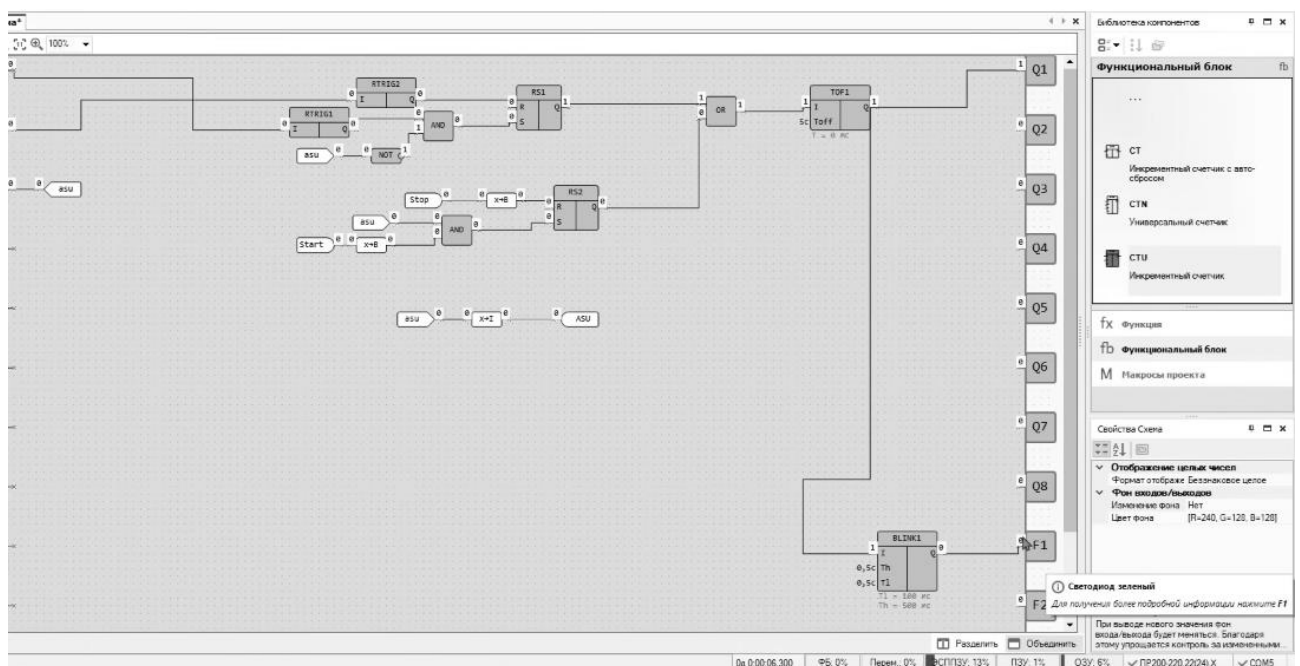


Рисунок 1.2. Загрузка программы в ПЛК в среде программирования OwenLogic

1.4.1 Кроссплатформенность

Кроссплатформенность разрабатываемого программного комплекса обеспечивает высокоуровневый интерпретируемый язык программирования, что обеспечит работу ПО на любом компьютере с установленным интерпретатором такого языка, а в случае необходимости присутствует возможность транспилирования созданного программного обеспечения для работы на встраиваемых операционных системах, установленных на ПЛК. При том программный комплекс не обеспечивает выполнение управляющей программы на ПЛК самостоятельно, а реализует стандартизированный заложенный производителем способ запуска отведенной на ПЛК управляющей программы.

1.4.2 Единый интерфейс

Большая часть ПЛК на предприятии объединена или подключена к компьютерной сети, в связи с чем за основной способ взаимодействия с ПЛК был выбран доступ через используемую на предприятии компьютерную сеть.

Так как разрабатываемое программное обеспечение рассчитано на работу во встраиваемой операционной системе ПЛК, то реализация всех нижестоящих уровней модели OSI лежит именно на используемой операционной системе. Использование стандартных способов взаимодействия и интеграции в уже существующие системы обеспечивает кроссплатформенность и простоту работы программного комплекса.

1.4.3 Выполнение программы

Создание качественного транслятора языка программирования высокого уровня является сложной и трудоемкой задачей. Поэтому выполнение управляющей программы на ПЛК происходит на виртуальной машине «Control Runtime System» или любой другой совместимой с ней, т.к. именно она является стандартом для большинства производителей ПЛК. Данная виртуальная машина уже интегрирована в ПЛК, поддерживающий стандарт МЭК 61131-3. В случае отсутствия данного программного решения производителем ПЛК будет предусмотрен иной способ выполнения программы, но совместимый с МЭК 61131-3, а значит и с разрабатываемым программным комплексом.

