МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
им. ГУМАРБЕКА ДАУКЕЕВА»
Кафедра IT-инжиниринг

«ДОПУЩЕН К ЗАЩИТЕ»
Зав. кафедрой PhD, доцент Досжанова А.А
_____ «____» _____2020 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка информационной системы пропускного режима образовательного учреждения
Специальность 5В070400 – «Вычислительная техника и программное обеспечение»
Выполнил Жумартов Н.Н. Группа ВТ-16-2
Научный руководитель к.т.н., доцент Балгабаева Л. Ш.

Консультанты:
по экономической части: к.э.н., профессор Габелашвили К.Р
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
по безопасности жизнедеятельности: к.т.н., доцент Приходько Н.Г
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
по программному обеспечению: ст.преп. Майкотов М.Н
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
Нормоконтролер: ст.преп. Абсатарова Б.Р
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
Рецензент:_____
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
им. ГУМАРБЕКА ДАУКЕЕВА»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Жумартову Нурсултану Нурсейтулы

Тема работы (проекта): Разработка информационной системы пропускного режима образовательного учреждения

Утверждена приказом по университету № от « » 2020 г.

Срок сдачи законченной работы « » 2020 г.

Исходные данные к работе (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): C# – язык программирования, Arduino – инструмент для проектирования электронных устройств, интегрированная среда Visual Studio.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

а) анализ и исследование предметной области;
б) проектирование приложения;
в) реализация приложения;;
г) вопросы безопасности жизнедеятельности и охраны труда;
д) экономическая эффективность работ по стандартизации.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 21 таблица, 41 иллюстрация.

Основная рекомендуемая литература:

1 Гинце А. Новые технологии в СКУД // Системы безопасности, 2005. – 266с.

2 Джон Скит. C# для профессионалов, 2008. – 605с.

3 Петин В. А., Биняковский А. А. Практическая энциклопедия Arduino, 2017. – 152с.

4 Шилдт Г. C#: Учебный курс – СПб.: Питер; Киев: BHV, 2003. - 512 с.

5 Лабор В. В. Си Шарп: Создание приложений для Windows/ В. В. Лабор. – Мн.: Харвест, 2003. - 384 с.

Консультации по работе с указанием относящихся к ним разделов работы

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Габелашвили К.Р.	24.04.2020	
Безопасности жизнедеятельности	Приходько Н.Г.	24.04.2020	
Программная часть	Майкотов М.Н.	14.05.2020	
Нормоконтролер	Абсатарова Б.Р.	18.05.2020	

ГРАФИК
подготовки дипломной работы (проекта)

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Анализ и исследование предметной области	01.11.2019 - 20.12.2019	
Проектирование приложения	21.12.2019 – 20.02.2020	
Программная реализация	21.02.2020 - 20.04.2020	

Дата выдачи задания «01» ноябрь 2020г.

Заведующий кафедрой _____ А.А.Досжанова

Научный руководитель работы _____ Л.Ш. Балгабаева

Задание принял к исполнению студент _____ Н.Н. Жумартов

Аңдатпа

Дипломдық жобаның тақырыбы: «білім беру ұйымы үшін ақпаратқа қол жеткізуді басқару жүйесін дамыту».

Бұл дипломдық жобада ақпаратқа қол жеткізуді басқару жүйесін дамыту жүзеге асырылды, мақсаты белгілі бір оқу орнында студенттердің қауіпсіздігін қамтамасыз ету, сонымен қатар ғимараттағы студенттер мен қызметкерлердің қозғалысын бақылау.

Дипломдық жоба қосымша форматында орындалды. Дипломдық жобаның мақсатына жету үшін C#, Arduino сияқты технологиялар қолданылды.

Сонымен қатар, бағдарламаны әзірлеу шығындары мен құнының экономикалық есебі, әзірленетін жобаның экономикалық орындылығын бағалау жүргізілді және өндірістік өрт сөндіру, ауа баптау жақсарту бойынша іс-шаралар ұсынылды.

Аннотация

Тема дипломного проекта: «Разработка информационной системы пропускного режима образовательного учреждения».

В данном дипломном проекте разработана информационная система пропускного режима, цель которой состоит в том, чтобы обеспечить безопасность для учащихся конкретного образовательного учреждения, и в мониторинге перемещения в здании учащихся и штата сотрудников.

Дипломный проект был выполнен в формате приложения. Для достижения цели разрабатываемого дипломного проекта были применены такие технологии, как C#, Arduino.

Также был проведен экономический расчет затрат и стоимости разработки программы, оценка экономической целесообразности разрабатываемого проекта и были предложены мероприятия по улучшению производственного кондиционирования и пожаротушения.

Abstract

The theme of the graduation project: "Development of an information system for access control of an educational institution".

This graduation project has implemented the development of an access control information system, the purpose of which is to ensure security for students of a particular educational institution, and to monitor the movement of students and staff in the building.

The graduation project was completed in an application format. To achieve the goal of the developed diploma project, technologies such as C #, Arduino were applied.

An economic calculation of the costs and costs of developing the program, an assessment of the economic feasibility of the project under development, and measures to improve industrial air conditioning and firefighting were also carried out.

Содержание

Введение

Темой данного дипломного проекта является разработка информационной системы пропускного режима образовательного учреждения.

На сегодняшний день все большую актуальность приобретают вопросы обеспечения безопасного функционирования объектов бизнеса, жилых зданий, производства, объектов социального назначения. В число основных требований, которые предъявляются к системам безопасности, контроля доступа сегодня следует относить максимально возможную эффективность и независимость от, так называемого, человеческого фактора, что активизирует разработчиков, проектировщиков, производителей, инсталляторов на повышение качества систем. Актуальность пропускных систем всегда была и остается актуальной, тем более для образовательных учреждений. Обеспечение безопасности учащихся – одна из главных установок образовательного учреждения. Именно в образовательных учреждениях учащиеся проводят основную часть своего времени, именно там все предназначено для удобного и комфортного пребывания школьников и студентов, и родители надеются, что место их обучения, соответствует самым высоким требованиям безопасности.

Тема контроля и управления доступом и поддержка безопасности образовательных учреждений во всемирном масштабе является не нова, но в Казахстане проблематика обеспечения безопасности, контроля доступа, и ограничения доступа для посторонних лиц подходит к своему пику. Так только с недавних времен школы, университеты и другие учреждения стали располагать контрольно-пропускными пунктами.

В первой главе был проведен подробный анализ предметной области. В этой главе рассматриваются вопросы о том, что такое СКУД, и в частности, система контроля пропусков. А также приведены примеры наиболее популярных систем.

Во второй главе дано объяснение, что собой представляет разработанный проект. Также был обоснован выбор средств и технологий для реализации дипломного проекта, их ключевые достоинства и недостатки, и за какую часть проекта они отвечают.

В третьей главе подробно описано о проектировании и разработке аппаратной и программной части, и их взаимодействие. Приведены необходимые иллюстрации для полного понимая работы программы, и фрагменты кода, объясняющие логику той или иной части.

В четвертой главе рассмотрены цели и задачи, решаемые экономической частью и выявления экономической целесообразности в разработке данного проекта.

В пятой главе проведён анализ и выявление наиболее опасных и вредных факторов для здоровья во время работы персонала с данной программой.

1 Аналитическая часть

1.1 Анализ предметной области

Защита любого объекта включает несколько рубежей, число которых зависит от уровня режимности объекта. При этом во всех случаях важным рубежом будет система управления контроля доступом (СКУД) на объект.

Хорошо организованная с использованием современных технических средств СКУД позволит решать целый ряд задач. К числу наиболее важным можно отнести следующие:

- противодействие промышленному шпионажу;
- противодействие воровству;
- противодействие саботажу;
- противодействие умышленному повреждению материальных ценностей;
- учет рабочего времени;
- контроль своевременности прихода и ухода сотрудников;
- защита конфиденциальности информации;
- регулирование потока посетителей;
- контроль въезда и выезда транспорта.

Кроме этого, СКУД является барьером для «любопытных». При реализации конкретных СКУД используют различные способы и реализующие их устройства для идентификации и аутентификации личности.

В качестве наиболее часто используемых СКУД можно назвать такие:

- турникеты обычные и настенные;
- турникеты для прохода в коридорах;
- шлюзовые кабины;
- автоматические калитки;
- роторные турникеты;
- вращающиеся двери;
- дорожные блокираторы;
- шлагбаумы;
- парковочные системы;
- круглые раздвижные двери;
- раздвижные турникеты.

Очень важным является вопрос о возможности интеграции СКУД с любой системой безопасности с использованием открытого протокола.

1.1.1 Система пропускного режима и система контроля и управления доступом

Система контроля и управления доступом, или СКУД, является элементом системы безопасности, созданной на основе технических устройств и электронных систем управления, интегрированных в единую сеть, которые работают на специально разработанном программном обеспечении. Кроме

того, система контроля доступа позволяет автоматизировать рабочее время бухгалтерии в компании и в офисе, упрощая обслуживание персонала организации.

Основными функциями, определяющими назначение системы контроля и управления доступом, являются: разграничение и контроль доступа к определенной зоне, помещения; расчет рабочего времени сотрудников и автоматизация службы безопасности.

Функция учета рабочего времени предназначена для работы с персоналом и упрощает контроль рабочего времени сотрудников в компании и их трудовой дисциплины. Кроме того, благодаря этой функции служба безопасности может регулировать доступ персонала на контролируемую территорию в выходные и праздничные дни, а также в нерабочее время. Эта функция выражается как:

- контроль и регистрация с архивированием времени въезда и выезда людей на контролируемую территорию;
- создание и оперативное регулирование базы данных сотрудников, их уровней доступа и способов работы.
- Электронная карта, брелок или отпечаток пальца - все это можно использовать как идентификатор для доступа в контролируемую комнату
- Система контроля управления доступом в охране должна отвечать следующим требованиям:
- контроль и управление временными рамками для прохождения людей и транспортных средств через точки доступа для постоянных зарегистрированных и разовых посещений;
- регистрация даты и времени въезда и выезда транспортного средства с фиксацией в базе данных;
- управление оперативной и мобильной точкой доступа - блокировка и разблокировка;
- дистанционное управление исполнительными устройствами [6].

1.1.2 Виды систем контроля и управления доступом

По типу работы и способу передачи информации установки контроля доступа бывают двух видов: автономные и сетевые.

Автономные - это системы, которые обеспечивают контроль над отдельной (офис, магазин и т. д.) или несколькими комнатами, объединенными для общей цели, когда работа контролируется отдельным (автономным) контроллером. В таких установках контроллер управления не подключен к другим электронным устройствам управления, а работает автономно. Точкой доступа к контролируемой территории обычно является входная дверь. Электронный замок или защелка используется в качестве триггера в автономных системах управления, а карта контроля доступа с различными типами считывателей (стержень, магнит, датчик приближения) используется в качестве идентификатора. Как правило, автономный контроллер работает

только «на входе» контролируемой территории, а «на выходе» использует кнопки управления или датчики.

Сетевые - это системы, оснащенные более мощным и функциональным контроллером (или группой), который обеспечивает работу систем контроля доступа на больших площадях, где одновременно может работать большое количество людей. В таких системах точки доступа являются контрольными точками компаний или других зданий, а винты или проходы используются в качестве стартера. Идентификаторы могут быть разных типов, а сами приводы оснащены дистанционными считывателями [6].

1.1.3 Состав СКУД

На сегодняшний день существует очень много разновидностей СКУД разных производителей, а также ее компонентов. Несмотря на уникальность каждой конкретной системы контроля доступа, она содержит 4 основных элемента: идентификатор пользователя (карта-пропуск, ключ, биометрический признак), устройство идентификации, управляющий контроллер и исполнительные устройства. Общая схема СКУД показана на рисунке 1.1

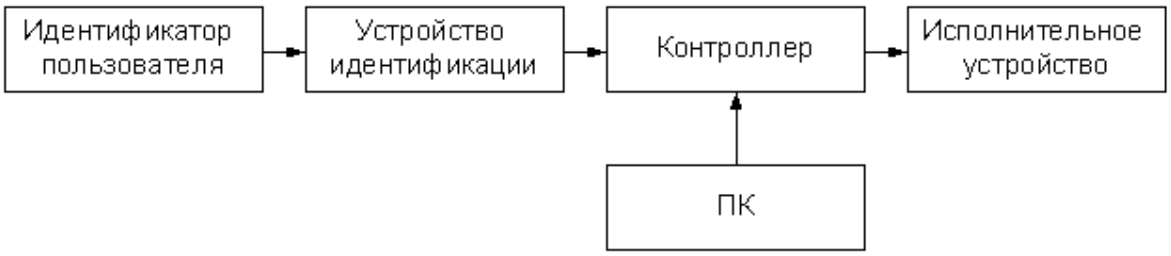

Рисунок 1.1 – Общая схема СКУД

Идентификатор пользователя - это устройство или признак, по которому определяется пользователь. Для идентификации применяются атрибутные и биометрические идентификаторы. В качестве атрибутных идентификаторов используют автономные носители признаков допуска: магнитные карточки, бесконтактные проксимити-карты, брелки «тач-мемори», различные радиобрелки, изображение радужной оболочки глаза, отпечаток пальца, отпечаток ладони, черты лица и многие другие физические признаки. Каждый идентификатор характеризуется определенным уникальным двоичным кодом. В СКУД каждому коду ставится в соответствие информация о правах и привилегиях владельца идентификатора. В настоящее время применяются:

- магнитные карты;
- карты Виганда (Wiegand);
- штрих-кодовые карты;
- ключ-брелок «тач-мемори» (touch-memory);
- биометрические идентификаторы.

Контроллеры - устройства, предназначенные для обработки информации от считывателей идентификаторов, принятия решения и управления исполнительными устройствами. Именно контроллеры разрешают проход через пропускные пункты. Контроллеры различаются емкостью базы данных и буфера событий, обслуживаемых устройств идентификации.

Любой контроллер СКУД состоит из четырех основных частей (рисунок 1.2): считывателя, схем обработки сигнала, принятия решения и схемы буфера событий.

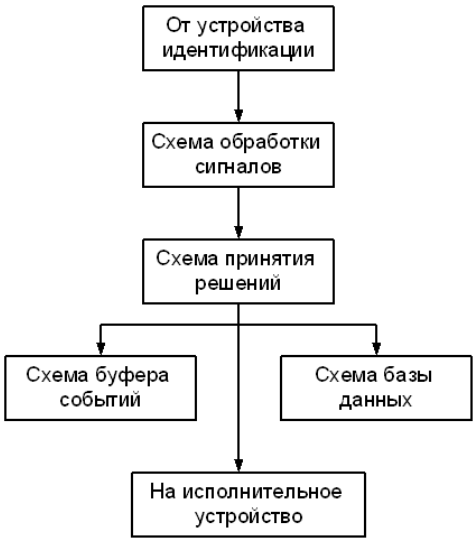

Рисунок 1.2 – Схема контроллера СКУД

Для идентификации личности современные электронные системы контроля доступа используют устройства нескольких типов в зависимости от применяемого вида идентификатора пользователя.

Устройства идентификации (считыватели) расшифровывают информацию, записанную на карточках или ключах других типов, и передают ее в контроллер чаще в виде цифровой последовательности. Считыватели карточек доступа могут быть контактные и бесконтактные. Возможны следующие способы ввода признаков:

- ручной, осуществляемый путем нажатия клавиш, поворота переключателей и т. д.;
- контактный - в результате непосредственного контакта между считывателем и идентификатором;
- дистанционный (бесконтактный) при поднесении идентификатора к считывателю на определенное расстояние.

Для съема информации о биологических признаках человека используют специальные биометрические считыватели (терминалы), а ввод ПИН-кода осуществляется с клавиатур различных типов [10].

1.2 Обзор отечественных и иностранных систем контроля и управления доступом

Аббревиатура СКУД, обозначающая Системы контроля и управления доступом, знакома многим руководителям компаний. Почти все крупные и средние производственные площадки по всему миру имеют как минимум один тип систем контроля и управления доступом. В настоящее время рынок СКУД достаточно велик, есть и российские аналоги средней ценовой категории, и дорогие европейские, и азиатские системы. Каждый бренд имеет свои положительные стороны и индивидуальные эксплуатационные характеристики, которые определяются конкретной задачей обеспечения безопасности.

Но тем не менее на рынке не имеется систем контроля и управления доступом казахстанского разработчика. Весь ассортимент товаров, который можно найти на рынке являются разработками иностранных компаний.

Рассмотрим примеры некоторых брендов систем контроля доступа, известных и популярных среди российских потребителей.

1.2.1 ParsecNET 3 компании НПО «Релвест»

Система ParsecNET 3 - это развитие российского производства, представленное НПО «Релвест». Он используется в различных точках системы контроля и доступа в компаниях. Базовое оборудование ParsecNET 3 обеспечивает интеграцию с системами видеонаблюдения, пожарной и охранной сигнализации.

Преимущества ParsecNET 3:

- может быть установлен как в небольших компаниях, так и в крупных компаниях, имеющих свои небольшие филиалы;
- четкий интерфейс оборудования, не требующий специальной подготовки персонала компании;
- имеется энергонезависимая память, которая подключается к автономному источнику питания в случае сбоя питания;
- вы можете настроить индивидуальные настройки контроля доступа.

В настоящее время бренд установлен на 20 тысяч точках как в России, так и за рубежом, где была пройдена международная сертификация оборудования [9].

1.2.2 СКУД от компании «РусГард»

Компания РусГард, начиная с 2010 года предлагает несколько моделей СКУД, а также отдельных блоков и компонентов, которые можно интегрировать с аналоговым оборудованием и другими учетными приборами СКУД. Наиболее распространенная линейка выпускается под брендом ACS.

Преимущества системы СКУД от РусГард:

- отличное решение для гостиничных комплексов;

- СКУД работает по принципу функционирования «двух дверей», что улучшает систему управления при повторной проверке;
- отличное решение для офиса. скуд контролирует данные не только на проходах, но и интегрируется в систему противопожарной защиты офисного помещения;
- некоторые модели имеют модифицированный принцип сбора данных для центрального пункта охраны.

Существует специальный терминал для регистрации посетителей компании. СКУД от РусГард можно интегрировать с другими системами, которые можно настроить для работы в соответствии с определенными параметрами [9].

1.2.3 СКУД «Сфинкс»

Бренд «Сфинкс» является одним из «умных» решений для современного обустройства систем контроля доступа для компаний. Существует множество функций, которые позволяют легко настроить работу оборудования.

Преимущества СКУД «Сфинкс»:
- учет контроля рабочего времени сотрудников компании;
- технология бейдж-редактора;
- контроль времени прохода на территорию;
- множественные методы идентификации - магнитные карты, электронные ключи, бумажные носители;
- взаимодействие с системой видеонаблюдения.

Важным плюсом «Сфинкс» является взаимодействие для отчета с системой 1С-Бухгалтерия [9].

2 Описание проекта

Данный проект представляет собой информационную систему пропусков образовательного учреждения или иного предприятия. Она не является полноправной системой контроля и управления доступом, а является дополнением, которая может легко быть интегрирована в уже существующую, установленную систему.

2.1 Выбор средств и технологий

В данном подразделе описываются какие технологии были использованы при разработке дипломного проекта, их ключевые достоинства и недостатки, почему были выбраны именно эти технологии, и за какую часть они отвечают.

