**Департамент образования и науки города Москвы**

**Государственное автономное образовательное учреждение**

**высшего образования города Москвы**

**«Московский городской педагогический университет»**

**Цифровая школа**



**ПРИЛОЖЕНИЯ-ПОМОЩНИКИ УЧИТЕЛЯМ ТЕХНИЧЕСКИХ КЛАССОВ**

Выполнил:

Морозов Илья Николаевич,

ученик 10 класса

«Цифровая школа» МГПУ

Руководитель:

Обозненко Арсений Михайлович,

учитель, «Цифровая школа» МГПУ

Оглавление

[Введение 3](#__RefHeading___Toc1534_4208850153)

[Цель 3](#__RefHeading___Toc1400_3717633682)

[Задачи 3](#__RefHeading___Toc1548_4208850153)

[Дорожная карта 5](#__RefHeading___Toc1536_4208850153)

[Работа над проектом 7](#__RefHeading___Toc1538_4208850153)

[Изучить рынок 7](#__RefHeading___Toc1550_4208850153)

[Подобрать оборудование 9](#__RefHeading___Toc1410_3717633682)

[Изучить возможности сервисов/сетевых протоколов 11](#__RefHeading___Toc1540_4208850153)

[Ход работы 13](#__RefHeading___Toc1542_4208850153)

[Продукт 18](#__RefHeading___Toc1552_4208850153)

[Перспективы работы над проектом: 19](#__RefHeading___Toc1245_1384715569)

[Приложение 19](#__RefHeading___Toc1546_4208850153)

# Введение

В современном мире образование активно переходит в цифровой формат, и учителя все чаще нуждаются в удобных и современных инструментах для работы. Такие инструменты должны помогать педагогам эффективно управлять учебным процессом, контролировать выполнение заданий и поддерживать взаимодействие с учениками. Однако многие школы сталкиваются с серьезной проблемой: у них нет единой платформы, которая позволяла бы учителям легко получать доступ к файлам на разных компьютерах и следить за тем, что происходит на экранах учеников. Это создает сложности в организации уроков и снижает эффективность обучения. Решение этой проблемы могло бы значительно упростить работу учителей и улучшить качество образовательного процесса.

# Цель

**Разработка приложения для удаленного контроля экранов и файлов учеников в IT-классах.**

# Задачи

**1. Изучить рынок доступных продуктов/аналогов**

* **1.1.** Определить **критерии сравнения** (цена, функционал, совместимость с оборудованием, безопасность).
* **1.2.** Составить список существующих **программ для удаленного управления** (например, TeamViewer, AnyDesk) и **платформ для обучения** (Google Classroom, Microsoft Teams).
* **1.3.** Проанализировать их **сильные и слабые стороны** в контексте задач учителей IT-классов.
* **1.4.** Сравнить с требованиями проекта: что можно заимствовать, а чего не хватает в текущих решениях.

**2. Подобрать оборудование**

* **2.1.** Определить **минимальные технические требования** для работы приложения:
  + Для серверов (мощность, объем памяти).
  + Для компьютеров учеников и учителей (ОС, процессор, оперативная память).
* **2.2.** Выбрать оборудование, которое поддерживает **удаленный мониторинг** и **совместимо с сетевыми протоколами**.
* **2.3.** Проверить совместимость оборудования с программным обеспечением (например, драйверы, поддержка ОС).

**3. Изучить возможности сервисов/сетевых протоколов**

* **3.1.** Исследовать **протоколы для удаленного доступа** (например, RDP, SSH, VNC) и их безопасность.
* **3.2.** Протестировать **сервисы для передачи файлов** (FTP, SMB) и их интеграцию в приложение.
* **3.3.** Определить, какие **сетевые инструменты** подходят для мониторинга экранов в реальном времени.
* **3.4.** Проверить совместимость протоколов с выбранным оборудованием.

**4. Создать прототип приложения**

* **4.1.** Разработать **интерфейс для учителя**:
  + Панель управления учениками.
  + Просмотр экранов в реальном времени.
  + Доступ к файловым системам.
* **4.2.** Реализовать **базовые функции**:
  + Подключение к компьютерам учеников через **сетевой протокол**.
  + Передача файлов между учителем и учениками.