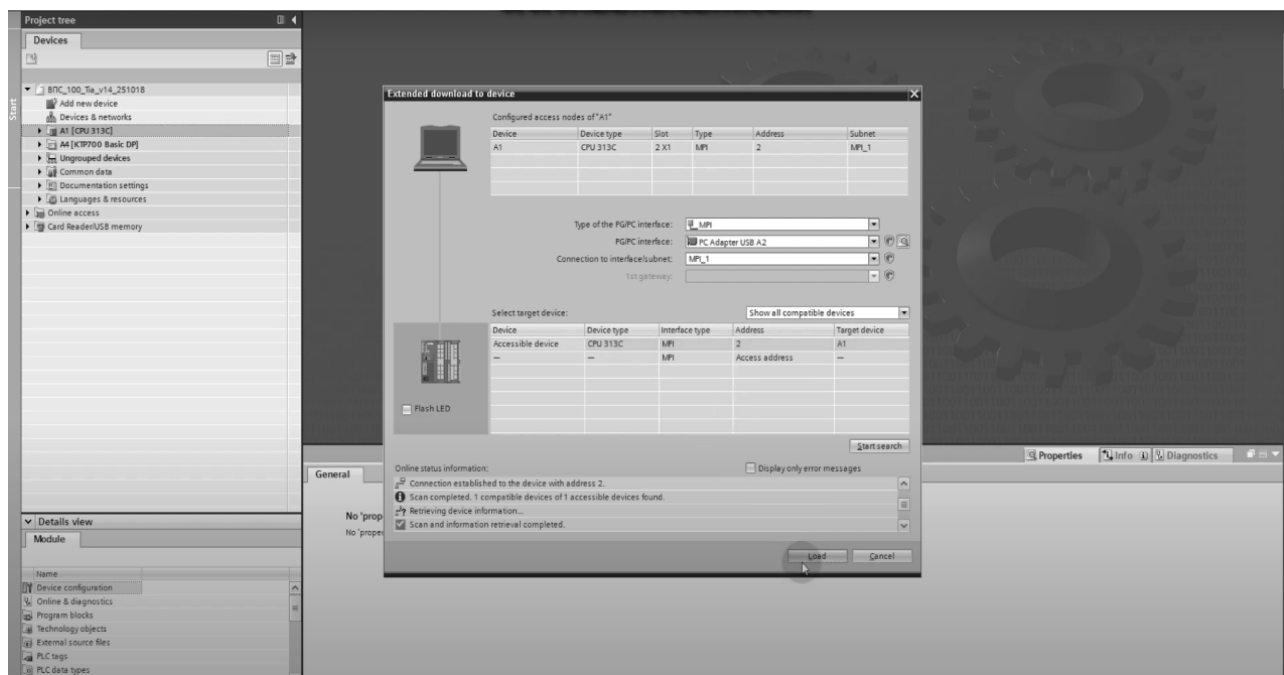


Рисунок 1.3. Загрузка программы в ПЛК в среде программирования Siemens TiaPorta

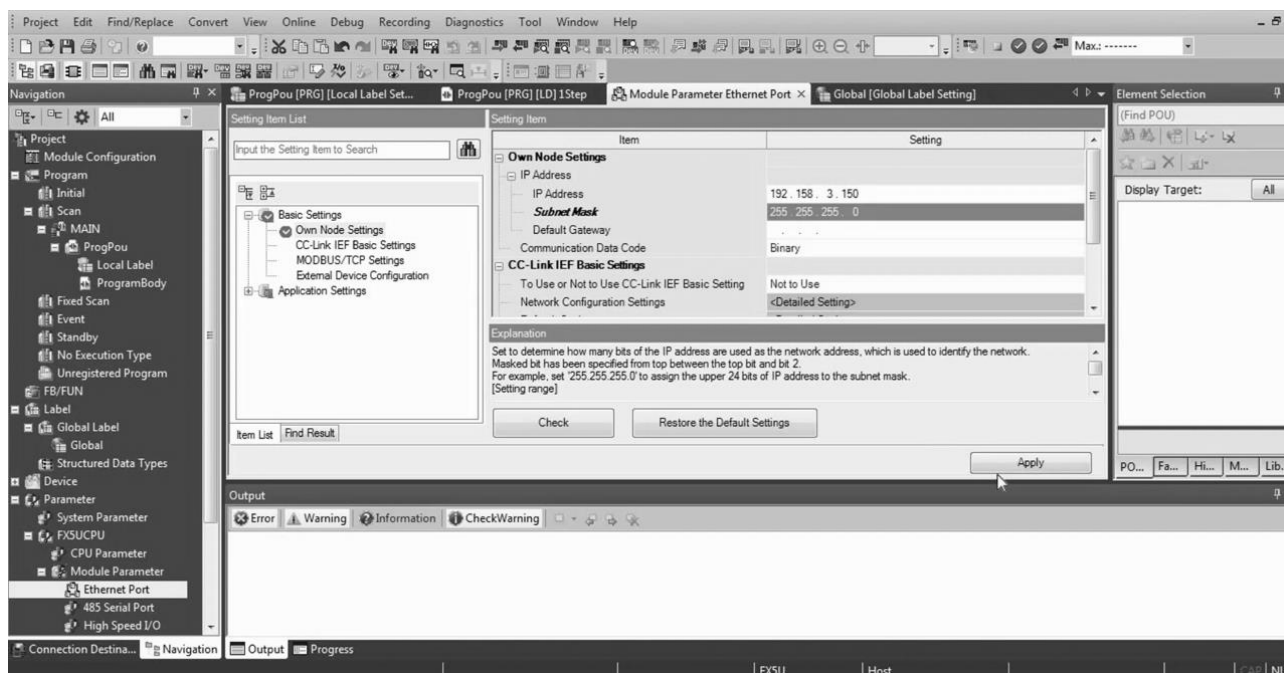


Рисунок 1.4. Загрузка программы в ПЛК в среде программирования GX Works 3

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Разработка собственного программного комплекса

На основе описанных ранее проблем было разработано программное обеспечение, предоставляющее унифицированный интерфейс загрузки управляющей программы на ПЛК. К данному программному обеспечению были выдвинуты следующие требования:

1. Универсальность – программа должна предоставлять единый интерфейс и способ загрузки вне зависимости от используемого ПЛК
2. Простота развертывания – внедрение программы на производство должно быть максимально простым и быстрым
3. Кроссплатформенность – программа должна работать на актуальных операционных системах (Windows, Linux, MacOS)

2.1.1 Архитектура

Разработанный программный комплекс состоит из 2 программ: сервера, запущенного на ПЛК и клиента на компьютере для взаимодействия с ПЛК. В качестве клиента может применяться программа «BuilderPLC» или любой современный браузер. Управляющая программа доставляется на ПЛК в виде отправляемого программного пакета, в состав которого входит:

1. Управляющая программа для выполнения на виртуальной машине ПЛК или любым другим способом
2. Исполняемый файл для загрузки виртуальной машины или выполнения управляющей программы
3. Любые другие необходимые файлы

Отправляемый программный пакет (рисунок 2.1) представляет из себя сжатый упакованный zip архив и собирается при помощи приложения «BuilderPLC», которое также может быть использовано для отправки и удаленного выполнения управляющей программы на ПЛК. Подобный пакет может включать в себя исполняемый файл, файл с управляющей программой для ПЛК и другие файлы.

