Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт
Вычислительная техника
Кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« ____» _____ 2022 г.

БАКАЛАВРСКАЯ РАБОТА

«Система удалённого доступа к лабораторному оборудованию. Подсистема удалённого управления платой STK500»
Тема

09.03.01 «Информатика и вычислительная техника»
код и наименование направления

Руководитель	_____ подпись, дата	доцент.каф. ВТ ИКИТ Канд. физ.-мат. наук должность, ученая степень	К.В. Коршун инициалы,фамилия
Выпускник	_____ подпись, дата		Н.В. Коптяев инициалы,фамилия
Нормоконтролер	_____ подпись, дата	доцент.каф. ВТ ИКИТ Канд. физ.-мат. наук должность, ученая степень	К.В. Коршун инициалы,фамилия

Красноярск 2022

РЕФЕРАТ

Настоящая бакалаврская работа посвящена разработке подсистемы удалённого управления платой STK500 для системы удалённого доступа к лабораторному оборудованию.

Данная пояснительная записка содержит 53 страницы текста с иллюстрациями и таблицами, 2 приложения, и 18 использованных источников.

УДАЛЁННЫЙ ДОСТУП, СИСТЕМА УДАЛЁННОГО ДОСТУПА, ВИРТУАЛЬНАЯ ЛАБОРАТОРИЯ, ПРОГРАММНЫЙ СИМУЛЯТОР, МИКРОПРОЦЕССОР, STK500, ЭМУЛЯЦИЯ, ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА, КНОПКИ, ATMEGA32A, КОМАНДНАЯ СТРОКА, HEX-ФАЙЛ.

Цель бакалаврской работы – разработка подсистемы удалённого управления платой STK500 для системы удалённого доступа к лабораторному оборудованию.

Задачи, решённые в процессе разработки:

- проведён анализ различных вариантов обеспечения удалённого доступа к лабораторному оборудованию;
- разработана архитектура разрабатываемой системы;
- разработана аппаратная часть системы;
- разработана система эмуляции периферийных устройств платы STK500;
- осуществлена прошивка платы STK500 через командную строку с помощью hex-файла;
- разработаны управляющие скрипты на языке Python для автоматизации формирования и ввода управляющих команд;
- проведено тестирование разработанной системы.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В.Непомнящий
подпись инициалы, фамилия
« ___» _____ 20__ г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы

Студенту Коптяеву Николаю Васильевичу

фамилия, имя, отчество

Группа КИ18-09Б Направление (специальность) 09.03.01

номер код

Информатика и вычислительная техника

наименование

Тема выпускной квалификационной работы Система удалённого доступа к лабораторному оборудованию. Подсистема удалённого управления платой STK500.

Утверждена приказом по университету № _____ от _____

Руководитель ВКР К.В. Коршун, доцент каф. ВТ, канд. физ.-мат. наук

инициалы, фамилия, должность, учёное звание и место работы

Исходные данные для ВКР: Провести анализ различных вариантов обеспечения удалённого доступа к лабораторному оборудованию. Разработать архитектуру разрабатываемой системы. Разработать аппаратную часть системы. Разработать систему эмуляции периферийных устройств платы STK500. Осуществлять прошивку платы STK500 через командную строку с помощью hex-файла. Разработать управляющие скрипты на языке Python для автоматизации формирования и ввода управляющих команд. Провести тестирование разработанной системы.

Перечень разделов ВКР: Анализ вариантов решения проблемы удалённого доступа к лабораторному оборудованию. Разработка системы удалённого управления платой STK500. Тестирование разработанной системы.

Перечень графического материала: Презентация в формате Power Point, структурная схема архитектуры системы удалённого доступа к лабораторному оборудованию, схема подключения Arduino Uno к плате STK500, схемы алгоритмов разработанных программ, фотоснимки лабораторного стенда при проведении тестирования, скриншоты работы разработанной системы при проведении тестирования.

Руководитель ВКР _____ К.В. Коршун

подпись инициалы, фамилия

Задание принял к исполнению _____ Н.В. Коптяев

подпись инициалы, фамилия

«___» _____ 20__г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время проблема удалённого доступа к лабораторному оборудованию является особенно актуальной. В условиях дистанционного обучения студентам необходимо иметь доступ к оборудованию для выполнения лабораторных работ. Для этого необходимо реализовать возможность удалённо взаимодействовать с лабораторными стендами, то есть эмулировать его периферийные устройства с помощью платы Arduino Uno R3, управляемой через командную строку компьютера. Также необходимо обеспечить автоматизацию ввода управляющих команд с помощью скриптов на языке Python.

В настоящей работе использовались лабораторный стенд STK-500, платформа Arduino Uno R3, цифро-аналоговый преобразователь MCP4725, а также среды разработки Arduino IDE и Microchip Studio. Программное обеспечение реализовано на языках Python, C/C++ и ассемблере.

1 Анализ вариантов решения проблемы удалённого доступа к лабораторному оборудованию

1.1 Актуальность выбранной темы

В современных реалиях всё более актуальной становится повсеместная цифровизация. Многие организации временно или постоянно переходят на удалённый режим работы, либо частично внедряют в рабочий процесс цифровые технологии. В связи с этим появляется проблема доступа к оборудованию, ведь в условиях удаленной работы или учёбы становится невозможно пользоваться некоторыми видами оборудования. Это создаёт определённые проблемы. В частности, проблемы возникают в сфере высшего образования, и конкретно в высших учебных заведениях, готовящих специалистов в области информационных технологий.

При подготовке специалистов в данной области необходимо формировать у студентов навыки работы с различными видами программного обеспечения и оборудования, в частности, проводятся занятия с использованием макетных плат и лабораторных стендов на базе микропроцессоров или ПЛИС, предназначенных для выполнения лабораторных работ. При переходе на дистанционное обучение студенты лишаются возможности работать с этим оборудованием, и не получают необходимых навыков. Студенты заочной формы обучения или студенты с ограниченными возможностями здоровья полностью лишены возможности такой работы. Однако навыки работы с оборудованием являются одними из важнейших для специалиста в сфере информационных технологий.

Для решения данной проблемы можно выделить несколько основных вариантов, которые рассмотрены в настоящей работе. К ним относятся виртуальные лаборатории, программные симуляторы и системы удалённого доступа.

1.2 Виртуальные лаборатории

Виртуальная лаборатория представляет собой программно-аппаратный комплекс, позволяющий проводить опыты без непосредственного контакта с реальной установкой или при полном отсутствии таковой. В первом случае мы имеем дело с так называемой лабораторной установкой с удаленным доступом, в состав которой входит реальная лаборатория, программно-аппаратное обеспечение для управления установкой и оцифровки полученных данных, а также средства коммуникации. Во втором случае все процессы моделируются при помощи компьютера [2].

Виртуальные лаборатории обладают рядом преимуществ, среди которых можно, согласно источнику [2] выделить:

- Отсутствие необходимости приобретения дорогостоящего оборудования и реактивов.

- Возможность моделирования процессов, протекание которых принципиально невозможно в лабораторных условиях.

- Наглядная визуализация процессов, трудноразличимых в реальных условиях на экране компьютера.

- Возможность проникновения в тонкости процессов и наблюдения происходящего в другом масштабе времени, что актуально для процессов, протекающих за доли секунды или, напротив, длящихся в течение нескольких лет.

- Безопасность в случаях, где идет работа, например, с высокими напряжениями или химическими веществами.

- Возможность быстрого проведения серии опытов с различными значениями входных параметров, что часто необходимо для определения зависимостей выходных параметров от входных.

- Автоматический ввод и обработка больших массивов полученных цифровых данных, которые выполняются на компьютере при проведении

серии экспериментов. Таким образом, экономится время и значительно уменьшается процент возможных ошибок.

- И, наконец, отдельное и важное преимущество заключается в возможности использования виртуальной лаборатории в дистанционном обучении, когда в принципе отсутствует возможность работы в лабораториях университета, либо в обучении студентов заочной формы.

Некоторые из этих преимуществ повторяются для программных симуляторов и систем удалённого доступа.

Рассмотрим несколько примеров виртуальных лабораторий.

STAR (Software Tools for Academics and Researchers) – программа Массачусетского технологического института (MIT) по разработке виртуальных лабораторий для исследований и обучения. Деятельность программы заключается в разработке обучающих и исследовательских приложений по общей биологии, биохимии, генетике, гидрологии, в области распределенных вычислений. На рисунке 1 представлен скриншот одной из программ – StarBiochem это 3D-визуализатор молекул белков, которая позволяет студентам в интерактивном режиме изучать ключевые концепции биологии белков [3, 4].

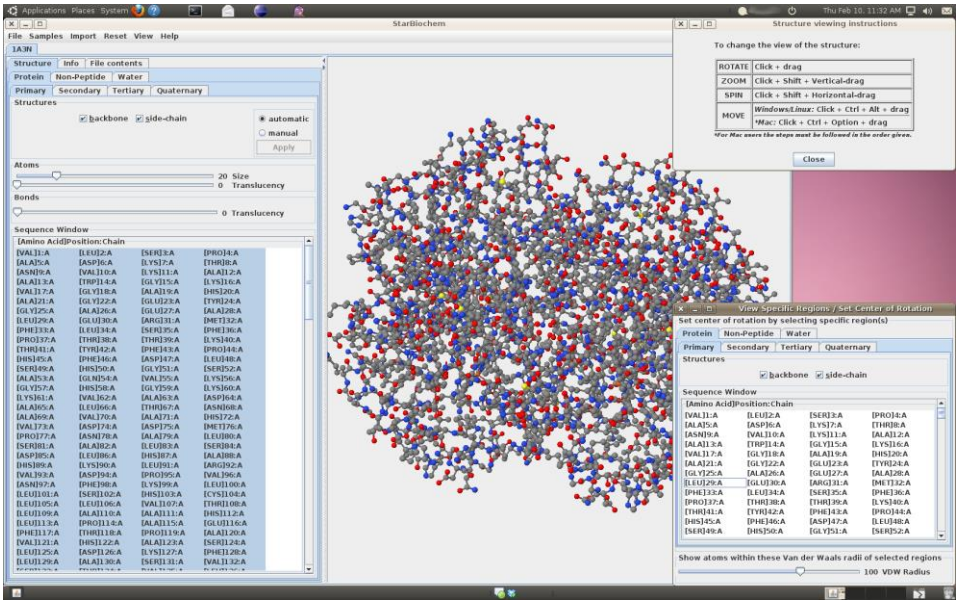

Рисунок 1 – StarBiochem

VirtuLab (рис. 2) – проект по разработке виртуальных лабораторных работ для учащихся по физике, химии, биологии, экологии. Виртуальные лабораторные работы реализованы при помощи технологии Flash. Отличаются узкой специализацией, в большинстве случаев линейностью опыта (вся последовательность действий и результаты опыта заданы заранее) [5, 6].

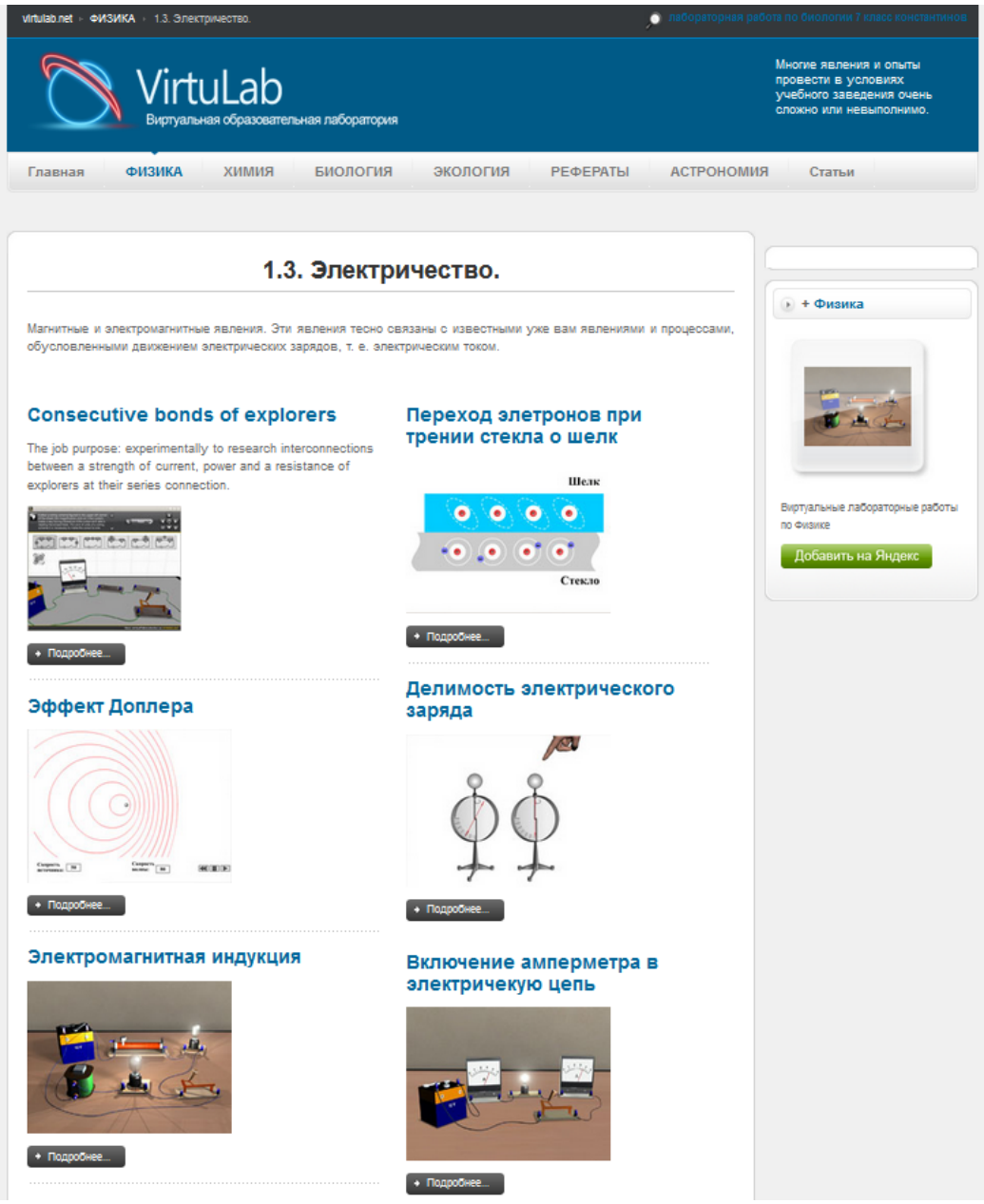

Рисунок 2 – Некоторые лабораторные работы в VirtualLab

Виртуальных лабораторий не так много, и по большей части они направлены на лабораторные работы для школьников, а не для студентов.

Также они обладают рядом недостатков, основным из которых является дороговизна разработки таких систем на профессиональном уровне, либо их узкая направленность для удовлетворительного результата, если разработчики не являются профессионалами.

1.3 Программные симуляторы

Симулятор – программное средство, способное имитировать работу микроконтроллера и его памяти. Как правило, симулятор содержит в своем составе:

- Отладчик;
- Модель ЦПУ и памяти.

Более продвинутые симуляторы содержат в своем составе модели встроенных периферийных устройств, таких, как таймеры, порты, АЦП, системы прерываний.

Симулятор должен уметь загружать файлы программ во всех популярных форматах, максимально полно отображать информацию о состоянии ресурсов симулируемого микроконтроллера, а также предоставлять возможности по симуляции выполнения загруженной программы в различных режимах. В процессе отладки модель выполняет программу, и на экране компьютера отображается текущее состояние модели [7].

Однако симулятор зачастую может показать только внешнее поведение системы, без оглядки на конкретные инструкции процессора. Такие системы не могут дать опыта, аналогичного опыту работы с реальным оборудованием. Чтобы приблизиться к программному воплощению работы реального устройства, необходимо смоделировать не только внешнего поведения, но и внутреннюю структуру и логику работы. Модели, которые действительно эмулируют устройство, а не только результат исполнения программы, правильнее называть эмуляторами, а не симуляторами.

Однако создание эмуляторов гораздо сложнее из-за большего объема функциональности, которую необходимо реализовывать в модели. Также они функционируют намного медленнее по сравнению с симуляторами внешнего поведения устройства. Из-за высокой сложности разработка таких систем слишком ведётся слишком долго и стоит дорого, а потому нерациональна в сравнении с симуляторами. Поэтому вместо того чтобы эмулировать всю систему целиком, эмулируются отдельные её компоненты, например, центральный процессор, на котором запускается лишь часть симуляционного процесса. Возможны различные гибридные схемы, когда часть симулятора является моделью верхнего уровня, часть – моделью нижнего уровня, часть в ПЛИС, а часть представляет собой реальное устройство [10].

Рассмотрим примеры некоторых программных симуляторов.

Одним из распространённых симуляторов является Proteus. Эта программа, разработанная британской компанией Labcenter Electronics, является системой моделирования, базирующейся на основе моделей электронных компонентов. Программа состоит из двух модулей. ISIS — редактор электронных схем с последующей имитацией их работы. ARES — редактор печатных плат, оснащенный автотрассировщиком Electra, встроенным редактором библиотек и автоматической системой размещения компонентов на плате. Кроме этого ARES может создать трехмерную модель печатной платы. Proteus VSM включает в себя электронные компоненты со всеми справочными данными, а также демонстрационные ознакомительные проекты [8]. На рисунке 3 представлен скриншот программы.

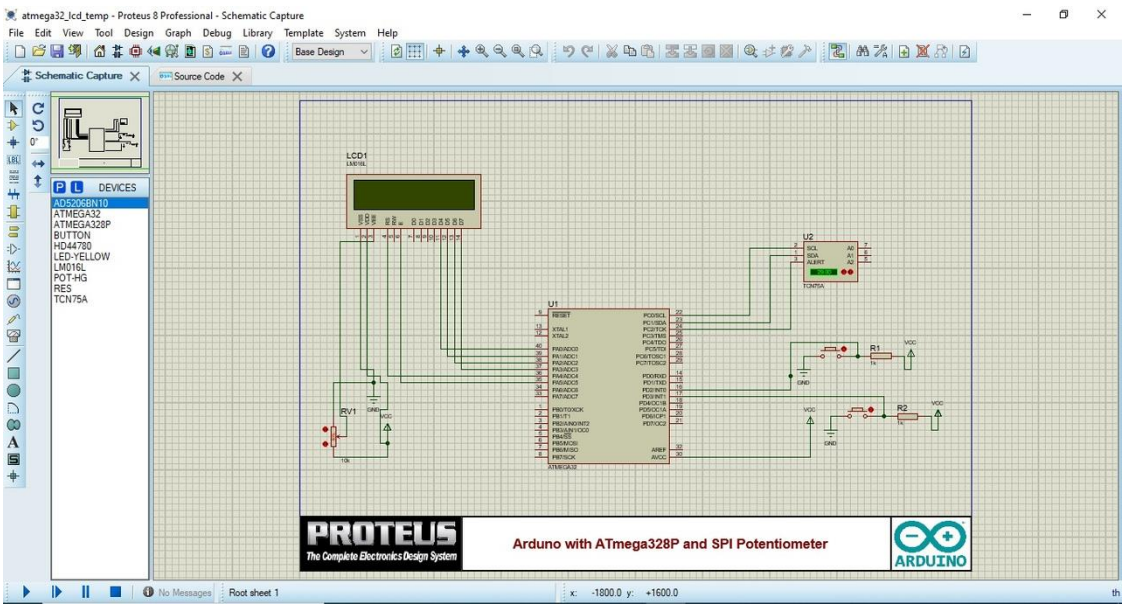

Рисунок 3 – Моделирование в Proteus

PSpice (Personal Simulation Program with Integrated Circuit Emphasis) — программа симуляции аналоговой и цифровой логики, описанной на языке SPICE, которая предназначена для персональных компьютеров (первая буква «P» в названии). Разработана компанией MicroSim и используется в автоматизации проектирования электронных приборов. Компания-разработчик была приобретена фирмой OrCAD, а затем Cadence Design Systems. В настоящее время программа также может симулировать и смешанные аналого-цифровые схемы.

Так как программа была выпущена в январе 1984 года, она стала одной из первых программ в области моделирования электроники. Сейчас на ядре SPICE основаны почти все подобные программы. В частности, с PSpice совместим описанный выше симулятор Proteus.

PSpice содержит большое количество готовых библиотек Spice-моделей и схемных символов, а кроме того, в интернете доступно огромное количество дополнительных моделей – как на сайтах производителей электронных компонентов, так и на специализированных порталах. Все они совместимы с PSpice. Программа предоставляет широкий набор средств как для моделирования, так и для обработки результатов анализа. Есть возможность

назначать модели, устанавливать точки для контроля напряжений, токов и мощности, пользоваться формулами для построения требуемых графиков и осциллограмм, строить совмещенные или разбитые по разным осям графики [9]. На рисунке 4 представлен скриншот программы.

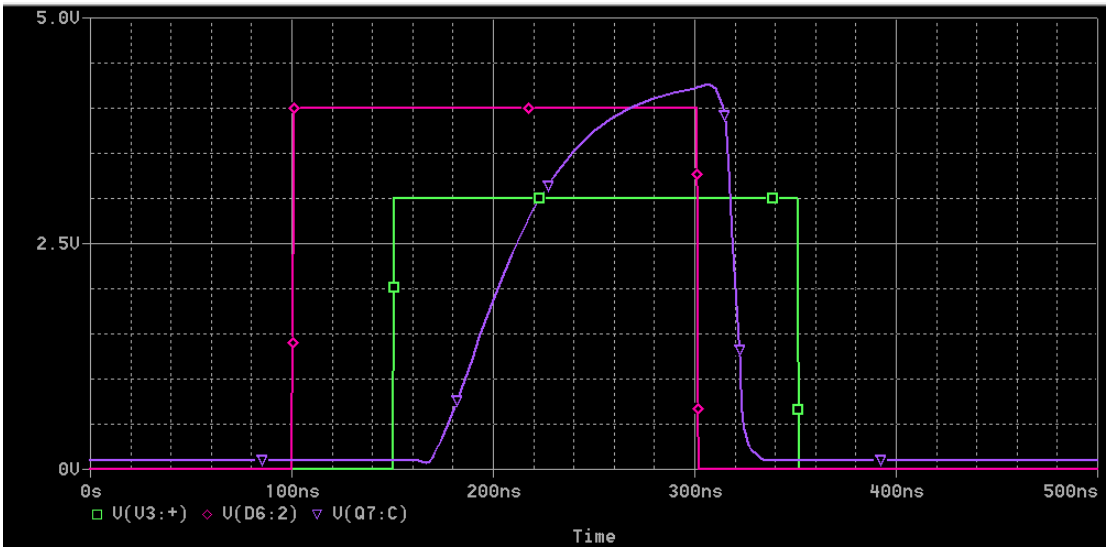

Рисунок 4 – Графики, построенные в PSpice

Таким образом, можно сделать вывод, что программные симуляторы широко применяются в образовательной сфере, а также на производствах. Они позволяют отладить программу после написания, не задействуя настоящее оборудование. Однако они не могут заменить это оборудование, так как не эмулируют его внутреннюю структуру и логику.

1.4 Системы удалённого доступа

Система удалённого доступа к оборудованию предполагает наличие в специально обустроенной лаборатории оборудования, имеющего подключение к сети, с возможностью удалённо управлять этим оборудованием. С точки зрения функционала, предоставляемого пользователям, система удалённого доступа во многом аналогична виртуальной лаборатории или программному симулятору, за исключением

того, что в случае системы удалённого доступа удалённый пользователь может следить за поведением и состоянием реального оборудования, с помощью датчиков либо камер, подключенных к сети. Таким образом, при использовании системы в образовательном учреждении, студент получает навыки работы с реальным оборудованием, даже находясь на дистанционном обучении, что является несомненным плюсом.

В настоящее время системы удалённого доступа к лабораторному оборудованию присутствуют в учебном процессе некоторых университетов России. В частности, в Высшей Школе Экономики производится разработка и наладка программно-аппаратного комплекса для обеспечения удаленного доступа и управления оборудованием учебной лаборатории систем автоматизированного проектирования [11]. Также лаборатории, оснащённые системами удалённого доступа, есть в Новосибирском Государственном Университете и в Национальном исследовательском ядерном университете «МИФИ».

Лаборатории, оснащённые системами удалённого доступа, как правило, разрабатываются для решения конкретных задач, стоящих перед организацией. Программное обеспечение для осуществления удалённого доступа тоже разрабатывается для каждой конкретной системы. В связи с этим не существует универсального решения для систем удалённого доступа – они могут быть сконфигурированы и настроены по-разному. В статье «Организация многопользовательского удаленного доступа к распределенной гетерогенной системе лабораторного оборудования на основе схем программируемой логики для дистанционных практикумов по цифровой схемотехнике» [1] авторы Ёхин М.Н., и Степанов М.М. приводят таблицу, где описаны различные, наиболее перспективные, варианты конфигурации таких систем (табл. 1).