**5. Создать установочные скрипты для программ**

* **5.1.** Написать **скрипты для автоматической установки** приложения:
  + Для Linux (Bash).
* **5.2.** Добавить в скрипты проверку **системных требований** (например, версия ОС, наличие библиотек).
* **5.3.** Протестировать установку на разных операционных системах.
* **5.4.** Создать **инструкцию по установке** для пользователей (шаги, возможные ошибки).

# Дорожная карта

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Этап | Сентябрь | Октябрь | Ноябрь | Декабрь | Январь | Февраль |
| Изучить рынок | ++ |  |  |  |  |  |
| Список аналогов | + |  |  |  |  |  |
| Технические требования |  | ++ |  |  |  |  |
| Выбор оборудования |  | + |  |  |  |  |
| Изучить сетевые протоколы | + | + | + |  |  |  |
| Создать установочные скрипты |  |  | + | + |  |  |
| Тестирование установки |  |  | + | + |  |  |
| Инструкция к установщику |  |  |  | + | + |  |
| Создание прототипа приложения |  |  |  |  | +- |  |
| Базовые функции |  |  |  |  | +- | +- |
| Тестирование |  |  |  |  |  | - |

# Работа над проектом

## Изучить рынок

На рынке существует несколько программ для удаленного управления и мониторинга экранов, которые распространяются под **открытой лицензией** и активно используются в образовательных учреждениях. Основные аналоги, которые можно рассмотреть:

1. **Veyon**
   * **Описание**: Программа для управления компьютерными классами, которая позволяет учителям наблюдать за экранами учеников, блокировать их компьютеры, отправлять сообщения и управлять доступом к приложениям.
   * **Особенности**:
     + Поддержка **Linux** и **Windows**.
     + Интеграция с **LDAP** для управления пользователями.
     + Возможность удаленного запуска приложений.
   * **Лицензия**: Открытая (GPLv2).
2. **LANSchool**
   * **Описание**: Популярное решение для управления учебным процессом в компьютерных классах. Позволяет учителям видеть экраны учеников, блокировать доступ к сайтам и приложениям, а также проводить тесты.
   * **Особенности**:
     + Поддержка **Windows** и **macOS**.
     + Встроенный чат для общения между учителем и учениками.
     + Возможность удаленного управления компьютерами.
   * **Лицензия**: Проприетарная, но с бесплатными пробными версиями.
3. **NetSupport School**
   * **Описание**: Программа для управления учебным процессом, которая предоставляет инструменты для мониторинга экранов, контроля интернета, проведения тестов и удаленного управления.
   * **Особенности**:
     + Поддержка **Windows**, **macOS** и **Linux**.
     + Встроенный модуль для проведения опросов и тестов.
     + Возможность блокировки USB-устройств.
   * **Лицензия**: Проприетарная, но с пробными версиями.

## Подобрать оборудование

Основной языкприложения— **Python**.  
Скрипты установки программ и конфигурации **Samba** написаны на **Bash**.  
Основной сетевой протокол — **VNC (Virtual Network Computing)**.

**Используемые библиотеки и инструменты**

1. **Для захвата и передачи экрана**:
   * **MSS** — библиотека для захвата экрана.
   * **OpenCV** — обработка и кодирование изображений.
   * **Socket** — передача данных по сети.
2. **Для графического интерфейса**:
   * **Kivy** — фреймворк для создания кроссплатформенных приложений.
3. **Для работы с файловыми системами**:
   * **Samba** — предоставление общего доступа к файлам по сети.

