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 Altera DE1-SoC»
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		П.Д. Неустроев 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 Altera DE1-SoC для системы удалённого доступа к лабораторному оборудованию.

Данная пояснительная записка содержит 52 страницs текста с иллюстрациями и таблицами, 2 приложения, и 14 использованных источников.

УДАЛЁННЫЙ ДОСТУП, СИСТЕМА УДАЛЁННОГО ДОСТУПА, ВИРТУАЛЬНАЯ ЛАБОРАТОРИЯ, ПРОГРАММНЫЙ СИМУЛЯТОР, ПЛИС, FPGA, DE1-SOC, ЭМУЛЯЦИЯ, ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА, ПЕРЕКЛЮЧАТЕЛИ, КНОПКИ, КОМАНДНАЯ СТРОКА, SOF-ФАЙЛ, QUARTUS PRIME.

Цель бакалаврской работы – разработка подсистемы удалённого управления платой Altera DE1-SoC 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 Altera DE1-SoC;
- осуществлена прошивка платы Altera DE1-SoC через командную строку с помощью sof-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 Неустроеву Павлу Дмитриевичу

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 Altera DE1-SoC.

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 Altera DE1-SoC. Осуществлять прошивку платы Altera DE1-SoC через командную строку с помощью sof-файла. Разработать управляющие скрипты на языке Python для автоматизации формирования и ввода управляющих команд. Провести тестирование разработанной системы.

Перечень разделов ВКР: Обзор систем удалённого доступа. Разработка системы удалённого управления платой Altera DE1-SoC. 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 Altera DE1-SoC, 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 _____ П.Д. Неустроев

подпись инициалы, фамилия

«___» _____ 20__г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время проблема доступа к лабораторному оборудованию во время дистанционного обучения является актуальной. В различных высших учебных заведениях применяются разные варианты решения этой задачи: системы удалённого доступа, виртуальные лаборатории, программные симуляторы оборудования. В данной работе был проведён анализ возможных вариантов её решения, и разработана подсистема удалённого управления платой Altera DE1-SoC для системы удалённого доступа к лабораторному оборудованию.

Целью выпускной квалификационной работы является разработка подсистемы удалённого управления платой Altera DE1-SoC для системы удалённого доступа к лабораторному оборудованию.

В данной работе использовалась платы Altera DE1-SoC, Arduino Uno R3, а также среды разработки Quartus Prime 18.0 Lite Edition и Arduino IDE. Программное обеспечение реализовано на языках C/C++, Python и Verilog.

1 Обзор систем удалённого доступа

1.1 Актуальность выбранной темы

В современных реалиях всё более востребованным становится дистанционное образование. Оно является удобным для студентов, так как позволяет более эффективно совмещать учёбу и работу, а также учиться из дома, в том числе находясь в другом городе. Однако дистанционное образование лишено некоторых преимуществ очного, из-за чего студенты могут не получить необходимых навыков. В особенности это касается тех образовательных программ, в которых большое внимание уделяется практике, в том числе работе с лабораторным оборудованием. При дистанционном обучении, либо при обучении в заочной форме у студентов нет возможности работать с реальным оборудованием, что отрицательно сказывается на качестве получаемых знаний и навыков.

Исходя из этого, можно сделать вывод, что проблема обеспечения удалённого доступа к лабораторному оборудованию в настоящее время является актуальной.

В результате поиска и анализа информации о данной проблеме были найдены несколько основных вариантов её решения: виртуальные лаборатории, программные симуляторы и системы удалённого доступа. Их преимущества и недостатки будут рассмотрены ниже.

1.2 Виртуальные лаборатории

Виртуальная лаборатория, согласно источнику [1] представляет собой программно-аппаратный комплекс, позволяющий проводить опыты без непосредственного контакта с реальной установкой или при полном отсутствии таковой.

Виртуальные лаборатории применяются в современном высшем образовании. В частности, в Томском политехническом университете есть собственный проект «Виртуальные лаборатории» (рис. 1) [2]. В данном проекте содержится более 200 виртуальных лабораторных работ по химии, экологии, гидрогазодинамике, методам анализа, электрическим машинам, энергетике, физике, микробиологии, метрологии, термодинамике.

Рисунок 1 – Виртуальная лаборатория

Автор [1] выделяет следующие преимущества у виртуальных лабораторий:

- Отсутствие необходимости приобретения дорогостоящего оборудования и расходных материалов.
- Возможность моделирования процессов, протекание которых принципиально невозможно в лабораторных условиях.
- Наглядная визуализация на экране компьютера.
- Возможность проникновения в тонкости процессов и наблюдения происходящего в другом масштабе времени.
- Безопасность.

- Возможность быстрого проведения серии опытов с различными значениями входных параметров.

- Автоматический ввод и обработка больших массивов полученных цифровых данных.

Согласно источнику, некоторые из этих преимуществ характерны и для систем удалённого доступа.

У виртуальных лабораторий можно также выделить ряд существенных недостатков. Главный из них – отсутствие опыта работы с реальным оборудованием, который необходим при обучении. Также недостатками виртуальных лабораторий являются дороговизна и сложность их разработки на профессиональном уровне, либо слишком узкая направленность, если разработкой такой лаборатории занимались не профессионалы.

С учётом приведённых выше преимуществ и недостатков можно сделать вывод, что для решения поставленной задачи виртуальная лаборатория не является подходящим решением.

1.3 Программные симуляторы

Программный симулятор предназначен для имитации на инструментальном (персональном) компьютере работы контроллера при выполнении разработанной программы управления и представляет собой программно-логическую модель проектируемого устройства [3].

Как правило, симулятор содержит в своем составе:

- Отладчик;
- Модель ЦПУ и памяти.

Более продвинутые симуляторы могут содержать в своем составе модели встроенных периферийных устройств, таких, как таймеры, порты, АЦП, системы прерываний [4].

Рассмотрим пример программного симулятора.

Multisim (рис. 2) – программа, разработанная National Instruments, применяется для моделирования и программирования схем для аналоговой, цифровой и силовой электроники в образовательной и исследовательской областях. Multisim интегрирует стандартную симуляцию на основе SPICE с интерактивной схемотехнической средой для мгновенной визуализации и анализа поведения электронных схем [5].

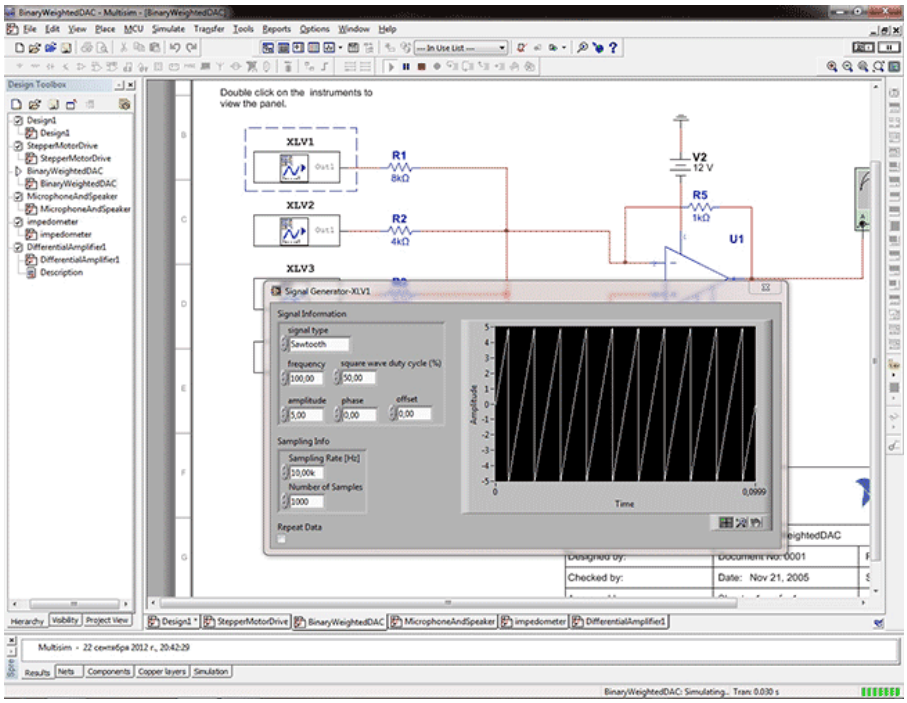

Рисунок 2 – Моделирование в Multisim

Программные симуляторы имеют возможность загружать файлы программ во всех популярных форматах, подходящих для эмулируемого оборудования, что может дать возможность для выполнения лабораторных работ студентами с использованием симуляторов. Помимо этого, следует выделить ещё несколько преимуществ программных симуляторов:

- быстрая проверка правильности алгоритма выполнения программы и отдельных операций;

- возможность вести наблюдение за исполнением любого фрагмента программы и реакцией оборудования на различные события;

- возможность исследования ситуаций, трудно воспроизводимых на реальной аппаратуре.

Пользователь имеет возможность запускать загруженную в симулятор программу в пошаговом или непрерывном режимах, задавать условные и безусловные точки останова, контролировать и свободно модифицировать содержимое ячеек памяти и регистров симулируемого оборудования. С помощью симулятора можно быстро проверить логику выполнения программы, правильность выполнения арифметических операций [3,4].

Из сказанного выше можно сделать вывод, что программные симуляторы наиболее подходят для отладки и тестирования программы перед её загрузкой в реальное оборудование. Важной особенностью программных симуляторов является то, что исполнение программ, загруженных в симулятор, происходит в масштабе времени, отличном от реального. Из-за этого могут возникать проблемы при запуске отлаженной программы на реальной аппаратуре, так как могут отличаться временные задержки на ней и в симуляторе. Это является недостатком программных симуляторов.

Таким образом, программные симуляторы не могут служить полноценной заменой реального оборудования, а значит, не являются подходящим решением для поставленной задачи.

1.4 Системы удалённого доступа

В отличие от виртуальных лабораторий и программных симуляторов системы удалённого доступа предполагают наличие реальной аппаратуры, имеющей доступ к сети, в специально оборудованной лаборатории. Пользователи имеют возможность подключаться к этому оборудованию по локальной сети либо через Интернет, и дистанционно управлять им, наблюдая за состоянием оборудования с помощью камер. Таким образом, при использовании системы в образовательном учреждении, студент получает навыки работы с реальным оборудованием, даже находясь на дистанционном

обучении, что является главным преимуществом такой системы. К недостаткам же систем удалённого доступа можно отнести задержки, неизбежно появляющиеся при связи с оборудованием по сети.