2.2 Особенности реализации программного комплекса

2.2.1 Универсальность

Универсальность данного программного комплекса и возможность использования на различных ПЛК реализуется за счет выполнения управляющей программы в среде «Control Runtime System» или на любой другой виртуальной машине, реализующей стандарт IEC 61131-3 (МЭК 61131-3). Запуск управляющей программы на виртуальной машине обеспечивается выполнением исполняемого файла, доставляемого в отправляемом программном пакете, что также обеспечивает универсальность программы и возможность запуска и развертывания любого программного обеспечения на ПЛК вне зависимости от его аппаратной платформы.

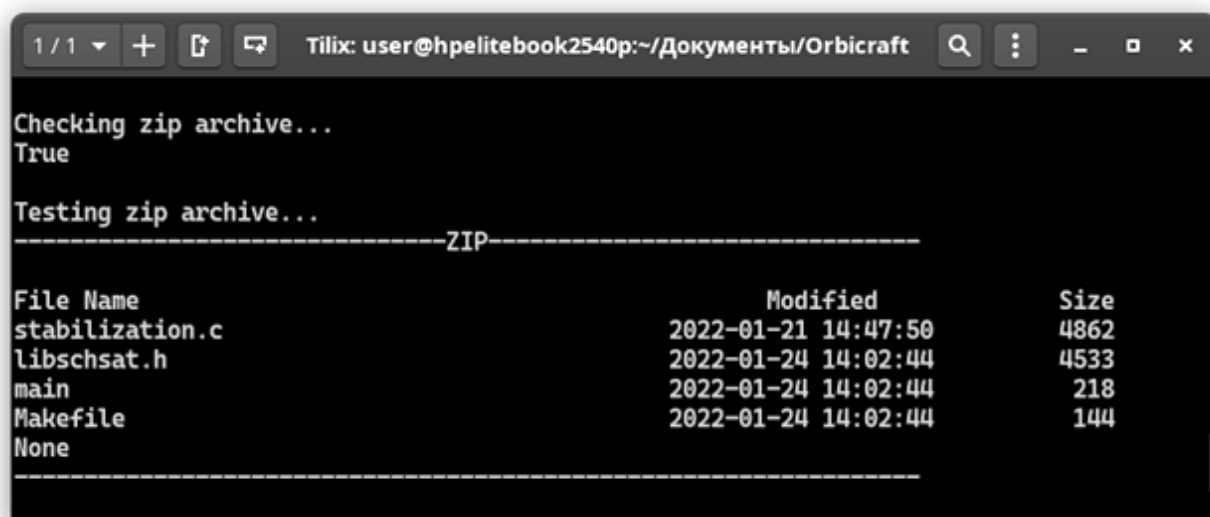


Рисунок 2.1. Содержание отправляемого пакета (zip архива)

2.2.2 Реализация кроссплатформенности

Программный комплекс написан на языке программирования Python версии 3.9 и выше, что позволяет запускать его на любой поддерживаемой операционной системе (Windows, Linux, MacOS) с установленным интерпретатором. Также сохраняется возможность произвести транспилиацию программы (рисунок 2.2) на компилируемый язык программирования, что ускорит работу программы и обеспечит более удобный способ ее распространения в виде самодостаточного исполняемого двоичного файла под выбранную при последующей компиляции операционную систему.

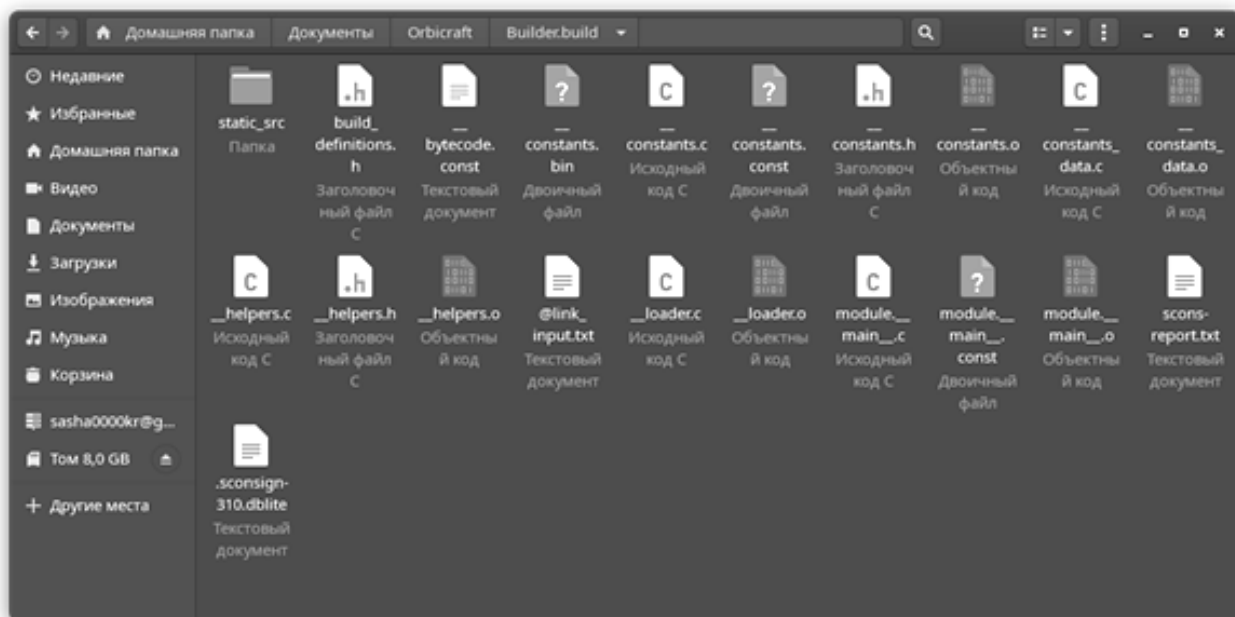


Рисунок 2.2. Транспирированная на си версия «BuilderPLC» на ос Linux

2.2.1 Выбор протокола подключения к ПЛК

Обмен данными происходит на прикладном уровне модели OSI, что обеспечивает работу программы в любых сетях, в том числе и промышленных вне зависимости от транспортного протокола и физической реализации шины обмена данными. В качестве протокола используется HTTP/HTTPS. Подобное решение обеспечивает работоспособность программы вне зависимости от аппаратной платформы используемого оборудования, в том числе и ПЛК. Также подобная сетевая архитектура позволяет размещать несколько ПЛК в одной сети. Пример возможной конфигурации сети представлен на