2.1.1 C#

C# (произносится си шарп) - объектно-ориентированный язык программирования. C # - этоуниверсальный мульти-парадигма язык программирования, включающий строгую типизацию, лексическую сферу применения, чрезвычайно важные, декларативные, функциональные, универсальные, объектно-ориентированные и компонентно-ориентированные дисциплинарные программы.

Данный язык использует объектно-ориентированный подход к программированию ко всему. Это означает, что пользователю нужно будет описывать абстрактные конструкции на основе предметной области, а потом реализовывать между ними взаимодействие. Этот подход является очень популярен среди разработчиков, потому что позволяет не держать в голове всю информацию.

В дипломном проекте язык программирования C# был выбран за его простоту освоения, и за предоставление бесплатной среды разработки. Этот язык программирования полностью отвечает за разработку клиентской части разрабатываемого проекта, а также за установку связи между клиентской и аппаратной частями.

2.1.2 Arduino

Arduino – это инструмент для проектирования электронных устройств (электронный конструктор) более плотно взаимодействующих с окружающей физической средой, чем стандартные персональные компьютеры, которые фактически не выходят за рамки виртуальности. Это платформа, предназначенная для «physical computing» с открытым программным кодом, построенная на простой печатной плате с современной средой для написания программного обеспечения.

Arduino применяется для создания электронных устройств с возможностью приема сигналов от различных цифровых и аналоговых датчиков, которые могут быть подключены к нему, и управления различными исполнительными устройствами. Проекты устройств, основанные на Arduino, могут работать самостоятельно или взаимодействовать с программным

обеспечением на компьютере (напр.: Flash, Processing, MaxMSP). Платы могут быть собраны пользователем самостоятельно или куплены в сборе. Среда разработки программ с открытым исходным текстом доступна для бесплатного скачивания.

Язык программирования Arduino является реализацией Wiring, схожей платформы для «physical computing», основанной на мультимедийной среде программирования Processing.

2.1.3 RFID

Радиочастотная идентификация (RFID) - этотехнология бесконтактной идентификации объектов при помощи радиочастотного канала связи. Идентификация объектов производится по уникальному идентификатору, который имеет каждая электронная метка. Считыватель излучает электромагнитные волны определенной частоты. Метки отправляют в ответ информацию – идентификационный номер, данные памяти и другое.

Преимуществами технологии RFID являются возможность скрытой установки меток, высокая скорость считывания данных, невозможность подделки, бесконтактная.

Основным преимуществом технологии RFID является бесконтактный метод идентификации, который не требует контакта между считывателем и носителем информации - идентификатором (карточкой, ключом, отметкой, отметкой и т. Д.). Идентификатор содержит электронный чип (с уникальным кодом) и антенну. Когда идентификатор попадает в область действия считывателя, код передается от первого ко второму по радиоканалу. После получения идентификационного кода считыватель отправляет его контроллеру ACS, где решение о принятии принимается автоматически. Этим типовым принципом работы ограничивается сходство различных систем, где применяется технология RFID (рисунок 2.1), поскольку в них могут использоваться различные частотные диапазоны, типы идентификаторов и прочее [16].

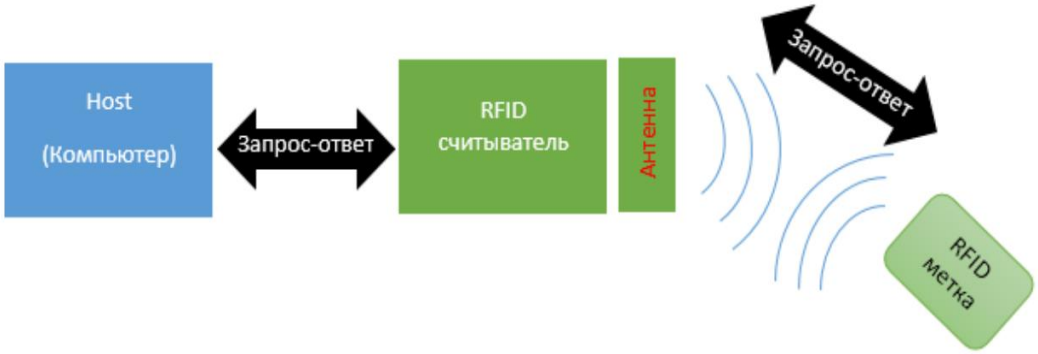

Рисунок 2.1 – Алгоритм идентификации RFID-метки.

2.1.4 Fritzing

Прорамма Fritzing предназначена для разработки электронных устройств от прототипа в виде макетной платы до конечного продукта в виде печатной платы.

Устройства создаются из готовых элементов, список которых можно посмотреть в правом верхнем углу программы, которые включают монтажную плату Arduino, различные аналоговые и цифровые микросхемы, транзисторы, светодиоды, резисторы, конденсаторы, кнопки, макетные платы, батарейки и даже моторчики.

Чтобы поместить их на схему – их достаточно выбрать из списка и перетащить на рабочее пространство левой кнопкой мышки.

Рисовать схему можно как в режиме макетной платы – так и в режиме принципиальной схемы. Вторая схема будет строиться автоматически! Т.е. набросав схему на макетной плате – она автоматически построится и в виде принципиальной схемы [15].

2.1.5. Среда .NET Framework

Среда, в которой работает язык программирования C# называется NET.Framework и поэтому лучше всего рассматривать их во взаимосвязи. На это есть несколько причин:

1) C# изначально разрабатывался компанией Microsoft для создания кода .NET Framework;

2) сама среда .NET Framework определяет библиотеки, оторые используются C#.

Так что же такое .NET Framework? Отвечая на вопрос, можно сказать, что .NET Framework определяет среду, которая поддерживает развитие и выполнение платформонезависимых приложений. Она допускает совместную работу в приложении различных языков программирования, а также обеспечивает переносимость программ и общую модель программирования для Windows. Однако среда .NET Framework не ограничена платформой Windows и написанные для этой платформы программы могут быть перенесены на другие платформы [18].

Язык C# использует две важные составляющие .NET Framework. Первая – это не зависящая от языка среда исполнения (Common Language Runtime, CLR). Это система позволяет программам быть переносимыми, а также поддерживает программирование с использованием нескольких языков. CLR управляет исполнением нашей программы и является частью технологии .NET Framework.

Вторая составляющая – библиотека классов .NET Framework, которая предоставляет программе доступ к среде исполнения, например, используя для ввода и вывода данных. Программа может быть запущена везде, где

поддерживается среда исполнения .NET до тех пор, пока она ограничена характеристиками, определенными библиотекой классов .NET.

Не зависящая от языка среда исполнения (CLR) управляет выполнением кода .NET. При компилировании C#-программы мы получаем не исполняемый код, а файл, содержащий специальный тип псевдокода, называемый промежуточным языком Microsoft (Microsoft Intermediate Language, MSIL). Язык MSIL определяет набор переносимых конструкций, не зависимых от конкретного процессора, то есть, по существу, MSIL определяет переносимый язык ассемблера.

Система CLR транслирует промежуточный код в исполняемый во время запуска программы. Любая программа, компилированная в MSIL, может быть запущена в любой операционной системе, для которой реализована среда CLR. Это часть механизма, с помощью которого .NET Framework достигается переносимость программ.

Промежуточный язык Microsoft превращается в исполняемый код при использовании JIT-компилятора (с англ. Just-in-time – в нужное время). При исполнении .NET-программы система CLR активизирует JIT-компилятор, который затем превращает MSIL во внутренний код данного процессора. Следовательно, ваша C#-программа фактически исполняется как внутренний код, хотя изначально она компилировалась в MSIL. Таким образом, время запуска вашей программы практически такое же, как если бы она была сразу скомпилирована во внутренний код, но при этом у вас появляется преимущество MSIL – переносимость программы.

Кроме того, при компилировании C#-программы в дополнение к MSIL вы получаете еще один компонент – метаданные, которые описывают данные, используемые вашей программой, и позволяют вашему коду взаимодействовать с другим кодом. Метаданные содержатся в том же файле, что и MSIL.

Платформа .NET Framework предоставляет приложениям следующие службы (услуги):

- управление памятью. В приложениях платформы .NET Framework среда CLR предоставляет сервисы для управления ресурсами памяти автоматически;
- система общего типа. В платформе .NET Framework базовые типы определяются системой, называемой CTS (Common Type System). Используются одни и те же базовые типы для всех языков .NET Framework;
- совместимость версий. Приложения определенной версии, могут выполняться без изменений на более поздней версии;
- взаимодействие языков. Языковые компиляторы .NET Framework компилируют приложение в промежуточный код, CIL (Common Intermediate Language), который затем компилируется во время исполнения приложения средой CLR. Следовательно, программы, написанные на одном языке, доступны в других языках;

- расширенная библиотека классов. Облегчает написание большого кода для стандартных операций, благодаря использованию легкодоступной библиотеки типов и членов из библиотеки классов .NET Framework;

- параллельное выполнение. .NET Framework помогает в разрешении конфликтов версий, разрешая установку нескольких версий среды CLR на одном компьютере.

При написании C#-программы создается так называемый управляемый код, который выполняется под контролем не зависящей от языка среды исполнения (CLR). Так как программа запускается под контролем CLR, управляемый код должен соответствовать определенным требованиям, при выполнении которых, он получает множество преимуществ, включая современное управление памятью, способность совмещать языки, высокий уровень безопасности передачи данных, поддержку контроля версии и понятный способ взаимодействия компонентов программного обеспечения. Требования, которым должен соответствовать управляемый код: компилятор должен создать MSIL-файл, предназначенный для CLR, а также использовать библиотеки .NET Framework.

Альтернативой управляемому коду является не управляемый код, который не выполняется CLR. До появления .NET Framework все Windows-программы использовали не управляемый код, сейчас же оба кода могут работать вместе, и язык C#, генерируя управляемый код, способен взаимодействовать с созданными ранее программами.

Все преимущества управляемого кода обеспечивается CLR. Если же ваш код будет использоваться программами, написанными на других языках, для максимальной совместимости необходимо придерживаться общей языковой спецификации (Common Language Specification, CLS). Эта спецификация описывает набор характеристик, являющихся общими для различных языков. Соответствие кода общей языковой спецификации особенно важно при создании компонентов программного обеспечения, которые будут использоваться другими языками.

2.1.6 Интерфейс программирования Windows Forms

Приложение разрабатывалось в виде оконного пользовательского интерфейса, который реализовывался при помощи методов визуального программирования и классов библиотеки Windows Forms.

Windows Forms представляет собой интерфейс программирования приложений (API), который используется для создания приложений, оснащенных графическим интерфейсом. Windows Forms является частью Microsoft .NET Framework. Интерфейс упрощает доступ к визуальным компонентам Microsoft Windows за счет того, что создает обертки для существующего Win32 API в управляемом коде [19]. Управляемый код не зависит от языка разработки, что позволяет использовать Windows Forms как при написании программного обеспечения на C#, C++, так и на VB.Net, J# и других языках.

Основной класс Windows Forms – форма, экземплярами которого являются главные и диалоговые окна. Форма – экранный объект, представляемый в виде прямоугольной формы, использующийся в основном для предоставления информации пользователю или для обработки вводимой информации пользователем. Они могут иметь вид стандартного диалогового окна, многодокументного интерфейса (MDI) и поверхность для отображения графической информации [20].

Формы являются потомками класса Form, определенного в пространстве имен System.Windows.Forms. Интерфейс Windows Forms позволяет работать в режиме конструктора, добавляя элементы управления (кнопки, поля для ввода текста, поля для отображения текста, меню и прочие компоненты) простым «перетаскиванием» на поверхность формы. В коде они представлены как поля класса формы. Все элементы управления окна являются объектами классов, содержащихся в System.Windows.Forms и являющихся потомками базового класса Control. Ответная реакция программы на определенное действие (событие), связанное с элементом управления – обработчик события - оформляется в виде метода формы.

Некоторые элементы управления, которые можно располагать на формах:

- Label (Надпись);
- Button (Кнопка);
- Panel (Панель);
- ListBox (Список);
- CheckBox (Флажок);
- RadioButton (Переключатель);
- MessageBox (Окно сообщений).
- Menu (Меню);
- TabControl (Управление вкладками);
- Toolbar (Панель инструментов);
- TreeView (Дерево);
- DataGrid (Сетка данных);
- PictureBox (Изображение);
- RichTextBox (Текстовое поле с поддержкой формата RTF) [].

2.2 Постановка цели и задачи

Целью данного дипломного проекта является разработка системы пропускного режима, которая если и не сможет полноценно заменить полную систему контроля и управления доступом, но которая сможет стать их частью и дополнением, которые уже основательно вписались в наши жизни. Она должна отвечать минимальным стандартам систем обеспечения физической безопасности и режимам контроля пропусков:

- учёт рабочего времени;
- ведение базы персонала / посетителей;

- световое и (или) звуковое оповещение о попытках НСД;
- учет индивидуального отработанного времени;
- улучшение работы охранных и учетных отделов;
- увеличение безопасности учреждения.

Для реализации данного проекта необходимо осуществить следующие требования:

- выбор оптимальных инструментов для разработки;
- сборка схемы микроконтроллера аппаратной части;
- разработка интерфейса для оконного приложения;
- установка связи между аппаратной и программной частью.

В ходе разработки необходимо: провести анализ систем контроля и управления доступом, изучить и описать наиболее популярные предложения рынка, разработать аппаратную, сделать ее структурную, функциональную и принципиальные схемы, и программную часть с удобным и приятным визуальным интерфейсом с возможностью редактирования базы данных, а также описать алгоритм их работы.

Данная программа будет реализована в виде оконного приложения. Это будет программа, состоящая из 8 окон. Первое окно, представляет собой окно выбора COM-порта, второе окно является главным меню, откуда можно получить доступ в окно отделы, предоставляющий список всех отделов, в окно должности, показывающий список всех должностей, в окно сотрудники, выдающий полный список сотрудников и в окно звания. Просмотр полных данных определённого сотрудника осуществляется с помощью окна профиль сотрудника, где добавление карты возможно из окна добавление карты.

Также программа предусматривает хранение, редактирование и удаление данных.

3 Проектирование и разработка программного продукта

Разрабатываемый проект представляет собой приложение, которое имеет две части: аппаратную часть и клиентскую часть. Данное приложение отображается и взаимодействует с пользователем через оконное приложение.

Клиентская часть представляет собой приложение, которая отображается пользователю, выполнена в форме приложения и визуально взаимодействует с пользователем. На этой стороне работает язык программирования C#.

Аппаратная часть представляет собой конструкцию собранной на базе Arduino. Код для Arduino написан на языке Arduino.

Связь между программной и аппаратной частью поддерживается в среде разработки Microsoft Visual Studio, и написан также на языке C#.

3.1 Проектирование и разработка аппаратной части

Проектирование аппаратной части была выполнена в программе Fritzing, программном обеспечении для виртуального моделирования электрических цепей, схем и электронного оборудования. С помощью программы можно преобразовать прототип на основе Arduino в топологию печатной платы для серийного изготовления. Также данное приложение помогает упростить откладку схем при помощи макетных плат.

Макетная плата, выполненная в программе Fritzing, даёт полное понимание о соединении микроконтроллера. В качестве микроконтроллера была выбрана Arduino Mega в связи с её широкими возможностями, являясь уникальным решением для постройки различных проектов. Конечно, можно было бы использовать микроконтроллер более дешевого образца, что на конечный продукт не повлиял бы (рисунок 3.1).

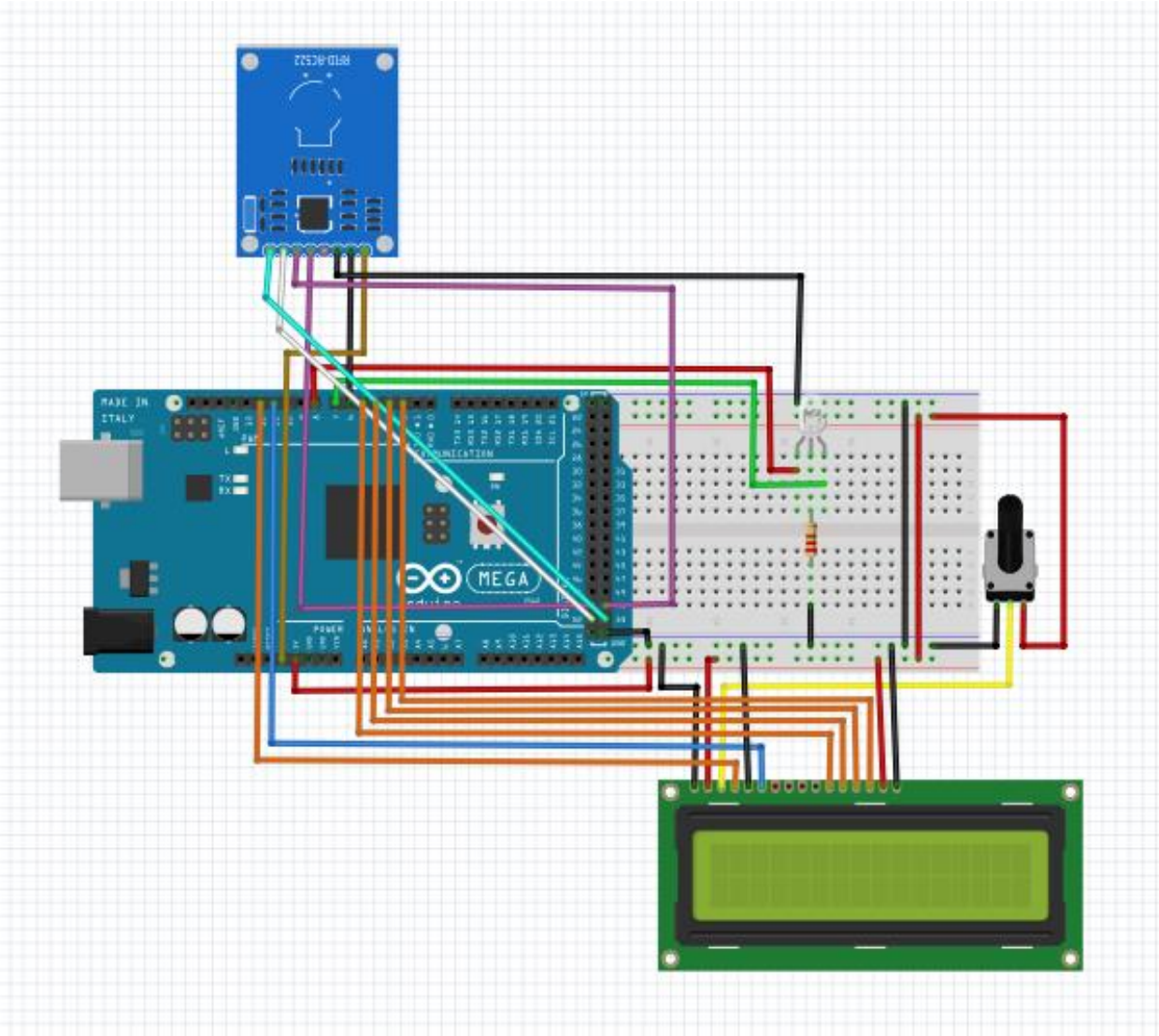

Рисунок 3.1 Макетная плата в программе Fritzing

После сбора схемы на макетной плате программа автоматически сгенерировала принципиальную схему (рисунок 3.2).

