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
«_____» _____ 2022 г.

БАКАЛАВАРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специальности

Мобильное приложение удалённого доступа к лабораторному оборудования для IOS
тема

Руководитель	_____	Старший преподаватель	И.В. Матковский
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		Д.О. Непомнящий
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	Старший преподаватель	И.В. Матковский
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном процессе обучения немалая часть занятий выделяется на работу с лабораторным оборудованием. В ходе лабораторных работ студенты все чаще работают с персональными компьютерами и стендами, установленными в аудиториях. Часто количество оборудования для обучения не может покрыть спрос студентов на работу с ним.

Помимо этого, у студента не всегда есть возможность присутствовать в аудитории для выполнения лабораторной работы в установленный срок. Также нельзя исключать и поломку оборудования. Процесс обслуживания стенда или ПК может надолго приостановить возможность проведения занятий, связанных с ними. Каждое рабочее место необходимо подготовить, настроить и адаптировать для работы с ним.

Для упрощения взаимодействия студента и преподавателя с лабораторным оборудованием актуальна задача предоставления удаленного доступа.

Особенно востребованной является реализация мобильных приложений удаленного доступа для тех случаев, когда нет возможности воспользоваться персональным компьютером или ноутбуком.

1 Анализ предметной области – технологии удаленного доступа посредством мобильных устройств

1.1 Удаленный доступ, основные понятия и определения

Удаленный доступ - достаточно широкое понятие, под которым подразумевают различные способы и варианты организации взаимодействия компьютеров, сетей и приложений, при рассмотрении схем удаленного доступа выделяют использование глобальных каналов, локальных или глобальных сетей. При удаленном доступе, отмечается несимметричность взаимодействия. При этом, когда, с одной стороны, имеется центральная сеть или центральный компьютер, а с другой - отдельный удаленный компьютер или локальная сеть, посредством которых пользователь получает доступ к информационным ресурсам основной сети [1].

В данном случае нам интересен удаленный доступ к оборудованию (научному, лабораторному или контрольно-проверочному). Такой доступ организуется посредством удаленного рабочего стола.

Удаленный рабочий стол (Remote Desktop) — это такой режим управления, при котором один компьютер получает права администратора по отношению к другому, удаленному. Связь между компьютерами организуется в реальном времени через Интернет или по локальной сети [2]. При этом режимы доступа задаются в зависимости от конкретных задач и могут конфигурироваться, например:

- подключение к рабочей сессии обеспечивает полный контроль и взаимодействие с удаленным компьютером. При этом разрешается запуск на нем приложений и любые манипуляции с файлами.

- в режиме ограниченного удаленного доступа разрешено вести наблюдения за процессами, а какие-либо изменения в системе недопустимы.

Обеспечение удаленного доступа и удаленного администрирования поддерживается практически всеми операционными системами. Кроме того,

имеется множество приложений, обеспечивающих удаленный доступ, которые расширяют встроенные функции операционных систем [3,4].

На основании темы и задания на ВКР нас интересуют программы и мобильные приложения позволяющие создавать виртуальные рабочие места с доступом к реальному оборудованию.

1.2 Программы и мобильные приложения для создания виртуальных рабочих мест

Среди подобных программ выделяют несколько типов в зависимости от операционной системы, под управлением которой они функционируют. Например, утилиты для MS Windows, Android и пр. При этом программы функционируют не только через мобильный интернет, но и по Bluetooth или Wi-Fi.

В Windows компания Microsoft встроила возможность удаленного доступа без установки дополнительного ПО. ОС использует протокол RDP, который хорошо защищен и считается довольно безопасным. Однако есть существенный недостаток, который состоит в том, что сервером может быть только компьютер или ноутбук с установленной Windows Pro или выше. При этом допускается подключение любых устройства под управлением Windows 7, 8, 10 и на базе Android [5].

Одной из распространенных программ является TeamViewer. В этом ПО имеется поддержка русского языка, а также простой и понятный интерфейс. TeamViewer доступен через Интернет в открытом доступе. Для личного пользования можно загрузить пробную версию приложения. Для единовременного подключения нет необходимости устанавливать на компьютер программу. Но для регулярного использования требуется устновка расширенной версии, которая поддерживающую совместную работу Windows 7, 8, 10 и Linux (Рисунок 1). Также имеются приложение для ОС Android и iOS.

Рисунок 1 – Интерфейс пользователя TeamViever

Компания Google предлагает инструмент - дополнение к браузеру Google Chrome. Данное ПО функционирует на всех ОС, поддерживающих Google Chrome. Кроме того расширение можно использовать на Android и iOS.

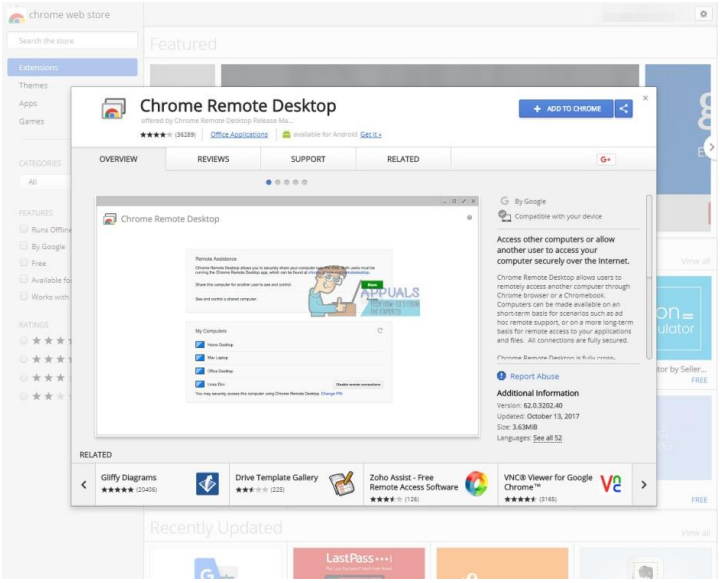

Рисунок 2 – Интерфейс Chrome Remote Desktop

Для использования приложения на персональном компьютере необходимо иметь аккаунт Google. Недостатком является довольно сложный интерфейс пользователя, а достоинством отсутствие необходимости инсталляции и высокий уровень безопасности, гарантированный Google.

Интерес вызывает небольшая программа AeroAdmin. Она имеет русскоязычный и простой интерфейс пользователя. По существу – это оптимальное ПО подобного рода для начинающих. Программа не имеет

расширенного функционала, но поддерживает самые необходимые функции. При этом ПО не нуждается в установке. Приложение имеет всего один файл, и не занимает много места на диске.

Из мощных программ выделяется функциональная утилита Remote Utilities, которая предусматривает коммерческое и личное использование. ПО позволяет организовать удаленный доступ одновременно для десяти персональных компьютеров (Рисунок 3).

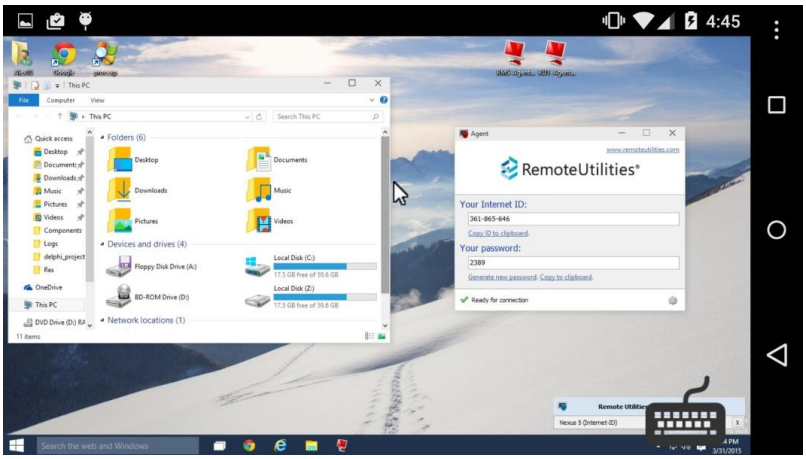

Рисунок 3 – Интерфейс Remote Utilities

ПО поддерживает следующие функции:

- открытие и получение доступа к реестру;
- поддержка двух и более мониторов;
- передача файлов посредством технологии Drag-n-Drop;
- прочие расширения.

Рассмотрим приложения для Android. Например, небольшое приложение Unified Remote, которое может преобразовать смартфон в пульт управления персональным компьютером. Поддерживает Windows, Linux и Mac. При этом пользователь может не только загрузить бесплатную версию, но и получить доступ ко всем программным модулям после единовременной оплаты.

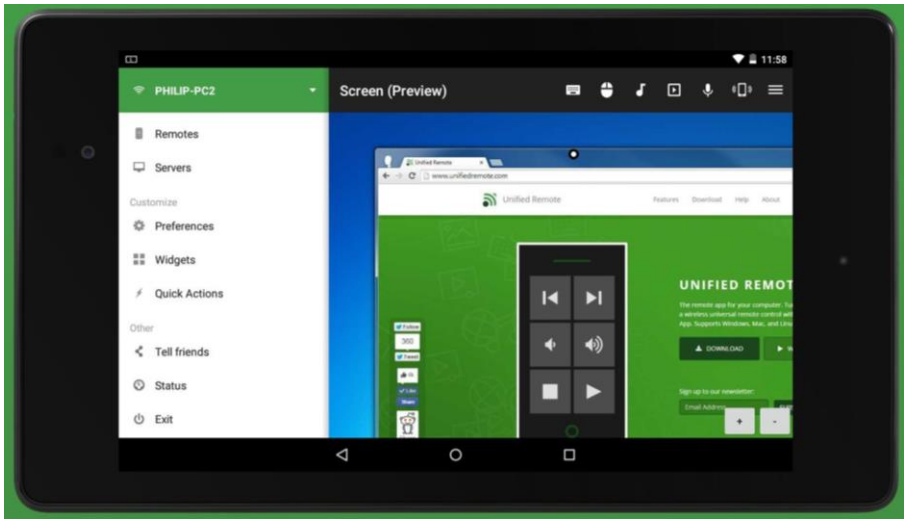

Рисунок 4 – Unified Remote

Программа Splashtop 2 поставляется в бесплатно, однако есть и лицензионная версия с полным функционалом. Это ПО пользуется спросом у системных администраторов и специалистов по инсталляции и настройке ПО, поскольку обеспечивает надежный и безопасный доступ к клиентским машинам.

Приложение KiwiMote позволяет управлять компьютером через смартфон, но устройства должны быть подключены к одной точке доступа. Для организации контроля, необходимо ввести на мобильном устройстве пин-код или воспользоваться QR-кодом. При помощи этого ПО можно печатать текст или управлять проигрывателями.

Рисунок 5 – KiwiMote

Из всех рассмотренных программных систем отдельно был выделен AnyDesk. Основным преимуществом данного ПО является его

кроссплатформенность. ПО может использоваться практически на любой ОС: Windows, macOS, Android, Linux, FreeBSD, Raspberry Pi и ОС Chrome. ПО обеспечивает независимый от платформы удаленный доступ к различным устройствам, например ПК и хост-устройства, что особенно актуально в нашем случае. ПО поддерживает удаленный доступ, передачу файлов, функции VPN, а также безопасный и надежный доступ.

Возможна организация удаленного доступ к рабочему столу, файлам или/и документам из любой точки посредством Интернет. В AnyDesk встроена адресная книга, которая позволяет отслеживать соединения и контакты, а так же позволяет видеть онлайн-статус этих соединений, что особенно актуально для администратора удаленного доступа к лабораторному оборудованию. Кроме того, разработчики AnyDesk утверждают, что программа будет функционировать сетях с низкой пропускной способностью и плохой связью.

В AnyDesk имеется режим администратора, это позволяет выполнять различные задачи, например, удаленная перезагрузка, создание отчетов о сеансах, анализ отчетов и пр. Кроме того эти инструменты позволяют исправлять и устранять проблемы на удаленном рабочем столе.

AnyDesk поддерживает функцию Drap and Drop, которая позволяет пользователю перетаскивать файлы и документы из хост-системы в клиентскую и наоборот. Но главное – это поддержка удаленного доступа к оборудованию. Пользователь может получить доступ к аппаратуре удаленных систем, например интерфейс USB, CD/DVD – привод и даже управлять включением и выключением питания.

Не менее важной является функция поддержки онлайн-встреч, презентаций или работы над одним и тем же проектом из разных мест.

При всем вышеперечисленном, AnyDesk – это упрощенный инструмент. Для его использования не требуется административный доступ или установка. Необходимо просто скачать EXE-файл (3 Мб) и запустить его (Рисунок 6). [6]

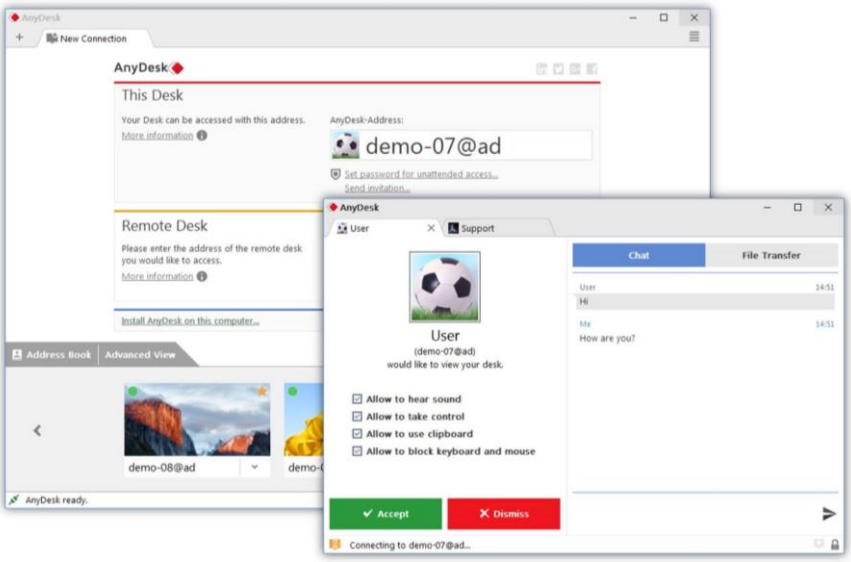

Рисунок 6 – Один из экранов пользователя AnyDesk

Для фиксации процесса выполнения лабораторных работ или проведения экспериментов AnyDesk поддерживает запись экрана - это полезная функция для обучения.

AnyDesk использует одну из самых безпастных систем доступа - стандартную банковскую технологию TLS 1.2. Серверы AnyDesk используют телекоммуникационную технологию Erlang для максимальной надежности.

Ниже перечислены дополнительные функции AnyDesk:

- Удаленный доступ для нескольких платформ
- Передача файлов и диспетчер файлов
- Удаленная печать
- VPN
- Автоматический доступ
- Виртуальная доска
- Автообнаружение
- Функция чата
- REST-API
- Собственные клиенты
- Протокол сеанса
- Двухфакторная аутентификация
- Индивидуальный хост-сервер

Немаловажным преимуществом имеется поддержка AnyDesk для Android.

Однако, AnyDesk является платным/условно-платным приложением и в последних версиях iOS имеет ограниченный функционал.

1.3 Обзор известных решений по управлению лабораторными стендами

1.3.1 Система удаленного доступа НИЯУ ВШЭ

Национальный исследовательский университет «Высшая школа экономики» г. Москва, использует в учебном процессе департамента компьютерной инженерии учебную лабораторию (УЛ) систем автоматизированного проектирования, которая предлагает удаленный доступ к оборудованию УЛ САПР [7].

В лаборатории развернуты рабочие станции с подключенными к ним отладочными платами ПЛИС DE1-SoC, DE10-standart и DE10-lite (Рисунок 7).

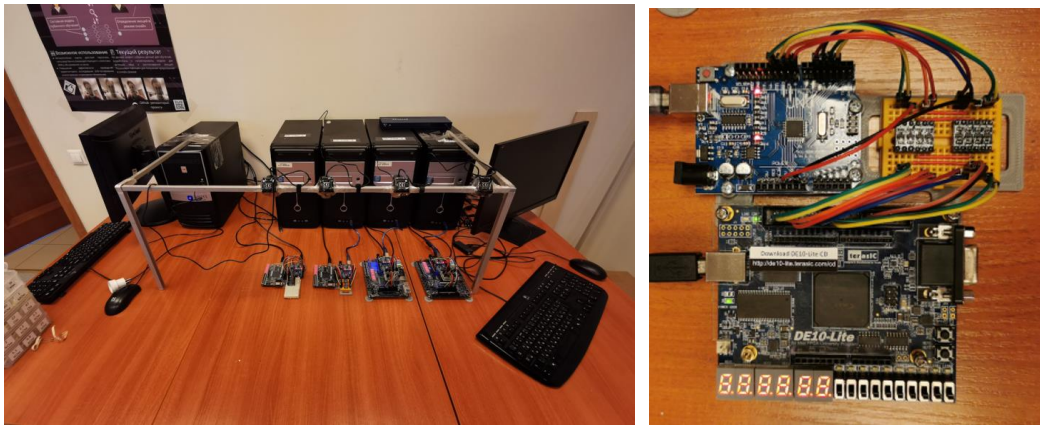

Рисунок 7 – Оборудование УЛ САПР НИУ ВШЭ

На компьютерах установлено все необходимое программное обеспечение Quartus, Modelsim и т.д. Студенты могут подключиться к рабочим станциям через ПО AnyDesk, загрузить или выполнить сборку своего проекта, после чего выполнить программирование стенда.

Для того, чтобы наблюдать за результатом работы проекта на плате стенды снабжены веб-камерами и на ПК установлена программа AMCap. С ее помощью можно наблюдать за платой и выводом информации на семисегментные индикаторы и лампочки индикации. Возможна настройка, зум и коррекция изображения в зависимости от условий освещенности.

Для того, чтобы осуществить управление платами дистанционно, разработано специализированное ПО Butt Emulator. С помощью этой программы студенты через COM-порт подключаются к Arduino, которая связана своими выводами с GPIO на ПЛИС.

Подробное описание режимов работы, способов подключения и прошивки ПЛИС приведено в [8].

Недостатком данной системы является необходимость запуска прикладного ПО на удаленном компьютере, работа с ограниченным типом отладочных плат, а также отсутствие возможности доступа администратора посредством мобильных устройств.

1.3.2 Удаленный доступ к стендам СибГУТИ

Лаборатория электронных средств обучения (ЛЭСО) СибГУТИ, г.Новосибирск предлагает лабораторию с удаленным доступом как альтернативу обучающему моделированию [9].

На текущий момент в образовательный процесс внедрено два комплекса лабораторного практикума по курсам «Физические основы электроники» и «Микроконтроллеры и их применение» и др. Студент, обучающийся по программе дистанционного образования, может, находясь в любой точке планеты через глобальную сеть Интернет, получить доступ к реальному оборудованию и выполнить лабораторную работу.

Например, для предметов по цифровой электронике, используется учебный стенд, который представляет собой печатную плату с программируемой логической интегральной схемой (ПЛИС) Altera EP1C3T144 и направленную на нее web-камеру (Рисунок 8).

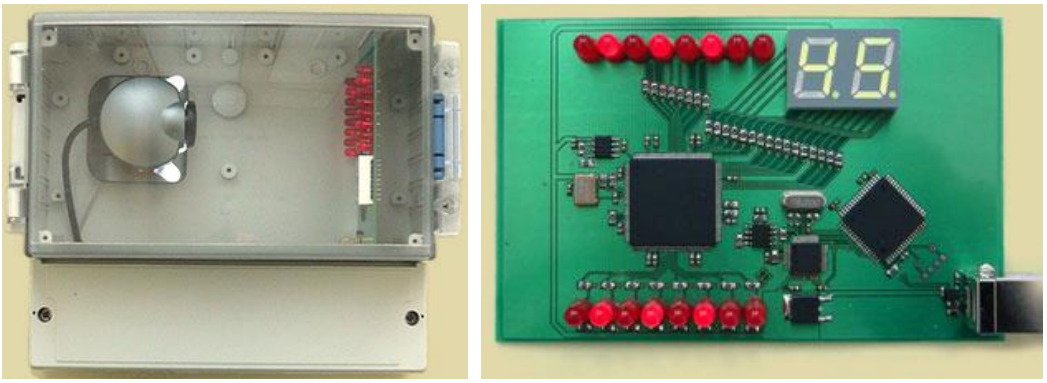

Рисунок 8 – Оборудование СибГУТИ

Печатная плата и web-камера подключены к серверу, имеющему доступ в сеть интернет. Для индикации логических состояний на выводах микросхемы используются светодиоды. Подача управляющих сигналов на входы ПЛИС выполняется с помощью специального микроконтроллера.

Для доступа к удаленной лаборатории по цифровой схемотехнике используется интернет-браузер. Для сборки цифровой схемы любой степени сложности используется файл конфигурации, созданный в среде системы автоматизированного проектирования Quartus II. На открывшейся странице лаборатории студент указывает путь к конфигурационному файлу ПЛИС, загружает его в удаленный стенд и наблюдает за выполнением работы через web-камеру. Программное обеспечение сервера ведет статистику выполнения всех лабораторных работ. По запросу статистика выполнения работ предоставляется преподавателю. Также преподаватель может вмешаться в выполнения работы и, при необходимости, скорректировать действие студента.

Основным недостатком стенда является его узкая направленность (только ПЛИС, только одной модели и только одного производителя) и отсутствие развитой поддержки пользователя/администратора, в том числе мобильных приложений и кроссплатформенности.

1.3.3 Виртуальные лаборатории ТПУ

Онлайн лаборатории национального исследовательского Томского политехнического университета предоставляют широкий сервис к виртуальным

лабораториям от аналитической химии до электроснабжения [10]. Все лабораторные установки виртуальные и представлены в 3D – формате (Рисунок 9), что и является основным недостатком.

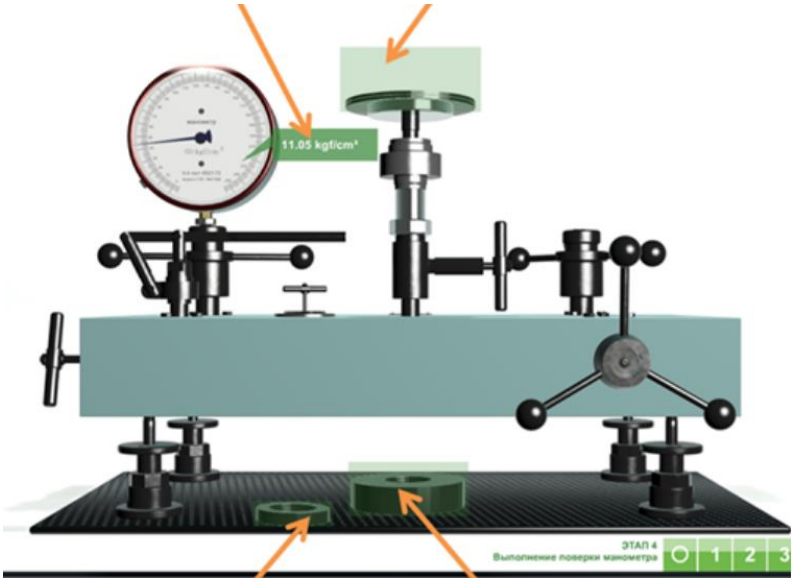

Рисунок 9 – Пример виртуального стенда ТПУ

1.3.4 Роботизированные линии удаленного доступа С-Петербурского политехнического университета им. П. Великого.

Особый интерес вызывает решение С-Петербурского университета им. Петра Великого, представленной Северо-Западным межвузовским региональным учебно-научный центром "СПбПУ - ФЕСТО". В состав консорциума входят более 20-ти ведущих Вузов России и Европы. Центр предоставляет для обучающихся доступ к роботизированным комплексам, роботам и оборудованию в режиме дистанционного доступа. Проводятся региональные и международные соревнования по программированию роботов. Проходят зимние и летние школы, реализуется ряд дистанционных программ подготовки как Российских, так и зарубежных студентов, в области интеллектуального управления и робототехники (Рисунок 10) [13].

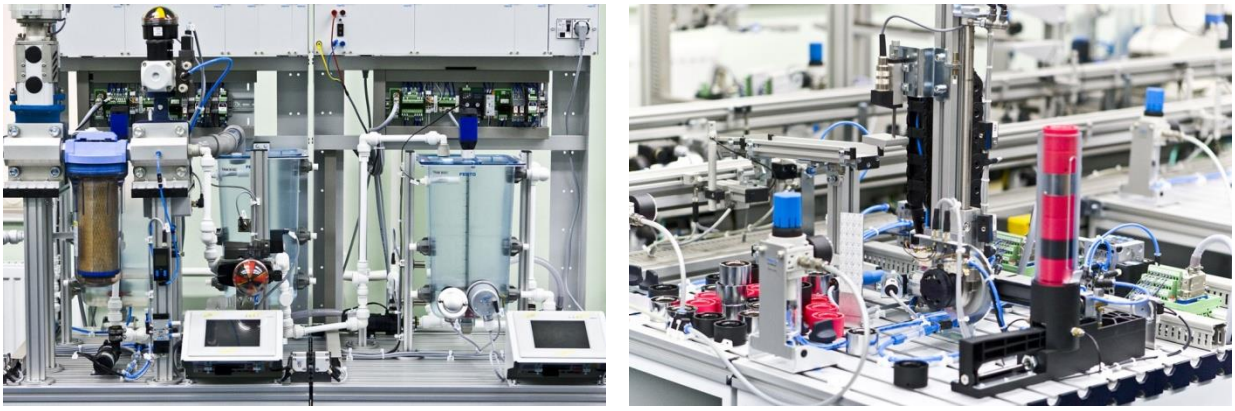

Рисунок 10 – Роботизированные линии удаленного доступа СЗ МУНЦ СПбПУ-ФЕСТО

В зарубежной литературе можно встретить гораздо больше описаний различных методик и практических решений по применения лабораторий удаленного доступа в обучающем процессе и создания программ и сервисов виртуальных и удаленных лабораторий [10-12].

1.4 Инструментальные средства разработки мобильных приложений для сетевого доступа

PhoneGap – бесплатно распространяемое ПО с открытым кодом. Обеспечивает доступ к аппаратуре мобильного устройства. Например, акселерометр, GPS, камера, звуковой контроллер и др. Имеется довольно широкий набор библиотек, которые могут применятся при разработке приложений. Входящий в состав пакета модуль PhoneGap Build позволяет делать сборки для iOS, Android и Windows Phone одновременно, что не требует освоения и установки дополнительного ПО. При этом можно делать сборки для iOS в облаке без Mac. Кроме того, поддерживается разработка приложений для Symbian, Palm, BlackBerry, iPhone и iPad [14]. Этот фреймворк позволяет создавать мобильные приложения, используя JavaScript, HTML и CSS3.

Следующий фреймворк Rhodes основан на языке программирования Ruby, функционирует под RhoMobile. Он позволяет создавать кроссплатформенные приложения, которые совместимы с широким диапазоном

ОС и смартфонов. Есть платный сервер и бесплатное средство для разработки приложений в облаке, но с ограниченным функционалом.

Appcelerator - открытый ресурс фреймворков для мобильных приложений. Этот ресурс является платформой разработки Titanium. Платформа позволяет создавать приложения для телефонов, планшетов и ПК. Поддерживаются языки JavaScript, HTML, PHP, Ruby и Python.

Xamarin – популярный инструмент для разработки приложений на различных языках. Платформа является уникальной, поскольку позволяет разработчикам работать с собственными IDE, API и языками. Этот инструмент поддерживает мониторинг качества и функциональности тестирования для различных устройств. По существу - это моно-фреймворк, он позволяет устанавливать и поддерживать связь с API на мобильных устройствах.

Фреймворк Ionic разработан на языке SASS CSS и является кроссплатформенным. Ionic преимущественно используется для создания гибридных мобильных приложений. У Ionic имеется хорошая библиотека оптимизированных для мобильных устройств компонентов, инструментов и жестов HTML, CSS и JS CSS.

Для начинающих можно рекомендовать платформу Appy Pie. При разработке не нужно изучать языки программирования, а полученные результаты можно публиковать в Google Play, Apple App Store, Windows App Store или в любом другом магазине. При разработке просто собирается готовый продукт из предопределенных функций. Для обучения можно использовать одноимённый канал компании на YouTube.

NativeScript как и предыдущие платформы является кроссплатформенным с открытым исходным кодом, но, в отличие от многих других он бесплатный. Поддерживаются языки JavaScript и TypeScript. В NativeScript имеется полная поддержка фреймворка AngularJS, а полученные мобильные приложения имеют полный доступ API платформ. С помощью NativeScript можно создавать приложения для Android и iOS. Эта платформа имеет интеграцию с Vue.JS, поддерживает множество плагинов, что расширяет ее функциональность [14].

1.5 Выводы по разделу 1

1. Рассмотрены наиболее распространенные технологии обеспечения удаленного доступа к лабораторному оборудованию. Определено, что в нашем случае следует использовать прямой доступ к рабочему оборудованию. При этом нужно обеспечить оба режима доступа: только наблюдение и полный контроль для администратора.
2. Рассмотрены известные программы, для мобильных приложений позволяющие организовать удаленный доступ к оборудованию. Выделен программный пакет AnyDesk обладающий наибольшими функциональными возможностями для нашего проекта. Принято решение при разработке мобильного приложения ориентироваться на предоставляемый этим приложением функционал.
3. Рассмотрены известные решения, применяющиеся в ВУЗах России для организации удаленных и виртуальных лабораторий. Выделена Система удаленного доступа НИЯУ ВШЭ. Принято решение ориентироваться на подобную архитектуру при разработке функционала и принципов взаимодействия мобильного приложения. Однако в отличие от известной системы следует обеспечить доступ к нескольким типам стендов, согласно заданию на ВКР. При этом, такой доступ следует сделать через мобильной устройство.

2 Разработка мобильного приложения для тестирования сетевого соединения и режима удаленного доступа к оборудованию

2.1 Анализ задания на проектирование

2.1.1 Описание средств разработки

Согласно заданию на ВКР, разрабатываемое приложение должно быть реализовано на языке Swift. Этот выбор не случайный, поскольку Swift — это быстрый и эффективный язык программирования, поддерживающий режим реального времени, а код на этом языке легко можно вставить в готовый код Objective-C. Компания Apple рекомендует именно этот язык для разработки. На официальном сайте размещены результаты сравнения приложений реализованных на этом языке с другими популярными языками. Например, обычный алгоритм поиска выполняется в Swift до 2,6 раза быстрее, чем в Objective-C и до 8,4 раза быстрее, чем в Python 2.7 [15,16]. Swift бесплатно доступен для использования разработчиками, преподавателями и студентами по лицензии на распространение ПО с открытым исходным кодом Apache 2.0. Apple предоставляет исходные файлы для OS X и Linux, которые позволяют создавать код для iOS, OS X, watchOS, tvOS и Linux.

Для разработки приложений используется бесплатная IDE XCode от компании Apple. Xcode. Приложение позволяет обрабатывать любые типы данных — от изображений до JSON и PLIST. Для разработки интерфейса в Xcode используется мощный инструмент Interface Builder. С его помощью можно быстро создавать прототип, а позже внести динамику во все детали интерфейса, подключив его к исходному коду. Его можно использовать, чтобы на ходу менять значения переменных и устанавливать точки останова, на которых будет прерываться программа (Рисунок 11).

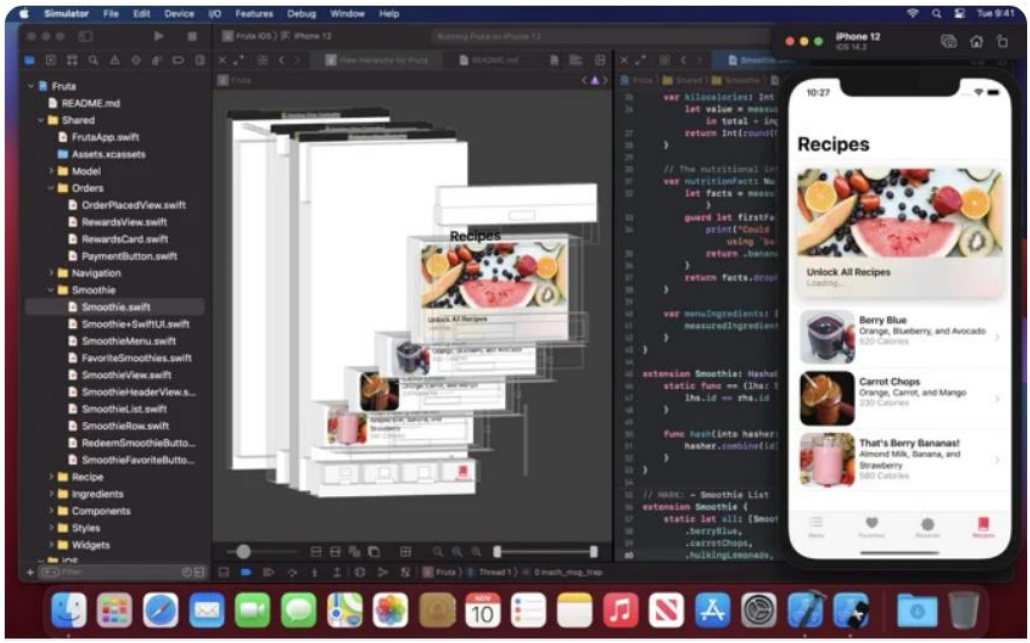

Рисунок 11 – Apple Xcode. Окно разработчика

2.1.2 Архитектура аппаратных средств

Согласно заданию на ВКР, требуется обеспечить взаимодействие пользователей с лабораторным оборудованием. В ИКИТ СФУ разворачивается лаборатория удаленного доступа, состав которой приведен на рисунке 12.

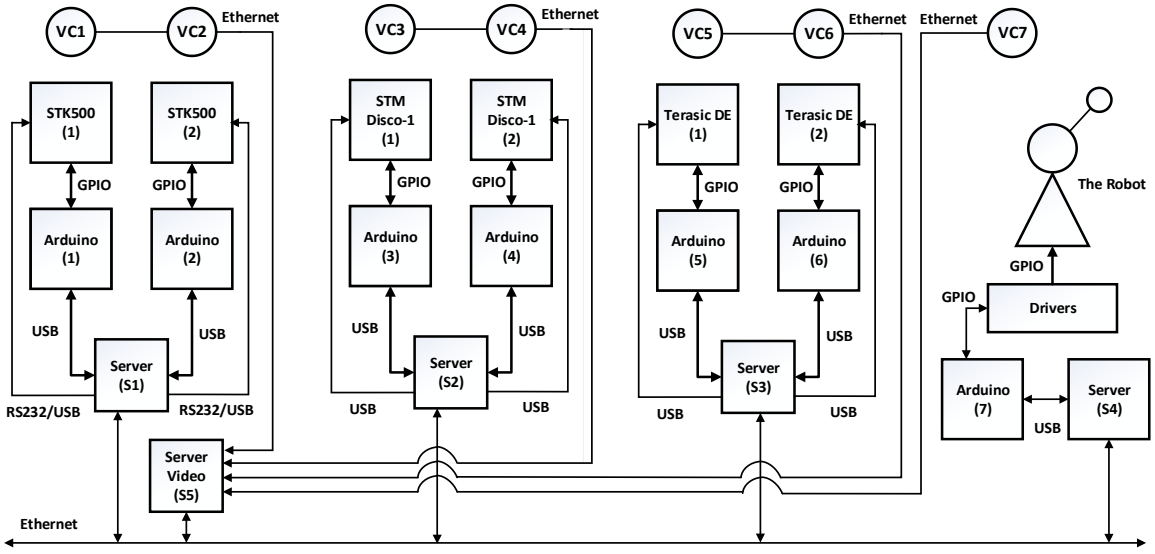

Рисунок 12 – Состав лаборатории удаленного доступа ИКИТ СФУ

В состав лаборатории входят лабораторные стенды STK500, STM32 и DE1-SoC.

Стенды подключены к серверам и управляются посредством одноплатных компьютеров Arduino и/или RAsberyPI. Для преобразования электрических сигналов в сигналы управления конечным оборудованием робота используются усилители (драйвера) двигателей. При подключении лабораторных стендов электрические параметры сигналов не требуют преобразования и поэтому они подключатся непосредственно при помощи кабельной системы. Однако для выполнения некоторых лабораторных работ к стендам подключены дополнительные датчики, интерфейсы и исполнительные устройства. Например, ЦАП, ЖКИ, кнопки и переключатели, они ходят в состав стендов. Для них, при помощи управляющих одноплатных компьютеров, происходит имитация входных сигналов для плат (Рисунок 13).

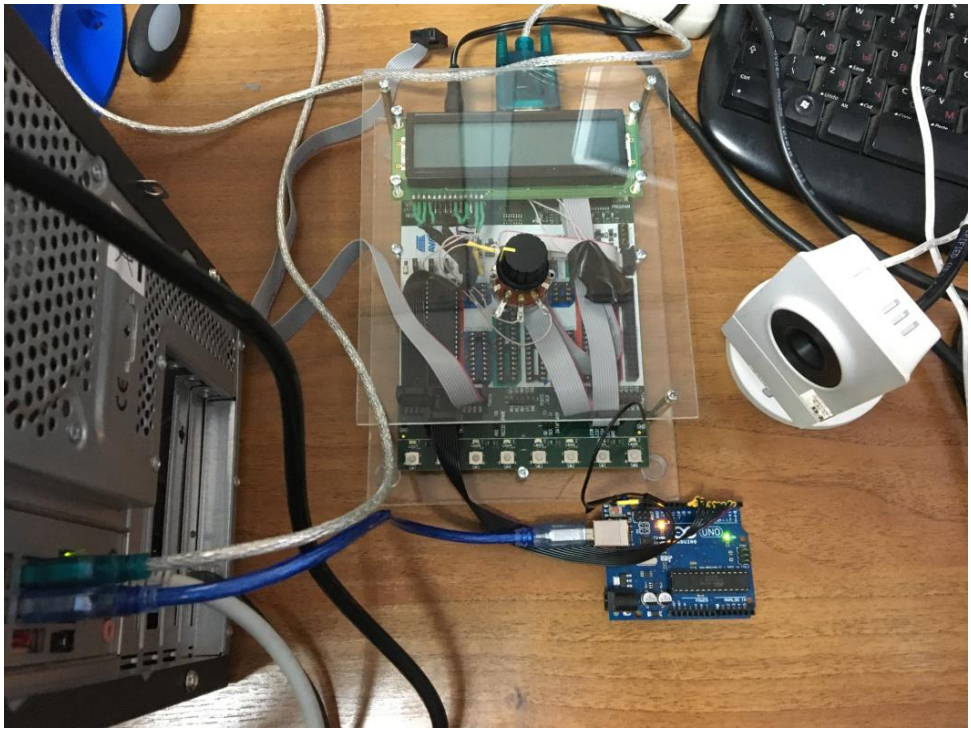

Рисунок 13 – Пример подключения STK500 к серверу

На рисунке 14 представлена общая архитектура системы.

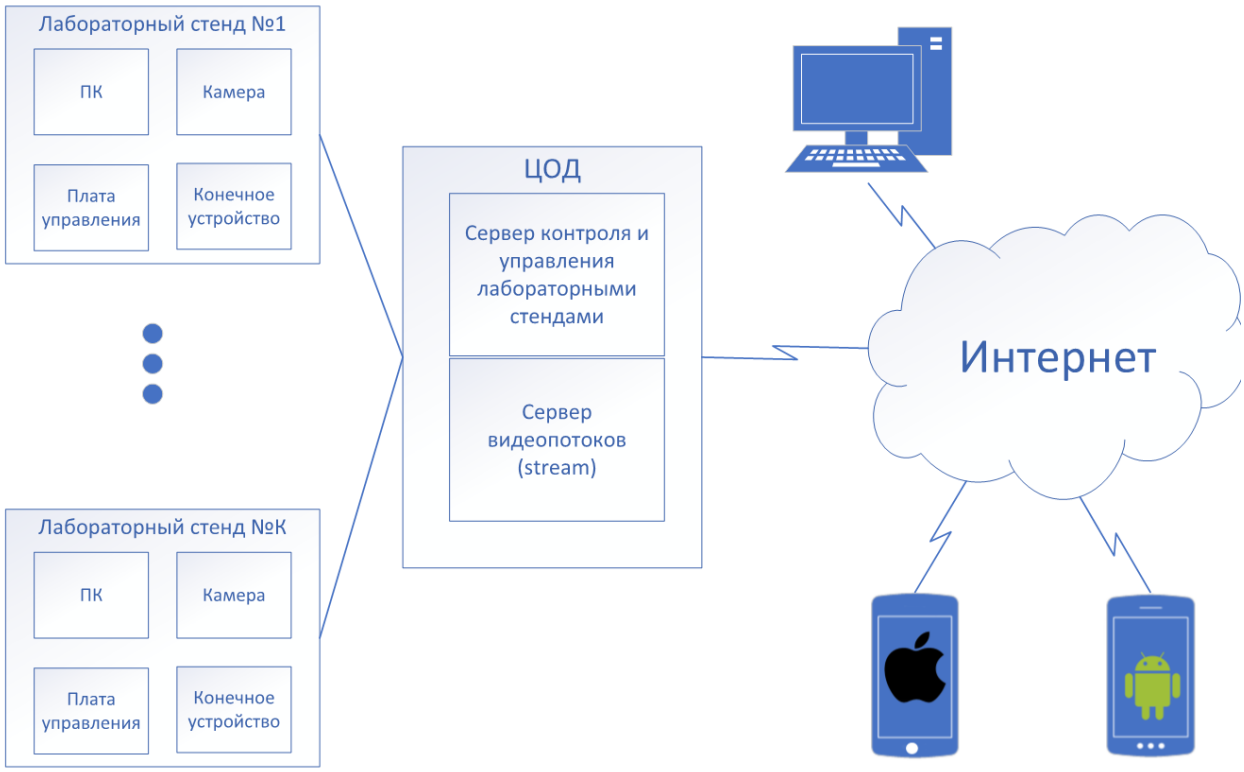

Рисунок 14 – Общая архитектура системы

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 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 отличие от браузера разработанное

мобильное приложение позволяет масштабировать окна, выполнять прокрутку и обеспечивает передачу видеопотока без задержки. Предполагается дальнейшее расширение пользовательских функций.

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.

2.1.3 Организация сетевого взаимодействия

На момент начала разработки доступ был организован в тестовом режиме и реализовывался следующим образом:

1. Администратор подключается к серверу при помощи текстового (терминального) интерфейса, используя встроенные в ОС средства (ssh, PuTTY).
2. На сервер закачивается исходный код для Arduino в текстовом виде.
3. Код компилируется и отправляется в Arduino. Изначально это делалось вручную, далее при помощи разработанных скрипов все операции выполняются автиоматически.
4. После отправки программы в Arduino она начинает выполняться.

При этом изображение работающего оборудования захватывается камерой, выполняется перекодировка видео для требуемых показателей качества.

Видео публикуется в формате HLS, зрители могут просматривать его при помощи браузера или любого видеоплеера. Администратор сам может быть зрителем (Рисунок 15).

Рисунок 15 – Пример видео транслируемого через сервер

Дополнительные материалы по сетевому взаимодействию приведены в следующих разделах пояснительной записки.

2.2 Общие требования к архитектуре мобильного приложения

На основании анализа задания на ВКР были разделены функции мобильного приложения между устройствами, подключаемыми к серверам. Для лабораторных стендов не имеет смысла организовать режим генерации кода со стороны пользователя, поскольку на мобильном устройстве нет возможности редактировать, компилировать, и отлаживать программы, предназначенные для прошивки в память установленных на стендах ЧИПов. Однако для стендов следует предусмотреть режим администратора (преподавателя) В котором возможно отслеживание результатов выполнения лабораторных работ,

например через журнал, администрирование, визуальный контроль за ходом работ, чат со студентами и сбор статистики. При этом для пользователя достаточно ввести возможность визуального контроля хода работы через видеокамеру и чат с преподавателем. Однако при дальнейшем проектировании эти функции могут быть расширены.

Для управления роботом следует реализовать полный функционал, как для администратора, так и для студента. При этом нужно обеспечить не только контроль робота, но и разработку программы для него (лабораторных работ по управлению роботом). Более подробно эти режимы и функции изложены в главе 3 пояснительной записки.

2.3 Мобильное приложение разработчика

Разработана следующая архитектура взаимодействия элементов мобильного приложения для тестирования удаленного доступа (Рисунок 16).

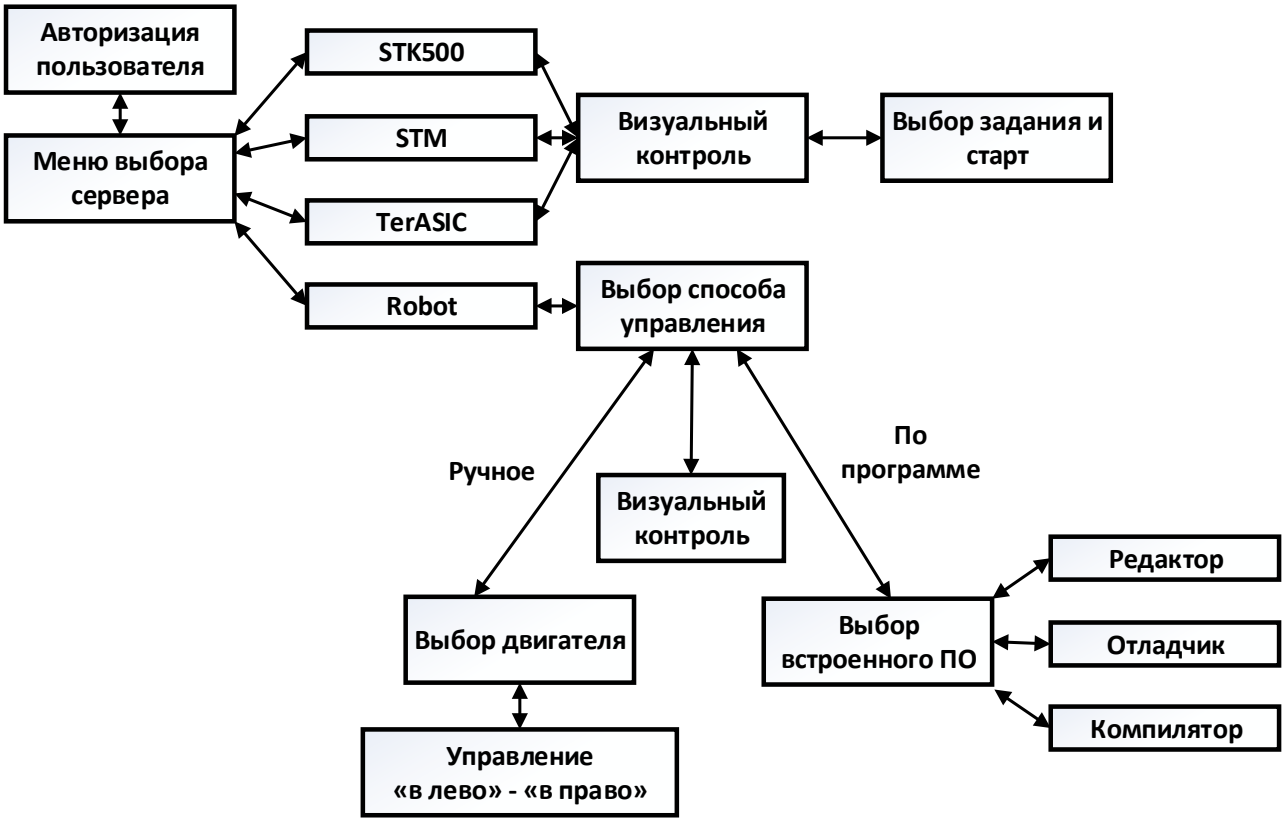

Рисунок 16 – Взаимодействие элементов мобильного приложения разработчика

При запуске мобильного приложения происходит авторизация пользователя в системе (Рисунок 17-а). После этого выбирается сервер для подключения (Рисунок 17-б). Имеются 4 возможных сервера: STK500, STM32, DE1-SoC и ROBOT.

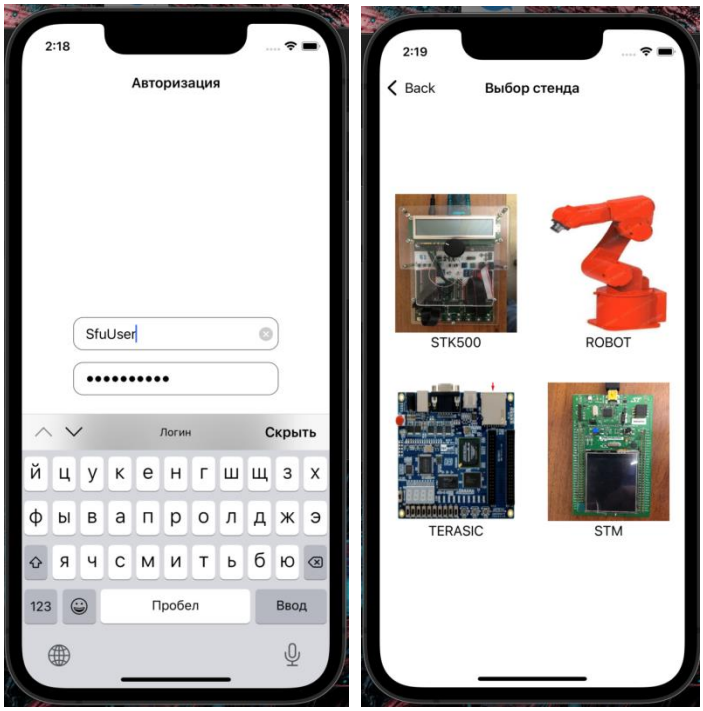

Рисунок 17 – Авторизация и выбор сервера

При подключении к первым трем доступен только режим визуального контроля через web-камеру. Однако предусмотрена возможность выбора тестового задания и его старта в режиме удаленного доступа (Рисунок 18).

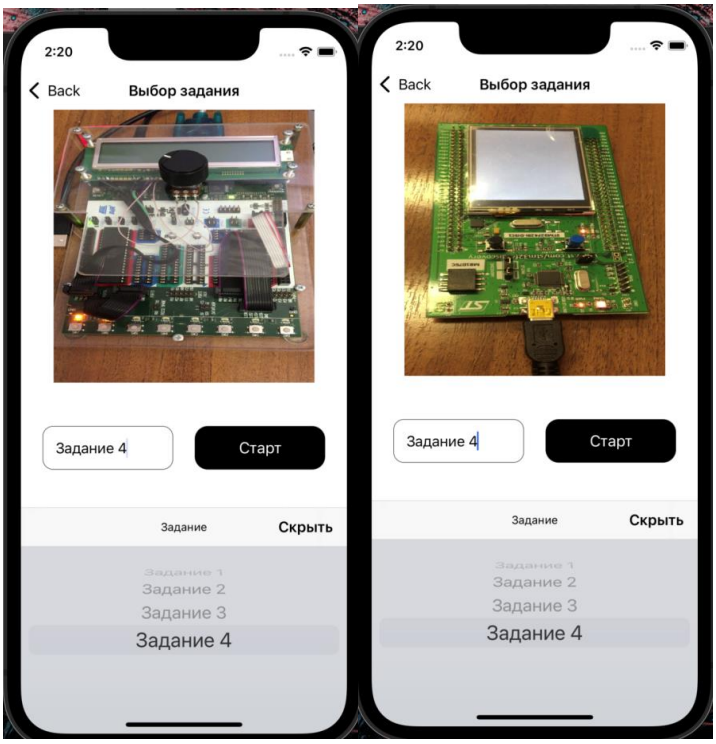

Рисунок 18 – Выбор тестового задания и визуальный контроль за платами

При подключении к роботу доступно ручное управление, а также разработка, отладка, компиляция и выполнение тестовых программ управления роботом (Рисунок 19).

В разработанном приложении предусмотрены запуск редактора текста программы, отладчика и компилятора, которые будут разрабатываться в ходе дальнейшей работы.

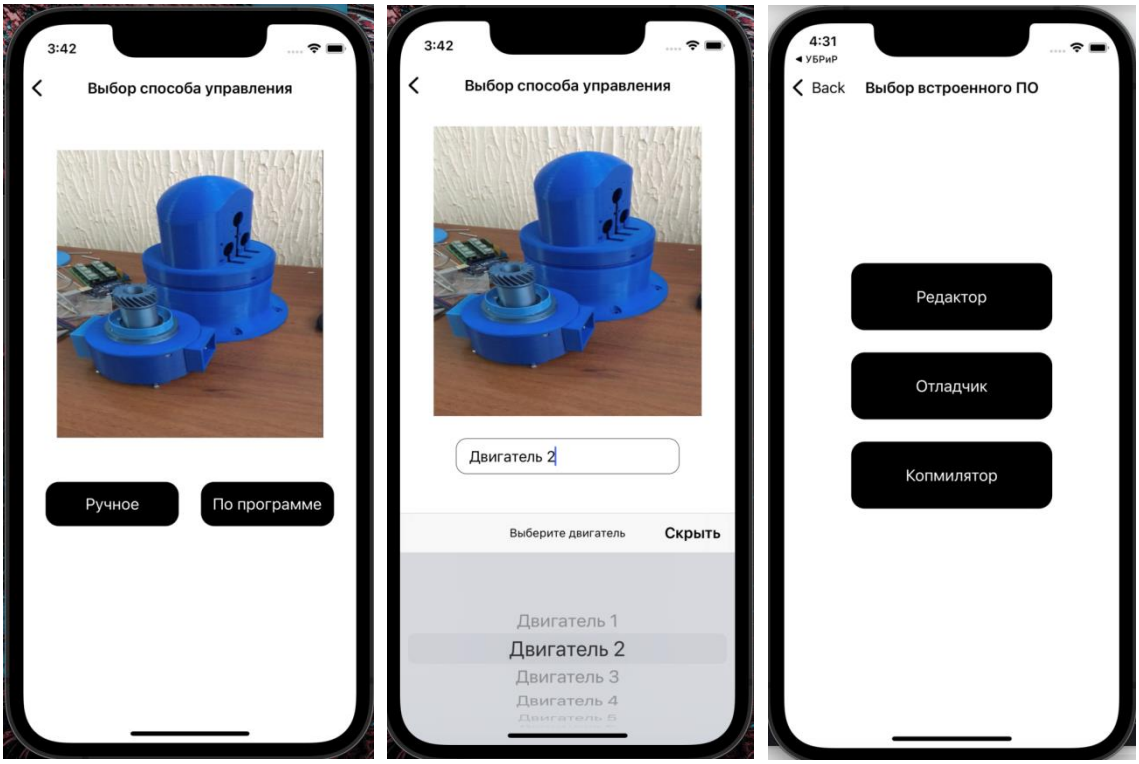

Рисунок 19 – Выбор режима тестирования и встроенного ПО

Для тестирования робота реализован ручной режим. При входе в этот режим имеется возможность выбора одного из семи двигателей робота и управление ими при помощи кнопок «ВПЕРЕД» и «НАЗАД».

При тестировании в ручном режиме на экран выводится изображение робота и/или двигателей получаемое от web-камеры (Рисунок 20).

Рисунок 20 – Управление двигателями в ручном режиме

2.4 Выводы по разделу 2

1. Рассмотрена общая архитектура и организация взаимодействия аппаратных средств. Предложен способ взаимодействия мобильного приложения с аппаратурой.
2. Рассмотрена организация сетевого взаимодействия. Это позволило перейти к разработке модулей взаимодействия пользователя с лабораторным оборудованием.
3. Разработана архитектура взаимодействия элементов тестового ПО.
4. Разработан общий алгоритм, функции и режимы тестирования.
5. Реализовано ПО для мобильного телефона позволяющее в демонстрационном режиме осуществлять взаимодействие с пользователем.
6. Полученные результаты позволяют перейти к разработке ПО и тестированию комплекса в лабораторных условиях.

3 Разработка мобильного приложения для организации удаленного доступа к лабораторному оборудованию

3.1 Разработка архитектуры мобильного приложения и создание программного кода

3.1.1 Архитектурный паттерн

На основании функционала разрабатываемого приложения был выбран оптимальный для его реализации архитектурный паттерн – MVC с дополнением в виде dataProvider. Вся работа по отображению визуального представления будет выполняться классами типа ViewController. Инициировать запросы к серверу и сохранять данные в виде моделей будут классы типа DataProvider.

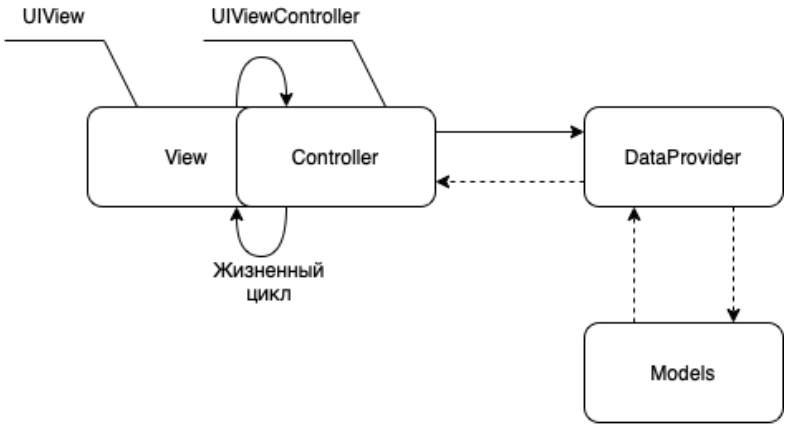

Рисунок 21 – Архитектурный паттерн MVC с DataProvider

3.1.1.1 Элемент DataProvider

Классы типа DataProvider общаются с сервером посредством Rest Api запросов. Для методов запроса в классе провайдера предусмотрены входные параметры в виде параметров запроса и замыканий из ViewController. В зависимости от типа замыкания в нем может передаваться статус запроса после его завершения или полученные данные.

