МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

ДОПУСТИТЬ К ЗАЩИТЕ

|  |  |  |
| --- | --- | --- |
| Зам. декана по УМР ФСПО |  |  |
| должность, уч. степень, звание |  | подпись, дата |  |

ДИПЛОМНЫЙ ПРОЕКТ

|  |  |
| --- | --- |
| на тему | Разработка программы работы с бесконтактными картами |

|  |  |
| --- | --- |
| выполнен |  |
| фамилия, имя, отчество студента в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по специальности | 09.02.03 |  | Программирование в компьютерных системах |
|  | код |  | Наименование специальности |
|  | | | |

Санкт-Петербург2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

УТВЕРЖДАЮ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Зам. декана по УМР ФСПО | |  |  | | |  | |  | |
| должность, уч. степень, звание |  | | | подпись, дата |  | | инициалы, фамилия | |

ЗАДАНИЕ НА ДИПЛОМНЫЙ ПРОЕКТ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| по специальности | | | 09.02.03 Программирование в компьютерных системах | |
|  | | | Код, наименование специальности | |
|  | | |  | |
|  | | |  | |
| студенту группы № |  |  | |  | |
|  |  |  | | (фамилия, имя, отчество) | |

|  |  |
| --- | --- |
| на тему | Разработка программы работы с бесконтактными картами |

|  |
| --- |
| утвержденную приказом ГУАП от |

1 Основные исходные данные на разработку:

|  |
| --- |
| - Двухчастотный модуль считывателя WG26 RFID Reader Wireless Module 13.56MHz 125KHz for IC/ID Card  - Информация поступающая с модуля |
| Функции приложения: |
| - Считывание данных с радиометок на частоте 13.56 МГц и 125 КГц |
| - Редактирование или перезапись данных радиометок |
| - Генерация ключей записываемых на радиометки для авторизации новых пользователей |
| - Автоматическое определение типа радиометки |
| - Добавление сгенерированных ключей в базу данных |
| Выходные данные: |
| - Отображение информации с радиометок |

2 Перечень и примерное содержание обязательных разделов дипломного проекта

|  |
| --- |
| Введение |
| 1 Теоретический раздел |
| 1.1 Постановка задачи |
| 1.2 Сравнение с существующими аналогами |
| 1.3 Обоснование выбора модуля считывателя |
| 1.4 Описание схемы работы модуля считывателя |
| 1.5 Описание технологии обмена данными |
| 1.6 Обоснование выбора сред программирования |
| 1.7 Структура программы |
| 1.8 Описание результатов отладки и сценариев тестирования |
| 1.9 Инструкция пользователя |
| 2 Обеспечение безопасности эксплуатации программного продукта  2.1 Общие правила безопасности при использовании компьютерной техники |
| 2.2 Общие принципы управления информационной безопасностью |

3 Задание на научно–библиографический поиск

1. Зыков, С. В. Программирование. Объектно-ориентированный подход: учебник и практикум для академического бакалавриата / С. В. Зыков. — М. : Издательство Юрайт, 2019. — 155 с.
2. Основные сведения о системах СКУД https://www.aktivsb.ru/kontrol\_dostupa/
3. Кудрявцев, К. Я. Методы оптимизации: учеб. пособие для вузов / К. Я. Кудрявцев, А. М. Прудников. — 2-е изд. — М. : Издательство Юрайт, 2019. — 140 с.

Срок сдачи дипломного проекта:\_\_\_\_\_01 июня\_\_\_\_\_\_\_2021\_г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc73519871)

[1 Теоретический раздел 7](#_Toc73519872)

[1.1 Постановка задачи 7](#_Toc73519873)

[1.2 Сравнение с существующими аналогами 8](#_Toc73519874)

[1.4 Описание схемы работы считывателя 12](#_Toc73519875)

[1.5 Описание технологии обмена данными 18](#_Toc73519876)

[1.6 Обоснование выбора сред программирования 19](#_Toc73519877)

[1.7 Структура программы 21](#_Toc73519878)

[1.8 Описание результатов отладки и сценариев тестирования 25](#_Toc73519879)

[1.9 Инструкция пользователя 30](#_Toc73519880)

[2 Обеспечение безопасности эксплуатации программного продукта 34](#_Toc73519881)

[2.1 Общие правила безопасности при использовании компьютерной техники 34](#_Toc73519882)

[2.2 Методы и механизмы безопасности ИТ 37](#_Toc73519883)

[ЗАКЛЮЧЕНИЕ 42](#_Toc73519884)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc73519885)

[ПРИЛОЖЕНИЕ А](#_Toc73519886)(обязательное). [Листинг программы 45](#_Toc73519887)

# ВВЕДЕНИЕ

На данный момент ни одно современное предприятие не обходится без системы контроля доступа, например: завод, торговый центр или офис.

Система контроля и управления доступом (СКУД) – это совокупность технических средств, направленных на контроль входа и выхода в помещение с целью обеспечения безопасности и регулирования посещения определённого объекта. Данная система позволяет объединить информационные технологии с системой безопасности здания, что даёт значительные преимущества перед аналоговым вариантом. Одной из сильных сторон системы являются электронные карточки, которые заменяют обычные ключи. В отличие от металлических собратьев, карточки выполнены из пластика, и весят крайне мало. Еще одним преимуществом электронных карточек является их возможность перезаписи и модификации уникального ключа – идентификатора, который привязан к носителю. Благодаря этому, можно, например, составить статистику посещения условного здания пользователями, или же, моментально заблокировать доступ, удалив ключ нежелательного сотрудника из системы СКУД всего парой команд с компьютера.[2]

Ключ-карта представляет собой маленький электронный чип, являющийся “мозгом карточки”. Он обеспечивает обмен данными между считывателем, одновременно храня уникальный идентификатор в своём блоке памяти, и небольшую катушку индуктивности, которая обеспечивает питание чипа. Ключ-карта может быть выполнена: в виде обычной пластиковой карточки, не отличающейся габаритами от банковской, а также в виде маленького брелка или даже имплантата, который можно вживить живому существу под кожу.

Самый распространенный стандарт карт это: EM4100 (EM4102, EM-Marin) — формат бесконтактных радиочастотных идентификационных карт компании EM Microelectronic-Marin (англ.), одни из самых распространённых в России. Относятся к классу пассивных RFID карт, поскольку не имеют встроенного источника питания. Работают в частотном диапазоне 125 кГц и хранят уникальный номер длиной 40 бит.[6]

Выпускаются в разнообразных конструктивах (наиболее распространены Clamshell-карты, ISO 7810-карты, брелоки). ISO-карты могут выпускаться дополнительно с магнитной полосой, идентификационным номером, изготовленным методом тиснения, полем для подписи держателя карты. Персонализация ISO-карт происходит с помощью термопечати, шелкографии, офсетной печати.

К картам частотой в 13.56 МГц относятся карты типа MIFARE Classic, MIFARE Classic EV1, MIFARE ID, MIFARE Mini, MIFARE Plus

Семейство MIFARE Classic состоит из карт MIFARE Classic 1K, MIFARE Classic 4K, MIFARE Classic EV1 1K, MIFARE Classic EV1 4K, MIFARE ID и MIFARE Mini. MIFARE Classic 1K была исторически первой.

Карты MIFARE Classic предлагают надстройку над ISO 14443A-3 с криптографической защитой данных. Содержат 4-байтный или 7-байтный неизменяемый уникальный код карты и 1 или 4 КБ пользовательских и конфигурационных данных карты. MIFARE ID и MIFARE Mini представляют собой MIFARE Classic, обрезанные до одного и пяти секторов соответственно (ёмкость 64 и 320 байт, включая технологические блоки).[7]

Основной операционной системой, под которую разрабатывалось данное программное обеспечение (ПО) - Windows 10, компании Microsoft. Главным преимуществом данной операционной системы (ОС) является её распространенность, однако, в дальнейшем ожидается адаптация ПО и под Linux.

# Теоретический раздел

# Постановка задачи

Целью данного дипломного проекта является разработка программного обеспечения для работы со специализированным RFID модулем (Radio Frequency Identification - радиочастотная идентификация), который соединяется с персональным компьютером по USB интерфейсу.

Пользователями данного программного обеспечения могут быть специалисты службы охраны, администраторы системы контроля доступа и другие.

* + 1. Требования к функционированию.

Данный программный продукт должен обеспечивать следующие

функции:

* + считывание данных с радиометок на частоте 13.56 МГц и 125 КГц;
  + редактирование или перезапись данных радиометок;
  + генерация ключей записываемых на радиометки для авторизации новых пользователей;
  + автоматическое определение типа радиометки;
* добавление сгенерированных ключей в базу данных.
  + 1. Входные данные.
* двухчастотный модуль считывателя WG26 RFID Reader Wireless; Module 13.56MHz 125KHz for IC/ID Card;
* информация, поступающая с модуля.
  + 1. Выходные данные.

Выходными данными является отображение информации с радиометок.

* + 1. Требования к пользовательскому интерфейсу.

Основные требования к пользовательскому интерфейсу:

* функциональность (соответствие задачам пользователя);
* соответствие технологии;
* понятность и логичность;
* обеспечение высокой скорости работы пользователя;
* обеспечение защиты от человеческих ошибок;
* быстрое обучение пользователя;
* субъективное удовлетворение пользователя.
  + 1. Минимальные требования к аппаратному и программному обеспечению.