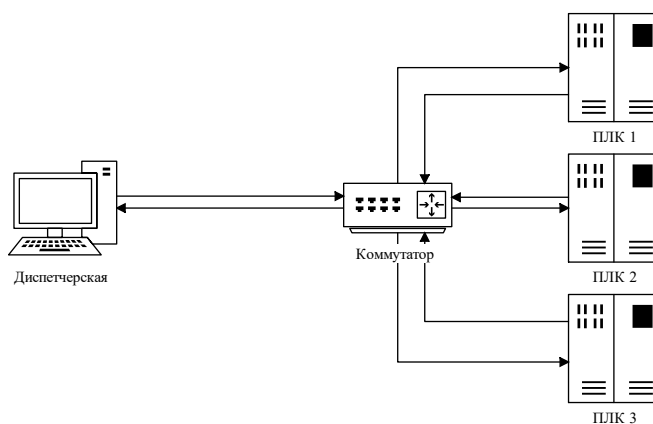


Рисунок 2.3. Архитектура промышленной сети

2.3 Особенности реализации «BuilderPLC»

Основные особенности программы:

1. Программа выполнена в виде отдельного приложения не привязанного к редактору кода или IDE, но оставляющая такую возможность
2. Кроссплатформенность (работа на основных популярных ОС - Windows, Linux, MacOS)
3. Использование актуального и поддерживаемого языка программирования
4. Минимизация внешних зависимостей
5. Подробный вывод отладочной информации о работе программы
6. Проверка и подробный вывод возможных ошибок, связанных с отсутствием возможности выполнения команд
7. Возможность обработки большого количества файлов
8. В приоритете надежность и скорость работы

При разработке данной программы было принято решение руководствоваться философией GNU, т.е. создать маленькое монолитное приложения, которое выполняет одну функцию и делает это хорошо.

Для упрощения кодовой базы и решения проблем с зависимостями используется монолитная архитектура, ООП не применяется, сторонние зависимости, не входящие в состав языка Python не используются.

Все действия над файлами производятся в директории, из которой запущен «BuilderPLC». В ходе работы программы используются только относительные пути, а значит нет ограничений на путь до рабочей директории, в которой запущено приложение. Также нет ограничений и на имя файла.

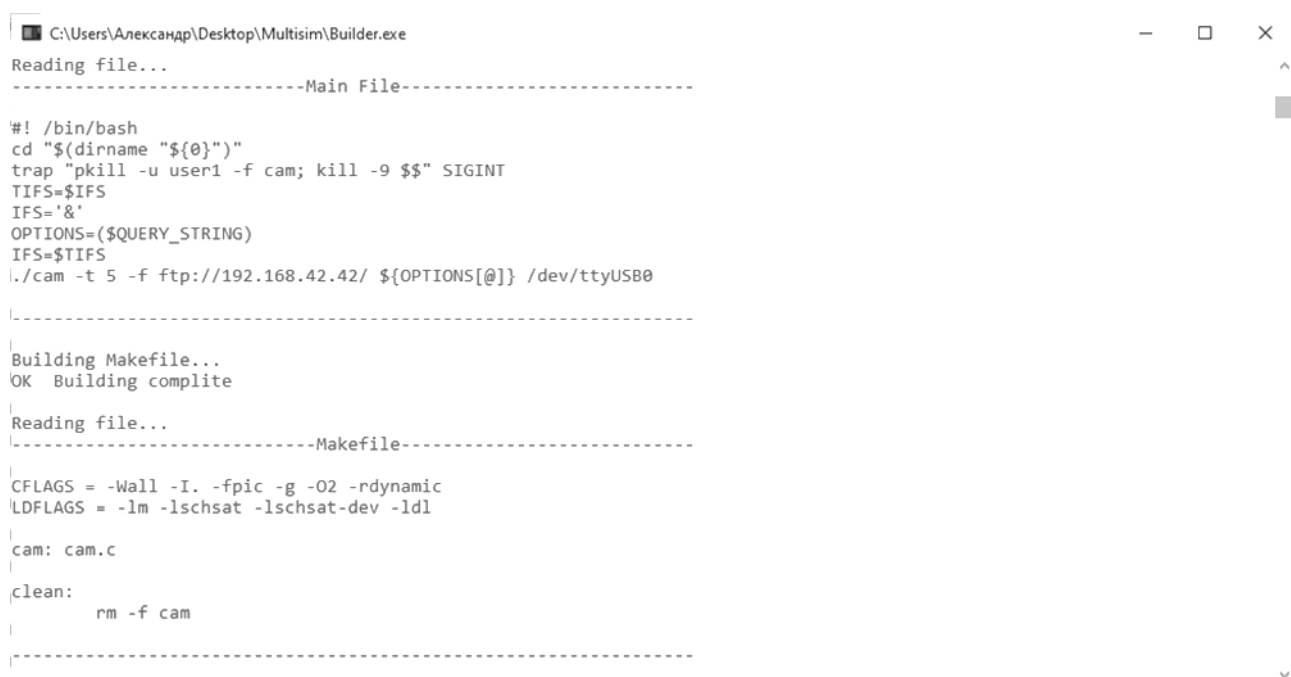
2.3.1 Интерфейс и использование

Для удобства использования предусмотрено несколько режимов работы:

1. Пользовательский режим
2. Режим терминала

Пользовательский режим (рисунок 2.4) наиболее прост в использовании и применяется по умолчанию при запуске программы без каких-либо флагов, т. е. традиционным для ОС Windows способом (двойной клик по иконке программы). В данном режиме происходит сборка всех пакетов, находящихся в директории, из которой была запущена программа, при этом выводится вся информация о состоянии работы программы. После сборки всех пакетов программа ожидает закрытия пользователем.