Отдельные классы типа DataProvider могут сохранять полученные данные в userDefaults (настройки пользователя, которые остаются неизменными после завершения работы приложения). Примером этого может послужить AuthDataProvider. После успешной авторизации jwt-токен и данные пользователя сохраняются для передачи в качестве параметров в следующих api запросах.

3.1.1.2 Элемент ViewController

ViewController содержит все необходимые параметры для отображения информации и взаимодействия с пользователем. Он хранит в себе объекты UIView, добавляет их на основную view, определяет их положение и размер. Также ViewController определяет логику взаимодействия с пользователем - какое окно или сообщение будет отображено при определенном ответе сервера, какие действия назначаются на нажатие кнопки, когда и как должно обновиться визуальное представление класса.

3.2 Вспомогательные классы

3.2.1 HTTP-менеджер

Для реализации Rest api запросов был создан отдельный класс - HTTPManager. Именно к нему обращаются все классы DataProvider для взаимодействия с сервером. HTTPManager содержит весь необходимый набор инструментов для реализации приложения. Помимо адреса api и параметров запроса, в него передается метод запроса – «GET», «POST», «DELETE», «HEAD», «PUT».

Параметры запроса можно передавать как в виде Query, так и в виде Json с помощью встроенного метода преобразования из Dictionary. HTTManager поддерживает установку заголовков для запросов. Например, в приложении реализована авторизация по Bearer токену. В каждый запрос добавляется значение заголовка «Authorization» соответствующее значению параметра jwt, полученного после авторизации.

3.2.2 Модули расширения

Проект содержит большой набор расширений стандартных классов (Extension) для комфортного взаимодействия с ними. Класс String расширен методами для получения преобразованного значения из строки:

- fromBase64 (формирует изображение из строки типа Base64)
- toDate (конвертирует строку в дату с помощью паттерна)
- replaceMatches (заменяет найденную в строке последовательность символов на другую)

Стандартный класс UIViewController расширен методами для отображения диалоговых окон:

- dialog (Отображает диалоговое окно с переданными параметрами)
- errorRetryDialog (Отображает диалоговое окно с кнопками «Повторить», «Закрыть» и переданным сообщением)

Помимо этого, в класс UserDefaults были добавлены методы для удобного доступа к данным по их типу и ключу – get<T>(for Key:) и set<T>(for Key:).

3.2.3 CodableHelper

После того как DataProvider получил данные из Rest api запроса ему нужно их декодировать для передачи во ViewController или сохраниения в UserDefaults. За декодирование данных в модели отвечает класс CodableHelper. Методы decode и encode, содержащиеся в CodableHelper, преобразуют данные из типа Data в формате Json в переданный в параметре метода класс модели и обратно. В каждый из методов встроен dateFormatter для корректного перевода строк, приходящий в запросе в формат Date.

3.3 Модели данных

Для декодирования данных необходимо определить их тип. Каждому типу данных соответствует модель (codable структура). Все модели хранятся в отдельном каталоге Models с разбиением по уровням. Например, чтобы получить данные о сессии после подключения к ней придется декодировать данные типа Session, вложенные параметр session, приходящий как ответ на запрос к api.

Проект содержит все необходимые структуры для обработки ответов Rest api запросов, предоставленных backend разработкой, такие как: User, Session, Equipment, CropedUser, AuthData и другие.

3.4 Базовые классы

Для некоторых классов проекта предусмотрены базовые классы. Каждый из базовых классов содержит предопределённые методы и параметры для всех классов-наследников. Например, для объектов типов ViewController, TabBarViewController, View и Cell предусмотрены базовые классы с методами конфигурации – addViews (добавляет на экран визуальные элементы), bindViews (назначает действия для взаимодействия с пользователем), configureLayout (устанавливает положение визуальных элементов) и другие.

Так как многие объекты типа DataProvider самостоятельно декодируют классы, был создан базовый класс с предопределенным методом для декодирования данных из Data в установленный в Generic тип данных – BaseDecoder<T>.

Из-за схожести экранов отображения списка сессий было решено перенести часть их функционала в базовый класс BaseSessionsViewController. В нем происходят запросы к получению списка сессий с различными для

дочерних классов параметрами, удаление сессии и открытия экрана взаимодействия с сессией после подключения к ней.

3.5 Процесс разработки модулей

После инициализации всех необходимых файлов проекта, установки сторонних библиотек и написания базовых классов была начата разработка модулей приложения. Каждый модуль выноситься в отдельную директорию и содержит класс типа UIViewController, основанный на одном из базовых классов. Для класса контроллера инициализируются и настраиваются все дочерние классы визуального представления.

Помимо класса контроллера модуль содержит DataProvider, инициализированный содержащийся в объекте класса UIViewcontroller. В контроллере обозначаются приватные методы по взаимодействию с провайдером. Обработка ошибок апи запросов (презентация пользователю диалоговых окон) также происходит в классе контроллера.

Модуль может содержать дополнительные классы типа View, используемые в классе типа ViewController. Например, для класса SessionViewController создана директория «Subviews», содержащая файлы с описанием сессии, видеотрансляцией и элементами взаимодействия со стендом.

3.6 Трансляция видеоизображения

Для трансляции видеоизображения был создан класс StreamView. StreamView расположен на экране сессии (SessionViewController) вместе с SessionInfoView (отображает информацию о сессии) и SessionInteractionView (содержит набор элементов, необходимых для взаимодействия со стендом).

Вместе с получением информации о том, что пользователь может подключиться к сессии, в качестве ответа приходит её описание с параметром «streamUrl». Перед тем как показывать пользователю элементы управления и видео с камеры необходимо проверить связь с сервером - посылается запрос «equipment/check-availability-server». После его успешного выполнения в

WKWebView открывается ссылка с видео. Если запрос вернул ошибку – на StremView появляется сообщение об ошибке и кнопка «Повторить», вызывающая запрос снова. На рисунке 22 изображен пример сессии с трансляцией видео.

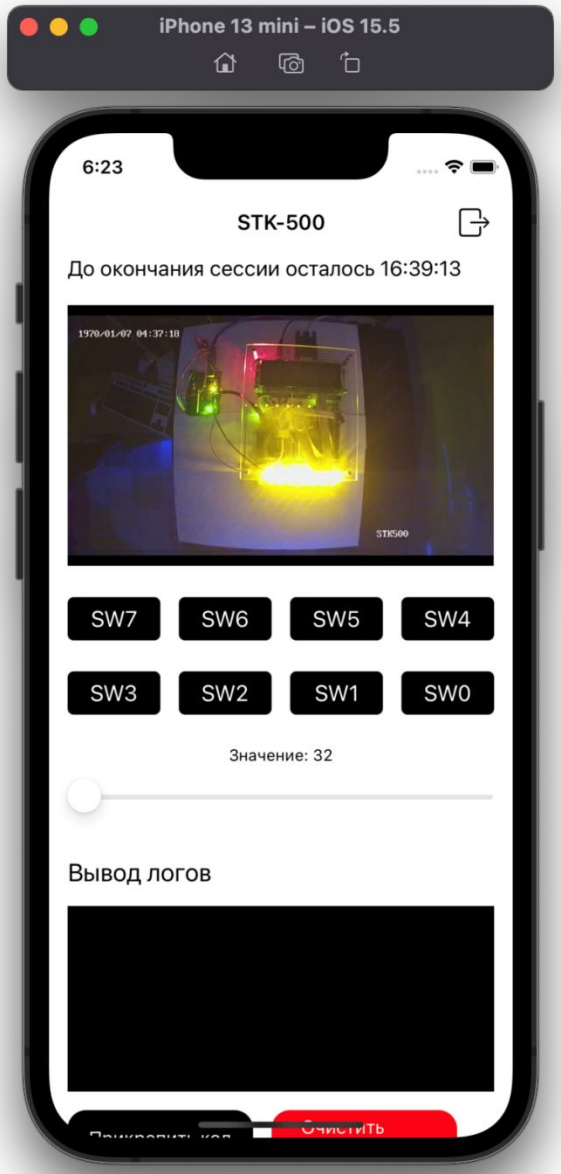

Рисунок 22 – Трансляция видео STK-500

3.7 Загрузка файла

Приложение поддерживает прошивку стенда файлами формата «*.hex». Для этого был использован стандартный класс DocumentPickerViewController. При нажатии кнопки «Прикрепить код» появляется всплывающее окно documentPicker, в котором отображаются все файлы устройства с возможностью выбора любого файла с расширением «hex». Далее файл передается на обработку класса FileUploader. По структуре запроса FileUploader схож с HTTPManager, но формирование тела запроса для передачи бинарного файла имеет ряд отличий, из-за которых было принято решение переноса кода в отдельный класс.

После запроса «equipment/send-file», обработанного классом FileUploader, пользователю выводиться сообщение в output окно о статусе выполненной операции. Такое сообщение в окне отображается и после любого взаимодействия с элементами управления или очистки памяти стенда.

Рисунок 23 – Выбор файла

3.8 Диаграммы классов

В качестве примера рассмотрим диаграмму классов модуля Session, приведенную на рисунке 24.

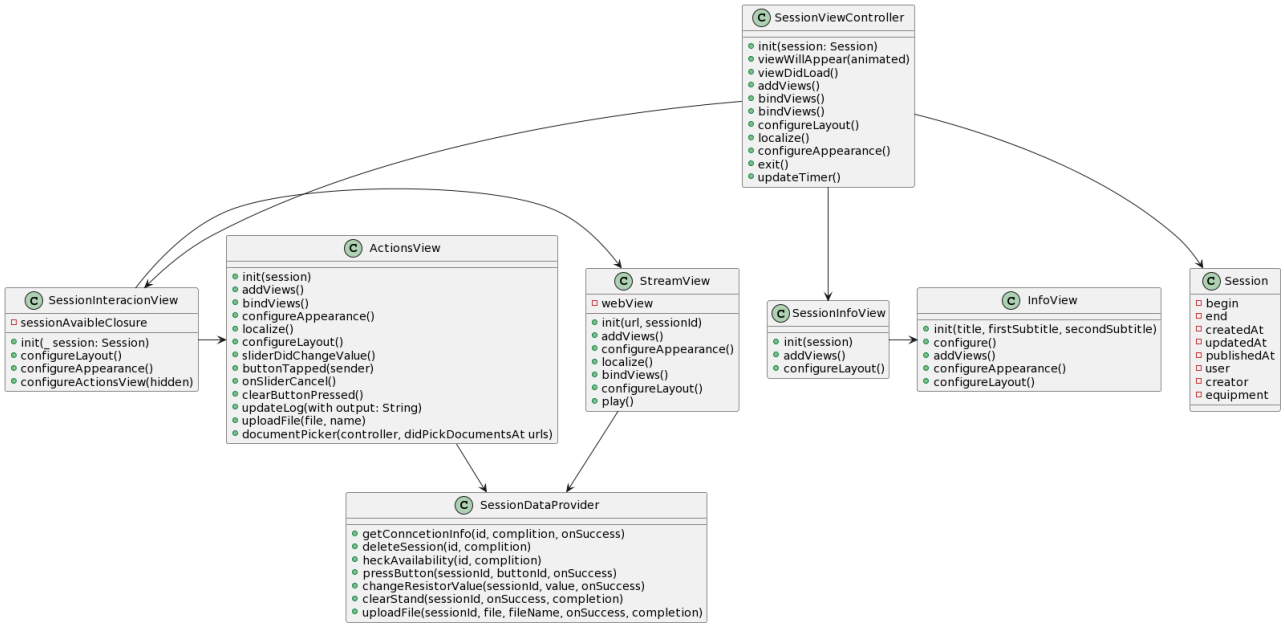

Рисунок 24 – Диаграмма класса SessionViewController

В состав диаграммы классов входят дочерние классы UIView, DataProvider и модели данных. Прочие диаграммы классов построены аналогичным образом. Их структура идентична диаграмме, приведенной на рисунке 24. Вышеприведенный рисунок иллюстрирует архитектурный подход к построению модулей. В связи с большим объемом диаграмм остальных классов, они приведены в приложении на электронном носителе.

3.9 Выводы по разделу 3

1. Переработан архитектурный паттерн для модулей приложения. Добавлены классы типа DataProvider для взаимодействия с api и сохранения данных.
2. В проект добавлены необходимые сторонние библиотеки и базовые классы для модулей. Стандартные классы расширены дополнительными методами.

3. Реализована связь с сервером посредством Rest Api запросов. В DataProvider добавлены методы для авторизации, создания и удаления сессии, методы взаимодействия с лабораторным оборудованием.

5. Добавлены модули для взаимодействия с сессиями.

6. Решена задача разработки мобильного приложения. Разработана архитектура, принципы взаимодействия программных компонентов и реализованы основные модули.

7. Дальнейшее усовершенствование мобильного приложение заключается в разработке требуемых модулей для различного дополнительного оборудования аналогично разработанному модулю для STK-500.

8. Полученные результаты позволили перейти к созданию модулей для STM-32 и DE1-SoC и тестированию ПО на лабораторном оборудовании.

4 Тестирование полученных технических решений

4.1Режим авторизации

Перед началом тестирования главного экрана приложения необходимо удостовериться в корректной работе процесса авторизации пользователя. Попадая на экран авторизации, пользователь вводит данные для входа – логин и пароль, затем нажимает кнопку «Войти» (Рисунок 25). На данный момент приложение работает в однопользовательском режиме и данные для авторизации подставляются в поля автоматически при входе на экран. После нажатия кнопки «Войти» происходит вызов запроса «auth/local», данные пользователя и токен, приходящие в ответе на запрос корректно обрабатываются и записываются во внутреннюю память устройства для последующей авторизации в автоматическом режиме. После этого пользователь попадает на экран списка сессий.

