Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
Л.Н. Храмова
подпись инициалы, фамилия
«14» 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОЙ РЕГИСТРАЦИИ ПОСЕЩАЕМОСТИ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ ARDUINO

Руководитель 14.06.24 канд. пед. наук, доцент А. В. Фирер
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 14.06.24 В. Д. Пермяков
подпись, дата инициалы, фамилия

Нормоконтролер 14.06.24 А. В. Фирер
подпись, дата инициалы, фамилия

Лесосибирск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Проектирование и разработка системы автоматической регистрации посещаемости на базе микроконтроллеров Arduino». Работа содержит 70 страниц текстового документа, 37 иллюстраций, 6 таблиц, 42 использованных источника и 2 приложения.

ARDUINO, СИСТЕМА КОНТРОЛЯ И УПРАВЛЕНИЯ ДОСТУПОМ, АВТОМАТИЗИРОВАННАЯ СИСТЕМА, РЕГИСТРАЦИЯ ПОСЕЩЕНИЙ.

Цель исследования — теоретически обосновать и разработать автоматизированную систему автоматической регистрации посещаемости студентами общежития с использованием микроконтроллеров Arduino.

Объект исследования — процесс регистрации посещаемости студентами общежития.

Предмет исследования — процесс проектирования и разработки автоматизированной системы регистрации посещаемости общежития.

Для достижения поставленной цели необходимо решить следующие задачи:

− изучить теоретические основы проектирования систем контроля и управления доступом;

− выделить требования к разрабатываемой системе;

− разработать автоматизированную систему автоматической регистрации посещаемости студентами общежития на базе микроконтроллера Arduino.

В ходе написания выпускной квалификационной работы автор принял участие в научно-практических конференциях различного уровня, приняты к публикации две статьи по теме исследования. В результате выполнения выпускной квалификационной работы спроектирована и разработана автоматизированная система регистрации посещений общежития на базе микроконтроллеров Arduino.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире, где информационные технологии играют ключевую роль в различных сферах жизни, автоматизация процессов становится необходимостью для повышения эффективности и надежности систем контроля и управления доступом. Одной из таких сфер является управление общежитиями учебных заведений, где ведение точного и своевременного учета проживающих студентов имеет важное значение. Введение автоматизированной системы учета посещений студентов на базе микроконтроллера Arduino позволяет решить множество задач, связанных с обеспечением безопасности, оптимизацией учебного процесса, управлением ресурсами и соблюдением правил проживания. Под автоматизированной системой понимается «…система, состоящая из комплекса средств автоматизации, реализующего информационную технологию выполнения установленных функций, и персонала, обеспечивающего его функционирование.» [10, с. 1].

«Управление предприятием невозможно без достоверной информации о процессах, происходящих в области его деятельности. Так, производителю товаров нужно иметь сведения о наличии на его складах сырья и торговой продукции, о состоянии производственных подразделений, о контрагентах, поставщиках и заказчиках, о расчетах с ними, о спросе на свою продукцию, об эффективности рекламы и т. п. Коллекционеру марок нужна информация о выпущенных марках, их ценности, о других коллекционерах и их коллекциях...» [21, с. 25].

В контексте проектирования и разработки системы для общежития возникает необходимость в автоматизации процесса учета проживающих, поскольку значимость своевременного и точного ведения информации о студентах непрерывно возрастает по нескольким причинам. Во-первых, фиксация посещений студентами общежития позволяет контролировать доступ и обеспечивать безопасность всех проживающих, что особенно важно в условиях современных вызовов в области безопасности. Во-вторых, точные данные о

посещениях позволяют администрации общежития эффективно планировать и организовывать учебные и воспитательные мероприятия, а также контролировать дисциплину студентов. В-третьих, знание численности и активности проживающих студентов помогает эффективно управлять ресурсами общежития, такими как электроэнергия и вода. Наконец, фиксация посещений помогает контролировать соблюдение правил проживания, включая сроки выезда и возвращения, а также обязательства по содержанию жилплощади. Таким образом, автоматизация процесса учета проживающих студентов в общежитии становится необходимой для обеспечения безопасности, организации учебного процесса, эффективного управления ресурсами и соблюдения правил проживания. Для этих целей используются системы контроля и управления доступом (далее — СКУД) — совокупность средств контроля и управления доступом, обладающих технической, информационной, программной и эксплуатационной совместимостью [9, с. 8].

Цель исследования — теоретически обосновать и разработать автоматизированную систему автоматической регистрации посещаемости студентами общежития с использованием микроконтроллеров Arduino.

Объект исследования — процесс регистрации посещаемости студентами общежития.

Предмет исследования — процесс проектирования и разработки автоматизированной системы регистрации посещаемости общежития.

Для достижения поставленной цели необходимо решить следующие задачи:

− изучить теоретические основы проектирования систем контроля и управления доступом;

− выделить требования к разрабатываемой системе;

− разработать автоматизированную систему регистрации посещаемости студентами общежития на базе микроконтроллера Arduino.

Методы исследования:

– теоретические: анализ учебной и научно-технической литературы по теме исследования, обобщение и сравнительный анализ, моделирование;

– эмпирические: наблюдение, тестирование программного обеспечения.

Результаты исследования представлены на следующих научных мероприятиях:

– VII Всероссийская научно-практическая конференция «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 1–2 ноября 2023 г., 3-е место).

– VIII Всероссийский (IX Региональный) молодежный форум «Российское могущество прирастать будет Сибирью…» (г. Лесосибирск, ЛПИ – филиал СФУ, 14 декабря 2023 г., участие).

– III Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» в секции «Информационные технологии в образовании, науке и производстве» (г. Лесосибирск, ЛПИ – филиал СФУ, 9 апреля 2024 г., участие).

– III Всероссийский конкурс научных работ «Молодежный научный потенциал» (г. Лесосибирск, ЛПИ – филиал СФУ, 10 апреля 2024 г., 3 место).

– XX Международная научная конференция студентов, аспирантов и молодых ученых «Проспект Свободный – 2024» (г. Красноярск, ЛПИ – филиал СФУ, 15-20 апреля 2024 г., участие).

По результатам исследования приняты к публикации статьи:

1. Пермяков В. Д. / Проектирование и разработка автоматизированной системы регистрации посещений общежития на базе микроконтроллера Arduino / В. Д. Пермяков // Цифровые технологии в образовании, науке и технике: материалы XX Международной научной конференции студентов, аспирантов и молодых ученых «Проспект Свободный – 2024», г. Красноярск, / Сибирский федеральный университет. – Красноярск, 2024.

2. Пермяков В. Д. / Arduino или как сделать мир вокруг себя умнее / В. Д. Пермяков / В. Д. Пермяков // Информационные технологии в образовании, науке и производстве: материалы III Всероссийского молодежного научного форума «Современное педагогическое образование: теоретический и прикладной аспекты» / ЛПИ – филиал СФУ. – Лесосибирск, 2024.

Структура работы – работа состоит из введения, двух глав, заключения, списка использованных источников, включающего 42 наименования, двух приложений. Результаты работы представлены в 37 иллюстрациях, 6 таблицах. Общий объем работы – 70 страниц.

1 Теоретические основы проектирования систем контроля и управления доступом

1.1 История развития

История систем регистрации посещений представляет собой захватывающий путь эволюции, начиная с первобытных методов и достигая современных высокотехнологичных систем. В древние времена, когда не существовало современных электронных систем, контроль доступа реализовывался различными способами, адаптированными к потребностям и возможностям того времени.

Физические барьеры включали стены, рвы и ворота, которые окружали древние города и поселения для ограничения доступа людей и товаров. Ворота обычно охранялись стражниками, проверяющими личность и разрешения на вход. Также использовались замки и ключи для защиты дверей, сундуков и других ценных предметов, причем сложность замков варьировалась от простых деревянных шпингалетов до самодельных металлических механизмов. Важные места, такие как дворцы и храмы, могли иметь секретные проходы или лабиринты, чтобы затруднить доступ посторонним. Социальные нормы и правила касались иерархии и статуса человека, определяя доступ к определенным местам или ресурсам. Например, только дворяне могли входить в определенные части дворца. Ношение определенной одежды или знаков указывало на статус или принадлежность к группе, давая или ограничивая доступ к тем или иным местам. Религиозные ритуалы и табу также играли роль в контроле доступа, так как некоторые места считались священными и доступными только для посвященных. Персонал и охрана включали стражников и часовых, которые физически охраняли важные места, такие как дворцы, храмы и казначейства. Шпионы и информаторы использовались для сбора информации о потенциальных угрозах и предотвращения несанкционированного доступа. Охранные собаки дрессировались для защиты людей и имущества от несанкционированного доступа.

Исследование методов контроля доступа в древние времена дает представление об общественных, политических и экономических структурах древних цивилизаций. Например, наличие сложных систем охраны дворцов говорит о централизованной власти и строгой социальной иерархии. Использование стен и рвов для защиты городов указывает на необходимость защиты ресурсов и торговли. Также это помогает оценить изобретательность и находчивость наших предков. Древние люди, не имея современных технологий, смогли создать эффективные системы контроля и управления доступом, используя подручные средства и социальные нормы. Потребность в контроле доступа существовала всегда, и методы его обеспечения совершенствовались по мере появления новых угроз и технологий.

Однако настоящий прорыв произошел с появлением электронных систем регистрации посещений в середине XX века. Как писали Воробьева А. А. и Пантюхин И. С. в учебном пособии «История развития программно-аппаратных средств защиты информации»: «…Начав свое активное развитие в 1980–х, системы биометрической идентификации продолжали совершенствоваться, повышалась точность распознавания. Системы были достаточно дороги и в основном применялись правительственными и военными организациями, а также крупными коммерческими компаниями. Основными целями применения биометрии правоохранительными органами была идентификация личности (поиск преступников и опасных лиц (в аэропортах, на спортивных мероприятиях)). В коммерческих организациях системы применялись в СКУД для физического разграничения доступа, контроля рабочего времени и идентификации сотрудников.» [5, с. 30].

В последние десятилетия активно применяются биометрические системы регистрации, основанные на уникальных физиологических или поведенческих характеристиках человека. Эти системы, такие как сканеры отпечатков пальцев или распознавание лица, обеспечивают высокий уровень безопасности и точности, снижая при этом риск манипуляций и фальсификаций.

Таким образом, история развития автоматизированных систем регистрации посещений отражает постоянное стремление к повышению точности, эффективности и удобства использования таких систем, что делает их неотъемлемой частью функционирования современных организаций и учреждений.

1.2 Сущность и классификация систем контроля и управления доступом

1.2.1 Определения основных понятий в области систем контроля и управления доступом

Системы контроля и управления доступом представляют собой комплекс технических и программных средств, предназначенных для контроля и управления доступом людей, товаров и транспортных средств на объект. В таблице 1 приведен ряд терминов и определений, необходимых для дальнейшего изложения материала.

Таблица 1 — Термины и определения

Термин	Определение	Пример
Аутентификация	Процесс опознавания субъекта или объекта путем сравнения введенных идентификационных данных с эталоном (образом), хранящимся в памяти системы для данного субъекта или объекта [9, с. 6]	Пользователь вводит свой пароль или использует биометрический идентификатор, чтобы подтвердить свою личность
Верификация	Подтверждение, посредством представления объективных свидетельств (3.8.3), того, что установленные требования (3.6.4) были выполнены [12, с. 18]	Система СКУД проверяет, соответствует ли отпечаток пальца пользователя тому, который хранится в базе данных
Идентификация	Процесс опознавания субъекта или объекта по присущему ему или присвоенному ему идентификационному признаку. Под идентификацией понимают также	Пользователь показывает свой пропуск или называет свое имя

Продолжение таблицы 1

Термин	Определение	Пример
	присвоение субъектам и объектам доступа идентификатора и (или) сравнение предъявляемого идентификатора с перечнем присвоенных идентификаторов. [9, с. 7]	
Пропускной режим	Пропускной режим устанавливается в целях обеспечения прохода (выхода) учащихся (воспитанников), сотрудников и посетителей в здание образовательной организации, въезда (выезда) транспортных средств на территорию образовательной организации, вноса (выноса) материальных ценностей, исключающих несанкционированное проникновение граждан, транспортных средств и посторонних предметов на территорию и в здание образовательной организации [9, с.7]	Пропускной режим может включать в себя правила проверки документов, досмотра сумок и использования пропусков
Электронный журнал регистрации событий безопасности (журнал событий)	Объект (файл в электроном виде) или их совокупность в информационной (автоматизированной) системе, предназначенный (предназначенные) для хранения записей о событиях безопасности [10, с. 2]	Журнал событий может быть использован для расследования инцидентов безопасности
Идентификатор доступа, идентификатор	Уникальный признак субъекта или объекта доступа. В качестве идентификатора может использоваться запоминаемый код, биометрический признак или вещественный код. Идентификатор, использующий вещественный код - предмет, в который (на который) с помощью специальной технологии занесен идентификационный признак в виде кодовой информации (карты, электронные ключи, брелоки и др. устройства) [9, с. 7]	Сотрудник может использовать свой пропуск в качестве идентификатора для входа в здание

Окончание таблицы 1

Термин	Определение	Пример
Устройство считывающее (УС), считыватель	Устройство, предназначенное для считывания (ввода) идентификационных признаков [9, с. 9]	Считыватель на турникете может считывать информацию с магнитных карт
Контроллер доступа	Аппаратное устройство в составе средств управления СКУД [9, с. 7]	Контроллер на турникете может разрешить или запретить проход пользователя
Устройства исполнительные	Устройства или механизмы, обеспечивающие приведение в открытое или закрытое состояние УПУ (электромеханические, электромагнитные замки, электромагнитные защелки, механизмы привода шлюзов, ворот, турникетов и другие подобные устройства) [9, с. 9]	Замок на двери может быть открыт с помощью электромагнитного замка
Программное обеспечение	Совокупность программ и программных документов, предназначенная для отладки, функционирования и проверки работоспособности АС [10, с. 4]	Программное обеспечение СКУД может использоваться для создания отчетов о посещаемости, управления правами доступа и генерации тревожных сообщений
База данных	Совокупность взаимосвязанных данных, организованных в соответствии со схемой базы данных таким образом, чтобы с ними мог работать пользователь [12, с. 1]	База данных СКУД может хранить информацию о ФИО сотрудников, их фотографиях, правах доступа и времени входа и выхода
Несанкционированный доступ	Доступ субъектов или объектов, не имеющих права доступа [9, с. 7]	Проникновение нарушителя в служебное помещение

Понимание этих основных понятий важно для изучения и работы с СКУД.

1.2.2 Назначение и классификация систем контроля и управления доступом

Назначение СКУД заключается в защите от несанкционированного доступа, контроле доступа к ресурсам, автоматизации пропускного режима и сборе статистических данных. Данные системы препятствуют проникновению на объект посторонних лиц и контролируют перемещение людей и товаров внутри него. Например, в жилом комплексе СКУД ограничивает доступ к подъездам, лифтам и другим помещениям, защищая жильцов от краж и вандализма. Также СКУД позволяют ограничивать доступ к определенным зонам объекта в зависимости от уровня полномочий пользователей. На производственном предприятии это может быть ограничение доступа к складам и цехам, где хранится ценное имущество или ведутся конфиденциальные работы. Автоматизация пропускного режима позволяет ускорить процесс проверки прав доступа, сокращая время ожидания и повышая эффективность работы системы, например, при автоматической проверке паспортов и посадочных талонов пассажиров в аэропорту. Кроме того, как отмечает А. В. Петров [24, с. 64] СКУД реализуют «…формирование базы данных и ведение журнала учета посетителей в организации.».

При выборе СКУД крайне важно учитывать ряд ключевых факторов, таких как размеры объекта, его функциональное назначение, уровень безопасности, а также имеющийся бюджет. Размеры объекта определяют необходимое количество компонентов СКУД и их расположение для обеспечения полного охвата помещений. Назначение объекта влияет на требования к системе, например, в жилых комплексах может потребоваться больше акцента на удобство использования, в то время как в офисных зданиях приоритет будет отдан безопасности. Специфические требования безопасности могут включать в себя необходимость биометрической идентификации, контроля доступа в определенные часы или интеграции с другими системами безопасности. Бюджет играет важную роль в определении того, какие компоненты и функциональные возможности могут быть реализованы в рамках доступных финансовых ресурсов.

Анализ источников по теме исследования позволил выделить классификации СКУД по различным признакам, описанным в ГОСТ Р 51241-2008 [9], которые представлены в таблице 2.