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. Стенды с удалённым доступом нужны для отладки и подготовки проектов и лабораторных работ. В лаборатории развернуты рабочие станции с подключенными к ним отладочными платами ПЛИС DE1-SoC, DE10-standart и DE10-lite. Также лаборатории, оснащённые системами удалённого доступа, есть в Новосибирском Государственном Университете и в Национальном исследовательском ядерном университете «МИФИ».

На рисунке 3 представлен скриншот программы AMCap, с помощью которой производится наблюдение за стендами учебной лаборатории систем автоматизированного проектирования ВШЭ в режиме удалённого доступа [6].

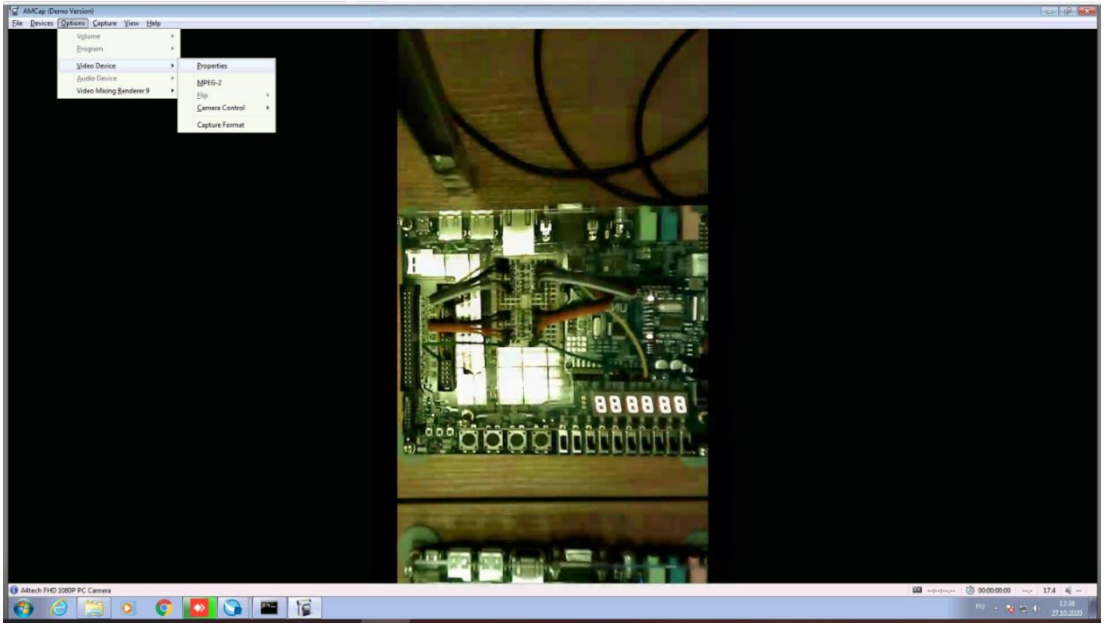

Рисунок 3 – Наблюдение за стендом в режиме удалённого доступа

Ещё один пример системы удалённого доступа – проект LabsLand (рис. 4) [7]. На сайте представлено множество «Лабораторий», предоставляющих доступ к реальному оборудованию, в том числе ПЛИС, программируемым роботам и лабораторным установкам для физических опытов.

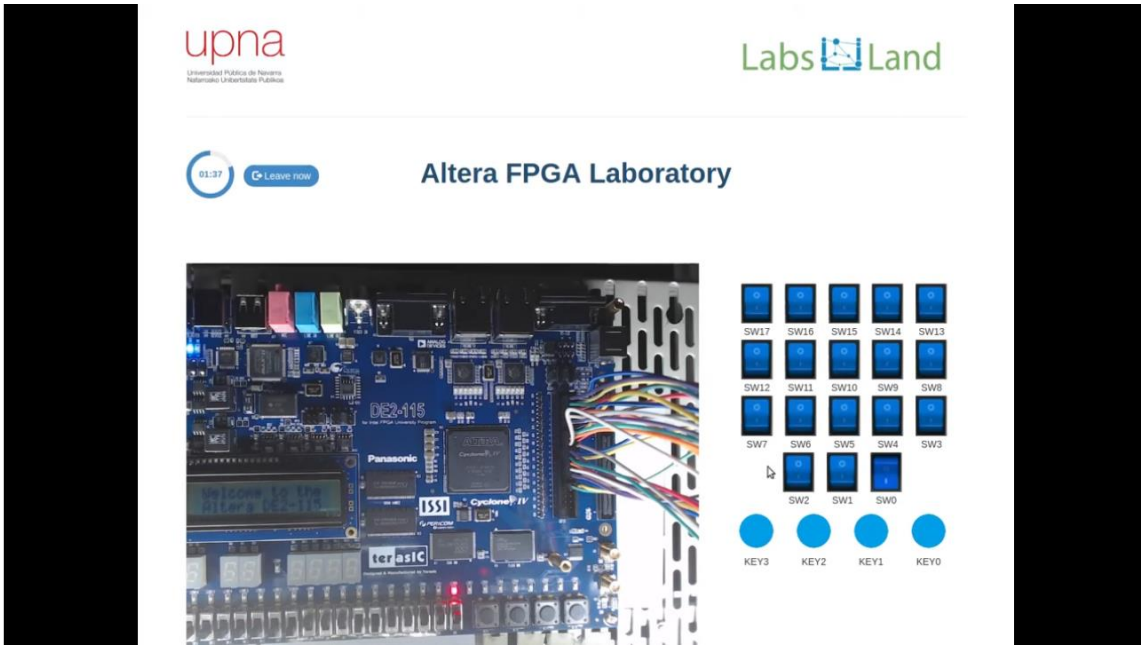

Рисунок 4 – Лаборатория LabsLand

1.5 Выводы

Проведён анализ различных вариантов решения проблемы удалённого доступа к лабораторному оборудованию. Рассмотрены примеры различных виртуальных лабораторий, программных симуляторов и систем удалённого доступа, выявлены их достоинства и недостатки. Это позволило уточнить основные задачи дальнейшей работы, а именно:

1. Разработать архитектуру системы удаленного управления платой Altera DE1-SoC.

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

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 Altera DE1-SoC 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 (переключателей, кнопок и светодиодов).

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 управления платой.

2 Разработка системы удалённого управления платой Altera DE1-SoC

2.1 Обзор поставленной задачи и оборудования

Согласно задания на ВКР рассматривается задача разработки подсистемы удалённого управления платой Altera DE1-SoC для системы удалённого доступа к лабораторному оборудованию.

Плата Altera DE1-SoC представляет собой аппаратную платформу, построенную на основе ALTERA System-on Chip (SoC) FPGA (ПЛИС), которая является комбинацией встроенного двухъядерного процессора Cortex-A9 и программируемой логики. В данной плате имеются процессор на основе ARM, интерфейсы, периферийные устройства и память, связанные с ПЛИС FPGA. Отладочная плата DE1-SoC включает в себя память DDR3, аудио- и видео-интерфейсы, Ethernet и другое [8 – 10].

На рисунке 5 представлен вид используемой платы.

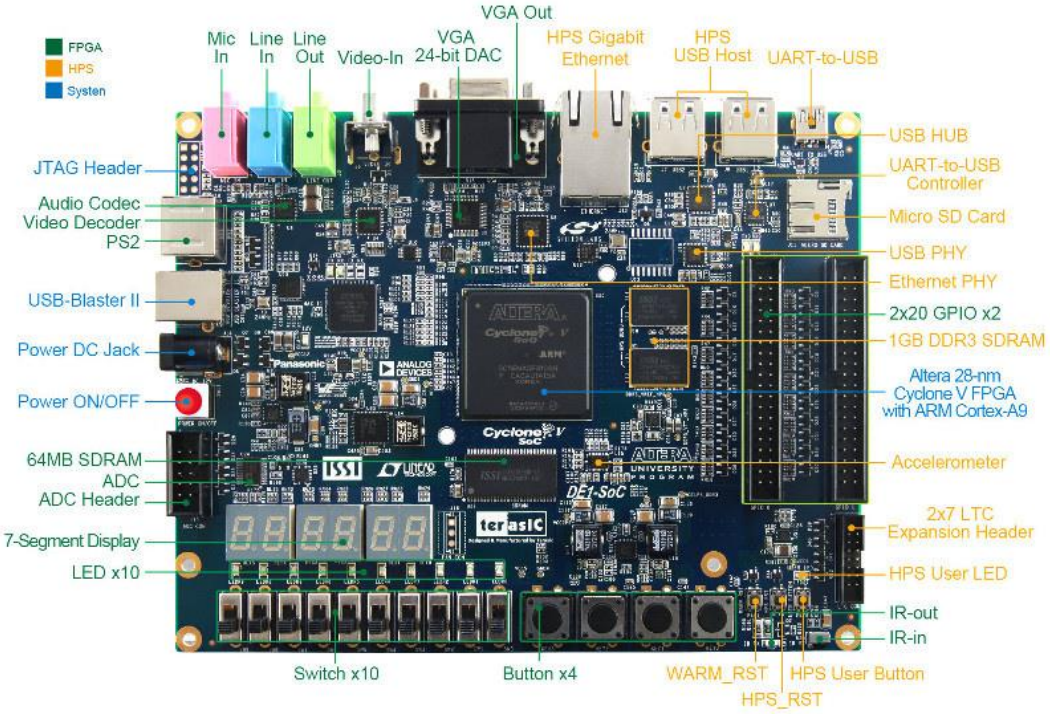

Рисунок 5 – Плата DE1-SoC

На данной плате имеется ряд периферийных устройств, которые используются как для вывода информации, так и для взаимодействия пользователя с платой. К этим устройствам относятся:

- 10 переключателей (switch);
- 4 кнопки (key);
- 10 светодиодов;
- 6 семисегментных индикаторов.

Также имеются два порта ввода-вывода (GPIO0 и GPIO1), каждый из которых имеет 40 пинов.

Пользователь может взаимодействовать с переключателями и кнопками. Так как при удалённом доступе к оборудованию возможности такого взаимодействия нет, необходимо обеспечить управление платой без использования физических периферийных устройств.

Задачу разработки подсистемы удалённого управления платой Altera DE1-SoC 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 Altera 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 (переключателей, кнопок и светодиодов);

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.