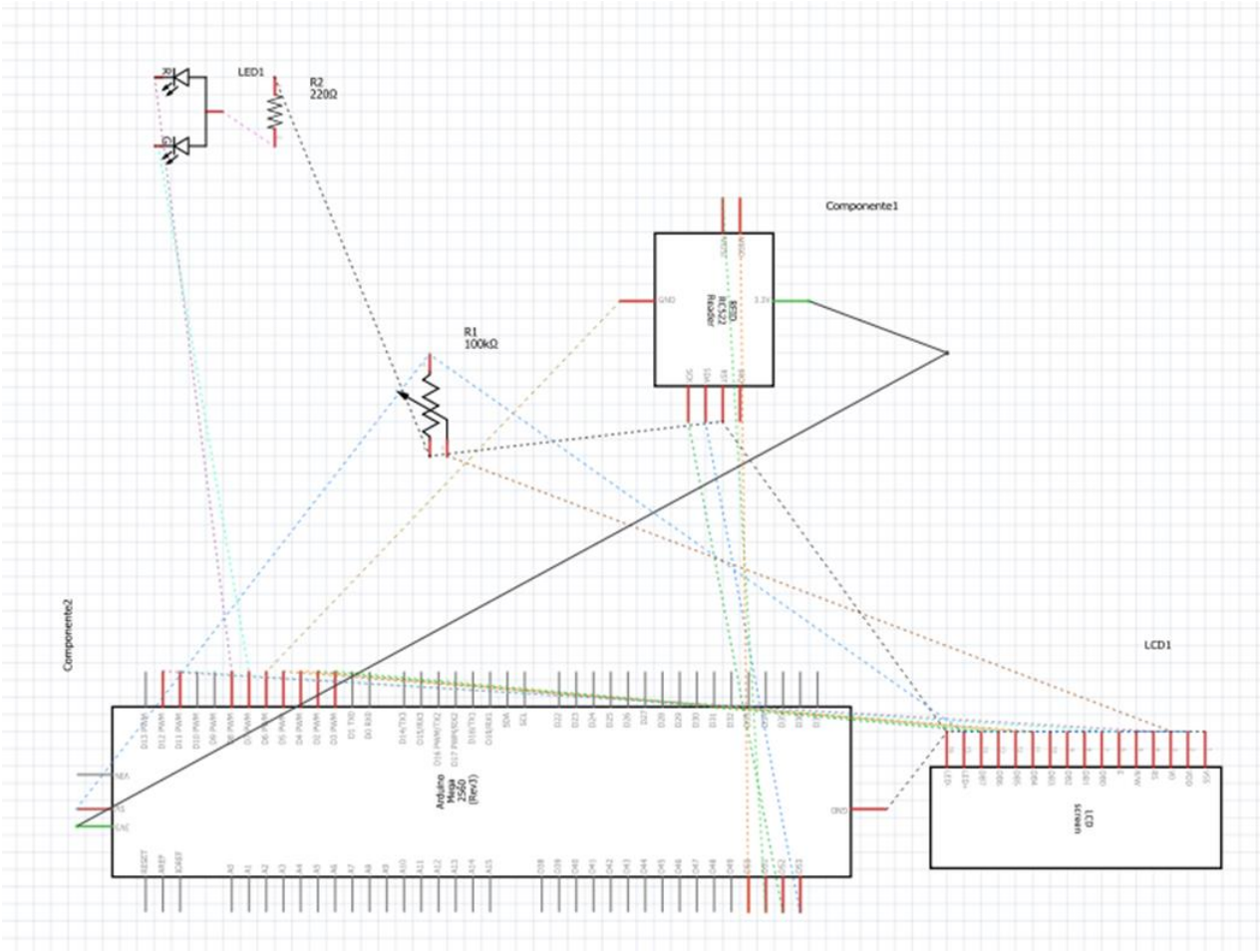

Рисунок 3.2 Принципиальная схема в программе Fritzing

После проектирования макетной платы необходимо написать код для микроконтроллера, чтобы та выполняла поставленные задачи.

Первым делом подключаются библиотеки, определяются пины и создается экземпляр MFRC522 rfid instance (рисунок 3.3).

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         49
#define SS_PIN          53
#define DIRECTION_PIN   A0
#define GREEN_PIN       2
#define RED_PIN         3
#define SOUND_PIN       4

MFRC522 rfid(SS_PIN, RST_PIN);
uint8_t header[]={0x37, 0x83};
```

Рисунок 3.3 – Базовая предустановка микроконтроллера

Затем определяется состояние направления и доступа (рисунок3.4).

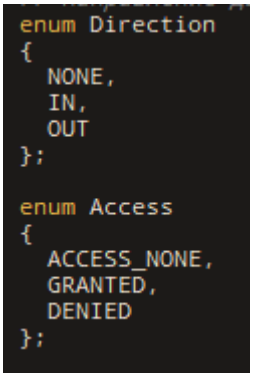

Рисунок 3.4 – Определение состояния

Дальше вызывается функция setup(), для инициализации переменных, определения режимов работы выводов, запуска используемых библиотек и т.д. Функция setup запускает только один раз, после каждой подачи питания или сброса платы Arduino (рисунок 3.5).

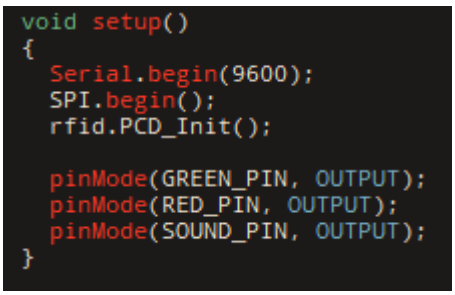

Рисунок 3.5 – Функция «setup()»

В следующей функции loop(), запускается цикл, позволяя программе совершать вычисления и реагировать на них.

Первая команда, это условие, где Arduino ожидает карту, после чего идёт считывание UID карты и проверка типа карты (рисунок 3.6).

```
void loop()
{
  // Ждем карту
  if ( ! rfid.PICC_IsNewCardPresent())
    return;

  // Проверяем, что UID считатн
  if ( ! rfid.PICC_ReadCardSerial())
    return;

  // Проверяем тип карты
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);

  // Check is the PICC of Classic MIFARE type
  if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
      piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
      piccType != MFRC522::PICC_TYPE_MIFARE_4K)
  {
    return;
  }
```

Рисунок 3.6 – Проверка карты

После проверки карты идёт процесс проверки направления движения и формируется пакет данных для отправки на ПК, после чего завершаются операции с картой (рисунок 3.7).

```
// Проверяем направление движение
Direction dir = get_direction();

// Формируем пакет
Serial.write(header, sizeof(header));
Serial.write(rfid.uid.uidByte, rfid.uid.size);
Serial.write((uint8_t*)&dir, sizeof(dir));
Serial.flush();

// Завершаем операцию с картой
rfid.PICC_HaltA();
rfid.PCD_StopCrypto1();
```

Рисунок 3.7 – Завершение операций с картой

После завершения операций с картой ожидается ответ от ПК. Если ответ не последовал издается соответствующий сигнал (рисунок 3.8).

```
// Ждем ответ
int accessRaw = get_response();
if(accessRaw == -1)
{
  // Таймаут, ответ не пришел
  beep(200, 100);
  return;
}
```

Рисунок 3.8 – фрагмент кода

Если же ответ от ПК пришел, то загорается зелёная или красная лампочка с характерным сигналом, соответственно разрешающая или запрещающая доступ (рисунок 3.9).

```
  // Если получен ответ от ПК
  Access access = (Access)accessRaw;

  if(access == GRANTED)
  {
    digitalWrite(GREEN_PIN, 1);
    beep_granted();
    delay(500);
    digitalWrite(GREEN_PIN, 0);
  }
  else if(access == DENIED)
  {
    digitalWrite(RED_PIN, 1);
    beep_denied();
    delay(500);
    digitalWrite(RED_PIN, 0);
  }
  else
  {
    beep_info();
  }
}
```

Рисунок 3.9 – Описание действия микроконтроллера при получении ответа от ПК

После идёт описывание реакции переключателя движения на отклонение от изначального положения (рисунок 3.10).

```
Direction get_direction()
{
  int dirRaw = analogRead(DIRECTION_PIN);
  Direction dir = NONE;

  if(dirRaw < 100)
    dir = IN;
  else if(dirRaw > 900)
    dir = OUT;

  return dir;
}
```

Рисунок 3.10 – Определение направления движения

Дальше следует описание звуковых сигналов и их интервал в зависимости от ответа устройства (рисунок 3.11).

```
void beep_info()
{
  beep(1000, 100);
}

void beep_granted()
{
  beep(1000, 300);
}

void beep_denied()
{
  for(int i = 0; i<3; i++)
  {
    beep(200, 100);
    delay(100);
  }
}

void beep(int freq, int duration)
{
  tone(SOUND_PIN, freq);
  delay(duration);
  noTone(SOUND_PIN);
}
```

Рисунок 3.11 – Логика взаимодействия звуковых сигналов

3.2 Проектирование и разработка клиентской части и его интерфейса

Проектирование и разработка клиентской части проводилась на языке программирования C# в среде разработки Microsoft Visual Studio 2017.

Логика взаимодействия между контроллером и ПО представлена на блок схеме (рисунок 3.12).

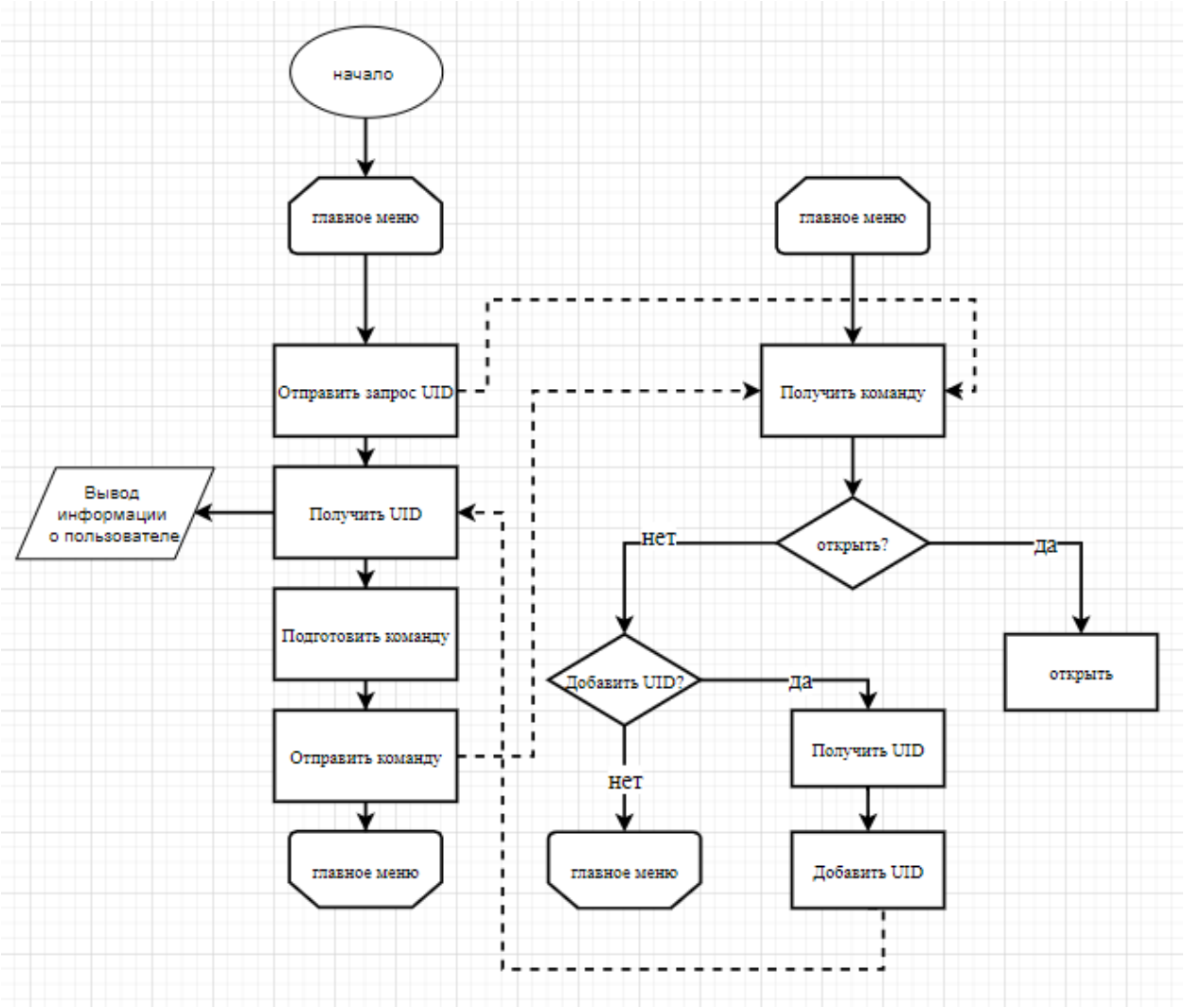

Рисунок 3.12 Блок схема процесса

Постройка оконного приложения была произведена в конструкторе Visual Studio.

Все взаимодействия пользователя с приложением происходят через окно главного меню (рисунок 3.13), оно является главным связующим звеном всей программной части. Конструктор XAML предоставляет визуальную рабочую область для редактирования документов XAML. Для работы некоторых функций Visual Studio, таких как функции IntelliSense для ресурсов и привязки данных, необходимо включить конструктор XAML.

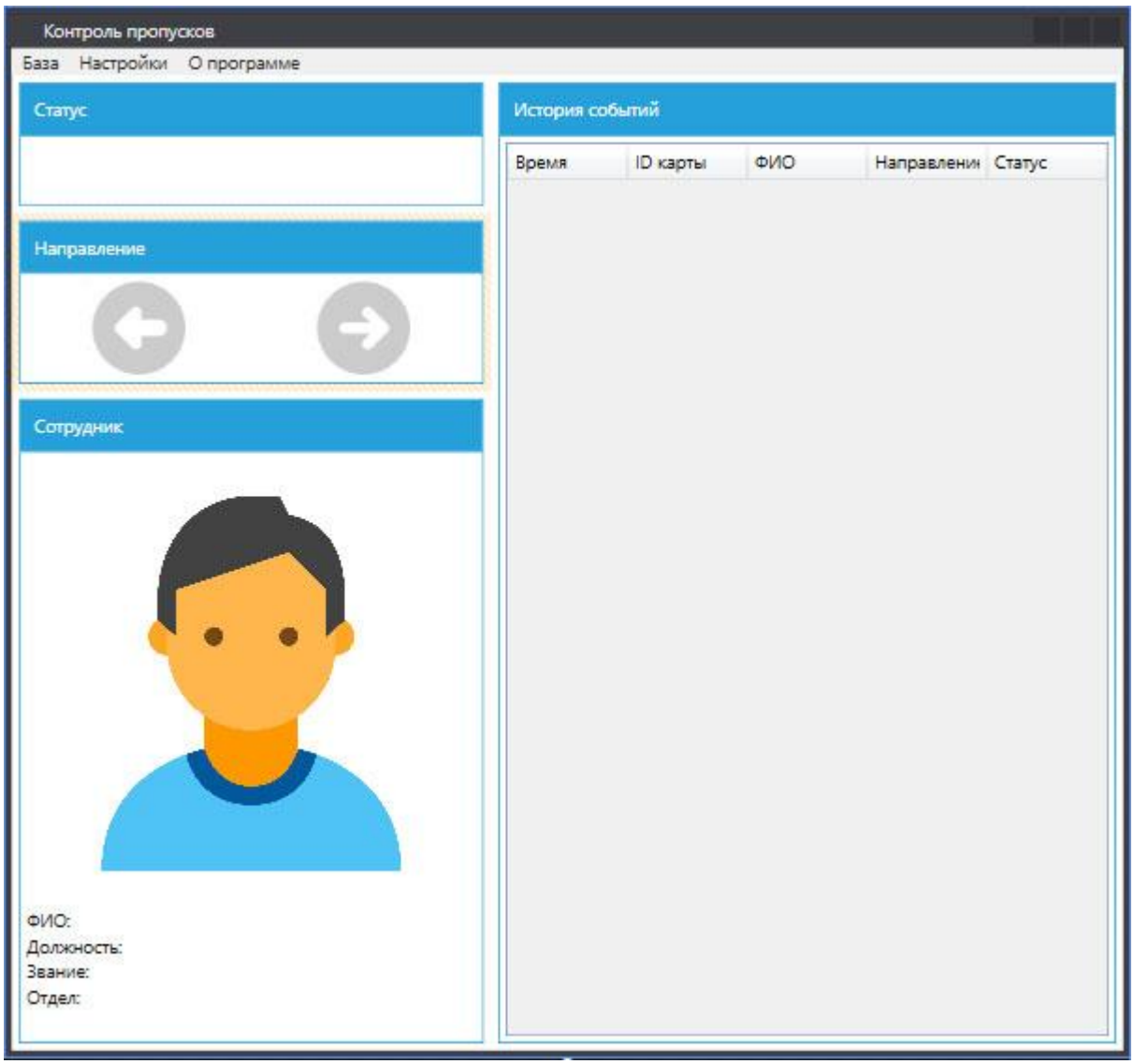

Рисунок 3.13 – Окно главного меню

Оно состоит из четырех отдельных окон, взаимодействующих между собой.

Первое – окно статуса, отображает разрешается ли доступ этому сотруднику доступ или нет, взаимодействие происходит за счет операторов выбора «if, else» (рисунок 3.14). Если карта сотрудника заблокирована, то выполняется условие else, что выдаёт команду DENIED. Если же карта не заблокирована, то выполняется условие в условии с учётом направления, на выход или же вход соответственно записывается время ухода или время прибытия в «историю событий» с данными времени, ID карты, ФИО сотрудника, направлением и статусом сотрудника и происходит выполнение команды GRANTED, разрешающая доступом.

```
if (User != null)
{
    if (direction == Domain.Direction.IN)
    {
        var ws = new WorkShift()
        {
            ArrivalTime = DateTime.Now,
            CardUid = (long)uid
        };
        _ctx.WorkShifts.Add(ws);
        _ctx.SaveChanges();
    }
    else
    {
        long id = (long)uid;
        var last = _ctx.WorkShifts.Where(x => x.CardUid == id).OrderByDescending(x => x.Id)
            .FirstOrDefault();
        if (last != null)
        {
            last.LeavingTime = DateTime.Now;
            _ctx.SaveChanges();
        }
    }

    Access = AccessStatus.GRANTED;
    Direction = direction;
}
else
{
    Access = AccessStatus.DENIED;
    Direction = null;
}
```

Рисунок 3.14 – Логика проверки пропусков, разрешающая или запрещающая проход

Второе окно – стрелки, при предоставлении доступа загорается стрелка указывающее в соответствующее направление IN или OUT. Работает за счет триггеров, заменяющие серую, неактивную стрелку на зелёную, активную (рисунок 3.15).

