Реферат

В данной работе представлена реализация графомоторных навыков на роботе на примере написания цифр.

Цель работы: распознать задание для игры, решить его и вывести с помощью канцелярских принадлежностей на бумагу.

В дипломной работе приводится алгоритм распознавания цифр. Выделяются особенности существующих алгоритмов, их достоинства и недостатки. Приводится сравнение существующих алгоритмов и их комбинация.

Был предложен алгоритм нанесения роботом цифр на лист бумаги. Также был создан программно-аппаратный комплекс, позволяющий применить разработанный метод.

В работе было проведено исследование работы предложенного метода.

Содержание

Обозначения и сокращения

API — Интерфейс программирования приложений (англ. Application Programming Interface)

ПО — Программное обеспечение

ОС — Операционная система

Android — Операционная система для мобильных устройств

NXT — LEGO MINDSTORMS NXT 2 набор сопрягаемых деталей и электронных блоков для создания программируемого робота

Bluetooth — Производственная спецификация беспроводных персональных сетей

OCR — Оптическое распознавание символов (англ. optical character recognition)

Введение

В настоящее время роботы вошли в жизнь человека в разных областях, но до сих пор нет четкого разделения среди робототехнических устройств, а также единой программной платформы. Разные производители делают разные и абсолютно несовместимые аппаратные средства. Технологии и инструменты затачиваются каждый раз под конкретные проблемы, их практически невозможно повторно использовать. Основной камень преткновения при создании роботов — искусственный интеллект.

Какие проблемы стоят перед современной робототехникой?

— надёжная и доступная механика;
— емкие и компактные элементы питания;
— мощная и малопотребляющая вычислительная система;
— точные и доступные датчики;
— интеллектуальная система управления.

В рамках данной дипломной работы мы попытаемся решить задачу, которая лежит на стыке 3ех проблем: механики, вычислительной системы и системы управления. Мы сформируем у робота мелкую моторику, которая будет заключаться в том, что мы научим робота писать простые символы. Другими словами, сформируем основы графомоторных навыков у робота на примере решения игры в Судоку.

Целью дипломной работы является создание такого программного комплекса. В его задачи должны входить:

а) сфотографировать изображение;
б) распознать изображение;
в) сформировать решение,
г) вывод решения.

После решения всех этих задач, мы сможем получить программно-аппаратный комплекс для демонстрации.

1 Аналитический раздел

В данном разделе производится анализ процессов распознавания изображения, передачи распознанного изображения, приема изображения и решения решение его на роботе. Производится анализ подсистем, входящих в реализуемы программно-аппаратный комплекс, формируются требования к создаваемой системе, выделяются функции её подсистем и описывается взаимодействие между ними.

1.1 Общее описание системы

В рамках данной дипломной работы разрабатывается автономный программно-аппаратный комплекс предоставляющий полный цикл для передачи и распознавания информации, для функционирования которой необходимо спроектировать все подсистемы программного комплекса и способы их взаимодействия.

Для достижения поставленной цели необходимо спроектировать и разработать следующие компоненты системы:

а) разработать систему распознавания изображения;
б) разработать протокол обмена между роботом и мобильным устройством;
в) разработать систему решения Судоку;
г) разработать систему отрисовки на бумаге полученного решения.

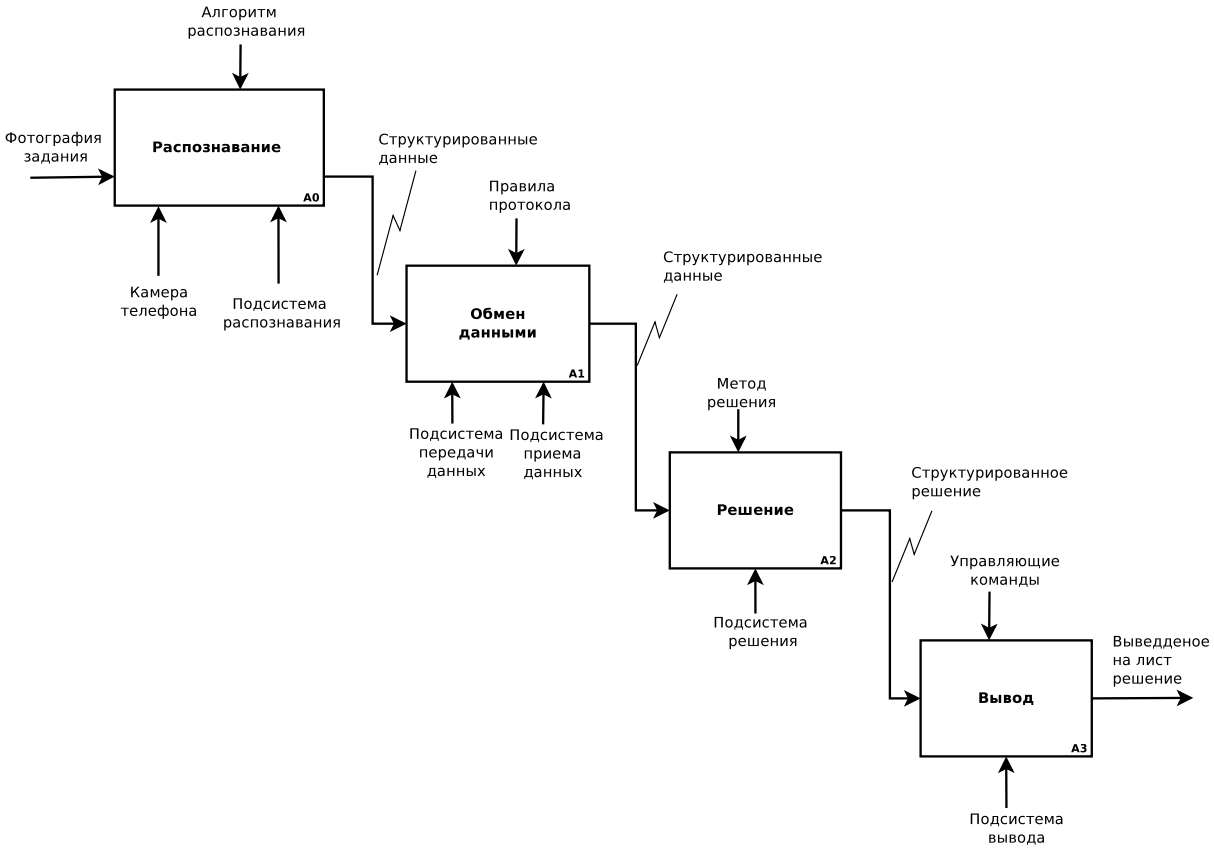

Рисунок 1.1 — Схема работы комплекса

Так же для удобства и простоты использования комплекса необходимо разработать мобильное приложение и протокол обмена между мобильным устройством и роботом, для того, чтобы стандартизировать сообщения и разрабатывать приложения для любых внешних платформ.

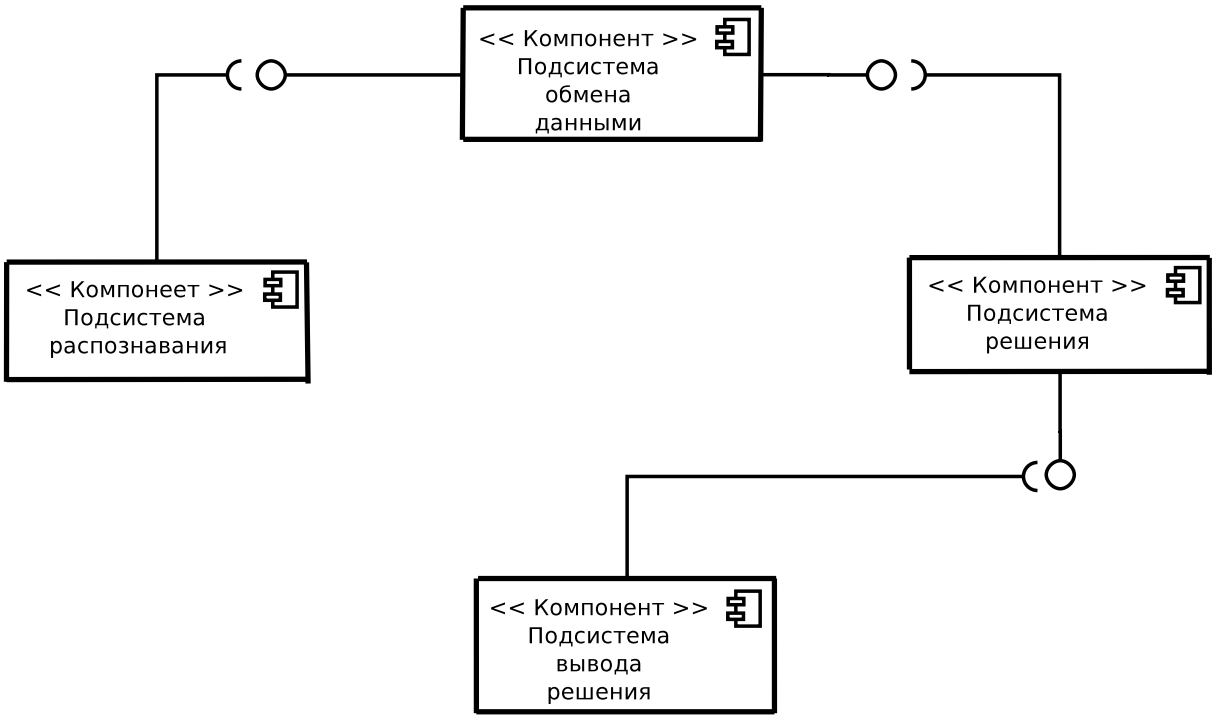

Рисунок 1.2 — Компоненты системы

1.2 Классификация мобильных роботов

Мобильный робот - автоматическая машина, в которой имеется движущееся шасси с автоматически управляемыми приводами. Такие роботы могут быть колёсными, шагающими и гусеничными (существуют также ползающие, плавающие и летающие мобильные робототехнические системы).

Микроконтроллер, микрокомпьютеры (англ. Micro Controller Unit, MCU) — микросхема, предназначенная для управления электронными устройствами. Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ.

Поскольку основная масса роботов отличается микроконтроллерами, введем классификацию по этому управляющему элементу.

Существующие робототехнические комплексы:

— LEGO Mindstorms;

— fischertechnik.

1.2.1 LEGO Mindstorms

LEGO Mindstorms - конструктор (набор сопрягаемых деталей и электронных блоков) для создания программируемого робота.

Наборы LEGO Mindstorms комплектуются набором стандартных деталей LEGO (балки, оси, колеса, шестерни) и набором, состоящим из сенсоров, двигателей и программируемого блока. Наборы делятся на базовый и ресурсный.

Все наборы содержат в себе одну и ту же версию интеллектуального блока NXT, отличаются только версии прошивки, но это не принципиально, так как прошивку можно легко обновить. Так что в этом плане все наборы совершенно равноценны.

В состав наборов могут входить управляющие блоки различных версий. В настоящее время их 3. Также у блоков существуют модификации (обозначается 1.0; 2.0 и т. д.)

Управляющие блоки:

— RCX - первое поколение управляющих блоков, в данный момент почти не используются из-за устаревшей конструкции модели;

— NXT - вторая версия коммерческого набора и самое распространенное поколение, 619 деталей в базовом комплекте, год выпуска 2009;

— EV3 - третье поколение, эволюция модели NXT, более 550 деталей, представлен в сентябре 2013 года.

Наборы LEGO Mindstorms располагают огромным количеством сенсоров как компании LEGO, так и сторонних производителей (HiTechnic, Mindsensors и др.).

1.2.2 fischertechnik

fischertechnik — пластмассовый развивающий конструктор для детей, подростков и студентов, изобретенный профессором Артуром Фишером в 1964 году. Наборы **fischertechnik** выпускает немецкая фирма **fischertechnik GmbH**, которая входит в состав крупного холдинга **fischertechnik GmbH & Co.KG**, дочерние фирмы которого выпускают крепёж, крепежный инструмент, детали для автомобилей и различные изделия из пластмассы.

Конструкторы fischertechnik часто используются для демонстрации принципов работы механизмов и машин в средних, специальных и высших учебных заведениях, а также для моделирования производственных процессов и презентационных целей.

Также в комплекты конструкторов входят программируемые контроллеры, двигатели, различные датчики и блоки питания, что позволяет приводить механические конструкции в движение, создавать роботов и программировать их с помощью компьютера.

Имеют только один вид контроллера: **ROBO TX** - это компактный программируемый контроллер для управления моделями, собранными из конструкторов fischertechnik.

Для разработки управляющих программ для контроллера ROBO TX используется среда программирования **ROBO Pro**. Готовые программы загружаются в контроллер через интерфейсы USB или Bluetooth.

1.2.3 Прочие робототехнические комплексы

Прочие робототехнические комплексы не представляют никакого интереса для изучения, т.к. имеют контроллеры специфичной конфигурации, узкое количество сенсоров и наборов, плохо документированные и поддерживаемые узкой группой энтузиастов среды для программирования.

1.2.4 Вывод

Мы будем использовать для написания данного дипломного проекта робототехнический комплекс **LEGO Mindstorms** по нескольким причинам:

— большое количество подключаемых модулей, как от самой компании производителя, так и от сторонних компаний;

— отличная документация на разных языках;

— самое большое сообщество робототехников, которые поддерживают и развивают данный комплекс;

— использование альтернативного ПО для программирования роботов.

В базовый комплект поставки Mindstorms NXT 2.0 уже включено большинство необходимых деталей для выполнения практически любых задач.

Но так как все описанные роботы располагают малым количеством встроенной FLASH-памяти (256 Кб), а изображение с заданием, которое необходимо будет распознать около 200 Кб, но на FLASH-память нам помимо сжатого изображения придется еще и записывать модули с другими системами, нам придется вывести **подсистему распознавания** на отдельное мобильное устройство, которое будет обладать достаточной разрешающей способностью для получения изображения нужного качества и обладающего достаточным количеством аппаратных ресурсов, чтобы выполнить возложенные на нее функции.

1.3 Подсистема распознавания

1.3.1 Общие представления о системе

Подсистема распознавания данных должна решать проблему перевода изображений и формирования из него структурированных данных, для последующей передачи другим подсистемам. Таким образом, распознанное изображение должно принять удобный

для хранения, использования и передачи, структурированный вид. Результатом работы такой системы должны быть данные, готовые к автоматизированной обработке.

Общая схема процесса представлена на (рис. 1.3).

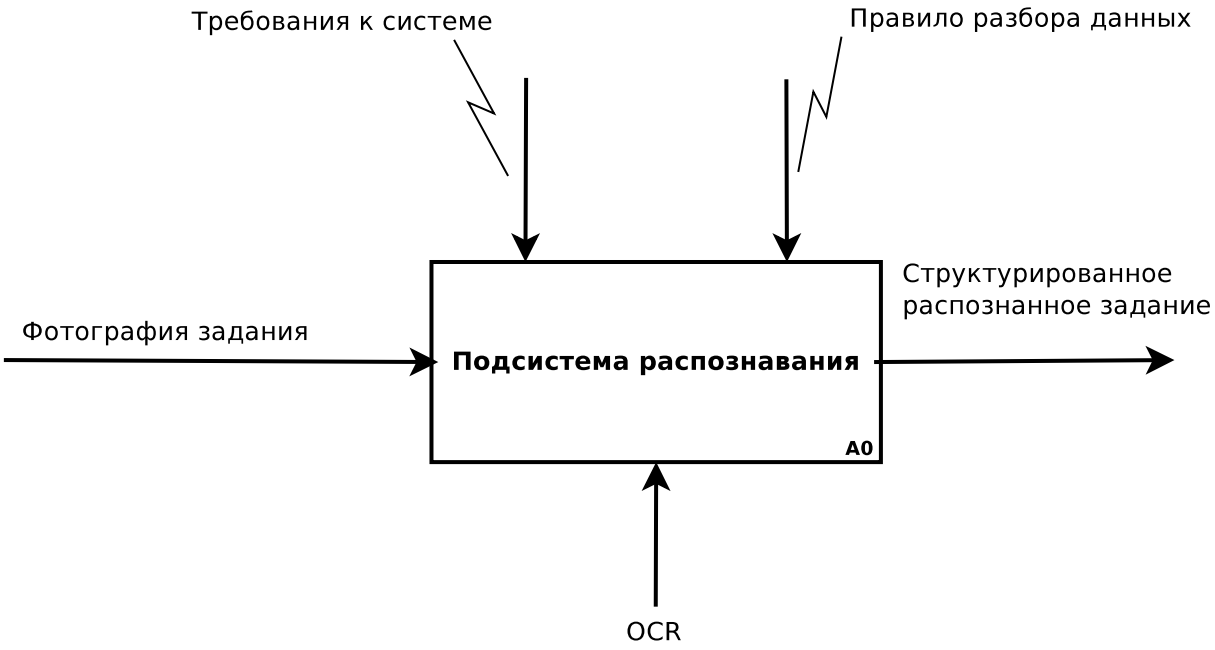

Рисунок 1.3 — Подсистема распознавания

1.3.2 Существующие системы распознавания изображений

На данный момент существует огромное количество программ, поддерживающих распознавание текста как одну из возможностей. Мы не будем рассматривать такие системы, так как в большинстве своем они избыточны, а наш программно-аппаратный комплекс будет работать в условиях ограниченных аппаратных ресурсов.

OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно использоваться в академических и коммерческих целях — распространяется в условиях лицензии BSD.

JavaANPR . Реализация – Java. Проект располагается по адресу `http://javaanpr.sourceforge.net`. Основное преимущество этой библиотеки в ее кроссплатформенности. Кроме этого, все алгоритмы написаны на Java без использования нативных библиотек, что сильно упрощает использование. Так же эту библиотеку с небольшой доработкой можно использовать на устройствах под управлением OS Android.

Огромный список подобного рода систем показывает, что распознавание информации является популярной и сложной задачей. Для универсального решения данной

задачи требуется использовать сложный программно-аппаратный комплекс, требующий огромных вычислительных мощностей(для наших мобильных устройств), что выходит за рамки данной работы и может воплотиться в виде развития рассматриваемой темы.

1.3.3 Возможности системы

Подсистема распознавания сбора данных должна обеспечивать получение изображение с камеры устройства, распознавание его и сохранение в нужной структуре. Таким образом приложение, реализующее данную систему, должно работать на мобильном устройстве и должна сохранять изображение в формате, пригодном для дальнейшего решения.

Для получения чисел из сетки судоку, нам надо определить, а где же наша сетка начинается и кончается. Эта часть является простейшей частью для человеческого мозга, но самой сложной для ПО. Почему? В них слишком много лишних данных. Очень часто газеты и журналы печатают несколько судоку рядом (хм, они явно не рассчитывают на компьютерное распознавание последних). На изображении будет слишком много лишних линий.

ПО очень сложно определить, какие линии относятся к необходимым нам, а какие являются всего лишь информационным шумом. Где конец нашей сетки и начало следующей.

Каждый алгоритм распознавания имеет три шага:

— определение необходимых признаков,
— тренировка,
— классификация (распознавание в реальном времени).

То есть на выходе мы должны иметь упорядоченную структуру в алфавите от 1 до 9.

1.3.4 Процесс распознавания

Для получения чисел из сетки судоку, нам надо определить, а где же наша сетка начинается и кончается. На изображении будет слишком много лишних линий.

Компьютеру очень сложно определить, какие линии относятся к необходимым нам, а какие являются всего лишь информационным шумом. Где конец нашей сетки и начало следующей.

После того, как мы определили границы, запустим алгоритм преобразования чтобы точно определить линии сетки. До сих пор мы не заботились о перекосах и других дефектах изображения. Только об угле поворота. Этот шаг исправит это. Мы получим точные положения линий сетки. Это поможет определить числа в сетке.

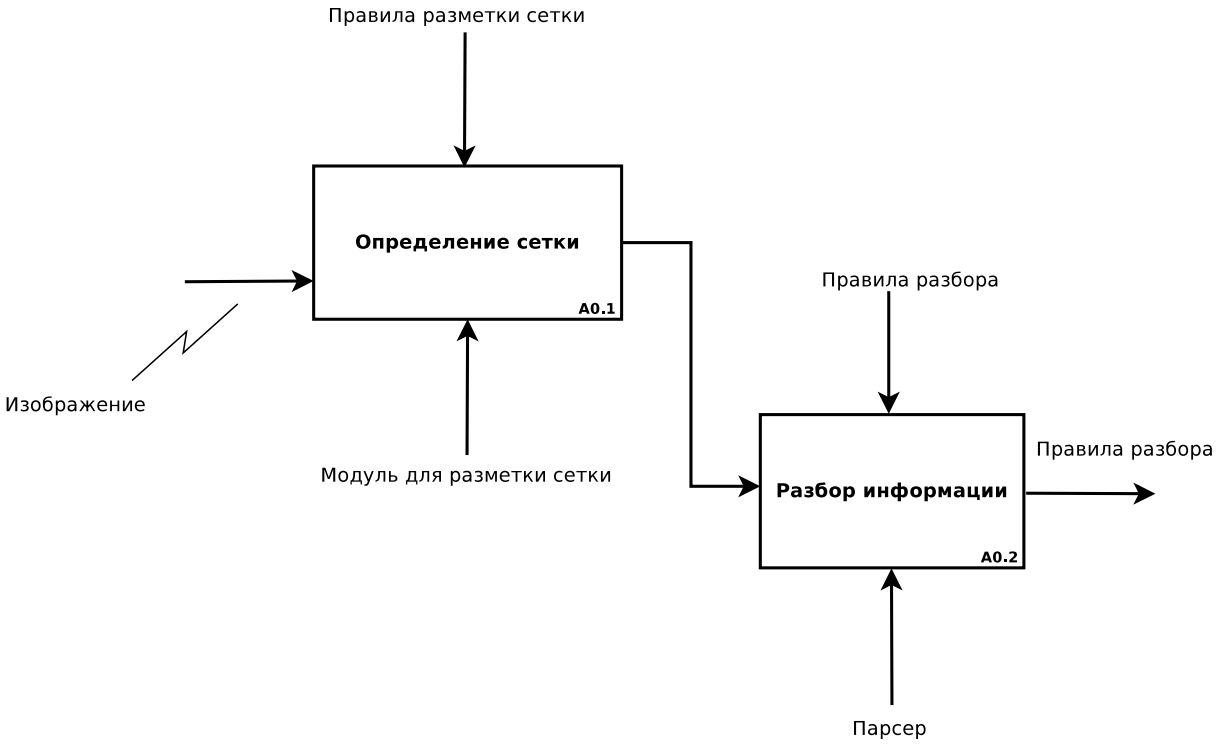

Рисунок 1.4 — Схема работы подсистемы сбора данных

После того, как мы определили, где должны находиться числа, нам необходимо распознать их. Это относительно легко. В алфавите только цифры от 1 до 9.

Скорректировать результаты распознавания.

Данный процесс представлен на (рис. 1.4).

1.3.5 Требования к подсистеме

В соответствии с проведенным анализом системы и процессов в ней сформулируем требования к подсистеме.

Требования к реализации

а) подсистема распознавания должна выполняться в виде отдельного сервиса;

б) новые данные должны распознаваться как можно быстрее;

в) она должна быть гибкой к изменениям угла обзора и поворота;

г) должен быть разработан удобный интерфейс добавления новых картинок с заданиями;

д) корректировка результатов OCR (в одной строке, столбце и блоке 3x3 не может находиться одна и та же цифра).

Входные данные подсистемы

а) Изображение, полученное с камеры мобильного устройства.

Выходные параметры

а) Результат обработки заявки в виде определенной структуры данных;

б) в случае, если распознавание не прошло полный цикл обработки, данные отправляются для повторного распознавания;

в) информация о состоянии распознавания.

1.4 Подсистема обмена данными

Главной задачей разрабатываемой системы является предоставление доступа сторонним приложениям доступ к данных, хранящихся на разных компонентах системы. Второй важнейшей задачей является передача данных. Эту проблему призвано решить вторая рассматриваемая подсистема.

В качестве данных в подсистеме обмена выступают распознанные изображения с заданиями, затем нам надо предоставить эти данные подсистеме решения. Такой интерфейс и будет предоставлять текущая подсистема.

Для обменна данными между разными аппаратными модулями единственной и **беспроводной** спецификацией остается **Bluetooth**.

Bluetooth — стандарт коротковолновых технологий, который обеспечивает связь между беспроводными устройствами: мобильными телефонами, КПК, планшетными компьютерами, ноутбуками с поддержкой беспроводной связи, обычными компьютерами и внешними устройствами. Как только связь между двумя устройствами устанавливается, ни одно другое устройство не сможет нарушить эту связь. Устройства не обязательно должны находиться в одном помещении и могут располагаться на расстоянии до 30 метров. Если у вас есть проблемы с использованием Bluetooth, просто щелкните здесь

Так как Bluetooth есть практически в любых совмренных смартфонах, планшетах и мобильных устройствах, то у нас будет доступ к API необходимого устройства и контроллере робота будет свой API, который задействует нужную нам спецификацию.

При создании подсистемы обмена данными наша задача сведется к следующим шагам:

— установить сопряжение смартфона и блока управления NXT;
— подключиться к блоку NXT;
— передать правильную команду;
— получить ответ.

Наша основная задача сводится к тому, что сформировать структуру сообщений, которыми будут обмениваться устройства для разных задач. Например, управляющие команды для робота стандартизированы и указаны в спецификации к контроллеру, а вот формат сообщений с заданиями нам надо согласовать, чтобы он был единым между смартфоном и контроллером робота.

1.4.1 Сопряжение устройств

Эту процедуру необходимо проделать один раз на самом первом этапе развертывания системы, в дальнейшем устройства будут уже сопряжены.

Инициализацией bluetooth-соединения принято называть процесс установки связи. Её можно разделить на три этапа:

— генерация ключа Kinit;
— генерация ключа связи (он носит название link key и обозначается, как Kab);
— Аутентификация.

Первые два пункта входят в так называемую процедуру сопряжения.

Сопряжение — процесс связи двух (или более) устройств с целью создания единой секретной величины Kinit, которую они будут в дальнейшем использовать при общении. В некоторых переводах официальных документов по bluetooth можно также встретить термин «подгонка пары». Перед началом процедуры сопряжения на обеих сторонах необходимо ввести PIN-код.

1.4.2 Подключение к блоку NXT

Подключение к блоку NXT это по большому счету операция сопряжения между устройствами, но нам предоставляется доступ к **управляющим командам** самого робота. Формат команд жестко задекларирован в [1] и [2].

1.4.3 Обмен данными и командами

Для обмена информацией и команд между устройствами на контроллере робота предусмотрен определенный формат обмена команд, под который нам нужно будет подстроить устройство с системой распознавания.

Листинг 1.1 — Формат команд для получения потока данных

```
1 MessageWrite
2
3 Byte 0: 0x00 or 0x80
4 Byte 3: Message size
5 Byte 4: Message Text
```

В данном примере **Byte 0** говорит нам, что это команда прошла успешно, **Byte 3** и **Byte 4** соответственно указывают на размер пересылаемого сообщения и на само сообщение.

Листинг 1.2 — Формат команды ответа

```
1 Return package:
2
3 Byte 0: 0x02
4 Byte 2: Status Byte
```

В ответном сообщении, **Byte 0** указывает нам, что это выходная команда (типа Output) и ее статус **Byte 2**. Для подсистемы обмена информации нам нужно отправлять заранее заготовленные команды на контроллер робота и сформировать их в отдельный сервис.

1.5 Подсистема решения

Судоку - популярная головоломка с числами.

Игровое поле представляет собой квадрат размером 9×9, разделённый на меньшие квадраты со стороной в 3 клетки. Таким образом, всё игровое поле состоит из 81 клетки. В них уже в начале игры стоят некоторые числа (от 1 до 9), называемые подсказками. От игрока требуется заполнить свободные клетки цифрами от 1 до 9 так, чтобы в каждой строке, в каждом столбце и в каждом малом квадрате 3×3 каждая цифра встречалась бы только один раз.

Сложность судоку зависит не от количества изначально заполненных клеток, а от методов, которые нужно применять для её решения.

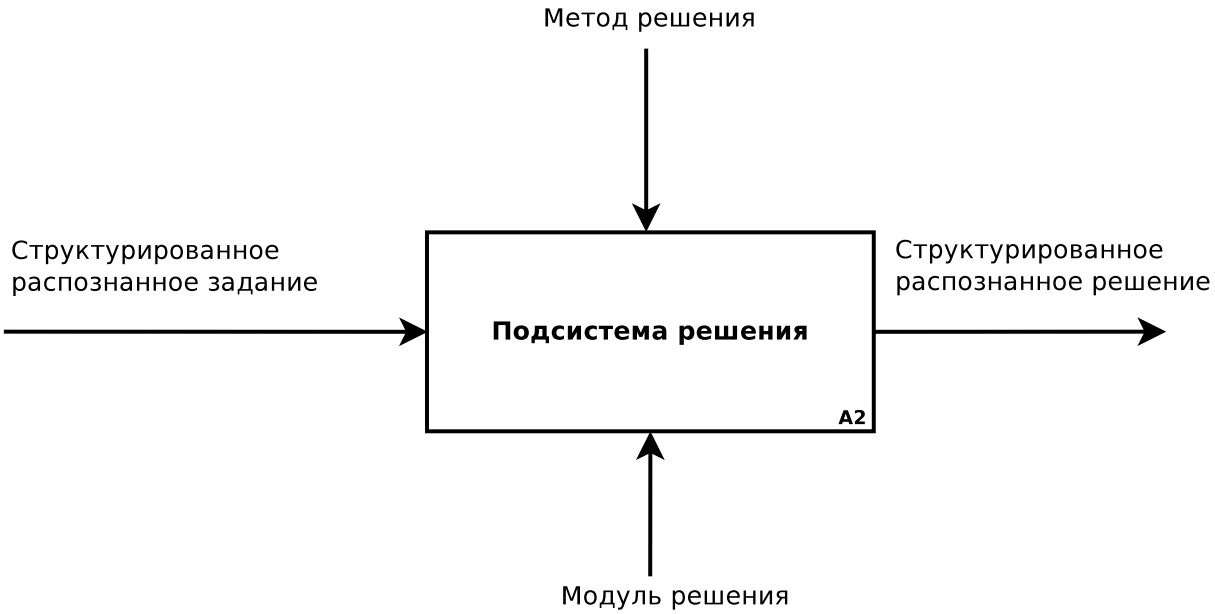

Рисунок 1.5 — Подсистема решения судоку

Правильно составленная головоломка имеет только одно решение. Тем не менее, встречаются варианты судоку с несколькими вариантами решения, а также с ветвлениями самого хода решения.

Общая схема процесса представлена на (рис. 1.5).

1.5.1 Методы решения

Существует несколько способов решения судоку.

Сложность судоку зависит не от количества изначально заполненных клеток, а от методов, которые нужно применять для её решения. Самые простые решаются дедуктивно: всегда есть хотя бы одна клетка, куда подходит только одно число.

Метод перебора

Это самый широко используемый программистами способ, который точно даёт решение независимо от уровня сложности. Но перебор может быть очень долгим, это зависит от глубины рекурсии. Вы никогда не узнаете, сколько же итераций вам понадобиться. Во время перебора в клетку ставятся поочерёдно возможные из значений от 1 до 9 и продолжается решение судоку с поставленным числом. Если мы зашли в тупик (получили в одной строке/столбце/блоке одинаковые цифры, то меняем цифру на следующую). На самом деле, может быть больше одного решения.

Во-первых, необходимо подготовить таблицу кандидатов – возможных значений для каждой пустой клетки. Рис. 1.5 объясняет, что такое кандидаты. Они окрашены в другой цвет и имеет меньший шрифт.

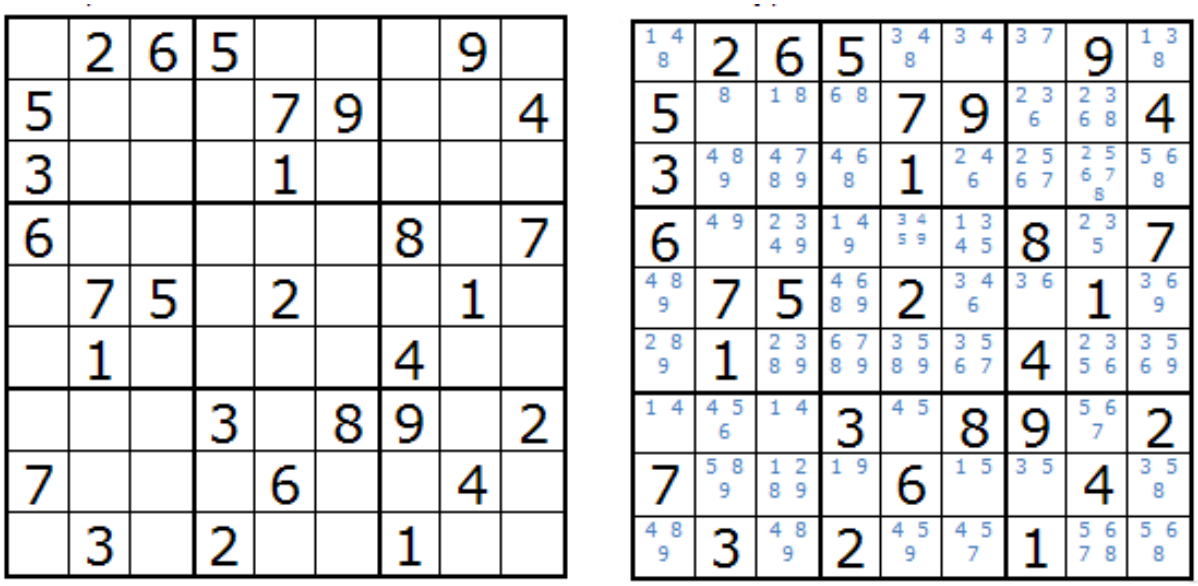

Рисунок 1.6 — Метод перебора

Метод перебора пробует скомбинировать кандидаты пока не найдёт решения.

Перебор может быть очень медленным, если решение требует много итераций.

Метод открытых цифр

Рис. 1.7 объясняет этот метод. Если ячейка имеет единственного кандидата, то мы с уверенностью можем сказать, что именно эта цифра должна стоять в этой клетке. После установки значения, следующим шагом будет перестроить список кандидатов. Список кандидатов уменьшается, пока существуют одиночные кандидаты.

Это очевидный и простой метод для машинного решения.

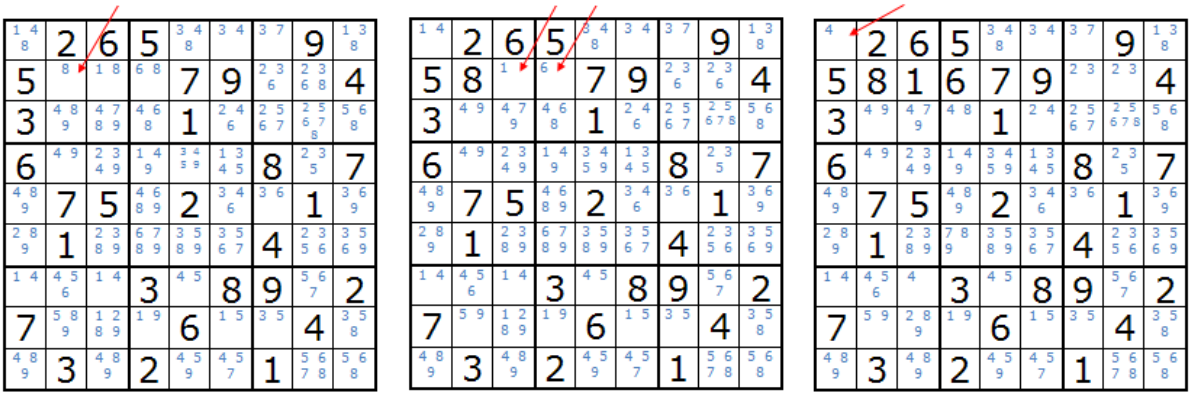

Рисунок 1.7 — Метод открытых цифр

Метод скрытых пар

Отличным способом раскрыть поле будет поиск скрытых пар. Этот метод позволяет убрать лишних кандидатов из ячейки и дать развитие более интересным стратегиям. Показан на рис. 1.8

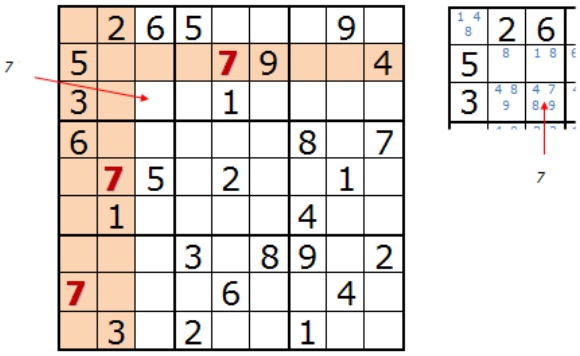

Рисунок 1.8 — Метод скрытых пар

1.5.2 Вывод

Все описанные методы имеют свои недостатки и преимущества.

Для нас одно из требований для поиска решения - **скорость обработки**.

Перебор не слишком быстрый метод. Таким образом, мы будем использовать комбинацию всех трёх методов.

Метод открытых цифр и метод скрытых пар очень быстрые, но могут решать только быстрые пазлы.

Так как наша подсистема должна решать любые пазлы, нам необходимо скомбинировать решения.

Комбинируя условия, мы можем утверждать, что в некоторых ячейках будут только конкретные значения и все другие кандидаты мы убираем.

1.6 Подсистема графомоторных навыков

1.6.1 Общие представления о системе

Одна из главных подсистем в рамках данной дипломной работы это система, которая позволит вывести нам наше решение на лист бумаги.

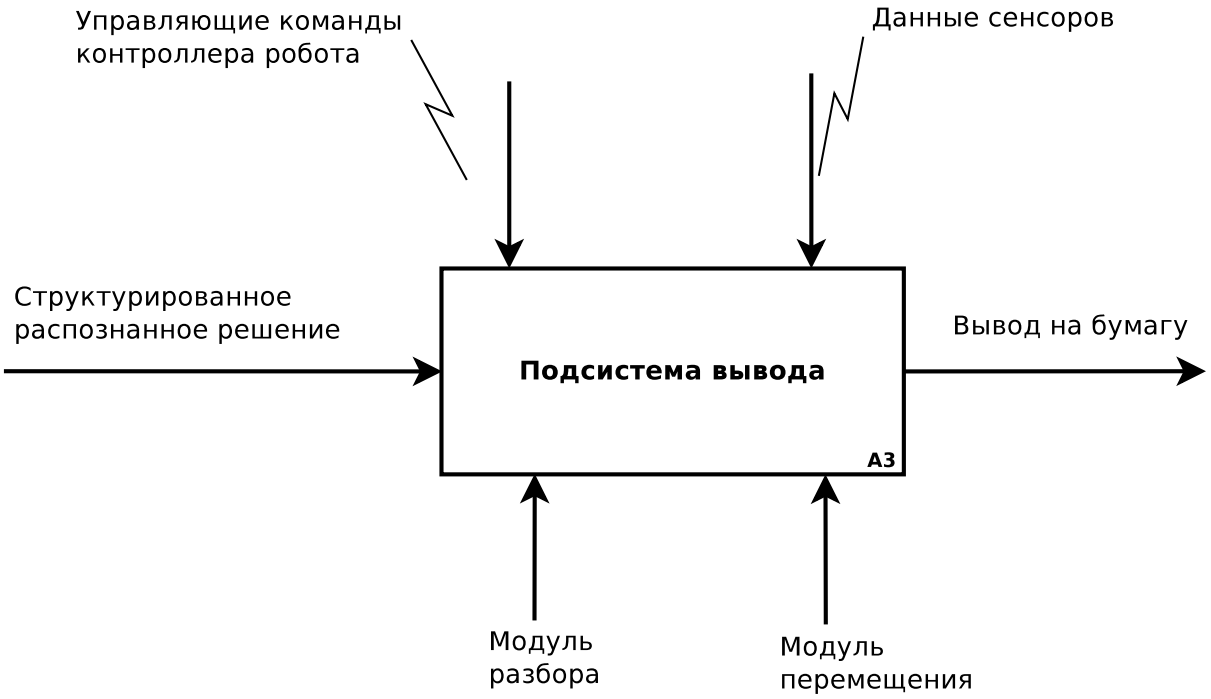

Рисунок 1.9 — Общая схема подсистемы графомоторных навыков

После получения решения, нашему контроллеру с данной системой предстоит решить несколько задач:

— определить ячейку и ее границы;
— определить есть ли там цифра из задания или нужно записать;
— записать нужную цифру;
— определить конец строки/столбца и начать заполнять новый.

Общая схема процесса представлена на (рис. 1.9).

Для того, чтобы заставить робота двигаться и рисовать плавно, нужно заставить его самому считать скорость движения.

Процесс вывода решения на бумагу представлен на (рис. 1.10).

1.6.2 Определение ячеек и их типа

Функцию зрения на нашем роботе будет осуществлять сенсор цвета.

Сенсор цвета выполняет три уникальных функции при использовании с блоком NXT.

а) может работать как датчик цвета и распознавать шесть цветов;

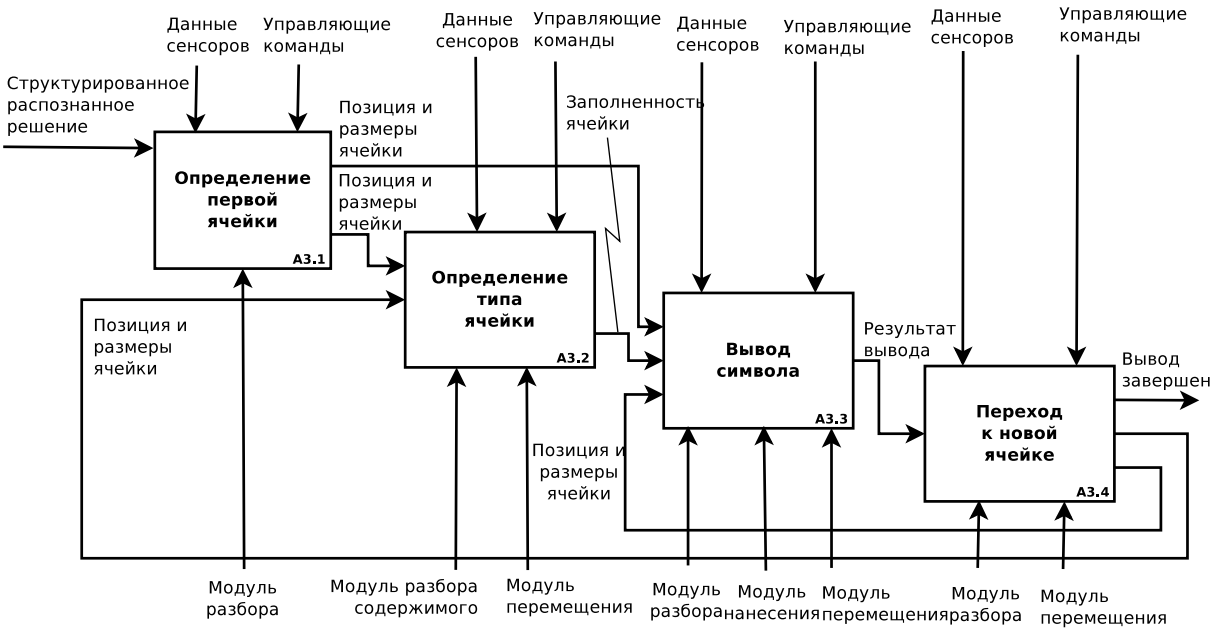

Рисунок 1.10 — Процесс вывода решения на бумагу

б) быть датчиком света, определяющим интенсивность освещения и регистрировать его уровень;

в) выполнять функции цветной лампы красного, зеленого и синего цветов.

Человек видит черную линию и ее четкую границу. Датчик освещенности работает несколько иначе.

Датчик освещенности видит некое подобие градиента от белого до черного.

На рис. 1.11 показана разница в восприятии черной границы для человеческого глаза и робота.

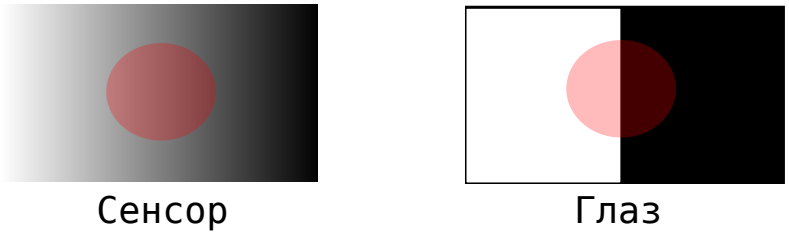

Рисунок 1.11 — Граница для человеческого глаза и сенсора цвета

Именно это свойство датчика освещенности – невозможность четко различить границу белого и черного – мы и будем использовать для расчета скорости движения.

Введем понятие **идеальной точки траектории**.

Идеальная точка – условная точка примерно посередине белого и черного цветов, следуя которой робот будет регулировать скорость при приближении черной линии.

$$I_{\text{идеал}} = \frac{I_{max}}{k} \tag{1.1}$$

где I_{max} — показатель освещенности; k — коэфициент отношения на полярных цветах.

Таким образом получая данные с сенсора цвета мы можем выполнить две фундаментальные задачи:

а) определить где граница ячейки;

б) определить не является ли эта ячейка заполненной.

1.6.3 Требования к системе

В соответствии с проведенным анализом системы и процессов в ней сформулируем требования к подсистеме.

Требования к реализации

а) подсистема распознавания должна выполняться в виде отдельного модуля;

б) размеры поля не фиксированы;

в) система должна уметь определить масштаб для нанесения решения.

Требования к надежности

а) все символы в системе должны доходить до финальной стадии своей обработки;

б) символы должны рисоваться с учетом размеров клеток, строго в своих границах;

в) символы из задания не должны быть испорчены, повторной попыткой нанести символ;

г) робот должен рисовать только в рамках границ поля.

Описание задачи

— рассчитать мощность вращения каждого из двигателей с учетом степени отклонения от идеальной точки.

Входные данные

— идеальная точка;

— текущие показания датчика освещенности.

Выходные данные

— тип движения, которые должен осуществить робот.

1.6.4 Вывод

Данная подсистема позволяет выводить элементы с учетом внешних факторов, таких как:

— находится ли робот на холсте для нанесения;
— можно ли выводить что-то на этом сегменте;
— не вышли ли мы за границы сегмента.

Она является автономной и при необходимости может переноситься для решения других задач, добавляя алфавит.

2 Конструкторский раздел

В данном разделе на основе проведенного анализа описывается взаимодействие между подсистемами, выделяются основные составляющие в каждой системе и проектируется их взаимодействие.

2.1 Конструкция робота

На рис. 2.1 показана конструкция робота со всеми датчиками и двигателями в сборке.

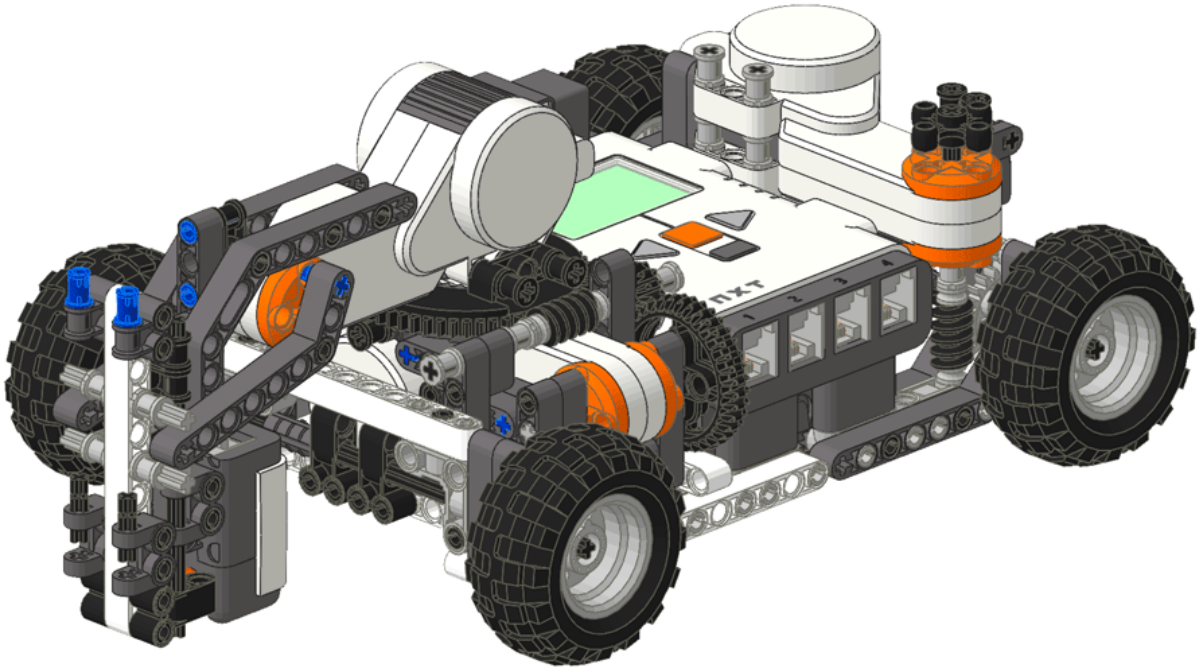

Рисунок 2.1 — Конструкция робота

Сенсоры и моторы, которые нам понадобятся:

а) сенсор цвета;
б) левый двигатель;
в) правый двигатель;
г) двигатель манипулятора.

Сенсор цвета нам необходим, чтобы робот мог получать входные данные о:

а) положении робота;

б) размерах ячеек;
в) положении ячеек;
г) содержимом ячеек.

Двигатели необходимы для передвижении робота на колесах по бумаге.

Двигатели манипулятора необходим для:

а) перемещения сенсора цвета;
б) для передвижения ручки.

Конструктивная особенность данного робота заключается в том, что часть, с которой поступает информация находится на манипуляторе, и та часть, которая ответственна за вывод так же находится на том же манипуляторе.

2.2 Подсистема распознавания

2.2.1 Основные задачи

В данной секции сформируем опишем уже формализованный список задач, которые нам необходимо будет произвести над картинкой. Можно выделить основные моменты при данной системой:

а) сфотографировать;
б) перевести изображение в черно-белые цвета;
в) определить угол наклона изображения;
г) определить сетку;
д) распознать и разместить цифры в сетке;
е) скорректировать распознавание под правила Судоку.

2.2.2 Фотографирование

Данный процесс делается стандартными средствами Android API. Необходимо лишь получить изображение из камеры или галереи, после чего передать изображение на обработку.

2.2.3 Конвертация в монохромное изображение

Каждое приложение с компьютерным зрением начинает с конвертации цветного (или чёрно-белого) изображения в монохромное. В будущем, возможно, будет какой-то алгоритм, который будет использовать цвета, но сегодня приложения компьютерного зрения работают с монохромными изображениями (они дальтоники). Самый простой метод для конвертации изображения – это общий порог. Предположим, что у вас есть пиксель с цветом RGB (200, 200, 200). Так как интенсивность компонент изменяется от 0 до 255, то пиксель очень яркий. Выбрав порог, как половину интенсивности: 256/2=128, мы получим,

что наш пиксель должен стать белым. Но общий порог редко используется в настоящих приложениях, так как он малополезен. Куда более полезен алгоритм локального порога.

На рис. 2.2 показаны оригинальное фото (1), фото после обработки с общим порогом (2) и адаптивным порогом (3) .

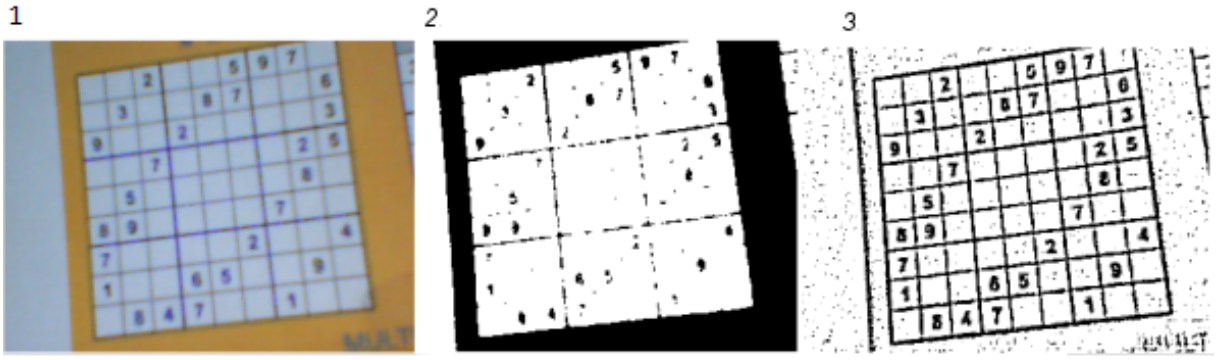

Рисунок 2.2 — Пороги при создании монохромного изображения

Для правильной конвертации изображения в монохромное, мы будем использовать адаптивный выбор порога. Он не использует фиксированное значение порога в 128. Вместо этого, он считает порог для каждого пикселя отдельно. Он берёт квадрат со стороной в l пикселей и с центром в нашем пикселе и суммирует интенсивность всех точек. Среднее значение интенсивности и будет являться порогом для данного пикселя. Формула интенсивности для текущего пикселя: порог $= \frac{\text{сумма}}{l^2}$. Если интенсивность нашего пикселя выше порогового значения, то он конвертируется в белый, если же нет, то в чёрный. На изображении ниже пиксель, для которого определяется порог, отмечен красным. И эти подсчёты производятся для каждого пикселя. Поэтому данный шаг является таким медленным, ведь алгоритм требует ширина$*$порог$*l^2$ чтений пикселя изображения. Схема показана на рис. 2.3

2.2.3.1 Целочисленная форма

Этот шаг может быть оптимизирован с помощью использования «Целочисленной формы». Целочисленная форма – это массив целых чисел с размерами изображения. Допустим у нас есть область на рис. 2.19 $12*12$, где интенсивность пикселей равна 1 (в реальном мире так не бывает), целочисленный образ – это сумма всех пикселей с левого верхнего до текущего (правого нижнего).

Следующий рис. 2.17 демонстрирует, чем может быть полезен целочисленный образ. Целью является посчитать сумму пикселей в сером прямоугольнике. Формула: сумма $= D-C-B+A$. В нашем случае: $110 - 44 - 40 + 16 = 42$.

Вместо чтения всех пикселей из серого прямоугольника (который может быть намного больше, чем в примере), нам необходимо всего лишь одно чтение памяти. Это

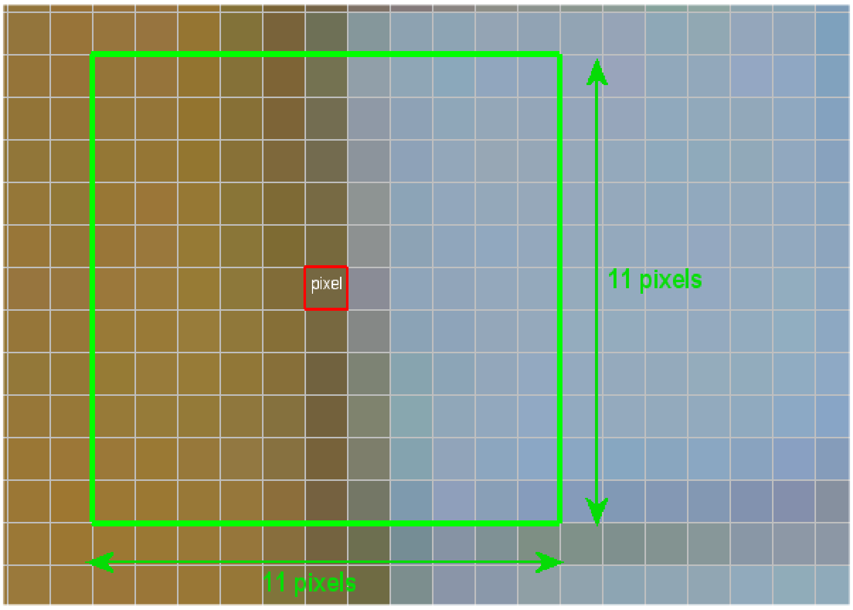

Рисунок 2.3 — Адаптивный выбор порога при создании монохромного изображения

1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

Рисунок 2.4 — Целочисленная форма

значительная оптимизация алгоритма. Но даже с ней, конвертация изображения в монохромное является очень тяжёлой.

2.2.3.2 Определение угла поворота

Камера не сканер. И картинка никогда не будет идеально ровно, а значит в порядке вещей немного скошенное и повёрнутое изображение. Чтобы определить угол поворота, мы будем пользоваться фактом, что изображение с судоку всегда имеет горизонтальные и вертикальные линии. Мы будем определять самые выразительные и жирные линии рядом с центром изображения. Самые выразительные линии не подвержены зашумлению.

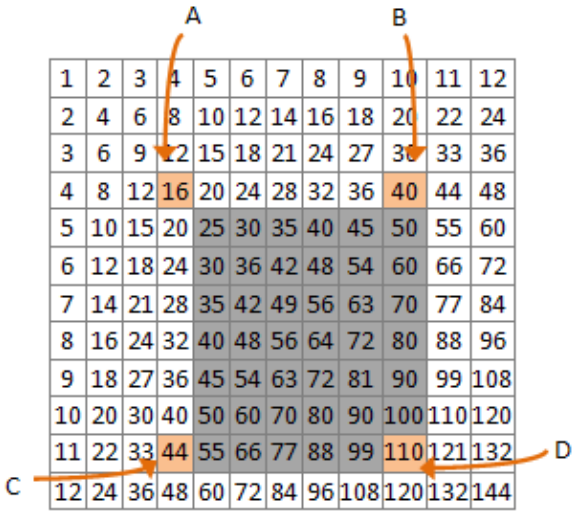

Рисунок 2.5 — Целочисленная форма в области

Алгоритм нахождения монохромных линий на изображении называется преобразованием Хафа.

$$y = \frac{x * \cos\theta + rho}{\sin\theta} \tag{2.1}$$

где θ — угол линии; rho — является расстоянием от линии до центра координат (0, 0). На рис. 3.1

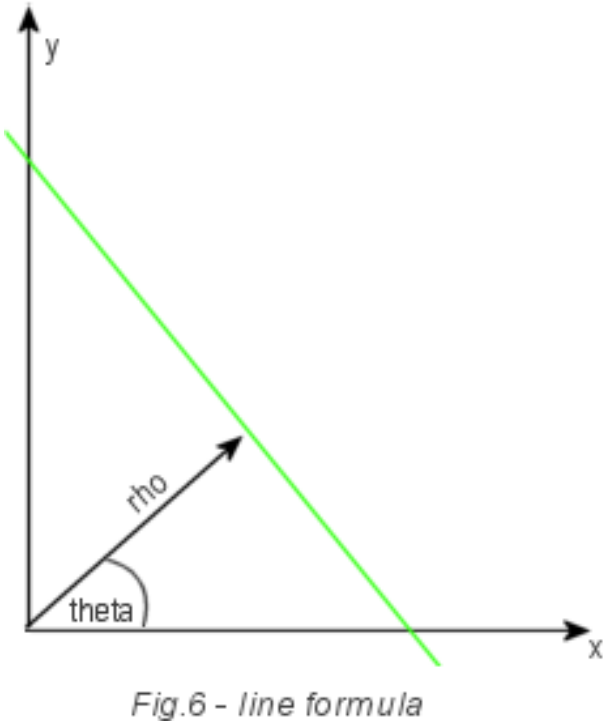

Рисунок 2.6 — Угол поворота

Важным является то, что линия может быть описана всего двумя переменными: углом наклона и расстоянием до центра координат.

Алгоритм проходит по всем пикселям в монохромном изображении, пропуская белые пиксели. Когда он попал на чёрный пиксель, то пытается «нарисовать» всевозможные линии, проходящие через этот пиксель с шагом в 1 градус. Это означает, что каждый пиксель имеет 180 воображаемых линий, проходящих через него, потому что углы 180-360 являются копиями линий с углами 0-180 градусов.

Это множество воображаемых линий называется накопителем, двумерным массивом с размерностями θ и rho, которые были в формуле выше. Каждая воображаемая линия представлена одним значением в накопителе. Кстати, метод называется преобразованием, потому что он преобразовывает линии из (x, y) в массив (θ, rho). Каждая воображаемая линия добавляет значение в накопитель, повышая вероятность того, что воображаемая линия совпадает с реальной. По аналогии с голосованием. Настоящие линии имеют больше всего «голосов».

После того, как все пиксели и их 180 воображаемых линий «проголосовали», мы должны найти максимальное значение в накопителе. (Накопителем он называется потому что накапливает голоса) Победителем голосования является самая выразительная линия. Её параметра θ и rho могут использоваться с помощью формулы выше чтобы нарисовать её.

На следующем рис. 2.7 приведён небольшой пример. Слева мы имеем линию, состоящую из трёх пикселей. Вы точно знаете, что это диагональная линия слева направо, но для компьютера это не так очевидно.

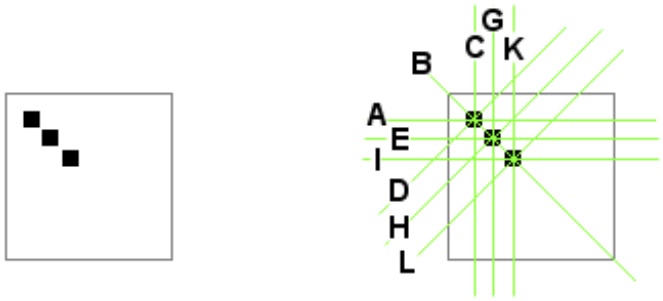

Рисунок 2.7 — Определение нужной линии

Посмотрев на это изображение, можно понять, как же работает обнаружение линий. Алгоритм преобразования Хафа пропускает белые пиксели. На каждом чёрном пикселе он рисует четыре воображаемых зелёных линии (на самом деле их 180, но для простоты мы возьмём четыре), проходящие через текущий пиксель. Первый пиксель голосует за линии A, B, C и D. Второй пиксель голосует за линии E, B, G, H. Третий за I, B, K, L. За линию B проголосовали все три пикселя, а остальные линии получили всего по одному голосу. Таким образом, линия B победила в голосовании.

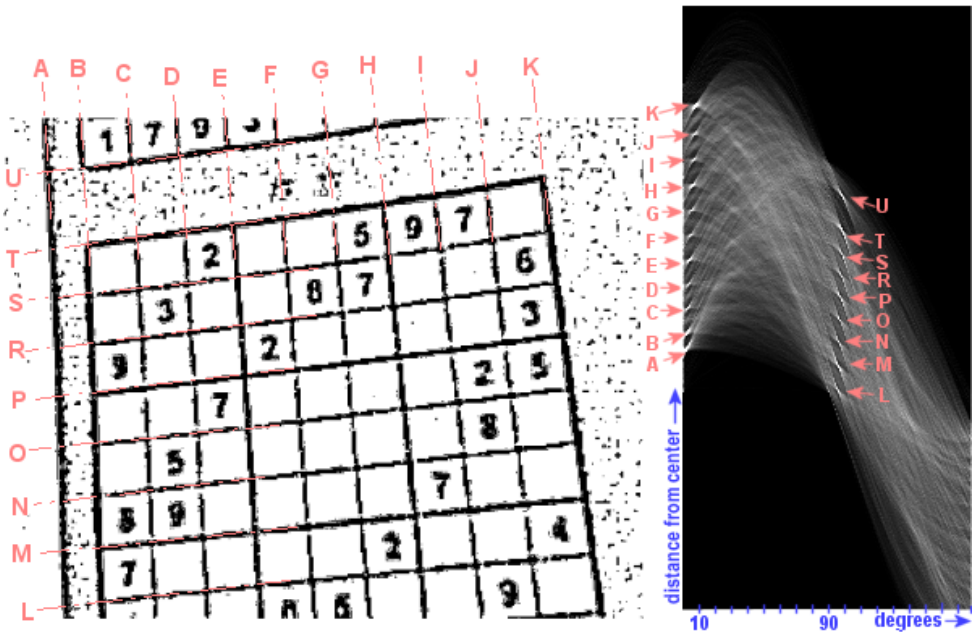

Рисунок 2.8 — "Голосование"за линию

Следующий рис. 2.8 является более сложным примером. Слева мы видим сетку судоку, а справа массив накопителя после прохождения по изображению алгоритма преобразования Хафа. Яркие светлые области являются линиями, которые получили много голосов. Темнота означает, что у нас нет шанса найти линию с такими параметрами на изображении. Сконцентрируемся только на ярких точках. Каждая линия $A - U$ имеет яркую точку в накопителе. Вы можете видеть, что все линии слегка повёрнуты (примерно на 6 градусов). Линия A меньше повёрнута, чем линия . Потому что изображение не только повёрнуто, но и скошено. Также, если вы посмотрите внимательнее, то увидите, что линии и ближе друг к другу, чем и . Это можно заметить на обоих изображениях.

Преобразование Хафа важно понять, если вы хотите заниматься распознаванием образов. Концепция виртуальных линий, которые могут быть реальными с помощью голосования, может быть распространена на любую геометрическую фигуру. Линия – простейшая из них, поэтому является самой простой для понимания.

Если вам надо найти окружность, то понадобится накопитель с тремя размерностями: x, y и r. Где x и y являются координатами центра нашей окружности, а r радиусом.

Преобразование Хафа может быть оптимизировано ограничением области и возможных углов исходного изображения. Нам не надо анализировать все линии, только те, что близки к центру. В нашем случае, именно от них мы и можем отталкиваться.

2.2.3.3 Определение сетки

Для получения чисел из сетки судоку, нам надо определить, а где же наша сетка начинается и кончается. Эта часть является простейшей частью для человеческого мозга, но самой сложной для автоматического распознавания. Почему? Почему бы не исполь-

зовать линии, найденные с помощью преобразования Хафа в предыдущем шаге? В них слишком много лишних данных. На изображении будет слишком много лишних линий.

Пример можно увидеть, посмотрев на рис. 2.9.

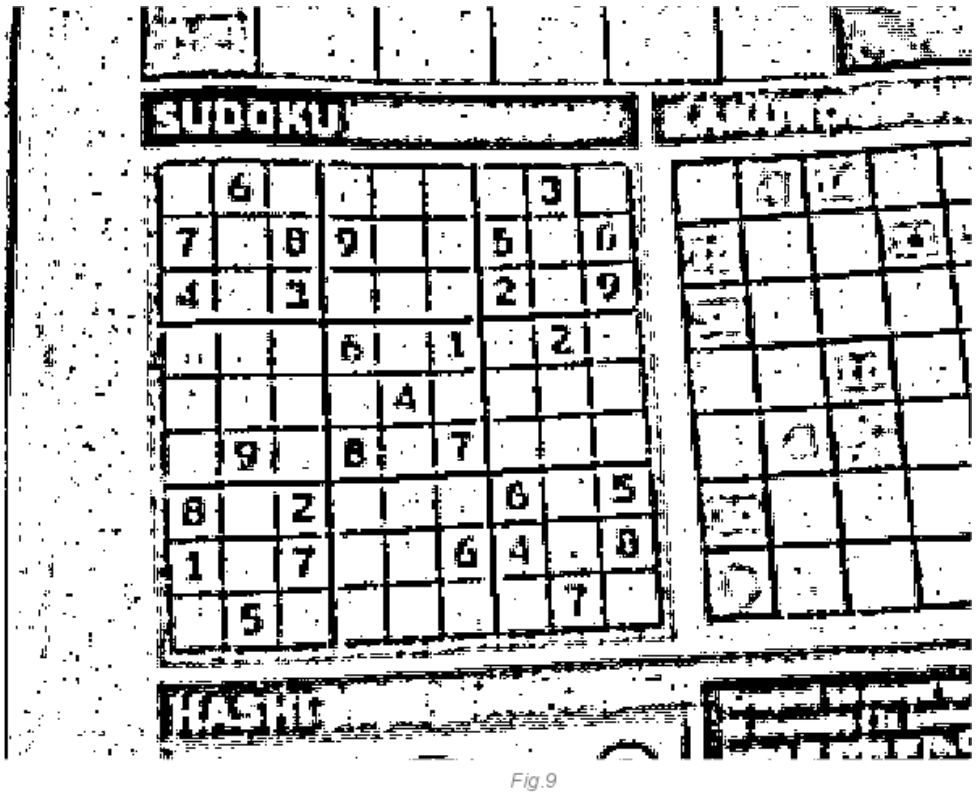

Рисунок 2.9 — Зашумленное изображение

При автоматическом распознавании очень сложно определить, какие линии относятся к необходимым нам, а какие являются всего лишь информационным шумом. Где конец нашей сетки и начало следующей.

Чтобы решить эту проблему, мы не будем определять чёрные линии. Вместо этого, мы будем определять белые области вокруг сетки. На рис. 2.10, вы можете увидеть это. Зелёная линия 1 никогда не пересекается с чёрными пикселями в то время, как линия 2 делает это 10 раз. Это означает, что за сеткой скорее находится линия 1, нежели линия 2. Просто подсчитав, сколько раз зелёная линия пересекает чёрные пиксели, мы можем сделать вывод, что она находится за границей сетки. Достаточно просто подсчитать как много переходов с белого на чёрный пиксель под линией. Кстати, вам не надо пробегаться по каждой линии, достаточно выполнять эти действия каждую третью линию для увеличения скорости выполнения.

После того, как мы определили границы, запустим алгоритм преобразования Хафа чтобы точно определить линии сетки. До сих пор мы не заботились о перекосах и других дефектах изображения. Только об угле поворота. Этот шаг исправит это. Путём запуска преобразования Хафа, мы получим точные положения линий сетки. Это поможет определить числа в сетке.

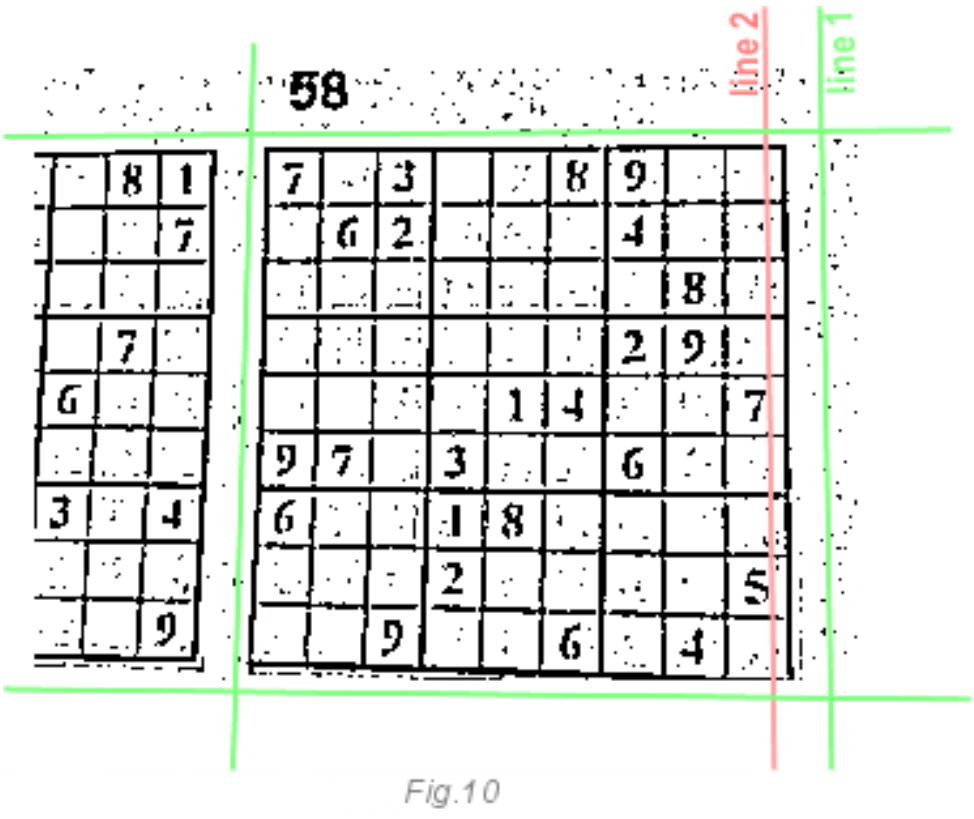

Рисунок 2.10 — Отбрасывание информационного шума

Этот шаг может быть более устойчивым к шуму.

2.2.4 Модуль OCR

После того, как мы определили, где должны находиться числа, нам необходимо распознать их. Это относительно легко. В алфавите только цифры от 1 до 9.

Теория. Каждый алгоритм распознавания имеет три шага:

а) определение необходимых признаков;
б) тренировка;
в) классификация (распознавание в реальности).

Определение необходимых признаков является частью разработки приложения. Например, цифра один тонкая и высокая. Именно это и отличает её от остальных. Цифра 8 имеет две окружности, в верхней половине и в нижней, и т. д.

Определение необходимых признаков может быть трудной и не интуитивной работой, в зависимости от того, что надо распознать. Например, что необходимо для распознавания чьего-то лица? Не любого, а какого-то конкретного.

2.2.4.1 Зоны

В нашем приложении был использован способ плотности зон. Следующим шагом (Это должно быть сделано заранее) является тренировка приложения на цифры от 1 до

9. Образцы эти изображения в каталоге хранятся в системе. Они уменьшены до размера 5x5, нормализованы и хранятся в статическом массиве **OCRDigit[10]**, который выглядит примерно так как на рис. 2.11:

Рисунок 2.11 — Нормализованные и уменьшенные образцы изображений

Уменьшение до размеров 5x5 называется зоннированием или разбивкой на зоны. Массив выше называется плотностью функции. Нормализация значит, что плотность варьируется от 0 до 1024. Без нормализации зоны нельзя сравнивать корректно.

Что происходит во время выполнения программы: когда цифра получена из исходного изображения, она уменьшается до размера 5x5. После этого каждый её пиксель сравнивается с каждым пикселем из девяти цифр из тренированного массива. Целью является найти цифру с минимальным различием.

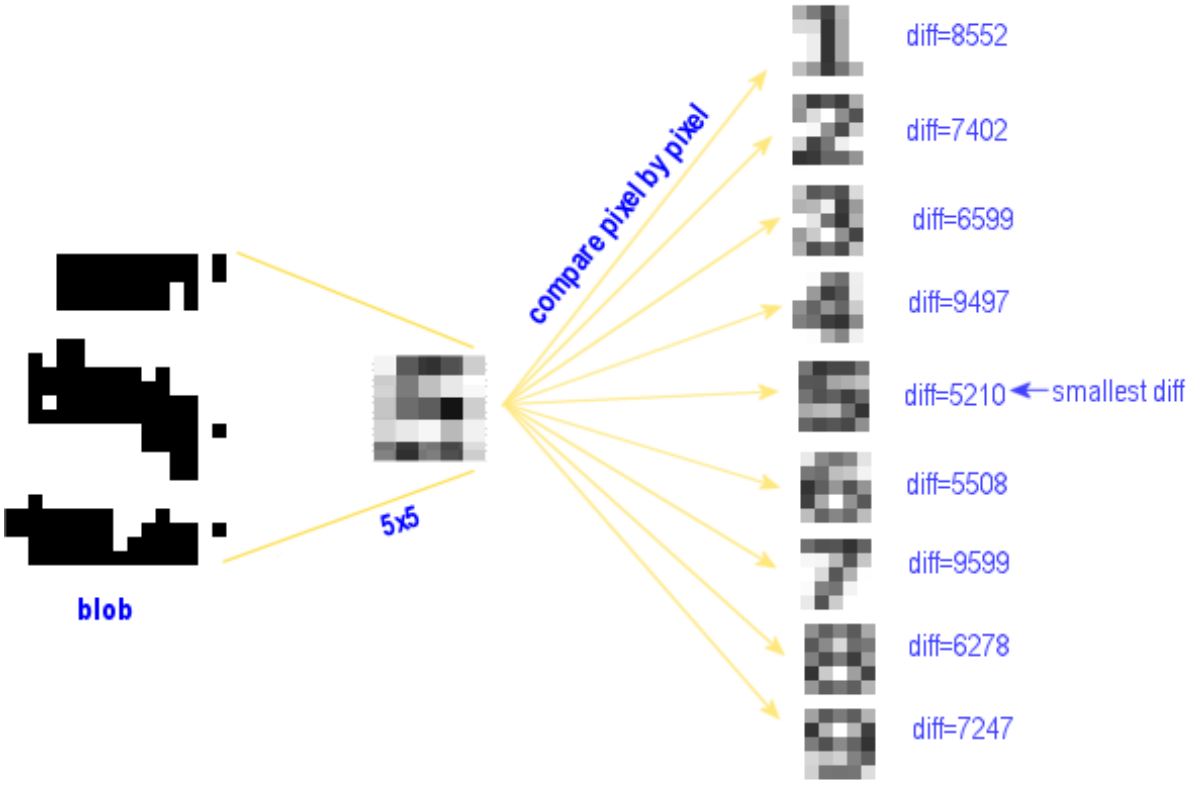

Рисунок 2.12 — Процесс уменьшения и сравнения с образцом

Этот метод инвариантен к размеру изображения, так как в любом случае мы используем зоны 5x5. Он чувствителен к поворотам, но мы уже позаботились об этом раньше. Проблема в том, что он чувствителен к позиции и смещению, а также не работает с

негативами (белым по чёрному) и перевёрнутыми вверх тормашками цифрами. Изображен на на рис. 2.12

2.2.4.2 Соотношение ширина/высота

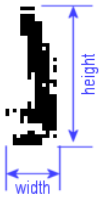

Рисунок 2.13 — Пример цифры 1

Цифра один на 2.14 является частным случаем . Так как она похожа на 4 и 7, то метод определения, данный выше, может оказаться ненадёжным. Специфичный параметр единицы: в нашем случае, её ширина на 40% меньше, чем высота. Нет другой такой цифры с таким же соотношением. У нас уже есть 25 зон (признаков), так вот это является 26-м признаком.

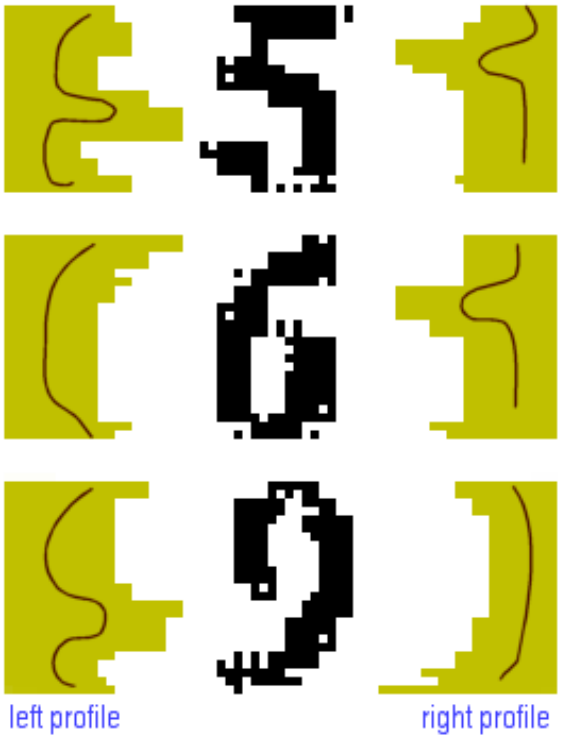

Рисунок 2.14 — Пример цифры 1

Качество OCR может быть улучшено введением новых признаков. Например, цифры 5,6 и 9 очень похожи при использовании разбиения на зоны. Для того, чтобы раз-

личать их, мы можем использовать их профили как признаки. Особенностью признака является количество пикселей (расстояние) между краем рамки цифры и её границей.

На изображении ниже, вы можете видеть, что профили цифр 5 и 6 похожи, но они отличаются от профиля 9. Левый профиль 5 и 9 похож, но отличается от профиля 6.

2.2.4.3 Корректировка результатов OCR

Как только OCR закончен, результаты корректируются в соответствии с правилами судоку.

В одной строке, столбце и блоке 3x3 не может находиться одна и та же цифра. Если правило нарушено, то необходимо ввести коррективы в полученные результаты. Мы заменим неправильную цифру той, что возможна из оставшихся.

Например, на рис. 2.12 выше, результат 5 потому что $diff = 5210$, которое является самым маленьким значением различия. Следующий возможный результат 6, так как $diff = 5508$ в этом случае. Таким образом, мы заменим 5 на 6. Чтобы решить, которая из двух цифр неверная, мы сравним значения их различий с их образцами, и та, у которой значение меньше, будет считаться верной.

2.2.5 Принцип работы системы

Область распознавания изображений является одной из самых захватывающих в современных компьютерных вычислениях, и она очень сложна. Что легко и просто для человеческого мозга, то очень сложно для системы. Многие вещи до сих пор остаются невозможными с сегодняшним уровнем развития IT.

На блок-схеме на рис. 2.12 представлены все этапы, которые проходит изображение перед тем, как будет распознано.

После распознавания мы сохраняем результаты в CSV-файл.

Для обмена данными между роботом и мобильным устройством будем использовать формат CSV.

CSV (от англ. Comma-Separated Values — значения, разделённые запятыми) — текстовый формат, предназначенный для представления табличных данных. Каждая строка файла — это одна строка таблицы. Значения отдельных колонок разделяются разделительным символом (delimiter) — запятой (,). Однако, большинство программ вольно трактует стандарт CSV и допускают использование иных символов в качестве разделителя.

Пример распознанного изображения в формате CSV:

Листинг 2.1 — CSV файл с заданием

```
1 5,3,,,7,,,,
2 6,,195,,
3 ,9,8,,,,6,
4 8,,,6,,,,3
```

Рисунок 2.15 — Блок-схема подсистемы распознавания

```
4 , , 8 , , 3 , , 1
7 , , , 2 , , , 6
 , 6 , , , , , 2 , 8 ,
 , , , 4 , 1 , 9 , , , 5
 , , , , 8 , , , 7 , 9
```

Нас данный вариант устраивает, так формат достаточно компактен для передачи и нам не нужно "захлямлять"информационным шумом потоки передачи данных.

2.3 Подсистема передачи данных

Нашей наиболее часто выполняемой задаче будет передача распознанного CSV-файла с мобильного телефона на контроллер робота.

В листинге 1.1 и листинге 1.2 приведены примеры обмена данными между роботом и телефоном, описанные в протоколе сообщений робота.

Процесс сопряжения мы можем произвести с помощью встроенных возможностей каждого из устройств, поэтому дублировать его не имеет смысла и воспользуемся стандартной процедурой сопряжения.

2.3.1 Обмен данными между устройствами

У нас есть список команд, описание протокола, сформированный файл, но проблема в том, что у нас лимит сообщения 64 байта.

Нам неудобно разрывать файл, управляющие команды и остальные сообщения на большое количество частиц.

Для этого мы воспользуемся **Bluetooth сокетами**.

Bluetooth сокет - API, позволяющий реализовывать взаимодействие между узлами или между процессами на одном узле. Данная технология может работать со множеством различных устройств ввода-вывода и драйверов, несмотря на то, что их поддержка зависит от реализации операционной системы.

С мобильного устройства у нас будут отправляться данные, а на роботе будут приниматься.

На телефоне мы создаем отправляющий сокет, приведенный в листинге 2.2

Листинг 2.2 — Создаем сокет с данными для отправки

```
1 Method m = device.getClass().getMethod("createRfcommSocket", new Class[] {
      int.class });
2 BluetoothSocket sendSocket = (BluetoothSocket) m.invoke(device, 1);
```

На роботе в свою очередь, для работы мы будем использовать тип **Message Write** и **Message Read**, который позволяет обмениваться произвольными сообщениями между устройствами в текстовом виде, а дальнейшая обработка ляжет уже на контроллер робота и второе устройство обмена.

Наша задача состоит в том, чтобы

а) открыть/создать сокет;

б) обменяться данными;

в) закрыть сокет (зафиксировав файл на конечном устройстве).

Выполнил эти три простые операции, мы гарантируем отправку и доставку данных. Причем целостность данных, как на уровне файлов, так и на уровне пакетов будут лежать уже на сокетах.

Сокеты поддерживаются обоими устройствами и API-робота дает нам возможность принимать данные.

2.3.2 Прием и отправка данных с мобильного устройства

Наше устройство под управлением ОС Android будет обмениваться данными с роботом.

Помимо файла с заданием мы будем получать от робота еще и некоторую диагностическую информацию.

Дело в том, что в Android'е для приема данных от какого-либо устройства необходимо создавать отдельный фоновый поток, чтобы у нас не зависало основное активити. Для этого мы задействуем thread и все данные будут приниматься в отдельном потоке.

Для работы с Bluetooth нам необходимо будем минимум 3 библиотеки, не считая Java-библиотек для работ с потоками:

Листинг 2.3 — Библиотеки для работы с Bluetooth на ОС Android

```
1 import android.bluetooth.BluetoothAdapter;
2 import android.bluetooth.BluetoothDevice;
3 import android.bluetooth.BluetoothSocket;
```

Первая библиотека в листинге 2.2, позволяет нам работать с нашим Bluetooth-адаптером, например проверить включен ли, получить адаптер через который будем работать и другие операции.

Вторая библиотека позволяет нам работать с устройствами Bluetooth, такие операции как получение удаленного устройства, получить операции с сокетами и другие.

Ну и третья библиотека для работы с сокетами, принцип работы которых был описан чуть выше.

Модуль на мобильном устройстве должен работать в двух режимах: должен отправлять задание и слушать порт, чтобы получать диагностическую информацию.

Диаграмма последовательности обмена сообщениями приведена на на рис 3.1

2.3.3 Прием и отправка данных с контроллера робота

Для работы с роботом нам предоставлено упрощенное API для работы с устройствами семейства Bluetooth.