**Требования к оборудованию**

****1. Сервер (учительский компьютер)****

* **Процессор**: Минимум 2 ядра, 2.0 ГГц (рекомендуется 4 ядра, 2.5 ГГц и выше).
* **Оперативная память**: Минимум 4 ГБ (рекомендуется 8 ГБ).
* **Графическая карта**: Интегрированная или дискретная с поддержкой OpenGL 2.0+.
* **Сеть**: Ethernet (рекомендуется Gigabit) или Wi-Fi (стандарт 802.11ac/ax).
* **Операционная система**: Linux (Ubuntu 20.04+, Debian 10+ или аналогичные).
* **Графическое окружение**: Одно из поддерживаемых:
  + **GNOME**
  + **XFCE**
  + **KDE Plasma**

**2. Клиентские устройства (ученические компьютеры)**

* **Процессор**: Минимум 2 ядра, 1.5 ГГц (рекомендуется 2 ядра, 2.0 ГГц и выше).
* **Оперативная память**: Минимум 2 ГБ (рекомендуется 4 ГБ).
* **Графическая карта**: Интегрированная с поддержкой OpenGL 2.0+.
* **Сеть**: Ethernet или Wi-Fi (стандарт 802.11n/ac).
* **Операционная система**: Linux (Ubuntu 20.04+, Debian 10+ или аналогичные).
* **Графическое окружение**: Одно из поддерживаемых:
  + **GNOME**
  + **XFCE**
  + **KDE Plasma**

**Сетевые требования**

1. **Единая сеть**: Все устройства (сервер и клиенты) должны быть подключены к одной локальной сети через **Ethernet** или **Wi-Fi**.
2. **Пропускная способность**:
   * Для Ethernet: минимум 100 Мбит/с (рекомендуется 1 Гбит/с).
   * Для Wi-Fi: минимум 802.11n (рекомендуется 802.11ac/ax).
3. **Порты**:
   * **Порт 5000** — для передачи данных между сервером и клиентами.
   * **Порт 139/445** — для работы Samba (если требуется доступ к файловым системам).

## Изучить возможности сервисов/сетевых протоколов

#### ****1. TigerVNC и удаленный доступ к экранам****

**TigerVNC** — это программа, которая позволяет удаленно управлять компьютером через **VNC-протокол** (Virtual Network Computing). В контексте вашего проекта на Python, TigerVNC может быть интегрирован для обеспечения удаленного доступа к экранам учеников.

* **Серверная часть**:
  + Это компьютер, к которому требуется доступ (например, компьютер ученика).
  + Серверная часть реализована через скрипт server.py, который использует библиотеку **MSS** для захвата экрана и **OpenCV** для кодирования изображения.
  + Изображение передается по сети с помощью **сокетов** на порт 5000.
* **Клиентская часть**:
  + Это компьютер, с которого осуществляется управление (например, компьютер учителя).
  + В вашем проекте клиентская часть реализована через скрипт client.py, который получает изображение с сервера и отображает его с помощью **OpenCV**.

**Преимущества использования Python для VNC:**

* **Гибкость**: Python позволяет легко адаптировать функционал под нужды учебного процесса.
* **Простота интеграции**: Библиотеки **MSS** и **OpenCV** упрощают захват и передачу экрана.
* **Кроссплатформенность**: Python работает на Linux, Windows и macOS, что делает решение универсальным.

**2. Samba и доступ к файловым системам**

**Samba** — это программа, которая позволяет организовать общий доступ к файлам и папкам между компьютерами в сети.

* **Основные возможности Samba:**
  + Создание **общих папок**, которые видны всем пользователям в сети.
  + Настройка **папок с ограниченным доступом** (например, только для учителя).
  + Использование **учетных записей и паролей** для защиты доступа.

**Пример использования в проекте:**

* Учитель может создать общую папку на своем компьютере, куда ученики будут загружать выполненные задания.
* Ученики могут получить доступ к материалам урока, которые учитель разместил в общей папке.

**Преимущества использования Samba:**

* **Простота настройки**: Samba легко интегрируется в Linux-системы.
* **Гибкость**: Можно настроить доступ на уровне пользователей и групп.
* **Совместимость**: Samba поддерживается на всех основных операционных системах.

****3. Сетевые протоколы и Python****

В вашем проекте используются следующие сетевые технологии:

* **Сокеты (Socket)**:
  + Основной инструмент для передачи данных между сервером и клиентами.
  + В скриптах server.py и client.py сокеты используются для передачи изображений экрана.