```
<Style x:Key="left_arrow_icon_style" TargetType="Image" BasedOn="{StaticResource arrow_icon_style}">
    <Setter Property="Source" Value="/Images/arrow_left_disable.png"/>
    <Style.Triggers>
        <DataTrigger Binding="{Binding}" Value="{x:Static models:Direction.IN}">
            <Setter Property="Source" Value="/Images/arrow_left_active.png"/>
        </DataTrigger>
    </Style.Triggers>
</Style>
<Style x:Key="right_arrow_icon_style" TargetType="Image" BasedOn="{StaticResource arrow_icon_style}">
    <Setter Property="Source" Value="/Images/arrow_right_disable.png"/>
    <Style.Triggers>
        <DataTrigger Binding="{Binding}" Value="{x:Static models:Direction.OUT}">
            <Setter Property="Source" Value="/Images/arrow_right_active.png"/>
```

Рисунок 3.15 – Описание действия триггеров

Третье окно – окно сотрудник, куда выводится информация о сотруднике, его ФИО, должность, звание, отдел, а также фотография, что позволит идентифицировать человека по внешности. Происходит вывод информации по заданному UID карты, логика взаимодействия представлена на рисунке 3.16.

```
public partial class UserProfileCard : UserControl
{
    public UserProfileCard()
    {
        InitializeComponent();
    }
}
```

Рисунок 3.16 – Логика взаимодействия для UserProfileCard

Четвертое окно – окно история событий, сюда идёт вывод записанной информации о сотрудниках, прошедших контрольно-пропускной пункт, ID его карты, ФИО, направление, статус и время выхода или входа. Логика взаимодействия представлена на рисунке 3.17.

```
public partial class LogView : UserControl
{
    public LogView()
    {
        InitializeComponent();
    }
}
```

Рисунок 3.17 – Логика взаимодействия для LogView

При нажатии кнопки «База» выпадает меню, которое позволяет получить доступ к окнам отделы, должности, звания, сотрудники (рисунок 3.18).

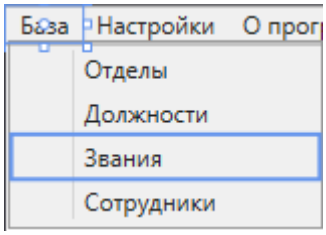

Рисунок 3.18 – выдающее меню «База»

Окно «Отделы» (рисунок 3.19), здесь выводится информация об отделах, ID и название соответствующего отдела.

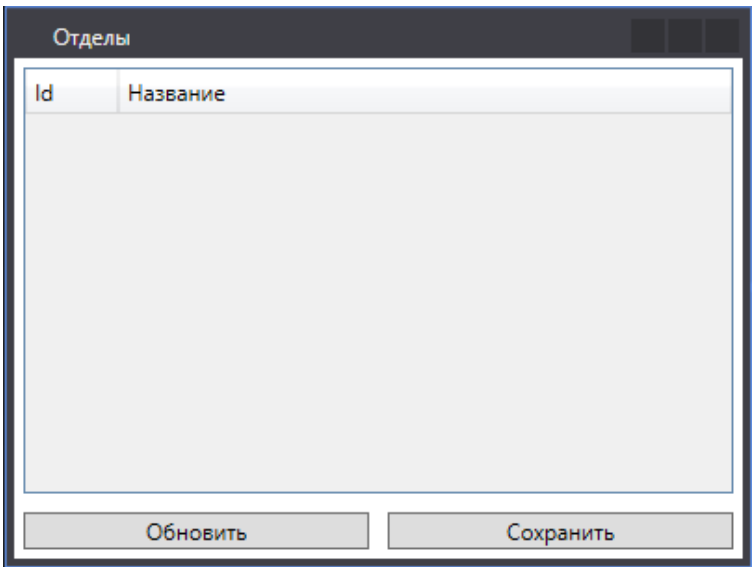

Рисунок 3.19 – Окно «Отделы»

Также имеются кнопки «обновить», которая обновляет окно сбрасывая не сохранённые данные и «сохранить», которая сохраняет изменённые данные и обновляет окно (рисунок 3.20).

```
private void BtnUpdate_OnClick(object sender, RoutedEventArgs e)
{
    Update();
}

private void BtnSave_OnClick(object sender, RoutedEventArgs e)
{
    _ctx.SaveChanges();
    Update();
}
```

Рисунок 3.20 – логика взаимодействия для кнопок «обновить» и «сохранить»

Также при нажатии правой кнопки мыши на выбранный отдел появляется контекстное меню, которая позволяет удалить его (рисунок 3.21).

```
private void MnuRemoveSelected_OnClick(object sender, RoutedEventArgs e)
{
    while (grid.SelectedItems.Count > 0)
    {
        var item = (Department)grid.SelectedItems[0];
        _ctx.Departments.Remove(item);
    }
}
```

Рисунок 3.21 – логика взаимодействия контекстного меню

Добавление нового отдела происходит автоматически при добавлении пользователя с новым отделом.

Окно «Звания» (рисунок 3.22), здесь так же, как и в окне «Отделы» выводится информация, но только о званиях сотрудников, с кнопками «Обновить» и «Сохранить» которые выполняют те же функции, разделами «ID» и «Название», выдающие соответствующую информацию, и вывод контекстного меню при нажатии правой кнопки мыши на определенный пункт.

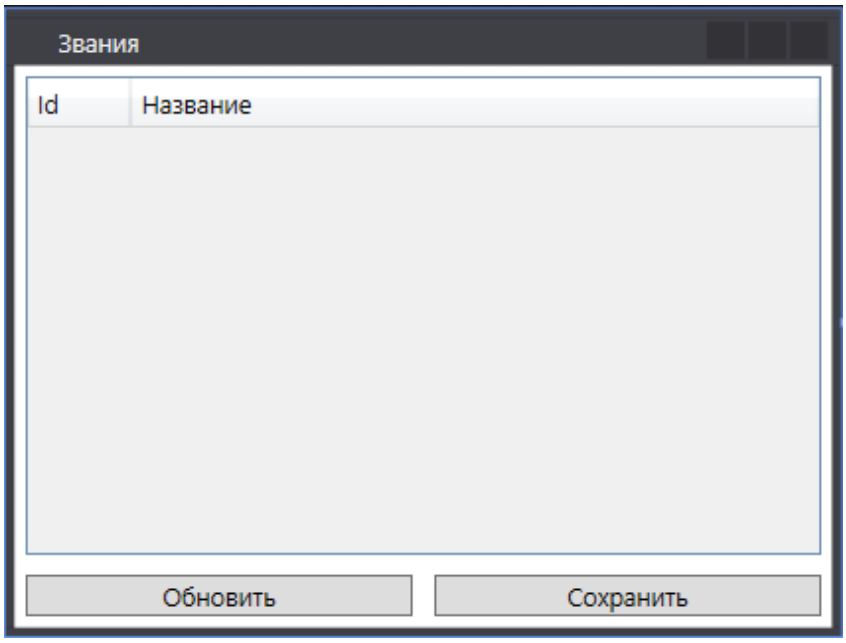

Рисунок 3.22 –Окно «Звания»

Окно «Сотрудники» (рисунок 3.23), здесь в списочной форме выводятся данные о сотрудниках. Поля «Id», «Фото», «Фамилия», «Имя», «Отчество», «Отдел», «Должность» и «Звание» выводят соответствующую информацию.

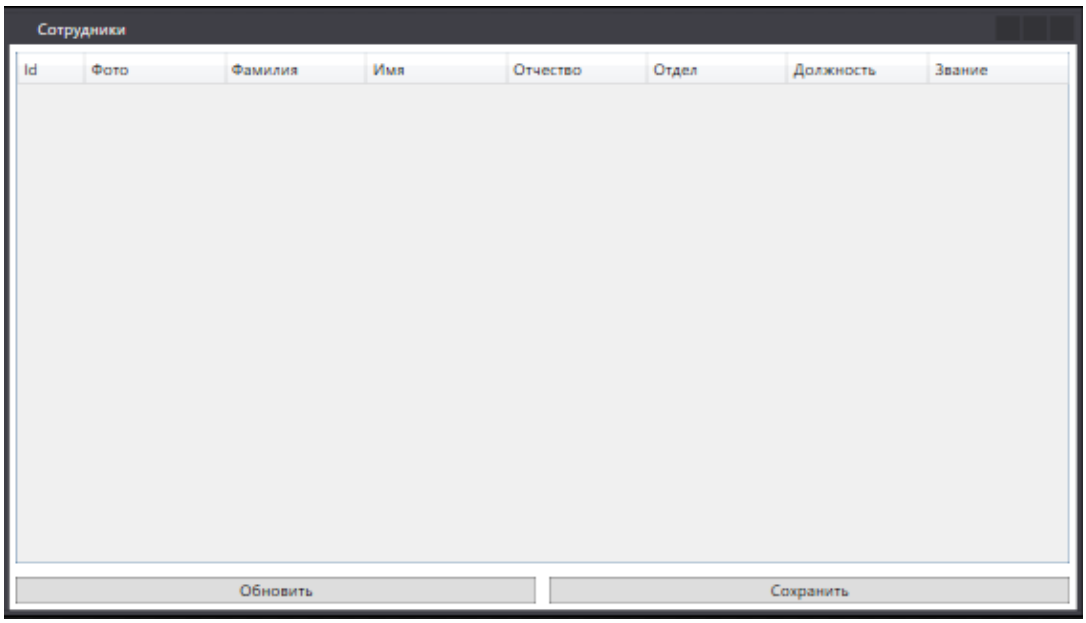

Рисунок 3.23 – Окно «Сотрудники»

Также присутствуют кнопки «Обновить» и «Сохранить» которые выполняют аналогичные функции, описанные выше, но с добавлением исключения. Так при неправильном сохранении данных выходит окно ошибки (рисунок 3.24).

```
catch (Exception ex)
{
    MessageBox.Show("Данные заполнены неверно", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
}
```

Рисунок 3.24 – Фрагмент кода, описывающее исключение

В окно «Сотрудники» можно удалять и редактировать их профили. Нажав, правую кнопку мыши вызывается контекстное меню, которое дает доступ к этим функциям (рисунок 3.25).

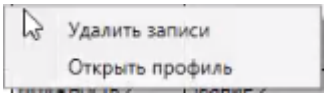

Рисунок 3.25 – Контекстное меню окна «Сотрудники»

При нажатии на «Удалить записи» стирает информацию о сотруднике. А при нажатии на «Открыть профиль» открывается окно «Профиль сотрудника» (рисунок 3.26). Здесь выводится вся информация о выбранном сотруднике.

В первом блоке, окно «Сотрудник», можно увидеть фотографию сотрудника, его ФИО, должность, звание и отдел. Этот блок представляет собой окно «Сотрудник», описанный выше.

Во втором блоке, окно «Управление», выводится информация о рабочей смене сотрудника. В поле «Начало» выводится время входа, а в поле «Конец» выводится время выхода, так в поле «Продолжительность» выводится о время, проведённое между началом и концом сессии.

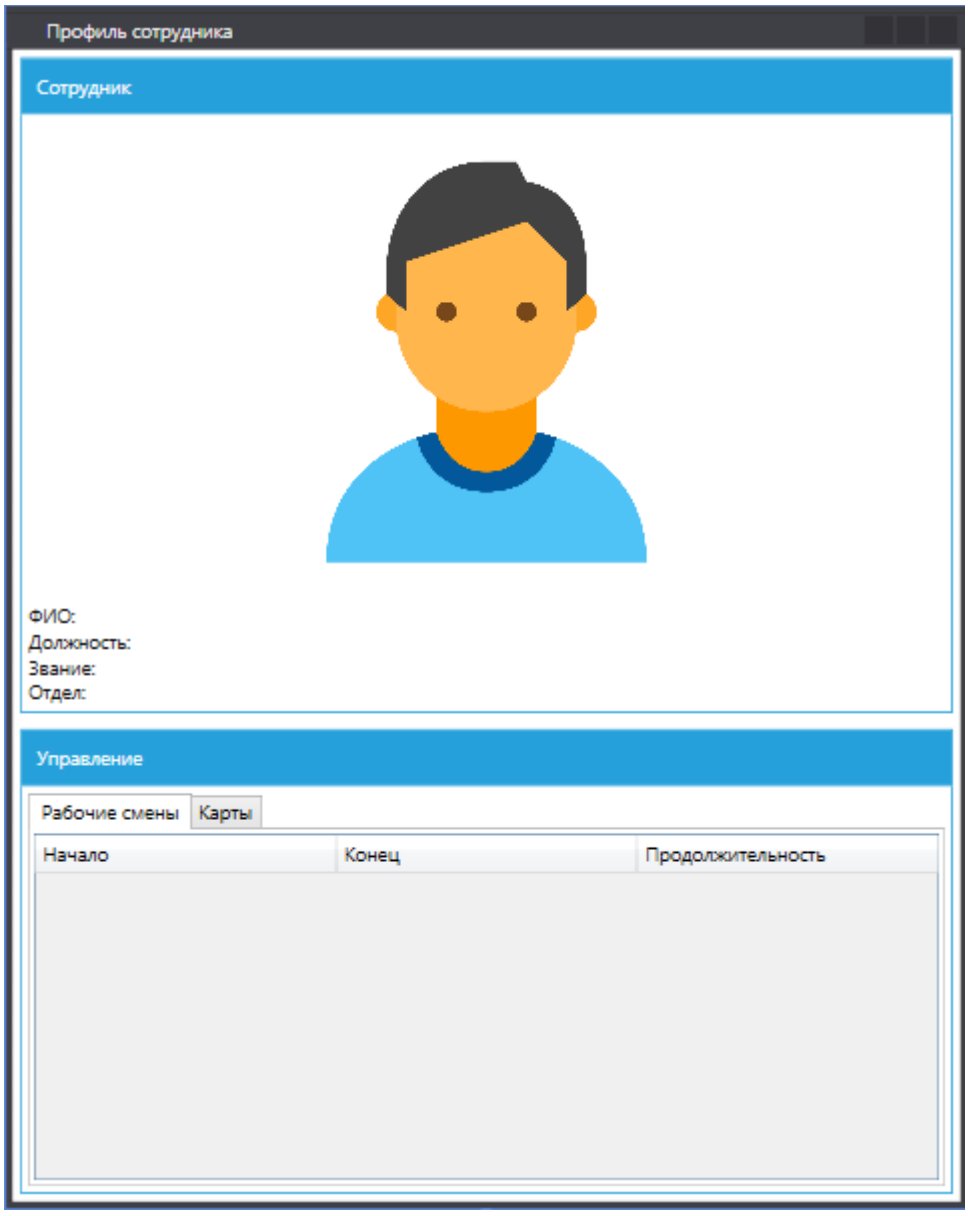

Рисунок 3.26 – Окно «Профиль сотрудника»

При нажатии на «Карты» производится переход в раздел, где выводится информация о привязанных на сотрудника картах (рисунок 3.27), это его UID, дата выдачи и дата истечения. Тут при нажатии правой кнопки мыши на выбранную карту выводится контекстное меню, где можно удалить её.

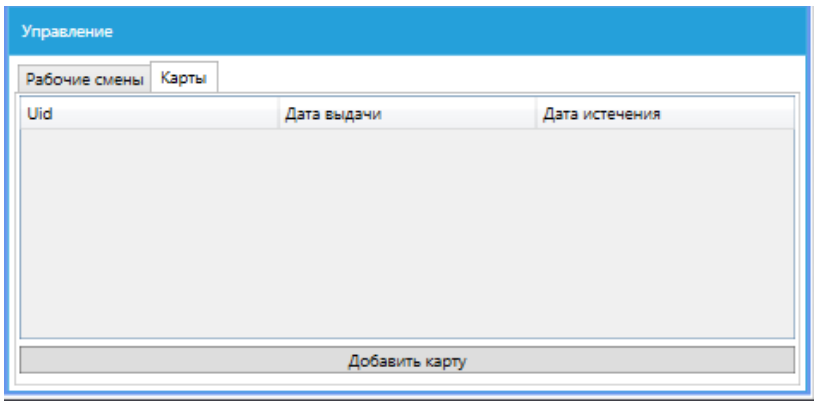

Рисунок 3.27 – Раздел «Карты» окна «Управление»

Также присутствует кнопка «Добавить карту», которая вызывает окно «Добавление карты» (рисунок 3.28), где можно добавить дополнительную карту сотруднику, выбрав срок действия.

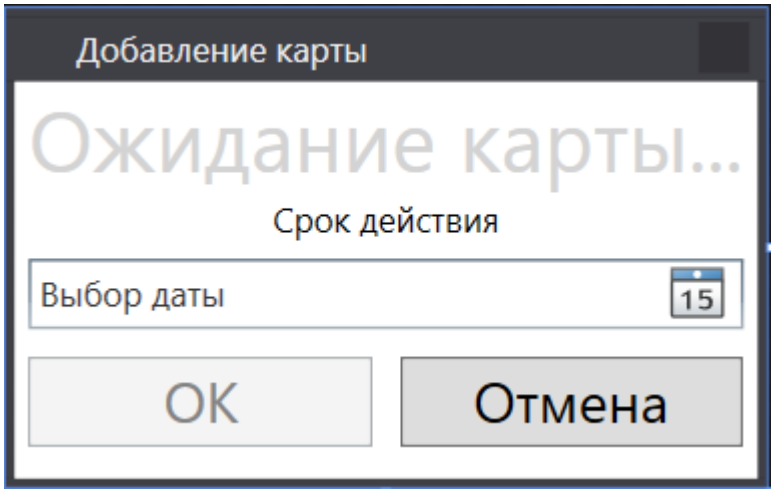

Рисунок 3.28 – Окно «Добавление карты»

3.3 Установка связи между программной и аппаратной частью

В данном разделе описывается взаимосвязь между Arduino и ПО, спроектирована в среде разработки MS Visual Studio на языке программирования C#.

Процесс установки связи, постройка пакетов данных и логика проверки пропусков, определения разрешения прохода и сохранения информации происходит в классах «ArduinoHardware.cs», «PackageBuilder.cs» и «AccessController.cs»

Класс «ArduinoHardware.cs» описывает установку связи между программной и аппаратной частью, считывание карт и управления турникетом.

Обмен данными с Arduino производится через COM-порт пакетами данных, скорость передачи составляет 9600 бит/с (рисунок 3.29).

```
private ArduinoGateway(string port)
{
    _builder = new PackageBuilder(new byte[] { 0x37, 0x83 }, 5);
    _builder.PackageReceived += _builder_PackageReceived;

    _port = new SerialPort(port, 9600);
    _port.Open();
    _port.DataReceived += _port_DataReceived;
}
```

Рисунок 3.29 – Связь с микроконтроллером

После считывания пакета данных, программа получает UID карты сотрудника и информацию о направлении движения, где операторы условий реагируют на три состояния движения (рисунок 3.30).

```
private void _builder_PackageReceived(object sender, byte[] package)
{
    // Получаем UID карты
    ulong uid = BitConverter.ToUInt32(package, 0);

    // Получаем направление движения
    Direction dir;
    byte dirByte = package[sizeof(int)];
    if (dirByte == 0)
        dir = Direction.NONE;
    else if (dirByte == 1)
        dir = Direction.IN;
    else if (dirByte == 2)
        dir = Direction.OUT;
    else
        throw new Exception("Invalid direction value");


    if (CardRead != null)
        CardRead(uid);
    else if (AccessRequested != null && dir != Direction.NONE)
        AccessRequested(uid, dir);
}
```

Рисунок 3.30 – Получение IUD и направления движения

Затем описывается реакция ПО на определённый статус прохода разрешающий или запрещающий доступ (рисунок 3.31).