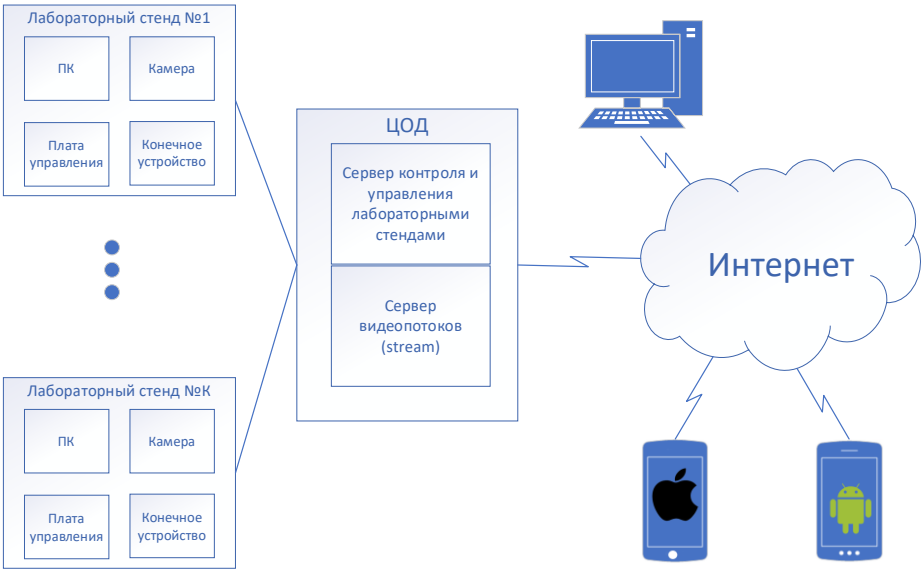

Рисунок 6 – Общая схема системы удалённого доступа к лабораторному оборудованию

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.

Под лабораторным стендом понимается совокупность устройств, плат и ПО позволяющая дистанционно управлять конечным устройством. Подробная схема лабораторного стенда, задействованного в данной работе, изображена на рисунке 7.

Рисунок 7 – Схема лабораторного стенда

Аппаратными элементами лабораторного стенда являются персональный компьютер, IP-камера, плата управления и конечное устройство – плата Altera DE1-SoC. На персональном компьютере установлен web-сервер, который отвечает за общение с конечным устройством пользователя и сервером управления. Веб-сервер может получать файлы для прошивки конечного устройства со стороны пользователя, а также управляющие команды для конечного устройства, после чего он запускает соответствующие скрипты управления. К ПК подключены IP-камера, плата управления и конечное устройство. Для взаимодействия с подключенными платами на ПК установлены необходимые драйвера.

IP-камера выполняет функцию наблюдения за конечным устройством. Она имеет встроенное программное обеспечение, в том числе web-сервер, позволяющий передавать видеопоток в режиме реального времени.

Плата управления необходима для передачи на конечное устройство команд для формирования входных сигналов от пользователя при помощи прошитой в неё программы – драйвера конечного устройства, к которому она напрямую подключена.

Конечное устройство – плата Altera DE1-SoC – может управляться

пользователем при помощи пользовательского интерфейса на сайте или в приложении.

Взаимодействие элементов разрабатываемой системы между собой представлено в виде схемы (рисунок 8):

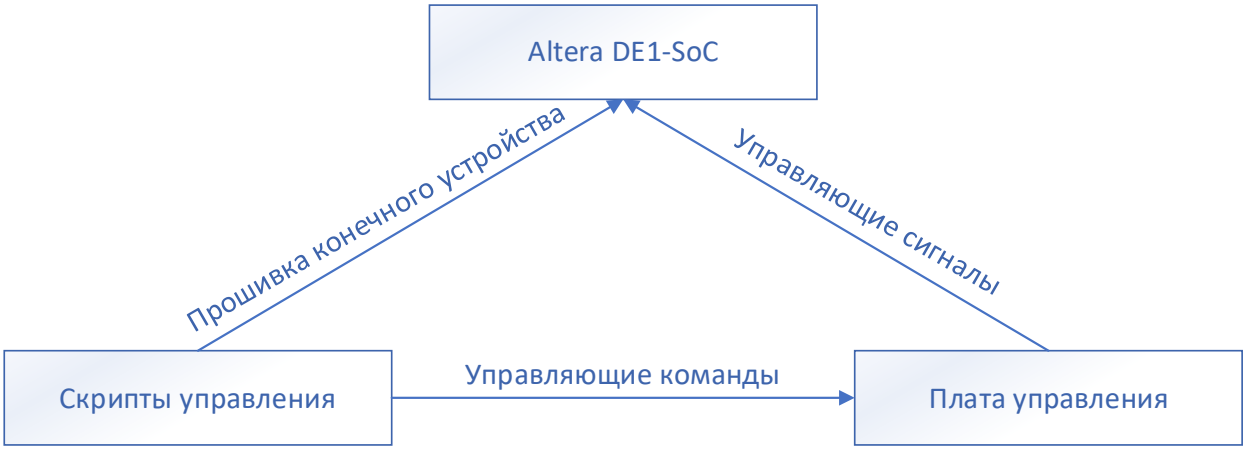

Рисунок 8 – Взаимодействие элементов системы

Управляющая плата подключается к конечному устройству стенда для управления периферийными устройствами, с которыми может физически взаимодействовать пользователь (переключатели, кнопки). Управляющие команды поступают на плату управления при помощи скриптов. Эти команды после обработки управляющей платой преобразуются в управляющие сигналы для платы Altera DE1-SoC. Также при помощи управляющих скриптов осуществляется прошивка конечного устройства стенда.

2.3 Разработка аппаратной части системы удалённого управления платой Altera DE1-SoC

2.3.1 Анализ и выбор аппаратного обеспечения для лабораторного стенда

В начале разработки системы удалённого управления платой Altera DE1-

SoC рассматривались следующие варианты платы управления: Arduino Uno и Arduino Mega. В их основе находятся микропроцессоры ATmega, которые возможно программировать на языке C/C++. Также эти платы являются достаточно надёжными в эксплуатации и с ними совместимо множество различных периферийных устройств, что предоставляет дополнительные возможности для расширения функционала лабораторного стенда.

В результате анализа и сравнения двух версий Arduino для дальнейшей работы была выбрана плата Arduino Uno, поскольку она обладает достаточным количеством портов ввода-вывода для подключения к Altera DE1-SoC и эмуляции периферийных устройств этой платы, использующихся при выполнении лабораторных работ. Также у этой платы достаточно памяти для хранения прошивки и данных. Arduino Mega имеет избыточное количество портов ввода-вывода.

2.3.2 Подключение Arduino Uno к Altera DE1-SoC

Для того, чтобы пользователь имел возможность управлять конечным устройством без использования реальных кнопок и переключателей, к нему подключается плата управления. На рисунке 9 показана схема подключения Arduino Uno к Altera DE1-SoC.

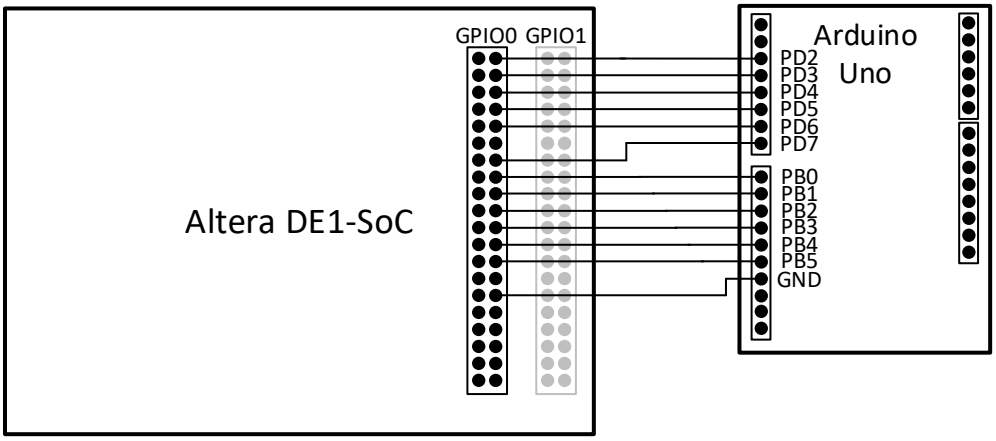

Рисунок 9 – Схема подключения

В данной схеме используется 12 цифровых выходов Arduino Uno, которые подключены к порту ввода-вывода GPIO0. Также соединены выходы GND обеих плат.

При таком подключении плата управления становится ещё одним периферийным устройством ПЛИС, и заменяет собой аппаратные переключатели и кнопки. При этом пользователю при написании программы для данной платы и подготовке её к компиляции необходимо указывать используемые выходы в коде программы на Verilog или же в Pin Planner.

На рисунке 10 представлена распиновка порта GPIO0 платы DE1-SoC.

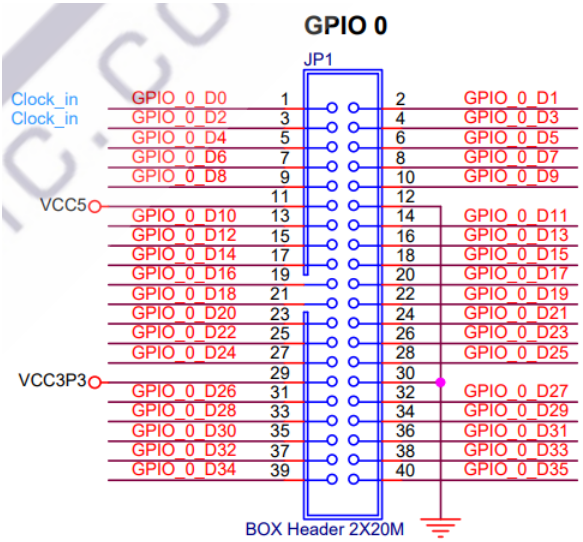

Рисунок 10 – Распиновка порта GPIO0

В таблице 1 приведено сопоставление выходов порта GPIO0 согласно схеме на рисунке 10, номера выходов согласно документации на плату DE1-SoC [11], а также подключенных к ним выходов Arduino Uno и эмулируемых периферийных устройств.