Таблица 2 — Классификация СКУД

Признак классификации	**Тип системы**	**Описание**
По функциональным характеристикам [9, с. 12]	1-й класс, системы с ограниченными функциями	Световая индикация о состоянии доступа [9, с. 25]
	2-й класс, системы с расширенными функциями	Установка временных интервалов доступа, возможность регулирования времени открывания УИ, подключение считывателей различных типов, световая индикация о состоянии доступа, контроль состояния УПУ, регистрация и хранение информации о событиях в энергонезависимой памяти, число событий, хранимых в энергонезависимой памяти, не менее 64, ведение даты и времени возникновения событий, возможность подключения устройства для вывода информации о событиях, возможность интегрирования с системой охранной сигнализации на релейном уровне [9, с. 25]
	3-й класс, многофункциональные системы	Установка уровней доступа, установка временных интервалов доступа, возможность регулирования времени открывания УИ, возможность идентификации по двум признакам, защита от повторного использования идентификатора для прохода в одном направлении, ввод специального идентификационного признака для открывания под принуждением, подключение считывателей различных типов, доступ по «правилу двух (и более) лиц», световая индикация о состоянии доступа, Контроль состояния УПУ, световое и/или звуковое оповещение о попытках НСД, Регистрация и хранение информации о событиях в энергонезависимой памяти, число событий, хранимых в энергонезависимой памяти, не менее 256, ведение даты и времени

Окончание таблицы 2

Признак классификации	Тип системы	Описание
		возникновения событий, возможность подключения устройства для вывода информации о событиях, возможность передачи информации о событиях на ЭВМ, возможность интегрирования с системой охранной сигнализации на релейном уровне, возможность интегрирования с системой охранного телевидения на релейном уровне [9, с. 25]
По способу управления [9, с. 12]	Автономные системы	Для управления одним или несколькими УПУ без передачи информации на центральное устройство управления и контроля со стороны оператора [9, с. 12]
	Централизованные (сетевые)	Для управления УПУ с обменом информацией с центральным пультом и контролем и управлением системой со стороны центрального устройства управления [9, с. 12]
	Универсальные (сетевые)	включающие в себя функции как автономных, так и сетевых систем, работающие в сетевом режиме под управлением центрального устройства управления и переходящие в автономный режим при возникновении отказов в сетевом оборудовании, центральном устройстве или обрыве связи [9, с. 12]
По числу контролируемых точек доступа [9, с. 12]	Малой емкости (не более 64 точек)	Точка доступа - место, где непосредственно осуществляется контроль доступа (например, дверь, турникет, кабина прохода, оборудованные необходимыми средствами) [9, с. 9]
	Средней емкости (от 64 до 256 точек)	
	Большой емкости (более 256 точек)	

1.2.3 Основные компоненты систем контроля и управления доступом

Для эффективной работы СКУД необходимо обеспечить наличие нескольких ключевых компонентов, каждый из которых выполняет свою важную роль. Эти компоненты, описанные в таблице 3, работают совместно, создавая целостную и надежную систему, способную обеспечить высокий уровень безопасности и удобства.

Таблица 3 — Основные компоненты СКУД

Компонент	Описание	Пример использования
Идентификаторы	Используются для идентификации пользователей (например, магнитные карты, RFID-метки, биометрические данные)	Идентификаторы могут быть выданы сотрудникам компании, жильцам жилого комплекса или посетителям мероприятия
Считыватели	Считывают информацию с идентификаторов	Считыватели могут быть установлены на турникетах, дверях, лифтах и других местах, где требуется контроль доступа
Контроллеры доступа	Управляют доступом на объект на основе информации, полученной от считывателей	Контроллеры могут разрешать или запрещать проход пользователей, а также генерировать тревожные сообщения в случае несанкционированного доступа
Устройство исполнительное	Выполняют команды контроллеров (например, турникеты, замки, шлагбаумы)	Исполнительные устройства могут блокировать или разблокировать двери, проходы и другие места, где требуется контроль доступа
Программное обеспечение	Обеспечивает управление СКУД, сбор и анализ данных	Программное обеспечение СКУД может использоваться для создания отчетов о посещаемости, управления правами доступа и генерации тревожных сообщений
Устройства преграждающие управляемые (УПУ)	Устройства, обеспечивающие физическое препятствие доступу и оборудованные исполнительными устройствами для управления их состоянием [9, с.9]	Турникеты, шлюзы, проходные кабины, двери и ворота, оборудованные исполнительными устройствами СКУД, а также другие подобные устройства

Помимо этих основных компонентов, СКУД может включать в себя и другие элементы, такие как системы видеонаблюдения, системы охранной сигнализации, системы контроля рабочего времени и т. д. В то же время, выбор компонентов входящих в состав СКУД зависит от требований к системе и потребностей пользователей.

1.2.4 Преимущества использования систем контроля и управления доступом

Использование СКУД имеет ряд преимуществ. Прежде всего, они повышают уровень безопасности, препятствуя проникновению на объект посторонних лиц и обеспечивая контроль перемещения людей и товаров внутри него, снижают риск краж, ограничивая доступ к ценному имуществу и уменьшая вероятность преступлений. Автоматизация пропускного режима с помощью таких систем ускоряет процесс проверки прав доступа пользователей, сокращая время ожидания и повышая эффективность системы. Кроме того, СКУД собирает статистические данные о посещаемости объекта, времени пребывания людей на нем и частоте использования различных проходов, что полезно для анализа и улучшения бизнес-процессов. Они также могут контролировать рабочее время сотрудников, способствуя оптимизации их работы и повышению производительности. СКУД позволяют сократить расходы на охрану, заменяя или дополняя традиционные методы охраны и снижая затраты на содержание охранников.

Исходя из вышеперечисленного, можно сделать вывод, что СКУД — это эффективный инструмент, который позволяет повысить уровень безопасности, оптимизировать работу персонала и сократить расходы организации.

1.3 Анализ существующих решений

Для успешной разработки автоматизированной системы регистрации посещений общежития важно провести тщательный анализ существующих решений в данной области. Изучение подобных систем позволяет не только понять общие тенденции и лучшие практики, но и выявить их сильные и слабые стороны. Это исследование служит основой для создания оптимальной системы, учитывающей все необходимые функции и обеспечивающей высокий уровень удобства и эффективности для пользователей.

На рынке программного обеспечения существует достаточное количество СКУД. Например, программное обеспечение Wisenet от компании ООО «МТ-ТЕХНО МСК» [22]. Производитель предлагает программное обеспечение для реализации контроля и управления доступом. Данное техническое решение позволяет реализовать контроль и управление доступом на любом предприятии с любыми нуждами. Готовое решение покрывает множество задач, цена за программное обеспечение начинается от 110 тысяч рублей, что для реализации необходимого функционала в рамках данного исследования является большой стоимостью.

Другим представителем, предлагающим готовое решение СКУД, является компания ООО «Прософт-Биометрикс» [41]. Эта компания предлагает готовое решение с необходимыми параметрами. Цена готового решения начинается от 120 тысяч рублей. При этом, для полноценной работы и реализации необходимого функционала, необходимо будет покупать дополнительные модули, которые будут интегрироваться с базовой системой, что снова не даёт возможности при небольших затратах получить необходимый функционал.

«БИТ.Общежитие 8» является техническим решением компании ООО «Первый Бит» [2]. Она имеет похожий функционал в сравнении с разрабатываемой системой. «БИТ.Общежитие 8» предоставляет пользователю функционал, в некоторых аспектах больше, чем необходим. В него входит: гибкое планирование индивидуальной структуры общежития; контроль за проживающими и оперативный учет проживающих в общежитии (заезд, перемещение, выезд); учет материальных ценностей, числящихся за общежитием или закрепленных за проживающими; учет лиц, дополнительно проживающих на жилплощади совместно с лицом, заключившим договор с общежитием; учет предоставления дополнительных услуг (прачечная, телефон и пр.) в общежитии; учет наличия койко-мест и их характеристик; отчетность по численности проживающих и дополнительно проживающих, взаиморасчетам и задолженности проживающих, наличию свободных койко-мест, материальным средствам [32].

Анализ успешных систем автоматизации учета проживания в общежитиях, таких как «БИТ.Общежитие 8», помогает выявить и изучить передовые методы и тенденции в дизайне, взаимодействии с пользователем, функциональности и других аспектах разработки. Несмотря на то, что система «БИТ.Общежитие 8» имеет готовое решение, она обладает недостатками, такими как высокая стоимость (средняя цена пакета составляет примерно 80 тысяч рублей), отсутствие нужного функционала и невозможность контроля доступа через RFID-метки, QR-коды или биометрию. Эта информация может быть использована в качестве основы для создания системы автоматической регистрации посещений, что приведет к улучшению пользовательского опыта.

Анализ существующих решений показал как их преимущества, так и недостатки, такие как высокая стоимость и отсутствие функционала, необходимого в рамках данного исследования, что подчеркивает необходимость разработки более продвинутого и доступного решения.

2 Проектирование и разработка автоматизированной системы автоматической регистрации посещений общежития на базе микроконтроллеров Arduino

2.1. Анализ требований

Целью разрабатываемой автоматизированной системы на базе микроконтроллеров Arduino является автоматизация процесса регистрации посещений студентами общежития.

Анализ требований является одним из ключевых этапов при проектировании и разработке автоматизированной системы автоматической регистрации посещений общежития на базе микроконтроллеров Arduino. Этот процесс включает в себя сбор, документирование и согласование потребностей и ожиданий пользователей, а также, нормативные требования, которым должна удовлетворять разрабатываемая система. Для начала анализа нужно определить, как проходит учет посещений в автоматизированной системе. Для этого разработаем концептуальную модель процесса автоматической регистрации посещения студентами общежития, которая представлена на рисунке 1.

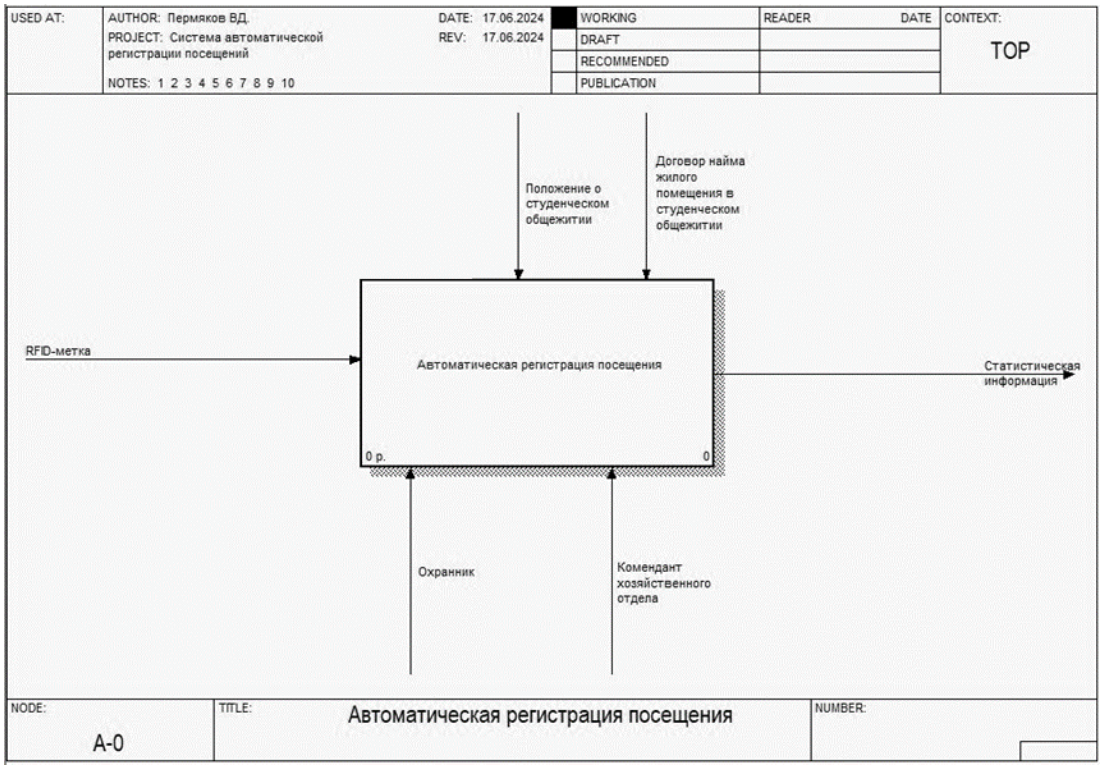

Рисунок 1 — Концептуальная модель системы

Учет посещений в автоматизированной системе регистрации на базе микроконтроллера Arduino осуществляется через последовательность шагов, которые обеспечивают точность и надежность регистрации входов и выходов студентов из общежития. Сначала студент проходит процедуру идентификации при входе в общежитие. Основными средствами идентификации являются считыватели RFID-карт. Студент прикладывает свою RFID-карту к считывателю. После этого микроконтроллер Arduino получает данные о личности студента. Эти данные включают уникальный идентификационный номер RFID-карты, который соответствует зарегистрированному пользователю. Затем полученные данные сравниваются с информацией в базе данных системы. В случае совпадения данных система подтверждает личность студента. После успешной верификации микроконтроллер регистрирует событие посещения, записывая время и дату входа или выхода студента в базу данных. Система обновляет статус присутствия студента в общежитии, что позволяет эффективно управлять доступом и контролировать посещаемость. На рисунке 2 изображена декомпозиция концептуальной модели системы второго уровня, которая графически отображает, описанные выше процессы.

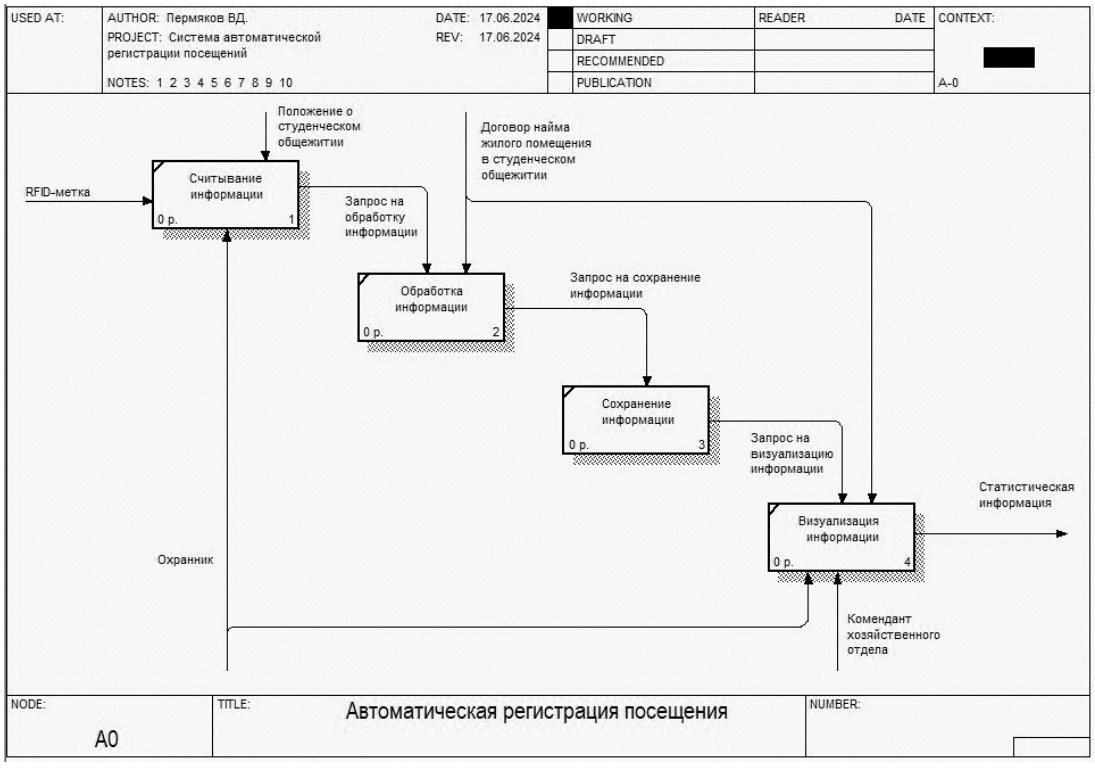

Рисунок 2 — Декомпозиция концептуальной модели системы

Потенциальными пользователями системы являются охранник и комендант хозяйственного отдела.

Определим функциональные и нефункциональные требования, которым должна удовлетворять разрабатываемая система.

2.1.1. Функциональные требования

Система автоматизированной регистрации посещений общежития должна выполнять следующие функции:

а) Автоматическая регистрация входа и выхода студентов:

1) система должна фиксировать время и дату каждого входа и выхода студента из общежития;
2) идентификация студента должна происходить автоматически, без участия вахтера или охранника;
3) данные о входах и выходах должны сохраняться в базе данных.

б) Идентификация студентов с помощью RFID-меток:

1) каждый студент должен иметь свою уникальную RFID-метку;
2) система должна считывать информацию с RFID-метки студента при его входе и выходе из общежития;
3) информация с RFID-метки должна использоваться для идентификации студента в системе.

в) Ведение журнала посещений:

1) журнал посещений должен содержать информацию о времени и дате каждого входа и выхода каждого студента;
2) в журнале должны отображаться ФИО студента и номер RFID-метки;
3) журнал должен быть доступен для просмотра в удобном формате, например, в виде таблицы.

г) Удобный интерфейс для просмотра журнала посещений:

1) пользователи должны иметь возможность легко и быстро просматривать журнал посещений;

2) пользователи должны иметь возможность фильтровать записи в журнале по различным критериям например по ФИО студента.

д) Возможность экспорта журнала посещений в различные форматы:

1) пользователи должны иметь возможность экспортировать журнал посещений в различные форматы, например, в XLSX, CSV или PDF;
2) экспортируемый файл должен содержать всю необходимую информацию из журнала посещений;
3) пользователи должны иметь возможность выбирать формат экспорта.

2.1.2. Нефункциональные требования

Помимо функциональных требований, система должна также соответствовать следующим нефункциональным требованиям:

а) Надежность и устойчивость к сбоям:

1) Система должна работать бесперебойно 24/7;
2) Система должна быть устойчива к сбоям питания и другим техническим сбоям;
3) Данные о входах и выходах студентов должны быть сохранены даже в случае сбоя системы.

б) Простота использования и обслуживания:

1) Система должна быть простой в использовании для студентов и персонала общежития;
2) Интерфейс системы должен быть понятным и интуитивно понятным;
3) Обслуживание системы должно быть простым и не требовать специальных знаний или навыков.