**Преимущества использования Python для работы с сетевыми протоколами:**

* **Библиотеки**: Python имеет богатую экосистему библиотек для работы с сетями (например, socket, paramiko для SSH, requests для HTTP).
* **Простота**: Python позволяет быстро разрабатывать и тестировать сетевые приложения.

# Ход работы

****1. Установка зависимостей и настройка окружения  
Цель: Установить необходимые пакеты и собрать проект.****

****1. Проверка прав доступа(рис.1):****

Скрипт автоматически запрашивает права суперпользователя (root/sudo), если запущен без них

**Sudo** (сокращение от SuperUser Do — «действовать от имени суперпользователя») — это способ выполнить команду с правами администратора.

**2.Определение дистрибутива**

Здесь скрипт определяет какой дистрибутив используется при запуске, это нужно для корректного получения библиотек.

**2.Настройка общего доступа через Samba.**

**Скрипт:samba-install.sh.(рис. 3)**

**1. Проверка на права суперпользователя(рис. 1)**

**2. Настройка samba и создание пользователя.**

Здесь скрипт путем простых вопросов помогает пользователю создать папку общего доступа и создает конфигурационный файл.

**3. Запуск службы, завершение настройки(рис.3)**

Скрипт перезапускает службу и выводит данные о созданном пользователе, папке общего доступа и директории,в которой находится конфигурационный файл.

**3. Работа с компьютерами учеников, доступ к содержимому экранами(рис.4)**

**Скрипты : main.py, server.py, client.py.**

****Цель:** Обеспечить просмотр экранов учеников и управление учебным процессом.**

Здесь в скрипте main.py импортируются компоненты для построения приложения, асинхронности и работы с сетью.

В классе MainScreen создается меню основного окна, генерируется список устройств в виде кнопок. По нажатию на кнопку пользователя переместит в окно работы с девайсом. Есть возможность написать в чат.

**3.Правильное завершение работы:(рис.6)**

Скрипт сохраняет работу учителя, и корректно закрывает процессы.

****Продукт****

В рамках проекта разработано **программное решение для учителей IT-классов**, которое позволяет:

1. **Удалённо просматривать экраны учеников** в реальном времени через интеграцию с **TigerVNC**.
2. **Организовывать общий доступ к файлам** между устройствами в локальной сети с помощью **Samba**.
3. **Автоматизировать настройку** среды с помощью **Bash-скриптов**, что упрощает установку даже для новичков.

Решение ориентировано на **Linux-системы** и подходит для школ, где важно:

* Централизованно контролировать учебный процесс.
* Быстро раздавать задания и собирать результаты.
* Избежать использования сложных проприетарных программ.

## Перспективы работы над проектом:

**Улучшение графического интерфейса:**

* Добавление интерактивных элементов: кнопки для блокировки экранов, чат для общения с учениками, панель быстрого доступа к файлам.
* **Расширение функционала удалённого доступа:**
* **Блокировка экранов учеников** на время объяснения материала.
* **Удалённый запуск приложений** (например, IDE для программирования).
* Поддержка **нескольких мониторов** для учителя.
* **Оптимизация сетевого взаимодействия:**
* Реализация **автоматического обнаружения устройств** в локальной сети.
* **Сжатие данных** для работы на слабых интернет-соединениях.
* Добавление **шифрования трафика** для защиты данных.

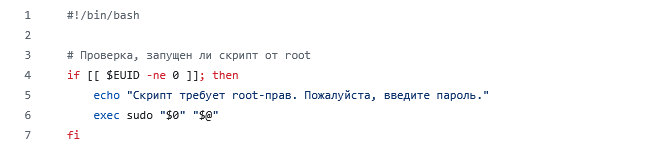
## Приложение

Примечания:

* **TigerVNC** — система для удаленного доступа к экрану компьютера через протокол VNC. Позволяет учителю видеть и управлять экранами учеников.
* **Samba** — программа для организации общего доступа к файлам в сети. Используется для обмена заданиями между учителем и учениками.
* **Bash** — командная оболочка Linux, используемая для автоматизации установки и настройки программ.
* **CMake** — инструмент для управления сборкой проекта. Помогает компилировать код на разных платформах.
* **Kivy** — фреймворк для создания графических интерфейсов на Python. Используется в вашем проекте для визуальной части.
* **OpenCV** — библиотека для обработки изображений. Нужна для кодирования и декодирования кадров экрана.
* **Socket** — технология для передачи данных по сети. Основа взаимодействия между сервером и клиентами.
* **MSS** — библиотека Python для захвата содержимого экрана.