Режим терминала (рисунок 2.5) предназначен для более продвинутых пользователей или для интеграции с редактором кода или другим ПО. В данном режиме программа принимает флаги, позволяющие настроить поведение программы при каждом запуске и принимает путь до файла с исходным кодом, что позволяет использовать всего одну копию программы для сборки пакетов в любой директории файловой системы компьютера, при этом нет ограничений на использование символов в пути и названии файлов. В данном режиме программа принимает флаги, позволяющие настроить поведение программы. После окончания работы программы она автоматически закрывается, не ожидая действия от пользователя. Лучше всего такой режим работает в UNIX операционных системах, вроде Linux и Mac OS.



```
C:\Users\Александр\Desktop\Multisim\Builder.exe
Reading file...
-----Main File-----

#!/bin/bash
cd "$(dirname "${0}")"
trap "pkill -u user1 -f cam; kill -9 $$" SIGINT
IFS=$IFS
IFS='&'
OPTIONS=($QUERY_STRING)
IFS=$IFS
./cam -t 5 -f ftp://192.168.42.42/ ${OPTIONS[@]} /dev/ttyUSB0

-----
Building Makefile...
OK Building complete
Reading file...
-----Makefile-----

CFLAGS = -Wall -I. -fpic -g -O2 -rdynamic
LDFLAGS = -lm -lschsat -lschsat-dev -ldl

cam: cam.c

clean:
    rm -f cam

-----
```

Рисунок 2.4. Снимок экрана «BuilderPLC» в пользовательском режиме

2.3.2 Работа с файлами

«BuilderPLC» работает не с одним файлом, а со всеми файлами в директории, за счёт чего происходит обработка множества файлов с минимальными усилиями со стороны пользователя.

После выполнения сборки всех файлов в рабочей директории программа переходит в автоматический режим (рисунок 2.6), т.е. проверяет время последнего сохранения файла и при изменении времени сохранения заново собирает пакет. Данный режим облегчает работу с программой и увеличивает скорость обработки файлов, нежели при ручной сборке и позволяет взаимодействовать с файлом в автоматическом режиме без интеграции в редактор кода.

2.4 Особенности реализации сервера

Основные особенности программы:

1. Использование прикладного уровня модели OSI и протокола HTTP/HTTPS
2. Кроссплатформенность (работа на основных встраиваемых ОС - Linux, возможность запуска транспирированного варианта на Windows CE)
3. Использование актуального и поддерживаемого языка программирования и высокопроизводительного веб-фреймворка
4. Использование простого и быстрого API
5. Проверка и подробный вывод возможных ошибок, связанных с отсутствием возможности выполнения команд
6. Автоматическое генерирование документации
7. Перезагрузка во время работы

Программа написана на веб-фреймворке FastAPI с использованием веб сервера Uvicorn. Программа в первую очередь ориентирована на работе в UNIX подобных операционных системах и разрабатывалась для ОС на базе ядра Linux, но может быть портирована на Windows CE при необходимости путем транспиляции python кода в компилируемый язык программирования.

2.4.1 Удобство развертывания

Для удобства развертывания сервера (рисунок 2.7) был написан `bash` скрипт, при запуске которого на целевой платформе (ПЛК) автоматически устанавливается, запускается и автоматически добавляется в автозагрузку сервер, что сильно упрощает его внедрение на производство.

2.4.2 Конфигурирование сервера

В ходе своей работы сервер выводит множество отладочных сообщений (рисунок 2.8), что позволяет легко следить за его состоянием и найти проблему в случае неполадок. Поток отладочных сообщений можно перенаправить в файл для анализа нештатных ситуаций в случае отсутствия постоянного наблюдения за работой сервера. Также сервер может выводить отладочные сообщения от управляющей программы ПЛК, выполняемый на нем.

2.4.3 Особенности использования веб-фреймворка

Встроенные функции веб-фреймворка FastAPI дают серверу перезагружаться во время работы «на лету», т.е. быстро вернуться к состоянию и конфигурации, действующей до перезагрузки.

Еще одно преимущество использования FastAPI является автоматически генерирующаяся документация (рисунок 2.8). При помощи подобной документации можно вручную выполнять какие-либо команды на сервере, а также моментально получить доступ к документации и изучить все возможные ответы сервера на запросы пользователя во время его работы.