2.2. Архитектура системы и выбор инструментальных средств

Система автоматизированной регистрации посещений общежития будет состоять из следующих модулей:

а) модуль считывателя:

1) будет отвечать за считывание информации с RFID-меток студентов;
2) в состав модуля будет входить RFID-считыватель, который будет подключаться к микроконтроллеру;
3) RFID-считыватель будет считывать уникальный код с RFID-метки студента при его входе и выходе из общежития.

б) модуль базы данных:

1) будет отвечать за хранение информации о студентах, их RFID-метках и журнале посещений;
2) база данных будет подключена к десктопному приложению;
3) база данных будет хранить информацию о ФИО студента, номер RFID-метки, время и дату каждого входа и выхода из общежития.

в) модуль пользовательского интерфейса:

1) предоставляет пользователям интерфейс для просмотра и экспорта журнала посещений;
2) в качестве модуля пользовательского интерфейса используется десктопное приложение, имеющее графический интерфейс пользователя, которое будет доступно через ярлык на рабочем столе;
3) приложение подключено к базе данных и получает из нее информацию о журнале посещений;
4) пользователи могут просматривать журнал посещений в виде списка или таблицы, а также экспортировать журнал в различные форматы;

Все модули системы будут взаимодействовать друг с другом, обеспечивая бесперебойную работу системы.

В качестве центрального контроллера доступа системы будет использоваться микроконтроллер Arduino Nano. Он представляет собой

микроконтроллерную плату, основанную на процессоре ATmega328P, который имеет достаточное количество портов ввода/вывода для подключения всех необходимых компонентов системы. Выбор Arduino Nano был обусловлен простотой в программировании и большим сообществом пользователей.

В качестве считывателя информации с RFID-меток пользователей будет использоваться RFID-считыватель RC522. Он представляет собой модуль RFID-считывателя, совместимый с картами и брелоками стандарта Mifare, который считывает уникальный код с RFID-метки студента, использующийся для его идентификации в системе.

В качестве среды разработки базы данных была выбрана программа MySQL Workbench 8.0. Эта среда разработки позволяет развернуть на рабочем компьютере сервер, на котором будет находиться база данных, и в будущем, используя модуль беспроводной связи для микроконтроллера Arduino отказаться от проводного подключения к компьютеру и осуществлять передачу данных по беспроводной сети. В качестве системы управления базами данных будет использована MySQL 5.7. MySQL — это бесплатная система управления реляционными базами данных с открытым исходным кодом. Она обладает высокой производительностью и надежностью. Созданная в СУБД MySQL база данных будет использоваться для хранения информации о студентах, их RFID-метках и журнале посещений. MySQL Workbench будет использоваться для управления и администрирования базы данных MySQL 5.7.

В качестве модуля пользовательского интерфейса будет использоваться десктопное приложение для персонального компьютера, написанное на языке программирования Python. Python — это универсальный язык программирования, который прост в изучении и использовании, Этот язык программирования был выбран по ряду причин. Язык программирования Python является вторым по популярности языком программирования за 2023 год по версии крупнейшей сети разработчиков Github [25]. На рисунке 3 показан общий рейтинг самых используемых языков программирования среди пользователей Github.

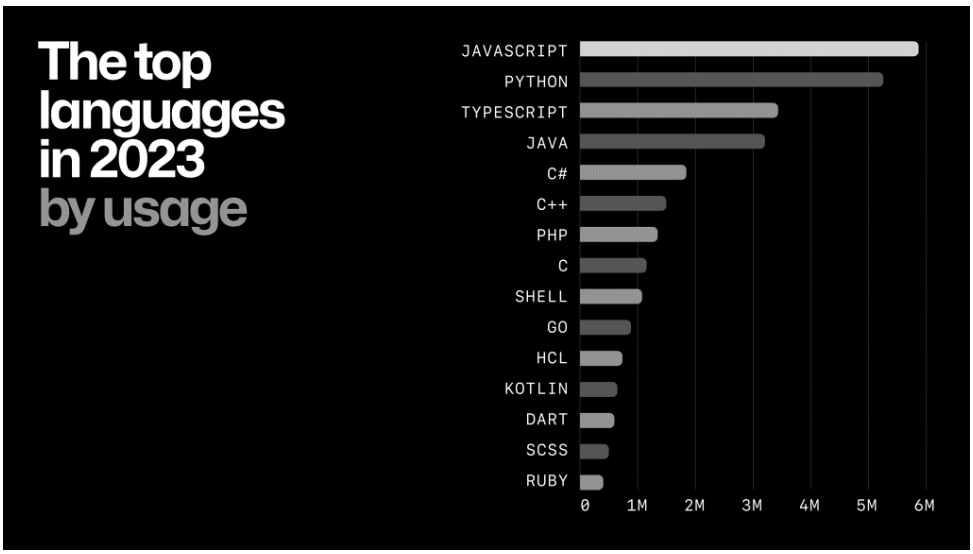

Рисунок 3 — Рейтинг самых распространенных языков программирования по версии Github

Для языка программирования Python существует огромное множество различных библиотек, созданных пользователями Github, что значительно упростит написание кода для десктопного приложения [1]. Также использование различных библиотек позволит в будущем добавлять в приложение различный дополнительный функционал. Вместе с этим, изучая различные технические решения, можно подчерпнуть что-то новое и необычное, что в последствии можно будет использовать в других проектах. Приложение будет использоваться для взаимодействия с базой данных MySQL 5.7 и отображения информации в графическом интерфейсе пользователя.

При выборе среды разработки основными критериями были: отзывчивость интерфейса, многофункциональность, возможность комфортной работы при небольших мощностях, удобный интерфейс, а также возможность подключить репозиторий версий, в котором можно поэтапно отслеживать процесс разработки и написания программы и не бояться безвозвратно потерять прогресс.

В качестве среды разработки программного кода десктопного приложения, написанного на языке Python, была выбрана среда разработки PyCharm 2023.1.3. Эта среда разработки удовлетворяет всем заданным критериям, а также позволяет сразу в графическом интерфейсе отслеживать изменения, что крайне эффективно сказывается на отслеживании ошибок и модернизации десктопного

приложения. Для написания десктопного приложения были использованы различные библиотеки. Прежде всего, это библиотеки configparser, sys, threading. Они являются основными и позволяют работать с файлами, разделять процессы на потоки, что в данном проекте очень важно, а также обеспечивают более удобную работу с классами и функциями. Библиотеки mysql.connector, Pymysql, serial, datetime позволяют работать с базами данных, последовательными портами персонального компьютера и использовать системные данные, к примеру дату и время. При написании кода приложения использовалось несколько фреймворков. Для работы с графическим интерфейсом пользователя использовались фреймворки PyQt5.QtCore(QDateTime), PyQt5.QtWidgets (QApplication, QMainWindow, QListWidgetItem, QDialog, QTableWidgetItem, QTableWidget), mysql.connector(Error). Готовые наборы инструментов помогли оптимизировать работу с интерфейсом и позволили автоматизировать однотипную работу.

В качестве среды разработки графического интерфейса был выбран Qt Designer version 5.11.1. Эта среда разработки очень проста и интуитивно понятна для создания графического интерфейса пользователя. Также важно отметить, что Qt Designer использует язык программирования C++. Qt Designer — это инструмент для создания графических интерфейсов пользователя, который позволяет легко создавать формы, диалоговые окна и другие элементы интерфейса. Графический интерфейс, созданный в Qt Designer, будет использоваться для предоставления пользователям удобного интерфейса для просмотра журнала посещений. В графическом интерфейсе десктопного приложения используется язык HTML и C++ [36]. В программном коде микроконтроллера Arduino, используется язык программирования C++. Использование этих двух языков программирования позволяют экспериментировать с вариациями создания на разных этапах разработки десктопного приложения.

Для отладки и проверки работоспособности десктопного приложения использовались встроенные системы отладки PyCharm и Qt Designer.

Встроенные инструменты отладки позволяют протестировать и найти ошибки, также встроенные инструменты предоставляют возможность оптимизации кода, а также его структурирование в соответствии со стандартами программного обеспечения.

2.3. Разработка программного обеспечения

2.3.1 Сборка и программирование микроконтроллера Arduino

Микроконтроллер Arduino Nano является основным устройством в проекте, который получает данные с чипа и передает их через COM порт в компьютер. Внешний вид и расположение пинов платы микроконтроллера Arduino Nano изображен на рисунке 4.

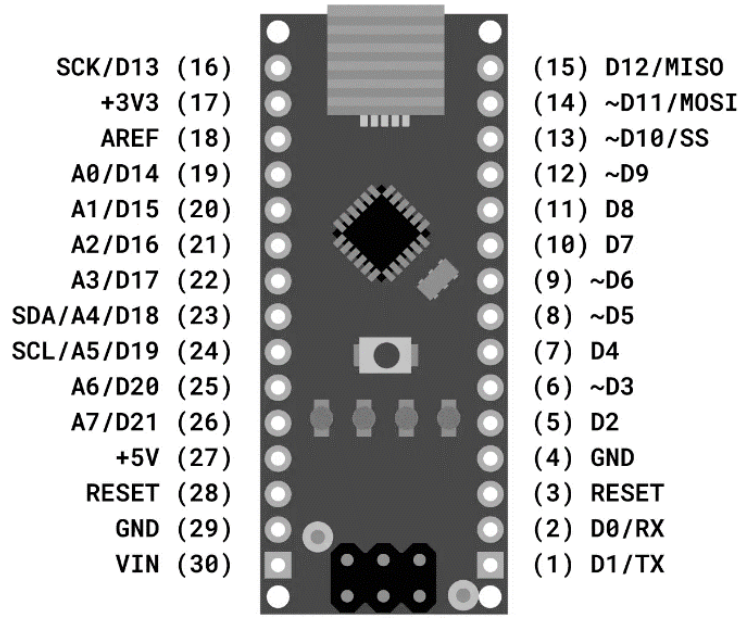

Рисунок 4 — Расположение пинов платы микроконтроллера Arduino Nano

Плата микроконтроллера Arduino Nano имеет следующие характеристики:

– напряжение питания 5 В;

– выходное напряжение питания 3.3 В;

– количество цифровых пинов – 10, из них 5 могут использоваться в качестве выходов ШИМ;

– 8 аналоговых входов;

– максимальный ток цифрового выхода 40 мА;

– флэш-память 32 Кб;

– ОЗУ 2 Кб;

– размеры 19 х 42 мм;

– вес 7 г.

В проекте, на плате Arduino Nano используются цифровые пины D9, D10, D11, D12, D13, питание 3.3 В, GND(GROUND). Помимо платы самого микроконтроллера, используется расширительная плата RFID-RC522, внешний вид и расположение пинов которой представлен на рисунке 5.

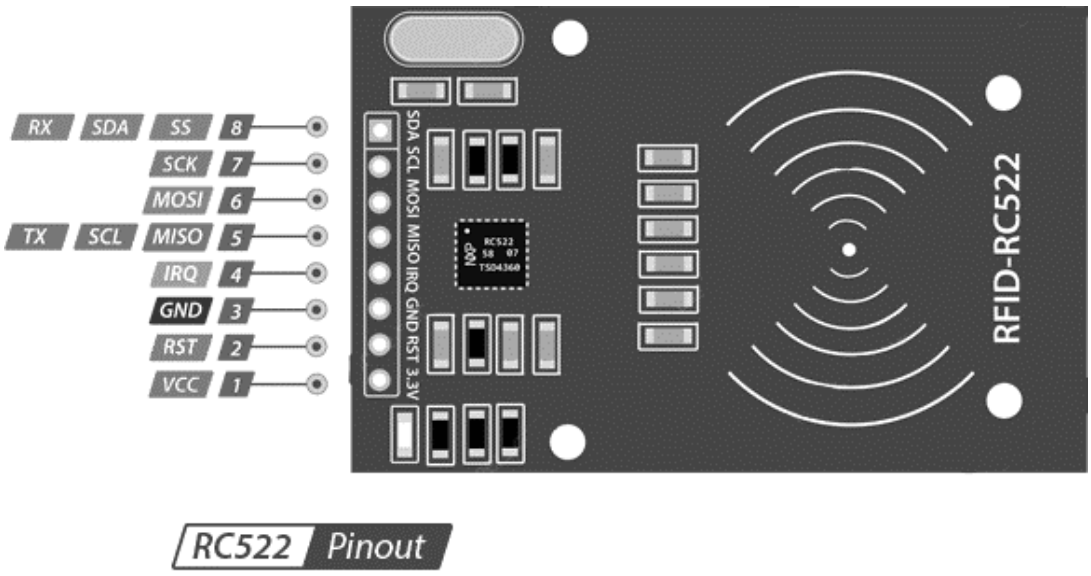

Рисунок 5 — Расположение пинов расширительной платы RFID-RC522

Опишем назначение выводов интерфейса SPI расширительной платы RFID-RC522:

1) VCC –питание 3.3 В;

2) RST – вывод для сброса;

3) GND – нулевой потенциал;

4) IRQ – вывод прерывания;

5) MISO – передача от slave к master;

6) MOSI – передача от master к slave;

7) SCK – сигнал синхронизации;

8) SDA – выбор ведомого.

Расширительная плата RFID-RC522 имеет следующие характеристики:

– напряжение питания: 3.3 В;

– потребляемый ток :13–26 мА;

– рабочая частота: 13.56 МГц;

– дальность считывания: 0–60 мм;

– интерфейс: SPI;

– скорость передачи: максимальная 10 МБит/с;

– размер: 40 мм х 60 мм;

Подключение платы RFID-RC522 к Arduino Nano происходит посредством проводов со штекером и разъемов на платах, изображенных на рисунках 6 и 7 соответственно. Подключение производится в соответствии с таблицей 4.

Рисунок 6 — Внешний вид проводов со штекером

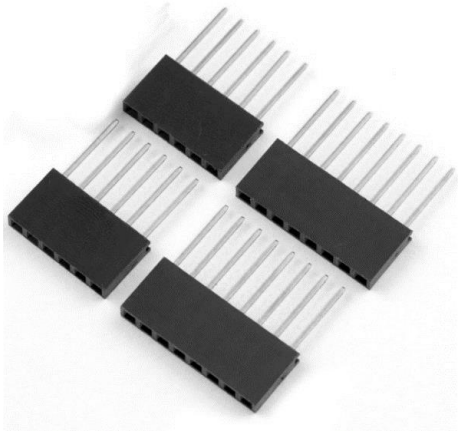

Рисунок 7 — Внешний вид разъемов на плате

Таблица 4 — Последовательность подключения пинов RFID-RC522 к Arduino Nano

RFID-RC522	Arduino Nano
RST	D9
SDA	D10
MOSI	D11
MISO	D12
SCK	D13
3.3 В	3.3 В
GND	GND

После установки разъемов в плату Arduino Nano и в расширительную плату RFID-RC522 можно приступать к подключению. Подключив платы согласно таблице 4, получается готовое к программированию устройство. Наглядная схема подключения показана на рисунке 8.

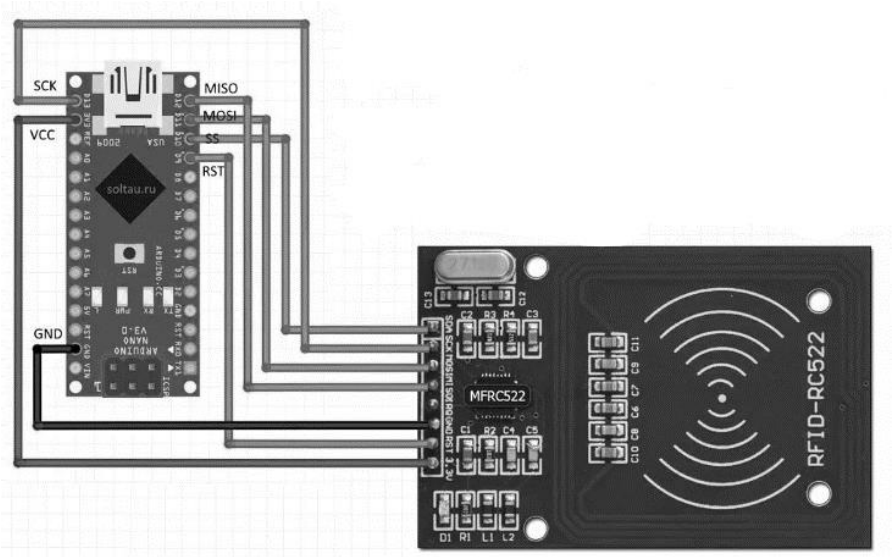

Рисунок 8 — Подключение расширительной платы RFID-RC522 к Arduino Nano

Для написания программного кода для платы Arduino Nano была выбрана среда разработки Arduino IDE, интерфейс которой представлен на рисунке 9. «Язык программирования на Arduino является упрощённой версией языка C++, вся необходимая информация (правила написания кода, синтаксис языка, основные команды) предоставлена в открытом доступе на официальном сайте

Arduino. Для работы с Arduino есть специальная среда «ArduinoIDE», работающая под Windows, MacOS и Linux.» [7, с.323].

Рисунок 9 — Интерфейс среды разработки Arduino IDE с фрагментом кода

Основными функциями в коде микроконтроллера будут Setup() и Loop(). Функция Setup() позволяет задать начальные настройки микроконтроллера, которые задаются при запуске. В начале нужно определить скорость, на которой будут передаваться данные по последовательному порту: 9600 БОД (= 9600 бит/с). После инициализации порта нужно инициализировать SPI интерфейс, который подключен к расширительной плате RFID-RC522. А также в конце стартовой настройки нужно инициализировать сам RFID считыватель. Следующая функция Loop() будет бесконечно исполняться пока не отключится питание микроконтроллера Arduino Nano. В этой функции выполняется проверка условия «Есть ли чип возле считывателя?». Работа отправки кода с чипа в последовательный порт показана на рисунке 10.