Таблица 1 – Соответствие выходов порта GPIO, Arduino и периферийных устройств ПЛИС

	GPIO_0[]	**PIN GPIO**	**FPGA Pin №**	**№ выхода Arduino**	**Заменяемое устройство**
1	GPIO_0[1]	2	PIN_Y17	2	SW[0]
2	GPIO_0[3]	4	PIN_Y18	3	SW[1]
3	GPIO_0[5]	6	PIN_AK18	4	SW[2]
4	GPIO_0[7]	8	PIN_AJ19	5	SW[3]
5	GPIO_0[9]	10	PIN_AJ16	6	SW[4]
6	–	12 (GND)	–	–	–
7	GPIO_0[11]	14	PIN_AH17	7	SW[5]
8	GPIO_0[13]	16	PIN_AE16	8	SW[6]
9	GPIO_0[15]	18	PIN_AG17	9	SW[7]
10	GPIO_0[17]	20	PIN_AA19	10	KEY[0]
11	GPIO_0[19]	22	PIN_AC20	11	KEY[1]
12	GPIO_0[21]	24	PIN_AJ20	12	KEY[2]
13	GPIO_0[23]	26	PIN_AK21	13	KEY[3]
14	–	30 (GND)	–	GND	–

На плате DE1-SoC имеется 10 переключателей и 4 кнопки, однако аппаратные возможности Arduino позволяют использовать только 12 цифровых выходов, что является достаточным для выполнения лабораторных работ с использованием данной полаты. В связи с этим переключатели SW[8], SW[9] не заменяются выходами Arduino, и, следовательно, не используются в удалённых лабораторных работах.

Вид подключения Arduino к конечному устройству лабораторного стенда показан на рисунке 11.

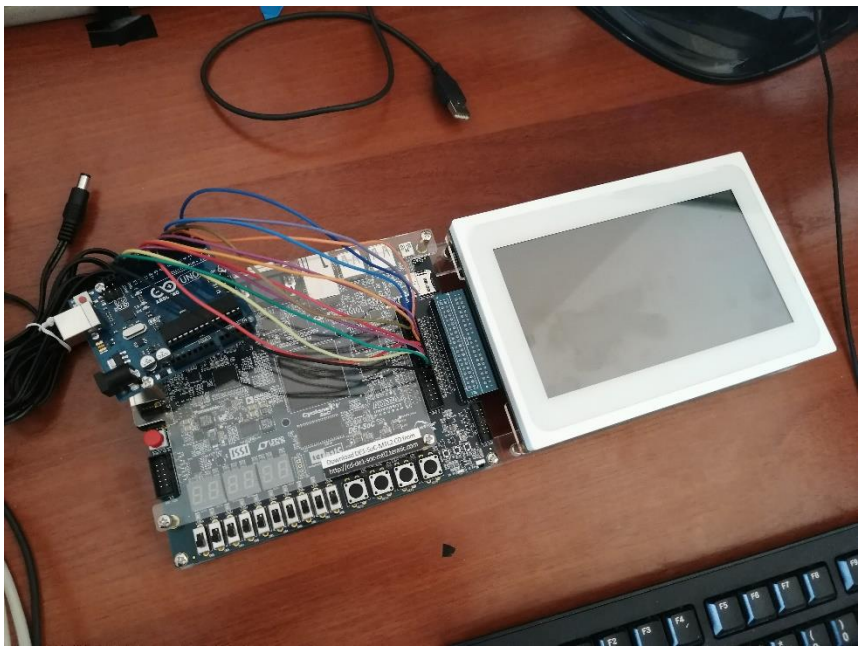

Рисунок 11 – Соединение Arduino и Altera DE1-SoC

2.4 Управление периферийными устройствами Altera DE1-SoC в режиме удалённого доступа

Для управления периферийными устройствами платы DE1-SoC – необходимо передавать команды с компьютера на Arduino Uno. Плата управления получает управляющие команды с компьютера через Serial Port (COM-порт). Информацию в данный порт возможно передавать через командную строку при помощи стандартной команды echo.

Для того, чтобы плата управления могла распознавать и обрабатывать передаваемые ей команды, была разработана программа для Arduino Uno, являющаяся драйвером конечного устройства. Программа написана на языке C++. Листинг программы приведён в приложении А.

При запуске программы происходит инициализация, в результате которой все цифровые выходы Arduino переводятся в режим вывода (OUTPUT), для вывода на них управляющих сигналов для конечного устройства стенда. Также на всех цифровых выходах устанавливается значение напряжения высокого уровня. 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 «key» на цифровых выходах PD2 – PB5 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.

В случае, если была получена ключевая последовательность «cle», на всех цифровых выходах управляющей платы устанавливается значение напряжения высокого уровня, что переводит все периферийные устройства в положение «выключено», поскольку напряжение низкого уровня, установленное служит сигналом для замыкания кнопки или переключателя, тогда как напряжение высокого уровня – сигналом для их размыкания.

На рисунке 12 приведён алгоритм работы программы для Arduino Uno.

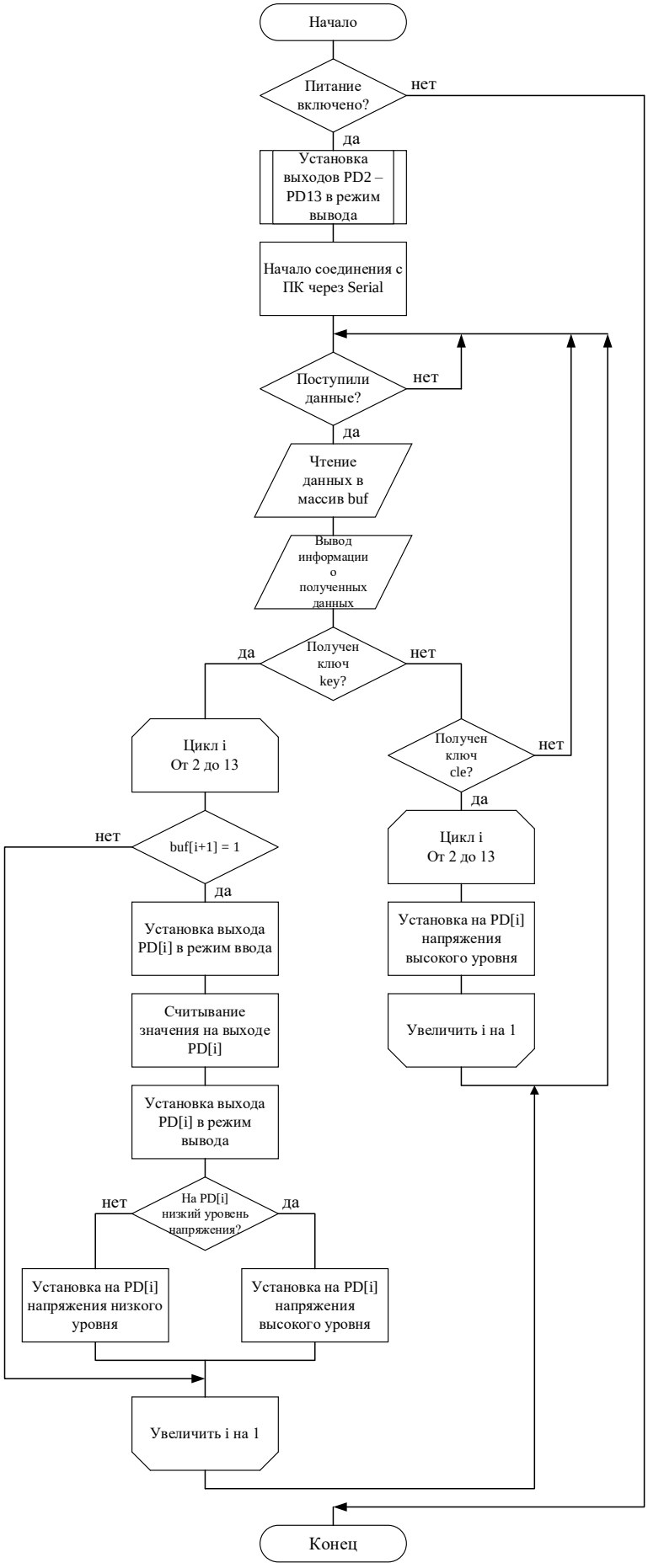

Рисунок 12 – Алгоритм работы программы

На рисунках 13, 14 приведён формат управляющих команд, вводимых в командную строку для управления периферийными устройствами Altera DE1-SoC.

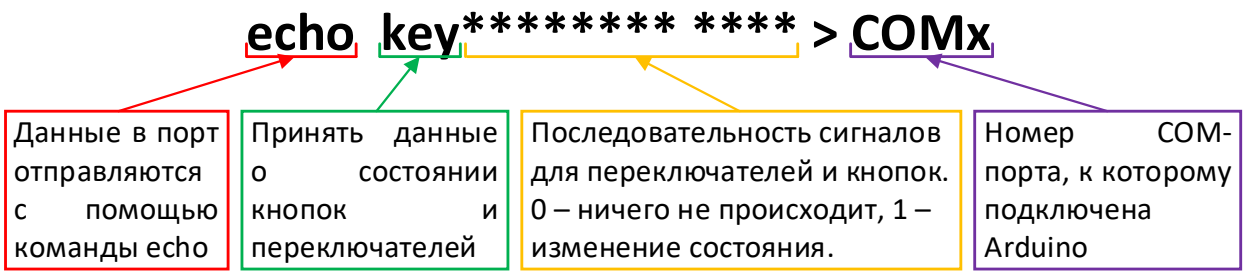

Рисунок 13 – Формат команды управления периферийными устройствами

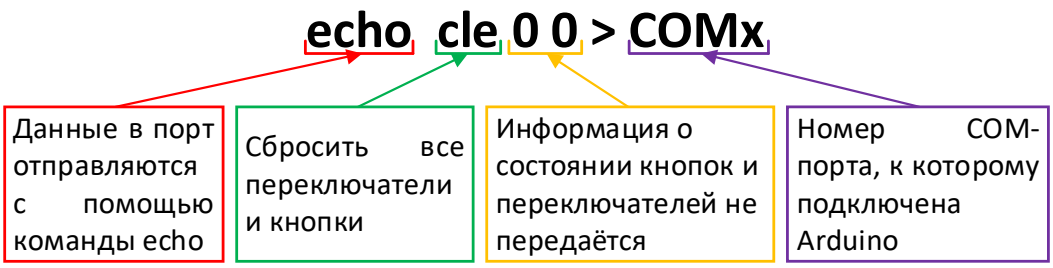

Рисунок 14 – Формат команды сброса периферийных устройств

Данные в COM-порт передаются с помощью команды:

echo [данные] > COMx,

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

[word]****** ****

Здесь [word] – ключевое слово, позволяющее избежать выполнения программы в случае подачи в порт мусорной информации или шума, а также определяющее, как именно будет работать программа. Ключевое слово может принимать два значения: «key» означает, что данные, стоящие в команде после ключевого слова, будут установлены на выходах Arduino в качестве управляющих сигналов для конечного устройства лабораторного стенда; ключевое слово «cle» означает, что все периферийные устройства платы

должны быть переведены в состояние «выключено», независимо от передаваемых данных.