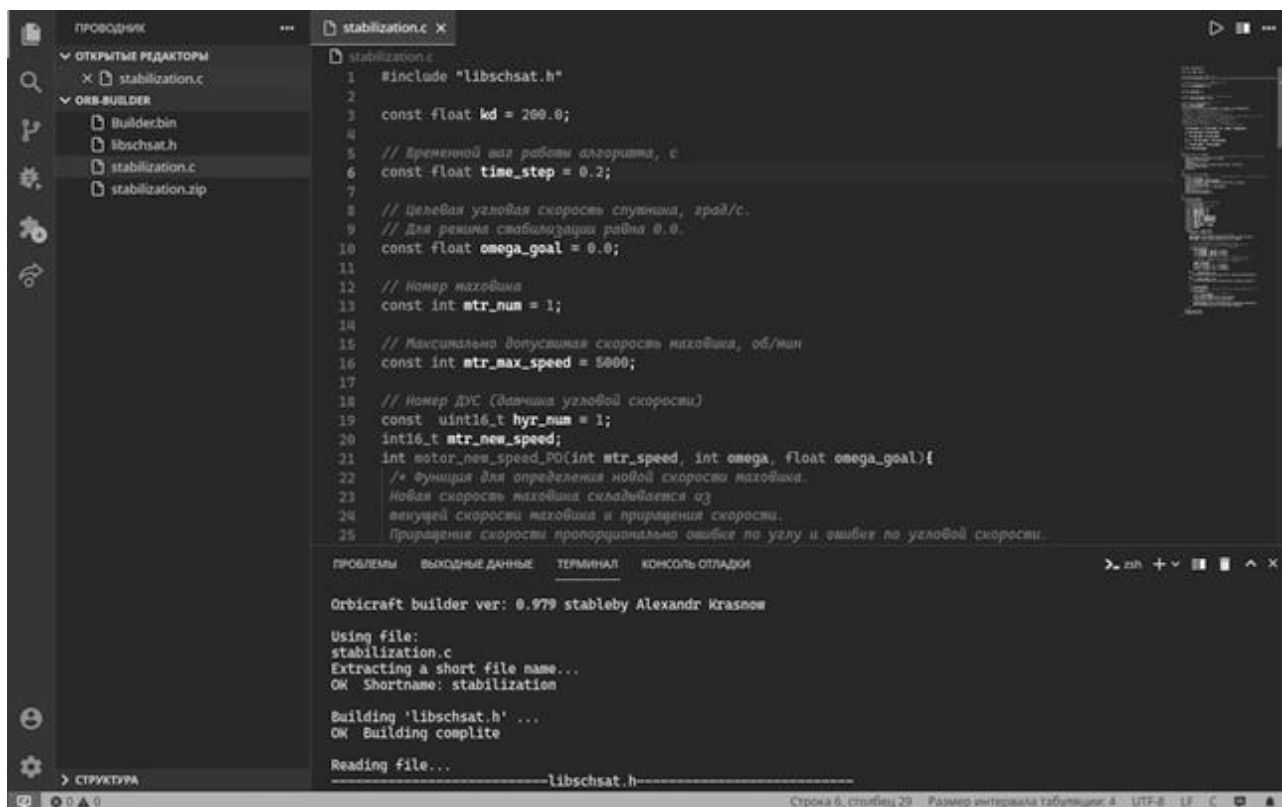


Рисунок 2.5. Работа «BuilderPLC» в режиме терминала (автоматическое закрытие после выполнения сборки) с интеграцией Visual Studio Code на ОС Manjaro Linux