Рисунок 25 – Экран авторизации

4.2 Тестирование списка сессий

На каждом экране, содержащем список сессий, в правом углу навигационной панели расположена кнопка выхода из приложения. При ее нажатии пользователю выводится диалоговое окно «Вы действительно хотите выйти?». После нажатия пользователем кнопки «Да» происходит очищение данных пользователя и токена, хранящихся на устройстве, и перенаправление на экран авторизации (Рисунок 26).

Рисунок 26 – Навигационная панель

4.2.1 Управление сессиями

Первая вкладка главного экрана перенаправляет пользователя на экран управления сессиями. На нем пользователю представлен список сессий, которые он создавал. Список сессий является ответом на запрос «sessions/by-current-creator». При успешном статусе запроса происходит обновление экрана и корректное отображение полученных сессий. Если список сессий не пришел в ответе на запрос или ответ содержит статус отличный от 200, то пользователю выводиться диалоговое окно «Не удалось загрузить сессии» с возможностью повторения запроса. У каждой сессии отображается дата ее проведения, ФИО пользователя, которому назначена сессия и ФИО пользователя, создавшего сессию. От других экранов списка сессий его отличает наличие возможности удаления сессии, приведенной в списке, и кнопка создания сессии, расположенная внизу экрана. По нажатию на кнопку «Удалить» в ячейке таблицы происходит запрос «sessions» с методом «DELETE». Если ответ на запрос пришел с корректным статусом – 200, то экран обновляется и ячейка сессии пропадает из списка. В противном случае пользователю показывается

диалоговое окно об ошибке с сообщением «Не удалось удалить сессию» с возможностью повторить запрос. По нажатию на кнопку «Создать сессию» пользователь попадает на экран создания сессии, который показывается поверх экрана управления сессиями (Рисунок 27).

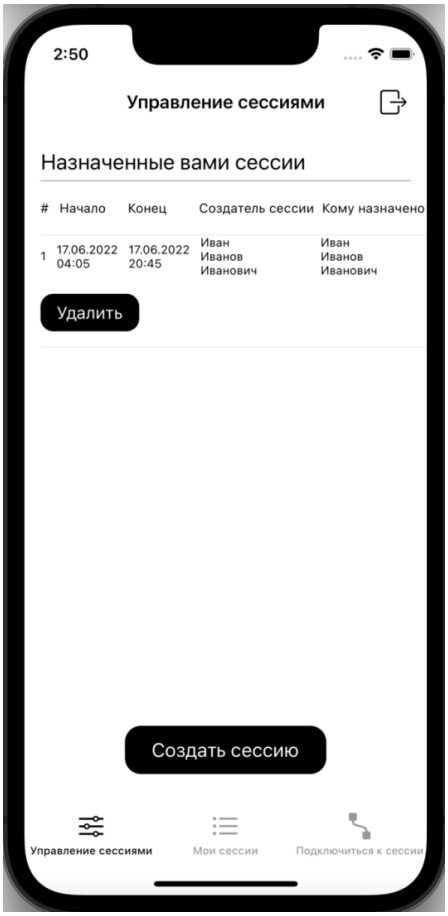

Рисунок 27 – Экран управления сессиями

4.2.2 Мои сессии

После авторизации пользователь попадает на экран, соответствующий средней вкладке главного экрана – «Мои сессии». На данном экране расположена таблица сессий, разделенная на две секции – «Начавшиеся сеансы» и «Сеансы, на которые вы записаны». При переходе на экран вызывается запрос «sessions/my-sessions». Если ответ на запрос содержит некорректный статус или список сессий, то подобно экрану управления сессиями пользователю выводиться диалоговое окно об ошибке. В ином случае список сессий фильтруется по дате и пользователю, на которого назначена

сессия. Если дата начала сеанса больше текущей, то ячейка с сессией отобразится в первой секции и у нее появляется кнопка подключения с наименованием оборудования, иначе ячейка перенесется во вторую секцию. По нажатию на кнопку подключения происходит запрос проверки возможности подключения пользователя к сессии - «sessions/\(id)/canConnect». Ответом на данный запрос является информация о сессии для экрана взаимодействия с сессией. При успешном ответе происходит переход на экран управления оборудованием, иначе показывается диалоговое окно с сообщением «Не удалось подключиться к сессии» и возможностью повторить запрос (Рисунок 28).

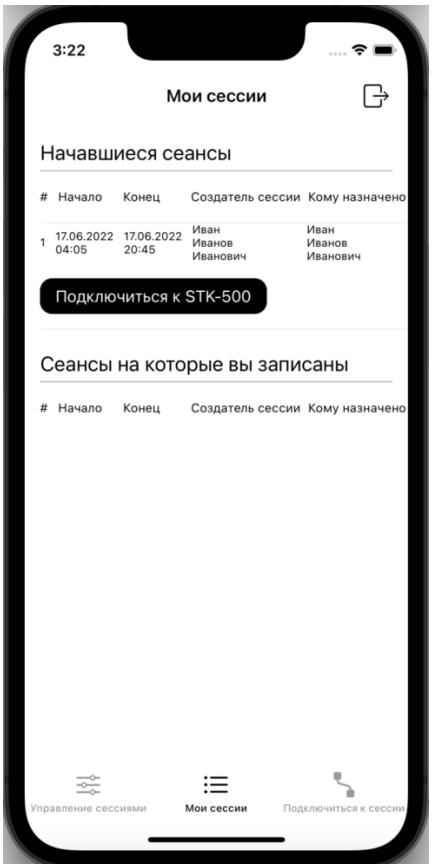

Рисунок 28 – Экран «Мои сессии»

4.2.2 Подключиться к сессии

Последняя вкладка главного экрана направляет пользователя на экран «Подключиться к сессии» (Рисунок 29). При его открытии происходит запрос

«sessions/ nearest-and-started». При успешном выполнении запроса на экран выводятся сессии, иначе пользователь видит диалоговое окно об ошибке, присутствующее на ранее описанных экранах списка сессий. Таблица экрана «Подключиться к сессии» имеет одну секцию – «Запушенные и ближайшие сеансы». При ее заполнении происходит фильтрация сессий пользователя по времени её начала. Оно должно быть не дольше чем через 10 минут от текущего времени. Каждая ячейка таблицы имеет кнопку подключения к сессии, имеющую функционал, аналогичный экрану «Мои сессии». Если по нажатию на кнопку подключения время начала сессии окажется больше текущего времени, то пользователю будет показано диалоговое окно об ошибке с сообщением «Эта сессия сейчас недоступна» и кнопкой «Закрыть».

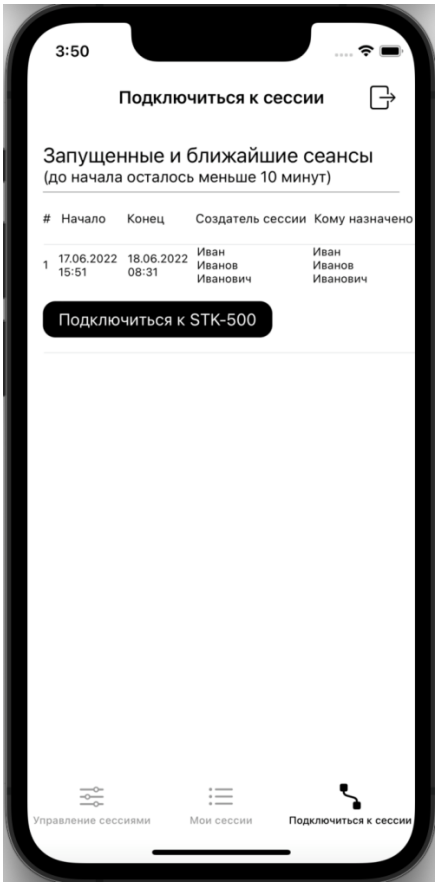

Рисунок 29 – Экран «Подключиться к сессии»

4.3 Создание сессии

На экране создания сессии пользователю предоставляется возможность выбора оборудования – «STK-500» (Выбрано по умолчанию), «STM-32» и «DE1-SoC». На данный момент можно создавать сессии для себя, в будущем справа от селектора «Записать себя» появиться селектор «Записать студента» (Рисунок 30). Ниже располагается выбор даты и время начала сессии с указанием ее длительности в минутах в отдельном поле. Внизу экрана расположена кнопка «Создать сессию». После ее нажатия происходит проверка заполненных данных. Если пользователь ввел дату, не большую текущей, то на экран будет выведено окно с сообщением «Выберите более позднюю дату». Также пользователь не может создать сессию на время, на которое назначена другая сессия – на экране появится диалоговое окно с сообщением «На это время назначена другая сессия». После этого проверяется содержимое поля продолжительности сессии. Если пользователь не ввел это значение, то отобразиться окно с сообщением «Необходимо указать длительность сеанса». При корректном заполнении всех полей после нажатия кнопки создания сессии будет вызван запрос «sessions» с передаваемыми параметрами в виде даты начала и окончания сессии и данных пользователя. При его успешном выполнении экран создания сессии будет закрыт и произойдет перенаправление на экран управления сессиями с последующим его обновлением. Если же статус ответа не равен «200», то пользователю будет показано диалоговое окно с сообщением «Не удалось создать сессию» и возможностью повторить запрос.

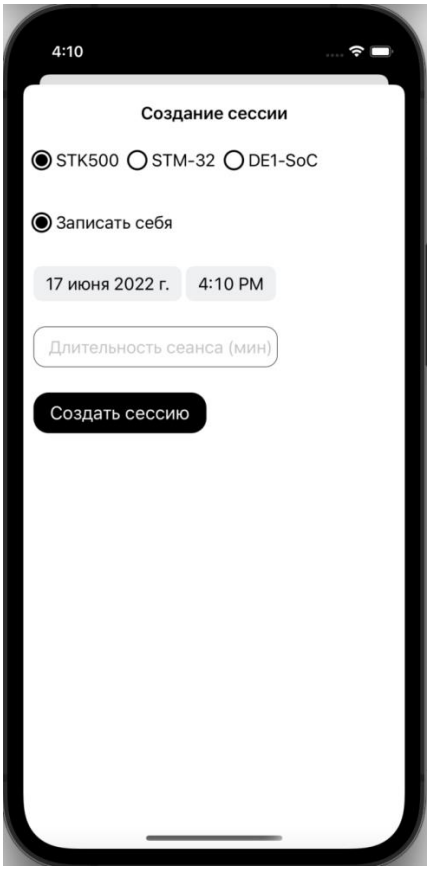

Рисунок 30 – Экран «Создание сессии»

4.4 Экран взаимодействия с оборудованием

После нажатия на кнопку подключения и успешного выполнения запроса «sessions/\(id)/canConnect» пользователь попадает на экран взаимодействия с оборудованием (Рисунок 31). Вверху экрана отображается таймер, показывающий время до окончания сессии. Этот экран разделен на несколько работающих независимо друг от друга модулей. Далее описано тестирование каждого из них. Набор модулей на экране зависит от типа оборудования. При открытии экрана происходит вызов запроса «equipment/check-availability-server/session». Он необходим для проверки доступности оборудования. После успешного ответа на запрос происходит добавление модулей взаимодействия с оборудованием на экран и начало трансляции видеоизображения. Если статус ответа отличается от «200», то на месте видеотрансляции появляется надпись «Не удалось подключиться» и кнопка «Повторить», инициирующая повторный вызов запроса.

Рисунок 31 – Пример экрана взаимодействия для STK-500

4.4.1 Тестирование видео

Первым и общим для всех типов оборудования является модуль трансляции видеоизображения. Ссылка для трансляции берется из поля «stream_url» ответа на запрос «sessions/\(id)/canConnect». После успешной проверки доступности оборудования модуль видеотрансляции получает ссылку и начинает загрузку страницы с потоковым видео (Рисунок 32).

Рисунок 32 – Пример трансляции видео для DE1-SoC

4.4.2 Тестирование элементов управления

Перечень модулей элементов управления оборудованием будет зависеть от типа оборудования, полученного из поля «equipment» ответа на запрос «sessions/\(id)/canConnect». Для «STK-500» будут отображены 8 кнопок, для «STM-32» - одна кнопка, для «DE1-SoC» - 8 переключателей и 4 кнопки. Для «STK-500» и «DE1-SoC» вызываются запросы соответственно «equipment/send-command/session/\(sessionId)?command=/resistor/» и «equipment/send-command/session/\(sessionId)?command=/status-switches/». Они необходимы для корректного отображения начального положения слайдера аналогового ввода и переключателей. Для слайдера ответом будет являться его текущее в диапазоне от 32 до 4095, для переключателей – строка типа «00110101», где 1 и 0 соответствую включенному и выключенному состояниям. После успешного ответа, соответствующий ему модуль обновляется и на экран выводится текущее значение.

4.4.2.1 Тестирование кнопок

Для каждого из типов оборудования на экране будет представлено фиксированное различное кнопок для взаимодействия. По нажатию и прекращению нажатия на каждую из них вызывается запрос «session/\(sessionId)?command=/button/\(buttonId)» с указанием номера сессии и индекса нажатой кнопки. Тип оборудования определяется автоматически по номеру сессии. Ответом на запрос, вызываемый после нажатия кнопки, будет строка, которая выводиться в окно «output», расположенное ниже элементов управления (Рисунок 33).

Рисунок 33 – Вид кнопок для STK-500

4.4.2.2 Тестирование переключателей

Если за сессией закреплено оборудование «DE1-SoC», то на экране выше кнопок будет отображено 8 переключателей (Рисунок 34). Их начальное состояние задается при открытии экрана. После нажатия на переключатель происходит вызов «equipment/send-command/session/\(sessionId)?command=/switch/\(switchId)». Переданное в нем восьми символьное значение в виде строки из 1 и 0 устанавливается в модуль переключателей. Помимо этого, в модуле «output» отображается сообщение об успешном нажатии переключателя с определенным индексом.

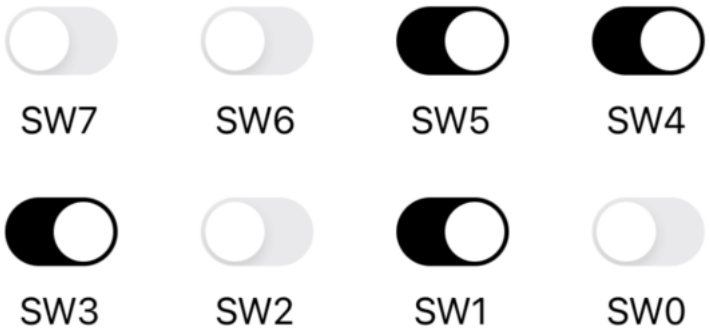

Рисунок 34 – Пример отображения переключателей для «DE1-SoC»

4.4.2.3 Тестирование аналогового ввода

Для сессии с типом оборудования «STK-500» ниже модуля кнопок отобразиться модуль аналогового ввода. На нем представлен заголовок, отображающий текущее значение, и слайдер, позволяющий его изменять (Рисунок 35). После того как пользователь изменил положение слайдера и отпустил его, тем самым закончив взаимодействие с ним, будет вызван запрос «equipment/send-command/session/\(sessionId)?command=/resistor/\(value)».
Ответом на этот запрос будет строка, выводящаяся в окно «output».