```
public void SendResponse(AccessStatus status)
{
    if (status == AccessStatus.GRANTED)
        _sendPacket[0] = 1;
    else if (status == AccessStatus.DENIED)
        _sendPacket[0] = 2;
    else
        _sendPacket[0] = 0;

    _port.Write(_sendPacket, 0, _sendPacket.Length);
}
```

Рисунок 3.31 – Статус доступа

В классе «PackageBuilder.cs» описывается логика приходящих по универсальном асинхронному приёмопередатчику пакетов, обнаруживая заголовок, имеется также возможность сборки пакетов из кусков, разделённый на несколько приемов.

Автомат приема - этопроцесс постройки пакета, полученного кусками (рисунок 3.32).

```
public PackageBuilder(byte[] header, int packageLength)
{
    _packageHeder = new byte[header.Length];
    Array.Copy(header, _packageHeder, header.Length);

    _packageLength = packageLength;

    _receivedQueue = new List<byte>();
    _currentHeaderByte = 0;
    _state = ReceiveState.RECEIVING_HEADER;
}
```

Рисунок 3.32 – Автомат приема

При приходе пакета кусками процесс разбивается на несколько этапов, это ожидание заголовка, определение и прочтение. Если же длина заголовка совпадает с текущим заголовком, то заголовок полностью считывается (рисунок 3.33).

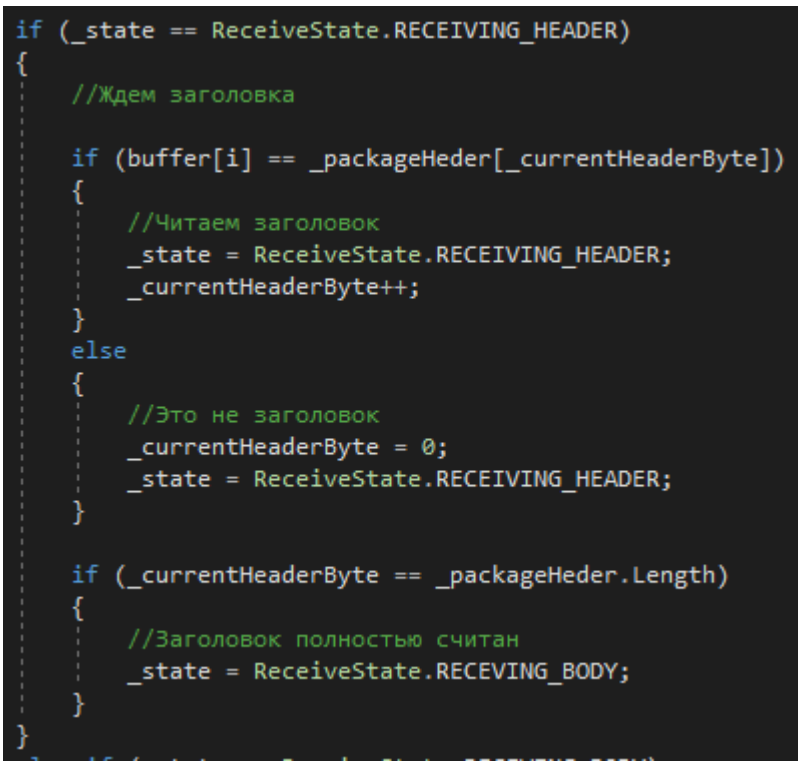

Рисунок 3.33 – Процесс сбора пакетов из кусков

При условии, что прочитанная программой информация равна пакету данных, то пакет считывается полностью (рисунок 3.34).

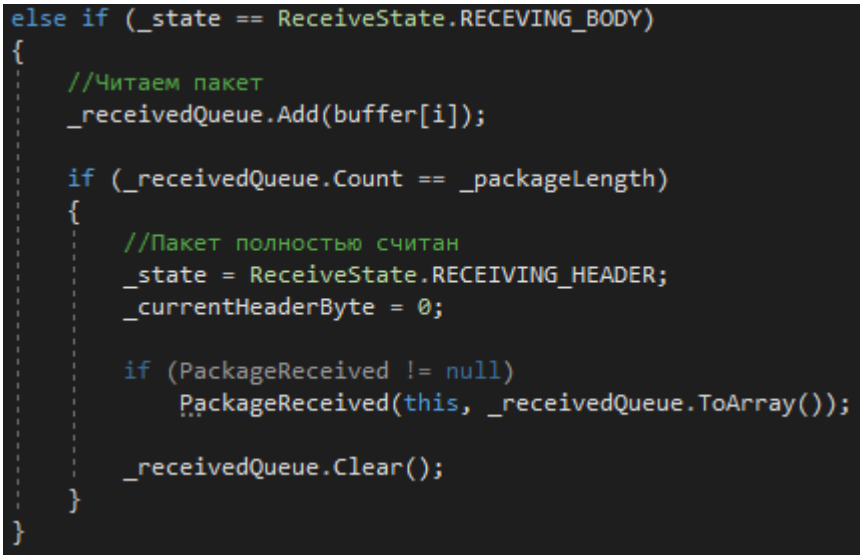

Рисунок 3.34 – Считывание пакетов

В классе «AccessController.cs» описывается основная логика поверки пропусков, определения разрешения на доступ, а также функции записи времени входа и выхода.

Если при условии, что карта пользователя определена, то программа даёт доступ перехода и в зависимости от направления движения записывается время прихода или ухода и UID пользователя. В противном случае система запрещает доступ. Полученная информация записывается в историю событий (рисунок 3.35).

```
if (User != null)
{
    if (direction == Domain.Direction.IN)
    {
        var ws = new WorkShift()
        {
            ArrivalTime = DateTime.Now,
            CardUid = (long)uid
        };
        _ctx.WorkShifts.Add(ws);
        _ctx.SaveChanges();
    }
    else
    {
        long id = (long)uid;
        var last = _ctx.WorkShifts.Where(x => x.CardUid == id).OrderByDescending(x => x.Id)
            .FirstOrDefault();
        if (last != null)
        {
            last.LeavingTime = DateTime.Now;
            _ctx.SaveChanges();
        }
    }