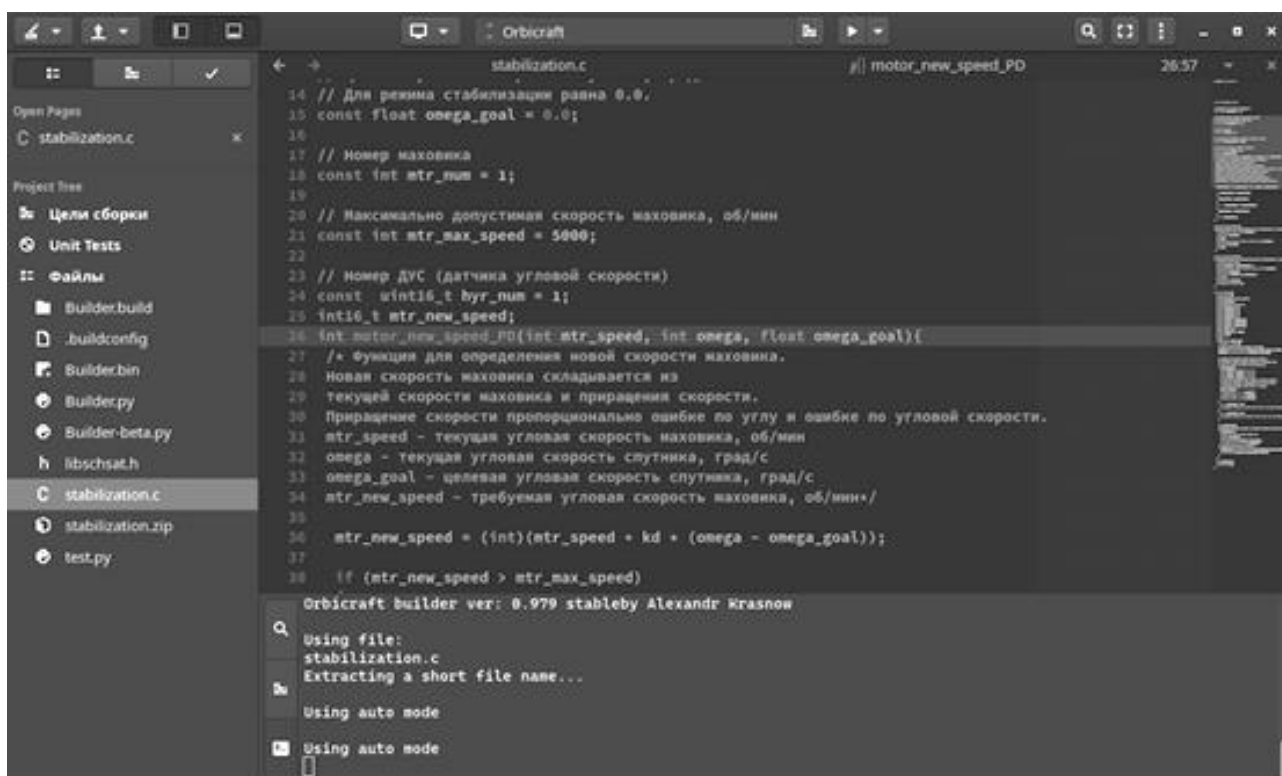


Рисунок 2.6. Работа «BuilderPLC» в автоматическом режиме на ОС Linux


```

Requirement already satisfied: pydantic<1.7,>=1.7.1,!=1.7.2,!=1.7.3,!=1.8,!=1.8.1,<2.0.0,>=1.6.2 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from fastapi) (1.9.1)
Requirement already satisfied: anyio<5,>=3.4.0 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from starlette==0.19.1->fastapi) (3.6.1)
Requirement already satisfied: typing-extensions<=3.7.4.3 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from pydantic!=1.7,!=1.7.1,!=1.7.2,!=1.7.3,!=1.8,!=1.8.1,<2.0.0,>=1.6.2->fastapi) (4.2.0)
Requirement already satisfied: idna<=2.8 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from anyio<5,>=3.4.0->starlette==0.19.1->fastapi) (3.3)
Requirement already satisfied: sniffio<=1.1 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from anyio<5,>=3.4.0->starlette==0.19.1->fastapi) (1.2.0)
Requirement already satisfied: uvicorn[standard] in /data/data/com.termux/files/usr/lib/python3.10/site-packages (0.17.6)
Requirement already satisfied: asgiref<=3.4.0 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from uvicorn[standard]) (3.5.2)
Requirement already satisfied: click<=7.0 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from uvicorn[standard]) (8.1.3)
Requirement already satisfied: h11<=0.8 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from uvicorn[standard]) (0.13.0)
Requirement already satisfied: websockets<=10.0 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from uvicorn[standard]) (10.3)
Collecting watchdog<=0.6
  Using cached watchdog-0.8.2-py3-none-any.whl (12 kB)
Collecting uvloop<=0.15.0,!=0.15.1,>=0.14.0
  Using cached uvloop-0.16.0.tar.gz (2.1 MB)
  Preparing metadata (setup.py) ... done
Collecting PyYAML<=5.1
  Using cached PyYAML-6.0-cp310-cp310-linux_armv7l.whl
Collecting python-dotenv<=0.13
  Using cached python-dotenv-0.20.0-py3-none-any.whl (17 kB)
Collecting httptools<=0.4.0
  Using cached httptools-0.4.0.tar.gz (174 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: anyio<4,>=3.0.0 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from watchdog<=0.6->uvicorn[standard]) (3.6.1)
Requirement already satisfied: sniffio<=1.1 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from anyio<4,>=3.0.0->watchdog<=0.6->uvicorn[standard]) (1.2.0)
Requirement already satisfied: idna<=2.8 in /data/data/com.termux/files/usr/lib/python3.10/site-packages (from anyio<4,>=3.0.0->watchdog<=0.6->uvicorn[standard]) (3.3)
Using legacy 'setup.py install' for httptools, since package 'wheel' is not installed.
Using legacy 'setup.py install' for uvloop, since package 'wheel' is not installed.
Installing collected packages: uvloop, PyYAML, python-dotenv, httptools, watchdog

```

Рисунок 2.7. Процесс автоматического развертывания сервера на ОС Linux

```

~/plc/TempMonitor $ python kernel.py
IP: 192.168.43.1
PORT: 2212
INFO: Will watch for changes in these directories: ['/data/data/com.termux/files/home/plc/TempMonitor']
INFO: Uvicorn running on http://192.168.43.1:2212 (Press CTRL+C to quit)
INFO: Started reloader process [21372] using statreload
INFO: Started server process [21374]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 192.168.43.237:51296 - "GET /docs HTTP/1.1" 200 OK
INFO: 192.168.43.237:51296 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 192.168.43.237:51301 - "GET /filelist HTTP/1.1" 200 OK
Filename: hello.zip
Short name: hello
Extracting file
Using directoy: /data/data/com.termux/files/home/plc/TempMonitor/hello
Starting subprocess: main.sh
Obtaining file execution rights
Starting subprocess: main.sh
WARNING: StatReload detected file change in 'hello/hello.py'. Reloading ...
Test Program
Clearing cache
Work directoty: /data/data/com.termux/files/home/plc/TempMonitor
INFO: 192.168.43.237:51307 - "GET /run/hello.zip HTTP/1.1" 200 OK
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [21374]
INFO: Started server process [21385]
INFO: Waiting for application startup.
INFO: Application startup complete.

```

Рисунок 2.8. Вывод отладочных сообщений и результат запуска управляющей программы «Hello World» на удаленном сервере (ПЛК)

Загрузчик ПЛК 0.1.0 OAS3

/openapi.json

Действия

POST **/load** Create Load File

Parameters

No parameters

Request body

multipart/form-data

file *

string(\$binary)

Выберите файл

Файл не выбран

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
'http://192.168.43.1:1024/load' \
-H 'accept: application/json' \
-H 'Content-Type: multipart/form-data' \
-F 'file=@hello.zip;type=application/x-zip-compressed'
```

Request URL

```
http://192.168.43.1:1024/load
```

Server response

Code	Details
200	<div><div>Response body</div><div>null</div><div>Download</div></div> <div><div>Response headers</div><div>content-length: 4 content-type: application/json date: Tue, 14 Jun 2022 14:41:38 GMT server: uvicorn</div></div>

Responses

Code	Description	Links
200	Successful Response	No links

Рисунок 2.9. Пример автогенерируемой документации, используемой как панель управления удаленным сервером (ПЛК)

3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

3.1 Экономическое обоснование

При разработке программного пакета использовалось преимущественно программное обеспечение с открытым исходным кодом, в связи с чем большую часть расходов составляет оборудование. Но большая часть оборудования было предоставлено «Технологическим университетом», поэтому количество реальных расходов составило около 5000 рублей. На создание данного программного пакета было затрачено 168 часов.

№	Статья расхода	Кол-во	Стоимость за шт, руб..	Сумма, руб.
1	Компьютер на Windows	1 шт.	50 000	50 000
2	Компьютер на Linux	1 шт.	50 000	50 000
3	Компьютер на Mac OS	1 шт.	110 000	110 000
4	Заработная плата разработчика	1 час	150	25 200
5	Мелкие и непредвиденные расходы			5 000
Итого				240 200

3.2 Лицензирование разработанного программного обеспечения

Исходный код сервера распространяется под лицензией MIT и представлен в приложении 1, любой желающий бесплатно может воспользоваться основными возможностями разработанного ПО. Программа «BuilderPLC» имеет закрытый исходный код и является отдельной составляющей программного пакета. И несмотря на то, что при использовании только сервера программный пакет будет не полным, его использование возможно и без «BuilderPLC». Оригинальный текст лицензии MIT и неофициальный перевод на русский язык представлены в приложении 2.

ЗАКЛЮЧЕНИЕ

Современный ПЛК стал чрезвычайно востребованным универсальным рабочим инструментом в системах автоматизации производственных процессов, а также для управления отдельными устройствами различного назначения. Это особый тип программируемых логических автоматов, отличающийся повышенной надежностью, легко встраиваемый и модернизируемый, способный длительное время работать практически без обслуживания. Не смотря почти на полную стандартизацию и унификацию языков программирования и наличие универсальных средств разработки, одна из проблем при использовании ПЛК – это отличие интерфейсов различных программных пакетов от множества производителей ПЛК, и в ходе данной курсовой работы было разработано одно из решений данной проблемы.

В результате выполнения курсовой работы был создан программный комплекс, состоящий из 2 программ: система сборки и отправки программных пакетов на ПЛК «BuilderPLC» и сервер, реализующий единый интерфейс загрузки и запуска управляющих программ, при том не зависящий от используемого ПЛК. На данный момент программный комплекс прошел тестирование и испытание на виртуальной машине с установленной ОС Linux, в связи с чем можно сказать, что цель данной работы была достигнута.

В ходе выполнения данной работы мною было изучено создание приложений и программирование на python и bash, основные веб технологии, API и фреймворки для создания веб приложений, использование системы контроля версий git, системное программирование, основы работы и настройка ОС GNU/Linux, научился конфигурировать и разворачивать веб сервера в автоматическом режиме.

Также хочется отметить, что данный проект являлся в первую очередь учебным, и в дальнейшем на его основе можно будет создать более мощное и конкурентоспособное коммерческое программное обеспечение.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. **CoDeSyS Интегрированный комплекс МЭК 61131-3 программирования** [В Интернете] / авт. ПК Пролог,. - ПК Пролог. - 20 06 2022 г.. - <http://www.prolog.smolensk.ru/>.
2. **CODESYS Профессиональные средства разработки приложений МЭК 61131–3** [В Интернете] / авт. ПК ПРОЛОГ. - ПК ПРОЛОГ. - 20 06 2022 г.. - www.prolog-plc.ru.
3. **Введение в ПЛК: что такое программируемый логический контроллер** [В Интернете] / авт. АО «КОМПЭЛ». - 14 12 2018 г.. - 20 06 2022 г.. - <https://www.compel.ru/lib/95591>.
4. **Программируемые контроллеры. Стандартные языки и приёмы прикладного проектирования** [Книга] / авт. В. Петров И.. - Москва : СОЛОН-Пресс, 2004. - Под ред. проф. В. П. Дьяконова..
5. **Программируемые логические контроллеры для управления технологическими процессами** [Книга] / авт. Ю. Е. Лившиц В. И. Лакин, Ю. И. Мониц. - Минск : БНТУ, 2014.

Приложение 1

Исходный код сервера