Символом «*» обозначены байты, которые могут принимать значения «0» или «1».

«******** ****» – последовательность нулей и единиц, соответствующая восьми переключателям и четырём кнопкам на плате DE1-SoC. Для удобства восприятия последовательности, отвечающие за переключатели и за кнопки, разделены пробелом. После обработки программой Arduino эти данные поступают на цифровые выходы PD2 – PB5 в виде напряжения высокого или низкого уровня. При этом «0» означает, что состояние переключателя не изменилось по сравнению с предыдущим состоянием, а «1» означает изменение состояние эмулируемого устройства.

Программа для Arduino написана таким образом, что все периферийные устройства эмулируются «с удержанием» – если подать «1» на кнопку в положении «выключено», она будет переведена в состояние «включено» и продолжит оставаться в таком режиме до следующего поступления на неё «1».

2.5 Прошивка платы DE1-SoC в режиме удалённого доступа

Для написания программ для ПЛИС на языке Verilog, их компиляции и прошивки в плату студентами используется программа Quartus Prime 18.0 Lite Edition. Эта программа имеет встроенный интерфейс командной строки, в котором присутствуют все те же функции, что и в стандартном интерфейсе. Это позволяет обеспечить возможность удалённого управления платой.

Запустить данный интерфейс в операционной системе Windows можно с помощью bat-файла Nios II Command Shell.bat, который располагается по пути C:\intelFPGA_lite\18.0\nios2eds, если программа была установлена в папку C:\intelFPGA_lite. В Unix-подобных операционных системах интерфейс командной строки Quartus Prime можно запустить при помощи файла nios2_command_shell.sh, расположенном в той же папке.

После запуска интерфейса командной строки появляется возможность пользоваться всеми функциями Quartus Prime. Для задач, поставленных в данной работе, достаточно будет одной утилиты – quartus_pgm, предназначенной для программирования платы. На рисунке 13 представлен формат команды для прошивки файла с расширением .sof, генерируемого при компиляции пользовательской программы, в ПЛИС.

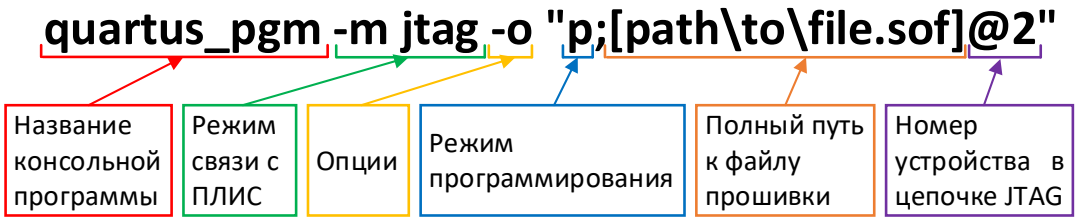

Рисунок 15 – Формат команды для прошивки программы в ПЛИС

ПК связывается с подключенной к нему ПЛИС по интерфейсу JTAG, что необходимо указывать в команде для прошивки. Кроме этого, должны быть указаны опции: переход в режим программирования, полный путь к файлу прошивки и номер устройства в цепочке JTAG, который в случае данной платы будет всегда равен 2.

Quartus Prime для командной строки выводит подробный отчёт о выполнении каждой команды. Причём, если команда выполнена успешно, цвет текста будет зелёным, а если возникают ошибки – красным. Таким образом, можно легко отследить возникающие в процессе программирования проблемы [12, 13].

2.6 Передача управляющих команд в командную строку с помощью скриптов на языке Python

На рисунках 13 – 15 настоящей пояснительной записки приведены форматы 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 разработанных 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 Altera DE1-SoC

Скрипт FPGA_prog.py был разработан для автоматизации ввода команды для прошивки конечного устройства лабораторного стенда. В качестве входного параметра скрипт получает имя sof-файла, который необходимо прошить в ПЛИС. Путь к файлу не нужно передавать вместе с именем файла.

В начале своей работы скрипт определяет текущее время и дату при помощи средств стандартной библиотеки time и выводит эту информацию в файл C:\Scripts\Log.txt. После этого формируется команда для прошивки ПЛИС путём подстановки имени файла в строку, содержащую команду. Полученная строка направляется в командную строку при помощи средств модуля subprocess. Использование данного модуля необходимо, поскольку при прошивке DE1-SoC нужно запустить среду Quartus Prime, а после в ней выполнить прошивку платы. Вывод Quartus Prime записывается в файл C:\Scripts\Log.txt. Скрипт запускается при помощи команды

python C:\Scripts\FPGA_prog.py [filename.sof],

где [filename.sof] – имя sof-файла вместе с расширением. На рисунке 14 представлен алгоритм работы данного скрипта.

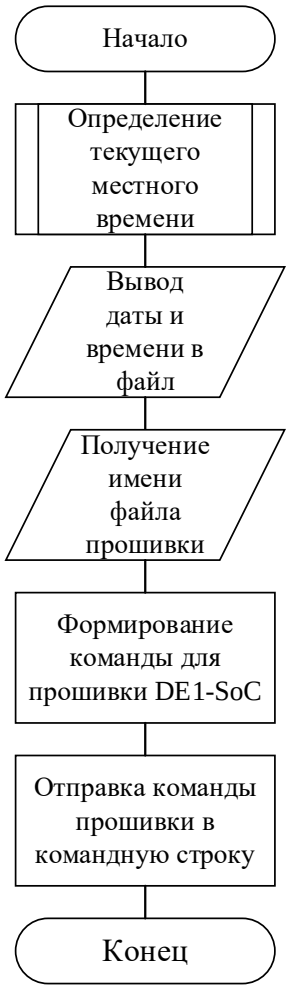

Рисунок 16 – Алгоритм работы скрипта FPGA_prog.py

2.6.2 Разработка скрипта для управления периферийными устройствами Altera DE1-SoC

Для автоматизации ввода команды для управления кнопкой B1 платы STM32 был разработан скрипт FPGA_buttons.py. Скрипт получает в качестве входных параметров ключевое слово (key или cle), последовательность из восьми нулей и единиц, обозначающих состояние переключателей и последовательность из четырёх нулей и единиц, обозначающих состояние кнопок. Скрипт определяет текущее время и дату при помощи средств стандартной библиотеки time и выводит эту информацию в файл C:\Scripts\Log.txt. Затем в соответствии с полученными входными параметрами 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 при помощи команды

python C:\Scripts\ \FPGA_buttons.py [word] ****** ****

На рисунке 17 представлен алгоритм работы данного скрипта.

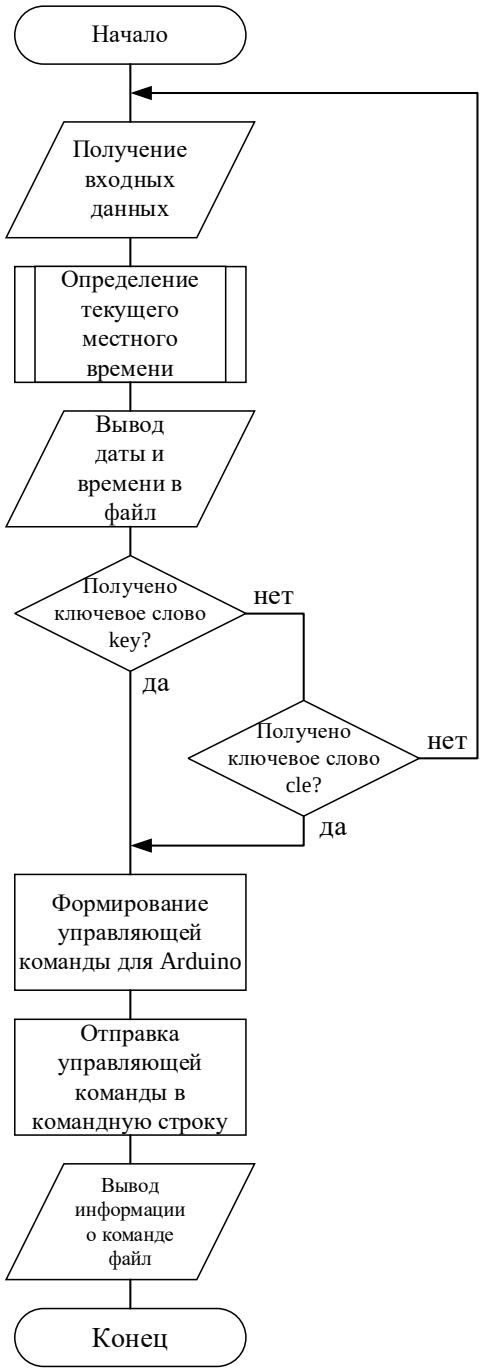

Рисунок 17 – Алгоритм работы скрипта FPGA_buttons.py

2.6.3 Разработка скрипта для очистки памяти платы Altera DE1-SoC

Скрипт FPGA_clean.py должен вызываться в конце каждой сессии пользователя. Он очищает папку C:\Scripts от всех находящихся в ней файлов прошивки и запускает команду для прошивки в ПЛИС тестового проекта, который показывает работоспособность всех светодиодов и семисегментных индикаторов платы. 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 прошивки платы Altera DE1-SoC, направляет её в командную строку, после чего удаляет все sof-файлы, находящиеся в каталоге C:\Scripts.

Скрипт запускается из командной строки командой

python C:\Scripts\FPGA_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 тестовой программы для Altera DE1-SoC

Для тестирования корректности работы разработанный системы, была написана тестовая программа для DE1-SoC, которая соответствует лабораторной работе по курсу ПЛИС. Данная программа имитирует работу четырёхразрядного сумматора. Переключатели SW[0] – SW[3] и KEY[0] – KEY[3] при выполнении данной программы выполнят роль входов сумматора, а вывод результата производится на светодиоды LED[0] – LED[3]. Написанная программа была скомпилирована с помощью среды разработки Quartus Prime 18.0 Lite Edition. В результате компиляции был получен файл с расширением .sof, являющийся файлом прошивки для платы DE1-SoC. Листинг этой программы приведён в приложении Б. Пример разработки

программы в интегрированной среде Quartus Prime приведён на рисунке 18.