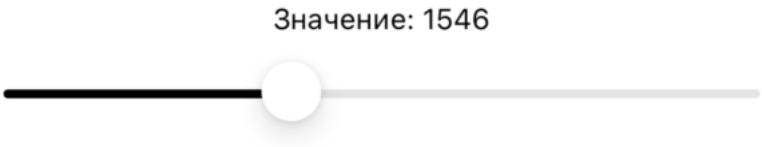

Рисунок 35 – Пример отображения слайдера для «STK-500»

4.4.3 Тестирование сообщений

Как было сказано выше, успешное взаимодействие с каждым из элементов управление инициирует вывод сообщения в окно «output». Данные выводятся построчно и пользователю предоставляется возможность пролистывать окно вывода сообщений для отображения ранее выведенных строк (Рисунок 36).

Рисунок 36 – Пример отображения окна сообщений

4.4.4 Тестирование режима программирования оборудования

Для всех типов стендов предусмотрена возможность прошивки оборудования файлами, находящимися на устройстве. После нажатия кнопки «Прикрепить код», расположенной ниже окна вывода сообщений, происходит открытие окна выбора файла (Рисунок 37). Для каждого типа оборудования обозначен заранее определенный тип файла, что исключает загрузку в

оборудование непредназначенного для него кода. Для «STK-500» - «*.hex», для «STM-32» - «*.bin», для «DE1-SoC» - «*.sof». Экран выбора файла настроен на открытие файлов только установленного типа (Рисунок 38). После выбора файла вызывается запрос «equipment/send-file/session/\(sessionId)?command=/upload». Предусмотрена обработка ошибок загрузки файла. При их возникновении пользователю будет выведено диалоговое окно с сообщением «Не удалось загрузить файл» и возможностью повторить запрос. Если в ответе на запрос пришел статус успешного выполнения, то в окно «output» будет выведено сообщение о прошивке оборудования.

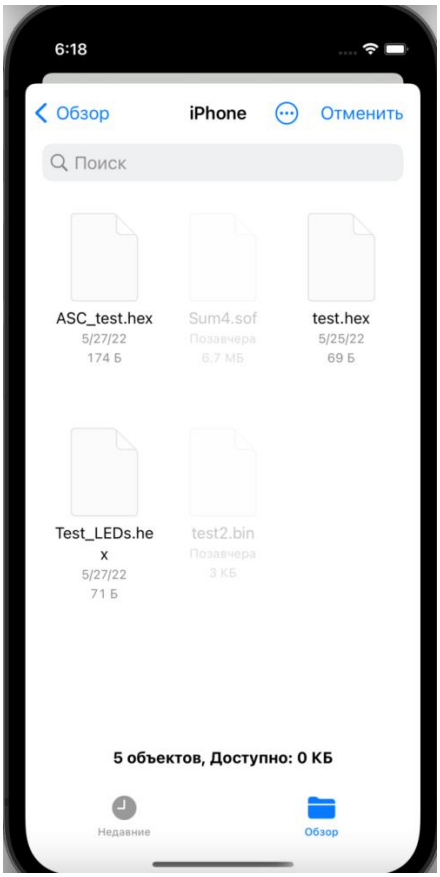

Рисунок 37 – Окно выбора файла для «STK-500»

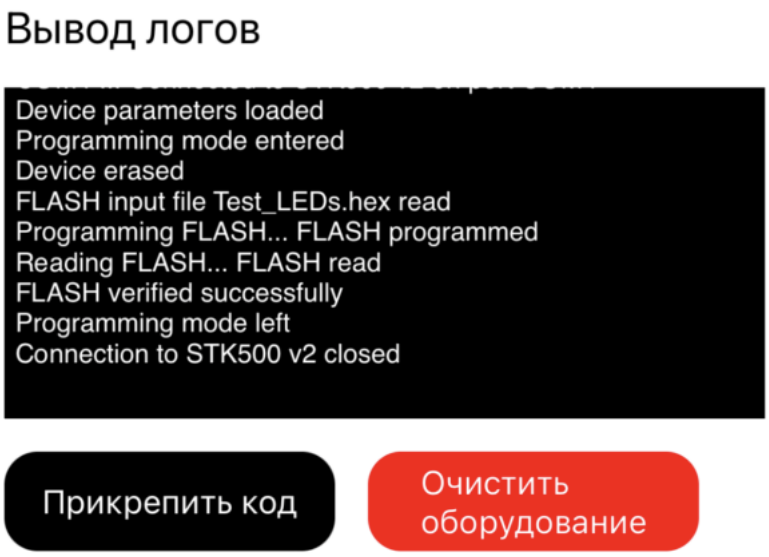

Рисунок 38 – Пример прошивки «STK-500»

4.4.5 Тестирование очистки оборудования

Помимо прикрепления кода в приложении присутствует возможность очисти каждого типа стенда. При нажатии кнопки «Очистить оборудование», расположенной справа от кнопки прикрепления кода происходил вызов запроса «equipment/send-command/session/\(sessionId)/?command=/clean» (Рисунок 39). При его успешном выполнении пользователь увидит сообщение в окне «output», иначе на экран будет выведено диалоговое окно с сообщением «Не удалось очистить оборудование» и возможностью повторения запроса.

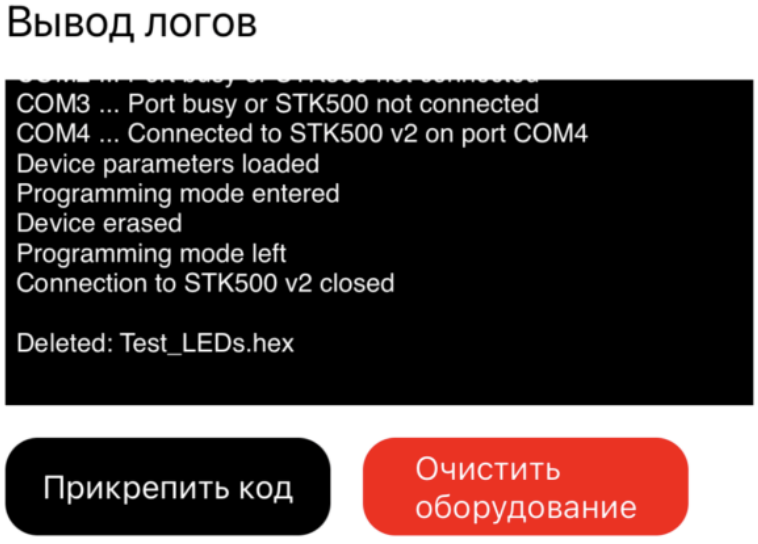

Рисунок 39 – Пример очистки «STK-500»

4.4.6 Отображение дополнительной информации

Внизу экрана взаимодействия с сессией расположен модуль информации о текущей сессии. Он обновляется при открытии экрана и отображает данные из запроса «sessions/\(id)/canConnect» (Рисунок 40).

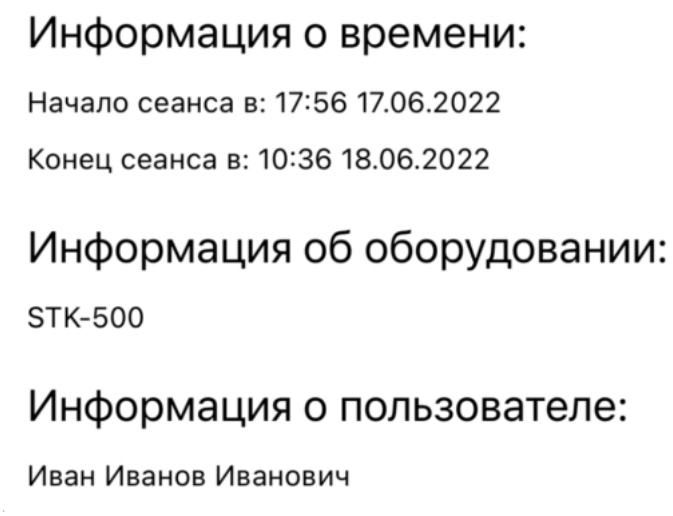

Рисунок 40 – Пример отображения информации о сессии для «STK-500»

4.7 Выводы по разделу 4

Выполнено тестирование пользовательского ПО для лабораторных стендов STK-500, STM-32 и DE1 SoC, при этом:

1. Выполнено тестирование основных диалоговых режимов: авторизации, списка сессий, создания, удаления и подключения к сессии.

2. Выполнено тестирование основных функциональных возможностей: трансляции потокового видео в режиме реального времени, функционирование элементов управления и выдачи сообщений, режима программирования, очистки оборудования и вывода дополнительной информации.

3. При тестировании определено, что разработанное ПО пользователя функционирует штатно в режиме лабораторных испытаний и позволяет выполнять тестирование режимов доступа к оборудованию в учебном процессе.

4. Несмотря на штатное функционирования ПО при доступе к платам обнаружилась значительная задержка серверного оборудования (приблизительно до 3-х секунд) при формировании отклика на подачу входного сигнала и аналогичная задержка трансляции видео.

5. Путем подбора параметров сети удалось снизить задержу отклика приблизительно до 1 секунды и исключить задержку при трансляции видео. Это позволяет выполнять довольно комфортный доступ к платам, но требует дальнейшей проработки.

6. При длительном прогоне (до 2-х недель) появляется «плавающая» ошибка – сбой управляющей платы Arduino на STK500. После перезагрузки плата входит в штатный режим. Требуется накопление и анализ случайных сбоев и возможная замена платы управления на стенде STK 500.

7. При трансляции видео изображение, 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 жидкокристаллических дисплеев.

8. 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. Также на основании анализа программных инструментов для реализации выбран программный пакет AnyDesk, как обладающий наибольшими функциональными возможностями среди прочих аналогов. На втором этапе, при создании ПО для удаленного тестирования рассмотрена разработанная общая архитектура и организация сетевого взаимодействия аппаратных средств, а также предложен способ взаимодействия мобильного приложения с аппаратурой. Это позволило перейти к разработке модулей взаимодействия пользователя с лабораторным оборудованием в режиме симуляции. При этом разработана архитектура взаимодействия элементов тестового ПО, общие алгоритмы, функции и режимы тестирования. На этом этапе было реализовано ПО для мобильной операционной системы IOS, предназначенное для использования при дальнейшей разработке, например подключения новых устройств к системе (станка с ЧПУ или стационарного робота). Это ПО предназначено для разработчика и позволяет с минимальными изменениями исходного кода тестировать сетевые соединения и визуально наблюдать за функционированием новых устройств при взаимодействии с пользователем. Полученные решения позволили перейти к разработке ПО пользователя и тестированию комплекса в лабораторных условиях. На третьем этапе была разработана архитектура, принципы взаимодействия программных компонентов и реализованы основные модули. Переработан архитектурный паттерн для модулей приложения, добавлены методы, классы и необходимые сторонние библиотеки. При этом стандартные классы расширены дополнительными методами. При разработке

исходных кодов реализована связь с сервером, методы для авторизации, создания и удаления сессии, методы взаимодействия с лабораторным оборудованием и модули для взаимодействия с сессиями. Полученные результаты третьего этапа позволили перейти к тестированию ПО на лабораторном оборудовании STK500, STM-32 и DE1-SoC. На заключительном этапе работ выполнено тестирование пользовательского ПО на базе созданного рабочей группой лабораторного комплекса доступа к оборудованию. Выполнялось тестирование всех основных режимов: авторизации, сессий, трансляции потокового видео, элементов управления, выдачи сообщений, режимов программирования, очистки оборудования и других. Результаты тестирования показали нормальное функционирование разработанного ПО во всех режимах, определенных заданием на ВКР. Тем не менее, тестирование показало некоторые недочеты в работе оборудования, а именно: задержку трансляции и отклика, случайные сбои в работе управляющей платы и 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 ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Файловый архив студентов StudFiles [Электронный ресурс] / ГМУ им. Адмирала Ф. Ф. Ушакова — Электрон. дан., 2015. — Режим доступа: https://studfile.net/preview/1665419/page:14/ — Загл. с экрана.
2. Блог компании СТЕК [Электронный ресурс] — Электрон. дан., 2018. — Режим доступа: https://stekspb.ru/blog/remote-desktop/ — Загл. с экрана.
3. СОФТЛИСТ. ТОП-4 программ удаленного доступа к компьютеру в 2021 году, [Электронный ресурс] — Электрон. дан., 2021. — Режим доступа: https://softlist.com.ua/articles/top-5-programm-udalennogo-dost/ — Загл. с экрана.
4. Т. Лэммл, Ш. Одом, Р. Педжен. CCNP. Удаленный доступ. Учебное руководство. Лори, 2018. 412С.
5. IT`s Support. Лучшие программы для удалённого доступа, [Электронный ресурс] — Электрон. дан., 2020. — Режим доступа: https://itssupport.ru/blog/luchshie-programmyi-dlya-udalyonnogo-dostupa.html — Загл. с экрана.
6. ASTER. Для чего нужна программа Anydesk, [Электронный ресурс] — Электрон. дан., 2021. — Режим доступа: https://www.ibik.ru/ru/what-purpose-anydesk/ — Загл. с экрана.
7. НИУ ВШЭ. Удаленный доступ к оборудованию УЛ САПР, [Электронный ресурс] — Электрон. дан., 2021. — Режим доступа: https://miem.hse.ru/edu/ce/cadsystem/remote_access — Загл. с экрана.
8. Лаборатория Электронных Средств Обучения (ЛЭСО) СибГУТИ, [Электронный ресурс] — Электрон. дан., 2020. — Режим доступа: http://www.labfor.ru/articles/education/philosophy — Загл. с экрана.
9. Oriel A. HerreraGustavo R. AlvesDavid FullerRoberto G. Aldunate. Remote Lab Experiments: Opening Possibilities for Distance Learning in Engineering Fields. IFIP World Computer Congress, TC 3 IFIP WCC TC3 2006: Education for the 21st Century — Impact of ICT and Digital Resources p.p. 321-325.