```
1      from fastapi import FastAPI, File, UploadFile
2      import multiprocessing
3      import subprocess
4      import zipfile
5      import shutil
6      import os
7      import uvicorn
8      import socket
9      import settings
10     from colorama import init
11
12
13     init()
14
15
16     from colorama import Fore, Back, Style
17
18     host, port = settings.init ()
19     if __name__ == '__main__':
20         print (Fore.GREEN + "IP" + Style.RESET_ALL + ":      " +
21               str(host))
22         print (Fore.GREEN + "PORT" + Style.RESET_ALL + ":      " +
23               str(port))
24
25     uvicorn.run(
26         "server:app",
27         host=host,
28         port=port,
29         reload=True
30     )
31
32     app = FastAPI()
33
34     def runfile (filename):
35         print ("Filename: " + str(filename))
36         shortfile = filename [::-1]
37         shortfile = shortfile [4:]
38         shortfile = shortfile [::-1]
39         print ("Short name: " + str (shortfile))
40
41         print ("Extracting file")
42         zipfile.ZipFile (filename).extractall (shortfile)
43         zipfile.ZipFile (filename).close ()
44
45         os.chdir (shortfile)
46         print ("Using directoy: " + str (os.getcwd()))
47         print ("Starting subprocess: " + "main.sh")
```

```

46         print ("Obtaining file execution rights")
47         os.system ('chmod +x %s.sh'% "main")
48         print ("Starting subprocess: " + "main.sh")
49         process = subprocess.run ("./%s.sh"% "main",
capture_output=True)
50         out = process.stdout
51
52
53         print ("Process STDOUT:" + str (out))
54         print ("Stop subprocess: \n" + str (process) +
"\nPROCESS STDOUT: " + str (out))
55
56         print ("Clearing cache")
57         os.chdir ("../")
58         print ("Work direcotory: " + str (os.getcwd ()))
59         os.system ("rmdir --ignore-fail-on-non-
empty %s" %shortfile)
60         return {"ExitCode": out, "STDOUT": out}
61
62
63     @app.post("/load")
64     async def create_load_file(file: UploadFile = File (...)):
65         with open (f"{file.filename}", "wb") as buffer:
66             shutil.copyfileobj (file.file, buffer)
67
68     @app.post("/load/run")
69     async def create_upload_file(file: UploadFile = File (...)):
70         with open (f"{file.filename}", "wb") as buffer:
71             shutil.copyfileobj (file.file, buffer)
72         runfile (file.filename)
73
74     @app.get("/remove/{zipfile_name}")
75     async def remove_item(zipfile_name: str):
76         out = os.remove (zipfile_name)
77         return out
78
79
80     @app.get("/filelist")
81     async def file_list():
82         return os.listdir (os.getcwd ())
83
84     @app.get("/run/{zipfile_name}")
85     async def read_item(zipfile_name: str):
86         runfile (zipfile_name)

```

**Оригинальный текст лицензии MIT и неофициальный перевод на
русский язык**

Оригинальный текст лицензии

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Неофициальный перевод

Данная лицензия разрешает лицам, получившим копию данного программного обеспечения и сопутствующей документации (в дальнейшем именуемыми «Программное обеспечение»), безвозмездно использовать Программное обеспечение без ограничений, включая неограниченное право на использование, копирование, изменение, слияние, публикацию, распространение, сублицензирование и/или продажу копий Программного обеспечения, а также лицам, которым предоставляется данное Программное обеспечение, при соблюдении следующих условий:

Указанное выше уведомление об авторском праве и данные условия должны быть включены во все копии или значимые части данного Программного обеспечения.

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА УЩЕРБ ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, В ТОМ ЧИСЛЕ, ПРИ ДЕЙСТВИИ КОНТРАКТА, ДЕЛИКТЕ ИЛИ ИНОЙ СИТУАЦИИ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.