    Access = AccessStatus.GRANTED;
    Direction = direction;
}
```

Рисунок 3.35 – Условие предоставления доступа

4 Экономический расчет

4.1 Цели и задачи, решаемые в экономической части

Темой дипломного проекта является «Разработка информационной системы пропускного режима образовательного учреждения»

В данном дипломном проекте описывается разработка информационной системы пропускного режима, которая должна идентифицировать лица, у которых есть доступ на заданную территорию, и ограничить доступ тех, у кого его нет. В результате расчетов находится себестоимость прикладной программы.

Для нахождения себестоимости необходимо учесть:

- трудоемкость разработки программного продукта;
- материальные затраты;
- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов;

4.2 Расчет трудоемкости разработки ПП

Таблица 4.1 - Распределение работ по этапам и видам и оценка их трудоемкости

Этап разработки	Вид работы	Трудоемкость разработки, чел.× ч.
1	Составление задачи	20
2	Разработка алгоритмов и блок схем	20
3	Создание схемы и кода Arduino	30
4	Реализация клиентской части	90
5	Разработка администраторской части проекта	100
6	Отладка и тестирование программы	30
7	Документация	35
ИТОГО		325

Чтобы определить сколько это будет в днях, нам нужно разделить итоговое число на 8(рабочий день): 325 / 8 = 40

4.3 Расчет затрат на разработку ПП

Определение затрат на разработку ПП производится путем составления соответствующей сметы, которая включает следующие статьи:

- материальные затраты;
- затраты на оплату труда;

- социальный налог;
- амортизация основных фондов;
- прочие затраты.

Таблица 4.2 - Затраты на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество	Цена за единицу, тг	Сумма, тг
Ноутбук Fujitsu nh532	шт.	1	212000	212000
Набор Arduino	шт.	1	16000	16000
Набор RFID считывателя на Mifare RC522	шт.	1	880	880
ИТОГО				228880,00

Общая сумма затрат на материальные ресурсы ($З_М$) определяется по формуле:

$$З_М = \sum_{i=1}^{n} P_i * Ц\,, \quad (4.1)$$

где, P_i - расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ - цена за единицу i-го вида материального ресурса, тг;

i - вид материального ресурса;

n - количество видов материальных ресурсов.

Если для разработки ПП используется электрооборудование, то необходимо рассчитать затраты на электроэнергию по форме, приведенной в таблице 4.3.2.

Общая сумма затрат на электроэнергию ($З_Э$) рассчитывается по формуле:

$$З_э = \sum_{i=1}^{n} M_i * K_i * T_i * Ц\,, \quad (4.2)$$

где, M_i - паспортная мощность i-го электрооборудования, кВт;

K_i - коэффициент использования мощности i-го электрооборудования (принимается K_i=0.9);

T_i - время работы i-го оборудования за весь период разработки ПП ч;

Ц - цена электроэнергии, тг/кВт×ч;

i - вид электрооборудования;

n - количество электрооборудования.

Затраты на электроэнергию находятся исходя из продолжительности периода разработки ПП, количества кВт/ч, затраченных на проектирование ПП и тарифа за 1 кВт/ч. Тариф по городу Алматы для юридических лиц в 2019 году составляет 19,17 тенге за 1 кВт/ч с учетом НДС (согласно данным представленным на официальном сайте ТОО «АлматыЭнергоСбыт»).

$$З_э=0,8\cdot0,9\cdot325\cdot19,17\approx4485,78 \text{ тг}$$

$$З_э = 0.3 \cdot 0,7 \cdot 325 \cdot 19,17 \approx 1308,35$$

Таблица 4.3 - Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электро энергии, тг/кВт·ч	Сумма, тг
Ноутбук Fujitsu nh532	0,8	0,9	325	19,17	4485,78
Освещение	0,3	0,7	325	19,17	1308,35
ИТОГО					5794,13

4.4 Расчет затрат на оплату труда

Заработная плата Middle Full stack разработчика в 2019 состовляет 200000 тенге (для города Алматы).

Рабочие часы сотрудника за месяц определяются по формуле:

$$Ч_м = N_м \cdot Ч_{рд}, \quad (4.3)$$

где, $Ч_м$- рабочие часы сотрудника за месяц;
$N_м$-количество рабочих дней за месяц;
$Ч_{рд}$-количество рабочих часов в день.

$$Ч_м = 22 \cdot 8 = 176 \text{ ч}$$

Ставка работника за час работы рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i}, \quad (4.4)$$

Инженер-разработчик:

$$ЧС_i = \frac{200000}{176} = 1136{,}36 \text{ тг}$$

где, $ЗП_i$ - заработная плата в месяц i-го работника, тг;

$ФРВ_i$ - фонд рабочего времени в месяц i-го работника, час.

Для определения трудоемкости разработки ПП используются данные из таблицы 4.2.1.

Трудоемкость разработки Middle Full stack разработчика

$$T_2 = 20 + 20 + 30 + 90 + 100 + 30 + 35 = 325 чел.\times ч.$$

Затраты на оплату труда в сумме ($З_{ТР}$) определяется по формуле:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i * T_i \,, \qquad (4.5)$$

где, $ЧС_i$ - часовая ставка i-го работника, тг;

T_i - трудоемкость разработки ПП, чел.×ч;

i - категория работника;

n - количество работников, занятых разработкой ПП.

Инженер-разработчик:

$$З_{тр} = 1136.36 \cdot 325 = 369317.00$$

Таблица 4.4 - Затраты на оплату труда

Квалификация	Трудоемкость разработки ПП, чел.×ч	Часовая ставка, тг/ч	Сумма, тг
Middle Full stack разработчик	325	1136,36	369317.00
ИТОГО			369317.00

Таблица 4.5 – Затраты на налоговые отчисления

Уплаченные налоги юридическим лицом	10,46	ФОТ	406248
СО (Социальные отчисления)	3,5	(ЗП - ОПВ)*3,5%	12796,83
ВОСМСЮ (Отчисления на ВОСМСЮ)	2,0	ЗП*2%	8124,97

Продолжение таблицы 4.5

СН (Социальный налог)	9,5	(ЗП - ОПВ - ВОСМСФ)*9,5%-СО	33962,39
Всего уплаченные налоги			54884,19

4.5 Расчет затрат по социальному налогу

Согласно Налоговому кодексу Республики Казахстан, социальный налог составляет 9,5% от фонта оплаты труда. Налоги, уплаченные за работников юридическим лицом можно рассчитать по следующей формуле:

Таблица 4.6 – Формулы расчета налогов

Уплаченные налоги юридическим лицом	10,46	ФОТ	369317
СО (Социальные отчисления)	3,5	(ЗП - ОПВ)*3,5%	11633,49
ВОСМСЮ (Отчисления на ВОСМСЮ)	2,0	ЗП*2%	7386,34
СН (Социальный налог)	9,5	(ЗП - ОПВ - ВОСМСФ)*9,5%-СО	19241,41
Всего уплаченные налоги			**38261,24**

Результаты расчетов представлены в таблице (4.7):

Таблица 4.7 – Начисление социального налога

Категория работника	Количество человек	Заработная плата, тг	Пенсионные отчисления,тг	Социальный налог, тг
Разработчик	1	369317	36931,7	38261,24
Итого:				**38261,24**

4.6 Амортизация основных фондов и прочие затраты

Нормы амортизации ОФ необходимо определить в соответствии с налоговым кодексом РК. Амортизацию ОФ можно определить по следующей формуле:

$$A_{\text{г}} = \frac{\text{С}_{\text{об}} * \text{Н}_{\text{а}}}{100} \quad (4.8)$$

где, $\text{С}_{\text{об}}$ – стоимость оборудования;

$\text{Н}_{\text{а}}$ – норма амортизации (норма амортизация = 25);

Формула (4.8) позволяет рассчитать нужную сумму для амортизационных отчислений за год для ноутбука:

$$A_{\text{г}} = \frac{212\,000 * 25}{100} = 53\,000 \text{ тенге}$$

Теперь необходимо рассчитать норму амортизации за период разработки:

$$A_{\text{г}} = \frac{53000 * 34}{365} = 4936{,}97 \text{ тенге}$$

Подобным образом необходимо рассчитать норму амортизации для всего оборудования. Результаты расчетов приведены в таблице (4.6).

Таблица 4.8 – Амортизация ОФ

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Сумма амортизации за год, тг	Сумма амортизации за время разработки, тг
Ноутбук	212 000	25	53 000	4936,97
Arduino	16 000	25	4000	32,87
RFID	880	25	220	13,56
Итого:			77050	4983,4

Смета расходов на разработку ПО.

На основе всех представленных расчетов необходимо оформить смету расходов на разработку ПО согласно форме, которая приведена в таблице (4.7). На рисунке () продемонстрирована диаграмма рабочих расходов.

Таблица 4.9 – Смета затрат на разработку ПО

Статьи затрат	Сумма, тг
Затраты на оборудование	228880
Затраты на программное обеспечение	0
Затраты на оплату труда	369317
Социальные налоги	38261,24
Затраты на электроэнергию	5794,13
Амортизация основных фондов	4983,4
Прочие расходы	
Итого по смете:	647235,77

Рисунок 4.1 – Диаграмма затрат

4.7 Определение возможной (договорной) цены ПО

Стоимость программного обеспечения определяется на основе качества разработанного продукта, сроков его разработки и производительности продукта. Стоимость $Ц_Д$ для программного обеспечения можно рассчитать по следующей формуле:

$$Ц_Д = З_{нир}\left(1 + \frac{P}{100}\right), \quad (4.9)$$

где $З_{нир}$ – затраты на разработку программного обеспечения,тг;

Р – средний уровень рентабельности ПО, (%). Данный параметр принят равным 25%.

$$Ц_Д = 647235{,}77\left(1 + \frac{25}{100}\right) = 809044{,}71 \text{ тенге}$$

Далее необходимо определить стоимость реализации с учетом НДС, ставка НДС устанавливается законодательством РК. На 2019 года ставка НДС составляет 12%. Стоимость реализации учитывая НДС можно рассчитать по следующей формуле:

$$Ц_р = Ц_д + Ц_д * НДС, \quad (4.10)$$

$$Ц_р = 809044{,}71 + 809044{,}71 * 0{,}12 = 906130{,}08 \text{ тенге}$$

Данную цену можно округлить до 906130 тенге.

4.8 Сравнительный анализ эксплуатационных затрат до и после внедрения

Рассмотрим список расходов после установки программного обеспечения:

- основная заработная плата Middle Full stack разработчика проекта, включая отчисляемыми налогами;
- материальные ресурсы;
- амортизация основных фондов;
- накладные расходы.

Список расходов до внедрения:

- основная заработная плата 2 охранников, вахтёра, ;
- материальные ресурсы;
- накладные расходы.

4.8.1 Расчет затрат до внедрения ПО

Таблица заработной платы и отчисляемых налогов представлена в таблице 4.10

Таблица 4.10 - Затраты на заработную плату

Сотрудник	Кол-во, чел	Заработная плата в месяц, тг	Заработная плата в год, тг	Социальное отчисление	Социальный налог	ВОСМС
Вахтёр	1	90000	1 080 000	34020	56268	21600
Охрана	2	200000 (400 000)	2 400 000 (4 800 000)	75600 (151200)	125040 (250080)	48000 (96000)
К оплате						6489168

Расчет затрат на необходимое для продукта ПО и оборудование представлен в таблице 4.11

Таблица 4.11 – Расчет затрат на оборудование и ПО, необходимое для проекта

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Ноутбук	Acer Extensa EX215-21-47NN	Штук	1	110000	110000
ТУРНИКЕТ ЭЛ	MEX PERCO-TTR-04.1R	Штук	1	476 000	476 000
Итого					586 000

Затраты, отнесенные к амортизационным отчислениям представлены в таблице 4.10

Таблица 4.10 - Амортизационные отчисления

Оборудование	Количество	Стоимость, тенге	Отчисления
Ноутбук Acer Extensa EX215-21-47NN	1	110 000	2561,64
ТУРНИКЕТ ЭЛ MEX PERCO-TTR-04.1R	1	476 000	11084,93
Итого	2	586 000	13646,57

Общие накладные расходы вычисляются по формуле:

$$З_{Н.Р.} = 6489168 * 0{,}2 = 1\,297833{,}6 \text{ тенге.}$$

В итоге, общие эксплуатационные расходы после внедрения программного обеспечения будут равны,

$$З = 6489168 + 586\,000 + 13646{,}57 + 1\,297833{,}6 = 8386648{,}17 \text{ тенге.}$$

4.8.2 Расчет затрат до внедрения ПО

Таблица заработной платы и отчисляемых налогов представлена в таблице 4.11

Таблица 4.11 - Затраты на заработную плату

Сотрудник	Кол-во, чел	Заработная плата в месяц одного, тг	Заработная плата в год, тг	Социальное отчисление	Социальный налог	ВОСМС
Разработчик	1	369317	4431804	139601,83	230896,98	88636,08
К оплате	4890938,89					

Расчет затрат на необходимое для программного продукта и оборудование представлен в таблице 4.12

Таблица 4.12 – Расчет затрат на оборудование и ПО, необходимое для проекта

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Ноутбук	Acer Extensa EX215-21-47NN	Штук	1	110000	110000
Набор Arduino	Arduino UNO	Штук	1	16000	16000
Набор RFID считывателей	rfid-rc522-v2	Штук	1	880	880
Итого	126880				

Затраты, отнесенные к амортизационным отчислениям представлены в таблице 4.13

Таблица 4.13 - Амортизационные отчисления

Оборудование	Количество	Стоимость, тенге	Отчисления
Acer Extensa EX215-21-47NN	1	110000	2561.64
Набор Arduino UNO	1	16000	372.6
Итого	2	126000	2934.24

Общие накладные расходы рассчитываются по формуле:

$$З_{Н.Р.} = 4890938{,}89 * 0{,}2 = 978187.78 \text{ тенге.}$$

Итого, общие эксплуатационные расходы до внедрения программного обеспечения будут равны,

$$З = 4890938{,}89 + 126880 + 2934.24 + 978187.78 = 5998940.91 \text{ тенге.}$$

Для болей ясности предоставим все данные в сравнительную таблицу 4.14

Таблица 4.14 - Годовые эксплуатационные затраты

Статьи	До внедрения ПП	После внедрения ПП
Годовая заработная плата	8386648.17	5998940.91
Оборудование	586000	126880
Амортизационные отчисления	13646.57	2934.24
Накладные расходы	1297833.6	978187.78
Всего	10224128.34	7106942.93

Годовая экономия денежных средств будет равна согласно расчету:

$$10224128.34 - 7106942.93 = 3117185.41 \text{ тенге.}$$

Также определим суммарные выгоды от внедрения программного продукта.

Технико-экономические показатели позволяют рационально определить целесообразность разработки ПО и ее установки, а также взвесить реальную выгоду компании, как для разработчика системы, так и для ее пользователя.

Также мы можем определить срок окупаемости программного обеспечения в месяцах по формуле:

$$П_{окуп} = 12 * \frac{З_{разработка}}{Э_{годовая}} \quad (4.11)$$

где $П_{окуп.}$ – период окупаемости в месяцах;

$З_{разр.}$ – затраты на разработку программы;

$Э_{годовая}$ – годовая экономия.

$$П_{окуп.} = 12 \cdot \frac{906130}{3117185.41} = 12 \cdot 0{,}2907 = 3.5 \text{ мес.}$$

Предвидя годовую выгоду продукта и годовые эксплуатационные затраты с внедрением программного обеспечения можно определить коэффициент эффективности ($К_{эф.}$) по формуле:

Расчетный коэффициент экономической эффективности капитальных вложений составляет:

$$\mathrm{E_р} = \frac{Э_{уг}}{К} \qquad (4.11)$$

$$E_{\mathrm{р}} = \frac{3117185.41}{906130} * 100\% = 344\% \text{ или } 3{,}44 \text{ тенге.}$$

Коэффициент эффективности указывает на соотношение сэкономленных тенге и потраченных. В заключении можно с уверенностью заявить о том, что на 1 затраченный тенге приходится 3.44 тенге экономии.

Внедрение данного программного продукта в рабочий процесс предприятия gпозволит значительно снизить стоимость приобретения дорогостоящих аналогов на рынке. Также установка данного программного комплекса позволяет сократить штат сотрудников, что в следствии сокращает затраты на выплату налогов, зарплат и прочих денежных расходов. Соответственно, уменьшится объем технического оснащения, амортизационные отчисления и затраты на электроэнергию.

5 Безопасность жизнедеятельности

В данном дипломном проекте описывается разработка информационной системы пропускного режима образовательного учреждения, которое является средством контроля и управления доступом, для организации мер безопасности, учета и анализа входящих и выходящих лиц.

Разработка будет применятся в пункте охраны. В помещении работает от одного до трёх человек поэтому было принято решение о создании комфортных условий труда соблюдая установленные нормы.

5.1 Анализ потенциально опасных и вредных факторов в офисе, воздействующих на персонал

На рабочем месте должны быть предусмотрены меры по защите от возможного воздействия опасных и вредных производственных факторов. Уровни этих факторов не должны превышать предельных значений, предусмотренных правовыми, техническими и санитарно-техническими нормами. Эти нормативные документы обязывают создавать условия труда либо на рабочем месте, при которых влияние опасных и вредных факторов на работников устраняется полностью, либо в допустимых пределах.

На рабочем месте должны быть предусмотрены меры по защите от возможного воздействия опасных и вредных производственных факторов. Уровни этих факторов не должны превышать предельных значений, предусмотренных правовыми, техническими и санитарно-техническими нормами. Эти нормативные документы обязывают создавать условия труда либо на рабочем месте, при которых влияние опасных и вредных факторов на работников устраняется полностью, либо в допустимых пределах.

В таких условиях труда наиболее характерными, вредными и опасными факторами для здоровья сотрудника, работающего в офисе, является долговременная сидячая работа.

При работе в положении сидя предъявляются те же основные требования: наиболее рациональная сидячая поза (прямое туловище, согнутые под прямым углом ноги в тазобедренных и коленных суставах). Такая поза обеспечивается высотой сиденья, равной расстоянию от пола до коленного сустава. Для сокращения статического напряжения мышц туловища рабочий стул обязательно снабжается спинкой с опорой на нее на уровне верхних поясничных и нижних грудных позвонков. Если во время работы руки не опираются на стол, верстак или другую 70 рабочую поверхность, то у стула целесообразно делать подлокотники на уровне локтевого сустава слегка отведенной вперед руки. Наиболее целесообразны для работы стулья с поднимающимися сиденьями, спинками и подлокотниками.

Важную роль в повышении работоспособности и профилактике утомления играют тренировки и упражнения, физкультура и спорт.

Поэтому при приеме на работу или переводе рабочего на новую операцию нельзя сразу требовать от него выполнения установленного плана,

ибо в стремлении его выполнить рабочий может привыкнуть к нерациональным или даже неправильным приемам, излишним движениям, что будет снижать его работоспособность и быстро приводить к утомлению. Переучить себя работать рационально всегда труднее, чем с самого начала постепенно освоить правильные приемы [4].

Статические перегрузки. Длительная и постоянная работа на ПЭВМ приводит к болезни рук, спины, плеч и шеи, а также тенденитам (воспалением тканей сухожилия) в результате длительных статических нагрузок, и связаны с использованием клавиатуры. Длительное пребывание в сидячем положении при работе с ПЭВМ приводит к перенапряжению мышц спины и ног. Это происходит в основном из-за нерациональной высоты рабочей поверхности стола и кресла, отсутствия опорной спинки и подлокотников, неудобное размещение рабочих документов, ПЭВМ и клавиатуры, отсутствия подставки для ног [5].

Остеохондроз – самое частое заболевание офисных работников. Боли в спине и шее наблюдаются примерно у 30% всех офисных работников в той или иной степени. Неприятные симптомы могут приходить неожиданно и также неожиданно исчезать.

В таких условиях труда наиболее характерными, вредными и опасными факторами для здоровья сотрудника, работающего в офисе, является небольшой уровень шума на рабочем месте. Он возникает в результате работы кулеров, принтеров и вентиляторов.

В течение рабочего дня шум на рабочем месте возникает постоянно. Постоянный шум характеризуется уровнем звукового давления в октановых полосах со среднегеометрическими частотами. Шум не только ухудшает внимание работников, но и утомляет их. При превышении допустимого уровня шум может сделать условия невозможными для работы.

Следующим важным фактором работы является требования к микроклимату помещения.

Работу офисных сотрудников привыкли считать одним из самых несложных видов труда. Но умственный труд, как и физический, изнуряет любого человека. Более того, работники офисов гораздо чаще подвержены стрессам, главной причиной которых являются умственные, зрительные, психологические перегрузки.

Нередко можно встретить офисы, где в одном кабинете работают по десять-пятнадцать человек. Происходит перегрузка не только оргтехникой, но и людьми, а это значительно ухудшает микроклимат помещения, повышает уровень напряженности.

Между тем, в помещениях, где работают на персональных компьютерах, температура, относительная влажность и скорость движения воздуха на рабочих местах должны соответствовать допустимым санитарным нормам микроклимата помещений (Санитарные нормы микроклимата производственных помещений № 1.02.006-94, утвержденные главным

государственным санитарным врачом РК Дерновым А. Г. 22 августа 1994 года, далее – Санитарные нормы микроклимата),

Рабочее место – это место, на котором работающий находится большую часть (более 50 %, или более двух часов непрерывно) своего рабочего времени. Если при этом работа осуществляется в различных пунктах рабочей зоны, постоянным рабочим местом считается вся рабочая зона.

Согласно постановлению Правительства РК «Об утверждении Санитарных правил "Санитарно-эпидемиологические требования к содержанию и эксплуатации жилых и других помещений, общественных зданий"» от 1 декабря 2011 года № 1431 (далее – Санитарно-эпидемиологические требования № 1431), площадь помещений административных зданий принимается из расчета 6 кв. м на одного работника, а для работающих инвалидов, пользующихся креслами-колясками – 5,65 и 7,65 кв. м соответственно. В таблице 1 приведены рекомендуемые расчетные нормативы площади и оснащенности разных типов помещений в административном здании (под административным зданием в Санитарно-эпидемиологических требованиях № 1431 понимаются в числе прочего строения с офисными помещениями) [4].

Следующим важным фактором является пожарная безопасность офисных помещений.

Для правильной организации противопожарных мер и оснащения офиса установками пожаротушения, необходимо изначально определить категорию помещения по взрывопожароопасности. Отнесение к одной из категорий осуществляется путем проверки помещения и расчета с помощью таблиц и формул, указанных в нормативных сводах.

Обычно офисы относятся к категории В и считаются пожароопасными. Такая классификация связана с наличием в офисных помещениях горючих и трудногорючих материалов и веществ, которые при контакте с воздухом горят без образования взрывоопасных смесей. Требования к микроклимату помещения.

Формирование перечня мероприятий по пожарной безопасности офиса зависит еще от нескольких факторов: места размещения помещения в здании, размеров офиса, числа сотрудников, количества электроприборов, наличия складов и производственных материалов, находящихся рядом с офисом.

Анализ полученных данных позволяет выработать ряд требований, выдвигаемых к офисным помещениям:

- наличие инструкции о мерах пожарной безопасности;
- оборудование системой пожарной сигнализации;
- оснащение офиса системами и средствами пожаротушения согласно классу помещения;
- организация эвакуационных путей [5].

План помещения, где будет применяться разработка, изображен на рисунке 5.1. В помещение в дневное время суток будут находиться 2 человека:

два сотрудника охраны. Охранникам потребуется благоприятные условия труда для долговременного нахождения в условиях закрытого помещения, долговременной сидячей работы, которое должно обеспечить высокую работоспособность, поддержание их здоровья и самочувствия.

Помещение, где работает охрана, содержит надлежащие габариты: длина – 6 м, ширина – 4 м, высота – 3 м, высота окошка – 1 м. Присутствует как натуральное, например и искусственного происхождения освещение. Присутствует 1 окно площадью 1,2х1 метр. Работники пространства размещаются между собой на расстоянии не наименее 1,5 м.

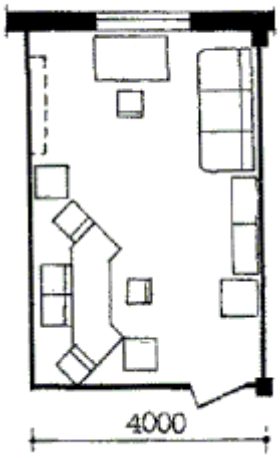

Рисунок 5.1 – План помещения

Информация о помещении охраны, где будет применяться разрабатываемый проект:

- Ширина помещения 4 метров, а длина 6 метров;
- Высота помещения 3 метра;
- В помещении имеется 1 окно, ширина одного окна равна 1200 мм;
- Вход в помещение возможен с помощью двери;

5.2 Расчет систем кондиционирования рабочего помещения.

Расчет кондиционирования необходим для обеспечения соответствия температурного режима в помещении нормам и обеспечить комфортные условия для сотрудников.

Выводимый воздух из помещения $t_{ВЫТ}$, ˚С,

$$t_{ВЫТ} = t_{РЗ} + \Delta t \cdot (h - z) \quad (5.1)$$

где $t_{РЗ}$ – оптимальная температура для работы ($t_{РЗ} \leq t_{ДОП}$), ˚С;

Для теплого периода года $t_{РЗ}$ = 22 ˚С;

h - расстояние от пола до вытяжки кондиционера на потолке h = 3,5 м;
Δt - температурный градиент по высоте помещения, ˚С;
z- высота рабочего помещения, z = 3 м.

$$t_{\text{ВЫТ}} = 22 + 1{,}4 \times (3{,}5 - 3) = 22{,}7\ ^{\circ}\text{C}$$

Видно, что присутствует избыток явной теплоты. Поэтому для нормализации температурного режима и поддержке на комфортном уровне, необходимо подавать воздух ($t_{\text{ПР}}$) на 5-7˚С ниже температуры воздуха в помещении.

$$t_{\text{ПР}} = 22 - 6 = 16^{\circ}\text{C}$$

Произвел расчет количества приточного воздуха $L_{ПР}$,м^3/ч:

$$L_{\text{ПР}} = \frac{Q_{\text{ИЗБ}}}{c \cdot p_{\text{ПР}} \cdot (t_{\text{ВЫТ}} - t_{\text{ПР}})} \quad (5.2)$$

где $p_{\text{ПР}}$ - 1,2 кг/м^3 - плотность поступающего в помещение воздуха;
с - 1кДж/(кг × ˚С) – удельная теплоемкость воздуха при постоянном давлении;
$t_{\text{ВЫТ}}$ - температура воздушной массы, удаляемой из рабочего пространства, ˚С;
$Q_{\text{ИЗБ}}$ – избыточное выделение явной теплоты, кДж/ч;
$t_{\text{ПР}}$ - температура приточного воздуха в рабочее помещение, ˚С.

Определил величину избыточного выделения явной теплоты $Q_{\text{ИЗБ}}$:

$$Q_{\text{ИЗБ}} = \sum Q - \sum Q_{YX} \quad (5.3)$$

где $\sum Q$ - суммарное количество поступающей явной теплоты из вне и выделяемой явной теплоты внутри помещения;
$\sum Q_{YX}$ - суммарное количество выводящейся из помещения теплоты.

Тепловыделения от искусственного освещения Q_2:

$$Q_2 = 1000 \times N \quad (5.4)$$

N – Мощность освещения, Вт.

$$Q_2 = 1000 \times 0{,}25 \times 2 = 500\ \text{Вт}$$

Тепловыделение от людей Q_3:

$$Q_З = n \times q \quad (5.5)$$

где n - число сотрудников;

q - количество тепла, от одного человека, Вт.

Таблица 5.1 - Количество тепла, от одного человека в зависимости от категории работ и температуры окружающей среды.

Категория выполняемых работ	Выделяемое тепло			
	Полное		Явное	
	10 ˚С	35 ˚С	10 ˚С	35 ˚С
Легкая	160 Вт	140 Вт	155 Вт	135 Вт

$$Q_З = 2 \times 140 = 280 \text{ Вт}$$

Количество тепла от солнечной радиации $Q_{ОСТ.РАД}$:

$$Q_{ОСТ.РАД} = F_{ОСТ} \times q_{ОСТ} \times A_{ОСТ} \quad (5.6)$$

Количество тепла от солнечной радиации в зависимости от внешнего покрытия:

$$Q_{П.РАД} = F_n \times q_n \times k_n \quad (5.7)$$

где $F_{ОСТ}$ и Fn - площадь через которое поступает солнечная радиация, м^2;

$q_{ОСТ}$ и q_n - теплопоступления через 1 м^2 поверхности. коэффициент теплопередачи равен 1 Вт/(м^2 × ˚С);

$A_{ОСТ}$ - коэффициент остекления рабочего помещения;

k_n - коэффициент теплопередачи покрытия, Вт/(м^2 × ˚С).

$q_{ОСТ}$ - принимается в пределах 70 – 210;

$A_{ОСТ}$ – принимается в пределах 0,25 - 1,25.

$$F_{ОСТ} = 1{,}2 \text{ м}^2$$

$q_{ОСТ}$ = 160 Вт/(м^2 × ˚С);

$A_{ОСТ}$ = 0,3 – окна в помещении не затемнены;

$$Q_{ОСТ.РАД} = 1{,}2 \times 160 \times 0{,}3 = 57{,}6 \text{ Вт}$$

$Q_{ПРАД}$ = 17 Вт - среднее теплопоступление покрытия;

$$Q_{П.РАД} = F_n \times q_n \times k_n \quad (5.8)$$

$$Q_{П.РАД} = 1,2 \times 1 \times 17 = 20,4 \text{ Вт}$$

Количество тепла которое уходит из рабочего помещения $Q_{УХ}$, кВт:

$$Q_{УХ} = \frac{\lambda \cdot S \cdot (t_{ВЫТ} - t_{ПР})}{\delta} \quad (5.9)$$

где λ - теплопроводность стен, Вт/(м × ˚С);

S - площадь, м^2;

δ - толщина стен, м;

δ = 0,35м толщина стен выполненных из бетона К750 с теплопроводностью 1,2 Вт/(м × ˚С).

$$Q_{УХ} = \frac{1,2 \cdot 24 \cdot (22,7 - 16)}{0,35} = 7581 \text{ Вт}$$

Сумма входящей теплоты в помещение:

$$\sum Q = Q_2 + Q_3 + Q_{ОСТ.РАД} + Q_{П.РАД} \quad (5.10)$$

$$\sum Q = 5250 + 2940 + 2400 + 85 = 10\,675 \text{ Вт}$$

Избыточное выделение явной теплоты:

$$Q_{ИЗБ} = 10\,675 - 7581 = 3094$$

Количество приточного воздуха необходимого для поддержания комфортных условий:

$$L_{пр} = \frac{3094}{1 \cdot 1,2 \cdot (22,7 - 16)} = 384,82 \ \frac{м^3}{ч}$$

Т.к. имеется избыточное выделение теплоты, рассчитал, что приведения текущих условий труда в комфортные и соответствию нормативам нужно установить кондиционер.

Выбрал Gree-42 R410A. Кассетная сплит – система с возможностью обогрева и охлаждения.

Таблица 5.2 Характеристики сплит-системы Gree-42

1	Мощность в режиме охлаждения	12 000 Вт
2	Мощность в режиме обогрева	13 500 Вт

Продолжение таблицы 5.2

3	Потребляемая мощность при охлаждении	4 200 Вт
4	Потребляемая мощность при обогреве	4 000 Вт
5	Уровень шума (мин/макс)	43/47 Дб
6	Обслуживаемая площадь	126 кв.м
7	Внутренний блок сплит-системы	850×850×325 мм
8	Наружный блок сплит-системы	1120×440×1100 мм

5.3 Расчет систем пожаротушения

Таблица 5.3 - Группы помещений

Группа помещений	Перечень характерных помещений, производств, технологических процессов
1	Помещения книгохранилищ, библиотек, цирков, хранения сгораемых музейных ценностей, фондохранилищ, музеев и выставок, картинных галерей, концертных и киноконцертных залов, ЭВМ, магазинов, зданий управлений, гостиниц, больниц

Таблица 5.4 – Исходные данные для группы помещения 1.

Группа помещений		1
Интенсивность орошения защищаемой площади, л/(с·м^2), не менее	Водой	0,08
	Раствором пенообразователя	-
Расход, л/с, не менее	Воды	10
	Раствора пенообразователя	-
Минимальная площадь спринклерной АУП, м^2, не менее		60
Продолжительность подачи воды, мин, не менее		30
Максимальное расстояние между спринклерными оросителями, м		4

Таблица 5.5 – Технические характеристики СВУ

Коэффициент производительности	0,60	0,77	0,84
Диапазон рабочего давления, МПа	0,05 – 1,00		
Защищаемая площадь, м^2	12		
Интенсивность орошения при 0,1(0,3) МПа	0,095(0,175)	0,120(0,200)	0,145(0,215)
Габариты	50x30x27		
Масса, не более, кг	0.055		
Присоединительная резьба	R 1/2		

Продолжение таблицы 5.5

Термочувствительный элемент	DI 937		
Коэффициент тепловой инерционности	>80		
Номинальная температура срабатывания, °С	68		
Номинальное время срабатывания, с	300		
Предельно допустимая рабочая температура, °С	50		
Цвет жидкости в колбе	Красный		
К-фактор, GPM/PSI (LPM/bar^{0,5})	8,0 (115)	10,1 (146,1)	11,0 (160)

Расчет расхода воды через оросителя СВУ с коэффициентом производительности 0,77:

$$q_1 = 10K\sqrt{P} \quad (5.11)$$

$$q_1 = 10 \times 0{,}77\sqrt{0{,}1} = 2{,}43 \text{ л/с}$$

где q_1 - расход оросителя, л/с;

К - коэффициент производительности оросителя, принимаемый по технической документации на изделие, л/(с×МПа^{0,5});

Р - давление перед оросителем, МПа.

Т.к. площадь помещения составляет 24 м^2, а один спринклер рассчитан на 12 м^2, то необходимо установить 2 спринклеров:

$$n = \frac{24}{12} = 2$$

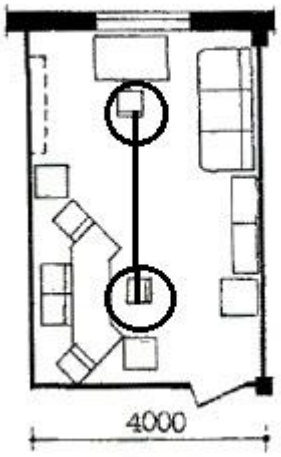

Рисунок 5.2 – Схема размещения оросителей

Заключение

В ходе выполнения дипломного проекта была разработана информационная система пропускного режима, которая поможет в поддержании учебной и трудовой дисциплины, и предоставляет возможности удовлетворяющие минимальные требования к системам контроля и управления доступом в настоящее время.

При реализации дипломного проекта были выполнены следующие задачи:

- был проведен анализ предметной области;
- проанализированы иностранные системы контроля и управления доступом;
- определены основные требования к программе;
- выбраны оптимальные инструменты для разработки;
- разработка устройства для считывания пропускных карт;
- разработка оконного приложения и её визуального интерфейса;
- установка связи между аппаратной и программной частью;
- произведено экономическое обоснование целесообразности разрабатываемого программного продукта, где были сделано следующее заключение:
- цена реализации окупает все затраты, потрачены на разработку. Прибыль от реализации проекта равна 161808,94;
- также были исследованы условия труда в помещении, где возможно будет применяться программа, и были предложены мероприятия по улучшению качества освещения в помещении.

Список литературы

1 Основы безопасности жизнедеятельности: Учебное пособие / сост.: М.К.Дюсебаев, Ж.С.Абдимуратов. -Алматы: АУЭС, 2013.-79 с

2 2. Калюжный Е.А. Безопасность жизнедеятельности: Учебное пособие / Е.А. Калюжный, С.В. Михайлова, С.Г. Напреев, Д.Г. Сидоров. - Арзамас:

3 Абрамов В.В. Безопасность жизнедеятельности: Учебное пособие для вузов. – Спб.: Питер, 2013. – 365 с.

4 https://sber-solutions.kz/press-center/articles/sanitarnye-trebovaniya-k-ofisnym-pomeshcheniyam/

5 https://center-avtomatiki.com/trebovaniya-k-pozharnoy-bezopasnosti-v-ofise/

6 Гинце А. Новые технологии в СКУД // Системы безопасности, 2005.

7 Тарасов Ю Контрольно-пропускной режим на предприятии. Защита информации // Конфидент, 2002. № 1.

8 Горлицин И. Контроль и управление доступом - просто и надежно КТЦ «Охранные системы», 2012.

9 https://habr.com/ru/company/intems/blog/316728/

10 В.А. Ворона В.А. ТихоновСистемы контроля и управления доступом: Москва Горячая линия - Телеком, 2010.

11 Петин В. А., Биняковский А. А. Практическая энциклопедия Arduino, 2017. – 152с.

12 Джон Скит. C# для профессионалов, 2008. – 605с.

13 Шилдт Г. C#: Учебный курс – СПб.: Питер; Киев: BHV, 2003. - 512 с.

14 Лабор В. В. Си Шарп: Создание приложений для Windows/ В. В. Лабор. – Мн.: Харвест, 2003. - 384 с.

15 Michael McRoberts. Beginning Arduino, 2010. – 472с.

16 Максим Власов. RFID: 1 технология – 1000 решений: Практические примеры использования RFID в различных областях. – М.: Альпина Паблишер, 2014. – 218 с.

17 Петцольд Ч. Программирование с использованием Microsoft Windows Forms. Мастер-класс / Пер. с англ. – М.: Русская редакция; СПб.: Питер, 2006. – 432 стр.

18

Приложение А
Техническое задание

1 Общие сведения

1.1 Наименование системы

Полное наименование системы:
Информационная система пропускного режима образовательного учреждения

1.2 Сроки начала и окончания работ

Дата начала: 17.02.2020
Дата окончания: 09.05.2020

2 Назначение и цели создания системы

2.1 Назначение системы

Информационная система пропускного режима предназначена для контроля, управления и учета доступа.

3 Рекомендации к разработке программы

Обучающая система может быть разработана на базе микроконтроллера Arduino, на языке программирования C# с использованием СУБД Microsoft SQL Server.

4 Требование к внешнему виду системы

Обязательным условием является простота, интуитивная понятливость интерфейса, не яркая, приятная цветовая гамма.

4 Технические требования

Специальных технических требований для ПК для работы приложения нет. Подойдёт любой ПК, с операционной системой Windows.

5 Экономические требования

- возможная (договорная) цена продукта составила 906 130 тг;
- стоимость разработки продукта составила 647 235 тг.

Приложение Б

(листинг программы)

Форма MainWindow.xaml.cs