10. The Hong Kong Polytechnic University. Department of Applied Physics. Remote Lab, [Электронный ресурс] — Электрон. дан., 2021. — Режим доступа: https://remotelab.ap.polyu.edu.hk/ — Загл. с экрана.

11. Javier García-Zubía. Remote Laboratories. Empowering STEM Education with Technology. Remote Laboratories, pp. i-xxiii (2021) p.268.

12. Северо-Западный межвузовский региональный учебно-научный центр "СПбПУ - ФЕСТО", [Электронный ресурс] — Электрон. дан., 2019. — Режим доступа: https://www.spbstu.ru/structure/educational-scientific-center-spbpu-festo/ — Загл. с экрана.

13. Ильдухина Н.В., Гордеев Д.Ю., Замалетдинов А.Ф., Старыгина С.Д. ОБЗОР СОВРЕМЕННЫХ СРЕДСТВ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ // Современные наукоемкие технологии. – 2019. – № 4. – С. 22-26;

14. Усов Василий. Swift. Основы разработки приложений под iOS, iPadOS и macOS. // Питер, серия Библиотека программиста. – 2020. – С.496.

15. Марк, Топли, Маскри - Swift 3. Разработка приложений в среде Xcode для iPhone и iPad с использованием iOS SDK. // Вильямс. – 2019. – . 896С.

16. Apple. Swift. Язык программирования с открытым кодом. Мощь, простота и потрясающие приложения", [Электронный ресурс] — Электрон. дан., 2021. — Режим доступа: https://www.apple.com/ru/swift/ — Загл. с экрана.

ПРИЛОЖЕНИЕ А

Примеры исходных кодов программ[1].

TaskSelectionViewController

```
import UIKit
import AVFoundation

class TaskSelectionViewController: UIViewController {

  let videoUrl: URL
  let standTranslationView = VideoView()
  let startButton = UIButton()
  let taskField = PaddingField()
  let taskOptionStack = UIStackView()
  let thePicker = UIPickerView()
  let tasks: [String] = [
    "Задание 1",
    "Задание 2",
    "Задание 3",
    "Задание 4"
  ]

  init(videoUrl: URL) {
    self.videoUrl = videoUrl
    super.init(nibName: nil, bundle: nil)
  }
```

[1] *Полный текст исходного кода приведен на электронном носителе.*

```
required init?(coder: NSCoder) {
  fatalError("init(coder:) has not been implemented")
}
override func viewDidLoad() {
  super.viewDidLoad()

  addViews()
  localize()
  configureLayout()
  configureAppaerance()
}

override func viewDidAppear(_ animated: Bool) {
  standTranslationView.configure(videoURL: videoUrl)
  standTranslationView.isLoop = true
  standTranslationView.play()
}

private func configureLayout() {
  standTranslationView.backgroundColor = .black

  standTranslationView.snp.makeConstraints {
    $0.width.equalToSuperview().multipliedBy(0.8)
    $0.height.equalToSuperview().multipliedBy(0.4)
    $0.centerY.equalToSuperview().multipliedBy(0.75)
    $0.centerX.equalToSuperview()
  }

  startButton.snp.makeConstraints {
    $0.height.equalTo(50)
```

```
    $0.width.equalTo(150)
  }

  taskOptionStack.snp.makeConstraints {
    $0.centerX.equalToSuperview()
    $0.top.equalTo(standTranslationView.snp.bottom).offset(50)
  }

  taskOptionStack.distribution = .fillEqually
  taskOptionStack.spacing = 25
}

private func localize() {
  navigationItem.title = "Выбор задания"
  startButton.setTitle("Старт", for: .normal)
  taskField.placeholder = "Задание"
}

private func addViews() {
  view.addSubview(standTranslationView)
  taskOptionStack.addArrangedSubview(taskField)
  taskOptionStack.addArrangedSubview(startButton)
  view.addSubview(taskOptionStack)
  taskField.inputView = thePicker
  thePicker.delegate = self
}

private func configureAppaerance() {
  view.backgroundColor = .white

  startButton.backgroundColor = .black
```

```
    startButton.setTitleColor(.white, for: .normal)
    startButton.layer.cornerRadius = 15

    taskField.layer.cornerRadius = 12
    taskField.layer.borderWidth = 0.5
    taskField.layer.borderColor = UIColor.black.cgColor
  }
}

extension TaskSelectionViewController: UIPickerViewDelegate,
UIPickerViewDataSource {
  func numberOfComponents(in pickerView: UIPickerView) -> Int {
    1
  }

  func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent
component: Int) -> Int {
    tasks.count
  }

  func pickerView( _ pickerView: UIPickerView, titleForRow row: Int,
forComponent component: Int) -> String? {
   return tasks[row]
  }

  func pickerView( _ pickerView: UIPickerView, didSelectRow row: Int,
inComponent component: Int) {
      taskField.text = tasks[row]
  }
}
```

```
class VideoView: UIView {

  var playerLayer: AVPlayerLayer?
  var player: AVPlayer?
  var isLoop: Bool = false

  func configure(videoURL: URL) {
    player = AVPlayer(url: videoURL)
    playerLayer = AVPlayerLayer(player: player)
    playerLayer?.frame = bounds
    playerLayer?.videoGravity = AVLayerVideoGravity.resize
    if let playerLayer = self.playerLayer {
      layer.addSublayer(playerLayer)
    }
    NotificationCenter.default.addObserver(self, selector:
#selector(reachTheEndOfTheVideo(_:)), name:
NSNotification.Name.AVPlayerItemDidPlayToEndTime, object:
self.player?.currentItem)
  }

  func play() {
    if player?.timeControlStatus != AVPlayer.TimeControlStatus.playing {
      player?.play()
    }
  }

  func pause() {
    player?.pause()
  }

  func stop() {
```

```
    player?.pause()
    player?.seek(to: CMTime.zero)
  }

  @objc func reachTheEndOfTheVideo(_ notification: Notification) {
    if isLoop {
      player?.pause()
      player?.seek(to: CMTime.zero)
      player?.play()
    }
  }
}
```

StandSelectionViewController

```
import UIKit

class StandSelectionViewController: UIViewController {

  struct Stand {
    let image: UIImage
    let name: String
  }

  let layout = UICollectionViewFlowLayout()
  let stands: [Stand] = [
    .init(image: .init(named: "STK500")!,
          name: "STK500"),
    .init(image: .init(named: "ROBOT")!,
          name: "ROBOT"),
    .init(image: .init(named: "TERASIC")!,
```

```
        name: "TERASIC"),
    .init(image: .init(named: "STM")!,
        name: "STM")
  ]

  lazy var standsCollectionView: UICollectionView = {
    let layout = UICollectionViewFlowLayout()
    layout.itemSize = .init(width: view.frame.width * 0.4, height: view.frame.width *
0.52)
    layout.sectionInset = .init(top: 0, left: layout.itemSize.width/8, bottom: 0, right:
layout.itemSize.width/8)
    layout.minimumLineSpacing = layout.itemSize.width/4
    let frame = view.frame
    return UICollectionView(frame: .init(origin: .init(x: frame.origin.x,
                                        y: view.frame.height/4),
                                size: frame.size),
                        collectionViewLayout: layout)
  }()

  override func viewDidLoad() {
    super.viewDidLoad()

    setupStandsCollection()
    addViews()
    configureAppearance()
    localize()
  }

  private func addViews() {

    view.addSubview(standsCollectionView)
```

```
  }

  private func localize() {
    navigationItem.title = "Выбор стенда"
  }

  private func configureAppearance() {
    view.backgroundColor = .white
    navigationController?.navigationBar.tintColor = .black
  }
}

extension StandSelectionViewController: UICollectionViewDelegate,
UICollectionViewDataSource {

  func collectionView(_ collectionView: UICollectionView,
numberOfItemsInSection section: Int) -> Int {
    4
  }

  func collectionView(_ collectionView: UICollectionView, cellForItemAt
indexPath: IndexPath) -> UICollectionViewCell {
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier:
"StandCollectionCell", for: indexPath) as! StandCollectionCell

    cell.standImageView.image = stands[indexPath.row].image
    cell.titleLabel.text = stands[indexPath.row].name

    return cell
  }
```

```
  func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let name = stands[indexPath.row].name
    if name != "ROBOT",
      let path = Bundle.main.path(forResource: name, ofType:"mp4"){

navigationController?.pushViewController(TaskSelectionViewController(videoUrl:
URL(fileURLWithPath: path)), animated: true)
    } else {

    }
  }

  func setupStandsCollection() {
    standsCollectionView.delegate = self
    standsCollectionView.dataSource = self
    standsCollectionView.register(StandCollectionCell.self,
forCellWithReuseIdentifier: "StandCollectionCell")
  }
}
```

AuthViewController

```
import UIKit
class AuthViewController: UIViewController  {
  let loginField = PaddingField()
  let passwordField = PaddingField()
  let enterButton = UIButton()
  let loginFieldsStack = UIStackView()
  override func viewDidLoad() {
    super.viewDidLoad()
    setupFileds([loginField, passwordField])
```

```
    addViews()
    configureAppearance()
    configureLayout()
    localize()
    bindViews()
  }

  private func bindViews() {
    enterButton.addTarget(self, action: #selector(didTapEnterButton), for:
.touchUpInside)
  }
  private func localize() {
    navigationItem.title = "Авторизация"
    loginField.placeholder = "Логин"
    passwordField.placeholder = "Пароль"
    enterButton.setTitle("Войти", for: .normal)
  }

  private func configureLayout() {
    loginFieldsStack.snp.makeConstraints {
      $0.centerX.centerY.equalToSuperview()
    }

    enterButton.snp.makeConstraints {
      $0.width.equalTo(250)
      $0.height.equalTo(50)
      $0.centerX.equalToSuperview()
      $0.top.equalTo(loginFieldsStack.snp.bottom).offset(30)
    }

    loginFieldsStack.spacing = 15
```

```
  loginFieldsStack.axis = .vertical
}

private func addViews() {
  loginFieldsStack.addArrangedSubview(loginField)
  loginFieldsStack.addArrangedSubview(passwordField)
  view.addSubview(loginFieldsStack)
  view.addSubview(enterButton)
}

private func configureAppearance() {
  view.backgroundColor = .white
  enterButton.backgroundColor = .black
  enterButton.setTitleColor(.white, for: .normal)
  enterButton.layer.cornerRadius = 15
}

private func setupFileds(_ fields: [UITextField]) {
  fields.forEach { field in
    field.clearButtonMode = .whileEditing
    field.layer.cornerRadius = 12
    field.layer.borderWidth = 0.5
    field.layer.borderColor = UIColor.black.cgColor
    field.snp.makeConstraints {
      $0.height.equalTo(40)
      $0.width.equalTo(250)
    }
  }
}

@objc func didTapEnterButton() {
```

```
    navigationController?.pushViewController(StandSelectionViewController(),
animated: true)
  }
}

class PaddingField: UITextField {
    let padding = UIEdgeInsets(top: 0, left: 15, bottom: 0, right: 5)
    override open func textRect(forBounds bounds: CGRect) -> CGRect {
        return bounds.inset(by: padding)
    }

    override open func placeholderRect(forBounds bounds: CGRect) -> CGRect {
        return bounds.inset(by: padding)
    }

    override open func editingRect(forBounds bounds: CGRect) -> CGRect {
        return bounds.inset(by: padding)
    }
}
```

StandCollectionCell

```
import UIKit
class StandCollectionCell: UICollectionViewCell {
  let standImageView = UIImageView()
  let titleLabel = UILabel()
  override init(frame: CGRect) {
    super.init(frame: frame)
    contentView.addSubview(standImageView)
    addSubview(titleLabel)
    titleLabel.textAlignment = .center
    titleLabel.snp.makeConstraints {
```

```
      $0.height.equalTo(frame.height/8)
      $0.width.equalToSuperview().multipliedBy(0.75)
      $0.centerX.bottom.equalToSuperview()
    }
    standImageView.snp.makeConstraints {
      $0.top.left.right.equalToSuperview()
      $0.bottom.equalTo(titleLabel.snp.top)
    }
  }

  required init?(coder: NSCoder) {
    fatalError("init(coder:) has not been implemented")
  }
}
```

HomeViewController

```
import UIKit
import SnapKit
class HomeViewController: UIViewController {
  let buttonRight = UIButton()
  let buttonLeft = UIButton()
  let buttonsStack = UIStackView()
  override func viewDidLoad() {
    super.viewDidLoad()
    navigationController?.setNavigationBarHidden(true, animated: true)
    buttonLeft.addTarget(self, action: #selector(didTapLeftButton), for:
.touchUpInside)
    buttonRight.addTarget(self, action: #selector(didTapRightButton), for:
.touchUpInside)
    buttonLeft.setImage(.init(systemName: "arrow.left"), for: .normal)
    buttonRight.setImage(.init(systemName: "arrow.right"), for: .normal)
```

```
    buttonLeft.tintColor = .black
    buttonLeft.contentVerticalAlignment = .fill
    buttonLeft.contentHorizontalAlignment = .fill

    buttonRight.tintColor = .black
    buttonRight.contentVerticalAlignment = .fill
    buttonRight.contentHorizontalAlignment = .fill
    buttonsStack.addArrangedSubview(buttonLeft)
    buttonsStack.addArrangedSubview(buttonRight)
    buttonsStack.spacing = 50
    view.addSubview(buttonsStack)
    view.backgroundColor = .white
    buttonsStack.snp.makeConstraints {
      $0.centerX.equalToSuperview()
      $0.bottom.equalToSuperview().inset(100)
    }
       buttonLeft.snp.makeConstraints {
      $0.height.equalTo(50)
      $0.width.equalTo(70)
    }
       buttonRight.snp.makeConstraints {
      $0.height.equalTo(50)
      $0.width.equalTo(70)
    }
  }
    @objc func didTapLeftButton() {
  }
  @objc func didTapRightButton() {
  }
}
```

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

«Мобильное приложение удаленного доступа к лабораторному оборудованию для IOS»
Тема

09.03.01 «Информатика и вычислительная техника»
код и наименование направления

Руководитель	22.06 подпись, дата	ст. пр-ль каф. ВТ ИКИТ должность, ученая степень	И.В. Матковский инициалы,фамилия
Выпускник	22.06 подпись, дата		Д.О. Непомнящий инициалы,фамилия
Нормоконтролер	22.06 подпись, дата	ст. пр-ль каф. ВТ ИКИТ должность, ученая степень	И.В. Матковский инициалы,фамилия

Красноярск 2022