Минимальные требования при установке системы

7941WCommunicator на компьютер с ОС Windows x64:

* операционная система – Windows 10/7/XP;
* файловая система – NTFS;
* оперативная память – 2Гб.;
* Microsoft .NET Framework 4.7 под текущую версию ОС Windows;
* пакет драйверов для USB преобразователя.

# Сравнение с существующими аналогами

Среди программ, позволяющих напрямую взаимодействовать с похожими модулями на “низком” уровне, существует огромное разнообразие.

Основная масса аналогов выполняет обобщённые функции обмена данными между ПК и модулями. Однако среди программ, позволяющих напрямую работать с модулем 7941W, аналогов не нашлось.

После исследования области был выделен следующий аналог: - Terminal v1.9b by Br@y++.

Данная программа позволяет составлять команды и напрямую отправлять их на модуль, и использовалась мной для отладки дипломного проекта.

Для идентификации проектируемого приложения было дано название - 7941WCommunicator.

В сравнении не затрагиваются малозначимые функции, которые не были предусмотрены во время проектирования и разработки ПО.

В таблице 1 представлено сравнение функционала разрабатываемого приложения с его аналогами

Таблица 1 – Сравнение 7941WCommunicator с аналогами

|  |  |  |
| --- | --- | --- |
| Характеристики | Приложение | |
| 7941WCommunicator | «Terminal v1.9b by Br@y++» |
| Обмен данными с модулем по SerialCOM | + | + |
| Возможность выбора скорости обмена данными | + | + |
| Наличие готовых, специализированных команд, предназначенных только для 7941W | + | - |
| Работа с БД | + | - |
| Генерация уникальных ключей | + | - |
| Необходимость наличия специализированных знаний | - | + |
| Упрощенный интерфейс | + | -(пользователь вынужден вручную вводить комманды) |
| Наличие команд на запись/перезапись уникальных ключей | + | - |

Изучив данные, можно выделить следующие преимущества 7941WCommunicator:

* программа специализирована на работу с определенным модулем;
* наличие уникальных функций;
* низкий порог вхождения;
* простота в использовании.

**1.3 Обоснование выбора модуля считывателя**

Во время разработки дипломного проекта было рассмотрено несколько модулей, позволяющих работать с радиометками: RDM6300, RC522 и 7941W.

Модуль RDM6300 позволяет работать с метками только с частотой 125 кГц, к тому же, единственная операция, доступная этому модулю – считывание уникального кода ключа, без возможности как-либо перезаписывать карту. Однако он очень прост в использовании, так как не требует каких-либо команд.

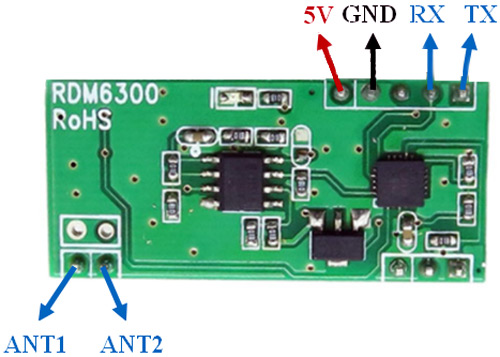


Рисунок 1 -Модуль считывателя RDM6300

Модуль RC522 позволяет работать с картами частотой 13.56 МГц, в отличие от первого. Данный модуль позволяет редактировать данные, записанные на карту, однако для его работы требуется подключение микроконтроллера, что увеличивает сложность проекта на несколько порядков.

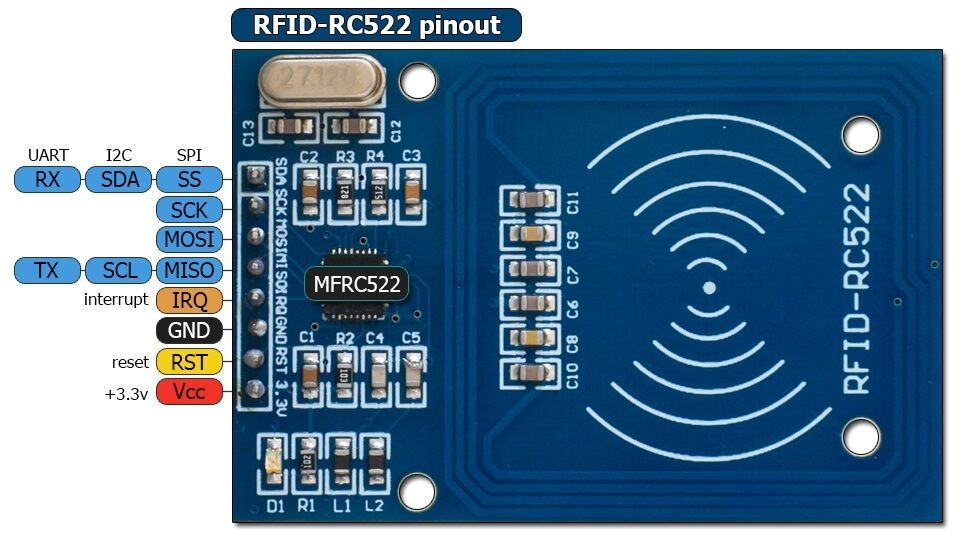


Рисунок 2 -Модуль считывателя RC522

Модуль 7941W – работает с двумя частотами одновременно, даёт возможность редактировать данные на любом типе карт, не требует внешнего микроконтроллера, так как уже имеет свой на плате. Интерфейс для связи с персональным компьютером (ПК) достаточно прост. Единственный минус данного модуля – для работы он требует команды с ПК.

Технические характеристики модуля 7941W:

* напряжение: DC 5В ;
* ток потребления: 50 мA;
* дистанция: Mifare>3cм; EM>5cм;
* размер: 47мм \*26мм\*5мм;
* поддерживаемые интерфейсы: UART, Wiegand.
* поддерживаемые кристаллы: ISO/IEC 14443 A/MIFARE, NTAG, MF1xxS20, MF1xxS70, MF1xxS50;
* поддержка записи перезаписи карт:EM4100, T5577;
* рабочая температура: -25~85 по Цельсию.

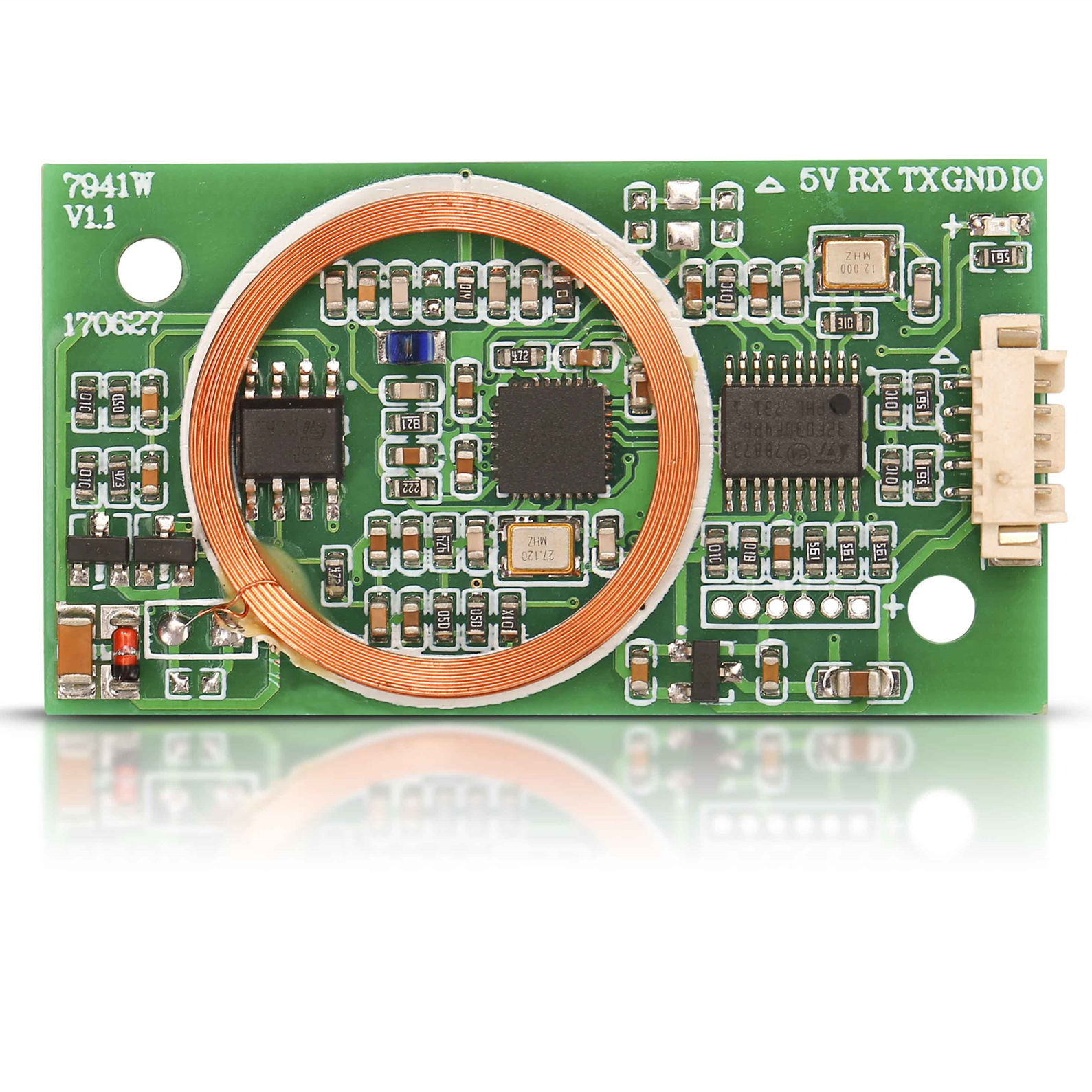


Рисунок 3 -Модуль считывателя 7941W

# 1.4 Описание схемы работы считывателя

Под схемой подразумевается взаимодействие разных элементов схемы для обмена данными. Сборочный чертеж модуля показан на рисунке 4.