```
using skud.Data;
using skud.Domain;
using skud.Domain.Models;
using skud.ViewModels;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using skud.Views.Windows;

namespace skud
{
   /// <summary>
   /// Логика взаимодействия для MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window, INotifyPropertyChanged
   {
      public AccessController Controller { get; set; }

      public MainWindow()
      {
         InitializeComponent();
         DataContext = this;


         var dlg = new SelectPortWindow();
         if (dlg.ShowDialog() == true)
         {
            string com = dlg.SelectedPort;
            ArduinoGateway.Init(com);
         }
         else
         {
```

```
 MessageBox.Show("COM-порт не выбран!", "Ошибка", MessageBoxButton.OK,
MessageBoxImage.Warning);
            Application.Current.Shutdown();
            return;
        }


        Controller = new AccessController(new SkudContext());
    }

    public event PropertyChangedEventHandler PropertyChanged;


    protected override void OnClosed(EventArgs e)
    {
        base.OnClosed(e);
        Application.Current.Shutdown();
    }

    private void mnuPositions_Click(object sender, RoutedEventArgs e)
    {
        new PositionsWindow().Show();
    }

    private void mnuRanks_Click(object sender, RoutedEventArgs e)
    {
        new RanksWindow().Show();
    }

    private void mnuUsers_Click(object sender, RoutedEventArgs e)
    {
        new UsersWindow().Show();
    }

    private void mnuHistory_Click(object sender, RoutedEventArgs e)
    {
        new HistoryWindow().Show();
    }

    private void MnuDepartments_OnClick(object sender, RoutedEventArgs e)
    {
        new DepartmentsWindow().Show();
    }

    private void Arrows_Loaded(object sender, RoutedEventArgs e)
    {

    }

    private void LogView_Loaded(object sender, RoutedEventArgs e)
```

```
{

        }
    }


}
```

Форма DepartmentsWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using skud.Data;
using skud.Domain.Models;
using skud.Helpers;

namespace skud.Views.Windows
{
    /// <summary>
    /// Логика взаимодействия для DepartmentsWindow.xaml
    /// </summary>
    public partial class DepartmentsWindow : Window
    {
        private SkudContext _ctx;

        public DepartmentsWindow()
        {
            InitializeComponent();
            DataContext = this;
            _ctx = new SkudContext();
            Update();
        }

        private void BtnUpdate_OnClick(object sender, RoutedEventArgs e)
        {
            Update();
        }
```

```
private void BtnSave_OnClick(object sender, RoutedEventArgs e)
        {
            _ctx.SaveChanges();
            Update();
        }

        private void MnuRemoveSelected_OnClick(object sender, RoutedEventArgs e)
        {
            while (grid.SelectedItems.Count > 0)
            {
                var item = (Department)grid.SelectedItems[0];
                _ctx.Departments.Remove(item);
            }
        }

        private void Update()
        {
            EditHelpers.DetachAllEntities<Department>(_ctx);
            _ctx.Departments.Load();
            grid.ItemsSource = _ctx.Departments.Local;
        }
    }
}
```

Форма AddCardWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using skud.Data;
using skud.Domain;
using skud.Domain.Models;

namespace skud.Views.Windows
{
    /// <summary>
    /// Логика взаимодействия для AddCardWindow.xaml
    /// </summary>
    public partial class AddCardWindow : Window, INotifyPropertyChanged
```

```
{
    public ulong? CardUid { get; set; }
    public DateTime ExpirationDate { get; set; }

    private int _userId;
    public Card Card { get; private set; }

    public AddCardWindow(int userId)
    {
        InitializeComponent();
        _userId = userId;
        DataContext = this;
        ExpirationDate = DateTime.Now.AddMonths(1);
        ArduinoGateway.Instance.CardRead += Instance_CardRead;
    }

    private void Instance_CardRead(ulong uid)
    {
        CardUid = uid;
        ArduinoGateway.Instance.SendResponse(AccessStatus.NONE);
    }

    private void BtnCancel_OnClick(object sender, RoutedEventArgs e)
    {
        DialogResult = false;
        Close();
    }

    private void BtnOk_OnClick(object sender, RoutedEventArgs e)
    {
        using (var ctx = new SkudContext())
        {

            if (ctx.Cards.Any(x => x.Uid == (long)CardUid))
            {
                MessageBox.Show("Карта уже зарегестрирована в системе", "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            Card = new Card()
            {
                UserId = _userId,
                IssueDate = DateTime.Now,
                ExpirationDate = ExpirationDate,
                Uid = (long)CardUid
            };


            ctx.Cards.Add(Card);
```

```
ctx.SaveChanges();
            }

            DialogResult = true;
            Close();
        }

        protected override void OnClosed(EventArgs e)
        {
            base.OnClosed(e);
            ArduinoGateway.Instance.CardRead -= Instance_CardRead;
        }

        public event PropertyChangedEventHandler PropertyChanged;
    }
}
```

Форма PositionsWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Data.Entity;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using skud.Data;
using skud.Domain.Models;
using skud.Helpers;

namespace skud.Views.Windows
{
    /// <summary>
    /// Логика взаимодействия для PositionsWindow.xaml
    /// </summary>
    public partial class PositionsWindow : INotifyPropertyChanged
    {
        private SkudContext _ctx;

        public PositionsWindow() : base()
        {
```

```
  InitializeComponent();
        DataContext = this;
        _ctx = new SkudContext();
        Update();
    }

    public event PropertyChangedEventHandler PropertyChanged;

    private void btnUpdate_Click(object sender, RoutedEventArgs e)
    {
        Update();
    }

    private void btnSave_Click(object sender, RoutedEventArgs e)
    {
        _ctx.SaveChanges();
        Update();
    }

    private void MnuRemoveSelected_OnClick(object sender, RoutedEventArgs e)
    {
        while (grid.SelectedItems.Count > 0)
        {
            var item = (Rank)grid.SelectedItems[0];
            _ctx.Ranks.Remove(item);
        }
    }

    private void Update()
    {
        EditHelpers.DetachAllEntities<Position>(_ctx);
        _ctx.Positions.Load();
        grid.ItemsSource = _ctx.Positions.Local;
    }
  }
}
```

Форма UserProfileWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data.Entity;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
```

```
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using skud.Data;
using skud.Domain.Models;

namespace skud.Views.Windows
{
    /// <summary>
    /// Логика взаимодействия для UserProfileWindow.xaml
    /// </summary>
    public partial class UserProfileWindow : Window, INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        // Пользователь
        public User User { get; private set; }

        // Рабочие смены
        public List<WorkShift> Shifts { get; private set; }

        private SkudContext _ctx;
        private int _userId;

        public UserProfileWindow(int userId)
        {
            InitializeComponent();
            DataContext = this;
            _userId = userId;

            // Загружаем пользователя
            _ctx = new SkudContext();
            User =
_ctx.Users.Include(x=>x.Position).Include(x=>x.Department).Include(x=>x.Rank).FirstOrDefault
(x => x.Id == userId);
            if (User == null)
            {
                MessageBox.Show("Пользователь не найден", "Ошибка", MessageBoxButton.OK,
MessageBoxImage.Warning);
                Close();
            }

            // Загружаем рабочие смены пользователя
            Shifts = (from shift in _ctx.WorkShifts
                      join card in _ctx.Cards on shift.CardUid equals card.Uid
                      where card.UserId == userId
                      select shift).ToList();

            // Загружаем карты
```

```
    UpdateCards();
        }

        // Открываем окно добавления кары
        private void BtnAddCard_OnClick(object sender, RoutedEventArgs e)
        {
            var dlg = new AddCardWindow(_userId);
            if (dlg.ShowDialog() ?? false)
            {
                // Если добавление прошло успешно, обновляем список карт
                UpdateCards();
            }
        }

        private void UpdateCards()
        {
            _ctx.Cards.Where(x => x.UserId == _userId).ToList();
            cardsGrid.ItemsSource = _ctx.Cards.Local;
        }

        // Удаление выбранных карт
        private void mnuRemoveSelected_Click(object sender, RoutedEventArgs e)
        {
            while (cardsGrid.SelectedItems.Count > 0)
            {
                var card = (Card)cardsGrid.SelectedItems[0];
                _ctx.Cards.Remove(card);
            }

            _ctx.SaveChanges();
        }
    }
}
```

Форма RanksWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data.Entity;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
```

```
using System.Windows.Shapes;
using skud.Data;
using skud.Domain.Models;
using skud.Helpers;

namespace skud.Views.Windows
{
    /// <summary>
    /// Логика взаимодействия для RanksWindow.xaml
    /// </summary>
    public partial class RanksWindow : Window, INotifyPropertyChanged
    {
        private SkudContext _ctx;

        public RanksWindow()
        {
            InitializeComponent();
            DataContext = this;
            _ctx = new SkudContext();
            Update();
        }

        private void BtnUpdate_OnClick(object sender, RoutedEventArgs e)
        {
            Update();
        }

        private void BtnSave_OnClick(object sender, RoutedEventArgs e)
        {
            _ctx.SaveChanges();
            Update();
        }

        private void MnuRemoveSelected_OnClick(object sender, RoutedEventArgs e)
        {
            while (grid.SelectedItems.Count > 0)
            {
                var item = (Rank)grid.SelectedItems[0];
                _ctx.Ranks.Remove(item);
            }
        }

        private void Update()
        {
            EditHelpers.DetachAllEntities<Rank>(_ctx);
            _ctx.Ranks.Load();
            grid.ItemsSource = _ctx.Ranks.Local;
        }

        public event PropertyChangedEventHandler PropertyChanged;
```

```
    }
}
```

Форма SelectPortWindow.xaml.cs

```
using System.Windows;
using skud.Domain;

namespace skud.Views.Windows
{
    /// <summary>
    /// Логика взаимодействия для SelectPortWindow.xaml
    /// </summary>
    public partial class SelectPortWindow : Window
    {
        public string SelectedPort { get; set; }
        public string[] Ports { get; set; }

        public SelectPortWindow()
        {
            InitializeComponent();
            DataContext = this;
            Ports = ArduinoGateway.GetPorts();
        }


        private void btnSave_Click(object sender, RoutedEventArgs e)
        {
            DialogResult = false;
            Close();
        }

        private void btnCancel_Click(object sender, RoutedEventArgs e)
        {
            DialogResult = false;
            Close();
        }
    }
}
```

Форма UsersWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data.Entity;
using System.Data.Entity.Validation;
using System.Linq;
using System.Windows;
using System.Windows.Forms;
using skud.Data;
using skud.Domain.Models;
using skud.Helpers;
```

```
using MessageBox = System.Windows.MessageBox;
using System.IO;

namespace skud.Views.Windows
{
    /// <summary>
    /// Логика взаимодействия для UsersWindow.xaml
    /// </summary>
    public partial class UsersWindow : Window, INotifyPropertyChanged
    {
        private SkudContext _ctx;

        public List<Department> Departments { get; set; }
        public List<Position> Positions { get; set; }
        public List<Rank> Ranks { get; set; }

        public UsersWindow()
        {
            InitializeComponent();
            DataContext = this;
            _ctx = new SkudContext();
            Update();
        }
        private void BtnUpdate_OnClick(object sender, RoutedEventArgs e)
        {
            Update();
        }

        private void BtnSave_OnClick(object sender, RoutedEventArgs e)
        {
            try
            {
                _ctx.SaveChanges();
                Update();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Данные заполнены неверно", "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Warning);
            }
        }

        private void MnuRemoveSelected_OnClick(object sender, RoutedEventArgs e)
        {
            while (grid.SelectedItems.Count > 0)
            {
                var item = (User)grid.SelectedItems[0];
                _ctx.Users.Remove(item);
            }
        }
```

```
        private void MnuShowProfile_OnClick(object sender, RoutedEventArgs e)
        {
            var user = (User)grid.SelectedItem;
            new UserProfileWindow(user.Id).Show();
        }

        private void Update()
        {
            EditHelpers.DetachAllEntities<User>(_ctx);
            Departments = _ctx.Departments.ToList();
            Positions = _ctx.Positions.ToList();
            Ranks = _ctx.Ranks.ToList();
            _ctx.Users.Include(x => x.Position).Include(x => x.Department).Include(x =>
x.Rank).Load();
            grid.ItemsSource = _ctx.Users.Local;
        }

        private void MnuBrowse_OnClick(object sender, RoutedEventArgs e)
        {
            var dlg = new OpenFileDialog();
            dlg.CheckFileExists = true;

            if (dlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                if(!Directory.Exists("Photos"))
                    Directory.CreateDirectory("Photos");


                string extension = Path.GetExtension(dlg.FileName);
                string saveFile = Path.Combine("Photos", Guid.NewGuid().ToString() + extension);
                File.Copy(dlg.FileName, saveFile);

                var item = (User)grid.SelectedItem;
                item.Photo = saveFile;
                grid.CommitEdit();
            }
            else
            {
                grid.CancelEdit();
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;
    }
}
```

Форма AccessStatus.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace skud.Views.Controls
{
    /// <summary>
    /// Логика взаимодействия для AccessStatus.xaml
    /// </summary>
    public partial class AccessStatus : UserControl
    {
        public AccessStatus()
        {
            InitializeComponent();
        }
    }
}
```