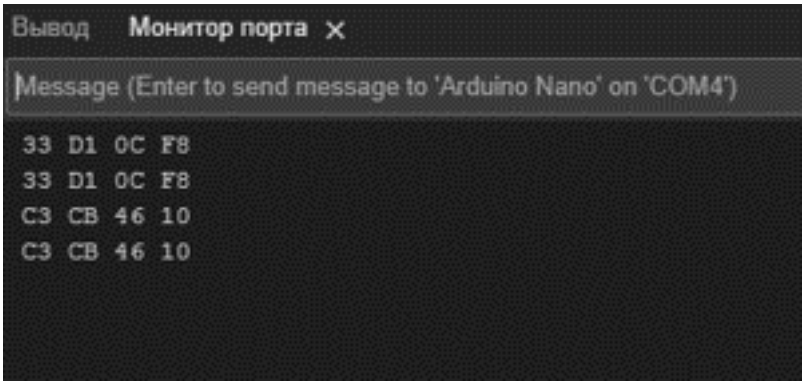

Рисунок 10 — Передача идентификаторов по последовательному порту

Данные, передаваемые по последовательному порту, получает десктопное приложение. Данные о проживающих в общежитии студентах и другая полезная информация хранится в базе данных MySQL.

2.3.2 Разработка программного кода десктопного приложения

Разработка программного кода десктопного приложения начинается с определения функций, которые должно реализовывать десктопное приложение. Основные функции, которые должно реализовывать программное обеспечение описаны в таблице 5.

Таблица 5 — Функции десктопного приложения

Название функции	**Действие функции**	**Описание функции**
Подключение к Arduino Nano	Установление соединения с Arduino Nano через USB-порт	Приложение устанавливает соединение при нажатии на кнопку «Запустить». Пользователь сможет вручную инициировать подключение или отключение. Статус подключения будет отображаться в интерфейсе («Не удалось подключиться к порту COM», «Все работает в штатном режиме. Хорошего дня!»)
Обработка данных	Извлечение информации событиях доступа	Приложение будет считывать данные с Arduino Nano, содержащие информацию о событиях доступа (ID пользователя). Данные будут сортироваться по времени (от старых к новым). Фильтрованные и отсортированные данные будут отображаться в таблице. Результаты поиска будут отображаться в таблице
Визуализация данных	Отображение информации о событиях доступа в виде списка	Отображение информации о событиях доступа в виде списка: Дата, Событие. Возможность просматривать статистику по каждому проживающему в общежитии. Возможность просматривать все события. Возможность просматривать кто из жильцов прямо сейчас находится в общежитии
Работа с базой данных MySQL	Подключение к базе данных MySQL	Приложение будет использовать библиотеку mysql.connector и pymysql для подключения и работы с базой данных MySQL. Пользователь сможет указать параметры подключения (адрес сервера, имя пользователя, пароль, имя базы данных). Загрузка информации о событиях доступа из базы данных: по запросу пользователя информация о событиях доступа будет загружаться из базы данных и отображаться в таблице

Написание программного кода, реализовывающего данные возможности начинается с импортирования всех нужных для работы приложения библиотек, а также макетов графического интерфейса, как продемонстрировано на рисунке 11.

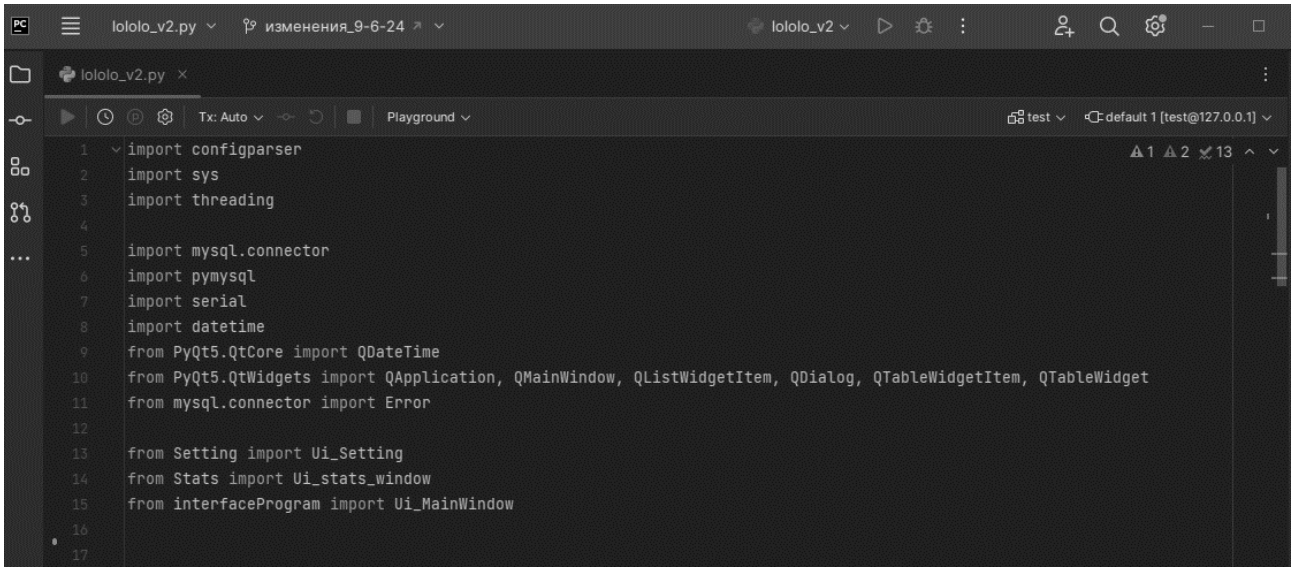

Рисунок 11 — Импортирование в программный код необходимых библиотек

После импортирования необходимых библиотек, можно приступить к написанию последовательности команд, которые будут выполняться при запуске программы. На рисунке 12 показано, что после запуска программы и выполнения простого условия, будет вызываться класс MyWindow. При вызове класса MyWindow выполняется функция инициализации, которая производит первоначальную настройку приложения, а также содержит в себе функции перехода к другим окнам и функции взаимодействия с интерфейсом.

Рисунок 12 — Описание условия при запуске программного кода

Программный код главного окна, без функций показан на рисунке 13. Полный программный код десктопного приложения расположен в приложении А.

```
2 usages
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)  # Инициализация главного окна
        self.ui.OnButt.setText("Запустить")  # Текст дял кнопки
        self.is_running = False  # Флаг запуска чтения порта
        self.can_add_to_list = False  # Флаг возможности добавления записей в QListWidget
        self.serial_port = None  # Флаг открытия порта
        self.ui.OnButt.clicked.connect(self.toggle_reading)  # Функция при нажатии на кнопку "Запустить"
        self.ui.Setting_Button.clicked.connect(self.open_settings_window)  # Функция при нажатии на кнопку "..."
        self.ui.stats_button.clicked.connect(self.open_stats_window)  # Функция при нажатии на кнопку "Статистика"
        self.stats_window = None  # Флаг окна статистики

        # Применить начальные настройки темы
        self.apply_theme()
```

Рисунок 13 — Программный код главного окна

При взаимодействии с графическим интерфейсом пользователь может переходить в окно настроек и окно статистики, поэтому для продолжения написания кода, необходимо создать графический интерфейс пользователя.

Для того чтобы приложение имело доступ к последовательному порту была создана функция start_reading. На рисунке 14 показано, что в функции происходит выбор порта, скорости обмена данными и задержка при подключении.

```
def start_reading(self):
    try:
        self.serial_port = serial.Serial('COM5', 9600, timeout=1)
        threading.Thread(target=self.read_from_serial, daemon=True).start()
        self.ui.listWidget.addItem("Все работает в штатном режиме. Хорошего дня!")
    except serial.SerialException:
        self.ui.listWidget.addItem("Не удалось подключиться к порту COM5")
```

Рисунок 14 — Программный код функции start_reading

Рабочее состояние программы отслеживается флагом is_running которое по умолчанию при запуске программы имеет значение False. При нажатии на кнопку запуск происходит вызов функции toggle_reading в котором проверяется условие

«программа остановлена или запущена». В зависимости от состояния флага is_running выбирается дальнейшее действие, «открыть порт и начать принимать данные с него или закрыть порт и подключение к базе данных.

Для подключения к базе данных использовалась функция mysql.connector.connect(host,user,password,database) внутри которой передавались параметры подключения к базе данных, каждый раз когда нужно было получить или отправить данные. Таким образом, когда пользователь подносил метку к считывателю, микроконтроллер отправлял считанный код по последовательному порту, программа принимала это значение, формировала запрос к базе данных, в котором был описан код метки, а также сравнивала значение с авторизованными метками в базе данных и выдавала в главном окне приложения разрешён вход или запрещён. Если человек находился в здании и вышел, тогда значение поля «Внутри» таблицы user_information, посредством запроса в базу данных, меняется с 1 на 0.

Для работы с окном настроек был создал класс SettingsDialog(QDialog), показанный на рисунке 15. В этом классе описаны стандартные настройки темы, находящиеся в файле config.ini, который создается при запуске программы. Он нужен для того, чтобы сохранялись настройки приложения.

```
class SettingsDialog(QDialog):

    def __init__(self, parent=None):
        super(SettingsDialog, self).__init__(parent)
        self.ui = Ui_Setting()
        self.ui.setupUi(self)  # Инициализация окна настроек
        self.ui.theme_combobox.currentIndexChanged.connect(self.change_theme)
        config = configparser.ConfigParser()
        config.read('config.ini')  # Функция чтения файла config.ini
        theme_type = config.get('Theme', 'theme', fallback='0')  # Получение переменных из файла config.ini
        background_color = config.get('Theme', 'Background_Color', fallback='#FFFFFF')  # Получение переменной из файла config.ini
        text_color = config.get('Theme', 'Text_Color', fallback='#000000')  # Получение переменной из файла config.ini
        self.ui.theme_combobox.setCurrentText("Светлая" if theme_type == '0' else "Темная")  # Получение переменной из файла config.ini
        self.apply_theme(theme_type, background_color, text_color)  # Вызов функции, которая применяет настройки темы
        MyWindow.apply_theme(parent)  # Определение родительского окна "Главного окна"
```

Рисунок 15 — Программный код класса SettingsDialog(QDialog)

Для работы окна статистики был создан класс StatsWindow(QMainWindow), показанный на рисунке 16, в котором описана работа фильтров статистики, а также выбора объекта фильтрации. Также как и в других окнах, вначале применяются настройки темы и кнопок управления.

```
class StatsWindow(QMainWindow):

    def __init__(self, parent=None):
        super(StatsWindow, self).__init__(parent)
        self.ui = Ui_stats_window()
        self.ui.setupUi(self)
        self.load_data_to_combobox()
        self.ui.pushButton.clicked.connect(self.clickonstats)

        config = configparser.ConfigParser()
        config.read('config.ini')
        theme_type = config.get('Theme', 'theme', fallback='0')
        background_color = config.get('Theme', 'Background_Color', fallback='#FFFFFF')
        text_color = config.get('Theme', 'Text_Color', fallback='#000000')
        self.apply_theme(theme_type, background_color, text_color)
```

Рисунок 16 — Программный код класса StatsWindow(QMainWindow)

В классе StatsWindow(QMainWindow) также созданы функции, которые реализуют возможность просмотра данных о посещениях. Для начала в элемент QComboBox загружаются информация из базы данных об элементах, по которым будет производиться фильтрация. На рисунке 17 показан программный код функции заполнения элемента QComboBox элементами из таблицы user_information.

```
def load_data_to_combobox(self):
    connection = pymysql.connect(host='localhost', user='admin', password='123qwe!@#QWE',
                                 database='test')

    try:
        with connection.cursor() as cursor:
            cursor.execute("SELECT FIM_user FROM user_information")
            result = cursor.fetchall()

            self.ui.comboBox.clear()  # Очистка ComboBox перед добавлением новых элементов

            for row in result:
                self.ui.comboBox.addItem(row[0])  # Добавление элементов из базы в combobox

    finally:
        connection.close()
```

Рисунок 17 — Заполнение элемента QComboBox элементами из таблицы user_information

После того как данные заполнились в список, можно выбрать по какому критерию нужно показать статистику. Для реализации функции показа статистики создана функция clickonstats, изображенная на рисунке 18, в которой описываются действия при выборе параметра фильтрации и нажатии на кнопку «Статистика».

```
try:
    with connection.cursor() as cursor:
        if selected_user == "Все":  # Параметр фильтрации "Все"
            query = "SELECT wi.Время, ui.FIM_user FROM work_information wi JOIN user_information ui ON wi.Код = ui.uid"
            cursor.execute(query)

        if selected_user == "Внутри":  # Параметр фильтрации "Внутри"
            query = "SELECT FIM_user FROM user_information where Внутри = 1"
            cursor.execute(query)

        if selected_user != "Все" and selected_user != "Внутри":   # Параметр фильтрации "*Пользователь*"
            query = ("SELECT Время, Код FROM work_information "
                     "WHERE Код = (SELECT uid FROM user_information WHERE FIM_user = %s)"
                     "ORDER BY Время DESC")
            cursor.execute(query, (selected_user,))

        result = cursor.fetchall()
        if selected_user == "Внутри":
            self.ui.statsTable.setRowCount(len(result))  # Устанавливаем количество строк
            self.ui.statsTable.setColumnCount(1)
            self.ui.statsTable.setHorizontalHeaderLabels(['Сейчас в здании общежития'])
            for row_index, row_data in enumerate(result):
                self.ui.statsTable.setItem(row_index, 0, QTableWidgetItem(str(row_data[0])))
        else:
            self.ui.statsTable.setRowCount(len(result))  # Устанавливаем количество строк
            self.ui.statsTable.setColumnCount(2)  # Устанавливаем количество столбцов
            self.ui.statsTable.setHorizontalHeaderLabels(
                ['Время', 'Пользователь'])  # Устанавливаем заголовки столбцов
            for row_index, row_data in enumerate(result):
                formatted_time = datetime.datetime.strptime(str(row_data[0]), "%Y-%m-%d %H:%M:%S").strftime(
                    "%d-%m-%Y %H:%M:%S")
                self.ui.statsTable.setItem(row_index, 0, QTableWidgetItem(formatted_time))
                self.ui.statsTable.setItem(row_index, 1, QTableWidgetItem(str(row_data[1])))

        self.ui.statsTable.resizeColumnsToContents()
finally:
    connection.close()
```

Рисунок 18 — Программный код функции clickonstats

Для эффективной работы с проектом были использованы репозитории Github, в которых хранятся этапы разработки программного кода, что позволило реализовать управление версиями десктопного приложения. На рисунке 19 показан скриншот репозитория последних изменений в программном коде десктопного приложения.

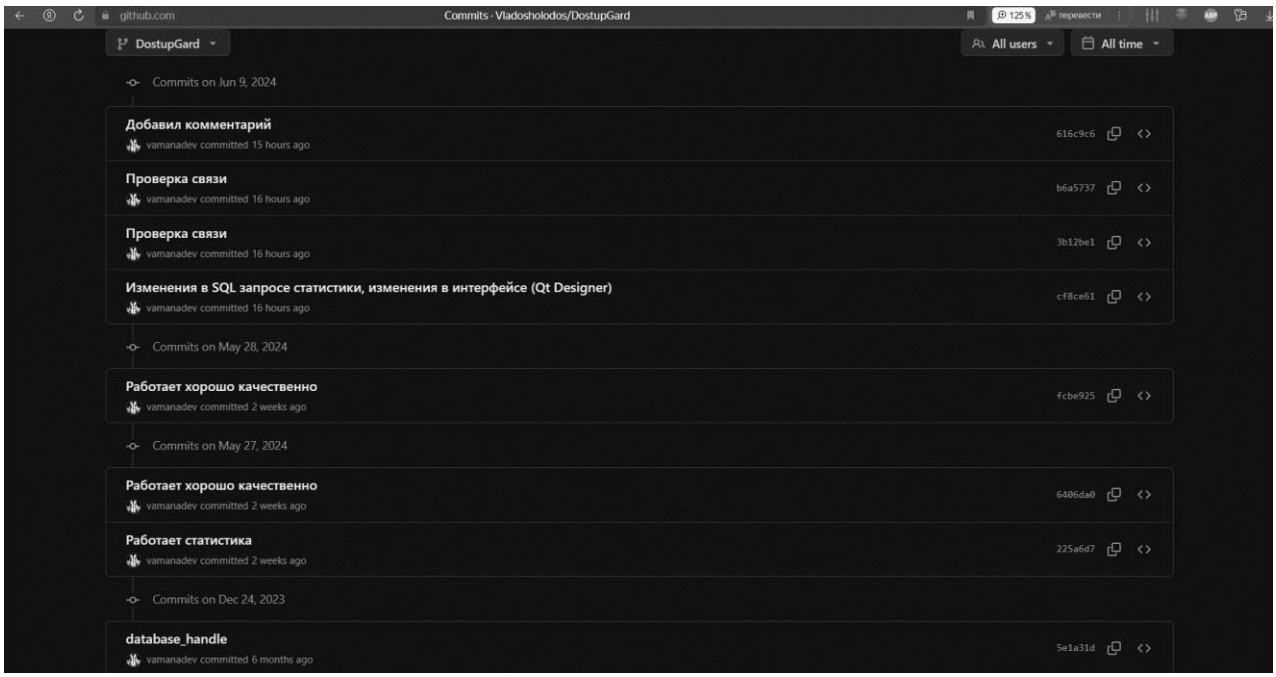

Рисунок 19 — Репозиторий с версиями десктопного приложения в Github

Рассмотрим создание графического интерфейса пользователя десктопного приложения. При открытии программы Qt Designer появляется стартовое окно, представленное на рисунке 20.

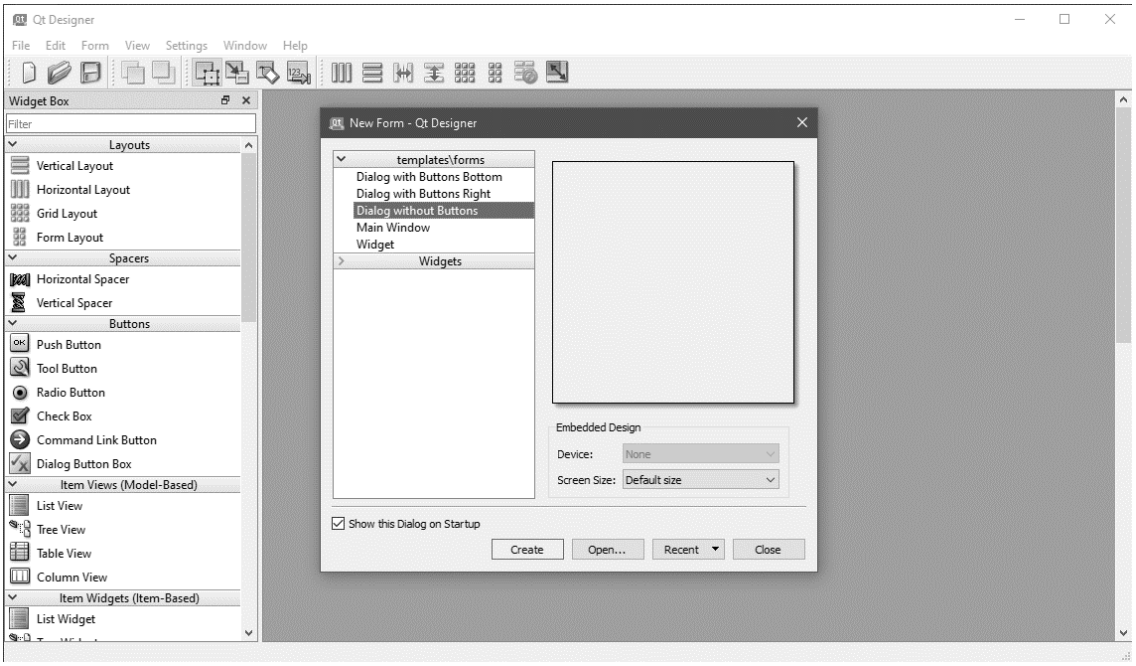

Рисунок 20 —Вид стартового окна в программе Qt Designer

Для того чтобы создать главное окно приложения нужно создать форму «Main Window». После того как создали главное окно, на него нужно

расположить нужные объекты, такие как QPushButton, QToolButton, QListWidget, QComboBox, QLabel, с которыми будет взаимодействовать пользователь. Очень важно задать понятное имя каждому объекту. Это нужно для дальнейшего использования в коде приложения. На рисунках 21–23 изображены формы главного окна, окна настроек и окна статистики десктопного приложения с добавленными элементами соответственно.

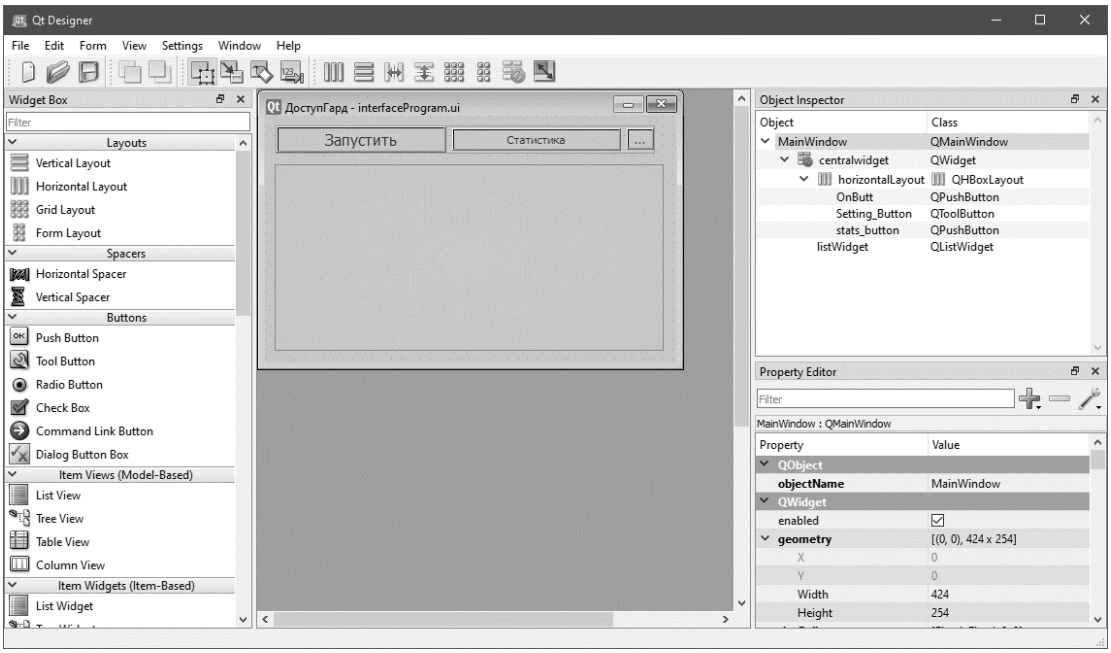

Рисунок 21 — Главное окно с добавленными элементами

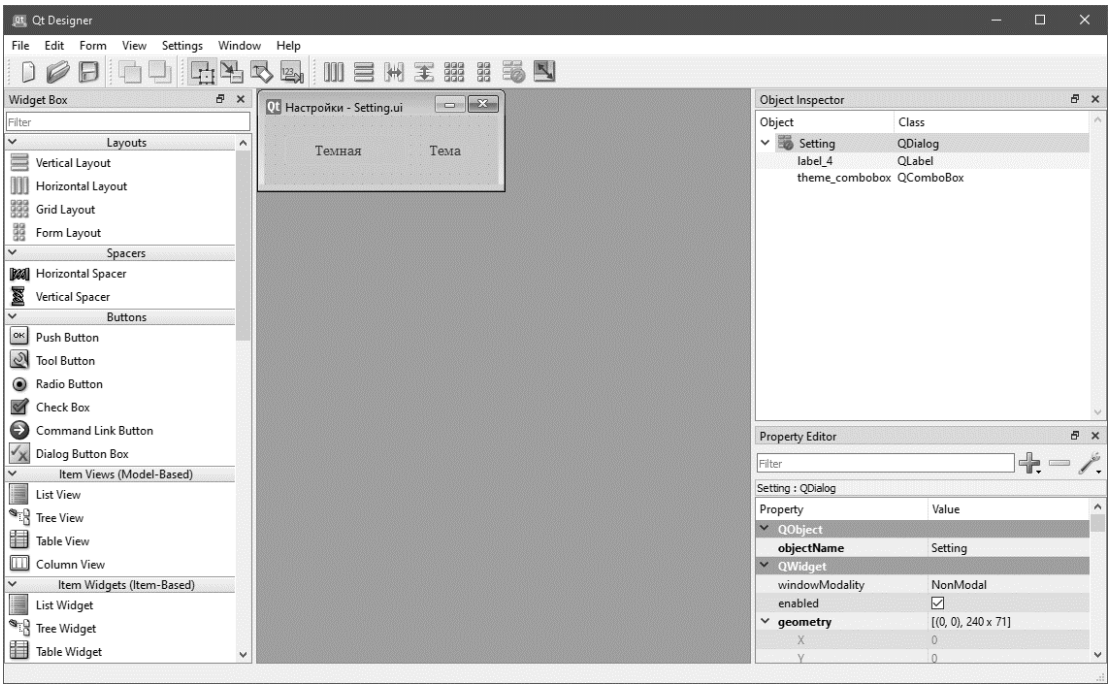

Рисунок 22 — Окно настроек с добавленными элементами

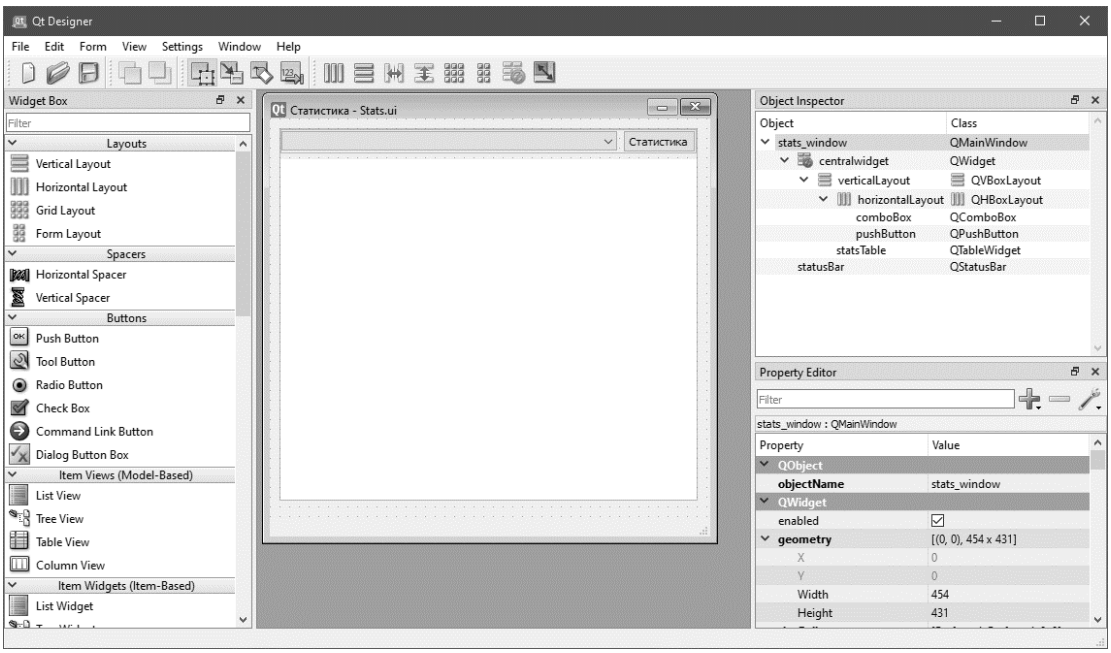

Рисунок 23 — Окно статистики с добавленными элементами

После того как готовы макеты графического интерфейса, необходимо их сохранить, а также конвертировать в другой формат, который необходим для работы с программным кодом. Для того, чтобы конвертировать макеты из расширения .ui в расширение .py необходимо с помощью командной строки выполнить команду «pyuic5 interfaceProgram.ui -o interfaceProgram.py», где «interfaceProgram» это названия файла, который нужно конвертировать. Для того, чтобы ускорить процесс создания и отладки были созданы bat файлы, в которых были заранее написанные команды. Список файлов изображен на рисунке 24. Содержимое файла «Обновить Setting.bat» изображено на рисунке 25.

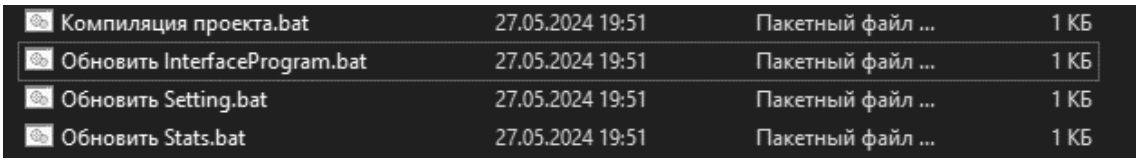

Рисунок 24 — Файлы, выполняющие команду конвертации расширения .ui в .py

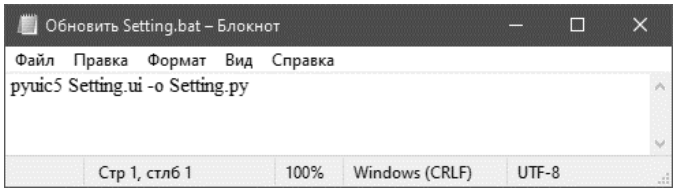

Рисунок 25 — Содержимое файла «Обновить Setting.bat»

Для дальнейшей работы создадим базу данных, которая будет отвечать всем требованиям, необходимым для реализации функциональных возможностей.

Для создания базы данных используется среда разработки MySQL Workbench. После установки и настройки сервера MySQL5.7, запускаем программу MySQL Workbench 8. Открывается главное окно, изображенное на рисунке 26, котором есть 3 вкладки слева, меню инструментов сверху и подключение к локальному серверу под надписью «MySQL Connections».

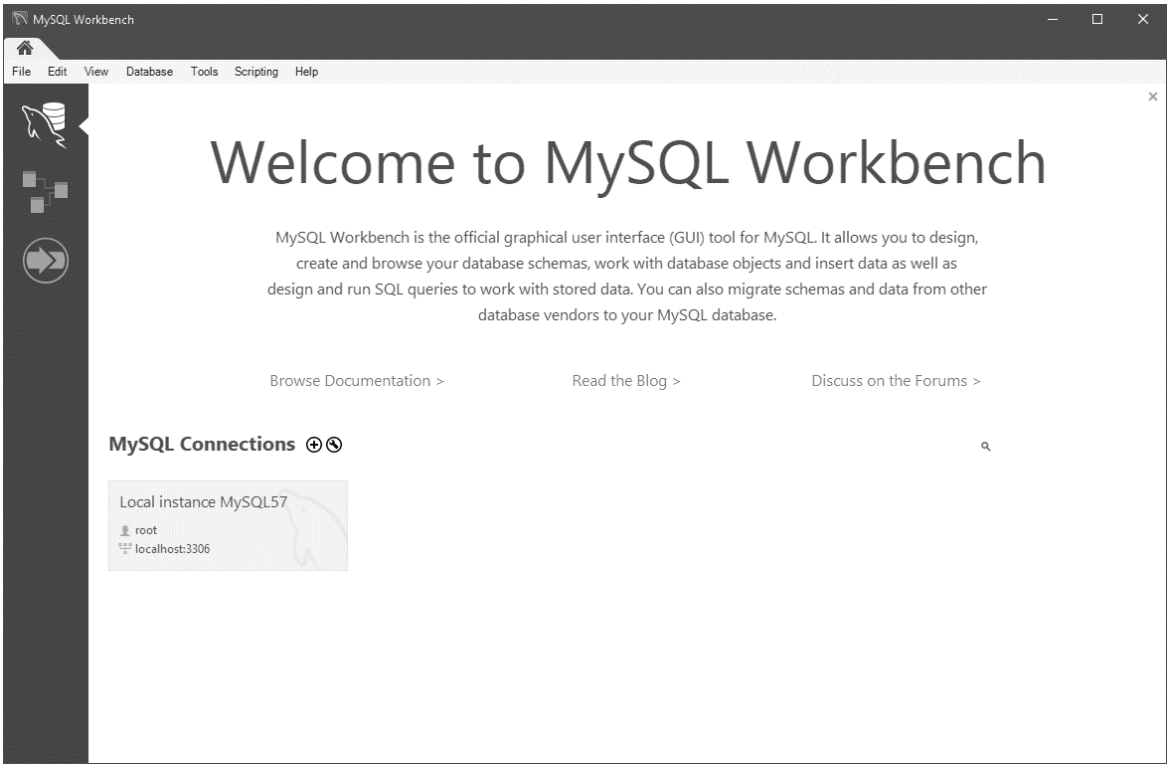

Рисунок 26 — Стартовое окно MySQL Workbench 8

После входа на локальный сервер, становятся видны стандартные и демонстрационные базы данных во вкладке schemas. Интерфейс локального сервера в MySQL Workbench 8 изображен на рисунке 27.

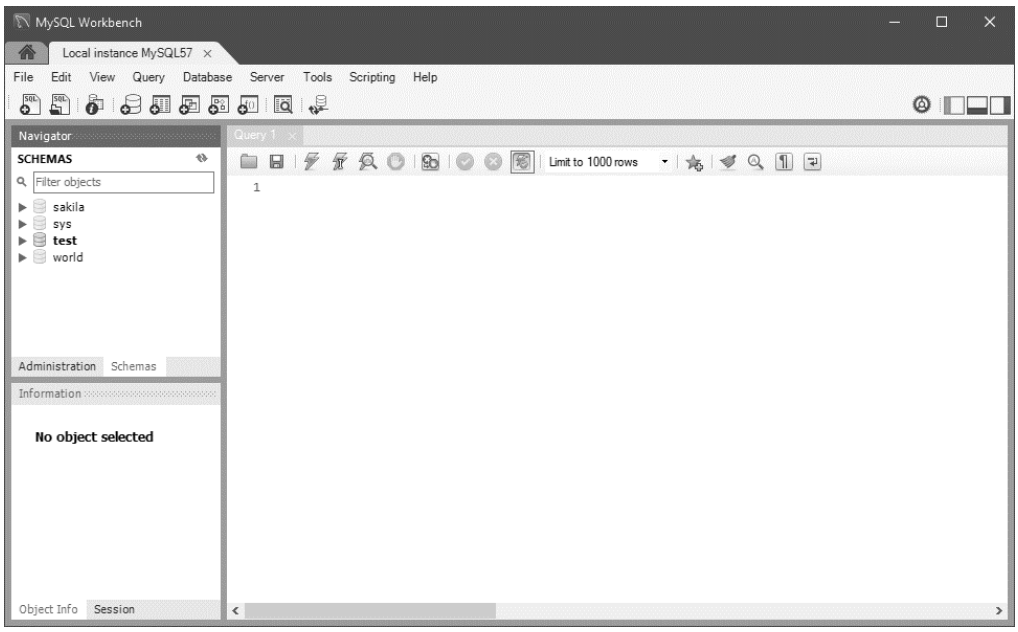

Рисунок 27 — Интерфейс локального сервера в MySQL Workbench

В базе данных test создаются 3 таблицы: work_information, user_information, authorized_users путем создания запроса с необходимыми настройками типов данных, количеством столбцов и первичным ключом, который будет уникальным идентификатором для связи между таблицами. Схема отношений и связей между таблицами изображена на рисунке 28.

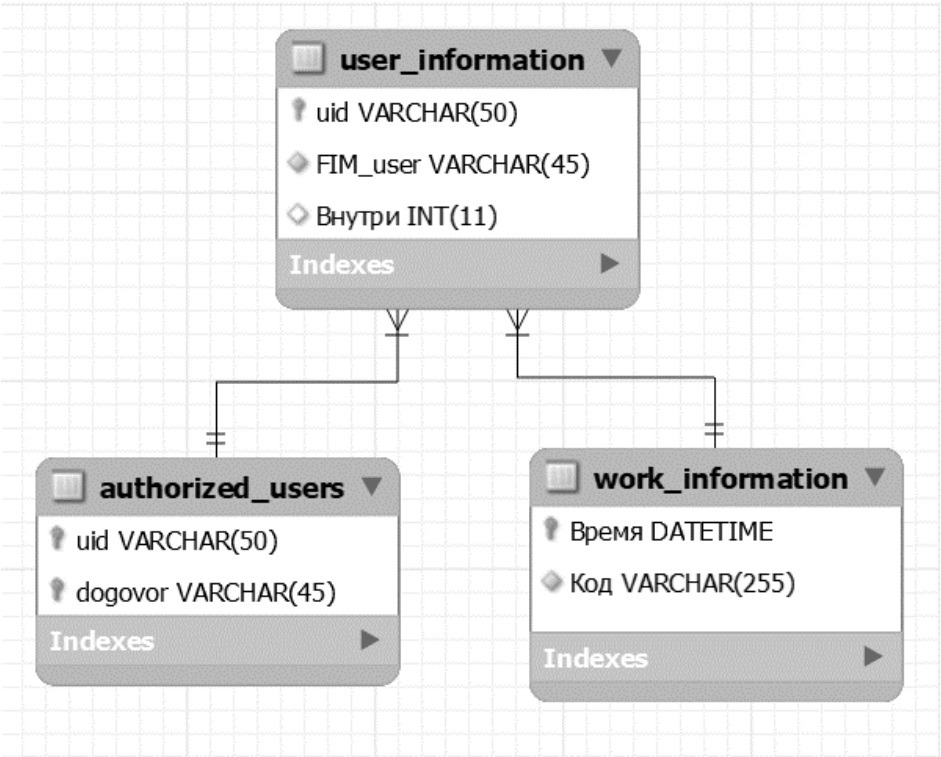

Рисунок 28 — Схема отношений между таблицами

На рисунке 29 изображен запрос создания таблицы authorized_users. Такими же запросами, но с небольшими изменениями создаются остальные таблицы.

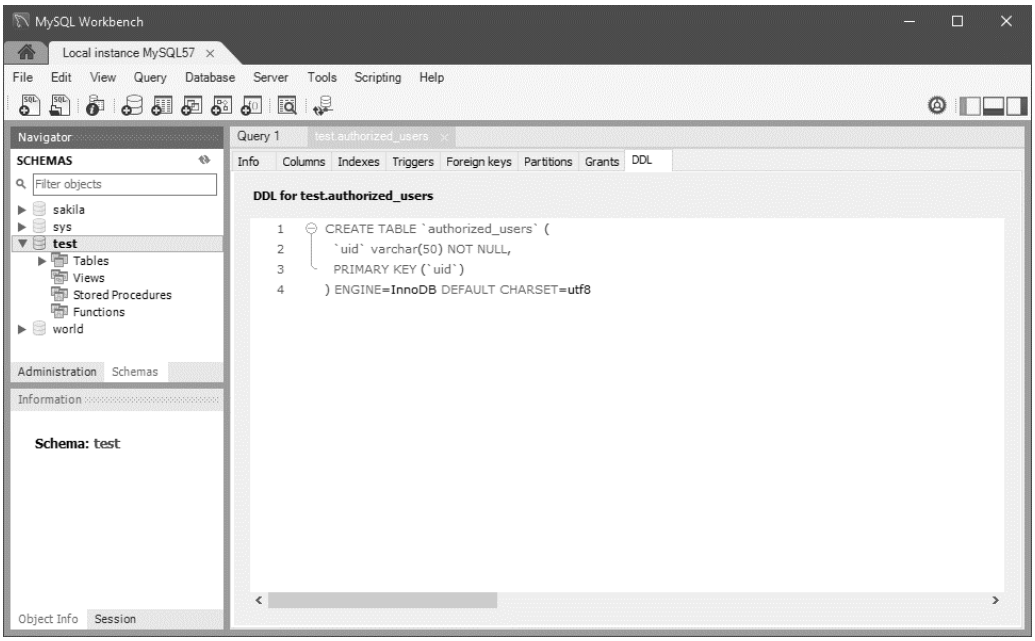

Рисунок 29 — Создание таблицы authorized_users

На рисунке 30 изображены список всех таблиц базы данных test.

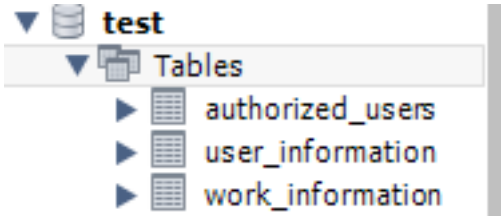

Рисунок 30 — Список таблиц базы данных test

Для работы с базой данных, необходимо добавить в таблицу user_information данные о принадлежащих пользователям уникальных кодах их меток. Также необходимо создать первые две строчки для фильтров статистики. На рисунке 31 показаны данные таблицы user_information.

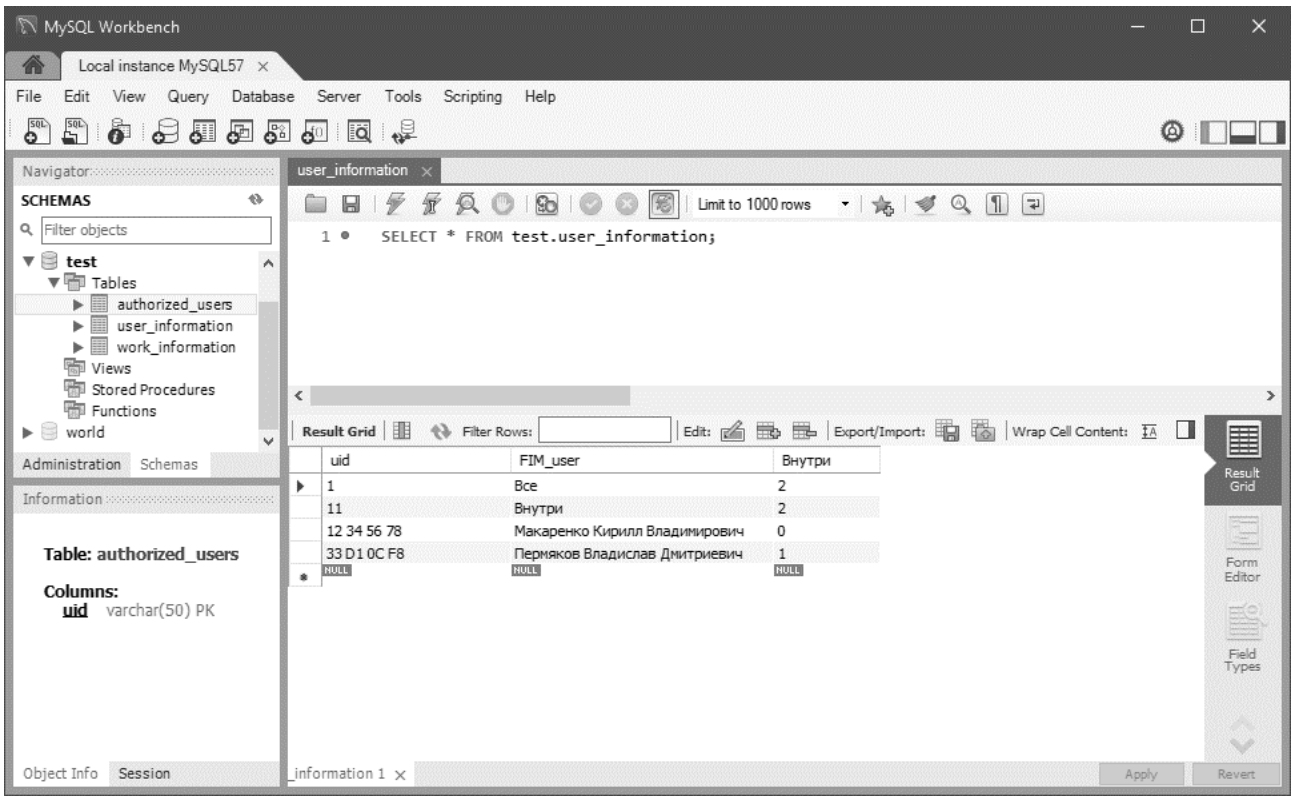

Рисунок 31 — Данные таблицы user_information

После создания локального сервера, на котором располагается база данных, можно продолжить разработку программного кода десктопного приложения. В таблице 6 описаны элементы графического интерфейса десктопного приложения.

Таблица 6 — Описание элементов графического интерфейса десктопного приложения.

Окно	**Элементы интерфейса**	**Описание**
ДоступГард	Кнопки	− «Запустить»: запускает процесс установления связи с Arduino Nano. При успешном подключении отображает сообщение: «Все работает в штатном режиме. Хорошего дня!». После нажатия надпись меняется на «Остановить», − «Статистика»: открывает окно статистики и данных из базы данных MySQL, − «...»: открывает окно настроек

Окончание таблицы 6

<table>
<tr><th>Окно</th><th>Элементы интерфейса</th><th colspan="4">Описание</th></tr>
<tr><td>ДоступГард</td><td>Список</td><td colspan="4">Отображает данные о времени (формат: ДД.ММ.ГГГГ ЧЧ:ММ:СС) и уникального идентификатора студента (uid), прошедшего через проход</td></tr>
<tr><td>Настройки</td><td>Выпадающий список для выбора темы приложения</td><td colspan="4">− «Светлая»: при выборе этого пункта оформление десктопного приложения сменяется на светлое (цвет текста: #000000 - черный, цвет фона: #A8D7E6 – светлый синий)

− «Темная»: при выборе этого пункта оформление десктопного приложения сменяется на темное (цвет текста: #FFFFFF - белый, цвет фона: #333333 – темный серый)</td></tr>
<tr><td rowspan="6">Статистика</td><td rowspan="3">Фильтр</td><td colspan="4">При выборе фильтра «Все» приложение показывает в таблице все уведомления, которые были в списке главного окна</td></tr>
<tr><td colspan="4">При выборе фильтра «Внутри» приложение показывает в таблице ФИО студентов, которые находятся в общежитии</td></tr>
<tr><td colspan="4">При выборе фильтра «*Пользователь из списка*» приложение в таблице показывает время прохода каждого студента по выбору из списка</td></tr>
<tr><td>Кнопки</td><td colspan="4">при нажатии обновляет таблицу и загружает данные из базы MySQL</td></tr>
<tr><td rowspan="4">Таблица</td><td>Фильтр</td><td>Название колонки 1</td><td colspan="2">Название колонки 2</td></tr>
<tr><td>Все</td><td>Дата и время</td><td colspan="2">Код студента</td></tr>
<tr><td>Внутри</td><td>ФИО студента</td><td colspan="2">-</td></tr>
<tr><td colspan="2">Пользователь из списка</td><td>Дата и время</td><td>Код студента</td></tr>
</table>

На рисунке 32 представлен вид главного окна десктопного приложения, на котором расположены элементы согласно таблице 6. В частности кнопка Запустить/Остановить, кнопка статистики, кнопка настроек и окно списка для отображения данных о времени в формате ДД.ММ.ГГГГ ЧЧ:ММ:СС, а также уникальный идентификатор студента. На рисунке 32 видно, что подключение с Arduino Nano произошло успешно.

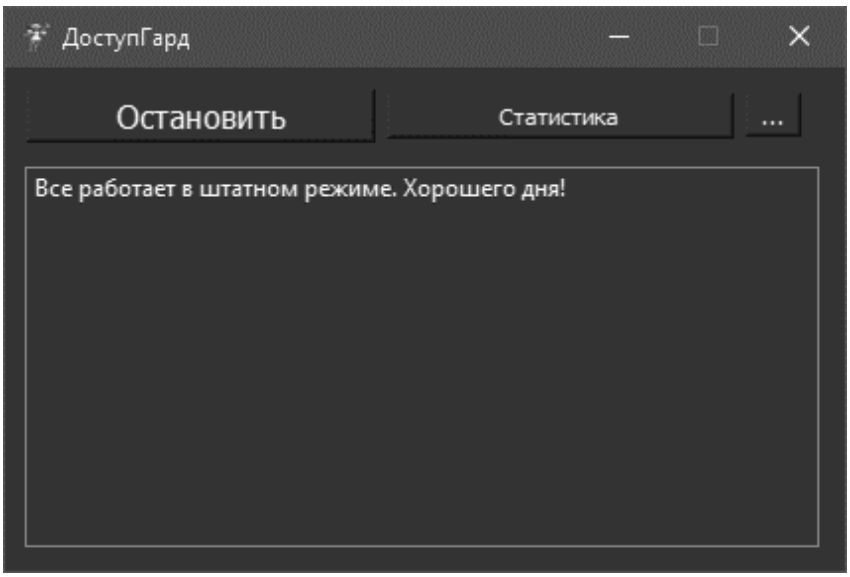

Рисунок 32 — Вид главного окна десктопного приложения

На рисунке 33 представлен вид начального окна статистики десктопного приложения, на котором расположены элементы согласно таблице 6.

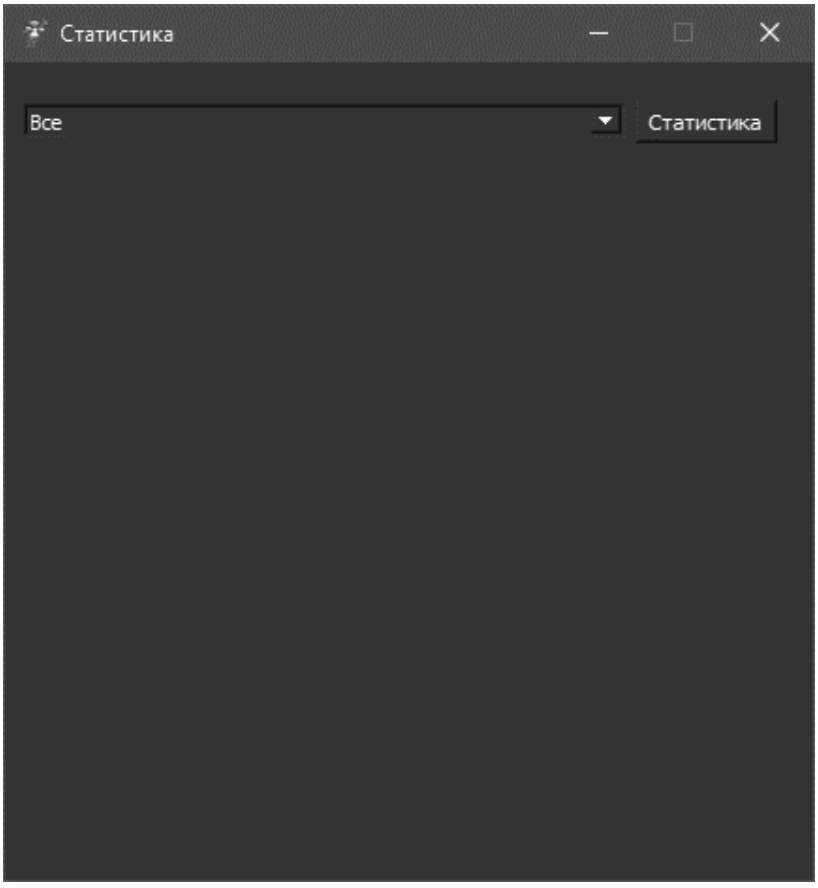

Рисунок 33 — Вид начального окна статистики десктопного приложения

На рисунке 34 представлен вид окна настроек десктопного приложения, на котором расположены элементы согласно таблице 6. Отметим, что тема применяется при перезапуске приложения.

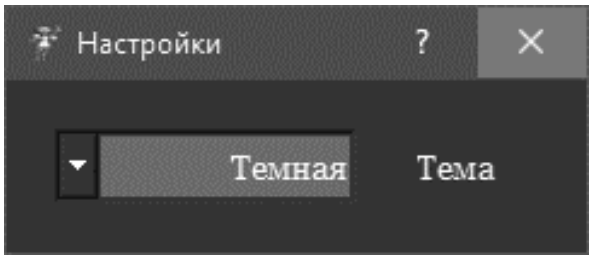

Рисунок 34 — Вид окна настроек десктопного приложения

На рисунке 35 представлено окно статистики десктопного приложения при выборе фильтра «Все», на котором показаны все записи регистрации посещений из базы данных.

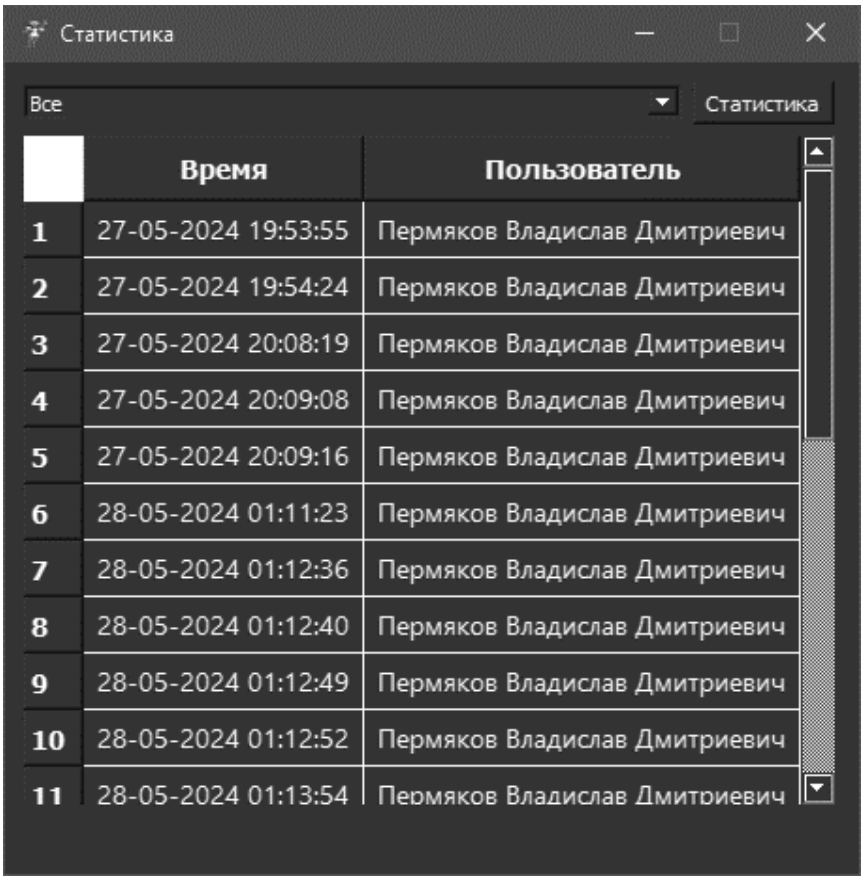

Рисунок 35 — Окно статистики десктопного приложения при выборе фильтра «Все»

На рисунке 36 представлено окно статистики десктопного приложения при выборе фильтра «Внутри», на котором показаны пользователи из базы данных, которые на данный момент находятся внутри здания.

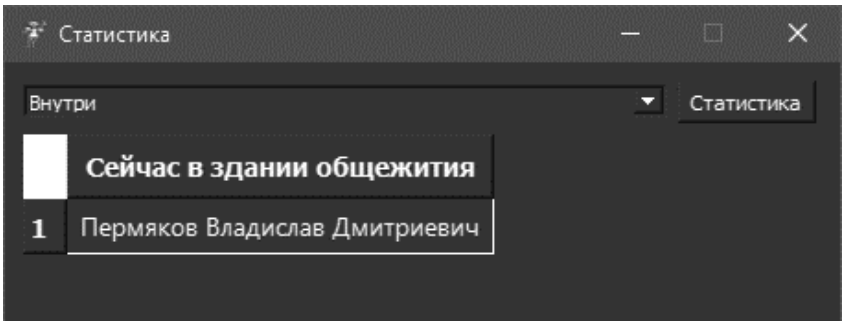

Рисунок 36 — Окно статистики десктопного приложения при выборе фильтра «Внутри»

На рисунке 37 представлено окно статистики десктопного приложения при выборе фильтра «Пермяков Владислав Дмитриевич», на котором показаны дата и время регистрации прохода, а также код пользователя, совершившего проход.

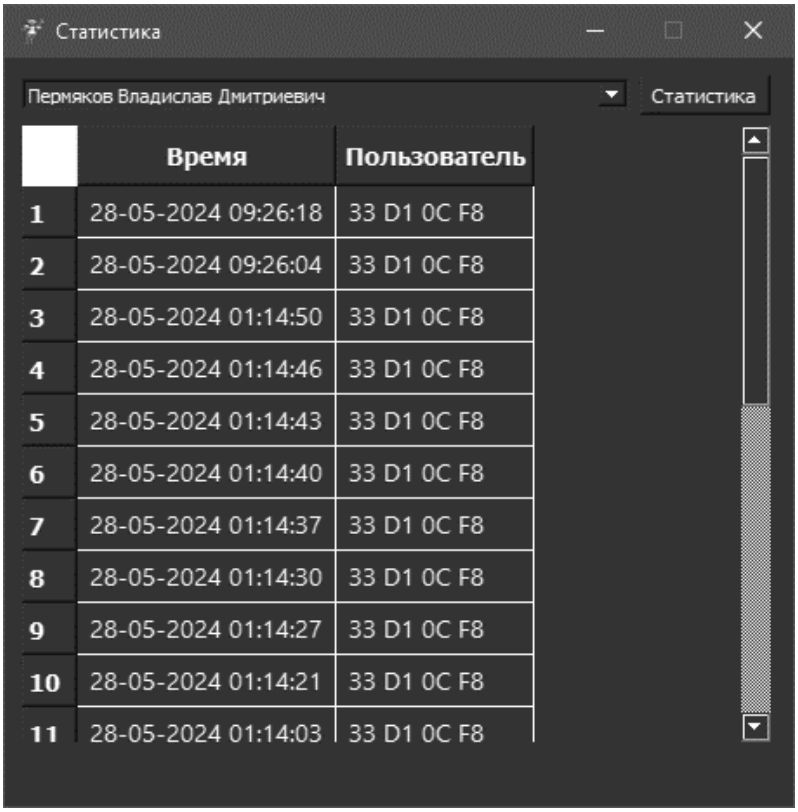

Рисунок 37 — Окно статистики десктопного приложения при выборе фильтра «Пермяков Владислав Дмитриевич»

В результате выполненной работы была спроектирована и разработана автоматизированная система автоматической регистрации посещения студентами общежития на базе микроконтроллеров Arduino. Система представляет собой десктопное приложение, способное работать с локальным сервером и базой данных. Цель создания системы, заключающаяся в автоматизации процесса регистрации посещений студентами общежития полностью реализована. Десктопное приложение имеет графический интерфейс пользователя с возможностью индивидуальных настроек и просмотра статистики посещений в соответствии с выбранными фильтрами.

ЗАКЛЮЧЕНИЕ

В рамках выполнения выпускной квалификационной работы были решены ключевые задачи, направленные на создание автоматизированной системы автоматической регистрации посещения студентами общежития. Проведенный анализ существующих методов и технологий автоматизации контроля и управления доступом позволил выделить требования к разрабатываемой системе и определить оптимальные решения для ее проектирования и последующей разработки.

На основе выделенных функциональных требований для разработки системы был выбран следующий набор инструментальных средств: PyCharm, MySQL Workbench 8, MySQL 5.7, Qt Designer, Arduino IDE, Github.

В ходе разработки были изучены основы языка Arduino C, необходимые для написания программного кода микроконтроллера Arduino Nano. Также актуализированы знания HTML, CSS, SQL, языков программирования C++, Python.

Автоматизированная система автоматической регистрации посещаемости общежития обладает рядом преимуществ, среди которых можно указать следующие: повышение эффективности процесса учета посещаемости за счет автоматизации, так как в отличие от ручного ведения журналов система автоматически фиксирует время и дату посещения каждого студента, исключая возможность человеческой ошибки и снижая административную нагрузку на персонал общежития; точность и достоверность данных о посещениях в реальном времени, достигаемая за счет поступления в базу данных информации, что позволяет мгновенно отслеживать присутствие студентов в общежитии; бесконтактная идентификация, реализованная за счет использования RFID-меток и обеспечивающая бесконтактный способ регистрации, что особенно актуально в условиях санитарных норм и требований; система масштабируема поскольку разработана с учетом возможности адаптировать и расширять систему для работы с большим количеством студентов или общежитий.

Результаты исследования были представлены на следующих научных мероприятиях:

– VII Всероссийская научно-практическая конференция «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 1–2 ноября 2023 г., 3-е место);

– VIII Всероссийский (IX Региональный) молодежный форум «Российское могущество прирастать будет Сибирью…» (г. Лесосибирск, ЛПИ – филиал СФУ, 14 декабря 2023 г., участие);

– III Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» в секции «Информационные технологии в образовании, науке и производстве» (г. Лесосибирск, ЛПИ – филиал СФУ, 9 апреля 2024 г., участие);

– III Всероссийский конкурс научных работ «Молодежный научный потенциал» (г. Лесосибирск, ЛПИ – филиал СФУ, 10 апреля 2024 г., 3 место);

– XX Международная научная конференция студентов, аспирантов и молодых ученых «Проспект Свободный – 2024» (г. Красноярск, ЛПИ – филиал СФУ, 15-20 апреля 2024 г., участие).

По результатам исследования приняты к публикации статьи:

1. Пермяков В. Д. / Проектирование и разработка автоматизированной системы регистрации посещений общежития на базе микроконтроллера Arduino / В. Д. Пермяков // Цифровые технологии в образовании, науке и технике: материалы XX Международной научной конференции студентов, аспирантов и молодых ученых «Проспект Свободный – 2024», г. Красноярск, / Сибирский федеральный университет. – Красноярск, 2024.

2. Пермяков В. Д. / Arduino или как сделать мир вокруг себя умнее / В. Д. Пермяков / В. Д. Пермяков // Информационные технологии в образовании, науке и производстве: материалы III Всероссийского молодежного научного форума «Современное педагогическое образование: теоретический и прикладной аспекты» / ЛПИ – филиал СФУ. – Лесосибирск, 2024.

В заключение можно отметить, что проектирование и разработка автоматизированной системы регистрации посещений общежития на базе микроконтроллеров Arduino являются важным шагом в улучшении процессов управления и контроля в образовательных учреждениях. Данная работа имеет значительный потенциал для дальнейшего развития и внедрения в различных сферах, требующих автоматизации учета и контроля доступа, что способствует повышению эффективности и безопасности данных процессов. Возможно добавление в систему исполнительных устройств (например, турникетов, электромагнитных замков и т. д.), ее интеграция с системой видеонаблюдения, подключение к беспроводной сети и удаленному серверу.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. База плагинов доступных к использованию в разработке приложений : сайт / Github. – 2024. – URL: https://github.com/topics/python-library (дата обращения: 10.02.2024).
2. БИТ.Общежитие : сайт. / Первый.Бит . – Москва, 2024. – URL: https://www.pulsar.ru/progs/1916/?ysclid=l3oufs53px (дата обращения: 10.06.2024).
3. Борсук, Н. А. Сравнительный анализ языков программирования Python и php / Н. А. Борсук , О. О. Козеева // Символ науки. – 2017. – № 4. – С. 33–36. – URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-yazykov-programmirovaniya-python-i-php (дата обращения: 10.06.2024).
4. Васильев, Д. А. Методические особенности изучения языка Python школьниками / Д. А. Васильев // Символ науки. – 2017. – № 1. – С. 170–172. – URL: https://cyberleninka.ru/article/n/metodicheskie-osobennosti-izucheniya-yazyka-python-shkolnikami (дата обращения: 10.06.2024).
5. Воробьева А. А. История развития программно-аппаратных средств защиты информации / А. А. Воробьева, И. С. Пантюхин. — СПб: Университет ИТМО, 2017. — 62 с. – URL: https://books.ifmo.ru/file/pdf/2188.pdf (дата обращения: 11.11.2023).
6. Глебов, С. И. Разработка файлового менеджера на языке Python / С. И. Глебов, И. А. Ивлиева // Форум молодых ученых. – 2019. – № 9 (37). – С. 100–121. – URL: https://cyberleninka.ru/article/n/razrabotka-faylovogo-menedzhera-na-yazyke-python (дата обращения: 10.06.2024).
7. Горбунов, Н. А., Основные возможности программируемой платы Arduino / Н. А. Горбунов, Е. Д. Горбунова // Теория и практика современной науки. – 2020. – № 2 (56). – С. 323–326. – URL: https://cyberleninka.ru/article/n/osnovnye-vozmozhnosti-programmiruemoy-platy-arduino (дата обращения: 10.04.2024).
8. ГОСТ 34.321-96. Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными : межгосударственный

стандарт : официальное издание : Принят Межгосударственным Советом по стандартизации, метрологии и сертификации (протокол №10 от 3 октября 1996 г.) : дата введения: 2001-07-01 : дата актуализации текста : дата актуализации текста: 06.04.2015. – Текст: электронный // Юридическая фирма «Интернет и право» сайт. – URL: https://internet-law.ru/gosts/gost/6808/ (дата обращения 21.01.2024).

9. ГОСТ Р 58485-2019. Оказание охранных услуг на объектах дошкольных, общеобразовательных и профессиональных образовательных организаций. : национальный стандарт Российской федерации : издание официальное: утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 9 августа 2019 г. № 492-ст. дата введения: 2019-08-9./ разработан Рабочей группой из представителей Саморегулируемой организации «Ассоциация предприятий безопасности «Школа без опасности» (СРО «Ассоциация «Школа без опасности»), Общероссийского отраслевого объединения работодателей в сфере охраны и безопасности «Федеральный координационный центр руководителей охранных структур» (ФКЦ РОС). Ассоциации ветеранов органов государственной охраны «Девятичи» (Ассоциация «Девятичи»), Общества с ограниченной ответственностью «Эталон» (ООО «Эталон»). – Москва. – 2024. – URL: https://clck.ru/3BMSLF (дата обращения 28.02.2024).
10. ГОСТ Р 59548-2022. Защита информации. Регистрация событий безопасности: национальный стандарт Российской Федерации : национальный стандарт Российской федерации: издание официальное : утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 13 января 2022 г. № 2-ст. : дата введения: 2022-01-13. / Юридическая фирма «Интернет и право» : сайт. – 2024. – URL: https://internet-law.ru/gosts/gost/77355/ (дата обращения 26.02.2024).
11. ГОСТ Р 59853-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения : национальный стандарт Российской Федерации : издание

официальное. – Текст: электронный // URL: https://clck.ru/3BMTQP (дата обращения 11. 12.2023).

12. ГОСТ Р ИСО 9000-2015. Системы менеджмента качества. Основные положения и словарь : национальный стандарт Российской Федерации : утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 28 сентября 2015 г. № 1390-ст. : дата введения: 2015-09-28. – Текст: электронный // сайт. – URL: https://clck.ru/3BGysx (дата обращения 21.02.2024).
13. ГОСТ 19.101-77. Единая система программной документации: межгосударственный стандарт : утвержден и введен в действие Постановлением Государственного комитета стандартов Совета Министров СССР от 20 мая 1977 г. N 1268 от 01-01-80 – Текст: электронный // URL: https://clck.ru/3BHx9M (дата обращения 11.06.2024).
14. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы: межгосударственный стандарт : утвержден и введен в действие Постановлением Государственного комитета СССР по стандартам от 24 марта 1989 г. No 661 : дата введения 1990-01-01 : переиздание июнь 2009 г. / разработан и внесен Государственным комитетом СССР по стандартам, Министерством приборостроения, средств автоматизации и систем управления СССР. – Москва : Стандартинформ. –2009. – Текст: электронный // Электронный фонт правовых и нормативно-технических документов сайт. – URL: https://docs.cntd.ru/document/1200006924 (дата обращения 29.05.2024).
15. ГОСТ Р 51241-2008. Средства и системы контроля и управления доступом. Классификация. Общие технические требования. Методы испытаний : национальный стандарт Российской Федерации : официальное издание. – Текст: электронный // URL: https://clck.ru/3BMTcx (дата обращения 13.12.2023).