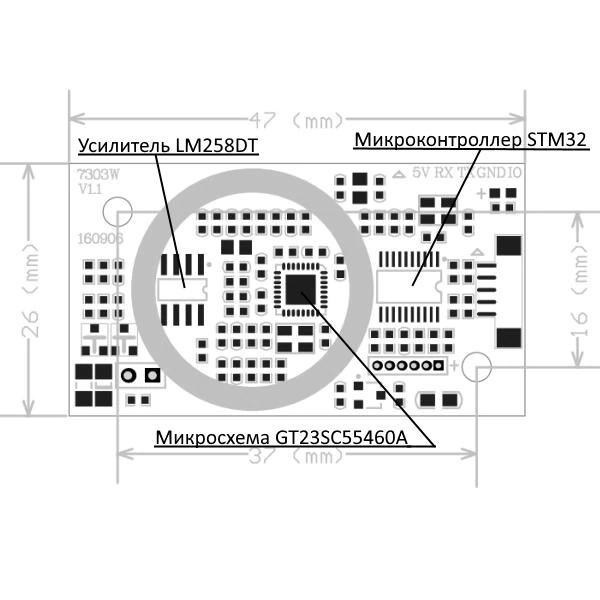


Рисунок 4 – Сборочный чертеж модуля

Основными компонентами модуля являются 3 микросхемы, которые обеспечивают его полную функциональность:

1. микроконтроллер STM32F030F4P6 - является мозгом считывателя. Он обеспечивает беспрерывную обработку данных, приходящих с микросхемы считывателя, одновременно поддерживая связь с персональны м компьютером.

Изображение выглядит как текст, электроника, цепь

Автоматически созданное описание

Рисунок 5 – Микроконтроллер STM32

1. Микросхема GT23SC55460A - отвечает за считывание и запись данных с радиометок.

Она содержит в себе:

* 1. блок интерфейсов для связи с микроконтроллером;
  2. аналоговый интерфейс для непосредственной связи с радиометкой;
  3. блоки буферной памяти для хранения “сырых” данных.

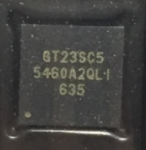


Рисунок 6 - Микросхема GT23SC55460A

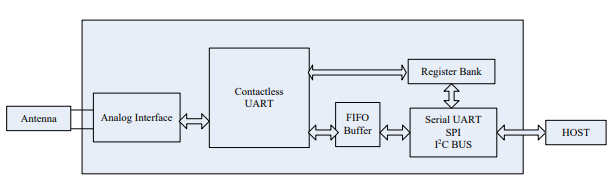


Рисунок 7 – Блоковая диаграмма микросхемы

1. LM258DT — это двухканальный операционный усилитель, он предназначен для организации управления цепью питания катушек интерфейсов.

Изображение выглядит как электроника, цепь

Автоматически созданное описание

Рисунок 8 – Микросхема LM258DT

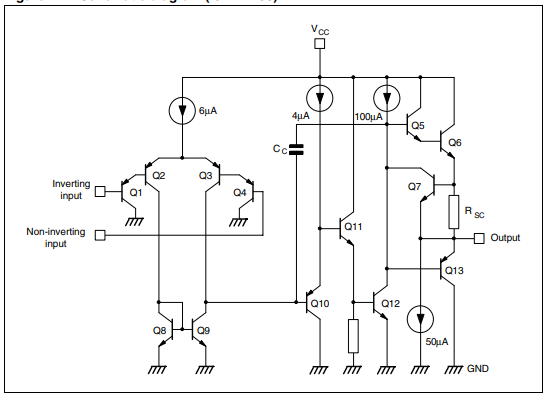


Рисунок 9 – Схематическая диаграмма микросхемы

На рисунке 3 приведена схема взаимосвязи блоков на считывателе.



Рисунок 10 – Схема работы считывателя

Благодаря обширным возможностям микроконтроллера, считывание данных с низкочастотных меток происходит при помощи цифровых выходов Dout1 и Dout2 (на рисунке отмечены фиолетовой и жёлтой линиями). Связь микроконтроллера с микросхемой GT23 происходит по интерфейсу I2C. Антенна на 13.56 МГц подключена напрямую к микросхеме GT23SC55460A.

Антенна на 125 кГц подключена к цифровым выводам микроконтроллера через два транзисторных ключа. Подробная схема цепи модуля представлена на рисунке 11.



Рисунок 11 –Схема цепи модуля

Модуль постоянно находится в режиме ожидания, и без получения команд с ПК ничего не делает. При подключении модуля посредством USB-TTL адаптера, система инициализирует преобразователь, как виртуальный COM-порт, и создает программную связь между устройствами.

При поступлении данных с ПК пользователя, на котором запущена программа, данные через USB преобразователь передаются напрямую в микроконтроллер. Далее микроконтроллер анализирует сообщение. Если в сообщение указано, что нужно прочитать данные с карточки на 13.56 МГц, то микроконтроллер формирует сообщение для микросхемы, она, в свою очередь активирует питание катушки и пытается считать нужные данные с карточки. Процесс считывания по частоте 125 кГц выполнен немного по-другому, а именно: микроконтроллер, посредством операционного усилителя, сам генерирует нужные для протокола импульсы и обеспечивает как считывание, так и запись на карточки. Затем, если не удалось прочитать данные, модуль возвращает сообщение об ошибке. Состав сообщения указан в таблице 2.

Таблица 2 - Состав сообщения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Заголовок | Адрес | Команда | Длина данных | Данные | Контрольная сумма XOR |
| Сообщение для отправки на модуль | | | | | |
| AB BA | 1 Байт | 1 Байт | 1 Байт | 1-255 Байт | 1 Байт |
| Сообщение, приходящее с модуля | | | | | |
| CD DC | 1 Байт | 1 Байт | 1 Байт | 1-255 Байт | 1 Байт |

Заголовок сообщения занимает два байта, они определяют тип сообщения: отправка на модуль или ответ с модуля. Байт адреса означает адрес считывателя, и он нужен для определения считывателей при наличии нескольких в системе СКУД, по умолчанию адрес равен нулю.

Модуль воспринимает большое количество команд:

1. 0x10 - считать UID номер карты на 13.56 МГц;
2. 0x11 - записать UID карты (4 байта), используя стандартный пароль ffffffffff;
3. 0x12 - считать определенную область данных;
4. 0x13 - записать определенную область данных;
5. 0x14 - модифицировать пароль группы A или группы B;
6. 0x15 - считать ID номер карточки на 125 КГц;
7. 0x16 - записать T5577 номер 125 КГц ;
8. 0x17 - считать все данные (M1-1K карта).

# 1.5 Описание технологии обмена данными

Для обмена данными между модулем и ПК используется мост в виде микросхемы USB-TTL. Данное устройство эмулирует COM-порт, и в системе распознается как виртуальный COM-порт, позволяя программе общаться с модулем посредством последовательного порта.

Последовательный порт или COM-порт - двунаправленный последовательный интерфейс, предназначенный для обмена байтовой информацией.

Схема обмена данными между модулем и компьютером представлена на рисунке 12.

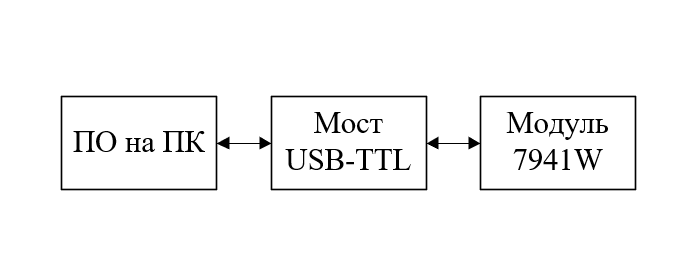


Рисунок 12 – Схема связи модуля с ПК.

Последовательный он, потому что информация через него передаётся по одному биту, бит за битом. Наиболее часто для последовательного порта персональных компьютеров используется стандарт RS-232C.

Ранее последовательный порт использовался для подключения терминала, позже для модема или мыши. Сейчас он используется для соединения с источниками бесперебойного питания, для связи с аппаратными средствами разработки встраиваемых вычислительных систем и прочей микроэлектроникой.

Из преимуществ последовательного порта можно отметить простоту работы при разработке программного обеспечения. Visual Studio имеет отдельный пакет классов для данного интерфейса, а именно: System.IO.Ports.

# 1.6 Обоснование выбора сред программирования

1.6.1 Язык программирования C#.

На стадии проектирования приложения выбирались средства создания программного продукта. Для разработки приложения было решено использовать среду разработки Visual Studio. В качестве языка программирования был выбран язык C#, который отлично подходит для поставленной задачи.

Преимущества C#:

* C# – это объектно-ориентированный, простой и в то же время мощный язык программирования, который позволяет разработчикам создавать многофункциональные приложения;
* C# относиться к языкам компилируемого типа, поэтому он обладает всеми преимуществами таких языков;
* C# объединяет лучшие идеи современных языков программирования Java, C++, Visual Basic и т.д.;
* из-за большого разнообразия синтаксических конструкций и возможности работать с платформой .Net, C# позволяет быстрее, чем любой другой язык, разрабатывать программные решения;
* C# отличается надежностью и элегантностью.

В Visual Studio представлены все необходимые для работы с последовательным портом инструменты.

1.6.2 Пространство имен System.IO.Ports.

Содержит классы для управления последовательными портами. Наиболее важный из них, класс [SerialPort](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=dotnet-plat-ext-5.0), предоставляет средства для синхронного и управляемого событиями ввода-вывода, для доступа к состоянию подключения-отключения устройства, а также для доступа к свойствам драйвера последовательного порта. Он может использоваться для упаковки объектов [Stream](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.stream?view=dotnet-plat-ext-5.0), что открывает доступ к последовательному порту из классов, использующих потоки.[12]

Таблица 3 – Свойства класса SerialPort, применяемые в проекте

|  |  |
| --- | --- |
| Название | Назначение |
| BytesToRead | Возвращает число байтов данных, находящихся в буфере приема. |
| ReadByte() | Считывает из входного буфера SerialPort один байт в синхронном режиме. |
| Write(Byte[], Int32, Int32) | Записывает указанное число байтов в последовательный порт, используя данные из буфера. |
| [SerialPort](https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports.serialport?view=dotnet-plat-ext-5.0) | Представляет ресурс последовательного порта. |
| DiscardInBuffer() | Удаляет данные из буфера приема последовательного драйвера. |
| Close() | Закрывает соединение порта, присваивает свойству IsOpen значение false и уничтожает внутренний объект Stream. |
| IsOpen | Возвращает значение, указывающее открытое или закрытое состояние объекта SerialPort. |

Язык программирования C# был выбран из-за наличия удобных инструментов для работы с потоками в реальном времени, что является неоспоримым преимуществом.

1.6.3 СУБД MySQL.

В качестве системы управления базы данных(СУБД) использовалась СУБД MySQL Workbench. MySQL Workbench — это унифицированный визуальный инструмент для архитекторов баз данных и разработчиков БД [8]. MySQL Workbench хорошо сочетается с языком C#, благодаря чему его настройка не занимает много времени.

Помимо универсальности и распространенности СУБД MySQL обладает целым комплексом важных преимуществ перед другими СУБД. В частности следует отметить такие качества как:

* бесплатна в большинстве случаев;
* гибкая система привилегий и паролей;
* простота в использовании. MySQL достаточно легко инсталлируется, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных;
* обширный функционал - система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта;
* безопасность - система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию;
* масштабируемость - являясь весьма универсальной СУБД, MySQL в равной степени легко может быть использована для работы и с малыми, и с большими объемами данных.
* скорость - высокая производительность системы обеспечивается за счет упрощения некоторых используемых в ней стандартов.[13]

# 1.7 Структура программы

1.7.1 Разработка базы данных.

База данных разрабатывалась исключительно для показательной работы системы контроля доступа.

Описание структур таблиц базы данных приведены в таблицах 4 и 5.

Таблица 4 – Описание логической структуры таблицы «administration»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя поля | Признак  ключа | Тип данных | Длина | Точность числа | Ограничения |
|  | ключа | Поля |  |  |  |
| idadministration | Pk | Int | Длинное целое | Авто | >0 |
| LogIn | - | Varchar | 45 | - | >0 |
| Password | - | Varchar | 45 | - | >0 |

Таблица 5 – Описание логической структуры таблицы «residents»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя поля | Признак ключа | Тип данных поля | Длина | Точность | Ограничения |
| idresidents | Pk | int | Длинное целое | Авто | >0 |
| Name | - | Varchar | 255 | - | - |
| Last Name | - | Varchar | 255 | - | - |
| GroupCode | - | int | Длинное целое | - | - |
| SerialNumber | - | Varchar | 70 | - | - |

На рисунке 13 представлена схема базы данных разработанная в MySQL Workbench.

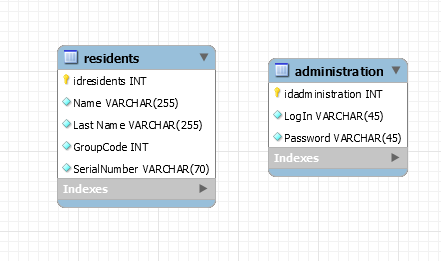


Рисунок 13 – Схема базы данных

1.7.2 Разработка приложения.

Прикладная программа, или приложение, - программа, предназначенная для выполнения определённых задач и рассчитанная на непосредственное взаимодействие с пользователем.

Пользовательский интерфейс (ПИ, графический интерфейс пользователя) – это комплекс средств для взаимодействия пользователя с технической системой , в т. ч. с программным приложением.

В понятие пользовательского интерфейса компьютерной системы входят следующие составляющие:

а) графическая среда – картинка на экране;

б) набор управляющих элементов пользовательского интерфейса и их расположение на экране;

в) технологии взаимодействия пользователя с системой.

Управляющие элементы пользовательского интерфейса – это графические элементы (кнопки, списки, диалоговые окна и т.п.), которые позволяют осуществлять какие-либо действия с компьютерной системой (например, выбирать пункты и свойства объектов).

Схема работы программы — искусственно выделенные программистом взаимодействующие части программы.

Схема работы программы приведена на рисунке 14.



Рисунок 14 – Схема работы программы

Таблица 6 – Функции программы

|  |  |
| --- | --- |
| Название функции | Назначение |
| RDM6300() | Тестовая функция для проверки работы программы с модулем RDM6300 |
| FindPorts() | Функция для сканирования доступных COM портов |
| XORCHeck(byte[] buffer) | Функция для расчёта контрольной суммы |
| GetAKey(byte[] message) | Функция принимает сообщение от модуля, и возвращает только ключ карточки |
| TryToSend(byte[] buffer) | Функция отправляет сообщение на модуль, при удачной отправке, она возвращает значение true |
| SendAndReadMessage(byte[] message) | Функция отправляет сообщение на модуль и возвращает ответ от модуля |
| Output(byte[] buffer) | Функция выводит данные на TextBox Output |

# 1.8 Описание результатов отладки и сценариев тестирования

При разработке приложения достаточно трудоемкими являются этапы отладки и тестирования. Цель тестирования программы заключается в выявлении имеющихся в программе ошибок. Задача отладки программы состоит в устранении ошибок в программе [11].

Нарушение соответствия между детально разработанной записью алгоритма в процессе кодирования программы относится к ошибкам, проходящим вследствие невнимательности программиста.

Учитывая разнообразие источников ошибок, при составлении плана тестирования ошибки разделяют на три типа:

1) синтаксические;

2) семантические;

3) логические.

Синтаксические ошибки – это ошибки в записи конструкций языка программирования (чисел, переменных, методов, выражений, операторов) [3].

Семантические ошибки – нарушение порядка следования параметров функции, некорректное использование переменных (использование индекса, выходящего за пределы массива), ошибки вычисления (незнание приоритетов выполнения операций).

Логические ошибки – изменение логики алгоритма, закономерностей его работы (отсутствие альтернативного пути в структуре выбора, нарушение условий выхода из цикла – зацикливание и т.п.).

Обнаружение синтаксических ошибок автоматизировано в среде разработки Visual Studio, и если в приложении была обнаружена такая ошибка, то оно не будет скомпилировано. Обнаружение логических и семантических - синтаксических, является достаточно сложным процессом. Они не могут быть обнаружены самой средой, поэтому для их обнаружения необходимо проводить тестирование.

На этапе тестирования были обнаружены следующие ошибки, представленные в таблице7.

Таблица 7 – Перечень выявленных ошибок

|  |  |
| --- | --- |
| Ошибка | Решение |
| Нет связи с модулем на скорости 9600 бод | При более подробном прочтении технической документации было установлено, что модуль общается на скорости равной 115200 бод |
| Модуль выдает непонятные цифры при запросе на чтение кода карты | Смена системы счисления выводимого сообщения с 10СС в 16СС |
| При выборе некоторых COMпортов приложение перестает работать | Отлов ошибки, и вывод соответствующего сообщения пользователю |
| Модуль зависает при частых запросах и выдает мусор в ответ | Добавление системных пауз размером в 100 мс. |
| При считывании иногда сообщение приходит неполным | Добавлена функция подсчета и сравнения контрольной суммы. |
| Продолжение таблицы 7 | |
| Ошибка | Решение |
| "Form1.SendAndReadMessage(byte[])": не все пути к коду возвращают значение. | Добавил return null. |
| Требуется catch или finally. | Добавил обработку catch условия |
| "Form1" не содержит определения "groupBox3\_Enter", и не удалось найти доступный метод расширения | Удалил часть ненужного кода из дизайнера формы. |

Тестирование методом «белого ящика» (также: прозрачного, открытого, стеклянного ящика; основанное на коде или структурное тестирование) – метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику. Мы выбираем входные значения, основываясь на знании кода, который будет их обрабатывать. Точно так же мы знаем, каким должен быть результат этой обработки. Знание всех особенностей тестируемой программы и ее реализации – обязательны для этой техники. Тестирование белого ящика – углубление во внутреннее устройство системы, за пределы ее внешних интерфейсов.

Тестирование методом «черного ящика» — это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы. Метод тестирования «черного ящика» – процедура получения и выбора тестовых случаев на основе анализа спецификации (функциональной или нефункциональной), компонентов или системы без ссылки на их внутреннее устройство.

Таблица 8 – Сценарий тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| Вводимые данные | Ожидаемый результат | Фактический результат | Результат тестирования |
| Ввод в поле Input вручную правильного значения ключа  53-0-C2-5D-93 | Отсутствие сообщения об ошибке | Отсутствие сообщения об ошибке | Не успешно |
| Нажатие кнопки Write после ввода правильного значения ключа | Сообщение об успешной записи ключа на карточку | Сообщение об успешной записи ключа на карточку | Не успешно |
| Нажатие кнопки Read для считывания ключа с карты | Отображение в поле Output считанного с карты ключа  Key: 53-0-C2-5D-93 | Отображение в поле Output считанного с карты ключа  Key: 53-0-C2-5D-93 | Не успешно |
| Ввод в поле Input вручную не правильного значения ключа  @-0-C2-5D-9\* | Сообщение об ошибке “Paste error” | Сообщение об ошибке “Paste error” | Не успешно |
| Нажатие кнопки GenKey для автоматической генерации случайного ключа | В поле Input отображается сгенерированный ключ  25-FD-8A-0B-A9 | В поле Input отображается сгенерированный ключ  25-FD-8A-0B-A9 | Не успешно |
| Нажатие кнопки Write для записи ключа на карту после генерации | Запись сгенерированного ключа на карту  25-FD-8A-0B-A9 | Запись сгенерированного ключа на карту  25-FD-8A-0B-A9 | Не успешно |
| Нажатие кнопки Read для считывания ключа с карты | Отображение в поле Output считанного с карты ключа  Key: 25-FD-8A-0B-A9 | Отображение в поле Output считанного с карты ключа  Key: 25-FD-8A-0B-A9 | Не успешно |
| Продолжение таблицы 8 | | | |
| Вводимые данные | Ожидаемый результат | Фактический результат | Результат тестирования |
| Нажатие кнопки  Write при отсутствии карты | В MessageBox вывелось сообщение об ошибке “Operation fail” | В MessageBox вывелось сообщение об ошибке “Operation fail” | Не успешно |
| Нажатие кнопки Scan для определения частоты метки | Вывод частоты считанной метки в LBStatus 13.56MHz | Вывод частоты считанной метки в LBStatus 13.56MHz | Не успешно |
| Попытка подключится к фиктивному COM порту | Вывод в LBStatus сообщения об ошибки подключения к COM порту | Вывод в LBStatus сообщения об ошибки подключения к COM порту | Не успешно |
| Попытка нажать кнопку Reconnect при выбранном фиктивном COM порте | Вывод в LBStatus сообщения об ошибке подключения к COM порту | Вывод в LBStatus сообщения об ошибке подключения к COM порту | Не успешно |
| Нажатие кнопки Clear для очистки поля Output | Поле Output очистилось | Поле Output очистилось | Не успешно |
| Нажатие кнопки Disconnect при выбранном COM порту | Вывод в LBStatus сообщения об отключении порта | Вывод в LBStatus сообщения об отключении порта | Не успешно |

# 1.9 Инструкция пользователя

Для начала работы соедините ваш модуль с USB-UART преобразователем и подсоединитесь к ПК. После установки соответствующих драйверов зайдите в диспетчер устройств.

В диспетчере устройств откройте вкладку COM-порты, как показано на рисунке 15.

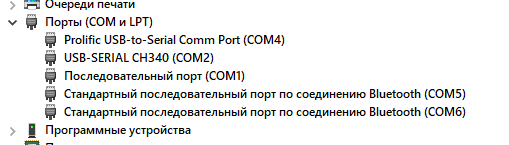


Рисунок 15 – Порты COM и LPT

Запомните номер порта, который появился при подключении модуля.

Запустите программу запустив файл 7941W Communicator.exe

Для подключения к модулю выберете соответствующий COM-порт.

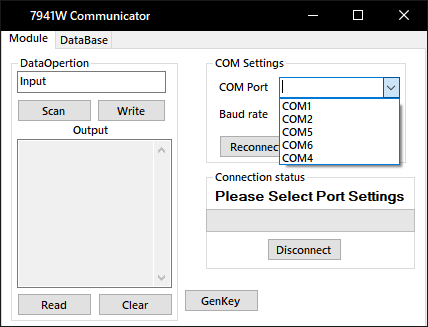


Рисунок 16 – Выбор COM-порта

После выбора COM-порта, программа сообщит об успешном подключении в окне Connectionstatus.

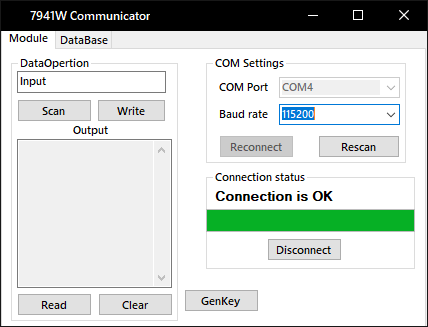


Рисунок 17 – Успешное подключение

Для навигации в программе на рисунке 6 приведено назначение элементов управления.

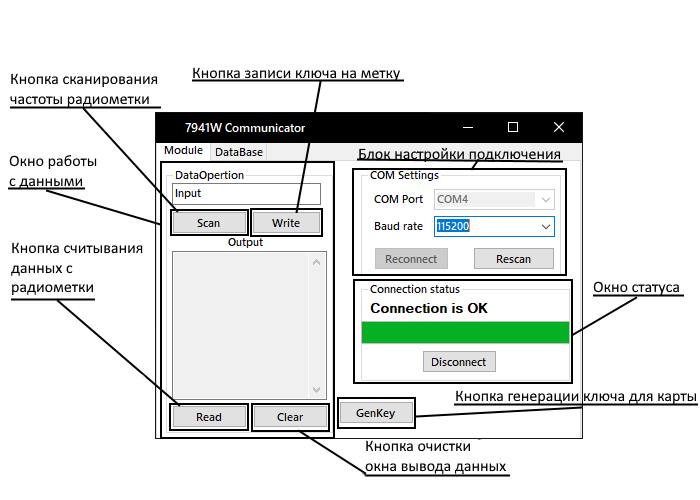


Рисунок 18 – Назначение элементов управления

Если не удалось определить частоту работы радиометки, можно определить её частоту нажатием кнопки Scan. При сканировании радиометки, в окне вывода отображается частота этой метки.

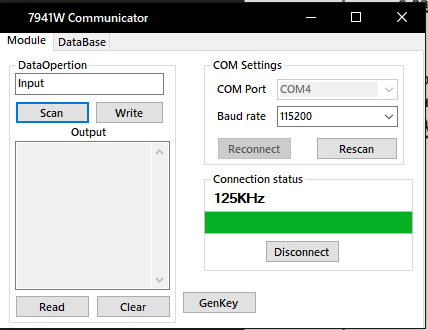


Рисунок 19 – Сканирование радиометки

При нажатии кнопки Read в поле вывода Output отобразится сообщение с модуля: номер карточки, которая была считана.

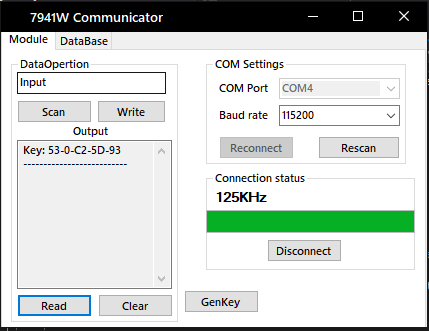


Рисунок 20 – Считывание ключа с карты

Можно сгенерировать ключ для карты, и перезаписать карту новым ключом.

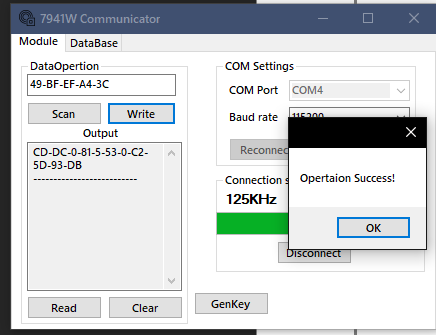


Рисунок 21 – Запись сгенерированного ключа

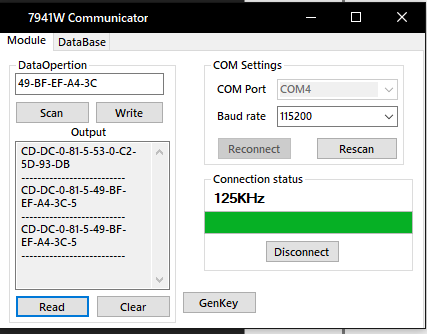


Рисунок 22 – Содержимое перезаписанной метки

# 2 Обеспечение безопасности эксплуатации программного продукта

# 2.1 Общие правила безопасности при использовании компьютерной техники

Техника безопасности при работе с компьютером на предприятии предусматривает наличие общедоступной инструкции, в которой указаны обязательные требования к обустройству рабочего места и процессу использования техники. Эти правила едины для всех организаций, их выполнение контролируется руководящими органами.

Основные правила организации пространства вокруг рабочего места:

* при длительном и интенсивном использовании, на поверхности модулей ПК (системный блок, монитор, мышка и т.д.) возникают небольшие разряды тока. Эти частицы активизируются во время прикосновений к ним и приводят к выходу техники из строя. Нужно регулярно использовать нейтрализаторы, увлажнители воздуха, антистатики;
* вокруг стола не должно быть свисающих проводов, пользователь не должен контактировать с ними;
* важна целостность корпуса розетки и штепсельной вилки;
* отсутствие заземления предэкранного фильтра проверяется с помощью измерительных приборов;
* желательно во время строительных работ в офисе использовать минимальное количество легко воспламеняемых материалов (дерева, пенопласта), а также горючего пластика в изоляции. Рекомендуется отдавать предпочтение кирпичу, стеклу, металлу и т.д.;
* помещение должно хорошо вентилироваться и охлаждаться в жаркую пору года. Важен своевременный отвод избыточного тепла от техники.

Техника безопасности при работе с компьютером

Требования безопасности

Для работников офиса должен быть проведен устный базовый инструктаж, в дальнейшем его печатный текст должен предоставляться для подробного изучения. Организация в обязательном порядке размещает информационный лист на видном месте.

Инструктаж охватывает полный цикл контакта человека с компьютером. Он начинается с установки оборудования сервисной службой и заканчивается утилизацией непригодного устройства.

Перед началом работы

Даже если речь идет о рабочем месте, которое используется каждый день и регулярно проверяется специалистами (как, например, в офисе или учебном заведении), нельзя терять бдительность.

Перед тем, как включить компьютер, необходимо уделить пару минут следующим действиям:

* нужно убедиться в том, что в зоне досягаемости отсутствуют оголенные провода и различные шнуры. Они не только мешают работе, но и несут потенциальную опасность в случае короткого замыкания;
* нельзя начинать работу на технике с видимым повреждением. В случае обнаружения трещины на корпусе или повреждений другого рода, нужно обратиться за помощью в сервисный центр. Это же относится к ПК с неисправным индикатором включения/выключения;
* предметы на столе не должны мешать обзору, пользованию мышкой и клавиатурой. Поверхность экрана должна быть абсолютно чистой;
* на системном блоке не должно находиться никаких предметов, так как в результате вибраций может нарушиться работа устройства. Нужно убедиться в том, что никакие посторонние предметы не мешают работе системе охлаждения.
* недопустимо включать персональный компьютер в удлинители и розетки, в которых отсутствует заземляющая шина;
* запрещается начинать работу в помещениях с повышенной влажностью, а также в случае, если рядом присутствуют открытые источники влажности (лужи, мокрый пол). Включить технику можно лишь после полного высыхания окружающих предметов;
* недопустимо часто включать и выключать компьютер в течение рабочего дня без особой нужды. Система просто не справляется с необходимостью быстро сворачивать все процессы.

При выполнении работы

Поскольку персональный компьютер обладает всеми свойствами электрического прибора, то на него распространяются основные правила безопасности при взаимодействии с проводниками тока:

1. нельзя размещать какие-либо вещи на поводах, а также самостоятельно менять их расположение без особой нужды;
2. рекомендуется избегать расположения жидкостей рядом с модулями компьютера. Поэтому кулер с водой или кофейный автомат необходимо размещать в стороне от рабочих мест в офисе. Пользователи должны осознавать опасность потенциального замыкания в случае пролития воды на клавиатуру или системный блок. Нельзя работать на ПК с мокрыми руками;
3. нельзя очищать поверхность компьютера от загрязнений, когда он находится во включенном состоянии;
4. недопустимо снимать корпус любой из составных частей ПК во время его работы. Кроме того, разбор и ремонт техники имеют совершают только специализированные работники;
5. во время работы на компьютере нельзя одновременно прикасаться к другим металлическим конструкциям, которые стоят на той же поверхности. Это касается отопительных батарей или трубопроводов;
6. в помещении с компьютерами непозволительно курить или употреблять пищу непосредственно на рабочем месте;
7. при ощущении даже незначительного запаха гари, нужно как можно быстрее выключить ПК из сети и обратиться к ответственному за обслуживание компьютерной техники.

# 2.2 Методы и механизмы безопасности информационных технологий

Для предотвращения и ликвидации угроз информационной безопасности используют правовые, программно-технические и организационно-экономические методы.

Правовые методы - предусматривают разработку комплекса нормативно-правовых актов и положений, регламентирующих информационные отношения в обществе, руководящих и нормативно-методических документов по обеспечению информационной безопасности.

Программно-технические методы – это совокупность средств: предотвращение утечки информации, исключение возможности несанкционированного доступа к информации, предотвращение воздействиям, которые приводят к уничтожению, разрушению, искажению информации или сбои или отказов в функционировании средств информатизации, выявление закладных устройств, исключение перехвата информации техническими средствами, использование криптографических средств защиты информации при передаче по каналам связи.

Организационно-экономические методы предполагают формирование и обеспечение функционирования систем защиты секретной и конфиденциальной информации, сертификацию этих систем согласно требованиям информационной безопасности, лицензирования деятельности в сфере информационной безопасности, стандартизации способов и средств защиты информации, контроль за действиями персонала в защищенных информационных системах.

Важное значение для предотвращения информационных угроз имеет мотивация, экономическое стимулирование и психологическая поддержка деятельности персонала, который обеспечивает информационную безопасность.

Методы обеспечения безопасности информации в ИС:

1. препятствие;
2. управление доступом;
3. механизмы шифрования;
4. противодействие атакам вредоносных программ;
5. регламентация;
6. принуждение;
7. побуждение.

Препятствие - метод физического преграждения пути злоумышленнику к защищаемой информации (к аппаратуре, носителям информации и т.д.).

Управление доступом - методы защиты информации регулированием использования всех ресурсов ИС и ИТ. Эти методы должны противостоять всем возможным путям несанкционированного доступа к информации.

Управление доступом включает следующие функции зашиты:

* идентификацию пользователей, персонала и ресурсов системы (присвоение каждому объекту персонального идентификатора);
* опознание (установление подлинности) объекта или субъекта по предъявленному им идентификатору;
* проверку полномочий (проверка соответствия дня недели, времени суток, запрашиваемых ресурсов и процедур установленному регламенту);
* разрешение и создание условий работы в пределах установленного регламента;
* регистрацию (протоколирование) обращений к защищаемым ресурсам;
* реагирование (сигнализация, отключение, задержка работ, отказ в запросе и т.п.) при попытках несанкционированных действий.

Механизмы шифрования - криптографическое закрытие информации. Эти методы защиты все шире применяются как при обработке, так и при хранении информации на магнитных носителях. При передаче информации по каналам связи большой протяженности этот метод является единственно надежным.

Противодействие атакам вредоносных программ предполагает комплекс разнообразных мер организационного характера и использование антивирусных программ. Цели принимаемых мер - это уменьшение вероятности инфицирования АИС, выявление фактов заражения системы; уменьшение последствий информационных инфекций, локализация или уничтожение вирусов; восстановление информации в ИС. Овладение этим комплексом мер и средств требует знакомства со специальной литературой.

Регламентация - создание таких условий автоматизированной обработки, хранения и передачи защищаемой информации, при которых нормы и стандарты по защите выполняются в наибольшей степени.

Принуждение - метод защиты, при котором пользователи и персонал ИС вынуждены соблюдать правила обработки, передачи и использования защищаемой информации под угрозой материальной, административной или уголовной ответственности.

Побуждение - метод защиты, побуждающий пользователей и персонал ИС не нарушать установленные порядки за счет соблюдения сложившихся моральных и этических норм.

Вся совокупность технических средств подразделяется на аппаратные и физические.

Аппаратные средства - устройства, встраиваемые непосредственно в вычислительную технику, или устройства, которые сопрягаются с ней по стандартному интерфейсу.

Физические средства включают различные инженерные устройства и сооружения, препятствующие физическому проникновению злоумышленников на объекты защиты и осуществляющие защиту персонала (личные средства безопасности), материальных средств и финансов, информации от противоправных действий. Примеры физических средств: замки на дверях, решетки на окнах, средства электронной охранной сигнализации и т.п.

Программные средства - это специальные программы и программные комплексы, предназначенные для защиты информации в ИС. Как отмечалось, многие из них слиты с ПО самой ИС.

Из средств ПО системы защиты выделим еще программные средства, реализующие механизмы шифрования (криптографии). Криптография - это наука об обеспечении секретности и/или аутентичности (подлинности) передаваемых сообщений.

Организационные средства осуществляют своим комплексом регламентацию производственной деятельности в ИС и взаимоотношений исполнителей на нормативно-правовой основе таким образом, что разглашение, утечка и несанкционированный доступ к конфиденциальной информации становится невозможным или существенно затрудняется за счет проведения организационных мероприятий. Комплекс этих мер реализуется группой информационной безопасности, но должен находиться под контролем первого руководителя.

Законодательные средства защиты определяются законодательными актами страны, которыми регламентируются правила пользования, обработки и передачи информации ограниченного доступа и устанавливаются меры ответственности за нарушение этих правил.

Морально-этические средства защиты включают всевозможные нормы поведения (которые традиционно сложились ранее), складываются по мере распространения ИС и ИТ в стране и в мире или специально разрабатываются. Морально-этические нормы могут быть неписаные (например, честность) либо оформленные в некий свод (устав) правил или предписаний. Эти нормы, как правило, не являются законодательно утвержденными, но поскольку их несоблюдение приводит к падению престижа организации, они считаются обязательными для исполнения. Характерным примером таких предписаний является Кодекс профессионального поведения членов Ассоциации пользователей ЭВМ США.

В разработанном мной ПО из средств защиты предусмотрен механизм ограничения доступа пользователям с помощью формы доступа к вкладке управления БД. При попытке пользователя переключиться к вкладке Database с целью внесения изменения данных в БД программа запускает форму авторизации, она отображена на рисунке 23. Логин и пароль пользователя, имеющего доступ к базе данных, хранится в таблице Administration.

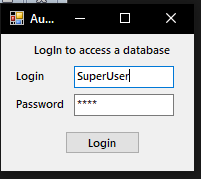


Рисунок 23 – Форма авторизации

# ЗАКЛЮЧЕНИЕ

За период работы над дипломным проектом был создан программный продукт 7941WCommunicator., который может быть использован в таких сферах человеческой деятельности как:

* организация системы контроля доступа;
* обслуживание домофонных систем;
* обслуживание проходок в учебных заведениях;
* настройка RFID и NFC карточек для личных целей.

Работая над темой дипломного проекта я углубил свои знания о способах взаимодействия персонального компьютера с различного рода электронными устройствами, работающими на микроконтроллерах. Так как в учебном плане по выбранной мной специальности 09.02.03 «Программирование в компьютерный системах» это не изучалось, то весь материал по данной тематике осваивал самостоятельно.

Так как в программе были реализованы не все команды микроконтроллера, поддерживаемые моим модулем, в будущем предполагается доработать программу с добавлением функций, реализующих эти команды, а именно:

* возможность считывания и модификации всех секторов памяти карточек типа Mifare classic;
* изменение пароля на защищенных карточках;
* считывание и модификация определенных пользователем секторов данных;
* возможность создания виртуальной визитки пользователя(vCard);

Также предполагается оптимизация кода под сторонние ОС (MacOS, Различные дистрибутивы Linux) для популяризации моего ПО среди специалистов разного толка.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Зыков, С. В. Программирование. Объектно-ориентированный подход: учебник и практикум для академического бакалавриата / С. В. Зыков. — М. : Издательство Юрайт, 2019. — 155 с.

2 Основные сведения о системах СКУД https://www.aktivsb.ru/kontrol\_dostupa/

3 Кудрявцев, К. Я. Методы оптимизации: учеб. пособие для вузов / К. Я. Кудрявцев, А. М. Прудников. — 2-е изд. — М. : Издательство Юрайт, 2019. — 140 с.

4 Гниденко, И. Г. Технология разработки программного обеспечения : учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М. : Издательство Юрайт, 2017. — 235 с.

5 Кубенский, А. А. Функциональное программирование: учебник и практикум для академического бакалавриата / А. А. Кубенский. — М. : Издательство Юрайт, 2019. — 348 с.

6 Основные сведения о карточках 125 КГц - https://ru.wikipedia.org/wiki/EM-4100

7 Сведения о технологии Mifare - https://ru.wikipedia.org/wiki/Mifare

8 Практическое руководство. Шварц Б., Зайцев П., Ткаченко В. "MySQL по максимуму" Питер, 2018 год, 864 с., 3-е изд., O'Reilly, ISBN 978-5-4461-0696-7;

9 Справочник для профессионалов. Албахари, Джозеф, Албахари, Бен. "C# Карманный справочник" Диалектика, 2020 год, 240 стр., ISBN 978-5-907203-14-3;

10 Учебное пособие. П.В. Евдокимов "C# на примерах" Наука и техника (НиТ), 2016 год, 304 с.

11 Практическое руководство. Фленов М. "Библия C#" БХВ-Петербург, 2019 год, 4-е изд., перераб. и доп., 512 с., ISBN 978-5-9775-4041-4;

12 Техническое пособие Microsoft по использованию API IO.Ports - https://docs.microsoft.com/ru-ru/dotnet/api/system.io.ports

13 Справочник по управлению данными MySQL https://depix.ru/articles/sistema\_upravleniya\_bazami\_dannyh\_mysql

14 Ошероув Р. «Искусство автономного тестирования с примерами на C#» ДМК Пресс, 2014 год, 360 с., 2-е изд., пер. с англ. Слинкин А.

15 Настройка ПК для WOL - https://habr.com/ru/post/77191/

16 Создание Telegram-бота - https://aftamat4ik.ru/pishem-bota-telegram-na-c/

17 Руководство для начинающих. Джон Пол Мюллер, Билл Семпф, Чак Сфер "С# для чайников" Диалектика", 2019 год, 608 с., ISBN 978-5-907144-43-9 (рус.);

18 Подробное руководство. Марк Дж. Прайс "C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов" Питер, 2018 год, 640 с., 3-е издание, ISBN: 978-5-4461-0516-8;

19 Основные сведения. Васильев А. Н. "Программирование на C# для начинающих" Эксмо, 2018 год, 592 с., ISBN 978-5-04-092519-3;

20 Подробное руководство. Джон Шарп " Microsoft Visual C#" Питер, 2017 год, 848 с. 8-е изд.

# ПРИЛОЖЕНИЕ А

# Листинг программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO.Ports;

using System.IO;

using MySql.Data.MySqlClient;

using System.Security.Cryptography;

using Microsoft.VisualBasic;

namespace \_79412W13.\_56MHz125KHz

{

public partial class Form1 : Form

{

byte[] bufWriteT5577 = { 0xAB, 0xBA, 0x00, 0x16, 0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };//Write T5577 key

byte[] bufReadT5577 = { 0xAB, 0xBA, 0x00, 0x15, 0x00, 0x15, 0x00 };//Read T5577 data

byte[] bufReadUID = { 0xAB, 0xBA, 0x00, 0x10, 0x00, 0x10, 0x00 };//Read UID

private void FindPorts()

{

object[] ports = SerialPort.GetPortNames();

foreach (string port in ports)

CB\_COMSelect.Items.Add(port);

}

public void RDM6300()

{

char[] buffer = new char[13];

serialPort1.Read(buffer, 0, 13);

for (int i = 1; i != 12; i++)

TB\_Output.Text += (buffer[i]);

TB\_Output.Text += Environment.NewLine;

}

//XOR check: result of other bytes check except protocol header (AB,BA)

public byte XORCHeck(byte[] buffer)

{

int Sum = 0;

byte request;

for (int i = 2; i != buffer.Length - 1; i++)

{

Sum = Sum ^ buffer[i];

}

request = Convert.ToByte(Sum);

return request;

}

static string ComputeSha256Hash(string rawData)

{

// Create a SHA256

using (SHA256 sha256Hash = SHA256.Create())

{

// ComputeHash - returns byte array

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

// Convert byte array to a string

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

// function clearout a message and returns a key

public byte[] GetAKey(byte[] message)

{

if (message != null)

{

byte[] data = new byte[message[4]];

for (int i = 0; i != message[4]; i++)

{

data[i] = message[i + 5];

}

return data;

}

return null;

}

//function send a message to module return true if operation succesfull

public bool TryToSend(byte[] buffer)

{

// check if port is ready for reading

if (serialPort1.IsOpen)

{

try

{

int[] data = new int[4];

buffer[buffer.Length - 1] = XORCHeck(buffer);

serialPort1.Write(buffer, 0, buffer.Length);

System.Threading.Thread.Sleep(100);//Module need some sleep, or data will zero

if (serialPort1.BytesToRead != 1 && serialPort1.BytesToRead!=0)

{

for (int i = 0; i != 4; i++)

{

data[i] = serialPort1.ReadByte();

}

// MessageBox.Show(data[3].ToString("X"));

serialPort1.DiscardInBuffer();

if (data[3].ToString("X") == "81") return true;

else

return false;

}

else return false;

}

catch (TimeoutException)

{

MessageBox.Show("Ошбикаподключения! Времяожиданияпревышено!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return false;

}

}

else

{

// give error warning

MessageBox.Show("Портнеоткрыт!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return false;

}

}

//function sends message and returns message from the module

private byte[] SendAndReadMessage(byte[] message)

{

// check if port is ready for reading

if (serialPort1.IsOpen)

{

try

{

message[message.Length - 1] = XORCHeck(message);

serialPort1.Write(message, 0, message.Length);

System.Threading.Thread.Sleep(100);//Module need some sleep, or data will zero

if (serialPort1.BytesToRead != 1 && serialPort1.BytesToRead !=0)

{

byte[] buffer = new byte[serialPort1.BytesToRead];

serialPort1.Read(buffer, 0, buffer.Length);

serialPort1.DiscardInBuffer();

return buffer;

}

else return null;

}

catch (TimeoutException)

{

MessageBox.Show("Ошбикаподключения! Времяожиданияпревышено!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return null;

}

}

else

// give error warning

MessageBox.Show("Портнеоткрыт!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return null;

}

private void Output(byte[] buffer)

{

for (byte i = 0; i != buffer.Length; i++)

{

TB\_Output.Text += buffer[i].ToString("X");

TB\_Output.Text += '-';

}

if (TB\_Output.Text.Length != 0) TB\_Output.Text = TB\_Output.Text.Substring(0, TB\_Output.Text.Length - 1);//Clear the last '-' symbol

TB\_Output.Text += Environment.NewLine;

TB\_Output.Text += "--------------------------";

TB\_Output.Text += Environment.NewLine;

}

public Form1()

{

InitializeComponent();

FindPorts();

}

private void CB\_COMSelect\_SelectedIndexChanged(object sender, EventArgs e)

{

TB\_Output.Text = String.Empty;

progressBar1.Value = 0;

if (CB\_BaudSelect.Text == String.Empty || CB\_COMSelect.Text == String.Empty)

LBStatus.Text = "Please Select Port Settings";

else

{

try

{

try

{

// make sure port isn't open

if (!serialPort1.IsOpen)

{

serialPort1.PortName = CB\_COMSelect.Text;

serialPort1.BaudRate = Int32.Parse(CB\_BaudSelect.Text);

//open serial port

serialPort1.Open();

progressBar1.Value = 100;

LBStatus.Text = "Connection is OK";

CB\_COMSelect.Enabled = false;

ReconnectBtn.Enabled = false;

}

else

LBStatus.Text = "Port isn't openned";

}

catch (System.IO.IOException) { LBStatus.Text = "COM timeout error"; }

}

catch (UnauthorizedAccessException)

{

LBStatus.Text = "UnauthorizedAccess";

}

}

}

private void CB\_BaudSelect\_SelectedIndexChanged(object sender, EventArgs e)

{

serialPort1.BaudRate = Int32.Parse(CB\_BaudSelect.Text);

}

//ReadButton

private void ReadBtn\_Click(object sender, EventArgs e)

{

// check if port is ready for reading

if (serialPort1.IsOpen)

{

try

{

if (TryToSend(bufReadT5577)) Output(SendAndReadMessage(bufReadT5577));

else

if (TryToSend(bufReadUID)) Output(SendAndReadMessage(bufReadUID));

else LBStatus.Text = "Read Err";

}

catch (TimeoutException)

{

LBStatus.Text = "Timeout Exception";

}

// Disable the init button

// the asynchronous operation is done.

CB\_COMSelect.Enabled = false;

}

else

// give error warning

LBStatus.Text = "Port Not Opened";

}

//disconnection button

private void DisconnectBtn\_Click(object sender, EventArgs e)

{

serialPort1.Close();

progressBar1.Value = 0;

CB\_COMSelect.Enabled = true;

LBStatus.Text = "Port has been disconnected";

ReconnectBtn.Enabled = true;

}

private void RescanBtn\_Click(object sender, EventArgs e)

{

CB\_COMSelect.Items.Clear();

FindPorts();

}

private void ReconnectBtn\_Click(object sender, EventArgs e)

{

TB\_Output.Text = String.Empty;

progressBar1.Value = 0;

if (CB\_BaudSelect.Text == String.Empty || CB\_COMSelect.Text == String.Empty)

LBStatus.Text = "Please Select Port Settings";

else

{

try

{

// make sure port isn't open

if (!serialPort1.IsOpen)

{

serialPort1.PortName = CB\_COMSelect.Text;

serialPort1.BaudRate = Int32.Parse(CB\_BaudSelect.Text);

//open serial port

serialPort1.Open();

progressBar1.Value = 100;

LBStatus.Text = "Connection is OK";

CB\_COMSelect.Enabled = false;

ReconnectBtn.Enabled = false;

}

else

LBStatus.Text = "Port isn't openned";

}

catch (UnauthorizedAccessException)

{

LBStatus.Text = "UnauthorizedAccess";

}

}

}

private void serialPort1\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

if (serialPort1.IsOpen)

{

try

{

TB\_Output.Invoke((MethodInvoker)delegate

{

});

}

catch (TimeoutException)

{

LBStatus.Invoke((MethodInvoker)delegate

{

LBStatus.Text = "Timeout Exception";

});

}

}

else

{// give error warning

LBStatus.Invoke((MethodInvoker)delegate

{

LBStatus.Text = "Port Not Opened";

});

}

}

private void button1\_Click(object sender, EventArgs e)

{

TB\_Output.Text = String.Empty;

}

private void SendBtn\_Click(object sender, EventArgs e)

{

if (serialPort1.IsOpen)

{

// this will read manually

try

{

if (TryToSend(bufReadT5577)) LBStatus.Text = "125KHz";

else

if (TryToSend(bufReadUID)) LBStatus.Text = "13.56MHz";

else LBStatus.Text = "Err";

}

catch (TimeoutException)

{

LBStatus.Text = "Timeout Exception";

}

}

else

// give error warning

LBStatus.Text = "Port Not Opened";

}

private void button2\_Click(object sender, EventArgs e)

{

byte[] message = bufWriteT5577;

for (byte i = 0; i != key.Length; i++)

{

message[5 + i] = key[i];

}

if (TryToSend(message) == true) MessageBox.Show("Opertaion Success!");

else MessageBox.Show("Opertaion fail!");

}

private void button3\_Click(object sender, EventArgs e)

{

var rand = new Random();

rand.NextBytes(key);

TB\_Input.Text = BitConverter.ToString(key);

}

private void tabPage2\_Click(object sender, EventArgs e)

{

AuthForm form = new AuthForm();

form.Show();

}

//DATA BASE CODE PART

public bool AccessFlag = false;

public string connect = "server=localhost;user=root;password=1337;database=vaulttechsecure;port=3306;charset=utf8;convert zero datetime=True";

string SerialNumber;

public byte[] key = new byte[5];

void update()

{

try

{

MySqlConnection mycon = new MySqlConnection(connect);

mycon.Open();

string sql = "SELECT \* FROM residents;";

MySqlDataAdapter dataAdapter = new MySqlDataAdapter(sql, mycon);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (TB\_Name.Text != "" && TB\_LastName.Text != "" && TB\_GroupCode.Text != "" && SerialNumber != "")

{

try

{

string str = "server=localhost;user=root;password=1337;database=vaulttechsecure;port=3306";

MySqlConnection mycon = new MySqlConnection(connect);

mycon.Open();

string query = "INSERT INTO `residents`(`Name` , `Last Name` , `GroupCode` , `SerialNumber`) VALUES ('" + TB\_Name.Text + "', '" + TB\_LastName.Text + "', '" + Convert.ToInt32(TB\_GroupCode.Text) + "', '" + SerialNumber + "');";

MySqlCommand command = new MySqlCommand(query, mycon);

command.ExecuteNonQuery();

mycon.Close();

update();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

else

MessageBox.Show("Всеполядолжныбытьзаполнены!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

TB\_Name.Text = ""; TB\_LastName.Text = ""; TB\_GroupCode.Text = ""; TB\_CardNumber.Text = ""; SerialNumber = "";

}

private void button6\_Click(object sender, EventArgs e)

{

var rand = new Random();

rand.NextBytes(key);

SerialNumber = BitConverter.ToString(key);

TB\_CardNumber.Text = SerialNumber;

byte[] message = bufWriteT5577;

for (byte i = 0; i != 5; i++)

{

message[5 + i] = key[i];

}

if (TryToSend(message) == true) MessageBox.Show("Card writing success!");

else MessageBox.Show("Card writing fail!");

}

private void button7\_Click(object sender, EventArgs e)

{

if (TryToSend(bufReadT5577) == true) MessageBox.Show("Card was found!");

else MessageBox.Show("Card read fail!");

SerialNumber = BitConverter.ToString(GetAKey(SendAndReadMessage(bufReadT5577)));

TB\_CardNumber.Text = SerialNumber;

}

private void button5\_Click(object sender, EventArgs e)

{

update();

}

private void Module\_Selected(object sender, TabControlEventArgs e)

{

Form1.ActiveForm.Hide();

update();

AuthForm form = new AuthForm();

form.Show();

}

private void button8\_Click(object sender, EventArgs e)

{

string idNumber = "";

idNumber = Interaction.InputBox("Enter id number of member to delete", "Deleting a member","");

if (idNumber != "")

try

{

using (MySqlConnection connection = new MySqlConnection(connect))

{

var command = new MySqlCommand("SET FOREIGN\_KEY\_CHECKS=0; Delete from residents where idresidents = @idresidents; SET FOREIGN\_KEY\_CHECKS=1", connection);

command.Parameters.Add("@idresidents", MySqlDbType.Int32).Value = Convert.ToInt32(idNumber);

connection.Open();

command.ExecuteNonQuery();

update();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

else MessageBox.Show("Plsease enter an id to delete a member!");

}

}

}