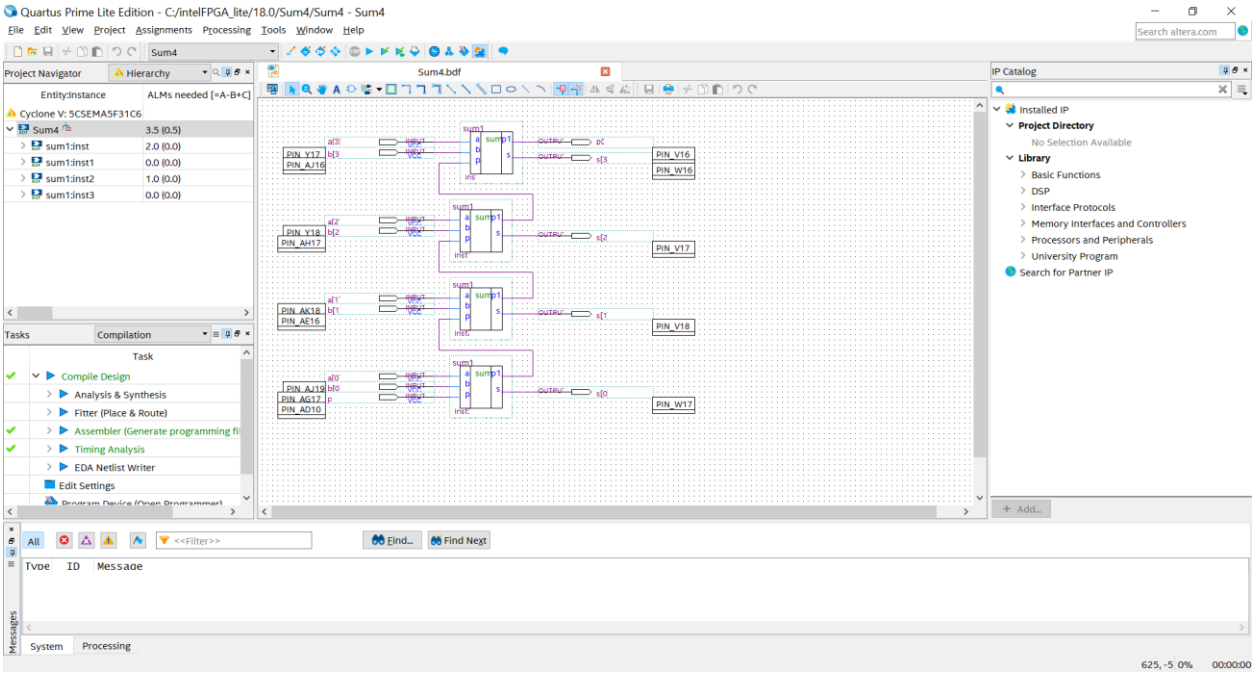

Рисунок 18 – Разработка программы лабораторной работы

2.8 Выводы

В ходе разработки подсистемы удалённого управления платой для системы удалённого доступа к лабораторному оборудованию была создана структурная схема лабораторного стенда. В соответствии с полученной схемой был собран лабораторный стенд с конечным устройством Altera DE1-SoC. Для данного устройства была разработана система управления его периферийными устройствами без прямого воздействия пользователя. Также было найдено решение для загрузки программы в конечное устройство при помощи командной строки, написаны управляющие скрипты, позволяющие автоматизировать формирование и запуск управляющих команд и разработана программа для дальнейшего тестирования системы.

Таким образом, после завершения всех этапов разработки была завершена работа над программной и аппаратной частями системы удалённого управления платой Altera DE1-SoC, что позволяет перейти к тестированию

разработанной системы.

3 Тестирование разработанной системы

3.1 Монтаж лабораторного стенда

По завершении разработки подсистемы удалённого доступа к плате Altera DE1-SoC для системы удалённого доступа к лабораторному оборудованию была выполнена сборка лабораторного стенда в соответствии с разработанными структурной схемой и схемой подключения (рис. 7, 9). Был выполнен монтаж стойки для размещения стенда. На эту стойку был установлен собранный стенд (рис. 19, 20)

Рисунок 19 – Общий вид лабораторного стенда

Рисунок 20 – Конечное устройство стенда

3.2 Тестирование без участия сервера

Первым этапом тестирования разработанной системы стало тестирование без использования сервера. Было выполнено тестирование как управляющих команд, так и скриптов на языке Python. Для подачи команд и вызова скриптов использовалась командная строка Windows.

Для тестирования корректности работы разработанной системы была написана программа для платы Altera DE1-SoC – четырёхразрядный сумматор. Переключатели SW[0] – SW[3] и KEY[0] – KEY[3] при выполнении данной программы выполнят роль входов сумматора, а вывод результата производится на светодиоды LED[0] – LED[3]. Написанная программа была скомпилирована с помощью среды разработки Quartus Prime 18.0 Lite Edition. В результате компиляции был получен файл с расширением .sof, являющийся файлом прошивки для платы DE1-SoC. Данный файл был размещён в папке C:\Scripts и при помощи команды quartus_pgm был прошит в конечное устройство лабораторного стенда. Вывод консольной программы показал, что

процесс прошивки прошёл и завершился успешно (рис. 21).

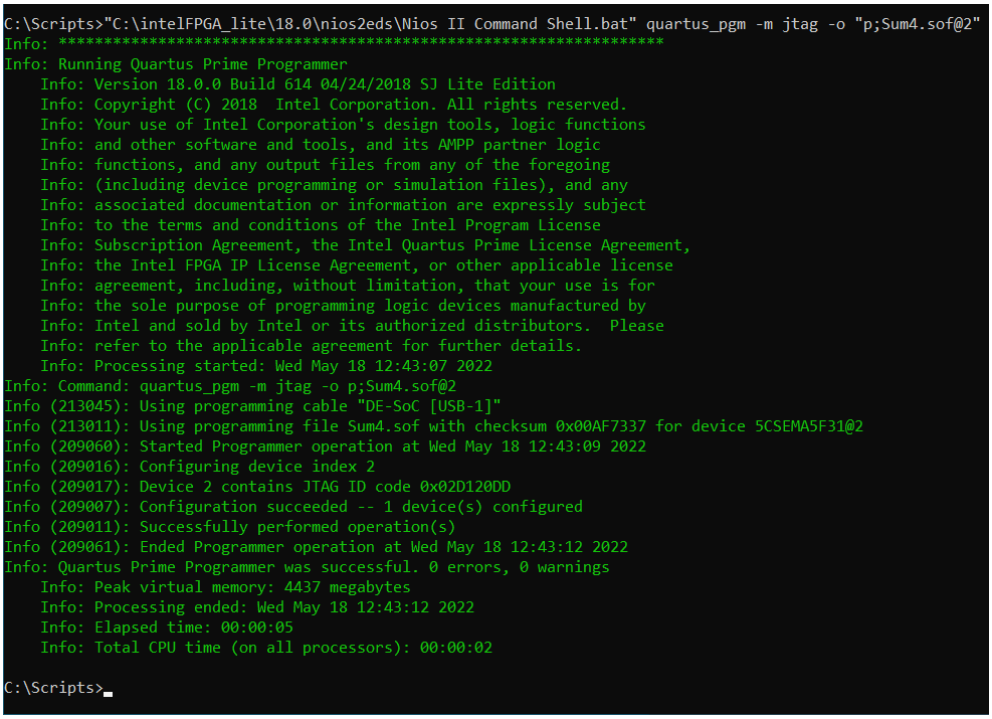

Рисунок 21 – Отчёт о прошивке платы DE1-SoC

После прошивки платы было протестировано управление периферийными устройствами с помощью платы Arduino Uno. При помощи команды echo данные для формирования управляющих сигналов передавались на управляющую плату. После каждого ввода данных светодиоды на плате DE1-SoC загорались и гасли в соответствии с полученными управляющими сигналами. Тесты показали, что управление периферийными устройствами платы DE1-SoC работает корректно.

После завершения тестирования управляющих команд были выполнены тесты для проверки правильности работы скриптов. Тестовая программа была прошита в конечное устройство лабораторного стенда про помощи скрипта FPGA_prog.py. Скрипт был успешно выполнен, в файле C:\Scripts\Log.txt появился отчёт о его выполнении. Прошивка платы завершилась успешно.

Были повторены тесты управления периферийными устройствами. Информация для Arduino подавалась при помощи скрипта FPGA_buttons.py.

Результаты тестов оказались аналогичными предыдущим. Скрипт работает корректно. После каждого выполнения скрипта файл C:\Scripts\Log.txt обновляется и содержит в себе информацию о последней переданной команде.

По завершении тестов прошивки и работы периферийных устройств была протестирована работа скрипта FPGA_clean.py. Данный скрипт удалил из папки C:\Scripts все файлы с расширением .sof, а также прошил в плату DE1-SoC тестовый проект. Данный скрипт также работает корректно.

Таким образом, все проведённые тесты показали положительный результат.

3.3 Тестирование при участии сервера

После завершения первого этапа тестирования на ПК лабораторного стенда был установлен сервер Node.js с API, разработанным рабочей группой проекта. Были проведены тесты с участием сервера, с использованием одного компьютера.

Доступ к API осуществлялся с помощью браузера на ПК лаюораторного 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 Altera DE1-SoC, предусмотрены следующие маршруты (рис. 22):

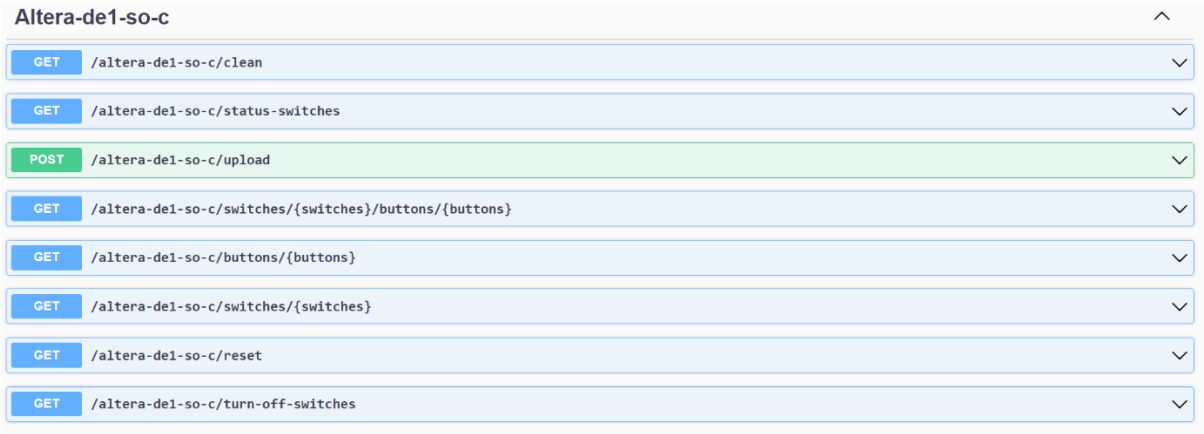

Рисунок 22 – Список маршрутов

1. /altera-de-1-so-c/clean – get-запрос, используемый для удаления загруженных пользователем на сервер файлов прошивки платы, а также прошивки в конечное устройство стенда тестового проекта. Вызывает скрипт FPGA_clean.py;

2. /altera-de-1-so-c/status-switches – get-запрос, используемый для отслеживания текущего статуса переключателей: сколько и какие из них в положениях «выключено» и «включено»;

3. /altera-de-1-so-c/upload – post-запрос, используемый для загрузки на сервер файлов прошивки конечного устройства и его прошивки. Вызывает скрипт FPGA_prog.py с именем загруженного файла в качестве входного параметра;

4. /altera-de-1-so-c/switches/{switches}/buttons/{buttons} – get запрос, используемый для управления периферийными устройствами платы DE1-SoC. {switches} и {buttons} – данные о состоянии переключателей и кнопок в виде набора нулей и единиц, при этом «1» означает изменение состояния устройства, а «0» показывает, что состояние устройства не изменилось. Данный запрос вызывает скрипт FPGA_buttons.py с параметрами, полученными в запросе;

5. /altera-de-1-so-c/buttons/{buttons} – get-запрос, используемый для управления кнопками. Аналогичен предыдущему запросу, с тем исключением, что в данном запросе отсутствует информация о переключателях. Данный запрос вызывает скрипт FPGA_buttons.py с данными о кнопках, полученными в запросе, и нулями вместо данных о переключателях;

6. /altera-de-1-so-c/switches/{switches} – get-запрос, используемый для управления переключателями. Аналогичен запросу 4, с тем исключением, что в данном запросе отсутствует информация о кнопках. Данный запрос вызывает скрипт FPGA_buttons.py с данными о переключателях, полученными в запросе, и нулями вместо данных о кнопках;

7. /altera-de-1-so-c/reset – get-запрос, используемый для перепрошивки конечного устройства файлом, ранее загруженным пользователем на сервер.

Вызывает скрипт FPGA_prog.py с именем загруженного файла в качестве входного параметра;

8. /altera-de-1-so-c/turn-off-switches – get-запрос, используемый для приведения всех переключателей и кнопок в положение «выключено».

После загрузки файла с расширением .sof с помощью маршрута /altera-de-1-so-c/upload он сохраняется в папке C:\Scripts и запускается скрипт FPGA_prog.py, который срабатывает аналогично предыдущим проведённым тестам. В файле логов C:\Scripts\Log.txt появляется отчёт об успешном завершении прошивки, так же, как и на странице в браузере (рис. 23)

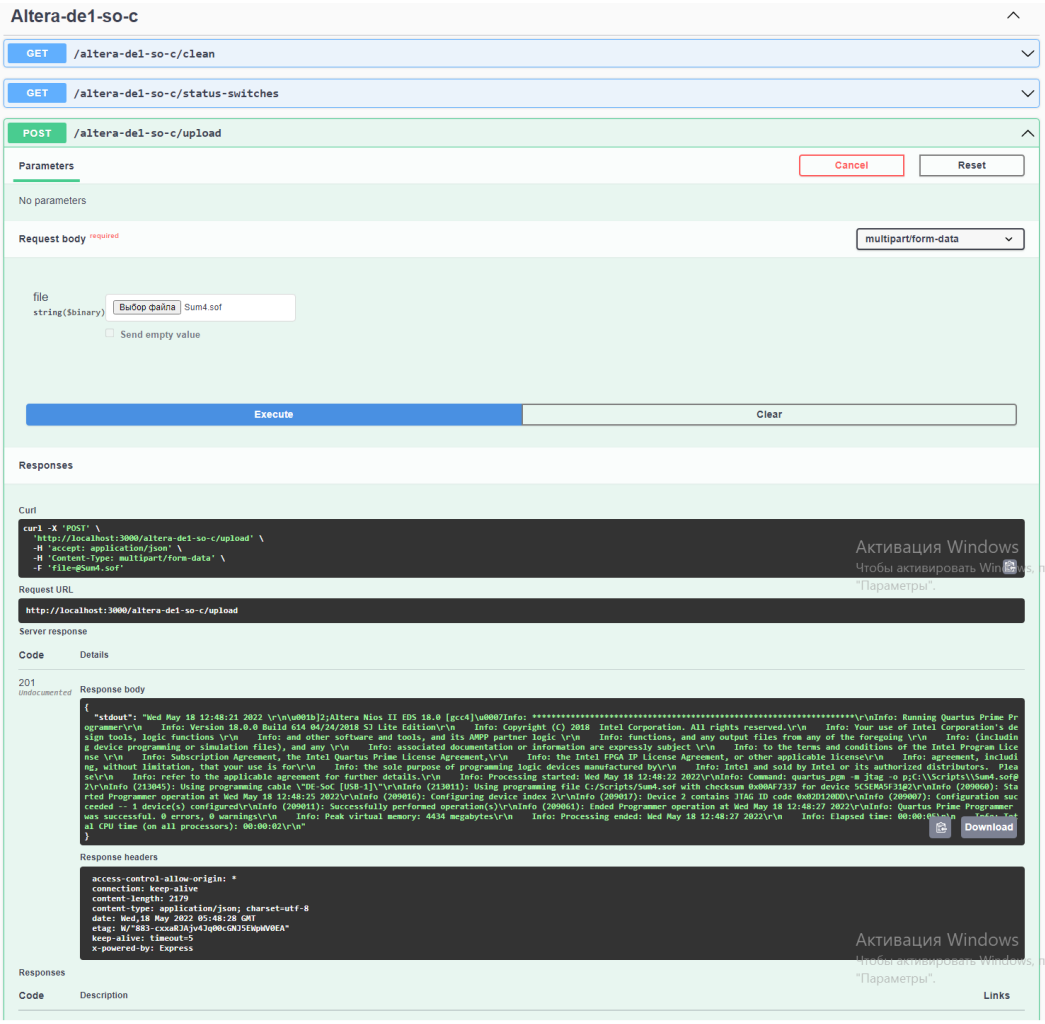

Рисунок 23 – Отчёт о завершении прошивки

Были проверены маршруты /altera-de-1-so-c/buttons/{buttons}, /altera-de-

1-so-c/switches/{switches}, и /altera-de-1-so-c/switches/{switches}/buttons/{buttons}. При выполнении каждого из этих запросов запускается скрипт FPGA_buttons.py с необходимыми параметрами. Результаты тестов оказались положительными.

При выполнении запроса reset происходит перепрошивка платы DE1-SoC ранее загруженным файлом с помощью скрипта FPGA_prog.py.

По запросу clean выполняется скрипт FPGA_clean.py, срабатывающий, как в предыдущих тестах.

По запросу /altera-de-1-so-c/turn-off-switches все переключатели и кнопки приходят в положение «выключено», что является успешным результатом теста.

Результаты выполнения всех скриптов отображаются в файле C:\Scripts\Log.txt. Результаты всех проведённых тестов совпали с результатами, полученными при тестировании системы без сервера.

3.4 Тестирование в режиме «точка-точка»

По завершении этапа тестирования системы на ПК лабораторного стенда был начат третий этап тестирования – тестирование в режиме «точка-точка». Было выполнено подключение к серверу с компьютера, подключённого к сети СФУ. С помощью браузера был открыт сайт, находящийся по адресу http://10.3.3.20:4200/. После авторизации на этом сайте можно увидеть изображение с камеры, подключенной к лабораторному стенду, а также пользовательский интерфейс для взаимодействия с ним. На рисунке 24 представлен интерфейс сайта.

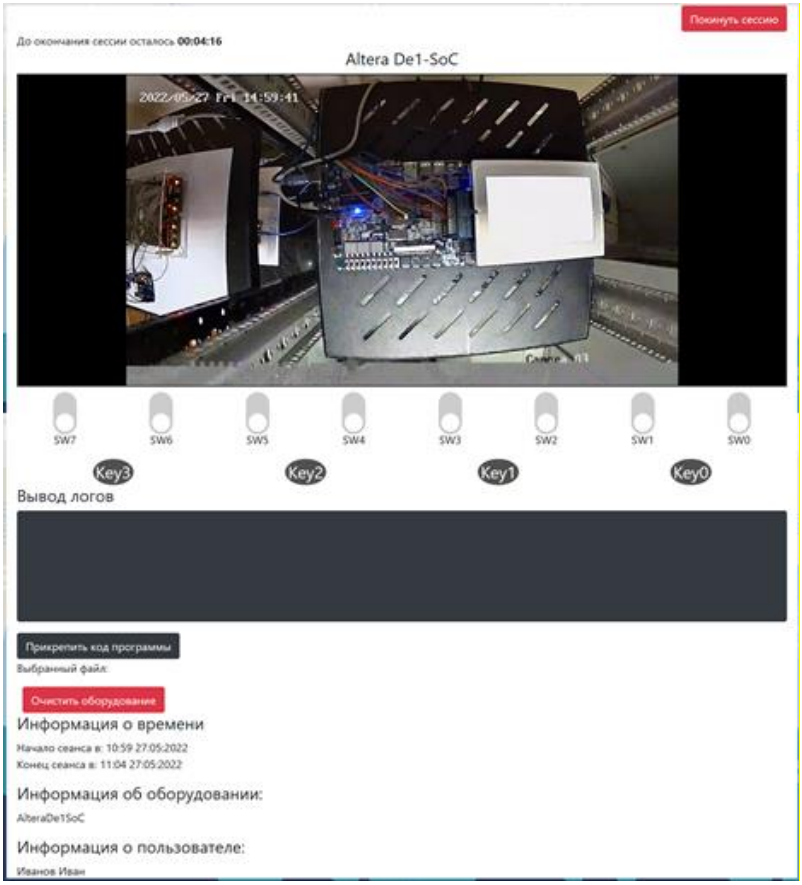

Рисунок 24 – Интерфейс сайта

Элементы интерфейса сайта при взаимодействии с ними генерируют запросы, описанные выше. Было проведено тестирование загрузки файла прошивки на сервер, управления периферийными устройствами платы DE1-SoC, перепрошивки конечного устройства стенда ранее загруженным файлом, приведения всех переключателей и кнопок в положение «выключено» и удаления загруженных файлов с ПК лабораторного стенда. Все проведённые тесты показали результаты, совпавшие с результатами предыдущих тестов. Тестирование показало положительные результаты.

3.5 Выводы

Было проведено три этапа тестирования: тестирование без участия сервера, тестирование с сервером в однопользовательском режиме 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 Altera DE1-SoC работает корректно в режиме лабораторных испытаний.

3. Несмотря на штатное функционирование ПО при доступе к плате обнаружилась значительная задержка серверного оборудования (приблизительно до 3-х секунд) при трансляции видео.

4. При перезагрузке сервера возникает ошибка: сбой управляющей платы Arduino Uno, которая решается перезагрузкой управляющей платы и 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 оборудования.

6. 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 Altera DE1-SoC рассмотрена разработанная рабочей группой общая архитектура и организация сетевого взаимодействия аппаратных средств, а также разработана архитектура лабораторного стенда с конечным устройством DE1-SoC и предложен способ взаимодействия элементов стенда, что позволило перейти к выбору требуемого аппаратного обеспечения и дальнейшей разработке аппаратной и программной частей лабораторного стенда. Также была выполнена интеграция ПО Quartus Prime для программирования конечного устройства стенда в режиме удалённого доступа. На этом этапе была разработана программа для управляющей платы, позволяющая управлять периферийными устройствами платы Altera DE1-SoC с помощью Arduino Uno. Также были разработаны скрипты для серверного ПО, позволяющие автоматизировать ввод управляющих команд, и программа для последующего тестирования системы.

На заключительном этапе работ, было проведено 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 корректное 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, 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
ПК	– Персональный компьютер
ПЛИС	– Программируемая логическая интегральная схема
ПО	– Программное обеспечение
ЦПУ	– Центральное процессорное устройство
API	– Application Programming Interface
ARM	– Advanced RISC Machine
COM	– Communications Port
DDR3 SDRAM	– Double-data-rate Three Synchronous Dynamic Random Access Memory
FPGA	– Field-Programmable Gate Array
GND	– Ground
GPIO	– General Purpose Input/Output
HLS	– HTTP Live Streaming
IDE	– Integrated Development Environment
IP	– Internet Protocol
JTAG	– Joint Test Action Group
RISC	– Reduced Instruction Set Computer
RTSP	– Real Time Streaming Protocol
SoC	– System-on Chip
USB	– Universal Serial Bus

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. А.В. Трухин. «Об использовании виртуальных лабораторий в образовании» /А.В. Трухин // Открытое и дистанционное образование. – 2002. – № 4 – URL: https://kpfu.ru/docs/F1666571770/statya_Truhin_ob_ispolzovanii_virtualnyh_laboratorij_v_obrazovanii.pdf (дата обращения: 23.12.2021).

2. Центр цифровых образовательных ресурсов // Томский политехнический университет [сайт]. – URL: https://portal.tpu.ru/ceor/v_lab (дата обращения: 21.04.2022).

3. Средства отладки. Программные, внутрисхемные симуляторы, отладочные платы [Электронный ресурс]. – Режим доступа: URL: https://pue8.ru/protsessory/697-sredstva-otladki-programmnye-vnutriskhemnye-simulyatory-otladochnye-platy-chast-1.html – Загл. с экрана (дата обращения: 20.04.2022).

4. 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).

5. Multisim [Электронный ресурс]. – Режим доступа: – URL: https://www.ni.com/ru-ru/support/downloads/software-products/download.multisim.html#312060 – Загл. с экрана (дата обращения: 23.12.2021).

6. 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).

7. LabsLand [сайт]. – URL: https://miem.hse.ru/edu/ce/cadsystem/remote_access (дата обращения: 21.04.2022).

8. DE1-SoC (TERASIC TECHNOLOGIES L.L.C. (USA)) [Электронный ресурс]. – Режим доступа: URL: https://krs.terraelectronica.ru/product/1297799 – Загл. с экрана (дата обращения: 21.04.2022).

9. Плата разработчика DE1-SoC. Обзор [Электронный ресурс]. – Режим доступа: URL: http://we.easyelectronics.ru/plis/plata-razrabotchika-de1-soc-obzor.html – Загл. с экрана (дата обращения: 21.04.2022).

10. Обзор плат на SoC ARM+FPGA. Часть 2. Мир Intel (Altera) [Электронный ресурс]. – Режим доступа: URL: https://habr.com/ru/post/406599/ – Загл. с экрана (дата обращения: 21.04.2022).

11. DE1-SoC User Manual

12. Quartus / Linux: Programming the FPGA with command-line [Электронный ресурс]. – Режим доступа: URL: https://www.01signal.com/vendor-specific/intel/fpga-command-line-jtag/ – Загл. с экрана (дата обращения: 21.04.2022)

13. Как использовать командную строку в для загрузки прошивки в Quartus II [Электронный ресурс]. – Режим доступа: URL: https://soltau.ru/index.php/plis/item/505-kak-ispolzovat-komandnuyu-stroku-v-dlya-zagruzki-proshivki-v-quartus-ii – Загл. с экрана (дата обращения: 21.04.2022)

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

ПРИЛОЖЕНИЕ А

Листинги управляющих программ

sketch_FPGA.ino

```
#include <Math.h>

const int lenght = 17;
byte buf[lenght];
int rlen;
byte incomingByte;

void setup() {
  for (byte i = 2; i <= 13; i++) {
    pinMode(i, OUTPUT);
  }
  Serial.begin(9600);
}

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]=='k'&&buf[1]=='e'&&buf[2]=='y'){
      for(int i = 2; i <= 13; i++) { //работа со свитчами и кнопками
        if (buf[i+1] == '1'){     //1 - "изменение состояния свитча"
          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);
        }
      }
    }
  }
  delay(10);
}
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А

FPGA_prog.py

```
import sys
import os
import time
import cgi

print('HTTP/1.0 200 OK')
print('Content-type: text/html')
print('')

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

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

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

#Формируем команду для cmd
com = 'cmd /C "C:\\intelFPGA_lite\\18.0\\nios2eds\\Nios II Command Shell.bat" quartus_pgm -m jtag -o "p;%s@2"' % (file_name)

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

FPGA_buttons.py

```
import sys
import os
import cgi
import time

print('HTTP/1.0 200 OK')
print('Content-type: text/html')
print('')

#Получаем параметры
com_word = sys.argv[1]
com_switches = sys.argv[2]
com_keys = sys.argv[3]

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

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

#Формируем команду
if com_word == 'key':
   com = 'cmd /C echo ' + com_word + com_switches + ' ' + com_keys + ' > COM3'
   com_log = 'echo ' + com_word + com_switches + ' ' + com_keys + ' >> C:\\Scripts\\Log.txt'
if com_word == 'cle':
   com = 'cmd /C echo ' + com_word + ' > COM3'
   com_log = 'echo ' + com_word + ' >> C:\\Scripts\\Log.txt'

#Отправляем команду в cmd
os.system(com)

#Записываем в лог
os.system(com_log)
```

FPGA_clean.py

```
import os
import sys
import time
import cgi

print('HTTP/1.0 200 OK')
print('Content-type: text/html')
print('')

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

#Записываем время в лог
time_com = 'cmd /C echo %s > C:\\Scripts\\Log.txt' % (time_start)
print(time_start + ' Script FPGA_clean worked!')
os.system(time_com)

#Формируем команду
com = 'cmd /C "C:\\intelFPGA_lite\\18.0\\nios2eds\\Nios II Command Shell.bat" quartus_pgm -m jtag -o "p;C:\FPGA\test.sof@2"'

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

#Удаляем sof-файлы с помощью python
fileDir = r"C:\Scripts"
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А

```
fileExt = r".sof"
for file_name in os.listdir(fileDir):
    if file_name.endswith(fileExt):
        com = 'cmd /C echo Deleted: %s >> C:\\Scripts\\Log.txt' % (file_name)
        os.remove("C:\\Scripts" + file_name)
        os.system(com)
```

ПРИЛОЖЕНИЕ Б

Листинг тестовой программы

sum4.v

```
module Sum4(
	p,
	a,
	b,
	p0,
	s
);


input wire	p;
input wire	[3:0] a;
input wire	[3:0] b;
output wire	p0;
output wire	[3:0] s;

wire	SYNTHESIZED_WIRE_0;
wire	SYNTHESIZED_WIRE_1;
wire	SYNTHESIZED_WIRE_2;


sum1	b2v_inst(
	.a(a[3]),
	.b(b[3]),
	.p(SYNTHESIZED_WIRE_0),
	.p1(p0),
	.s(s[3]));


sum1	b2v_inst1(
	.a(a[2]),
	.b(b[2]),
	.p(SYNTHESIZED_WIRE_1),
	.p1(SYNTHESIZED_WIRE_0),
	.s(s[2]));


sum1	b2v_inst2(
	.a(a[1]),
	.b(b[1]),
	.p(SYNTHESIZED_WIRE_2),
	.p1(SYNTHESIZED_WIRE_1),
	.s(s[1]));


sum1	b2v_inst3(
```

```
        .a(a[0]),
        .b(b[0]),
        .p(p),
        .p1(SYNTHESIZED_WIRE_2),
        .s(s[0]));


endmodule
```

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 Altera DE1-SoC»
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		П.Д. Неустроев инициалы,фамилия
Нормоконтролер	_____ подпись, дата	доцент.каф. ВТ ИКИТ Канд. физ.-мат. наук должность, ученая степень	К.В. Коршун инициалы,фамилия

Красноярск 2022