16. Журавлев, А. Е. Корпоративные информационные системы. Администрирование сетевого домена : Учебное пособие для СПО / А. Е. Журавлев, А. В. Макшанов, Л. Н. Тындыкарь. – Санкт-Петербург: Лань, 2021. – 172 с. – ISBN 978-5-8114-8417-1.
17. Исси Коэн, Лазаро. Полный справочник по HTML, CSS и JavaScript / Лазаро Исси Коэн, Джозеф Исси Коэн ; [пер. с англ. А. Ливадного]. – Москва : ЭКОМ, 2007. – 1166 с. – ISBN 978-5-9790-0009-1.
18. Копытова, М. А. Актуальность языка программирования Python / М. А. Копытова // Экономика и социум. – 2016. – № 10 (29). – С. 943–945. – URL: https://cyberleninka.ru/article/n/aktualnost-yazyka-programmirovaniya-python (дата обращения: 18.06.2024).
19. Красочкин, С. Г. Изображения и визуализация данных в Python / Научный журнал. – 2022. – № 2 (64). – URL: https://clck.ru/3BLRJZ (дата обращения: 18.06.2024).
20. Мехта, Ч. MySQL 8 для больших данных / Ч. Мехта, Ш. Чаллавала, Д. Лакхатария. – Москва: ДМК Пресс. – 2017. – 228 с. – ISBN 978-5-97060-653-7.
21. Официальный сайт компании ООО «Прософт-Биометрикс»: официальный сайт / Прософт-биоматрикс. – 2024. – URL: https://bio-smart.ru/ (дата обращения 14.06.2024).
22. Официальный сайт компании ООО «ТЕХНО МСК»: сайт / Техно-мск. – 2024. – URL: https://mosmt-t.ru/catalog/access-control/ (дата обращения: 13.06.2024).
23. Перевозникова, О. С Реализация информационно-технологических проектов по подготовке информационной системы в рамках выполнения выпускной квалификационной работы / О. С. Перевозникова, И. В. Шимов // Актуальные вопросы преподавания математики. информатики и информационных технологий : межвузовский сборник научных работ – Екатеринбург. – 2022. – С. 308-313.
24. Петров, А. В. Развитие систем контроля и управления доступом // Вестник Уральского государственного университета. Серия: Экономика и управление.

– 2008. – № 3. – С. 123–132. – URL: https://clck.ru/3BFyc9 (дата обращения: 12.11.2023).

25. Рейтинг самых используемых языков программирования за 2023 год по версии крупнейшей социальной сети для разработчиков: сайт / Github. – 2024. – URL: https://github.blog/2023-11-08-the-state-of-open-source-and-ai/ (дата обращения: 11.06.2024).
26. Робсон, Э. Изучаем HTML, XHTML и CSS / Э Робсон, Э Фримен. – 2-е изд. – Санкт-Петербург : Питер, 2019. – 720 с. – ISBN: 978-5-4461-1247-0.
27. Российская Федерация. Законы. О коммерческой тайне : Федеральный закон № 98-ФЗ : [принят Государственной думой 8 июля 2006 года : одобрен Советом Федерации 14 июля 2006 года]. – URL: https://clck.ru/3BMLTD. (дата обращения: 13.04.2024).
28. Российская Федерация. Законы. О персональных данных : Федеральный закон № 152-ФЗ : [принят Государственной думой 9 июля 2004 года : одобрен Советом Федерации 15 июля 2004 года // Официальный интернет-портал правовой информации. – URL: https://clck.ru/3BMaRA (дата обращения: 12.04.2024).
29. Российская Федерация. Законы. Об информации, информационных технологиях и о защите информации : Федеральный закон № 149-ФЗ : [принят Государственной думой 8 июля 2006 года : одобрен Советом Федерации 14 июля 2006 года] // Официальный интернет-портал правовой информации. – URL: http://pravo.gov.ru/proxy/ips/?docbody&nd=102108264 (дата обращения: 11.04.2024).
30. Студенческие общежития // Лесосибирский педагогический институт – филиал Сибирского федерального университета : официальный сайт. – 2024. – URL: https://lpi.sfu-kras.ru/node/506 (дата обращения: 18.06.2024).
31. Ткаченко, В. MySQL по максимуму : оптимизация, репликация, резервное копирование : 16+ / Б. Шварц, П. Зайцев, В. Ткаченко ; перевел с английского А. Колышкин. – 3-е изд. – Санкт-Петербург : Питер ; Минск : Питер, 2020. – 864 с. – ISBN 978-5-4461-0696-7.

32. Управляйте общежитием легко и просто вместе с «БИТ.Общежитие : сайт. / Первый.Бит . – Москва, 2024. – URL: https://clck.ru/3BLU6g (дата обращения: 11.06.2024).
33. Усольцев А. А : Информационные системы в экономике: Конспект лекций, 2009. 69 с. – URL: https://clck.ru/3BMRXS (дата обращения: 12.02.2024).
34. Arduino IDE : официальный сайт / Arduino IDE. – 2024. – URL: https://www.arduino.cc/en/software (дата обращения: 15.11.2023).
35. C++: сайт / Свободная энциклопедия. – URL: https://clck.ru/3BMMMk (дата обращения: 09.04.2024).
36. HTML5 BOOK : сайт. / HTML5BOOK.RU. – 2024. – URL: https://html5book.ru/ (дата обращения: 04.06.2024).
37. MySQL : сайт / MySQL. – 2024. – URL: https://www.mysql.com/ (дата обращения: 12.06.2024).
38. MySQL Installer (Archived Versions) : официальный сайт – / MySQL Product Archives. – URL: https://downloads.mysql.com/archives/installer/ (дата обращения: 21.02.2022).
39. MySQL Workbench : сайт / Свободная энциклопедия. — URL: https://ru.wikipedia.org/wiki/MySQL_Workbench (дата обращения: 03.06.2024).
40. MySQL Workbench: визуальный дизайн базы данных : сайт / MySQL. 2024. – URL: https://www.mysql.com/products/workbench/design/ (дата обращения: 01.06.2024).
41. PyCharm IDE : официальный сайт / Jetbrains. – 2024. –URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 11.06.2024).
42. Python : сайт / Свободная энциклопедия. – 2024. – URL: https://ru.wikipedia.org/w/index.php?title=Python&stable=1 (дата обращения: 11.06.2024).

ПРИЛОЖЕНИЕ А

Программный код основной программы

```
import configparser
import sys
import threading
import mysql.connector
import pymysql
import serial
import datetime
from PyQt5.QtCore import QDateTime
from PyQt5.QtWidgets import QApplication, QMainWindow, QListWidgetItem, QDialog, QTableWidgetItem, QTableWidget
from mysql.connector import Error
from Setting import Ui_Setting
from Stats import Ui_stats_window
from interfaceProgram import Ui_MainWindow

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)  # Инициализация главного окна
        self.ui.OnButt.setText("Запустить")  # Текст дял кнопки
        self.is_running = False  # Флаг запуска чтения порта
        self.can_add_to_list = False  # Флаг возможности добавления записей в QListWidget
        self.serial_port = None  # Флаг открытия порта
        self.ui.OnButt.clicked.connect(self.toggle_reading)  # Функция при нажатии на кнопку "Запустить"
```

```
        self.ui.Setting_Button.clicked.connect(self.open_settings_window)  # Функция при нажатии на кнопку "..."
        self.ui.stats_button.clicked.connect(self.open_stats_window)  # Функция при нажатии на кнопку "Статистика"
        self.stats_window = None  # Флаг окна статистики
        # Применить начальные настройки темы
        self.apply_theme()
    def open_settings_window(self):
        settings_dialog = SettingsDialog(self)
        settings_dialog.exec_()
    def open_stats_window(self):
        stats_window = StatsWindow(self)
        stats_window.show()
    def toggle_reading(self):
        self.is_running = not self.is_running
        if self.is_running:
            self.ui.OnButt.setText("Остановить")
            self.ui.listWidget.clear()
            self.can_add_to_list = True
            self.start_reading()
        else:
            self.ui.OnButt.setText("Запустить")
            self.stop_reading()

    def start_reading(self):
        try:
            self.serial_port = serial.Serial('COM5', 9600, timeout=1)
            threading.Thread(target=self.read_from_serial, daemon=True).start()
            self.ui.listWidget.addItem("Все работает в штатном режиме. Хорошего дня!")
```

```
        except serial.SerialException:
            self.ui.listWidget.addItem("Не удалось подключиться к порту COM5")
    def stop_reading(self):
        self.can_add_to_list = False
        if self.serial_port and self.serial_port.is_open:
            self.serial_port.close()
    def read_from_serial(self):
        while self.is_running and self.serial_port and self.serial_port.is_open:
            line = self.serial_port.readline().decode().strip()
            if line and self.can_add_to_list:
                self.add_to_list(line)
                # Добавляем данные в базу данных при каждом событии
                add_data_to_database(line)
                print(line)
    def add_to_list(self, data):
        current_time = QDateTime.currentDateTime().toString("yyyy-MM-dd
HH:mm:ss")
        label_info = f"Событие: {data}, Время: {current_time}"
        list_item = QListWidgetItem(label_info)
        self.ui.listWidget.addItem(list_item)
        self.ui.listWidget.addItem("")
    def apply_theme(self):
        config = configparser.ConfigParser()
        config.read('config.ini')
        index = config.get('Theme', 'index', fallback='0')
        background_color = config.get('Theme', 'background_color',
fallback='#FFFFFF')
        text_color = config.get('Theme', 'text_color', fallback='#000000')
        self.setStyleSheet(f"background-color: {background_color}; color:
{text_color};")
```

```
        self.ui.listWidget.setStyleSheet(f"background-color: {background_color}; color: {text_color};")
        self.ui.OnButt.setStyleSheet(f"background-color: {background_color}; color: {text_color};")
        self.ui.Setting_Button.setStyleSheet(f"background-color: {background_color}; color: {text_color};")
def add_data_to_database(data):
    try:
        # Подключение к базе данных
        connection = mysql.connector.connect(
            host='localhost',
            user='admin',
            password='123qwe!@#QWE',
            database='test'  # Название базы данных)
        if connection.is_connected():
            cursor = connection.cursor()
            # SQL-запрос для вставки данных
            insert_query = "INSERT INTO work_information (Время, Код) VALUES (%s, %s)"
            current_time = QDateTime.currentDateTime().toString("yyyy-MM-dd HH:mm:ss")
            data_to_insert = (current_time, data)
            # Выполнение запроса
            cursor.execute(insert_query, data_to_insert)
            connection.commit()
            print("Данные успешно добавлены в базу данных")
            # Получение текущего значения Внутри
            uid = data
            select_in_value_query = "SELECT Внутри FROM user_information WHERE uid = %s"
```

```
            cursor.execute(select_in_value_query, (uid,))
            current_in_value = cursor.fetchone()[0]
            # Вычисление нового значения "in"
            if current_in_value == 1:
                new_in_value = 0
            if current_in_value == 0:
                new_in_value = 1
            # Обновление значения "in"
            update_in_value_query = "UPDATE user_information SET Внутри = %s
WHERE uid = %s"
            cursor.execute(update_in_value_query, (new_in_value, uid))
            connection.commit()
    except Error as e:
        print(f"Ошибка: {e}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Подключение к базе данных закрыто")
class SettingsDialog(QDialog):
    def __init__(self, parent=None):
        super(SettingsDialog, self).__init__(parent)
        self.ui = Ui_Setting()
        self.ui.setupUi(self)  # Инициализация окна настроек
        self.ui.theme_combobox.currentIndexChanged.connect(self.change_theme)
        config = configparser.ConfigParser()
        config.read('config.ini')  # Функция чтения файла config.ini
        theme_type = config.get('Theme', 'theme', fallback='0')  # Получение
переменных из файла config.ini
```

```
        background_color = config.get('Theme', 'Background_Color', fallback='#FFFFFF')  # Получение переменной из файла config.ini
        text_color = config.get('Theme', 'Text_Color', fallback='#000000')  # Получение переменной из файла config.ini
        self.ui.theme_combobox.setCurrentText("Светлая" if theme_type == '0' else "Темная")  # Получение переменной из файла config.ini
        self.apply_theme(theme_type, background_color, text_color)  # Вызов функции, которая применяет настройки темы
        MyWindow.apply_theme(parent)  # Определение родительского окна "Главного окна"
    def change_theme(self, index):
        theme = '0' if index == 0 else '1'
        background_color = "#A8D7E6" if theme == '0' else '#333333'
        text_color = '#000000' if theme == '0' else '#FFFFFF'
        self.apply_theme(theme, background_color, text_color)
    def apply_theme(self, theme, background_color, text_color):
        theme = theme
        background_color = background_color
        text_color = text_color
        # Применить цветовые настройки к окну настроек
        self.setStyleSheet(f"background-color: {background_color}; color: {text_color};")
        self.ui.theme_combobox.setStyleSheet(f"background-color: {background_color}; color: {text_color};")
        self.ui.Language_Box.setStyleSheet(f"background-color: {background_color}; color: {text_color};")
        theme = "0" if theme == '0' else "1"
        # Сохранить настройки в файл config.ini
        config = configparser.ConfigParser()
        config['Theme'] = {
```

```
            'theme': theme,
            'Background_Color': background_color,
            'Text_Color': text_color}
        with open('config.ini', 'w') as configfile:
            config.write(configfile)
# Статистика-----------------------------------------------
class StatsWindow(QMainWindow):
    def __init__(self, parent=None):
        super(StatsWindow, self).__init__(parent)
        self.ui = Ui_stats_window()
        self.ui.setupUi(self)
        self.load_data_to_combobox()
        self.ui.pushButton.clicked.connect(self.clickonstats)
        config = configparser.ConfigParser()
        config.read('config.ini')
        theme_type = config.get('Theme', 'theme', fallback='0')
        background_color = config.get('Theme', 'Background_Color',
fallback='#FFFFFF')
        text_color = config.get('Theme', 'Text_Color', fallback='#000000')
        self.apply_theme(theme_type, background_color, text_color)
    def apply_theme(self, theme_type, background_color, text_color):
        self.setStyleSheet(f"background-color: {background_color}; color:
{text_color};")
        self.ui.statsTable.setStyleSheet(f"""
            QTableWidget {{
                background-color: {background_color};
                color: {text_color};
                gridline-color: {text_color};
                font-size: 14px;}}
            QHeaderView::section {{
```

```
                background-color: {background_color};
                color: {text_color};
                padding: 4px;
                font-size: 14px;
                font-weight: bold;
            }}
            QTableWidget::item {{
                padding: 4px;
                background-color: {background_color};
                color: {text_color};}}""")
    def clickonstats(self):
        connection = pymysql.connect(host='localhost', user='admin',
password='123qwe!@#QWE',
                                     database='test')  # Подключение к базе
        selected_user = self.ui.comboBox.currentText()
        try:
            with connection.cursor() as cursor:
                if selected_user == "Все":  # Параметр фильтрации "Все"
                    query = "SELECT wi.Время, ui.FIM_user FROM work_information wi
JOIN user_information ui ON wi.Код = ui.uid"
                    cursor.execute(query)
                if selected_user == "Внутри":  # Параметр фильтрации "Внутри"
                    query = "SELECT FIM_user FROM user_information where Внутри =
1"
                    cursor.execute(query)
                if selected_user != "Все" and selected_user != "Внутри":    # Параметр
фильтрации "*Пользователь*"
                    query = ("SELECT Время, Код FROM work_information "
                             "WHERE Код = (SELECT uid FROM user_information WHERE
FIM_user = %s)"
```

```
                    "ORDER BY Время DESC")
                cursor.execute(query, (selected_user,))
            result = cursor.fetchall()
            if selected_user == "Внутри":
                self.ui.statsTable.setRowCount(len(result))  # Устанавливаем количество строк
                self.ui.statsTable.setColumnCount(1)
                self.ui.statsTable.setHorizontalHeaderLabels(['Сейчас в здании общежития'])
                for row_index, row_data in enumerate(result):
                    self.ui.statsTable.setItem(row_index, 0, QTableWidgetItem(str(row_data[0])))
            else:
                self.ui.statsTable.setRowCount(len(result))  # Устанавливаем количество строк
                self.ui.statsTable.setColumnCount(2)  # Устанавливаем количество столбцов
                self.ui.statsTable.setHorizontalHeaderLabels(
                    ['Время', 'Пользователь'])  # Устанавливаем заголовки столбцов
                for row_index, row_data in enumerate(result):
                    formatted_time = datetime.datetime.strptime(str(row_data[0]), "%Y-%m-%d %H:%M:%S").strftime(
                        "%d-%m-%Y %H:%M:%S")
                    self.ui.statsTable.setItem(row_index, 0, QTableWidgetItem(formatted_time))
                    self.ui.statsTable.setItem(row_index, 1, QTableWidgetItem(str(row_data[1])))
            self.ui.statsTable.resizeColumnsToContents()
        finally:
            connection.close()
```

```
    def load_data_to_combobox(self):
        connection = pymysql.connect(host='localhost', user='admin',
password='123qwe!@#QWE',
                                     database='test')
        try:
            with connection.cursor() as cursor:
                cursor.execute("SELECT FIM_user FROM user_information")
                result = cursor.fetchall()
                self.ui.comboBox.clear()  # Очистка ComboBox перед добавлением
новых элементов
                for row in result:
                    self.ui.comboBox.addItem(row[0])  # Добавление элементов из базы в
combobox
        finally:
            connection.close()
if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MyWindow()
    window.show()
    sys.exit(app.exec_())
```

ПРИЛОЖЕНИЕ Б

Программный код микроконтроллера arduino

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);
void setup() {
  Serial.begin(9600); // Инициализация последовательного порта
  SPI.begin();       // Инициализация SPI-интерфейса
  mfrc522.PCD_Init(); // Инициализация RFID-считывателя}
void loop() {
  if (!mfrc522.PICC_IsNewCardPresent()) {
    return; // Проверка наличия карты  }
  if (!mfrc522.PICC_ReadCardSerial()) {
    return; // Считывание серийного номера карты}
  // Получение UID метки
  String uid = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    uid += String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : " ");
    uid += String(mfrc522.uid.uidByte[i], HEX);
    // Добавление пробела после каждого символа
    if (i != mfrc522.uid.size - 1) {uid += " ";}}
  uid.toUpperCase(); // Преобразование UID в верхний регистр
  // Удаление лишних пробелов
  uid.replace("  ", " "); // Замена двух пробелов одним
  // Отправка UID в последовательный порт
  Serial.println(uid);
  delay(3000); // Задержка 3 секунды перед следующей попыткой
}
```