**Рисунки**

Рисунок 1:



## Рисунок 2:



Рисунок 3:

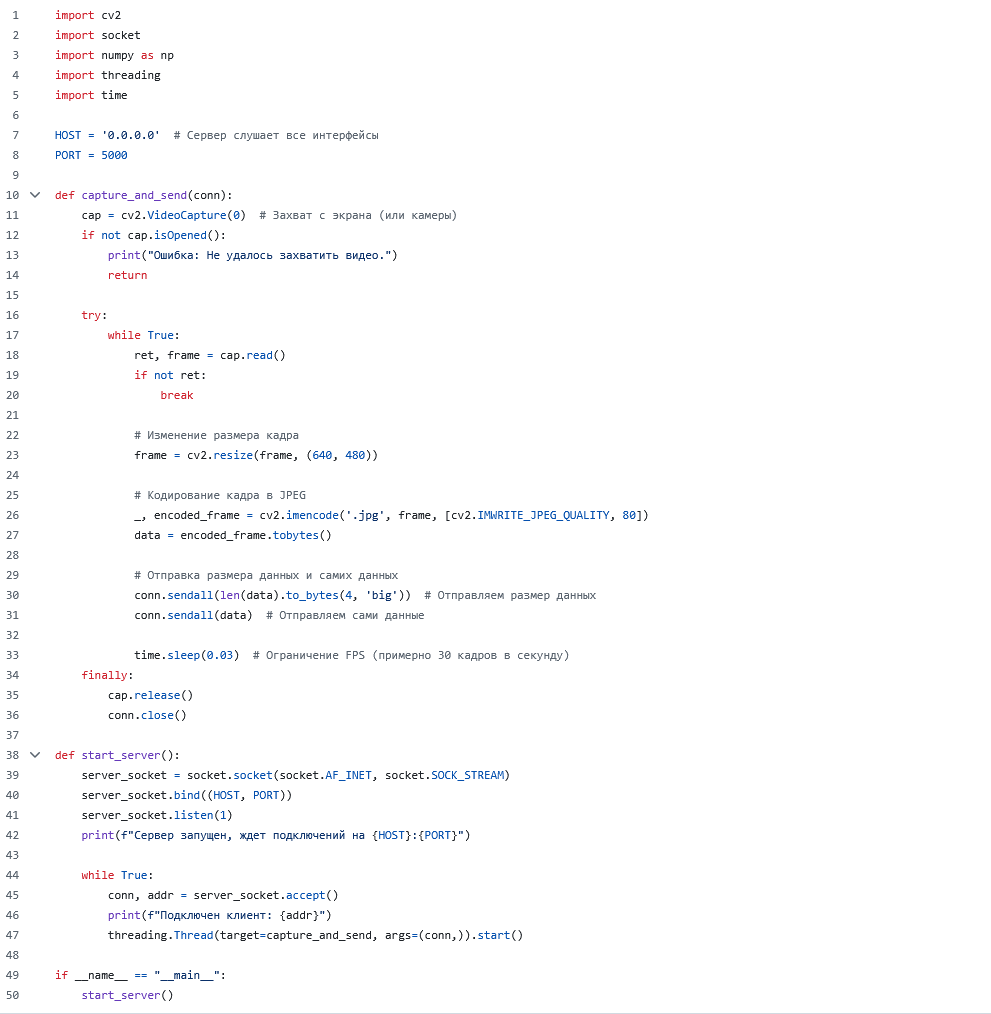
Рисунок 4:

Рисунок 5:

Рисунок 6:

