

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 10.05.03 –

Информационная безопасность
автоматизированных систем

Специализация -

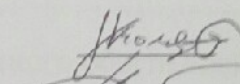
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

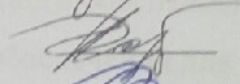
на тему

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РАСПОЗНАВАНИЯ И ДЕКОДИРОВАНИЯ
ШТРИХОВОГО КОДА DATA MATRIX В УСТРОЙСТВЕ ВВОДА КЛЮЧЕВОЙ
ИНФОРМАЦИИ

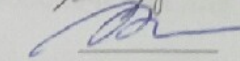
Студент

 Колядо К.А.

Руководитель

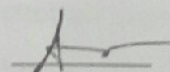
 Картамышев И.В.

Нормоконтролёр

 Иванов А.П.

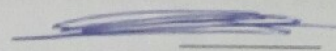
Рецензент заместитель начальника НИОИ

АО «ПНИЭИ»

 Акмаев А.Ж.

Работа допущена к защите (протокол заседания кафедры от 21.06.19 № 16)

Заведующий кафедрой

 С.И. Зефирова

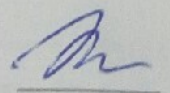
Работа защищена с отметкой

Одобрено

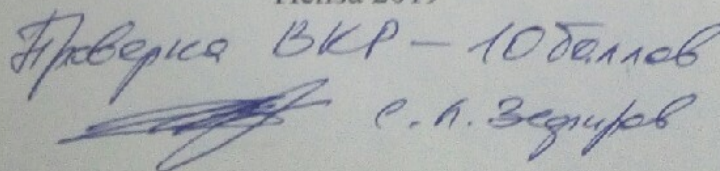
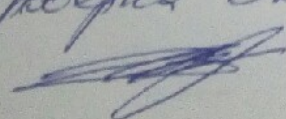
(протокол заседания ГЭК

от 25.06.19 № 21)

Секретарь ГЭК

 А.П. Иванов

Пенза 2019

 Проверка ВКР – 10 Беллов
 С.И. Зефирова

Оглавление

Введение.....	4
1 Обзор возможностей использования штрихового кода Data Matrix в сфере информационной безопасности.....	6
1.1 Обзор существующих технологий штрихового кодирования.....	6
1.2 Сравнение штриховых кодов.....	8
1.2.1 Сравнение штриховых кодов по объему кодируемой информации.	9
1.2.2 Сравнение по восстанавливающей способности алгоритмов контроля и исправления ошибок.....	9
1.2.3 Результаты сравнения штриховых кодов.....	10
1.3 Обзор существующего программного обеспечения для распознавания и декодирования символа штрихового кода Data Matrix.....	11
1.4 Сравнение качества распознавания символов штрихового кода Data Matrix у существующего программного обеспечения.....	12
1.5 Бумажный носитель информации с нанесенным на него символом штрихового кода Data Matrix.....	14
2 Разработка программного обеспечения распознавания и декодирования штрихового кода Data Matrix.....	17
2.1 Разработка архитектуры программного обеспечения.....	17
2.1.1 Архитектура программного обеспечения для устройства ввода ключевой информации.....	17
2.1.2 Архитектура программного обеспечения распознавания и декодирования штрихового кода Data Matrix.....	19
2.2 Анализ символа Data Matrix используемого в качестве носителя информации.....	23
2.3 Описание алгоритма определения черной и белой области на изображении.....	27
2.4 Алгоритм распознавания символа ШК Data Matrix.....	30
2.5 Алгоритм функции контроля и исправления ошибок.....	39

2.6 Алгоритм функции декодирования данных.....	44
2.7 Алгоритм функции гарантированного уничтожения информации.....	44
2.8 Алгоритм контроля целостности программного обеспечения.....	45
3 Обзор архитектуры процессора 1892BM10Я и его основных характеристик	48
4 Определение параметров быстродействия, объема используемой оперативной памяти и объема полученного кода разработанного программного обеспечения для процессора 1892BM10Я.....	53
4.1 Определение быстродействия разработанного программного обеспечения.....	53
4.2 Определение требуемых ресурсов памяти разработанного программного обеспечения.....	53
5 Выявление пороговых значений искажений изображения для разработанного ПО.....	55
5.1 Тестирование программного обеспечения на изображениях без внесения дополнительных искажений.....	55
5.2 Обзор возможных искажений изображения.....	55
5.3 Тестирование программного обеспечения на изображениях с дополнительными искажениями.....	56
6 Обзор вредных психофизиологических факторов, влияющих на сердечно- сосудистую систему специалиста по защите информации. Методы и средств физической культуры, снижающие их воздействие.....	59
Заключение.....	66
Список использованных источников.....	68

ВВЕДЕНИЕ

В настоящее время в области защиты информации имеется тенденция использования штрихового кодирования для предоставления на бумажном носителе ключевой информации. Данные носители информации обладают высокими показателями надежности и могут храниться длительное время.

В рамках дипломного проектирования стоит задача создания программного обеспечения распознавания символа штрихового кода Data Matrix версии ECC 200, размером 88x88 модулей в устройстве ввода ключевой информации с использованием языка Си (С89).

Для того, чтобы создать устройство, позволяющее сканировать бумажные носители ключевой информации необходимо решить ряд задач, одной из которых является разработка программного обеспечения для распознавания символов штрихового кода. Поскольку получаемое изображение обладает низкой разрешающей способностью и плохим контрастом, вследствие чего использование готового и свободно распространяемого программного обеспечения невозможно из-за плохих показателей качества распознавания, было принято решение разрабатывать собственное программное обеспечение для распознавания символа штрихового кода.

Для того, чтобы создать носитель ключевой информации, позволяющий закодировать в нем максимально возможное количество информации, необходимо решить ряд задач, в число которых входит:

- определение геометрических размеров носителя;
- выбор штрихового кода, позволяющего закодировать на определенном пространстве наибольшее количество информации;
- выбор штрихового кода у которого алгоритмы контроля и исправления ошибок обладают достаточной восстанавливающей способностью.

Технология штрихового кодирования Data Matrix позволяет кодировать большой объем полезной информации, имеет достаточный уровень коррекции ошибок и имеет маленький размер свободной зоны, что позволяет сэкономить пространство на носителе ключевой информации и является оптимальной для использования в качестве представления на бумажном носителе ключевой информации.

1 Обзор возможностей использования штрихового кода Data Matrix в сфере информационной безопасности

1.1 Обзор существующих технологий штрихового кодирования

Штриховой код — графическая информация, наносимая на поверхность, маркировку или упаковку изделий, предоставляющая возможность считывания её техническими средствами — последовательность чёрных и белых полос, либо других геометрических фигур [1].

Существуют линейные и двумерные штриховые коды. Линейными называются штриховые коды, читаемые в одном направлении, по горизонтали. Наиболее распространённые линейные коды: EAN, UPC, Code128, Codabar. Объем данных таких кодов не ограничен и зависит от размера кода. В сравнении с двумерными, данные штриховые коды имеют значительно больший размер при той же информационной ёмкости, из-за чего не рассматриваются [1].

Двухмерные штриховые коды были разработаны для кодирования большого объёма информации. Распознавание такого кода проводится в двух измерениях (по горизонтали и по вертикали)[1].

Двухмерные коды подразделяются на многоуровневые и матричные. Многоуровневые штриховые коды появились исторически ранее, и представляют собой поставленные друг на друга несколько обычных линейных кодов. Матричные же коды более плотно упаковывают информационные элементы по вертикали[1].

В настоящее время разработано множество двумерных штриховых кодов. Был проведен первичный сравнительный анализ различных двумерных штриховых кодов, для определения пригодности их использования в качестве носителей ключевой информации. Критерии по которым проводилось сравнение приведены в таблице 1. Результат приведен в таблице 1. Для использования в качестве носителя ключевой информации подходят следующие штриховые коды: Aztec Code, QR code, Data Matrix, PDF 417.

Таблица 1 – Сравнительный анализ двумерных штриховых кодов [2]

Характеристика	Штриховой код							
	Actec Code	Data Matrix	QR code	PDF 417	Maxi code	Shot code	Microsoft Tag	Ez code
Внешний вид								
Считывание кода автономно	Да	Да	Да	Да	Да	Нет, требуется обращение к интернет-серверу	Нет, требуется обращение к интернет-серверу	Нет, требуется обращение к интернет-серверу
Печать на черно-белом принтере	Да	Да	Да	Да	Да	Да	Нужен цветной принтер	Да
Оптимизация для существующих технологий печати с низким разрешением	Да. Символ состоит из квадратных пикселей	Да. Символ состоит из квадратных пикселей	Да. Символ состоит из квадратных пикселей	Да. Символ состоит из квадратных пикселей	Нет. Центр состоит из окружностей Затруднена печать с малым разрешением	Нет. Символ состоит из окружностей Затруднена печать с малым разрешением	Не совсем. В символ входят треугольники Информацию несут их центры	Да. Символ состоит из квадратных пикселей
Коррекция ошибок	Есть	Есть	Есть	Есть	Есть	Есть	Есть	Есть
Открытость формата	Формат открыт	Формат открыт	Формат открыт	Формат открыт	Формат открыт	Формат открыт	Формат закрыт	Формат открыт

1.2 Сравнение штриховых кодов

Было принято решение исключить из штриховой код PDF 417 из списка сравниваемых кодов ввиду того, что данный штриховой код содержит слишком большое количество дополнительной информации, что значительно сокращает его информационную емкость. Шаблон символа штрихового кода PDF 417 приведен на рисунке 1, где видно, что количество служебной информации составляет более 50% символа штрихового кода.



Рисунок 1 – Структура символа штрихового кода PDF-417 [3]

Для того, чтобы сравнить штриховые коды было принято решение установить следующие ограничения на размер символа:

- размер одного модуля должен быть равен 3 пикселям;
- символ должен уместиться на изображении размером 270x270 пикселей.

Таким образом максимально возможный символ штрихового кода равен 90x90 модулей (270 разделить на 3 равно 90). Эти ограничения связаны с геометрическими размерами носителя информации.

Анализировались штриховые символы Aztec Code, QR code, Data Matrix.

В соответствии со стандартом каждого символа, для сравнения была выбрана наиболее подходящая по размеру, количеству модулей версия символа штрихового кода. Для штрихового кода Aztec Code это символ размером 87x87 модулей [4], для штрихового кода QR code символ размером 89x89 модулей [5], для штрихового кода Data Matrix символ размером 88x88 модулей [6].

1.2.1 Сравнение штриховых кодов по объему кодируемой информации

Было проведено сравнение по объему кодируемой полезной информации для каждого из выбранных символов. Результаты приведены в таблице 2. Ввиду того, что штриховой код Aztec Code требует, чтобы пользователь задавал объем полезной информации и восстанавливающую способность кода вручную было принято решение рассчитывать объем полезной информации исходя из того, что восстанавливающая способность алгоритмов контроля и исправления ошибок будет составлять 15% слов в символе.

Таблица 2 – Результаты сравнения штриховых кодов по объему кодируемой полезной информации

Штриховой код	Версия символа	Всего модулей	Информационных модулей	Модулей в шаблонах	Всего слов данных	Количество полезной информации (байт)
Aztec Code	87x87	7569	6520	1049	652	454
Data Matrix	88x88	7744	6400	1344	800	576
QR code	89x89 L	7921	7211	710	901	718
	89x89 M	7921	7211	710	901	560
	89x89 Q	7921	7211	710	901	394
	89x89 H	7921	7211	710	901	310

1.2.2 Сравнение по восстанавливающей способности алгоритмов контроля и исправления ошибок

Было проведено сравнение штриховых кодов по восстанавливающей способности алгоритмов контроля и исправления ошибок. Результаты приведены в таблице 3.

Было принято решение считать восстанавливающую способность достаточной при максимально возможном восстановлении приблизительно 15% слов символа.

Для символа Aztec Code, чтобы задать максимальную восстанавливающую способность 15%, требуется выделить 196 слов под коды коррекции ошибок. Поскольку используются коды Рида-Соломона, восстанавливающая способность в этом случае будет равна 98 словам, что составляет 15% от общего объема кодовых слов: $98/650 \cdot 100 = 15$.

Символ QR code позволяет пользователю выбрать один из четырех режимов кодирования и создания кодовых слов исправления ошибок: L – с низким уровнем коррекции ошибок, M – с рекомендованным уровнем коррекции ошибок, Q – повышенным уровнем коррекции ошибок, H – высоким уровнем коррекции ошибок.

Таблица 3 – Результаты сравнения алгоритмов контроля и исправления ошибок штриховых кодов

Штриховой код	Версия символа	Слов исправления ошибок	Блоков коррекции ошибок	Корректирующая способность	
				Минимум	Максимум
Aztec Code	87x87	196	4	24	98
Data Matrix	88x88	224	4	28	112
QR code	89x89 L	180	6	15	90
	89x89 M	338	13	13	169
	89x89 Q	504	18	14	252
	89x89 H	588	21	14	294

1.2.3 Результаты сравнения штриховых кодов

Результаты сравнения из таблицы 2 и таблицы 3 были сведены в одну итоговую таблицу, приведенную в таблице 4. Было принято решение, что в символе QR code вариант кодирования L не подходит из-за низкой восстанавливающей способности алгоритмов контроля и исправления ошибок. Варианты кодирования Q и H не подходят из-за слишком малого количества

полезной информации. Принимая во внимание все вышеизложенное, было принято решение использовать для анализа QR code версии M.

Принимая во внимания вышеизложенные результаты анализа, можно сделать вывод, что целесообразнее всего применять в качестве носителя ключевой информации символ штрихового кода Data Matrix из-за того, что данный символ имеет наивысший показатель по количеству кодируемой полезной информации, имеет достаточный уровень коррекции ошибок и в сравнении с штриховым кодом QR code имеет меньший размер свободной зоны, что позволяет сэкономить пространство на носителе ключевой информации.

Таблица 4 – Результаты сравнения штриховых кодов

Штриховой код	Версия символа	Всего слов	Слов данных	Слов коррекции ошибок	Восстанавливающая способность		Необходимая свободная зона
					Слов	%	
Aztec Code	87x87	652	454	196	98	15	1 модуль
Data Matrix	88x88	800	576	224	112	14	1 модуль
QR code	89x89 M	901	560	338	169	18	4 модуля

1.3 Обзор существующего программного обеспечения для распознавания и декодирования символа штрихового кода Data Matrix

Было принято решение анализировать существующее программное обеспечение исходные коды которого находятся в открытом доступе. Для поиска такого программного обеспечения использовался интернет ресурс Git Hub. В ходе анализа исходных кодов на Git Hub были обнаружены всего две библиотеки, позволяющие распознавать символ штрихового кода Data Matrix: libdmtx и zxing. Остальное программное обеспечение либо не закончено, либо использует данные библиотеки.

Libdmtx – библиотека, распространяемая по лицензии BSD, позволяющая программно распознавать и создавать штриховые коды Data Matrix версии ECC200. Библиотека была изначально написана на языке Си, стандарт C89,

после чего была портирована на другие языки программирования: python 3, java, php, delphy. Адрес проекта в сети интернет: <https://github.com/dmtx/libdmatrix/blob/master/> [7].

ZXing ("zebra crossing") – открытое программное обеспечение, позволяющее программно распознавать и создавать 1D / 2D штриховые коды на изображении. Поддерживает различные 2D штриховые коды:

- QR-code;
- Data Matrix;
- Aztec Code;
- Maxi Code;
- PDF 417.

Библиотека изначально была написана на языке программирования Java. После этого была портирована на другие языки программирования такие как: C++, Objective C, Qt framework, Ruby, JavaScript, Python, .NET и C#, PHP, Delphi. Адрес проекта в сети интернет: <https://github.com/zxing/zxing/blob/master/> [8].

1.4 Сравнение качества распознавания символов штрихового кода Data Matrix у существующего программного обеспечения

Было проведено сравнение качества распознавания символов штрихового кода Data Matrix у библиотек libdmatrix и zxing. Для этого каждая библиотека была скачана, скомпилирована. Для того, чтобы обеспечить работу с изображением у библиотеки libdmatrix была дополнительно скачана библиотека libpng. Было осуществлено фотографирование носителя символа штрихового кода Data Matrix. Всего было получено порядка 100 фотографий. Одна из фотографий приведена на рисунке 2.

После этого было осуществлено распознавание всех полученных изображений каждой из библиотек. Результат приведен на рисунке 3.

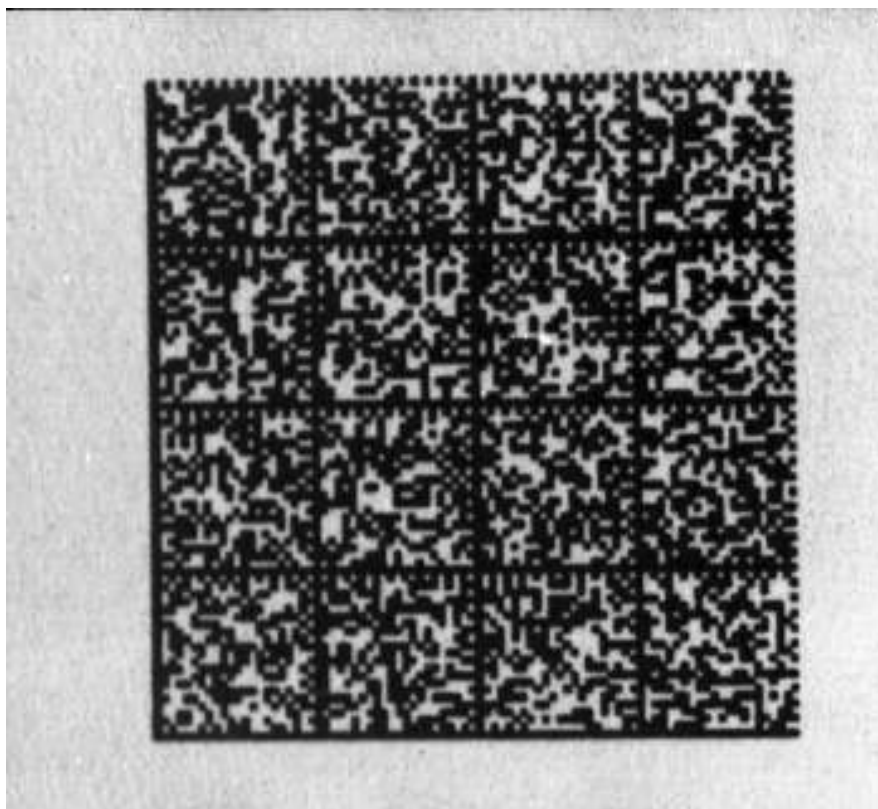


Рисунок 2 – Полученное изображение символа Data Matrix

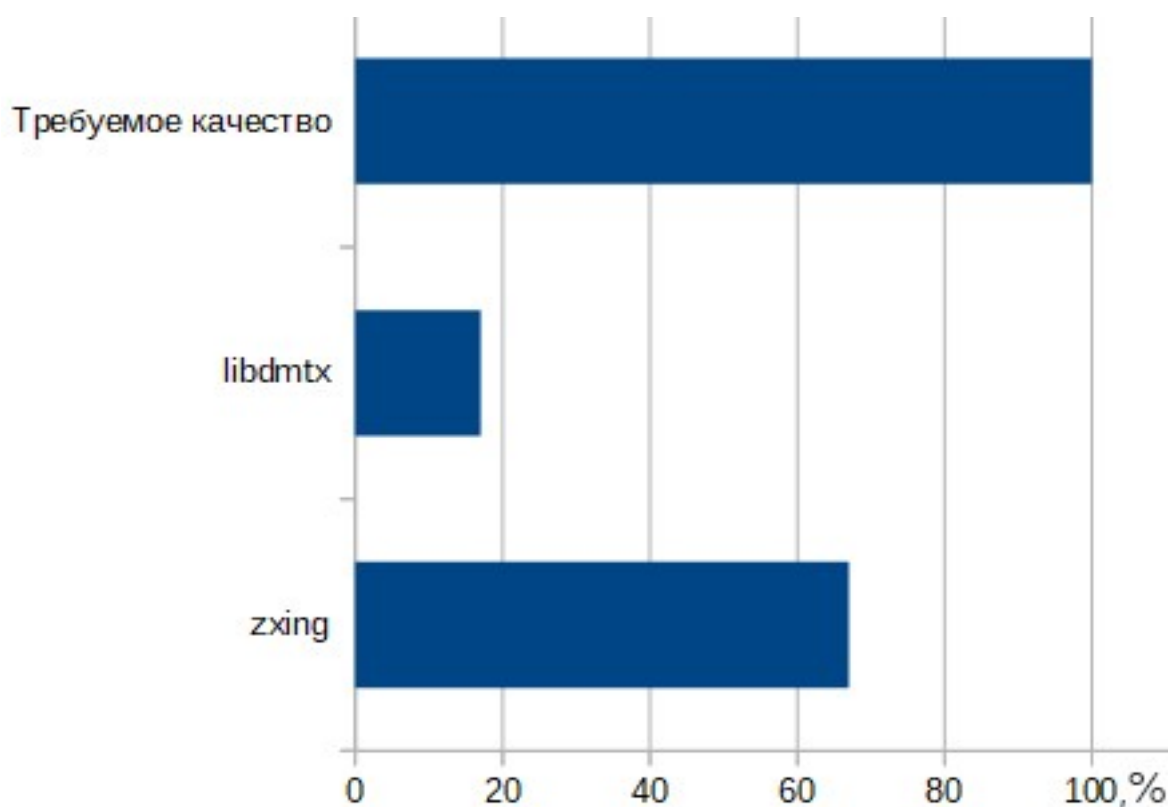


Рисунок 3 – Результаты распознавания имеющимся в открытом доступе программным обеспечением

Как видно из результатов распознавания ни одна из библиотек не позволяет осуществить 100% распознавание всех символов штрихового кода. Это связано с низким разрешением получаемого изображения — 350x320 и плохим контрастом изображения.

Таким образом, принимая во внимание вышеизложенные результаты тестирования существующего программного обеспечения можно сделать вывод, что на данный момент не существует дешевого, и, тем более, свободно распространяемого программного обеспечения которое написано на языке Си (C89) и позволяет с вероятностью близкой к 100% осуществлять распознавание получаемых изображений символа штрихового кода Data Matrix. Кроме того, уже разработанное программное обеспечение не осуществляет такую процедуру как гарантированное уничтожение информации.

1.5 Бумажный носитель информации с нанесенным на него символом штрихового кода Data Matrix

На носители информации кроме самой марки символа штрихового кода Data Matrix должно быть место для помещения некоторой пояснительной информации, указывающей, для чего этот носитель был создан. Также на носителе должно быть эффективно использовано место для размещения символа штрихового кода Data Matrix. При этом фотографируется не вся область, а только её часть, в которой возможно размещение символа Data Matrix.

Носитель ключевой информации имеет геометрические размеры 63x40 мм. Область для размещения символа штрихового кода Data Matrix, которая попадает в область видимости фотографирующего элемента и включает зону для компенсации погрешности, которая получается при неточной установке носителя в сканер, имеет размер 48x21 мм. Поскольку «минимальный размер свободной зоны вокруг символа Data Matrix должен быть равен размеру

Х со всех четырех сторон, где Х — размер модуля», было принято решение принять размеры рабочей области 47х20 мм.

Расчет размеров исходного изображения, на котором будет размещен символ Data Matrix осуществлялся по формуле (1), приведенной ниже

$$p = n / K * l / a, \quad (1)$$

где:

n – размер стороны фотографируемой области в мм,

K - коэффициент оптического искажения камеры,

a - размер одной ячейки на ПЗС-матрице.

Используемая камера в сканере имеет коэффициент оптического искажения равный 3,7, а размер одной ячейки на ПЗС-матрице равен 0,02 мм. Исходя из размеров матрицы, размеров одного поля для символа Data Matrix, которое имеет размер 20х20 мм. был произведен расчет количества пикселей на ПЗС-матрице, которые покроют указанную область. Требуемое количество пикселей равно 270. Исходя из расчетов, исходное изображение, на котором будет размещен символ Data Matrix должен иметь размер 270х270 пикселей.

Вся рабочая область имеет размер 47х20. При расчете по формуле (1) получается, что размер рабочей области в пикселях составляет 635х270 пикселей. На данной области можно разместить два символа штрихового кода Data Matrix.

Принимая во внимание все вышеизложенное, был составлен чертеж носителя ключевой информации с символом штрихового кода Data Matrix, который приведен на рисунке 4.

При размещении таким образом двух символов Data Matrix версии 88х88 количество полезной информации увеличивается в два раза и составляет 1138 байт.

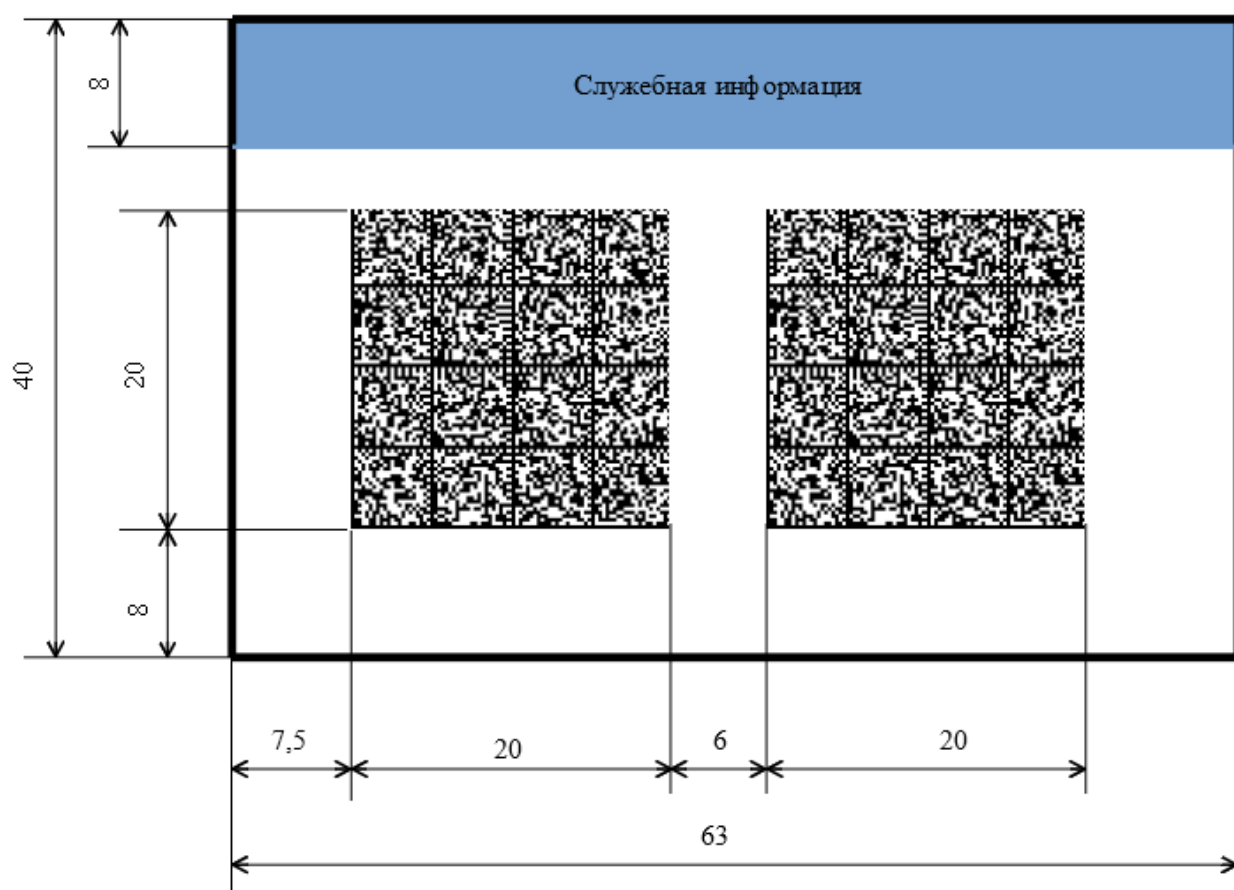


Рисунок 4 – Чертеж носителя ключевой информации с символом штрихового кода Data Matrix

2 Разработка программного обеспечения распознавания и декодирования штрихового кода Data Matrix

2.1 Разработка архитектуры программного обеспечения

2.1.1 Архитектура программного обеспечения для устройства ввода ключевой информации

Согласно стандарту IEEE 1471 - «Рекомендуемые методы описания архитектуры программных систем» архитектура – это базовая организация системы, которая описывает связи между компонентами этой системы (и внешней средой), определяет принципы её проектирования и развития [9].

Было принято решение использовать модульную архитектуру для разработки программного обеспечения. Данный метод позволит обеспечить такие свойства для программного обеспечения как масштабируемость, заменимость модулей, возможность раздельного тестирования модулей [9].

Разрабатываемое программное обеспечение будет использоваться в качестве составной части в другом программном обеспечении, которое состоит из следующих модулей:

- модуль фотографирования;
- модуль анализа изображения;
- модуль взаимодействия с периферийным устройством;
- модуль подсчета контроля целостности;
- модуль гарантированного уничтожения информации.

Диаграмма потоков данных программного обеспечения для устройства ввода ключевой информации приведена на рисунке 5.

Устройство ввода ключевой информации для которого разрабатывается программное обеспечение является составной частью другой аппаратуры. Для взаимодействия с внешними устройствами используется модуль взаимодействия с периферийным устройством. При приеме одной из команд,

модуль взаимодействия с периферийным устройством передает управление одному из модулей, приведенных выше, чтобы тот выполнил соответствующую операцию. Результат выполнения операции передается внешнему устройству.

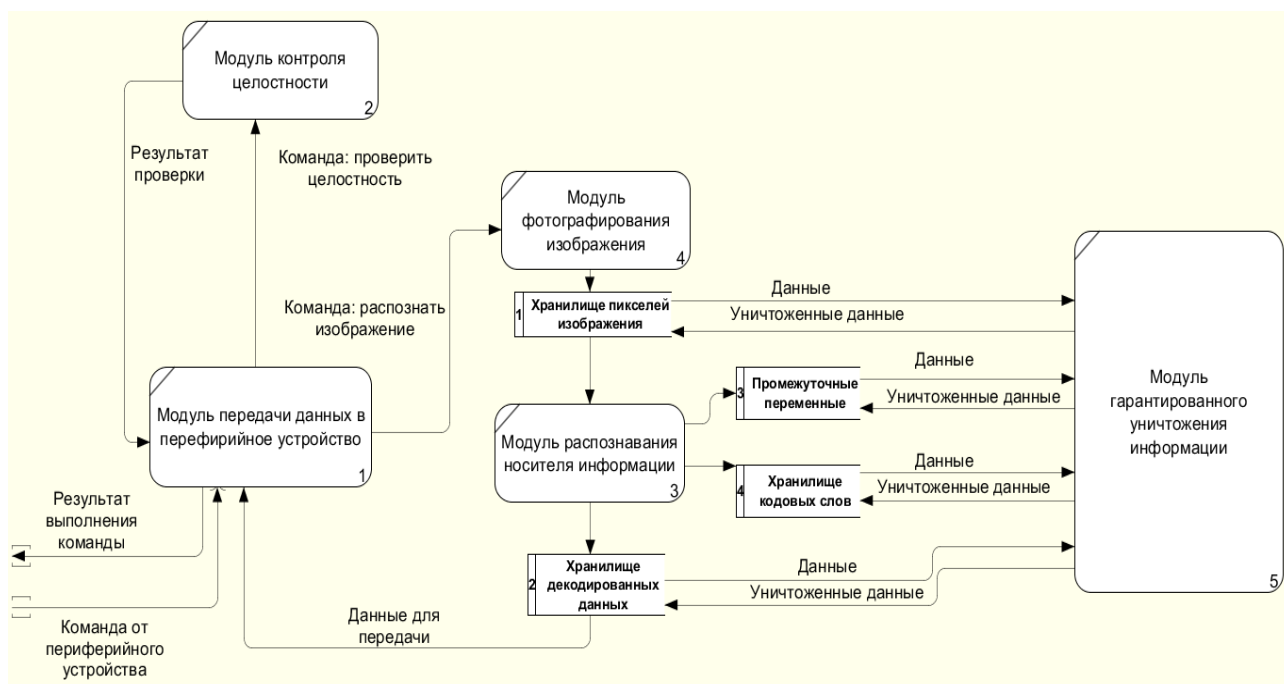


Рисунок 5 – DFD программного обеспечения для устройства ввода ключевой информации

Модуль контроля целостности осуществляет проверку целостности зашифрованного ПО путем подсчета контрольной суммы. После подсчета контрольной суммы, модуль контроля целостности сравнивает её с уже ранее подсчитанной и хранимой в ПЗУ. Это необходимо для того, чтобы обеспечить защиту от повреждений микросхемы. Проверка защиты от НСД осуществляется в рамках общей аппаратуры и не входит в задачи данного модуля.

Модуль фотографирования изображения осуществляет фотографирование носителя ключевой информации, после чего записывает данные о пикселях изображения во внешнее ОЗУ.

Модуль гарантированного уничтожения информации осуществляет затирание области памяти, в которой хранится ключевая информация. Очистка памяти осуществляется методом определенным в руководящем документе ФСТЭК «Автоматизированные системы. Защита от несанкционированного

доступа к информации. Классификация автоматизированных систем и требования по защите информации» для класса защищенности АЗ.

2.1.2 Архитектура программного обеспечения распознавания и декодирования штрихового кода Data Matrix

Разрабатываемое программное обеспечение является единым модулем, в котором можно выделить следующие составные части:

- модуль распознавания изображения;
- модуль исправления ошибок;
- модуль декодирования данных.

Кроме основных модулей, в процессе разработки программного обеспечения было принято решение добавить специальные отладочные модули, которые позволят облегчить процесс отладки. Такими модулями являются модуль маркировки, который позволяет создать увеличенное изображение и печатать на нем отладочную информацию, и модуль автоматизированно распознавания изображений, который позволяет осуществлять распознавание большого количества изображений и строить по результатам отчет. Данные модули являются отладочными и в релизной версии программного обеспечения не присутствуют.

Для того, чтобы описать взаимодействие между модулями программного обеспечения распознавания и декодирования штрихового кода Data Matrix было принято решение использовать диаграмму потоков данных (DFD). Диаграмма первого уровня приведена на рисунке 6, диаграмма второго уровня приведена на рисунке 7.

Первоначально данные представлены в виде пикселей изображения. Эти данные подаются на вход модуля распознавания изображения. Данный модуль осуществляет распознавание символа штрихового кода Data Matrix и составляет

кодовые слова данных по результатам распознавания. Важно отметить, что распознавание может быть неточным, и некоторые кодовые слова данных будут составлены с ошибками.

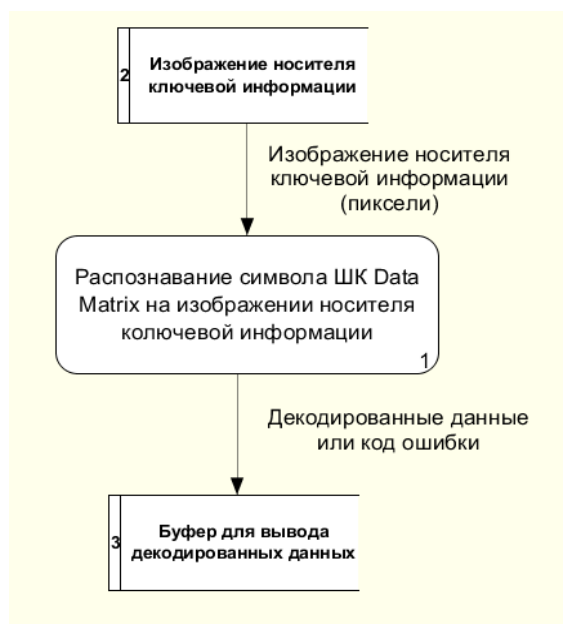


Рисунок 6 – DFD программного обеспечения распознавания и декодирования штрихового кода Data Matrix первого уровня

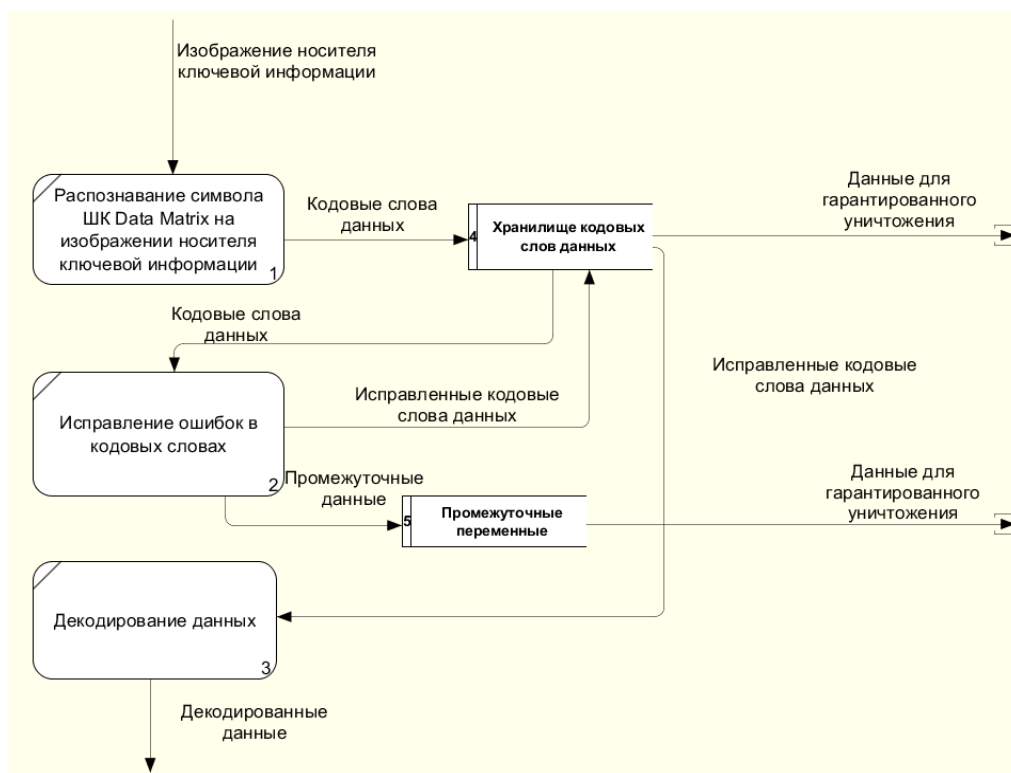


Рисунок 7 – DFD программного обеспечения распознавания и декодирования штрихового кода Data Matrix второго уровня

После успешного завершения процедуры распознавания символа ШК Data Matrix на изображении, кодовые слова данных передаются в качестве входного параметра модулю исправления ошибок. Результатом работы модуля являются исправленные кодовые слова данных.

Последняя процедура — декодирование данных. Эта процедура осуществляется модулем декодирования данных. Данному модулю в качестве входного параметра подаются исправленные кодовые слова данных. Результатом работы модуля являются декодированные данные.

Таким образом, принимая во внимание всё вышеизложенное, можно сделать вывод, что программное обеспечение имеет следующий интерфейс: на вход подается изображение носителя ключевой информации, на выходе получают декодированные данные или код ошибки.

Данный интерфейс позволит при необходимости заменить процедуру распознавания символа Data Matrix на процедуру распознавания другого символа штрихового кода.

Как показано на рисунке 7, в программе обрабатываемые данные, содержащие ключевую информацию, хранятся в следующих областях памяти:

- массив с пикселями изображения;
- массив с кодовыми словами данных;
- буфер, в котором хранятся декодированные данные;
- промежуточные переменные.

Для обеспечения гарантированного уничтожения информации данные затираются с использованием функций модуля гарантированного уничтожения информации, который входит в состав общего программного обеспечения для устройства ввода ключевой информации. Данные затираются в следующем порядке:

- пиксели изображения - после процедуры распознавания символа ШК Data Matrix на изображении;

- кодовые слова данных - после декодирования данных;
- буфер, в котором хранятся декодированные данные - после передачи данных внешнему устройству;
- промежуточные переменные - после прекращения обработки информации, хранимой в переменных.

Принимая во внимание все вышеизложенное был составлен алгоритм работы программного обеспечения, приведенный на рисунке 8.

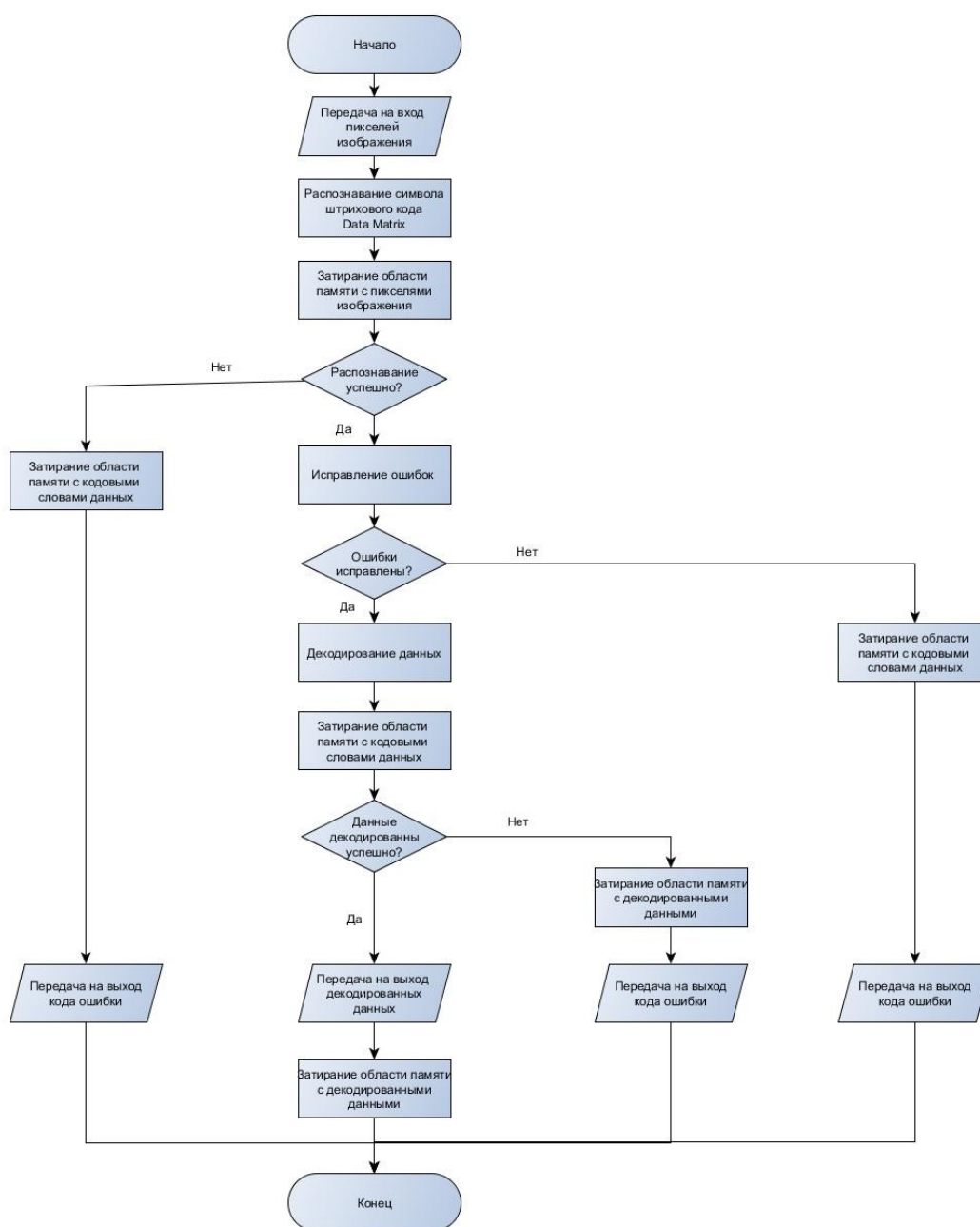


Рисунок 8 – Алгоритм работы программного обеспечения

2.2 Анализ символа Data Matrix используемого в качестве носителя информации

Предполагается использовать в качестве носителя информации символ ШК Data Matrix размером 88*88 модулей, где информационная составляющая равна 80*80 модулей, пример такого символа приведен на рисунке 9.

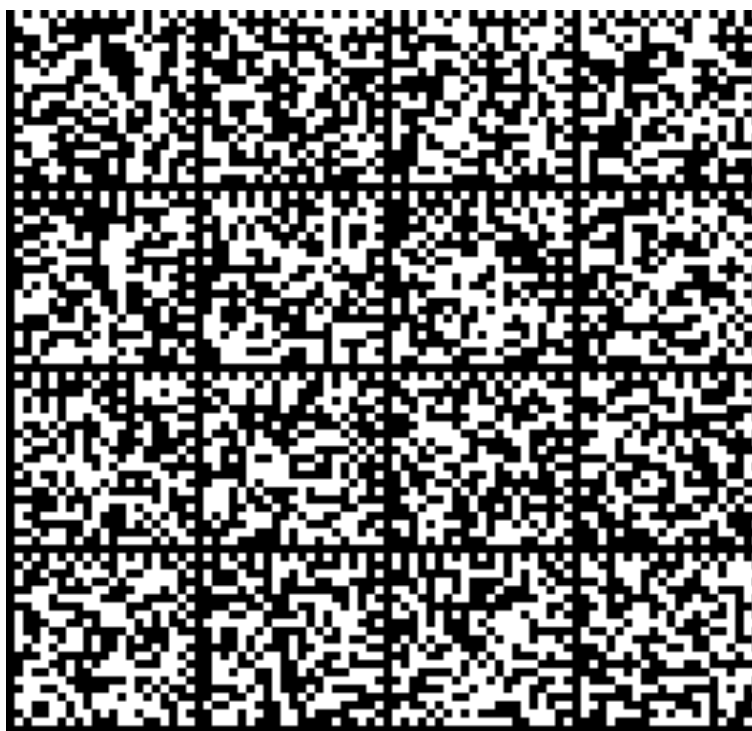


Рисунок 9 – Пример используемого символа Data Matrix

Символ Data Matrix данной версии состоит из 16 составных символов Data Matrix каждый из которых имеет размер 22x22 модулей, где информационная составляющая равна 20x20 модулей. При кодировании данных схемой b256, позволяющей кодировать любые байты данных информационная емкость составляет 573 байта. В случае использования технологии структурного соединения, предусмотренной стандартом Data Matrix, ГОСТ Р ИСО/МЭК 16022 — 2008, количество полезной информации в одном символе составляет 569 байт.

Для кодирования большего объема информации предполагается использовать структурное соединение символов. Пример расположения

символов Data Matrix при структурном соединении приведен на рисунке 10. Структурное соединение позволяет объединить до 16 различных символов Data Matrix, чтобы закодировать данные большого объема. В этом случае данные разделяются на несколько блоков, каждый из которых кодируется независимым символом Data Matrix.

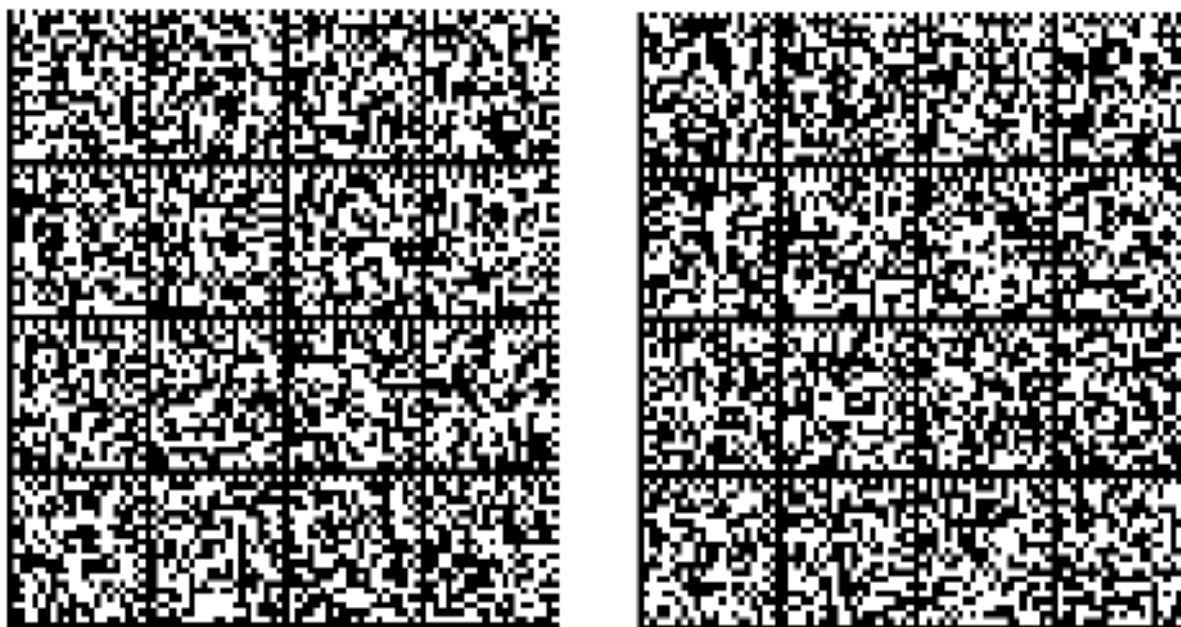


Рисунок 10 – Расположение символов Data Matrix при структурном соединении двух символов Data Matrix на одном носителе

В символах Data Matrix реализовано помехоустойчивое кодирование Рида-Соломона. В используемом символе присутствуют четыре блока Рида-Соломона, каждый из которых имеет размер 200 слов, 144 информационных, 56 проверочных, восстанавливающая способность равна 28 словам. Блоки Рида-Соломона составлены таким образом, что в них чередуются кодовые слова данных. Это позволяет повысить восстанавливающую способность. Схема чередования слов в блоках Рида-Соломона приведена на рисунке 11.

При этом кодовые слова в символе Data Matrix располагаются следующим образом: каждое кодовое слово состоит из восьми бит — восьми модулей, которые располагаются таким образом, как приведено на рисунке 12.

Поток кодовых слов	Кодовые слова данных										Кодовые слова исправления ошибок													
	1	2	3	4	...		573	574	575	576	1	2	3	4	...		221	222	223	224				
Блок 1	Кодовые слова данных										Кодовые слова исправления ошибок													
	1	5	...				569	573				1	5	...				217	221					
Блок 2	Кодовые слова данных										Кодовые слова исправления ошибок													
		2	6		...			570	574					2	6	...				218	222			
Блок 3	Кодовые слова данных										Кодовые слова исправления ошибок													
			3	7	...				571	575				3	7	...				219	223			
Блок 4	Кодовые слова данных										Кодовые слова исправления ошибок													
				4	8	...				572	576				4	8	...				220	224		

Рисунок 11 – Схема чередования слов в блоках Рида-Соломона

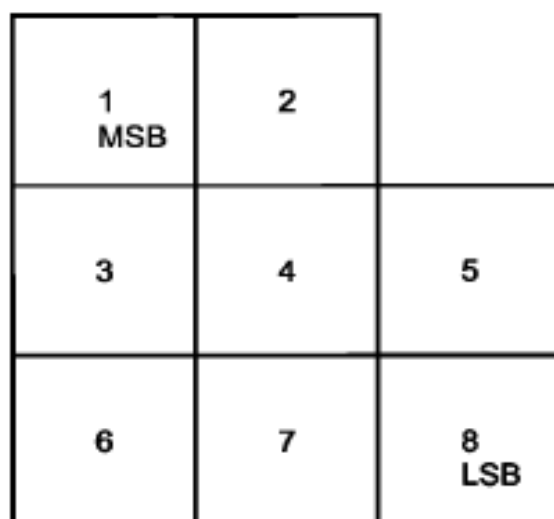


Рисунок 12 – Представление кодового слова (MSB – старший значащий разряд, LSB – младший значащий разряд)

Последовательность расположения кодовых слов определена следующим образом: весь символ разделяется на несколько областей, соответствующим представлению кодового слова. Пример такого расположения приведен на рисунке 13. После этого начиная с модуля в первом столбце и пятой строке проводится линия под углом 45 градусов. Линия последовательно пересекает MSB и LSB. В случае, если линия пересекает LSB считается, что это i -ое кодовое слово. С каждым новым LSB счетчик i увеличивается.

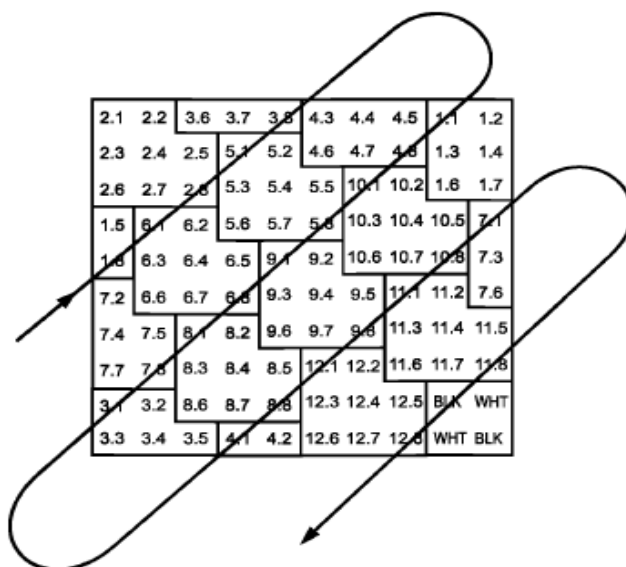


Рисунок 13 – Последовательность расположения кодовых слов

Символ штрихового кода Data Matrix имеет шаблон поиска: L образная структура, расположенная слева и снизу символа — и шаблон коррективки: чередующиеся модули на верхней и правой стороне символа. Кроме того, поскольку используемый символ Data Matrix составлен из нескольких символов Data Matrix меньшего размера, шаблоны коррективки присутствуют и внутри символа Data Matrix. Все шаблоны поиска и коррективки приведены на рисунке 14.

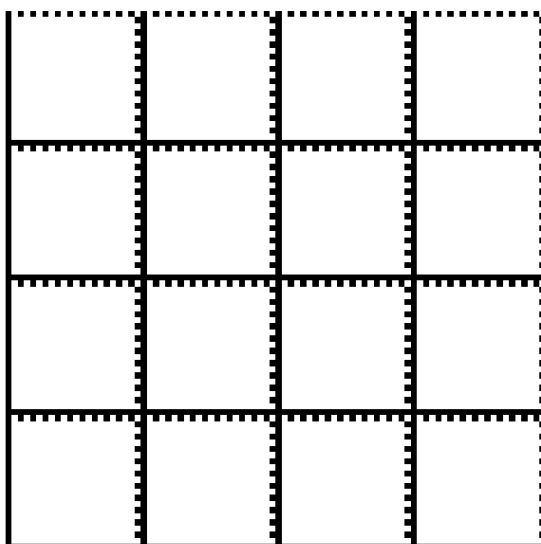


Рисунок 14 – Шаблоны поиска и коррективки анализируемого символа штрихового кода Data Matrix

Эти шаблоны не несут никакой информационной составляющей и необходимы для того, чтобы упростить процедуру распознавания символа ШК Data Matrix.

2.3 Описание алгоритма определения черной и белой области на изображении

Чтобы распознать символ Data Matrix на изображении необходим инструмент, позволяющий определить, где черная, а где белая составляющая. Для этого была разработана функция корреляции.

Предположим: существует выборка однобайтовых значений. Данная выборка соответствует некоторой области на изображении. Каждый байт является значением яркости одного из пикселей.

Существует шаблон области на изображении, которую необходимо найти. Шаблон описывает искомую область и состоит из элементов, которые принимают значения 1, что соответствует полностью белому пикселю, или -1, что соответствует полностью черному пикселю. Пример такого шаблона приведен на рисунке 15. Число 1 соответствует полностью белый пиксель (значение яркости 255, символ «Б» на рисунке 15), числу -1 соответствует полностью черный пиксель (значение яркости 0, символ «Ч» на рисунке 15).

При этом будем считать, что шаблон имеет две части: белую и точно такую же противоположную ей черную. Таким образом шаблон условно разделен на две части и соответствует месту соприкосновения двух контрастных областей изображения.

Функция корреляции принимает на вход шаблон, выборку значений и подсчитывает по формуле (2) некоторое число, которое может принимать значения от 1 до -1.

$$c = \sum S_i * x_i, \quad (2)$$

где:

- S_i - значение шаблона на i -ой позиции (знак);
- X_i - значение яркости пикселя, деленное на 255.

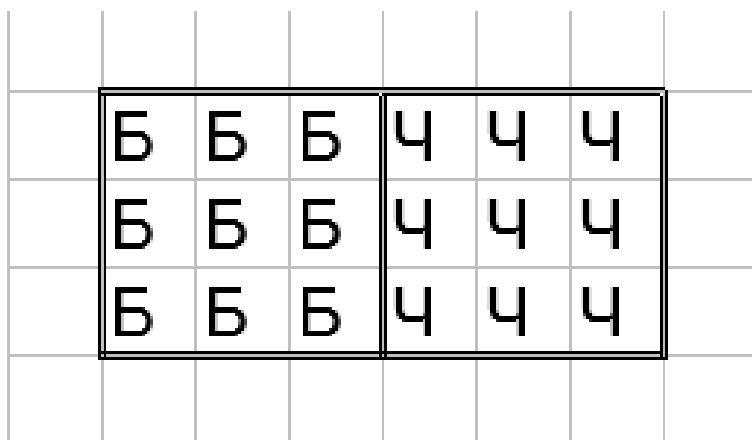


Рисунок 15 – Пример шаблона корреляции

Таким образом значение равное «1» или близкое к нему соответствует тому, что полученная выборка значений соответствует указанному шаблону, значение «-1» тому, что выборка соответствует инверсному значению шаблона, а значение «0», что выборка соответствует области изображения с одинаковой яркостью.

Пример работы функции корреляции приведен на рисунке 16. В данном случае используется шаблон, приведенный на рисунке 10. Сверху приведена область из двух противоположных друг-другу по цвету модулей. Внутри неё обозначена яркость пикселей. Значение функции корреляции для этой области равно 0,47.

Внизу приведена область из двух одинаковых по цвету модулей. Внутри неё синим цветом обозначена яркость пикселей. Значение функции корреляции для этой области равно -0,25.

Однако ввиду того, что изображение состоит не из идеальных черных и белых пикселей, значение корреляции не всегда близко к единице. В этом случае необходимо сравнивать несколько значение функции корреляции от областей расположенных таким образом, что каждая следующая смещена от

предыдущей на один столбец или строчку в сторону. В этом случае будет обнаружена критическая точка функции корреляции, локальный экстремум. Пример такой работы функции корреляции приведен на рисунке 17. Обнаружение экстремума соответствует тому, что найдена граница черной и белой области.

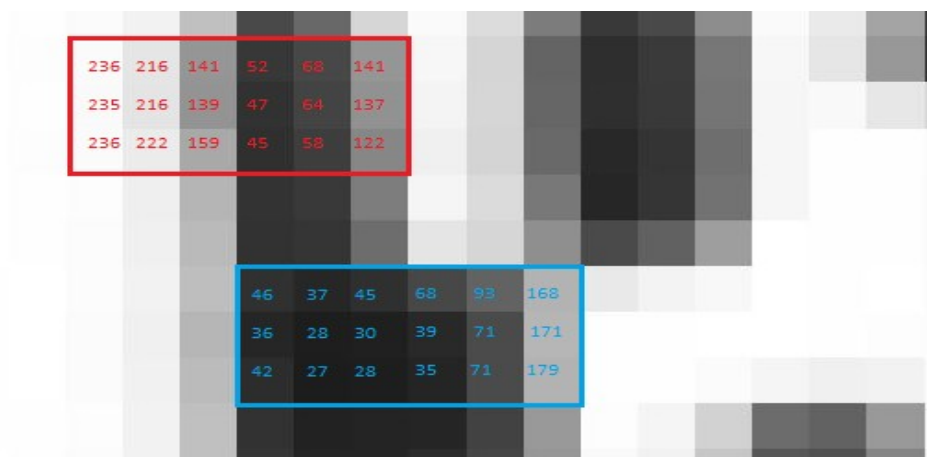


Рисунок 16 – Пример работы функции корреляции

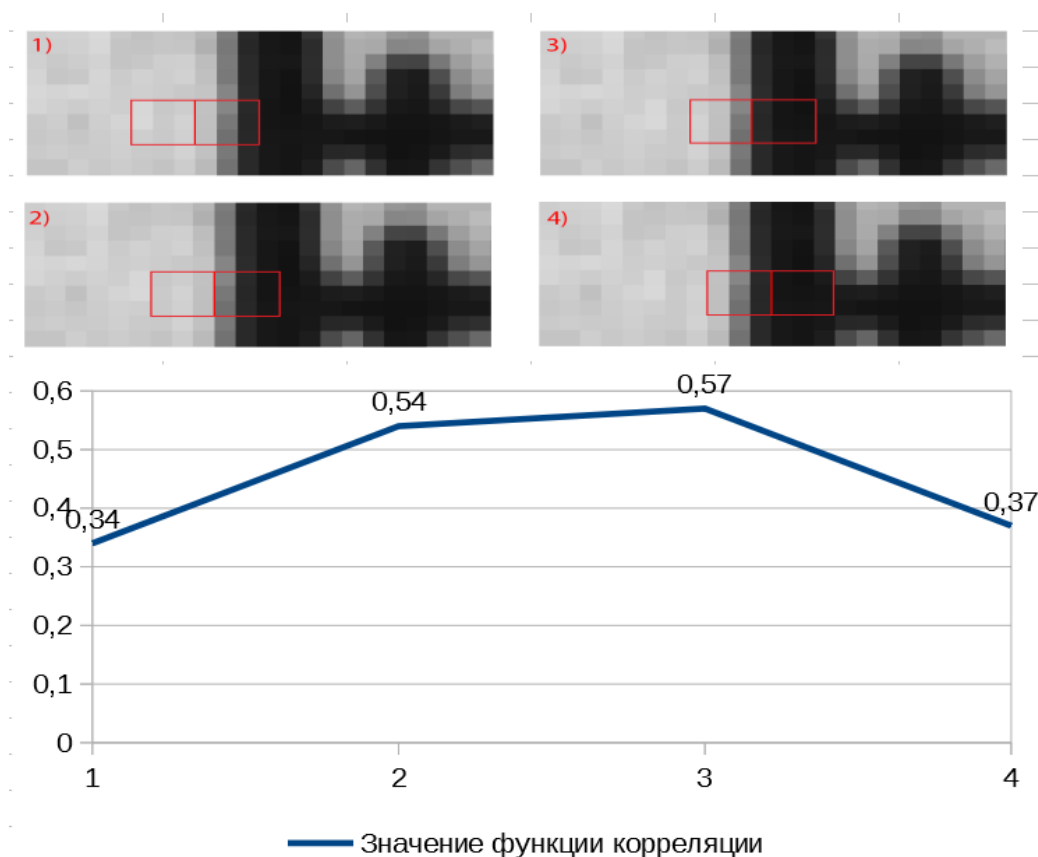


Рисунок 17 – Значения функции корреляции при последовательном смещении сканера. Экстремум корреляции

2.4 Алгоритм распознавания символа ШК Data Matrix

Распознавание начинается с поиска символа Data Matrix и определения его ориентации. Для этого осуществляется «просечка», сканер устанавливается на край одной из четырех сторон символа Data Matrix и движется, пока не встретит черную область. Для определения начала черной области ищется экстремум корреляции. Если экстремум не был найден и была осуществлена просечка до противоположного края изображения, то анализируется другой край изображения таким же образом. Просечка схематично приведена на рисунке 18.

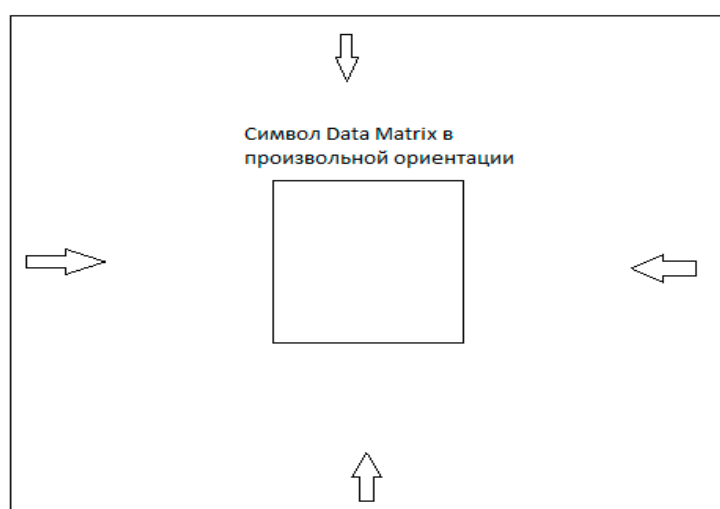


Рисунок 18 – Схематичная просечка изображения на предмет наличия символа Data Matrix

Как только была найдена черная область проверяется, является ли черная область линией шаблона поиска L символа Data Matrix – черной линии L образной формы. Делается предположение, что обнаруженная черная область является линией шаблона поиска L, после чего осуществляется попытка найти два конца линии шаблона поиска — попытка двигаться по линии. При успешной попытке найти концы линии шаблона поиска L вычисляется длина данной линии. Если длина превышает минимально допустимое значение для длины линии шаблона поиска, то считается, что найдена одна из линий шаблона

поиска. Если оказывается, что найденная линия не является линией шаблона поиска L продолжается просечка символа Data Matrix.

Если найденная линия является линией шаблона поиска L, то осуществляется поиск второй линии шаблона поиска. Для этого с обоих краев найденной линии в направлении просечки осуществляется попытка движения вдоль черной линии. После этого вычисляется длина найденных линии, она из двух длин будет маленькой, размером длиной в модуль, другая больше. На основе этого параметра вычисляется вторая линия шаблона поиска L.

После этого различные параметры распознавания устанавливаются таким образом, что символ Data Matrix независимо от реального поворота анализируется как символ положение шаблона поиска которого соответствует букве L, такое положение приведено на рисунке 19, на котором схематично изображена описанная выше процедура распознавания.

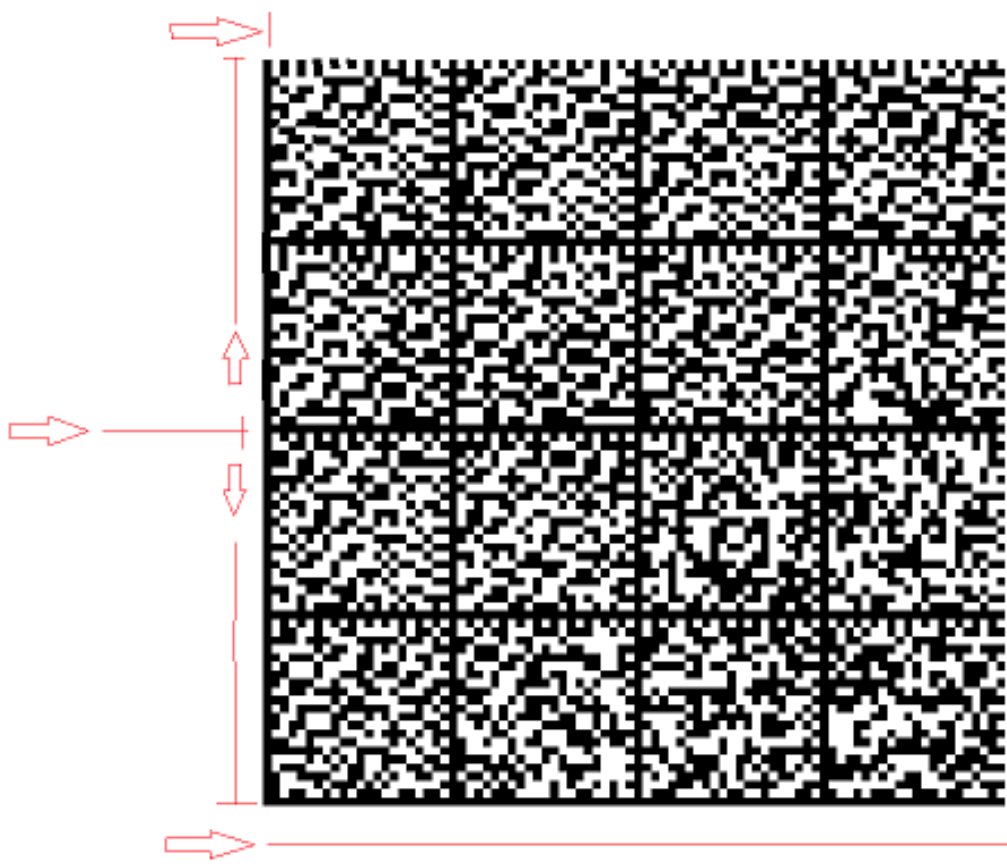


Рисунок 19 – Процедура распознавания шаблона поиска L символа Data Matrix

Движение по черной линии шаблона поиска L осуществляется следующим образом: устанавливаются два сканера на линию таким образом, как приведено на рисунке 20. После этого осуществляется смещение сканеров на один пиксель и подсчет значения корреляции. Как только будет обнаружен экстремум корреляции от сканера, установленного на черную линию, то считается, что найден конец линии. Вертикальный сканер необходим для того, чтобы подстраиваться при наклоне черной линии. На каждом шаге проверяется, будет ли значение корреляции ближе к единице при смещении вправо или влево горизонтального сканера.

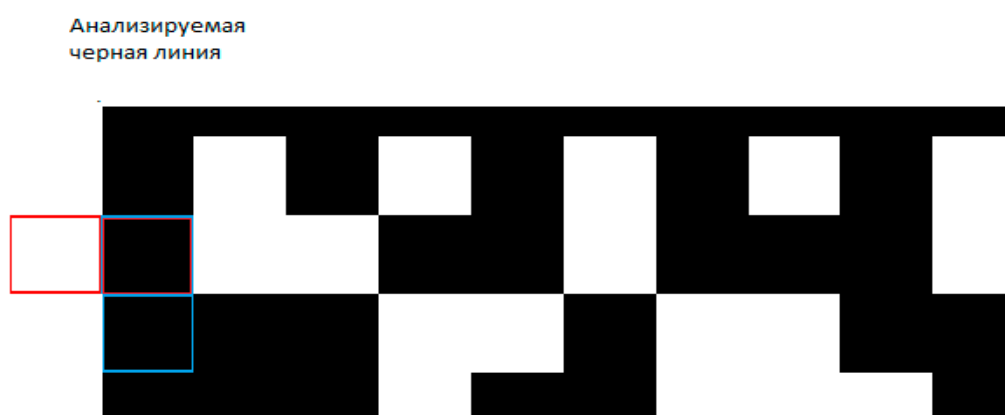


Рисунок 20 – Процесс движения по черной линии шаблона поиска L

После того, как был найден шаблон поиска L символа Data Matrix, вычисляется угол наклона символа Data Matrix. Угол наклона определяется как отношение между верхней и нижней координатой X шаблона поиска, что приведено на рисунке 21.

После того, как был определен угол наклона символа Data Matrix, осуществляется поиск шаблона корректировки — чередующихся модулей по краю изображения. При этом, на правой стороне сохраняются координаты всех модулей, а на верхней стороне сохраняются координаты начала шаблона поиска и корректировки для внутренних, составных символов Data Matrix. Процесс поиска шаблона корректировки приведен на рисунке 22.

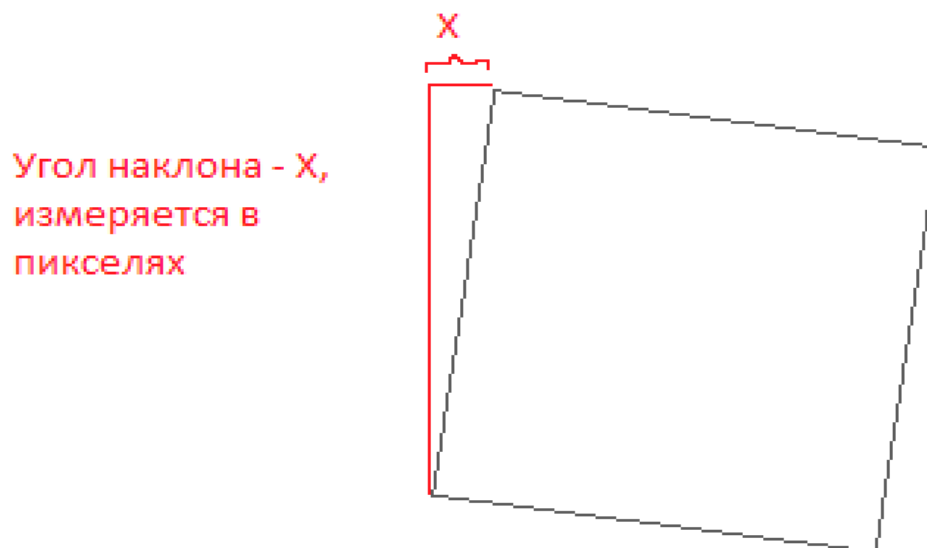


Рисунок 21 – Процесс определения угла наклона символа Data Matrix

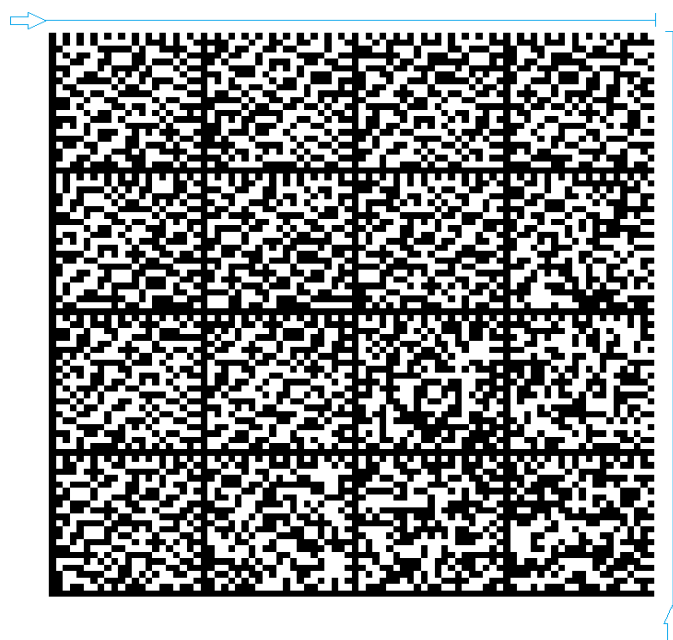


Рисунок 22 – Процесс поиска шаблона корректировки

Движение по линии шаблона корректировки осуществляется следующим образом: два сканера устанавливаются, как приведено на рисунке 23. Осуществляется движение вверх и подсчет корреляции. В случае, если был обнаружен экстремум корреляции считается, что был найден один модуль шаблона корректировки. Как только был обнаружен черный модуль шаблона корректировки, осуществляется подстройка. Проверяется будет ли значение

корреляции ближе к единице при смещении горизонтальной и вертикальной лупы вправо или влево.

После того, как были найдены внешние шаблон поиска L и шаблон корректировки, осуществляется поиск внутренних шаблонов поиска и корректировки. Процесс анализа приведен на рисунке 24.



Рисунок 23 – Процесс анализа шаблона корректировки

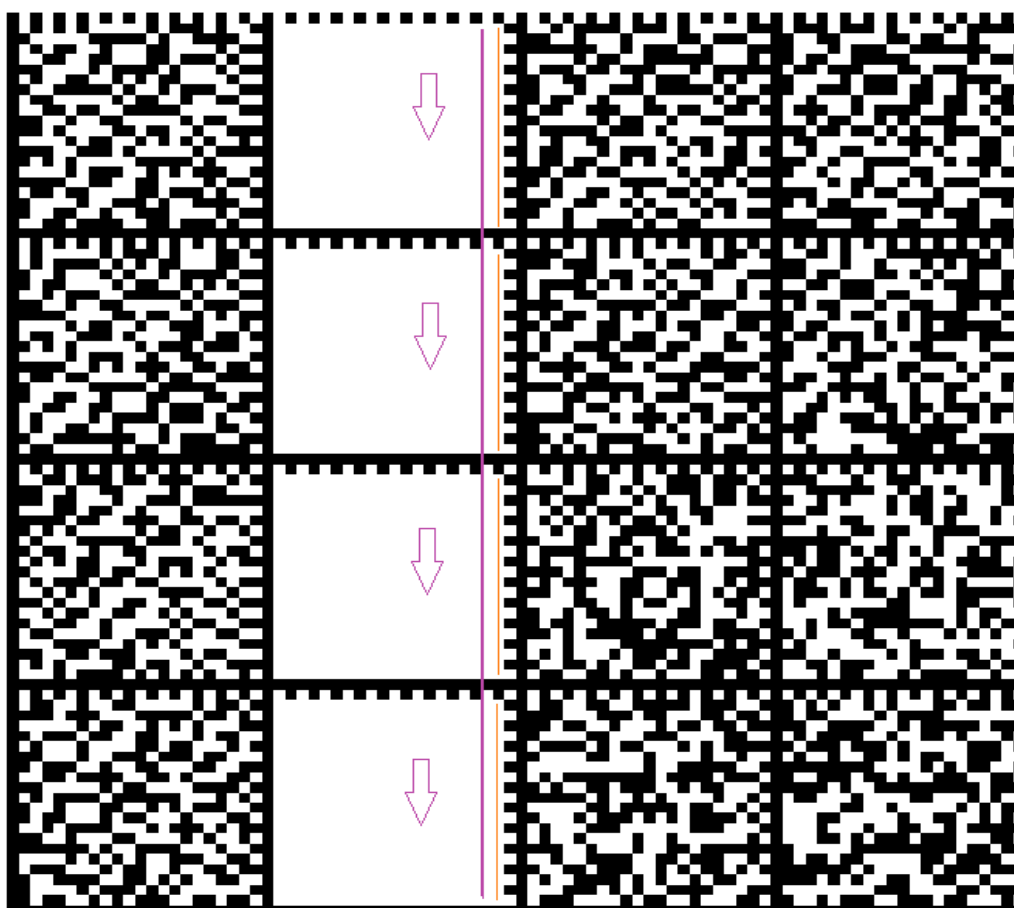


Рисунок 24 – Процесс анализа внутренних шаблонов поиска и корректировки

После того, как были найдены все внутренние шаблоны корректировки, для трех столбцов из составных символов Data Matrix уже известны начальные и конечные координаты для горизонтальной строки информационных модулей второго, третьего и четвертого столбца составных символов Data Matrix. Для того, чтобы узнать начальную координату столбца информационных модулей первого столбца составных символов Data Matrix осуществляются сканирование вертикальной линии шаблона поиска L и смещенной линии на один модуль влево из информационных модулей. Если при сканировании необходимо осуществить смещение из-за угла наклона на линии шаблона поиска L, осуществляется и смещение на линии из информационных модулей. Если при анализе линии из информационных модулей был найден экстремум корреляции, то считается, что найдена начальная координата для горизонтальной линии информационных модулей, иначе считается, что анализируется область из одинаковых по цвету модулей и осуществляется смещение на один модуль вниз. Подробнее процесс приведен на рисунке 25.

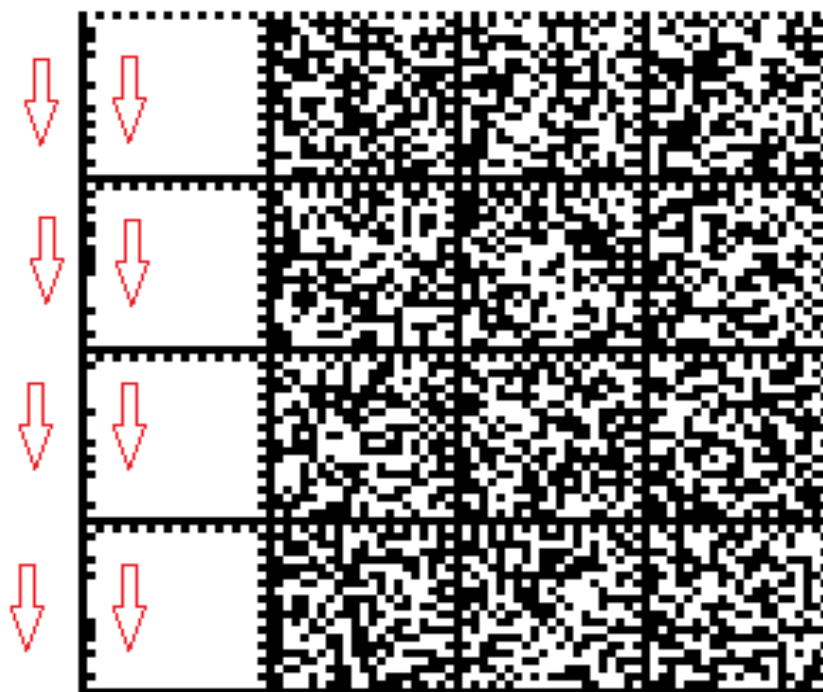


Рисунок 25 – Поиск начала горизонтальной линии информационных модулей в первом столбце составных символов Data Matrix

Как только были найдены все шаблоны осуществляется анализ информационных модулей каждого составного символа Data Matrix. При анализе информационных модулей осуществляется проход горизонтальной строки из информационных модулей и подсчет значения корреляции. В случае, если был обнаружен экстремум корреляции, считается, что был найден модуль. Если несколько модулей одного цвета расположены подряд, то движение останавливается при прохождении четырех пикселей (один запасной). В этом случае считается, что был найден модуль такого же цвета, что и предыдущий.

Для того, чтобы определить траекторию движения используется алгоритм Брезенхема. Алгоритм Брезенхема — это алгоритм, определяющий, какие точки двумерного раstra нужно закрасить, чтобы получить близкое приближение прямой линии между двумя заданными точками [10].

Алгоритм построен так, что в процессе работы одна из координат либо x , либо y (в зависимости от углового коэффициента) изменяется на единицу. Изменение другой координаты зависит от действительным положением отрезка и ближайшими координатами сетки. Пример линии построенной по такому алгоритму приведен на рисунке 26 [10].

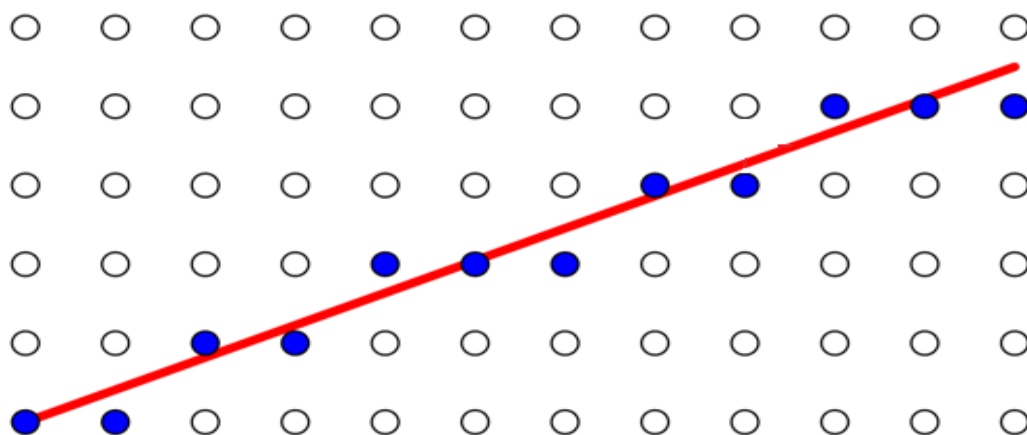


Рисунок 26 – Пример линии построенной по алгоритму Брезенхема

Для того, чтобы обеспечить движение сканера по такой линии был разработан алгоритм прыжков. Трудность заключается в том, что сканер вытянут на несколько модулей по горизонтали, из-за чего нельзя осуществить

сразу смещение на один пиксель вверх или вниз — это приведет к тому, что часть сканера будет установлена неверно.

Предположим необходимо выбрать область нестандартной формы, которая приведена на рисунке 27. Для того, чтобы выбрать пиксели, соответствующие данной области, существуют вертикальный и горизонтальный дескрипторы приращений. Использоваться может только один. Это накладывает ограничение, что смещать можно только столбец или строку. Одновременно столбец и строку нельзя.

Приращение - количество пикселей, которое будет смещено относительно левого нижнего угла для *i*-ой строки или столбца. Дескриптор приращений содержит массив с значениями приращения для каждого столбца или строки.

	-2	#	#	#	#	#	#								
	1				#	#	#	#	#	#					
	0			#	#	#	#	#	#						
	2			#	#	#	#	#	#						
	1		#	#	#	#	#	#							
	0	#	#	#	#	#	#								
	0	#	#	#					2			#	#	#	
	0	#	#	#					2			#	#	#	
	2			#	#	#			1		#	#	#		
	1		#	#	#				1		#	#	#		
	1		#	#	#				0	#	#	#			
	0	#	#	#					0	#	#	#			

Рисунок 27 – Возможные области нестандартной формы

Для того, чтобы построить линию, соответствующую линии, составленной по алгоритму Брезенхема необходимо выполнить прыжок. Прыжок представляет собой смещение дескриптора при движении направо (налево) на один пиксель вверх (вниз). Пример такого смещения приведен на

рисунке 28. Прыжок начинает свою работу, как только сканер достигает определенной координаты. Дескриптор приращений с каждым шагом изменяется таким образом, что движение сканера выглядит таким образом, как приведено на рисунке 28.

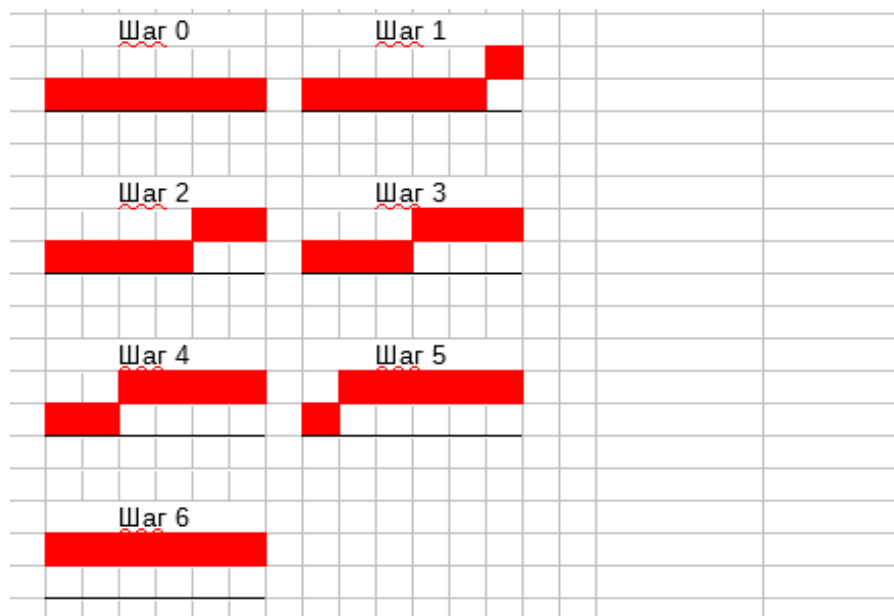


Рисунок 28 – Механизм прыжка сканера

Как только сканер полностью перепрыгивает на линию выше её на один пиксель, механизм прыжков выключается, сканер смещается на один пиксель в сторону прыжка.

При большом угле наклона часто возникает состояние, когда необходимо сделать так, чтобы работали два прыжка одновременно. Пример такого движения приведен на рисунке 29. Данный механизм предполагает использование одного дескриптора несколькими прыжками.

В этом случае при движении на каждом шаге совершается проверка на достижение координаты прыжка. В случае, если координата прыжка достигнута, количество активных прыжков увеличивается на единицу, в массиве, хранящем текущее состояние всех прыжков указывается, что прыжок с определенным индексом находится на 0-ой позиции.

Одновременно с этим, при движении, на каждом шаге, если есть активные прыжки, то для каждого активного прыжка выполняются следующие действия:

1 проверяется состояние прыжка (его позиция на дескрипторе приращений);

2 если позиция не соответствует завершению прыжка (не пройден весь дескриптор), то изменяется дескриптор приращений (следующий шаг прыжка);

3 если позиция соответствует завершению прыжка, то осуществляется выход из состояния текущего прыжка.



Рисунок 29 – Механизм многоярусных прыжков

После того, как был найден информационный модуль, осуществляется вычисление его координат в сетке модулей и изменение соответствующего кодового слова. В результате, после анализа всех информационных модулей будут составлены кодовые слова данных. При анализе информационных модулей возможна ситуация, когда распознавание модулей будет произведено неверно (ошибочно).

2.5 Алгоритм функции контроля и исправления ошибок

Кодовые слова исправления ошибок являются остатком отделения кодовых слов данных на полиномиальную функцию $g(x)$, используемую для кодов Рида-Соломона. Полиномом $g(x)$ является следующий полином:

$$\begin{aligned} \partial(x) = & x^{56} + 46 x^{55} + 143 x^{54} + 53 x^{53} + 233 x^{52} + 107 x^{51} + 203 x^{50} + 43 x^{49} + \\ & 155 x^{48} + 28 x^{47} + 247 x^{46} + 67 x^{45} + 127 x^{44} + 245 x^{43} + 137 x^{42} + 13 x^{41} + 164 x^{40} + \\ & 207 x^{39} + 62 x^{38} + 117 x^{37} + 201 x^{36} + 150 x^{35} + 22 x^{34} + 238 x^{33} + 144 x^{32} + 232 x^{31} + \\ & 29 x^{30} + 203 x^{29} + 117 x^{28} + 234 x^{27} + 218 x^{26} + 146 x^{25} + 228 x^{24} + 54 x^{23} + 132 x^{22} + \\ & 200 x^{21} + 38 x^{20} + 223 x^{19} + 36 x^{18} + 159 x^{17} + 150 x^{16} + 235 x^{15} + 215 x^{14} + 192 x^{13} + \\ & 230 x^{12} + 170 x^{11} + 175 x^{10} + 29 x^9 + 100 x^8 + 208 x^7 + 220 x^6 + 17 x^5 + 12 x^4 + 238 x^3 \\ & + 223 x^2 + 9 x + 175 [6]. \end{aligned}$$

При этом степень полинома k связана с максимальным количеством исправляемых ошибок t следующим образом: $k = 2 \cdot t$. Кодовые слова исправления ошибок позволяют исправлять два типа ошибочных кодовых слов: стирания (ошибочные кодовые слова с известными позициями) и ошибки (ошибочные кодовые слова с неизвестными позициями) [6].

В символе Data Matrix используемой версии присутствует четыре блока Рида-Соломона, в каждом из которых находятся 144 информационных слов и 56 проверочных, всего 200 слов. Количество ошибочных слов, которые можно восстановить для одного блока составляет $t = 28$ ($56 / 2 = 28$). Таким образом в лучшем случае может быть восстановлено 112 кодовых слов.

Полиномиальные арифметические вычисления для символов ШК Data Matrix проводятся с использованием побитового арифметического устройства для битовых операций по модулю 2 и арифметического устройства для байтовых операций по модулю 100101101 (десятичное значение 301). Это поле Галуа 2^8 , где 100101101 соответствует простому минимальному многочлену поля $x^8 + x^5 + x^3 + x^2 + 1$.

Для того, чтобы совершать такие арифметические операции в полях Галуа, как сложение и вычитание используется операция XOR. Таким образом функция сложения и вычитания в полях Галуа написанная с использованием языка Си будет выглядеть следующим образом [11]:

```
int gf_Sum(int a, int b) { return a^b; }
```

Для того, чтобы совершать такие арифметические операции в полях Галуа как умножение и деление, используют таблицы умножения и деления. Это связано с тем, что функция "настоящего" умножения Галуа настолько сложна и ресурсоемка, что для упрощения ее реализации приходится к временному преобразованию полиномов в индексную форму, последующему сложению индексов, выполняемому по модулю GF, и обратному преобразованию суммы индексов в полиномиальную форму. Индекс — показатель степени при основании два, дающий искомый полином. Например, индекс полинома 8 равен 3 ($2^3 = 8$), а индекс полинома 2 равен 1 ($2^1 = 2$). Легко показать, что $a * b = 2^i + 2^j = 2^{(i+j)}$. В частности, $2 * 8 = 2^3 + 2^1 = 2^{(3+1)} = 16$. В силу того, что количество полиномов всякого поля Галуа равно количеству всевозможных индексов, можно определенным образом сопоставить их друг другу. Таблица степеней примитивного полинома обычно обозначается как `alpha_of`, индексы, соответствующие данному полиному обычно обозначается как `index_of`. С помощью таких таблиц легко можно осуществлять преобразование из полиномиальной формы в индексную и наоборот, а следовательно умножение и деление. Таким образом функция умножения в полях Галуа, написанная с использованием языка Си, может выглядеть следующим образом [11]:

```
int gf_Mul(int a, int b)
{
    int sum;
    if (a == 0 || b == 0) return 0;
    sum = alpha_of[a] + alpha_of[b];
    if (sum >= GF-1) sum -= GF-1;
    return index_of[sum];
}
```

Деление в полях Галуа осуществляется точно так, как и умножение с той лишь разницей, что индексы не прибавляются, а вычитаются друг из друга: $a/b = 2^i/2^j = 2^{(i-j)}$. Для перевода из полиномиальной в индексную форму и наоборот может использоваться те же таблицы. Таким образом функция

быстрого табличного деления в полиномов в полях Галуа, написанная с использованием языка Си, может выглядеть следующим образом [11]:

```
int gf_Div(int a, int b)
{
    int diff;
    if (a == 0) return 0;
    if (b == 0) return -1;    //Проверка деления на 0
    diff = alpha_of[a] - alpha_of[b];
    if (diff < 0) diff += GF-1;
    return index_of[diff];
}
```

Типовая схема декодирования кодов Рида-Соломона, получившая название авторегрессионного спектрального метода декодирования, состоит из следующих шагов [11]:

- 4 вычисление синдрома ошибки;
- 5 построение полинома ошибки;
- 6 нахождения корней полинома ошибки;
- 7 определения характера ошибки;
- 8 исправление ошибочных символов.

Синдром это остаток деления декодируемого кодового слова $s(x)$ на порожденный полином $g(x)$, и, если этот остаток равен нулю, кодовое слово считается неискаженным. Ненулевой остаток свидетельствует о наличии по меньшей мере одной ошибки. Остаток от деления дает многочлен, независимый от исходного сообщения и определяемый исключительно характером ошибки.

Полученный синдром описывает конфигурацию ошибки, но еще не говорит, какие именно символы полученного сообщения были искажены. Степень синдромного полинома, равная $2t$, много меньше степени полинома сообщения, равной n , и между их коэффициентами нет прямого соответствия .

Полином, коэффициенты которого напрямую соответствуют коэффициентам искаженных символов, называется полиномом локатора ошибки [11].

Если количество искаженных символов не превышает t , между синдромом и локатором ошибки существует следующее однозначное соответствие, выражаемое следующей формулой: $\text{НОД}[x \cdot n - 1, E(x)] = \lambda(x)$ и вычисление локатора сводится к задаче нахождения наименьшего общего делителя, успешно решенной еще Евклидом и элементарно реализуемой как на программном, так и на аппаратном уровне. На практике обычно применяют более эффективный, но и более сложный для понимания алгоритм Берлекэмп-Месси [11].

В случае, если полином локатора ошибки нам известен, то его корни определяют местоположение искаженных символов в принятом кодовом слове. Для нахождения полинома локатора ошибки используется процедура Ченя, аналогичная по своей природе обратному преобразованию Фурье и фактически сводящаяся к перебору всех возможных вариантов. Все 2^m возможных символов один за другим подставляются в полином локатора, после чего выполняется расчет полинома. Если результат обращается в ноль, то — считается, что искомые корни найдены [11].

После того, как найдены корни полинома известны искаженные слова, но не известно, каким образом они искажены. Используя полином синдрома и корни полинома локатора, можно определить характер разрушений каждого из искаженных символов. Обычно для этой цели используется алгоритм Форни, состоящий из двух стадий: сначала путем свертки полинома синдрома полиномом локатора ошибки $\lambda(x)$ получают некоторый промежуточный полином $\Omega(x)$. Затем, на основе $\Omega(x)$ полинома, вычисляется нулевая позиция ошибки, которая в свою очередь делится на производную от $\lambda(x)$ полинома. В результате получается битовая маска, каждый из установленных битов которой соответствует искаженному биту и для восстановления кодового слова в

исходный вид, все искаженные биты должны быть проинвертированы, что осуществляется посредством логической операции XOR [11].

2.6 Алгоритм функции декодирования данных

Согласно стандарту ГОСТ Р ИСО/МЭК 16022 — 2008. Автоматическая идентификация. Кодирование штриховое. Спецификация символики Data Matrix, схему кодирования по основанию 256 используют для кодирования любых 8-битовых байтов данных. Для переключения на схему используют кодовое слово 231. После этого один или два байта указывают длину закодированных данных в байтах. Если длина (N) данных меньше 250, то используется один байт для указания длины. Иначе используется два байта, где первый из них равен $N \setminus 250 + 249$ (значение от 250 до 255), а второй равен $N \text{ MOD } 250$ (значение от 0 до 249).

При этом, для кодирования каждого байта, после кодового слова 231 используют алгоритм рандомизации. Это необходимо для того, чтобы символ Data Matrix не состоял из одних белых или черных модулей. Вычисление значения кодового слова в соответствии с данным алгоритмом осуществляется в две операции:

$$1 \quad R = ((149 * P) \text{ MOD } 255) + 1;$$

$$2 \quad CW = (V + R) \text{ MOD } 256;$$

где R – вспомогательное значение, P – индекс кодового слова, V – значение кодового слова, CW – получаемое новое кодовое слово. Для декодирования необходимо выполнить соответствующую обратную процедуру.

2.7 Алгоритм функции гарантированного уничтожения информации

Существует множество стандартов для обеспечения гарантированного уничтожения информации. Одним из них является ГОСТ 50739-95 «Средства

вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования» в соответствии с которым в средстве вычислительной техники должен быть комплекс средств защиты информации, который должен осуществлять очистку оперативной и внешней памяти [12]. Очистка должна производиться путем записи маскирующей информации в память при ее освобождении перераспределении.

В качестве метода для очистки было принято решение использовать метод описанный в руководящем документе ФСТЭК «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации», утвержденным решением председателя Государственной технической комиссии при Президенте Российской Федерации от 30 марта 1992 г., для класса защищенности 3А. Для гарантированного уничтожения информации предлагается осуществлять двукратную произвольную запись в освобождаемую область памяти, ранее использованную для хранения защищаемых данных [13].

Существуют и другие стандарты и методы для гарантированного уничтожения информации, но все они сводятся к тому, что необходима двух-трех-и кратная перезапись оперативной и внешней памяти.

В качестве данных для записи было принято решение использовать псевдослучайную последовательность, получаемую от рекуррентной линии задержки. В качестве начального заполнения для линии было принято решение использовать значение счетчика тактов процессора 1892ВМ10Я.

2.8 Алгоритм контроля целостности программного обеспечения

В соответствии с руководящем документе ФСТЭК «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации», утвержденным решением председателя Государственной

технической комиссии при Президенте Российской Федерации от 30 марта 1992 г, в качестве метода для контроля целостности было принято решение использовать алгоритм выработки имитовставки ГОСТ 28147-89 «Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования» [13].

Имитовставка — средство обеспечения имитозащиты в протоколах аутентификации сообщений с доверяющими друг другу участниками — специальный набор символов, который добавляется к сообщению и предназначен для обеспечения его целостности и аутентификации источника данных. Имитовставка обычно применяется для обеспечения целостности и защиты от подделки передаваемой информации. В данном случае, имитовставка применяется для проверки целостности прошивки и защиты от дефектов внешней памяти [14].

Имитовставка вырабатывается для $M \geq 2$ блоков открытого текста по 64 бит. Алгоритм выработки имитовставки приведен ниже и на рисунке 30.



Рисунок 30 – Алгоритм выработки имитовставки

1 Блок открытых данных записывается в регистры N_1 и N_2 , после чего подвергается преобразованию, соответствующему первым 16 циклам шифрования в режиме простой замены;

2 К полученному результату побитно по модулю 2 прибавляется следующий блок открытых данных. Последний блок при необходимости дополняется нулями. Сумма также шифруется в соответствии с пунктом 1;

3 После добавления и шифрования последнего блока из результата выбирается имитовставка длиной L бит: с бита номер $32-L$ до 32 (отсчёт начинается с 1) [14].

Было принято решение использовать имитовставку длиной 32 бита. Ключ шифрования и значение имитовставки хранятся во внешней памяти процессора 1892BM10Я.

3 Обзор архитектуры процессора 1892BM10Я и его основных характеристик

Процессор 1892BM10Я это трехядерный сигнальный процессор, архитектура процессора MIPS32, тактовая частота 250 МГц, размер встроенного ОЗУ около 4 Мбайт, внутренняя память процессора, C RAM, имеет размер 128 кбайт. Процессор имеет 32-разрядный порт внешней памяти SRAM/ ROM/ SDRAM/ Mobile SDRAM/ EPROM/ FLASH и внешних устройств [15].

Для достижения максимального быстродействия необходимо правильно разместить разработанный код в памяти процессора. Это позволит уменьшить время затрачиваемое на обращение к ячейкам памяти через порт внешней памяти.

В архитектуре MIPS32 существуют понятия «физический адрес» и «виртуальный адрес». Адрес, указанный в инструкции доступа к памяти это всегда виртуальный адрес. Он, будучи декодирован процессором из инструкции (или извлечен из регистра), преобразуется в физический адрес – для этого служит устройство управления памятью (Memory management unit – MMU). Устройство управления памятью (MMU), реализует интерфейс между устройством исполнения и контроллером кэш. После преобразования происходит определение, к какой области памяти он относится, и тогда уже данный адрес передается либо на внутреннюю шину процессора для получения значения из внутренней памяти, либо в порт внешней памяти для получения значения из памяти, подключенной к процессору снаружи [15].

MMU процессора 1892BM10Я может работать в двух режимах: FM и TLB. Режим преобразования определяется битом FM регистра CSR. По умолчанию задан режим FM.

Режим FM – это режим фиксированного отображения (Fixed mapped, FM), в нем процессор работает сразу после снятия сигнала nRST. Режим TLB – это режим TLB, который более необходим при реализации многозадачных

операционных систем. Используется режим FM. Данный режим работает по следующему алгоритму: получая виртуальный адрес, устройство управления памятью анализирует его старшие три разряда. В зависимости от их значения, определяется сегмент, к которому этот адрес относится (useg/kuseg, kseg0, kseg1, kseg2, kseg3), и физический адрес получается либо вычитанием строго заданного значения, либо отображается напрямую [15].

Карта памяти виртуальных адресов приведена на рисунке 31, используется режим, где бит ERL регистра Status CP0 установлен в единицу. Карта памяти физических адресов микросхемы приведена на рисунке 32. Карта внутренней памяти процессора 1892BM10Я приведена на рисунке 33.

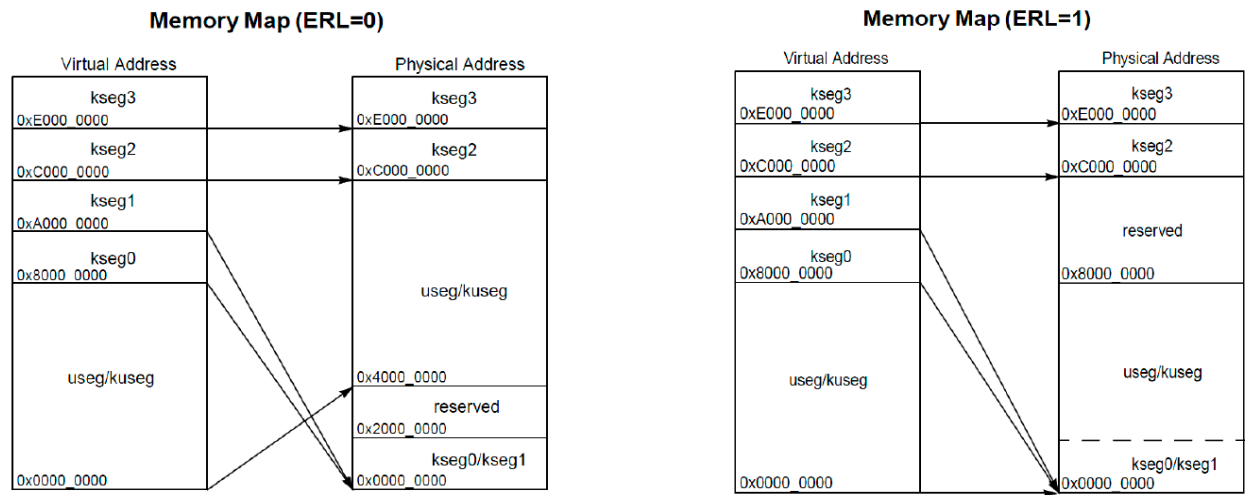


Рисунок 31 – Карта памяти процессора 1892BM10Я

Диапазон адресов	Название области	Объем области, Мбайт
FFFF_FFFC	Внешняя память	3584
2000_0000	Внешняя память (ПЗУ)	64
1FFF_FFFC	Внутренняя память	64
1C00_0000	Внутренняя память	64
1BFF_FFFC	Внутренняя память	64
1800_0000	Внутренняя память	64
17FF_FFFC	Внутренняя память	64
0000_0000	Внешняя память	384

Рисунок 32 – Карта памяти физических адресов микросхемы

Диапазон адресов	Название области	Объем области, Кбайт
1BFF_FFFF 1880_0000	Резерв	56000
187F_FFFF 1840_0000	Память и регистры DSP-ядра	4096
183F_FFFF 1830_0000	Резерв	1024
182F_FFFF 182F_0000	Регистры CPU	64
182E_FFFF 1802_0000	Резерв	3000
1801_FFFF 1800_0000	Память C RAM	128

Рисунок 33 – Карта внутренней памяти процессора 1892BM10Я

Таким образом, располагая код по виртуальному адресу 0x18000000 код будет выполняться из внутренней памяти процессора, а при расположении, к примеру по адресу 0x80000000 или 0xA0000000 код будет выполняться из внешней памяти процессора.

При выполнении из внешней памяти процессора время выполнения увеличивается из-за того, что необходимо запросить данные через MPORT (порт внешней памяти).

В случае, если необходимо разместить код во внешней памяти процессора, необходимо выбрать один из сегментов (useg/kuseg, kseg0, kseg1, kseg2, kseg3).

Виртуальные сегменты памяти, на которые делится адресное пространство, различаются в зависимости от режима работы процессора. Ядро входит в режим Kernel после аппаратного сброса или когда происходит исключение. В режиме Kernel программное обеспечение имеет доступ к полному адресному пространству и ко всем регистрам CP0. В режиме User доступ ограничен подмножеством виртуального адресного пространства

(0x0000_0000 - 0x7FFF_FFFF) и запрещен доступ к функциям CP0. В режиме User недоступны виртуальные адреса 0x8000_0000 - 0xFFFF_FFFF и обращение к ним вызывает исключение.

Каждый из сегментов, показанных на рисунке 31, является либо отображаемым (mapped), либо неотображаемым (unmapped). В неотображаемом сегменте механизмы TLB или FM для преобразования виртуального адреса в физический адрес не используются.

Особенно важно иметь неотображаемые сегменты памяти после аппаратного сброса, потому что TLB еще не запрограммировано и не может осуществлять преобразования.

Для неотображаемых сегментов преобразование виртуального адреса в физический является фиксированным.

Все неотображаемые сегменты, за исключением kseg0, никогда не кэшируемы. Кэшируемость kseg0 определяется полем K0 регистра Config CP0. Для режима FM отображаемые сегменты имеют закрепленное преобразование виртуального адреса в физический. Кэшируемость сегмента определяется значениями полей K23 и KU регистра Config CP0. При FM-преобразовании невозможна защита сегментов от записи. В таблице 5 приведены сегменты и их характеристика.

Таблица 5 – Сегменты памяти процессора 1892BM10Я и их характеристика

Сегмент	Диапазон	Размер (МБ)	Характеристика
kuseg	0x00000000 0x7FFFFFFF	2048	Отображаемый. Доступен в режиме user и режиме kernel
kseg0	0x9FFFFFFF 0x80000000	512	Неотображаемый. Доступен в режиме kernel
kseg1	0xA0000000 0xBFFFFFFF	512	Неотображаемый. Доступен в режиме kernel
kseg2	0xC0000000 0xDFFFFFFF	512	Отображаемый. Доступен в режиме kernel
kseg3	0xE0000000 0xFFFFFFF	512	Отображаемый. Доступен в режиме kernel

Таким образом наиболее предпочтительными для размещения в них кода при доступе к внешней памяти являются сегменты kseg2 и kseg3 поскольку поддерживают возможность кеширования, что незначительно увеличивает скорость выполнения кода.

4 Определение параметров быстродействия, объема используемой оперативной памяти и объема полученного кода разработанного программного обеспечения для процессора 1892BM10Я

4.1 Определение быстродействия разработанного программного обеспечения

Было осуществлено тестирование программного обеспечения. При выполнении программного обеспечения из внешней памяти требуется приблизительно 108000000 тактов, что составляет 0,45 секунды. При выполнении программного обеспечения из внутренней памяти, SRAM, требуется приблизительно 91000000 тактов, что составляет 0,37 секунды.

Поскольку в внутренней памяти процессора будут располагаться и другие данные, не только выполняемый код для распознавания и декодирования символа Data Matrix, то не получится расположить весь код во внутренней памяти процессора, а только наиболее часто выполняемые функции.

4.2 Определение требуемых ресурсов памяти разработанного программного обеспечения

Размер прошивки, содержащей разработанное программное обеспечение, и все остальные модули общего программного обеспечения составляет 403 кБайт. При этом размер программного обеспечения распознавания символа ШК Data Matrix, исправления ошибок и декодирования данных составляет 104 кБайт памяти. По результатам анализа слепка памяти программы были определены размеры секций, которые приведены в таблице 6.

Можно сделать вывод, что требуемый размер памяти составляет 104 КБ. Размер кучи не имеет значения, поскольку не используется динамическое выделение памяти. Требуемый размер стека составляет 8 КБ, что было определено путем установки размера стека равного 1 КБ и последующего

изменения размера стека после неуспешной попытки выполнения программного обеспечения.

Таблица 6 – Слесток памяти программы

Секция	Размер секции (КБ)
.text	65
.data	37
.sdata	0
.rodata	0
.sbss	0
.bss	2

5 Выявление пороговых значений искажений изображения для разработанного ПО

5.1 Тестирование программного обеспечения на изображениях без внесения дополнительных искажений

Для тестирования было получено 260 изображений верно распознаны все изображения. Среднее количество ошибок (неправильно распознанных модулей) при анализе изображения составляет 9,5 модулей. Количество неправильно распознанных модулей не всегда соответствует количеству кодовых слов данных с ошибками ввиду того, что одно слово состоит из 8-и разрядов, кодируемых 8-ю модулями.

Было обнаружено, что качество распознавания (количество неправильно распознанных модулей) зависит не только от качества изображения, но и от угла поворота. Это связано с тем, что при фотографировании, черные модули неточно ложатся на матрицу фотографирующего элемента, из-за чего на разные ячейки приходится разное количество энергии, вследствие, получается большое количество однотонные, серых пикселей. При повороте, этот эффект усиливается.

5.2 Обзор возможных искажений изображения

На изображении возможен ряд искажений, которые связаны с особенностями фотографирующего элемента. Возможные искажения приведены на рисунке 34.

Первое искажение — поворот изображения. Данное искажение возникает из-за того, что носитель информации неровно накладывается на матрицу фотографирующего элемента.

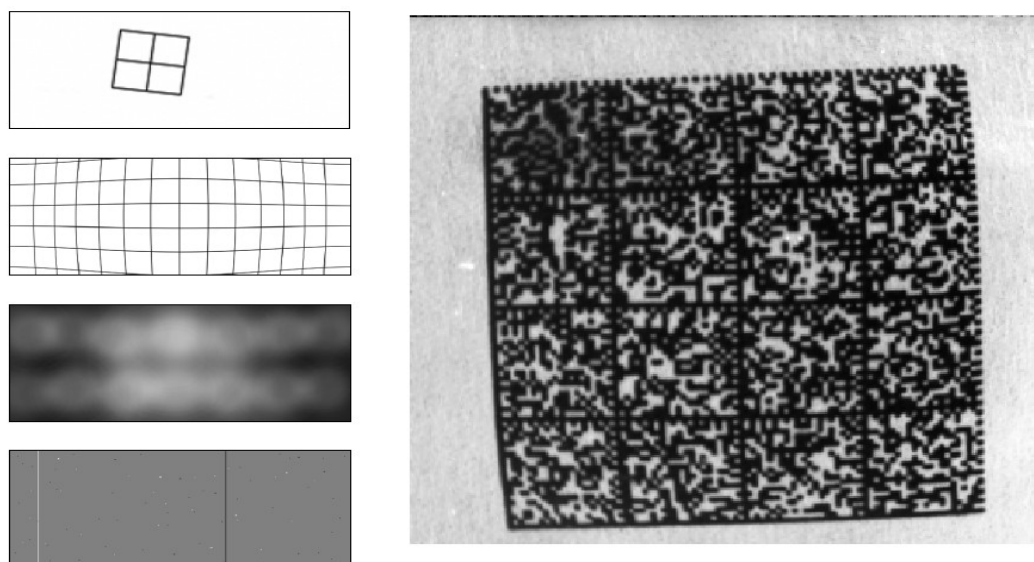


Рисунок 34 – Возможные искажения изображения

Второе искажение — изменение размеров модулей на сетке модулей. В этом случае, некоторые модули становятся больше. Это связано с тем, что расстояние до разных точек носителя от фотографирующего элемента разное.

Третье искажение — неравномерность освещенности. Оно связано с тем, что при фотографировании в качестве вспышки используется несколько диодов. Время, когда диоды зажигаются, горят, и выключаются различное, из-за чего разное количество энергии попадает на ячейки матрицы фотографирующего элемента.

Четвертое искажение — появление шумовых полос, точек. Оно связано с тем, что некоторые ячейки матрицы могут выходить из строя или считываться неправильно.

Данные искажения, могут проявляться отдельно друг от друга, но чаще проявляются все вместе в той или иной мере.

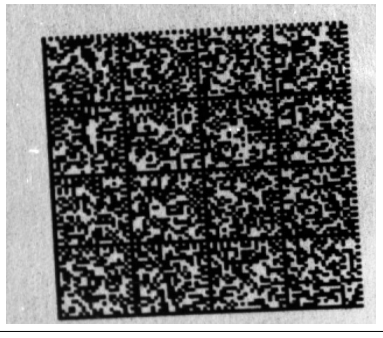
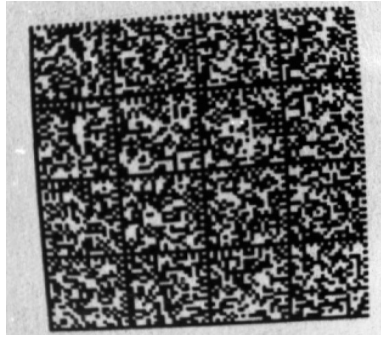
5.3 Тестирование программного обеспечения на изображениях с дополнительными искажениями

Внесение искажений в уже имеющиеся фотографии осуществлялось с помощью программы adobe Photoshop CS6 версии 13.0.

Было осуществлено искусственное затемнение и осветление изображения. Для осуществления данной операции на изображение накладывался фильтр коррекции «Уровни» с шагом 0,25, при положительном увеличении, изображение становилось светлее, при отрицательном, темнее. Некоторые результаты анализа таких изображений приведены в таблице 7. Примеры изображений приведены в таблице 7.

Была осуществлена искусственная деформация изображения. Для осуществления данной операции изображение подвергалось операции «Трансформирование», «Деформация». Некоторые результаты анализа таких изображений приведены в таблице 7. Примеры изображений приведены в таблице 7.

Таблица 7 – Результаты моделирования искажений

№	Коррекция	Результат	Изображение
1	Изменение яркости. +4	Верно	
2	Изменение яркости. -0,2	Верно	
3	Деформация	Верно	

Изменение угла наклона не производилось ввиду того, что до этого были получены изображения с различными углами наклона. Изменение шума не проводилось, ввиду того, что изображение получается достаточно шумным, в нем присутствуют «выбитые» пиксели.

Было обнаружено, что успех распознавания зависит от того, насколько большой контраст между соседними модулями. Так осветление изображения не влияет сильно на качество распознавания. При затемнении белые модули, которые окружены черными становятся очень темными и их распознавание затруднено из-за чего возникает большое количество ошибок.

Было обнаружено, что деформация не сильно влияет на качество распознавания ввиду того, что символ Data Matrix имеет в своем составе большое количество шаблонов поиска и корректировки, которые позволяют синхронизировать положение сканера на символе.

6 Обзор вредных психофизиологических факторов, влияющих на сердечно-сосудистую систему специалиста по защите информации. Методы и средств физической культуры, снижающие их воздействие

Сердечно-сосудистая система — система органов, обеспечивающая циркуляцию крови в организме человека и животных. Благодаря её деятельности кислород и питательные вещества доставляются к органам и тканям тела, а углекислый газ, другие продукты метаболизма и отходы жизнедеятельности выводятся из организма [16].

Циркуляция крови дополняется лимфооттоком от органов и тканей организма по системе лимфатических сосудов, узлов и протоков лимфатической системы, впадающих в венозную систему в месте слияния подключичных вен [16].

В состав сердечно-сосудистой системы входит сердце — мышечный орган, заставляющий кровь двигаться, ритмически нагнетая её в кровеносные сосуды — полые трубки различного диаметра, по которым происходит циркуляция крови [16].

Сердечно-сосудистые заболевания (ССЗ) — общее название заболеваний сердца и кровеносных сосудов. В настоящее время именно эти заболевания являются основной причиной смерти и инвалидизации населения во всем мире, в том числе и в России. С годами эта неблагоприятная ситуация прогрессирует еще больше: болезни сердца поражают не только стариков, но и молодое население [17].

В ССЗ входят следующие заболевания:

- гипертония (высокое кровяное давление);
- ишемическая болезнь сердца (инфаркт);
- нарушения мозгового кровообращения (инсульт);
- заболевания периферических сосудов;

- сердечная недостаточность;
- ревматические заболевания сердца;
- врожденные пороки сердца;
- кардиомиопатии [17].

Состояние сердечно-сосудистой системы зависит не только от наследственности, образа жизни, продолжительности и интенсивности рабочего дня, но и от уровня требований начальства. Большую роль играет профессия человека [18].

Специалисты в области информационной безопасности находятся в группе риска таких заболеваний. Причина в том, что темп офисной жизни не позволяет расслабляться надолго, и люди, которые еще вчера нежились на солнце у моря, попадают сразу в напряженный трудовой график, что не может не вызывать стресс. Постоянные проверки, ужесточение требований законодательства в области информационной безопасности увеличивают и без того высокую нагрузку, которая ложится на работников, что зачастую превышает их физические и психические возможности [18].

Такое положение дел и нехватка квалифицированных специалистов в области информационной безопасности приводит к тому, что традиционные профессиональные требования для сотрудника повышаются почти вдвое, что может стать причиной сердечно-сосудистых заболеваний и даже внезапной смерти в трудоспособном возрасте. Напряженная психологическая обстановка и длительные стрессы на работе являются факторами риска развития ишемической болезни сердца и других коронарных патологий. Такой же эффект оказывают неудовлетворенность результатами труда или его оплатой [18].

Плохо сказываются на сердце и сосудах и ненормированный трудовой день, постоянные переработки. Если человек «горит» на работе, он может и «сгореть» [18].

По мнению врачей, самыми неблагоприятными профессиями с точки зрения кардиологии являются те, в которых сочетаются малая физическая активность, необходимость принимать важные решения в ситуации дефицита времени (или других сложных условиях) и повышенная ответственность за принимаемые решения, а такая деятельность напрямую связана с деятельностью специалиста в области информационной безопасности [18].

Чтобы минимизировать вред такой ответственной работы для сердца, надо правильно отдыхать. Как правило такой отдых подразумевает пешие или велосипедные прогулки на свежем воздухе, лечебная физкультура, занятие теннисом, волейболом. Основная цель такого занятия — расслабление и отдых [18].

Тем, чья работа связана с повышенными нервными нагрузками, необходим здоровый, восьмичасовой сон. При бессоннице, не рекомендуется употребление снотворного и транквилизаторов – это усугубляет ситуацию и дает повышенную нагрузку на печень и другие органы. Хороший снотворный эффект дают две чайные ложки меда перед отходом ко сну. В крайнем случае, можно успокаивать нервы с помощью настоек и отваров пустырника и валерианы [18].

Одной из важнейших мер по защите сердечно-сосудистой системы и её сохранению является профилактика. Данная мера позволяет предупредить сердечно-сосудистые заболевания даже при генетической склонности к ним. В меры профилактики входит следующее:

- 1 отказ от курения, а также ограничение употребления алкоголя [19];
- 2 контроль артериального давления. При наличии гипертензии необходимо завести специальный дневник, в который нужно регулярно записывать показания. Уровень давления необходимо контролировать даже при его однократном повышении [19];

3 физическая активность, соответствующая рекомендациям врача и уровню подготовки пациента. Данная мера позволит тренировать сердце, как и любую другую мышцу организма [19];

4 правильное питание. Отказ от фаст-фуда, включение в рацион блюд с полиненасыщенными жирными кислотами, рыбы, орехов, овощей, фруктов. Необходимо соблюдать разнообразный рацион, употребление одних и тех же продуктов наносит вред организму. Рекомендуется ограничить потребление поваренной соли, сахара [19];

5 минимизация стресса. Если исключить или снизить стрессовые ситуации невозможно, необходимо чаще отдыхать и расслабляться. Рекомендуются парные занятия йогой [19];

6 контроль уровня холестерина и сахара в крови. Для этого рекомендуется регулярно сдавать анализы. Данная мера позволит выявить на ранней стадии и другие заболевания, что несомненно является плюсом [19];

7 диспансеризация для выявления заболеваний сердца. В соответствии со статьей 46 Федерального закона от 21 ноября 2011 г. № 323-ФЗ «Об основах охраны здоровья граждан в Российской Федерации» [20] и Приказ Министерства здравоохранения РФ от 26 октября 2017 г. № 869н «Об утверждении порядка проведения диспансеризации определенных групп взрослого населения» [21] бесплатную диспансеризацию могут пройти все желающие, застрахованные в системе ОМС, каждые три года, начиная с возраста 21 год. Диспансеризация проводится в отношении граждан, которым в текущем году исполняется или исполнилось следующее количество лет: 21, 24, 27 и так далее. В случае если гражданин не попадаете под диспансеризацию по возрасту, то он может пройти профилактический осмотр, который проводится раз в два года.

Профилактические мероприятия позволяют выявить предрасположенность к болезням сердечно-сосудистой системы. По

завершении профилактического осмотра или диспансеризации врач обязательно даст дальнейшие рекомендации и при необходимости назначит лечение» [19].

Чтобы определить относится ли сотрудник к группе риска можно ответить на ряд вопросов, приведенных ниже. В случае, если сотрудник отвечает на любые три вопроса «Да», то он относится к группе риска сердечно сосудистых заболеваний. При любых пяти да рекомендуется обратиться в ближайшее время к врачу для обследования [19].

1 Ваш возраст: 40 лет и старше (мужчины), 50 лет и старше (женщины)?

2 Есть ли у Ваших родственников следующие заболевания (или одно из них): артериальная гипертензия, ранний атеросклероз, стенокардия, инфаркт миокарда, инсульты, сахарный диабет?

3 Вы курите?

4 Вы не придерживаетесь правильного питания?

5 У Вас имеется избыточный вес (окружность талии у женщин более 88 см, у мужчин более 92 см)?

6 Вы не часто занимаетесь физкультурой?

7 У Вас были случаи повышенного артериального давления (выше 130/80)?

8 У Вас повышен уровень холестерина в крови (выше 5,0 ммоль/л)?

9 У Вас повышен уровень сахара в крови (выше 5,6 ммоль/л) или есть диагноз сахарный диабет?

10 Вы испытываете стрессы на работе, в быту? [19]

Для того, чтобы снизить негативное воздействие на сердечно сосудистую систему во время рабочего дня, рекомендуется использовать дыхательную гимнастику. Дыхание и сокращения сердечной мышцы представляют собой единый механизм для поддержания жизнедеятельности организма. Поэтому при помощи изменения длительности вдохов и выдохов, а также пауз между ними можно тренировать сердце. Для дыхательной гимнастики не нужна

выносливость, затраты финансов и времени, она доступна всем и требует только систематичности выполнения [22].

Для таких занятий не требуется специальный инвентарь или особое помещение, но и у этого вида тренировок есть правила:

- комната должна быть хорошо проветрена, лучше заниматься у открытого окна;
- нужно избегать сквозняка;
- одежда выбирается просторная, натуральная, без тугих поясов и воротников;
- чаще всего занимаются в положении сидя на стуле или на полу со скрещенными ногами;
- голова не запрокидывается, спина прямая;
- все тело обязательно расслабляется.

Способ восстановления сердечного ритма при помощи дыхания основан на том, что размеренный ритм вдохов и выдохов, который человек может контролировать, нормализует ритмичность сокращений сердечной мышцы. Поэтому любой вариант с подсчетом длительности дыхательных движений помогает при аритмии. Для этого можно на 2 — 3 счета делать вдох и на 4 — 6 — выдох. При этом нужно находиться в положении сидя с ровной спиной. Постепенно рекомендуется растягивать вдохи и выдохи, ориентируясь на свои ощущения.

После освоения этого упражнения для профилактики приступов рекомендуется такой вид легочной гимнастики:

- 1 левая рука за спиной;
- 2 правая кисть находится вблизи лица;
- 3 большим пальцем правой руки закрыть правую ноздрю;
- 4 сделать вдох через левый носовой вход;

5 закрыть безымянным пальцем левую и выдохнуть через правую ноздрю.

После 8 — 10 дыхательных циклов руки нужно поменять. Продолжительность на первом занятии не должна превышать 2 — 3 минут, затем можно ее медленно увеличивать. Длительность вдохов и выдохов определяется комфортными ощущениями.

Для нормализации сосудистого тонуса и повышения работоспособности проводится такое упражнение: после обычного (нефорсированного) вдоха нужно сделать резкий и короткий выдох, втягивая при этом живот на выдохе. После этого следует пассивный вдох и резкий выдох. Делают 10 циклов правой ноздрей при закрытой левой, меняют сторону, также проводят 10 вдохов-выдохов, затем обеими ноздрями 15 раз.

ЗАКЛЮЧЕНИЕ

В результате дипломного проектирования было разработано программное обеспечение распознавания и декодирования символа штрихового кода Data Matrix в устройстве ввода ключевой информации.

В первом разделе выпускной квалификационной работы выполняется обзор различных технологий штрихового кодирования, осуществляется выбор оптимальной технологии для использования в качестве носителя ключевой информации, создается чертеж бумажного носителя ключевой информации и определяется информационная емкость такого носителя.

Во втором разделе осуществляется разработка архитектуры проектируемого программного обеспечения, разработка алгоритмов распознавания символа штрихового кода Data Matrix, обзор алгоритма контроля и коррекции ошибок, алгоритма гарантированного уничтожения информации и алгоритма подсчета целостности программного обеспечения.

В третьем разделе был осуществлен обзор основных характеристик процессора 1892BM10Я, который используется для запуска программного обеспечения.

В четвертом разделе было осуществлено определение скоростных характеристик разработанного программного обеспечения. Скоростные характеристики соответствуют заявленным требованиям в техническом задании на дипломное проектирование. Были определены требуемые ресурсы памяти: размер кода, размер требуемой оперативной памяти, размер стека и размер кучи. Требуемые ресурсы памяти соответствуют заявленным требованиям в техническом задании на дипломное проектирование.

В пятом разделе были выявлены пороговые искажения изображений для разработанного программного обеспечения. Были определены наиболее критичные искажения.

В шестом разделе был осуществлен обзор различных психофизиологических факторов, влияющих на сердечно-сосудистую систему специалиста по защите информации. Был осуществлен обзор методов и средств физической культуры, снижающих их воздействие.

Таким образом, все пункты технического задания на выпускную квалификационную работу выполнены полностью, в соответствии с техническим заданием.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Штриховой код — [электронный ресурс]
https://ru.wikipedia.org/wiki/Штриховой_код - режим доступа: свободный. - Дата обращения: 05.03.2019;
- 2 Сравнение_характеристик_штрихкодов — [электронный ресурс]
https://ru.wikipedia.org/wiki/Сравнение_характеристик_штрихкодов — режим доступа: свободный. Дата обращения: 05.03.2019;
- 3 ГОСТ Р ИСО/МЭК 51294.9 — 2002 Автоматическая идентификация. Кодирование штриховое. Спецификация символики PDF417;
- 4 ГОСТ Р ИСО/МЭК 24778 — 2010 Информационные технологии. Технологии автоматической идентификации и сбора данных. Спецификация символики штрихового кода Aztec Code;
- 5 ГОСТ Р ИСО/МЭК 18004 — 2015 Информационные технологии. Технологии автоматической идентификации и сбора данных. Спецификация символики штрихового кода QR Code;
- 6 ГОСТ Р ИСО/МЭК 16022 — 2008 Автоматическая идентификация. Кодирование штриховое. Спецификация символики Data Matrix;
- 7 libdmtx - [электронный ресурс] <https://github.com/dmtx/libdmtx/blob/master/> - режим доступа: свободный. Дата обращения: 05.03.2019;
- 8 zxing open source library - [электронный ресурс]
<https://github.com/zxing/zxing/blob/master/> - режим доступа: свободный. Дата обращения: 05.03.2019;
- 9 IEEE 1471 - «Рекомендуемые методы описания архитектуры программных систем»;
- 10 Алгоритм Брезенхэма — [электронный ресурс]
https://ru.wikipedia.org/wiki/Алгоритм_Брезенхэма — режим доступа: свободный. Дата обращения: 05.06.2019;

11 Крис Касперски «Техника защиты компакт-дисков от копирования» Санкт-Петербург, БХВ-Петербург 2004.

12 ГОСТ 50739-95 «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Общие технические требования»;

13 ФСТЭК «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации», утвержденным решением председателя Государственной технической комиссии при Президенте Российской Федерации от 30 марта 1992 г;

14 ГОСТ 28147-89 «Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования»;

15 Микросхема интегральная 1892ВМ10Я. Руководство пользователя. РАЯЖ.431282.012Д17. АО НПЦ Элвис.

16 Сердечно-сосудистая система — [электронный ресурс] https://ru.wikipedia.org/wiki/Сердечно-сосудистая_система - режим доступа: свободный. Дата обращения: 05.03.2019;

17 Сердечно-сосудистые заболевания - [электронный ресурс] https://ru.wikipedia.org/wiki/Сердечно-сосудистые_заболевания - режим доступа: свободный. Дата обращения: 05.03.2019;

18 Советы трудоголикам: какие профессии вредны для сердечно-сосудистой системы - [электронный ресурс] <https://www.liveinternet.ru/users/2462573/post335953892/> - режим доступа: свободный. Дата обращения: 05.03.2019;

19 Эксперт «СОГАЗ-Мед»: предотвратить сердечно-сосудистые заболевания можно [электронный ресурс] <https://sakhalinmedia.ru/news/748901/> - режим доступа: свободный. Дата обращения: 05.03.2019;

20 Приказ Министерства здравоохранения РФ от 26 октября 2017 г. № 869н «Об утверждении порядка проведения диспансеризации определенных групп взрослого населения»;

21 Федеральный закон от 21 ноября 2011 г. № 323-ФЗ «Об основах охраны здоровья граждан в Российской Федерации».

22 Правильная и полезная дыхательная гимнастика для сердца - [электронный ресурс] <http://cardiobook.ru/dyhatelnaya-gimnastika-dlya-serdca/> - режим доступа: свободный. Дата обращения: 05.03.2019;