Таблица 1 – Варианты конфигурации оборудования лаборатории с удаленным доступом

	Без рабочих станций	Одна рабочая станция – один стенд	Одна рабочая станция – ассоциированная группа стендов	Одна рабочая станция – все стенды
	Вариант 1	Вариант 2	Вариант 3	Вариант 4
Сервер – рабочая станция	–	Локальная сеть	Локальная сеть	Локальная сеть
Рабочая станция – стенд	–	Точка-точка	Локальная сеть (ассоциированная группа стендов)	Локальная сеть (все стенды лаборатории)
Сервер – стенд	Точка-точка	Точка-точка	Локальная сеть	Локальная сеть
Стенд – стенд (опционально)	Полевая шина /Точка-точка	Полевая шина /Точка-точка	Полевая шина /Точка-точка	Полевая шина /Точка-точка

1.5 Выводы

Проведён анализ различных вариантов решения проблемы удалённого доступа к лабораторному оборудованию. Рассмотрены виртуальные лаборатории, программные симуляторы и системы удалённого доступа. Выявлены их преимущества и недостатки. Это позволило уточнить и сформулировать основные задачи дальнейшей работы:

1. Разработать архитектуру системы удаленного управления платой STK500.

2. Разработать принципы взаимодействия и протоколы нижнего уровня обмена данными между элементами системы доступа.

3. Выполнить анализ и обоснованный выбор требуемого аппаратного обеспечения для создания лабораторного стенда.

4. Выполнить сборку, монтаж и подключение сетевых и электрических соединений стенда.

5. Выполнить интеграцию программного обеспечения (ПО) производителя STK500 для прошивки загрузочных кодов в стенд.

6. Разработать алгоритмическое и программное обеспечение управляющей платы согласно задания на ВКР.

7. Разработать примеры лабораторных работ для тестирования встроенных узлов стенда (светодиодов и кнопок).

8. Разработать скрипты для серверного ПО.

9. Выполнить тестирование и отладку разработанного аппаратного и программного обеспечения.

Сформированный перечень задач позволяет перейти к разработке архитектуры системы удаленного доступа.

2 Разработка системы удалённого управления платой STK500

2.1 Обзор поставленной задачи и оборудования

Согласно задания на ВКР рассматривается задача разработки подсистемы удалённого управления платой STK500 для системы удалённого доступа к лабораторному оборудованию.

STK-500 (рис. 5) – отладочный модуль с интегрированным программатором, предназначенный для разработки и отладки устройств на МК AVR [12]. На лабораторном стенде установлен микропроцессор ATmega32A. Этот микропроцессор имеет четыре порта ввода-вывода, к которым можно подключить различные периферийные устройства лабораторного стенда, такие, как кнопки, светодиоды, аналого-цифровой преобразователь (АЦП), или LCD экран.

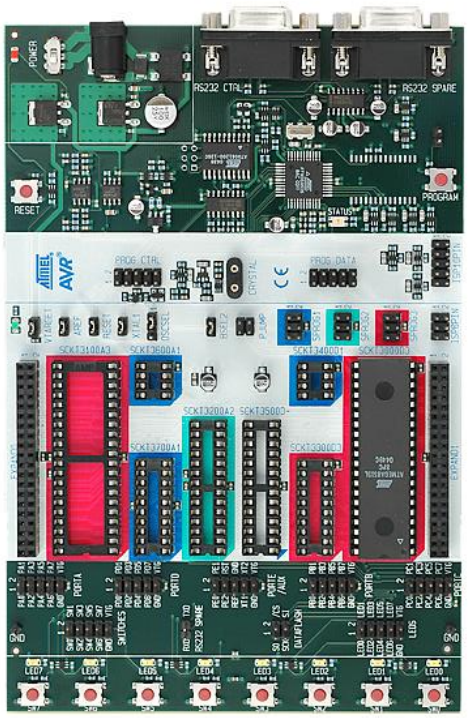

Рисунок 5 – Плата STK500

Для программирования микропроцессора ATmega32A используется

среда программирования Microchip Studio или Atmel Studio. Существует возможность написания программ на языках C/C++ или на языке ассемблера. Также в этих средах программирования есть режим симулятора, что позволяет компилировать программу удалённо, без подключения лабораторного стенда к компьютеру, что может быть полезно при организации удалённого доступа.

Для осуществления удалённого доступа к лабораторному стенду необходимо обеспечить эмуляцию периферийных устройств, используемых пользователем для ввода информации.

Для выполнения лабораторных работ на данном стенде используются все вышеперечисленные периферийные устройства, но эмуляция необходима только для кнопок и резистора переменного сопротивления, который служит регулятором АЦП, так как пользователь напрямую взаимодействует с ними.

Задачу разработки подсистемы удалённого доступа к плате STK500 для системы удалённого доступа к лабораторному оборудованию можно разделить на следующие пункты:

1. разработка структурной схемы разрабатываемой системы;
2. разработка принципов взаимодействия и протоколов нижнего уровня обмена данными между элементами системы доступа;
3. интеграция ПО производителя STK500 для прошивки загрузочных кодов в стенд;
4. анализ и обоснованный выбор требуемого аппаратного обеспечения для создания лабораторного стенда;
5. сборка, монтаж и подключение сетевых и электрических соединений стенда;
6. разработка алгоритмического и программного обеспечения управляющей платы согласно задания на ВКР;
7. разработка примеров лабораторных работ для тестирования встроенных узлов стенда (светодиодов, кнопок, резистора переменного сопротивления);
8. разработка скриптов для серверного ПО.

Подробное описание выполнения данных пунктов представлено ниже.

2.2 Архитектура разрабатываемой системы

Перед началом проектирования всеми участниками проекта по разработке системы удалённого доступа к лабораторному оборудованию была разработана структурная схема системы, представленная на рисунке 6.

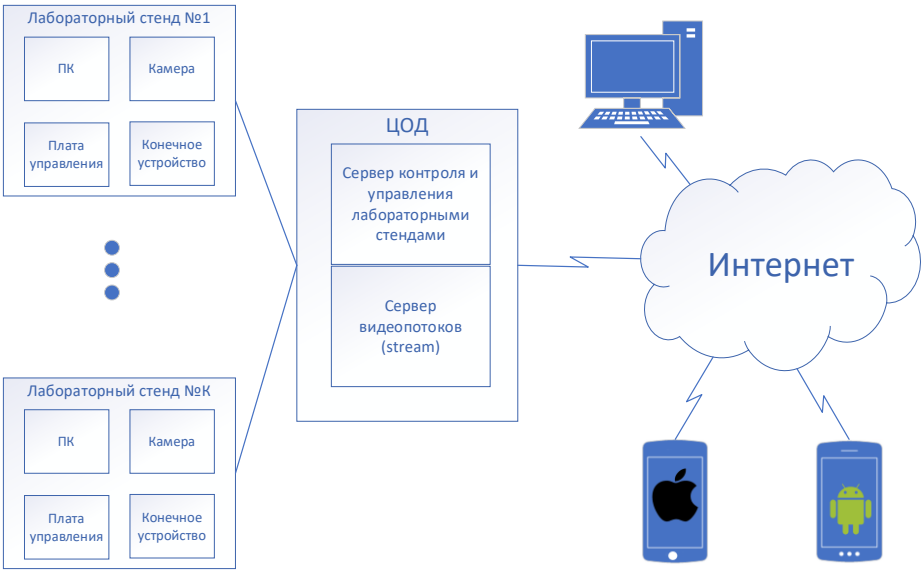

Рисунок 6 – Общая схема системы удалённого доступа к лабораторному оборудованию

Разрабатываемая система имеет комплексную архитектуру, сочетающую в себе большое количество различной аппаратуры и программного обеспечения (ПО).

Главными аппаратными компонентами системы являются:

- конечное устройство пользователя;
- центр обработки данных;
- лабораторный стенд.

Для доступа к лабораторному стенду пользователю необходимо иметь конечное устройство. Конечным устройством пользователя может являться

любой персональный компьютер (ПК) или мобильное устройство на базе операционной системы Android или IOS.

В случае использования в качестве конечного устройства ПК, пользователю достаточно открыть веб-сайт с помощью любого браузера, а при использовании мобильного устройства рекомендуется скачать соответствующее мобильное приложение.

В центре обработки данных запущено два веб-сервера:

- сервер контроля и управления лабораторным стендом;
- сервер видеопотоков (stream).

Сервер управления отвечает за общение с базой данных и передачу данных пользователю, а также за предоставление пользователю доступа к лабораторному стенду.

Сервер видеопотока отвечает за предоставление прямых трансляций с лабораторных стендов. Он выступает в роли прокси сервера и занимается перекодировкой исходных видеопотоков, поступающих по протоколу RTSP, в требуемый формат, который зависит от конечного устройства пользователя. После перекодировки видеопотоки могут передаваться клиентскому приложению по протоколу HLS, который может читаться браузерами на любом устройстве.

Подробная схема лабораторного стенда, задействованного в данной работе, изображена на рисунке 7.

Рисунок 7 – Схема архитектуры стенда

Лабораторный стенд представляет собой систему, состоящую из персонального компьютера, IP-камеры, управляющей платы, модуля ЦАП и конечного устройства – платы STK500.

На персональном компьютере установлен Web-сервер, подключенный к центру обработки данных. К компьютеру подключена IP-камера, управляющая плата, плата STK500 и установлены драйвера для них. Также на нём находятся скрипты для управления устройствами стенда.

IP-камера необходима для того, чтобы пользователь мог наблюдать за лабораторным стендом в режиме реального времени. В камере имеется встроенное программное обеспечение, позволяющее осуществить эту функцию.

Плата управления и подключенный к ней модуль цифро-аналогового преобразователя (ЦАП) необходимы для эмуляции периферийных устройств платы STK500, что описывается ниже. Сама плата STK500 является конечным устройством стенда.

Взаимодействие элементов разрабатываемой системы между собой можно представить в виде следующей схемы (рисунок 8):

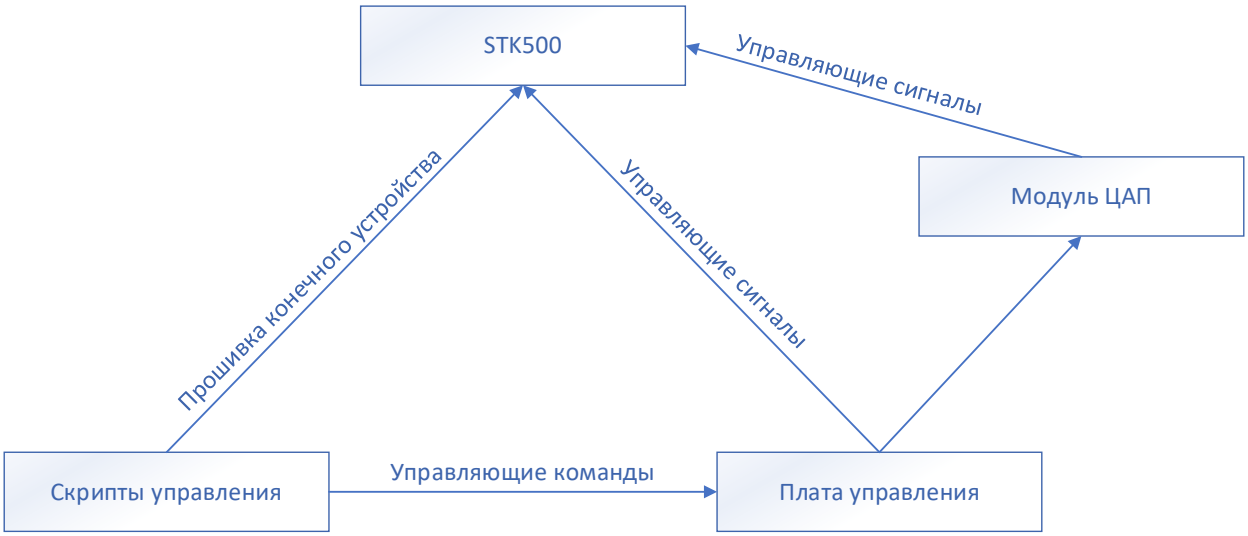

Рисунок 8 – Взаимодействие элементов системы

Плата управления подключена к STK500 для управления её периферийными устройствами (кнопки, резистор переменного сопротивления). Для управления аналоговым вводом к управляющей плате и к STK500 также подключен модуль ЦАП, позволяющий преобразовывать поступающие на него цифровые значения в аналоговые. На плату управления поступают управляющие команды при помощи управляющих скриптов. Эти команды после обработки управляющей платой преобразуются в управляющие сигналы для платы STK500 и подаются на неё напрямую либо через ЦАП. Также при помощи управляющих скриптов осуществляется прошивка платы.

2.3 Разработка аппаратной части системы удалённого управления платой STK500

2.3.1 Анализ и выбор аппаратного обеспечения для лабораторного стенда

На первых этапах разработки в качестве платы управления рассматривались платы Arduino Uno, Arduino Mega и Arduino Nano, в их

основе которых находятся микропроцессоры ATmega, которые возможно программировать на языке C++. Также эти платы являются достаточно надёжными в эксплуатации и с ними совместимо множество различных периферийных устройств, что может быть полезным для расширения функционала лабораторного стенда.

В результате анализа и сравнения трёх версий Arduino для дальнейшей работы была выбрана плата Arduino Uno, поскольку она обладает достаточным количеством портов ввода-вывода для подключения к STM32, а также дальнейшего расширения функционала стенда, например, подключения различных датчиков. Также у этой платы достаточно памяти для хранения прошивки и данных. Arduino Mega имеет слишком много портов ввода-вывода, а Arduino Nano, напротив, недостаточно для дальнейшего расширения конструкции лабораторного стенда.

Для работы с резистором переменного сопротивления и управления им в режиме удалённого доступа необходимо подавать на него аналоговые значения, которые можно получить, подавая цифровые значения на модуль ЦАП через плату управления. Для дальнейшей работы был выбран модуль MCP4725, подключаемый к Arduino по интерфейсу I2C. Это 12-разрядный модуль, который получает с аналоговых выходов Arduino A4 и A5 числовые значения и преобразует их в напряжение, подаваемое на выход Out.

2.3.2 Подключение Arduino Uno к STK500

Для эмуляции восьми кнопок понадобится восемь цифровых выходов, а для эмуляции АЦП понадобится модуль ЦАП, подключаемый к Arduino по протоколу I2C и аналоговый выход.

Перед началом подключения необходимо убедиться, что выходные напряжения на обеих платах одинаковы. На STK500 оно составляет 5В, как и на Arduino Uno, из чего следует, что платы можно безопасно соединять друг с другом.

После проверки выходных напряжений необходимо соединить выходы GND лабораторного стенда и Arduino, подключив, таким образом, общую землю. После этого можно приступать к остальным соединениям.

На лабораторном стенде STK500 имеется восемь кнопок, подключенных к порту PORTD платы STK500, и резистор переменного сопротивления, через который напряжение подаётся на АЦП, подключенный к пину PA7 порта PORTA. Для их эмуляции к порту D вместо физических кнопок были подключены цифровые выходы Arduino – PD2 – PB1. Пины PD0 и PD1 нельзя использовать для подключений, так как они используются для связи Arduino с компьютером по USB.

Чтобы эмулировать работу регулятора АЦП, используется цифро-аналоговый преобразователь (ЦАП), выход Out которого соединён с выходом PA7 STK500, заменяя, таким образом, резистор переменного сопротивления.

Схема подключения Arduino к STK500 показана на рисунке 9. На рисунке 10 показан вид подключения Arduino к STK500.

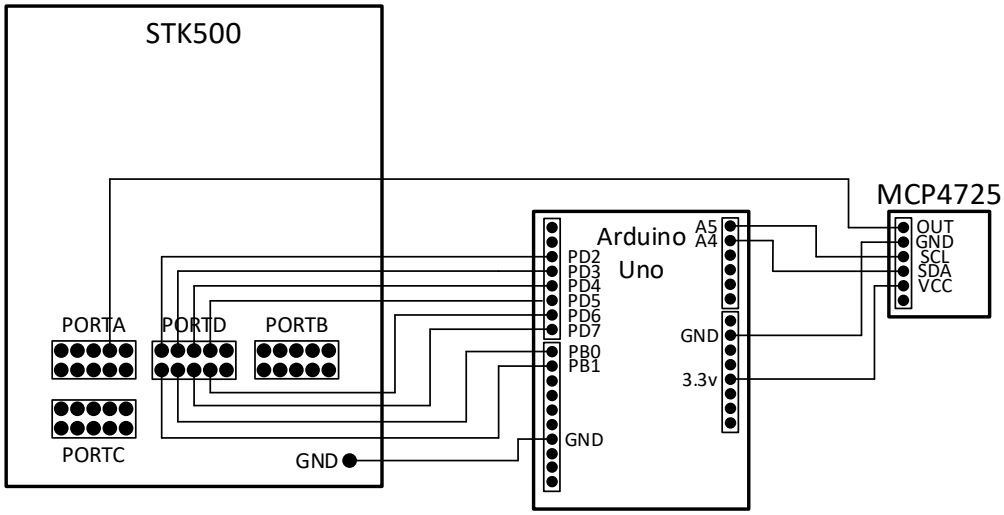

Рисунок 9 – Схема подключения

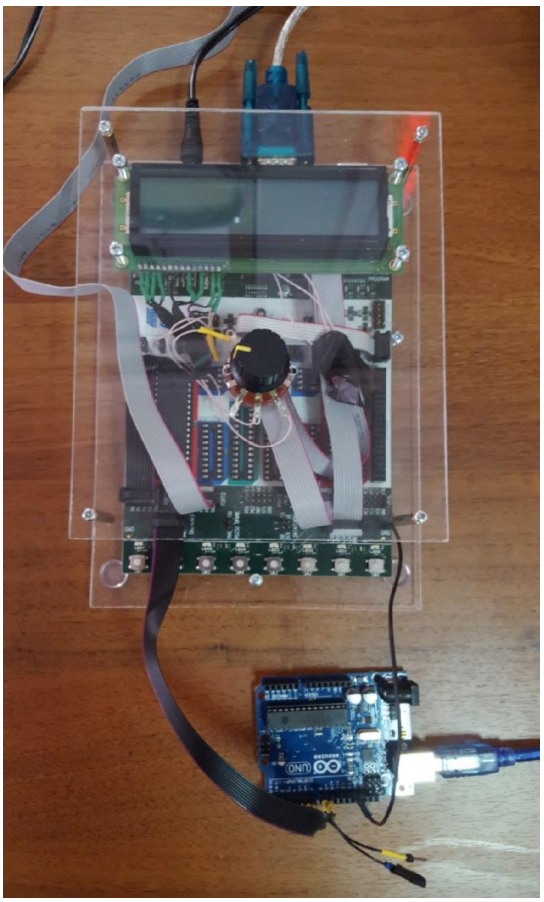

Рисунок 10 – Соединение Arduino и STK500

2.4 Управление периферийными устройствами STK500 в режиме удалённого доступа

Чтобы обеспечить удалённый доступ к периферийным устройствам лабораторного стенда, необходимо управлять Arduino при помощи команд, подаваемых с компьютера через командную строку.

Управление Arduino Uno R3 с компьютера осуществляется через Serial Port (COM-порт). Номер порта, к которому подключена Arduino, можно узнать в Arduino IDE (меню «Инструменты» – Порт), или через диспетчер устройств. Нужный порт будет обозначен как Arduino Uno.

Чтобы плата управления могла распознавать и обрабатывать передаваемую ей информацию, была разработана программа для Arduino Uno, являющаяся драйвером конечного устройства. Программа написана на языке

C++. В приложении А представлен листинг программы.

При запуске программы происходит инициализация, в результате которой цифровые выходы Arduino PD2 – PB1 переводятся в режим вывода (OUTPUT), для вывода управляющих сигналов для STK500. Также при помощи метода Serial.begin инициируется последовательное соединение с ПК по интерфейсу Serial. После инициализации программа входит в бесконечный цикл, в котором отслеживает приходящие через Serial-порт символы в виде байтов. При обнаружении ключевой последовательности байтов comm на цифровых выходах PD2 – PB9 устанавливаются значения в виде напряжений низкого или высокого уровня, в зависимости от полученной управляющей команды, которая является последовательностью нулей и единиц, соответствующих отдельным эмулируемым устройствам. В случае, если полученный байт равен «1», значение на соответствующем цифровом выходе Arduino меняется на противоположное. Если же был получен другой символ, состояние выхода останется неизменным. Важно отметить, что напряжение низкого уровня, установленное на цифровом выходе, служит сигналом для замыкания соответствующей ему кнопки на плате STK500, тогда как напряжение высокого уровня – сигналом для её размыкания.

После последовательности из восьми нулей и единиц в команде через пробел следует последовательность из пяти символов, где первый символ – «0» или «1», а остальные – любые цифры. Эта последовательность отвечает за работу с ЦАП. Если первый символ в этой последовательности равен «1», ЦАП используется, и программа обработает идущие следом цифры, установив полученное число в качестве значения ЦАП. Для работы с ЦАП применяется библиотека Adafruit_MCP4725.

На рисунке 11 приведён алгоритм работы программы для Arduino Uno.

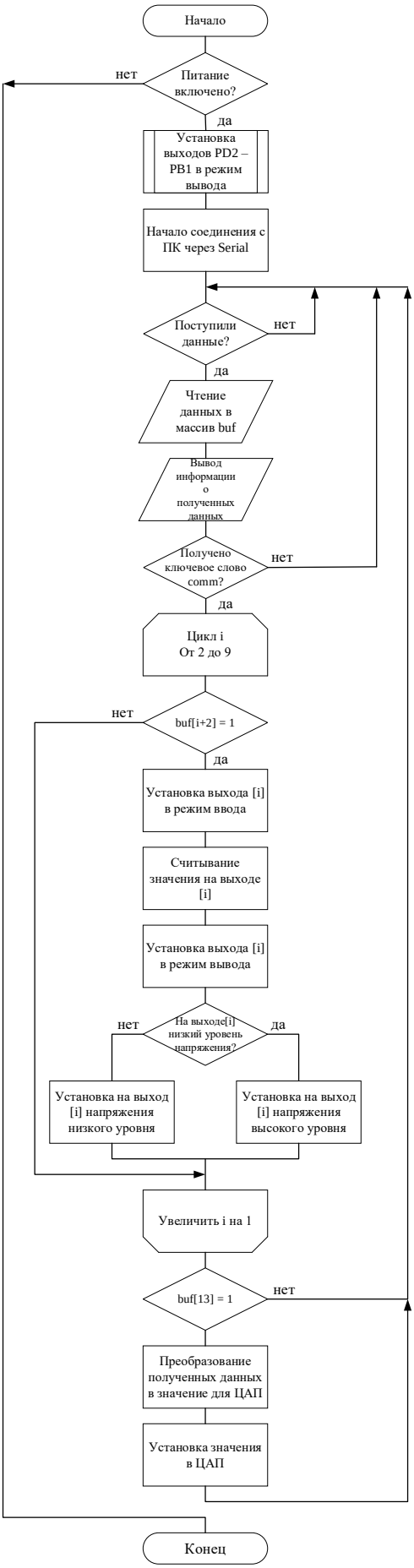

Рисунок 11 – Алгоритм работы программы

На рисунке 12 приведён формат управляющей команды, вводимой в командную строку для эмуляции периферийных устройств STK500.

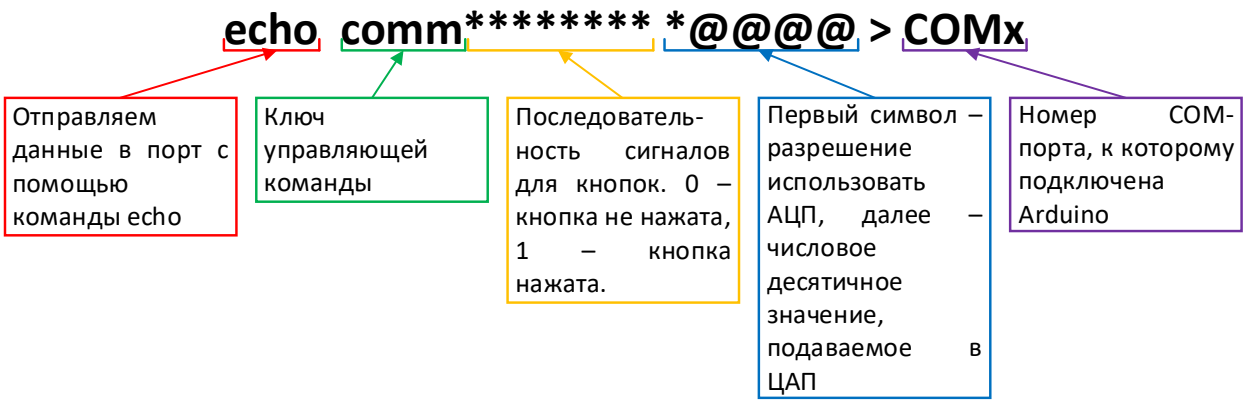

Рисунок 12 – Формат команды управления Arduino

Данные в COM-порт передаются с помощью стандартной команды:

echo [данные] > COMx,

где «x» – номер порта, в который нужно передать информацию, а «[данные]» – передаваемая информация. В нашем случае данными является строка формата

comm****** *@@@@**

Здесь comm – ключ команды, позволяющий избежать выполнения программы в случае подачи в порт мусорной информации или шума.

Символом «*» обозначены байты, которые могут принимать значения «0» или «1». Символом «@» обозначены байты, способные принимать значения от «0» до «9».

«********» – последовательность нулей и единиц, соответствующая кнопкам на лабораторном стенде STK-500. После обработки программой Arduino эти данные поступают на цифровые выходы PD2 – PB1 в виде напряжения высокого или низкого уровня. При этом «0» будет соответствовать напряжению высокого уровня, то есть кнопка не будет нажата, а «1» – напряжению низкого уровня, то есть нажатию кнопки.

Программа для Arduino написана таким образом, что кнопки эмулируются «с удержанием» – если подать «1» на не нажатую кнопку, она будет нажата и продолжит оставаться в таком режиме до следующего поступления на неё «1». Это не соответствует имеющимся на плате STK500 физическим кнопкам, работающим без удержания, но необходимо для выполнения студентами лабораторной работы №3 по курсу «Микропроцессорные системы. Часть 1», где требуется контролировать нажатие и отпускание кнопок для отслеживания нарастающего и спадающего фронтов сигнала.

Последовательность «*@@@@», отделённая от предыдущей последовательности пробелом, несёт в себе информацию, предназначенную для управления модулем ЦАП, подключенным к Arduino. Первый символ «1» или «0» сигнализирует о необходимости соответственно использования или не использования ЦАП и АЦП в текущей работе со стендом. Остальные четыре символа, принимающие значения от «0» до «9», несут в себе число в десятичном формате, которое при обработке программой Arduino подаётся на входы SDA и SCL модуля ЦАП по интерфейсу I2C. Это число может принимать любые значения от 32 до 4095 включительно. Нижняя граница выбрана таким образом, чтобы напряжение, подаваемое на АЦП STK-500 было достаточным и не возникало обратных токов из-за недостатка напряжения. Верхняя граница установлена в соответствии с разрядностью модуля MCP4725: он 12-разрядный.

Если число, подаваемое в ЦАП, содержит меньше четырёх разрядов, в старшие разряды необходимо дописать нули. Например, если подаётся десятичное число 64, и в работе со стендом необходимо использовать АЦП, последовательность байт после пробела должна выглядеть как «10064».

2.5 Прошивка платы STK500 в режиме удалённого доступа

Для выполнения студентами лабораторных работ в режиме удалённого доступа необходимо предусмотреть возможность запускать написанные ими программы для микропроцессора ATmega32A на лабораторном стенде удалённо и вне сред Atmel Studio или Microchip Studio, работа в которых может производиться только лично пользователем. В результате поиска в сети Интернет был найден существующий способ запуска программы для микропроцессора через командную строку с помощью hex-файла.

Intel HEX — формат файла, предназначенного для представления произвольных двоичных данных в текстовом виде. По историческим причинам является стандартом при прошивке разнообразных микросхем с памятью (микроконтроллеров, ПЗУ, EEPROM и т. п.).

Способ запуска программы для микропроцессора через командную строку с помощью hex-файла представляет собой утилиту «STK500», запускаемую из командной строки [16]. Данная утилита устанавливается вместе с AVR Studio 4.19 (build 730) и после установки по умолчанию располагается по адресу C:\Program Files (x86)\Atmel\AVR Tools\STK500

Файл нужного формата (hex-файл) создаётся автоматически при компиляции программы в Atmel Studio или Microchip Studio, независимо от того, подключено ли к компьютеру физическое устройство или для компиляции используется симулятор. Это позволяет студенту, выполняющему работу удалённо, самостоятельно сформировать данный файл. После завершения компиляции программы hex-файл можно найти в папке Debug, расположенной в папке текущего проекта Microchip Studio.

Чтобы утилита распознала hex-файл и смогла прошить его в микропроцессор, файл должен находиться в папке с утилитой либо в её подкаталоге. Таким образом, готовые hex-файлы, поступающие на компьютер в режиме удалённого доступа, следует сохранять в нужную папку.

Для прошивки программы в микропроцессор необходимо в командной

строке перейти в папку, где располагается утилита STK500 и затем прописать команду формата, представленного на рисунке 13.

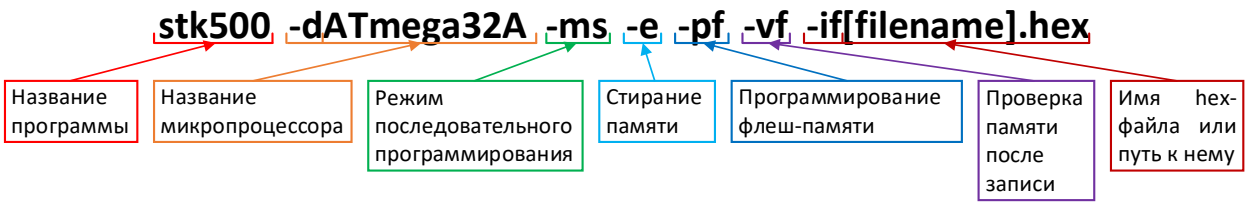

Рисунок 13 – Формат команды для прошивки программы в микропроцессор

В случае, если необходимо перезапустить загруженную студентом программу, выполняется та же команда, которая перепрошивает программу в микропроцессор.

Полный список параметров утилиты STK500 приведён в таблице 2.

Таблица 2 – Параметры утилиты STK500

d		Наименование МК. Указывается при программировании. См. список ниже.
m		Режим программирования
	s	Последовательный (по умолчанию)
	p	Параллельный
if		Имя hex-файла для записи во флэш-память
ie		Имя hex-файла для записи в ЭСППЗУ
of		Имя записываемого hex-файла при чтении из флэш-памяти
oe		Имя записываемого hex-файла при чтении из ЭСППЗУ
s		Чтение сигнатуры
O		Чтение калибровочного байта встроенного генератора
Sf	addr	Запись калибровочного байт во флэш-память по адресу «addr»
Se	addr	Запись калибровочного байт в ЭСППЗУ по адресу «addr»
e		Стирание памяти
p		Программирование
	f	флэш-памяти
	e	ЭСППЗУ
	b	флэш-памяти и ЭСППЗУ
r		Чтение
	f	флэш-памяти
	e	ЭСППЗУ
	b	флэш-памяти и ЭСППЗУ
v		Проверка после записи
	f	флэш-памяти
	e	ЭСППЗУ

	b	флэш-памяти и ЭСППЗУ
l	value	Установка байта защиты по значению «value» в 8-разрядном 16-ричном формате (00-FF)
L	value	Сравнение байта защиты с значением «value» в 8-разрядном 16-ричном формате (00-FF)
f	value	Установка конфигурационных байт по значению «value» в 16-разрядном 16-ричном формате (0000-FFFF)
E	value	Установка расширенного конфигурационного байта по значению «value» в 8-разрядном 16-ричном формате (00-FF)
F	value	Сравнение конфигурационных байт со значением «value» в 16-разрядном 16-ричном формате (0000- FFFF)
G	value	Сравнение расширенного конфигурационного байта со значением «value» в 8-разрядном 16-ричном формате (00- FF)
q		Чтение конфигурационных байт
x	0x00-0xff	Заполнить неиспользуемое пространство памяти значением 0x00-0xff. По умолчанию свободное пространство не программируется
af	start,stop	Диапазон адреса флэш-памяти. Определяет диапазон адреса для дальнейших действий (по умолчанию вся флэш-память)
ae	start,stop	Диапазон адреса ЭСППЗУ. Определяет диапазон адреса для дальнейших действий (по умолчанию все ЭСППЗУ)
c	com1… com8	Выбор коммуникационного порта. По умолчанию ищется порт с подключенным STK500
ut	0.0-6.0	Установка целевого напряжения VTARGET в вольтах.
ua	0.0-6.0	Установка опорного аналогового напряжения AREF в вольтах.
wt		Считать текущее значение VTARGET
wa		Считать текущее значение AREF
b		Определить код версии
	h	аппаратного обеспечения
	s	программного обеспечения
!	0…3690000	Установка частоты тактового генератора в Гц
t		Считывание частоты тактового генератора
g		Работа без выдачи сообщений
z		Не показывать индикатор прогресса
h или ?		Вызов справки (перечисление вышеописанных параметров)

Для вызова помощи необходимо выполнить «STK500 -?» или «STK500 -h».

Программа возвращает ERRORCODE 0, если указанные действия выполнены успешно или ERRORCODE 1, если действие прервано. На рисунке 14 представлен скриншот работы утилиты.

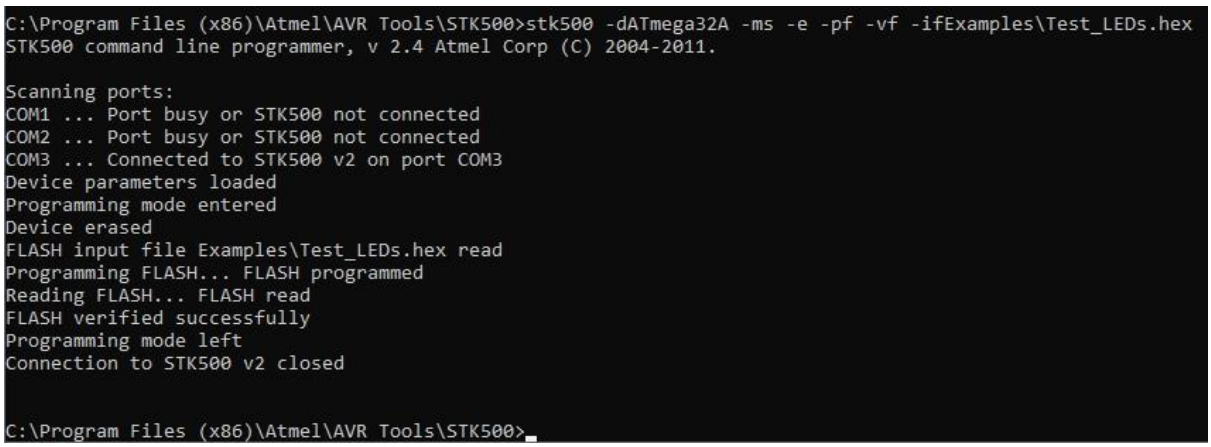

Рисунок 14 – Скриншот работы утилиты

2.6 Разработка управляющих скриптов для автоматизации ввода управляющих команд

На рисунках 12 и 13 настоящей пояснительной записки приведены форматы двух управляющих команд, необходимых для управления конечным устройством лабораторного стенда. Чтобы автоматизировать ввод управляющих команд в командную строку, были разработаны скрипты на языке Python, которые, получая в качестве параметров данные для команд, формируют их и направляют в командную строку. Листинги скриптов приведены в приложении А.

В качестве средства разработки был выбран язык Python, так как он удобен для работы с командами для командной строки, а также является достаточно гибким инструментом, подходящим для решения подобных задач.

2.6.1 Разработка скрипта для программирования STK500

Скрипт STK_prog.py был разработан для автоматизации ввода команды прошивки STK500. В качестве входного параметра скрипт получает имя hex-файла, находящегося в папке C:\Scripts. Путь к папке не нужно передавать вместе с именем файла.

Скрипт определяет местное время и дату при помощи средств

стандартной библиотеки time и выводит эту информацию в файл C:\Scripts\Log.txt. Затем получает входной параметр и выполняет подстановку имени файла в строку команды. Сформированная команда направляется для выполнения в командную строку. Вывод утилиты STK500 также записывается в файл C:\Scripts\Log.txt. Скрипт запускается из командной строки командой

python C:\Scripts\ STK_prog.py [filename],

где [filename] – имя hex-файла вместе с расширением.

2.6.2 Разработка скрипта для управления периферийными устройствами STK500

Скрипт STK_but_adc.py был разработан для автоматизации ввода команды для управления периферийными устройствами платы STK500. Скрипт получает в качестве входных параметров последовательность из восьми нулей и единиц, обозначающих состояние кнопок и пять цифр, первая из которых обозначает, используется ли ЦАП или АЦП, а четыре других несут информацию для передачи в модуль ЦАП.

После получения входных параметров скрипт определяет местное время и дату при помощи средств стандартной библиотеки time и выводит эту информацию в файл C:\Scripts\Log.txt. Затем формирует команду для управления Arduino, передаёт её в командную строку, и записывает информацию о переданной команде и времени её запуска в файл C:\Scripts\Log.txt. Скрипт запускается из командной строки командой

python C:\Scripts\ STK_but_adc.py ****** *@@@@**,

где входные параметры аналогичны таковым для управляющей команды Arduino.

На рисунке 15 представлен алгоритм работы данного скрипта.

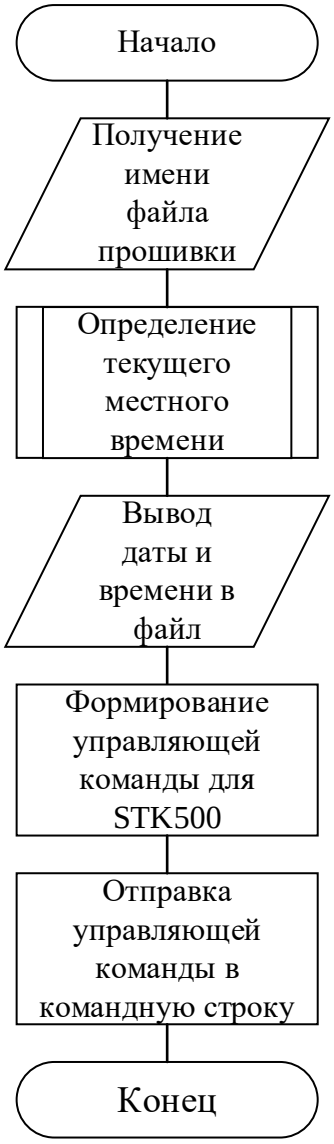

Рисунок 15 – Алгоритм работы скрипта STK_but_adc.py

2.6.3 Разработка скрипта для очистки памяти платы

Скрипт STK_clean.py должен вызываться в конце каждой сессии пользователя. Он очищает папку C:\Scripts от всех находящихся в ней файлов с расширением .hex и запускает утилиту STK500 с параметрами, необходимыми для очистки памяти микропроцессора. У этого скрипта нет входных параметров. После запуска он определяет текущие местные время и дату, выводит их в файл C:\Scripts\Log.txt, затем формирует команды для очистки памяти платы STM32 и её перезагрузки и поочерёдно направляет их в командную строку, после чего удаляет все bin-файлы, находящиеся в

каталоге C:\Scripts. Он запускается из командной строки командой

python C:\Scripts\ STK_clean.py

На рисунке 16 представлен алгоритм работы данного скрипта.

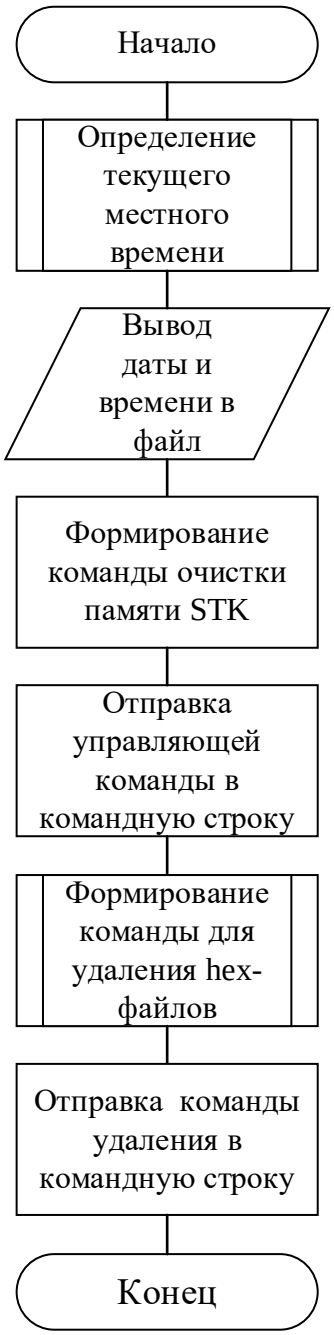

Рисунок 16 – Алгоритм работы скрипта STK_clean.py

Скрипты хранятся в папке C:\Scripts. В той же папке хранится и текстовый файл Log.txt, содержащий информацию о последней выполненной команде.

2.7 Разработка тестовых программ для STK500

Для тестирования правильности работы разработанный системы, были написаны две тестовые программы для платы STK500, которые соответствуют лабораторным работам по курсу «Микропроцессорные системы». Первая программа принимает сигналы от кнопок и зажигает соответствующий нажатой кнопке светодиод на плате STK500. Вторая программа считывает напряжение, приходящее с модуля ЦАП и устанавливает его на АЦП. АЦП обрабатывает полученную информацию и выводит её на светодиоды на плате STK500. Данные программы разработаны для микропроцессора ATmega32A на языке ассемблера. Листинги программ приведены в приложении Б. Примеры разработки программ в интегрированной среде STMCube приведены на рисунках 17 и 18.

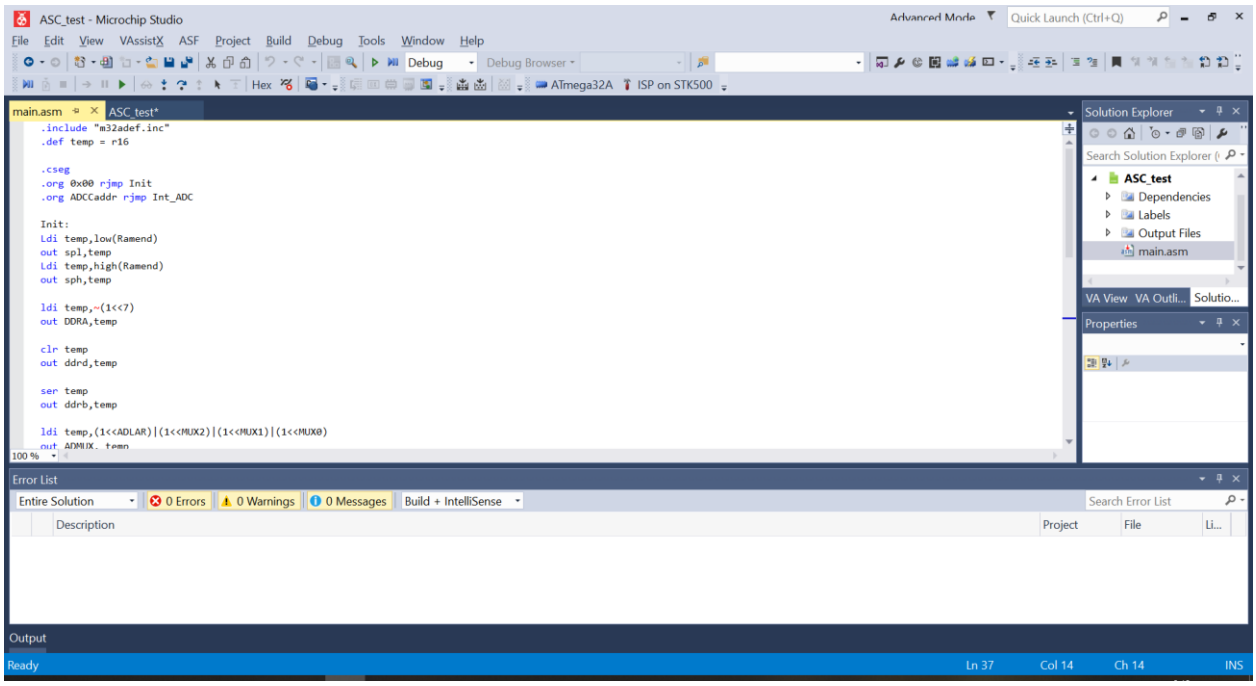

Рисунок 17 – Разработка программы лабораторной работы №2

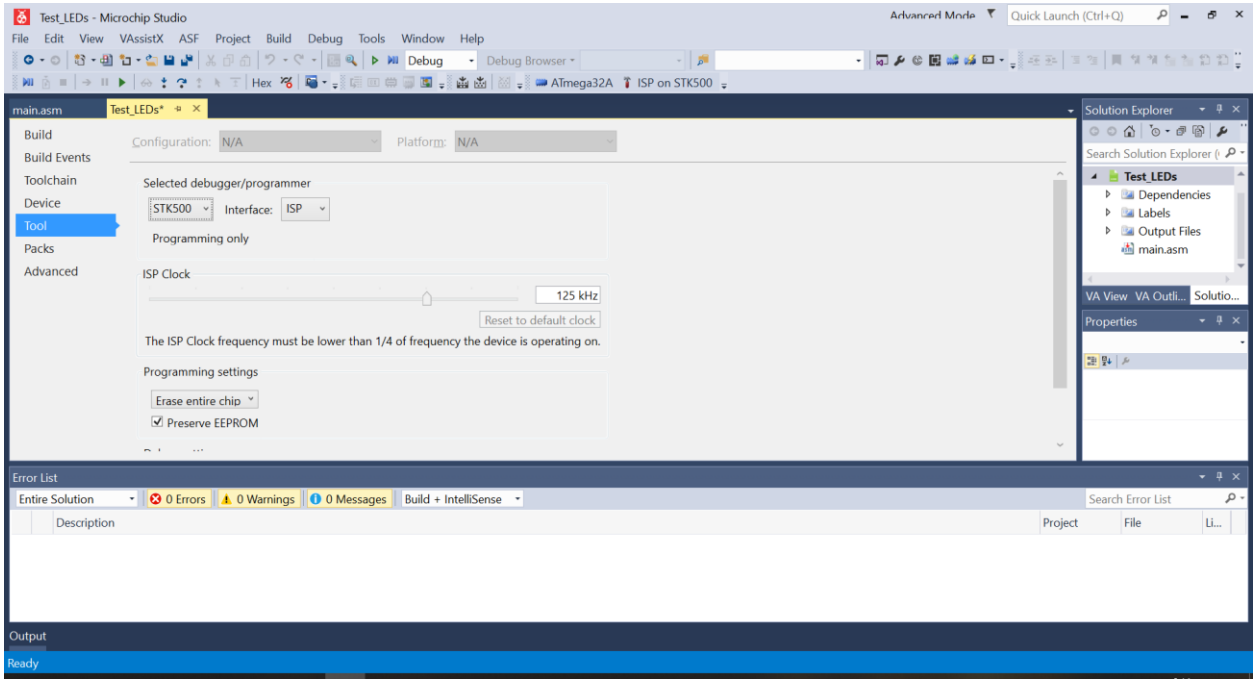

Рисунок 18 – Отладка программы лабораторной работы №1

2.8 Выводы

В результате разработки подсистемы удалённого управления платой STK500 для системы удалённого доступа к лабораторному оборудованию была разработана структурная схема лабораторного стенда. В соответствии с полученной схемой был собран лабораторный стенд. Для конечного устройства стенда было найдено решение для управления устройством через командную строку. Была разработана система эмуляции периферийных устройств платы STK500 при помощи Arduino Uno, написаны управляющие скрипты для автоматизации формирования и запуска управляющих команд и разработаны программы для дальнейшего тестирования системы.

Таким образом, после завершения всех этапов разработки была завершена работа над программной и аппаратной частями системы удалённого управления платой STK500, что позволяет перейти к тестированию разработанной системы.

3 Тестирование разработанной системы

3.1 Монтаж лабораторного стенда

После того, как был завершён этап разработки подсистемы удалённого доступа к плате STK500 для системы удалённого доступа к лабораторному оборудованию, была проведена сборка лабораторного стенда в соответствии с разработанными структурной схемой и схемой подключения (рис. 7, 9). Был выполнен монтаж стойки, на которой был размещён стенд. На рисунках 19, 20 представлен вид собранного стенда.

Рисунок 19 – Общий вид собранного стенда

Рисунок 20 – Конечное устройство стенда

3.2 Тестирование без участия сервера

В результате разработки была получена система, позволяющая эмулировать периферийные устройства лабораторного стенда STK500 и управлять ими, а также прошивать программу в микропроцессор лабораторного стенда без сред разработки. Данная система была протестирована при помощи тестовых программ для микропроцессора ATmega32A, листинги которых приведены в приложении Б.

Обе тестовые программы были скомпилированы в программе Microchip Studio для получения hex-файлов, после чего эти файлы были помещены в папку C:\Scripts и прошиты в микропроцессор при помощи утилиты STK500, вызванной с помощью консольной команды stk500. После этого стенд работал в соответствии с прошитой программой.

Для проверки работы кнопок использовалась первая тестовая программа (таблица 1 приложения Б). После ввода в командную строку команды echo comm11111111 00000 > COM3 на стенде поочерёдно зажглись все светодиоды, так как кнопки были поочерёдно нажаты и оставались в таком

положении до повторного ввода той же команды, которая привела все кнопки в начальное положение.

Для проверки работы АЦП использовалась вторая тестовая программа (таблица 2 приложения Б). В командную строку несколько раз вводилась команда echo comm00000000 1**** > COM3, где символ * принимал значения от 0 до 9. После каждого ввода команды светодиоды на стенде загорались в соответствии с переданным в ЦАП значением.

После успешных тестов без использования скриптов на языке Python обе программы были поочерёдно прошиты в микропроцессор при помощи скрипта STK_prog.py. Программы успешно запустились, а в файле log.txt появился отчёт о выполнении команды. Затем были повторены предыдущие тесты, с тем отличием, что управляющие команды подавались при помощи скрипта STK_but_adc.py. Тесты показали аналогичные результаты. После выполнения каждой команды файл log.txt успешно обновлялся и всегда содержал информацию о последней выполненной команде.

После завершения тестов работы периферийных устройств лабораторного стенда STK500 был запущен скрипт STK_clean.py. В результате его работы из папки C:\Scripts были удалены все файлы, а память микропроцессора успешно очищена.

Таким образом, все проведённые тесты показали положительный результат.

3.3 Тестирование при участии сервера

Следующим этапом тестирования разработанной системы стало тестирование с участием сервера. На ПК лабораторного стенда был установлен сервер Node.js с API, разработанным рабочей группой. Доступ к API осуществлялся с помощью браузера на ПК лабораторного стенда по адресу localhost:3000/api. По данному адресу располагается список всех маршрутов для взаимодействия с лабораторными стендами. Для стенда,

конечным устройством которого является плата STK500, предусмотрены следующие маршруты (рис. 21):

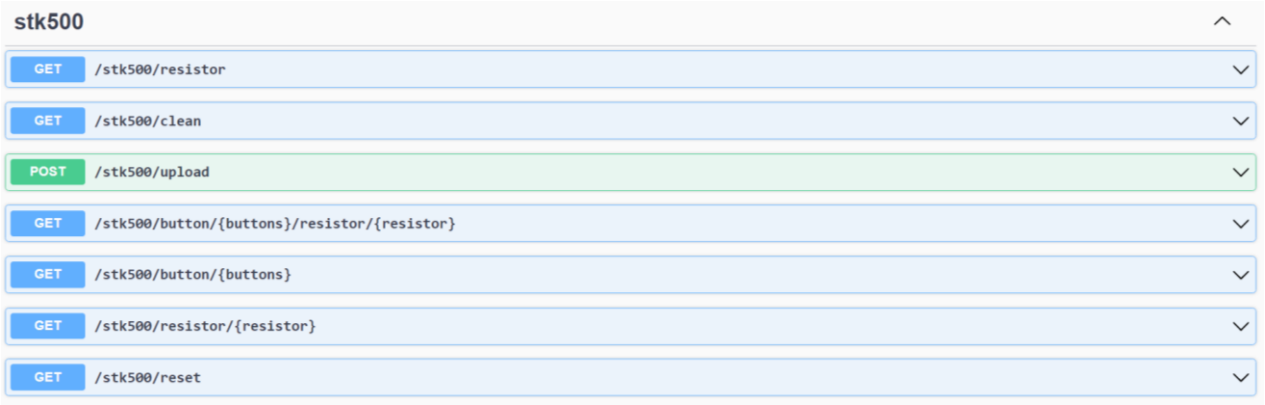

Рисунок 21 – Список маршрутов

1. /stk500/resistor – get-запрос, использующийся для получения информации об эмулируемом резисторе переменного сопротивления.
2. /stk500/clean – get-запрос, используемый для удаления загруженных пользователем на сервер файлов прошивки платы, а также очистки памяти микропроцессора. Вызывает скрипт STK_clean.py;
3. /stk500/upload – post-запрос, используемый для загрузки на сервер hex-файлов и их прошивки в конечное устройство. Вызывает скрипт STK_prog.py с именем загруженного файла в качестве входного параметра;
4. /stk500/buttons/{buttons}/resistor/{resistor} – get-запрос для управления периферийными устройствами платы STK500. {buttons} и {resistor} – данные о состоянии кнопок и резистора. Формат этих данных соответствует формату входных параметров скрипта STK_but_adc.py, который вызывается по данному запросу;
5. /stk500/button/{buttons} – get-запрос, используемый для управления кнопками. Аналогичен предыдущему запросу, с тем исключением, что в данном запросе отсутствует информация о резисторе переменного сопротивления. Данный запрос вызывает скрипт STK_but_adc.py с данными о кнопках, полученными в запросе, и нулями вместо данных о резисторе;

6. /stk500/resistor/{resistor} – get-запрос, используемый для управления резистором переменного сопротивления. Аналогичен предыдущему запросу, с отличием, что в данном запросе вместо информации о кнопках передаётся информация о резисторе. Данный запрос вызывает скрипт STK_but_adc.py с данными о резисторе, полученными в запросе, и нулями вместо данных о кнопках;

7. /stk500/reset – get-запрос, используемый для перепрошивки конечного устройства файлом, ранее загруженным пользователем на сервер. Вызывает скрипт STK_prog.py с именем загруженного файла в качестве входного параметра;

После загрузки hex-файла с помощью маршрута /stk500/upload он сохраняется в папке C:\Scripts. После этого запускается скрипт STK_prog.py с именем загруженного файла в качестве входного параметра. Скрипт выполняется аналогично предыдущим тестам: прошивка платы завершается успешно, отчёт об этом сохраняется в файле Log.txt. Содержимое этого файла также отображается на странице в браузере.

Были проверены маршруты /stk500/buttons/{buttons}/resistor/{resistor}, /stk500/button/{buttons}, и /stk500/resistor/{resistor}. Каждый из них вызывает скрипт STK_but_adc.py с переданными в запросе параметрами. Результаты всех тестов совпали с предыдущими и оказались положительными.

При вызове запроса reset происходит перепрошивка конечного устройства лабораторного стенда ранее загруженным файлом с помощью скрипта STK_prog.py.

По запросу clean выполняется скрипт STK_clean.py, успешно срабатывающий, как и в предыдущих тестах.

Результаты выполнения всех скриптов отображаются в файле C:\Scripts\Log.txt и выводятся на страницу со списком запросов. Результаты всех проведённых тестов совпали с предыдущими результатами, полученными при тестировании системы без сервера.

3.4 Тестирование в режиме «точка-точка»

Заключительным этапом тестирования разработанной системы стало тестирование в режиме «точка-точка». Подключение к серверу производилось с ПК, подключенного к сети СФУ. По адресу http://10.3.3.20:4200/ находится сайт, где представлен пользовательский интерфейс для взаимодействия с лабораторными стендами. После авторизации имеется возможность подключиться к нужному стенду и увидеть изображение с камеры, подключенной к нему. На рисунке 22 представлен интерфейс сайта.

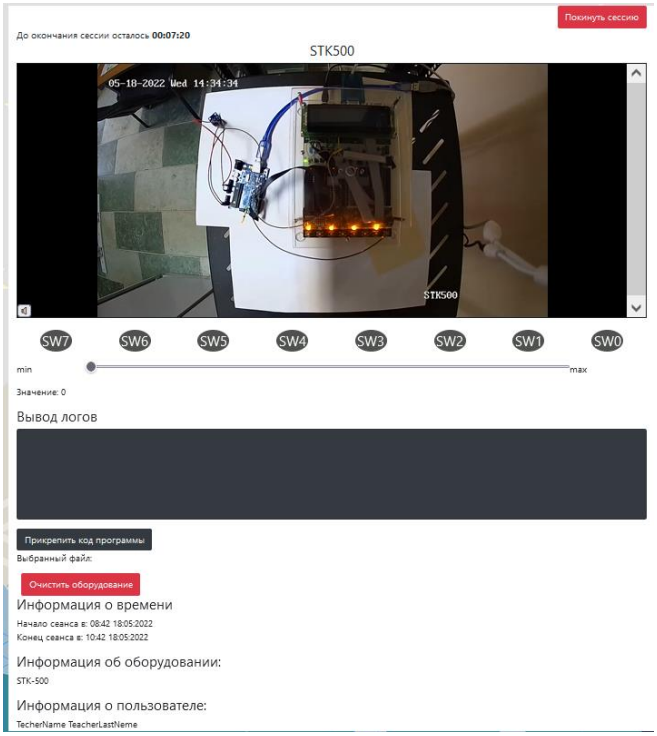

Рисунок 22 – Интерфейс сайта

При взаимодействии пользователя с элементами интерфейса формируются запросы, описанные выше. Были проведены тесты загрузки hex-файла на сервер и прошивки платы STK500, управления кнопками и резистором переменного сопротивления, перепрошивки конечного устройства стенда ранее загруженным файлом, очистки памяти микропроцессора и удаления загруженных пользователем файлов. Все проведённые тесты

показали положительные результаты, полностью совпавшие с результатами тестов, проведённых ранее.

3.5 Выводы

Было проведено три этапа тестирования разработанной системы: без участия сервера, тестирование с сервером на ПК лабораторного стенда и тестирование в режиме «точка-точка». При этом:

1. Выполнено тестирование управляющих команд для командной строки, управляющих скриптов, а также взаимодействия разработанных скриптов с API.
2. При тестировании определено, что разработанное программное обеспечение для системы удалённого управления платой STK500 в режиме лабораторных испытаний работает корректно.
3. Несмотря на нормальное функционирование ПО при доступе к плате обнаружилась значительная задержка серверного оборудования (приблизительно до 3-х секунд) при трансляции видео.
4. При перезагрузке сервера возникает значительная ошибка: сбой управляющей платы Arduino Uno. Ошибка решается перезагрузкой управляющей платы и однократным вводом данных через Serial порт в ручном режиме. Данная проблема требует дальнейшей проработки.
5. При трансляции видеоизображение, получаемое с камер, имеет расширенный формат за счет увеличенного угла обзора, что требует замены камер на камеры с меньшим фокусным расстоянием. Так же требуется обеспечить дополнительную подсветку для оборудования.
6. Необходимо разработать корпус для платы управления и модуля ЦАП для более аккуратного и безопасного расположения.
7. Полученные результаты позволяют сформировать перечень основных требований по дальнейшему развитию проекта дистанционного доступа к лабораторному оборудованию.

ЗАКЛЮЧЕНИЕ

В процессе реализации проекта поэтапно решались определенные по результатам анализа задания на ВКР задачи. На начальном этапе были рассмотрены известные программные и аппаратные решения по удаленному доступу к лабораторному оборудованию. Это позволило из прочих выбрать принцип организации доступа, примененный в НИЯУ ВШЭ, расширив его возможностью подключения различного оборудования, а также доступом через сайт и мобильное приложение вместо удалённого рабочего стола.

На втором этапе, при создании системы удалённого управления платой STK500 рассмотрена разработанная рабочей группой общая архитектура и организация сетевого взаимодействия аппаратных средств, а также разработана архитектура лабораторного стенда и предложен способ взаимодействия элементов стенда. Это позволило перейти к выбору требуемого аппаратного обеспечения и дальнейшей разработке аппаратной и программной частей лабораторного стенда. Также была выполнена интеграция ПО производителя STK500 для программирования конечного устройства стенда в режиме удалённого доступа. Также на этом этапе была разработана программа для управляющей платы, позволяющая управлять периферийными устройствами платы STK500 с помощью Arduino Uno. Также были разработаны скрипты для серверного ПО, позволяющие автоматизировать ввод управляющих команд, и программы для последующего тестирования системы удалённого управления STK500.

На третьем этапе работ, было выполнено тестирование разработанной системы в трёх режимах: без использования сервера, с участием сервера на ПК лабораторного стенда и в режиме «точка-точка». При тестировании использовалось API и серверное ПО, созданное рабочей группой проекта. Результаты тестирования показали нормальное функционирование всех частей разработанной системы, определённых заданием на ВКР. Тем не менее, тестирование показало некоторые недочёты в работе системы, а именно:

задержку трансляции видео, сбои в работе управляющей платы при перезагрузке сервера, необходимость разработать корпус для управляющей платы и модуля ЦАП, а также недостаточную информативность при передаче видео. Полученные результаты тестирования можно использовать при разработке дальнейшего плана модификации лабораторного комплекса.

Таким образом, все поставленные задачи ВКР решены, что позволяет сделать вывод о достижении цели работы.

СПИСОК СОКРАЩЕНИЙ

АЦП	– Аналогово-цифровой преобразователь
МК	– Микроконтроллер
ПК	– Персональный компьютер
ПЛИС	– Программируемая логическая интегральная схема
ПО	– Программное обеспечение
ЦАП	– Цифро-аналоговый преобразователь
ЦПУ	– Центральное процессорное устройство
API	– Application Programming Interface
AVR	– Advanced Virtual RISC
COM	– Communications Port
GND	– Ground
HEX	– Hexadecimal Source File
HLS	– HTTP Live Streaming
I2C	– Inter-Integrated Circuit
IDE	– Integrated Development Environment
IP	– Internet Protocol
LCD	– Liquid Crystal Display
RISC	– Reduced Instruction Set Computer
RTSP	– Real Time Streaming Protocol
SPI	– Serial Peripheral Interface
UART	– Universal Asynchronous Receiver-Transmitter
USB	– Universal Serial Bus

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ёхин М.Н., Степанов М.М. Организация многопользовательского удаленного доступа к распределенной гетерогенной системе лабораторного оборудования на основе схем программируемой логики для дистанционных практикумов по цифровой схемотехнике / Ёхин М.Н., Степанов М.М. // Современные информационные технологии и ИТ-образование – 2017 – Т.13 – №4 – URL: https://cyberleninka.ru/article/n/organizatsiya-mnogopolzovatelskogo-udalennogo-dostupa-k-raspredelennoy-geterogennoy-sisteme-laboratornogo-oborudovaniya-na-osnove/viewer (дата обращения: 22.12.2021).

2. А.В. Трухин. «Об использовании виртуальных лабораторий в образовании» /А.В. Трухин // Открытое и дистанционное образование. – 2002. – № 4 – URL: https://ido.tsu.ru/files/pub2002/4%288%29309Truhin_A._%28TUSUR%29.pdf (дата обращения: 22.12.2021).

3. Виртуальные лаборатории // Казанский федеральный университет – URL: https://kpfu.ru/docs/F324157708/Virtualnye.laboratorii.pdf (дата обращения: 23.12.2021).

4. Software Tools for Academics and Researchers [сайт]. – URL: http://star.mit.edu/ (дата обращения: 23.12.2021).

5. Лабораторные работы дистанционно – как это? / ITMO Expert [сайт] – URL: http://expert.itmo.ru/labs (дата обращения: 23.12.2021).

6. VirtuLab [сайт]. – URL: http://www.virtulab.net/ (дата обращения: 23.12.2021).

7. Программные симуляторы [Электронный ресурс]. – Режим доступа: URL: https://de.ifmo.ru/bk_netra/page.php?tutindex=25&index=72. – Загл. с экрана (дата обращения: 23.12.2021).

8. Proteus программа // All-Audio.pro: Статьи, Схемы, Справочники – URL: https://all-audio.pro/c24/instruktsii/proteus-programma.php (дата обращения: 23.12.2021).

9. Моделирование и расширенный анализ схем в PSpice 2017. Часть 1. [Электронный ресурс]. – Режим доступа: – URL: https://www.pcbsoft.ru/pspice-statiya-1-ch1 – Загл. с экрана (дата обращения: 23.12.2021).

10. Симуляторы компьютерных систем – похожи ли на реальность // Хабр [сайт]. – URL: https://habr.com/ru/company/auriga/blog/504086/ (дата обращения: 23.12.2021).

11. Удаленный доступ к оборудованию УЛ САПР // Национальный исследовательский университет «Высшая школа экономики» [сайт]. – URL: https://miem.hse.ru/edu/ce/cadsystem/remote_access (дата обращения: 23.12.2021).

12. Использование AVR Studio [Электронный ресурс] – Режим доступа: – URL: http://www.gaw.ru/html.cgi/txt/app/Atmel/micros/avr/stk500/5.htm – Загл. с экрана (дата обращения: 21.12.2021).

13. Уроки Arduino и робототехники [Электронный ресурс] – Режим доступа: – URL: https://alexgyver.ru/lessons/ (дата обращения: 21.12.2021).

14. MCP4725 12-Bit DAC Tutorial [Электронный ресурс] – Режим доступа: – https://learn.adafruit.com/mcp4725-12-bit-dac-tutorial/using-with-arduino?view=all (дата обращения: 09.02.2022).

15. Подключение MCP4725 к Arduino [Электронный ресурс] – Режим доступа: – https://arduino-ide.com/modules/33-podkljuchenie-mcp4725-k-arduino.html (дата обращения: 09.02.2022).

16. Раздел 5 Использование AVR Studio [Электронный ресурс] – Режим доступа: – http://www.gaw.ru/html.cgi/txt/app/Atmel/micros/avr/stk500/5.htm (дата обращения: 16.12.2021).

17. Python 3 для начинающих [сайт]. – URL: https://pythonworld.ru/ (дата обращения: 21.02.2022).

18. СТО 4.2–07–2014 «Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности».

ПРИЛОЖЕНИЕ А

Листинги управляющих программ

sketch_STK500.ino

```
#include <Wire.h>
#include <Math.h>
#include <Adafruit_MCP4725.h>
#define MCP4725In A1
Adafruit_MCP4725 MCP4725;

const int lenght = 19;
byte buf[lenght];
int rlen;
byte incomingByte;

void setup() {
  for (byte i = 2; i <= 9; i++) {
    pinMode(i, OUTPUT);
  }
  Serial.begin(9600);
  MCP4725.begin(0x60);
}

void loop() {
  if (Serial.available() > 0) {

    rlen = Serial.readBytes(buf, lenght); //прочитали команду
    Serial.println("rlen: ");
    Serial.println(rlen, DEC);
    Serial.println("I received: ");
    for(int i = 0; i < rlen; i++) {
      Serial.print(buf[i] - '0', DEC);
      Serial.print(" ");
    }

    if(buf[0]=='c'&&buf[1]=='o'&&buf[2]=='m'&&buf[3]=='m'){
      for(int i = 2; i <= 9; i++) {         //работа с книпками
        if (buf[i+2] == '1'){
          pinMode(i, INPUT);
          int lv = digitalRead(i);
          pinMode(i, OUTPUT);
          if(lv == 1)
            digitalWrite(i, LOW);
          else if(lv == 0)
            digitalWrite(i, HIGH);
          delay(1000);
        }
}
```

```
    if(buf[13] == '1'){                    //работа с АЦП
      Serial.println("ADC enabled");
      uint32_t MCP4725_value = 0;

        for(int i = 0; i < 4; i++)
        {
          MCP4725_value = MCP4725_value + (buf[rlen - 2 - i] - '0')*pow(10, i);
        }

        Serial.println("Converce: ");
          Serial.println(MCP4725_value);
        //MCP4725.begin(0x60);
        MCP4725.setVoltage(MCP4725_value, false);
        delay(1000);
    }
    else if(buf[8] == '0') {
      Serial.println("ADC disabled");
    }
   }
  }

  delay(10);
}
```

STK_but_adc.py

```
import sys
import os
import cgi
import time

#Получаем параметры
com_buttons = sys.argv[1]
com_adc = sys.argv[2]

#Определяем текущее местное время
seconds = time.time()
time_start = time.ctime(seconds)

#Записываем время
time_com = 'echo ' + time_start + ' > C:\\inetpub\\wwwroot\\Log.txt'
print(time_start + 'Script STK_but_adc worked!')
os.system(time_com)

#Формируем команду
com = 'cmd /C echo comm' + com_buttons + ' ' + com_adc + ' > COM5'

#Отправляем команду в cmd
```

```
os.system(com)

#Записываем в лог
com = 'echo comm' + com_buttons + ' ' + com_adc + ' >> C:\\inetpub\\wwwroot\\Log.txt'
os.system(com)
com = 'echo comm' + com_buttons + ' ' + com_adc + ' >> C:\\STK-scripts\\Logs\\Log.txt'
os.system(com)
```

STK_prog.py

```
import sys
import os
import time
import cgi

#Определяем текущее местное время
seconds = time.time()
time_start = time.ctime(seconds)

#Вывод времени
time_com = 'echo ' + time_start + ' > C:\\inetpub\\wwwroot\\Log.txt'
os.system(time_com)
print(time_start + ' Script STK_prog worked!')

#Получаем имя .hex-файла
file_name = sys.argv[1]

#Формируем команду для cmd (прошиваем STK и пишем лог)
com = 'cmd /C "C:\Program Files (x86)\Atmel\AVR Tools\STK500\Stk500.exe" -dATmega32A -ms -e -pf -vf -if%s >> C:\\inetpub\\wwwroot\\Log.txt' % (file_name)

#Отправляем команду
os.system(com)
```

STK_clean.py

```
import os
import sys
import time
import cgi

#Определяем текущее время
seconds = time.time()
time_start = time.ctime(seconds)

#Записываем время в лог
time_com = 'cmd /C echo %s > C:\\inetpub\\wwwroot\\Log.txt' % (time_start)
print(time_start + ' Script STK_clean worked!')
```

```
        os.system(time_com)

        #Формируем команду стирания памяти STK-500
        com_erase = 'cmd /C "C:\Program Files (x86)\Atmel\AVR Tools\STK500\Stk500.exe" -
dATmega32A -e >> C:\\inetpub\\wwwroot\\Log.txt'

        #Отправляем команду в командную строку
        os.system(com_erase)

        #Удаляем hex-файлы с помощью python
        fileDir = r"C:\inetpub\wwwroot"
        fileExt = r".hex"
        for file_name in os.listdir(fileDir):
            if file_name.endswith(fileExt):
                com = 'cmd /C echo Deleted: %s >> C:\\inetpub\\wwwroot\\Log.txt' % (file_name)
                os.remove("C:\\inetpub\\wwwroot\\" + file_name)
    os.system(com)
```

ПРИЛОЖЕНИЕ Б

Листинги тестовых программ

Test_LEDs.asm

```
ldi r30, 0
out ddrd, r30
ldi r30, 255
out ddrb, r30
m1:
in r30, pind
out portb, r30
rjmp m1
```

ASC_test.asm

```
.include "m32adef.inc"
.def temp = r16
.cseg
.org 0x00 rjmp Init
.org ADCCaddr rjmp Int_ADC

Init:
Ldi temp,low(Ramend)
out spl,temp
Ldi temp,high(Ramend)
out sph,temp

ldi temp,~(1<<7)
out DDRA,temp
clr temp
out ddrd,temp

ser temp
out ddrb,temp
ldi temp,(1<<ADLAR)|(1<<MUX2)|(1<<MUX1)|(1<<MUX0)
out ADMUX, temp
ldi temp,(1<<ADEN)|(1<<ADATE)|(1<<ADIE)|(1<<ADPS0)|(1<<ADPS1)
out ADCSRA, temp
sei
sbi ADCSRA, ADSC

m1: rjmp m1

Int_ADC:
in r30, ADCL
in r31, ADCH
out PORTB,r31
reti
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт
Вычислительная техника
Кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«20» 06 2022 г.

БАКАЛАВРСКАЯ РАБОТА

«Система удалённого доступа к лабораторному оборудованию. Подсистема удалённого управления платой STK500»
Тема

09.03.01 «Информатика и вычислительная техника»
код и наименование направления

Руководитель	_____ подпись, дата	доцент.каф. ВТ ИКИТ Канд. физ.-мат. наук должность, ученая степень	К.В. Коршун инициалы,фамилия
Выпускник	_____ подпись, дата		Н.В. Коптяев инициалы,фамилия
Нормоконтролер	_____ подпись, дата	доцент.каф. ВТ ИКИТ Канд. физ.-мат. наук должность, ученая степень	К.В. Коршун инициалы,фамилия

Красноярск 2022