Форма Arrows.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace skud.Views.Controls
{
    /// <summary>
    /// Логика взаимодействия для Arrows.xaml
    /// </summary>
    public partial class Arrows : UserControl
    {
```

```
    public Arrows()
      {
         InitializeComponent();
      }
   }
}
```

Форма LogView.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace skud.Views.Controls
{
   /// <summary>
   /// Логика взаимодействия для LogView.xaml
   /// </summary>
   public partial class LogView : UserControl
   {
      public LogView()
      {
         InitializeComponent();
      }
   }
}
```

Форма UserProfileCard.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
```

```
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace skud.Views.Controls
{
    /// <summary>
    /// Логика взаимодействия для UserProfileCard.xaml
    /// </summary>
    public partial class UserProfileCard : UserControl
    {
        public UserProfileCard()
        {
            InitializeComponent();
        }
    }
}
```

Класс Card.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace skud.Domain.Models
{
    /// <summary>
    /// Пропуск
    /// </summary>
    public class Card
    {

        //public int Id { get; set; }

        /// <summary>
        /// Уникальный идентификатор карты
        /// </summary>
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.None)]
        public long Uid { get; set; }

        /// <summary>
        /// Дата выдачи
        /// </summary>
        public DateTime IssueDate { get; set; }

        /// <summary>
        /// Дата истечения
```

```
 /// </summary>
        public DateTime ExpirationDate { get; set; }

        /// <summary>
        /// Пользователь, кому выдана карта
        /// </summary>
        public User User { get; set; }
        public int UserId { get; set; }
    }
}
```

Класс Department.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace skud.Domain.Models
{
    /// <summary>
    /// Отдел
    /// </summary>
    public class Department
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

Класс Position.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace skud.Domain.Models
{
    /// <summary>
    /// Позиция
    /// </summary>
    public class Position
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

Класс Rank.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace skud.Domain.Models
{
    /// <summary>
    /// Звание
    /// </summary>
    public class Rank
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

Класс User.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace skud.Domain.Models
{
    public class User : INotifyPropertyChanged
    {
        public int Id { get; set; }

        /// <summary>
        /// Фамилия
        /// </summary>
        public string Surname { get; set; }

        /// <summary>
        /// Имя
        /// </summary>
        public string Name { get; set; }

        /// <summary>
        /// Отчество
        /// </summary>
        public string Middlename { get; set; }

        /// <summary>
```

```
/// Имя файла фотографии
        /// </summary>
        public string Photo { get; set; }

        /// <summary>
        /// Должность
        /// </summary>
        public Position Position { get; set; }
        public int PositionId { get; set; }

        /// <summary>
        /// Звание
        /// </summary>
        public Rank Rank { get; set; }
        public int RankId { get; set; }

        /// <summary>
        /// Список пропусков, выданных пользвователю
        /// </summary>
        public ICollection<Card> Cards
        {
            get { return _cards ?? (_cards = new List<Card>()); }
            set { _cards = value; }
        }
        private ICollection<Card> _cards;

        /// <summary>
        /// Отдел
        /// </summary>
        public Department Department { get; set; }
        public int DepartmentId { get; set; }

        public string FIO { get { return String.Format("{0} {1} {2}", Surname, Name, Middlename);
} }




        public event PropertyChangedEventHandler PropertyChanged;
    }
}
```

Класс WorkShift.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace skud.Domain.Models
{
    /// <summary>
    /// Рабочая смена
    /// </summary>
    public class WorkShift
    {
        public int Id { get; set; }

        public long CardUid { get; set; }
        public Card Card { get; set; }

        /// <summary>
        /// Дата и время прихода
        /// </summary>
        public DateTime? ArrivalTime { get; set; }

        /// <summary>
        /// Дата и время ухода
        /// </summary>
        public DateTime? LeavingTime { get; set; }

        public TimeSpan? Duration
        {
            get { return LeavingTime - ArrivalTime; }
        }
    }
}
```

Класс PackageBuilder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace skud.Domain
{
    public delegate void PackageReceivedDelegate(object sender, byte[] package);

    /// <summary>
    /// Собирает приходящие по UART пакеты (обнаруживая заголовок).
    /// Может собрать пакет из кусков, разделенный на несколько приемов
    /// </summary>
    public class PackageBuilder
    {
        public event PackageReceivedDelegate PackageReceived;

        enum ReceiveState
        {
```

```
RECEIVING_HEADER,
        RECEVING_BODY
    };

    //Автомат приема
    private ReceiveState _state;
    private List<byte> _receivedQueue;
    private int _currentHeaderByte;

    private int _packageLength;
    private byte[] _packageHeder;

    public PackageBuilder(byte[] header, int packageLength)
    {
        _packageHeder = new byte[header.Length];
        Array.Copy(header, _packageHeder, header.Length);

        _packageLength = packageLength;

        _receivedQueue = new List<byte>();
        _currentHeaderByte = 0;
        _state = ReceiveState.RECEIVING_HEADER;
    }

    /// <summary>
    /// Обработать очередной кусок принятых данных
    /// </summary>
    /// <param name="buffer"></param>
    public void ProcessPart(byte[] buffer)
    {
        int i = 0;
        while (i < buffer.Length)
        {
            if (_state == ReceiveState.RECEIVING_HEADER)
            {
                //Ждем заголовка

                if (buffer[i] == _packageHeder[_currentHeaderByte])
                {
                    //Читаем заголовок
                    _state = ReceiveState.RECEIVING_HEADER;
                    _currentHeaderByte++;
                }
                else
                {
                    //Это не заголовок
                    _currentHeaderByte = 0;
                    _state = ReceiveState.RECEIVING_HEADER;
                }
```

```
    if (_currentHeaderByte == _packageHeder.Length)
                {
                    //Заголовок полностью считан
                    _state = ReceiveState.RECEVING_BODY;
                }
            }
            else if (_state == ReceiveState.RECEVING_BODY)
            {
                //Читаем пакет
                _receivedQueue.Add(buffer[i]);

                if (_receivedQueue.Count == _packageLength)
                {
                    //Пакет полностью считан
                    _state = ReceiveState.RECEIVING_HEADER;
                    _currentHeaderByte = 0;

                    if (PackageReceived != null)
                        PackageReceived(this, _receivedQueue.ToArray());

                    _receivedQueue.Clear();
                }
            }

            i++;
        }
    }
  }
}
```

Класс Direction.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace skud.Domain
{
   public enum Direction
   {
      NONE, IN, OUT
   }
}
```

Класс ArduinoHardware.cs

```
using System;
using System.Threading;
using System.IO.Ports;
```

```
using System.Configuration;
using System.Linq;
using skud.Properties;

namespace skud.Domain
{
    /// <summary>
    /// Общается с Arduino, считывает карты и управляет турникетом
    /// </summary>
    public sealed class ArduinoGateway : IDisposable
    {
        public delegate void AccessRequestHandler(ulong uid, Direction direction);
        public delegate void CardReadEventHandler(ulong uid);

        /// <summary>
        /// Событие запроса доступа
        /// </summary>
        public event AccessRequestHandler AccessRequested;

        /// <summary>
        /// Событие чтения карты (без запроса)
        /// </summary>
        public event CardReadEventHandler CardRead;

        private static readonly Object s_lock = new Object();
        private static ArduinoGateway instance = null;

        private SerialPort _port;
        private PackageBuilder _builder;
        private byte[] _sendPacket = new byte[1];



        public static void Init(string port)
        {
            if (instance != null)
                throw new Exception("Already initialized");

            Monitor.Enter(s_lock);
            var temp = new ArduinoGateway(port);
            Interlocked.Exchange(ref instance, temp);
            Monitor.Exit(s_lock);
        }

        private ArduinoGateway(string port)
        {
            _builder = new PackageBuilder(new byte[] { 0x37, 0x83 }, 5);
            _builder.PackageReceived += _builder_PackageReceived;

            _port = new SerialPort(port, 9600);
```

```
_port.Open();
        _port.DataReceived += _port_DataReceived;
    }

    // Получены данные по COM-порту
    private void _port_DataReceived(object sender, SerialDataReceivedEventArgs e)
    {
        int size = _port.BytesToRead;
        byte[] buffer = new byte [size];
        _port.Read(buffer, 0, size);

        _builder.ProcessPart(buffer);
    }

    // Пакет полностью считан
    private void _builder_PackageReceived(object sender, byte[] package)
    {
        // Получаем UID карты
        ulong uid = BitConverter.ToUInt32(package, 0);

        // Получаем направление движения
        Direction dir;
        byte dirByte = package[sizeof(int)];
        if (dirByte == 0)
            dir = Direction.NONE;
        else if (dirByte == 1)
            dir = Direction.IN;
        else if (dirByte == 2)
            dir = Direction.OUT;
        else
            throw new Exception("Invalid direction value");


        if (CardRead != null)
            CardRead(uid);
        else if (AccessRequested != null && dir != Direction.NONE)
            AccessRequested(uid, dir);
    }



    // Синглтон

    public static ArduinoGateway Instance
    {
        get
        {
            if (instance != null) return instance;
            throw new Exception("Not initialized");
        }
```

```
 }

      public void SendResponse(AccessStatus status)
      {
         if (status == AccessStatus.GRANTED)
            _sendPacket[0] = 1;
         else if (status == AccessStatus.DENIED)
            _sendPacket[0] = 2;
         else
            _sendPacket[0] = 0;

         _port.Write(_sendPacket, 0, _sendPacket.Length);
      }

      public void Dispose()
      {
         _port?.Dispose();
      }

      public static string[] GetPorts()
      {
         return SerialPort.GetPortNames();
      }
   }
}
```

Класс AccessStatus.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace skud.Domain
{
   public enum AccessStatus
   {
      NONE,
      GRANTED,
      DENIED
   }
}
```

Класс AccessController.cs

```
using skud.Data;
using skud.Domain.Models;
using skud.ViewModels;
using System;
using System.Collections.Generic;
```

```
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Data.Entity;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace skud.Domain
{
    /// <summary>
    /// Основная логика - проверка пропусков, определение разрешения или
    /// запрета прохода, сохранение смен
    /// </summary>
    public class AccessController : INotifyPropertyChanged
    {
        public ObservableCollection<LogItemViewModel> EventLog { get; set; }
        public User User { get; set; }
        public Direction? Direction { get; set; }
        public AccessStatus? Access { get; set; }

        private SkudContext _ctx;

        public AccessController(SkudContext ctx)
        {
            _ctx = ctx;
            EventLog = new ObservableCollection<LogItemViewModel>();

            ArduinoGateway.Instance.AccessRequested += Instance_AccessRequested;
        }

        private void Instance_AccessRequested(ulong uid, Direction direction)
        {

Application.Current.Dispatcher.BeginInvoke(System.Windows.Threading.DispatcherPriority.Bac
kground, new Action(() => AccessRequested(uid, direction)));
        }

        private void AccessRequested(ulong uid, Direction direction)
        {
            User = GetUser(uid);

            if (User != null)
            {
                if (direction == Domain.Direction.IN)
                {
                    var ws = new WorkShift()
                    {
                        ArrivalTime = DateTime.Now,
                        CardUid = (long)uid
```

```
};
                _ctx.WorkShifts.Add(ws);
                _ctx.SaveChanges();
            }
            else
            {
                long id = (long)uid;
                var last = _ctx.WorkShifts.Where(x => x.CardUid == id).OrderByDescending(x =>
x.Id)
                    .FirstOrDefault();
                if (last != null)
                {
                    last.LeavingTime = DateTime.Now;
                    _ctx.SaveChanges();
                }
            }

            Access = AccessStatus.GRANTED;
            Direction = direction;
        }
        else
        {
            Access = AccessStatus.DENIED;
            Direction = null;
        }

        EventLog.Add(new LogItemViewModel()
        {
            Date = DateTime.Now,
            CardId = uid,
            Fio = User?.FIO,
            Direction = direction,
            Status = Access.Value
        });

        ArduinoGateway.Instance.SendResponse(Access.Value);
    }

    private User GetUser(ulong uid)
    {
        long id = (long) uid; //EF не поддерживает unsigned
        return _ctx.Cards
            .Where(x => x.Uid == id && x.IssueDate <= DateTime.Now && x.ExpirationDate >=
DateTime.Now)
            .Select(x => x.User)
            .Include(x=>x.Department)
            .Include(x => x.Position)
            .Include(x => x.Rank)
            .FirstOrDefault();
    }
```

```
  public event PropertyChangedEventHandler PropertyChanged;
   }
}
```

Класс Configuration.cs

```
namespace skud.Migrations
{
   using System;
   using System.Data.Entity;
   using System.Data.Entity.Migrations;
   using System.Linq;

   internal sealed class Configuration : DbMigrationsConfiguration<skud.Data.SkudContext>
   {
      public Configuration()
      {
         AutomaticMigrationsEnabled = false;
      }

      protected override void Seed(skud.Data.SkudContext context)
      {

      }
   }
}
```

Класс init.cs

```
namespace skud.Migrations
{
   using System;
   using System.Data.Entity.Migrations;

   public partial class Init : DbMigration
   {
      public override void Up()
      {
         CreateTable(
            "dbo.Cards",
            c => new
               {
                  Uid = c.Long(nullable: false),
                  IssueDate = c.DateTime(nullable: false),
                  ExpirationDate = c.DateTime(nullable: false),
                  UserId = c.Int(nullable: false),
               })
            .PrimaryKey(t => t.Uid)
            .ForeignKey("dbo.Users", t => t.UserId, cascadeDelete: true)
            .Index(t => t.UserId);

         CreateTable(
```

```
"dbo.Users",
            c => new
                {
                    Id = c.Int(nullable: false, identity: true),
                    Surname = c.String(maxLength: 4000),
                    Name = c.String(maxLength: 4000),
                    Middlename = c.String(maxLength: 4000),
                    Photo = c.String(maxLength: 4000),
                    PositionId = c.Int(nullable: false),
                    RankId = c.Int(nullable: false),
                    DepartmentId = c.Int(nullable: false),
                })
            .PrimaryKey(t => t.Id)
            .ForeignKey("dbo.Departments", t => t.DepartmentId, cascadeDelete: true)
            .ForeignKey("dbo.Positions", t => t.PositionId, cascadeDelete: true)
            .ForeignKey("dbo.Ranks", t => t.RankId, cascadeDelete: true)
            .Index(t => t.PositionId)
            .Index(t => t.RankId)
            .Index(t => t.DepartmentId);

        CreateTable(
            "dbo.Departments",
            c => new
                {
                    Id = c.Int(nullable: false, identity: true),
                    Name = c.String(maxLength: 4000),
                })
            .PrimaryKey(t => t.Id);

        CreateTable(
            "dbo.Positions",
            c => new
                {
                    Id = c.Int(nullable: false, identity: true),
                    Name = c.String(maxLength: 4000),
                })
            .PrimaryKey(t => t.Id);

        CreateTable(
            "dbo.Ranks",
            c => new
                {
                    Id = c.Int(nullable: false, identity: true),
                    Name = c.String(maxLength: 4000),
                })
            .PrimaryKey(t => t.Id);

        CreateTable(
            "dbo.WorkShifts",
            c => new
```

```
  {
                Id = c.Int(nullable: false, identity: true),
                CardUid = c.Long(nullable: false),
                ArrivalTime = c.DateTime(),
                LeavingTime = c.DateTime(),
            })
        .PrimaryKey(t => t.Id)
        .ForeignKey("dbo.Cards", t => t.CardUid, cascadeDelete: true)
        .Index(t => t.CardUid);
    }

    public override void Down()
    {
        DropForeignKey("dbo.WorkShifts", "CardUid", "dbo.Cards");
        DropForeignKey("dbo.Users", "RankId", "dbo.Ranks");
        DropForeignKey("dbo.Users", "PositionId", "dbo.Positions");
        DropForeignKey("dbo.Users", "DepartmentId", "dbo.Departments");
        DropForeignKey("dbo.Cards", "UserId", "dbo.Users");
        DropIndex("dbo.WorkShifts", new[] { "CardUid" });
        DropIndex("dbo.Users", new[] { "DepartmentId" });
        DropIndex("dbo.Users", new[] { "RankId" });
        DropIndex("dbo.Users", new[] { "PositionId" });
        DropIndex("dbo.Cards", new[] { "UserId" });
        DropTable("dbo.WorkShifts");
        DropTable("dbo.Ranks");
        DropTable("dbo.Positions");
        DropTable("dbo.Departments");
        DropTable("dbo.Users");
        DropTable("dbo.Cards");
    }
  }
}
```

Класс EditHelpers.cs

```
using System.Data.Entity;
using System.Linq;
using skud.Data;

namespace skud.Helpers
{
  public static class EditHelpers
  {
    public static void DetachAllEntities<T>(SkudContext ctx)
    {
      var changedEntriesCopy = ctx.ChangeTracker.Entries()
        .Where(e => e.State == EntityState.Added ||
                    e.State == EntityState.Modified ||
                    e.State == EntityState.Deleted)
        .ToList();
```

```
            foreach (var entity in changedEntriesCopy)
                ctx.Entry(entity.Entity).State = EntityState.Detached;
        }
    }
}
```

КлассSkudContext.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity;
using skud.Domain.Models;

namespace skud.Data
{
    public class SkudContext : DbContext
    {
        public SkudContext():base("DbConnection")
        {
        }

        public DbSet<User> Users { get; set; }
        public DbSet<Card> Cards { get; set; }
        public DbSet<Department> Departments { get; set; }
        public DbSet<Position> Positions { get; set; }
        public DbSet<Rank> Ranks { get; set; }
        public DbSet<WorkShift> WorkShifts { get; set; }
    }
}
```

Класс IsNullConverter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Data;

namespace skud.Converters
{
    public class IsNullConverter : IValueConverter
    {

        public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            return value == null;
```

```
}

        public object ConvertBack(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

Приложение В
Акт внедрения

«Утверждаю»

Проректор по учебной работе
Утепкалиева К.М.
«2» июня 2020 г

АКТ

о внедрении информационной системы пропускного режима образовательного учреждения

Жумартова Нурсултана

Комиссия в составе:

Председатель:Габбасова Ж.Д. – к.т.н., ассоц.профессор, заведующая кафедрой программной инженерии АтГУ им.Х.Досмухамедова
Члены комиссии: Рашбаев Ж.М.–к.ф.м.н., профессор, Махатова В.Е.– к.т.н., ассоц. профессор, Турмуханова Г.Б. – магистр, ст.преподаватель.

Составили настоящий акт о том, что научные разработки и результаты исследований, представленные в дипломном проекте студента-выпускника Алматинского университета энергетики и связи им. Гумарбека Даукеева, Жумартова Н.Н. на тему «Разработка информационной системы пропускного режима образовательного учреждения».

Использование системы пропускного режима позволит:
- вести учёт рабочего времени;
- ведение базы персонала / посетителей;
- получать световое и (или) звуковое оповещение о попытках НСД;
- вести учет индивидуального отработанного времени;
- улучшить работу охранных и учетных отделов;
- увеличить безопасности учреждения.

Рассмотренный проект неминуемо приведёт к модернизации образовательного процесса и работысотрудников. Внедрение новых информационных технологий в учебный процесс способствует более тщательному мониторингу посещаемости, реализовать идеи развития безопасности образовательных учреждений, повысить продуктивность учащихся.

Председатель комиссии Габбасова Ж.Д.