У нас есть всего одно пространство имен:

Листинг 2.4 — Пространство для работы с Bluetooth на контроллерах NXT

```
1 javax.bluetooth
```

И всего 4 класса, которые предоставляют весь нам необходимый функционал:

Листинг 2.5 — Классы для работы с Bluetooth на контроллерах NXT

```
1 DeviceClass
2 DiscoveryAgent
3 LocalDevice
4 RemoteDevice
```

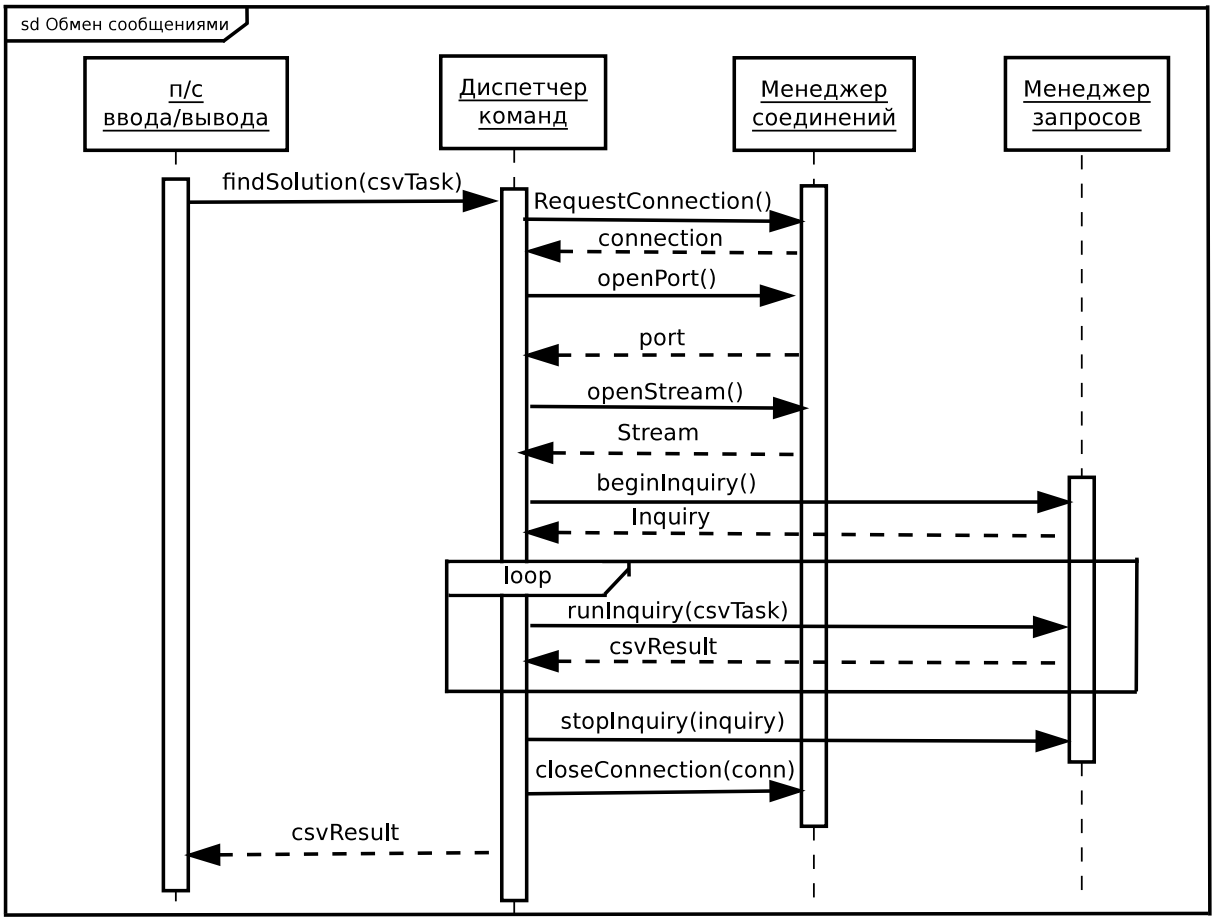

Рисунок 2.16 — Диаграмма последовательности обмена сообщениями

Первый класс предоставляем нам возможности стандартной спецификации Bluetooth и по большому мало применим.

Второй класс предоставляем нам весь набор команд и возможностей для обмена данными и прочей обработки, возвращается как парамет при вызове некоторых методов из **LocalDevice** и **RemoteDevice**. То есть этот класс набор инструкций.

И последние 2 класса нам нужны для работы с локальным устройством Bluetooth, в нашем случае с устройстом контроллера и удаленным устройством.

Наша задача состоит в том, чтобы RemoteDevice принял данные, отправленные через наш сокет и принялся их решать.

2.3.4 Обмен диагностической информацией

Чтобы мы могли узнать успешно ли прошла операция, сколько времени на нее затратилось, какое решение имеет то или иное задание, мы будем использовать диагностическую информацию приведенную в данном разделе.

Во-первых, чтобы не отправлять постоянной файлик с заданием, мы применим для него хеш-функцию MD5.

MD5 (англ. Message Digest 5) — 128-битный алгоритм хеширования.

Хеш содержит 128 бит (16 байт) и обычно представляется как последовательность из 32 шестнадцатеричных цифр. На 2.6,

Листинг 2.6 — Пример MD5 хеша

```
1 MD5("md5") = 1bc29b36f623ba82aaf6724fd3b16718
```

Мы будем брать MD5 хеш от задания, что не передавать каждый раз задание целиком.

Информация, которая нам необходимо собирать в процессе решения:

а) фотография, которая соответствует заданию;
б) само задание;
в) статус решено или нет;
г) время решения;
д) ответ.

Последние 3 пункта мы получаем с робота.

Для идентификации заданий, мы будем хешировать каждое задание и будем индексировать его по значению его хеша.

Схему базы данных можно посмотреть на рис 2.17

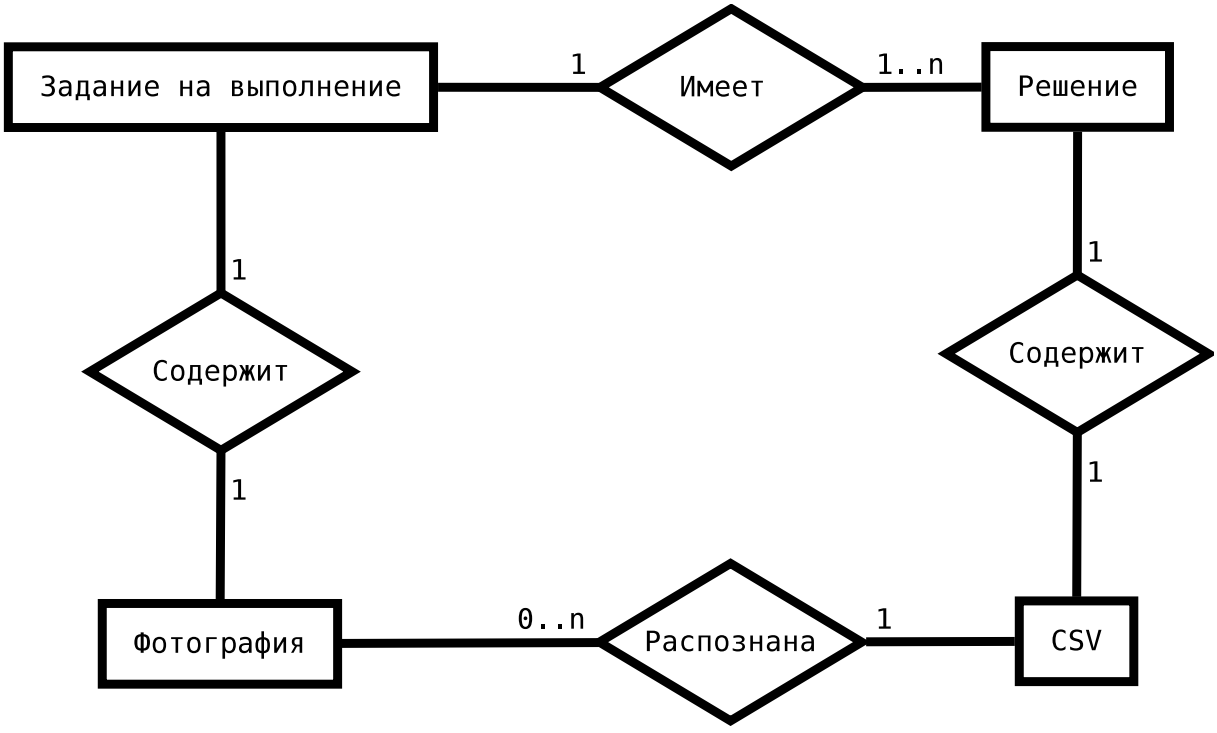

Рисунок 2.17 — ER диаграмма

У нескольких фотографий может быть один и тот же хеш, так же необходимо учесть, что у одного задания может быть несколько решений, при плохо спроектированном задании Судоку.

2.4 Подсистема решения

Так для подсистемы определили, что скорость решения одна из существенных характеристик для данной подсистемы, мы будем использовать комбинацию из

а) метода перебора;
б) метода открытых пар;
в) метода скрытых пар.

Перебор не слишком быстрый метод. Методы открытых и скрытых пар очень быстрые, но могут решать только быстрые пазлы. Так как мы просим приложение решить пазл за нас, то это значит, что он действительно сложный.

Наша подсистема должна решать любые пазлы, поэтому скомбинируем решение методов в такой последовательности:

а) метод открытых пар;
б) метод скрытых пар;
в) метод перебора.

В начале расположены два быстрых метода, и только если им не удаётся решить пазл, управление переходит к решению методом перебора, уменьшив работу для этого алгоритма.

На рис. 2.19 приведён алгоритм работы.

Рисунок 2.18 — Плохой случай для метода перебора

Только если методы открытых и скрытых пар не могут решить пазл, программа начинает передирать значения. И даже тогда, алгоритм перебора ограничен по времени 600000 итерациями. Также существует три попытки, после чего программа сдаётся. Между каждой попыткой, рекурсивная последовательность переставляется случайным образом с надеждой, что новая последовательность поможет решить пазл быстрее.

Перебор может быть очень медленным, если решение требует много итераций. Например, следующее изображение на рис. 2.18 это очень плохой случай для нашего алгоритма. Начиная с левой верхней клетки будет слишком много переборов, но есть и хорошие новости: начинайте с клетки, где вариантов поменьше. Это ускорит решение.

Есть ещё несколько оптимизаций для увеличения производительности алгоритма, таких как сортировка рекурсивной последовательности от клеток с наименьшим количеством кандидатов к клеткам с наибольшим. Но мы не используем этот алгоритм, так как это только частичная оптимизация.

Все судоку имеют плохие клетки, и обрабатываются не столь быстро для приложений в реальном времени.

Метод скрытых цифр, возможно, сможет решить весь пазл целиком, но лучше работает с открытых цифр. Когда в методе открытых цифр кончаются одиночные кандидаты, на сцену выходит метод скрытых цифр.

2.5 Подсистема вывода

2.5.1 Основные задачи

В данной секции опишем список задач, которые нам необходимо будет произвести для вывода ответа на задания манипулятором, который держит ручку.

а) определить ячейку, ее границы и размер ;
б) определить, есть ли в ячейке цифра из задания или нужно записать;
в) записать нужную цифру с учетом размера ячейки;
г) определить конец столбца и при необходимости начать заполнять новый.

2.5.2 Абстракция вывода символов

Наша задача сводится к тому, чтобы разработчику предоставить уровень абстракции, который позволит работать с символами.

Сейчас у нас все символы выводятся с помощью управляющих команд на моторы робота. Мы должны абстрагировать вывод символов и дать разработчику работать с уже обученным алфавитом символов.

Для этого нам необходимо:

а) подготовить алфавит для вывода,
б) предоставить интерфейс для взаимодействия на уровне абстракции символов.

2.5.3 Передвижение робота по полю

Для передвижения робота по полю нам предоставляются методы из API робота на вход которых мы подаем мощность.

Робот будет использовать 2 мотора для перемещения в горизонтальном пространстве, назовем из моторами B и C.

Чтобы двигаться относительно линий, нам на вход будут подаваться данные с сенсора света.

В аналитической части было сказано, что для определения границ и линий, мы будем использовать идеальную точку.

Формализуем задачу.

Рассчитать мощность вращения каждого из двигателей с учетом степени отклонения от идеальной точки.

Исходные данные:

а) идеальная точка,
б) текущие показания датчика освещенности.

Результат:

а) мощность вращения мотора B,
б) мощность вращения мотора C.

Решение.

Рассмотрим две ситуации:

а) робот отклонился от черной линии в сторону белого,
б) робот отклонился от белого в сторону черной линии.

В первом случае робот должен увеличить мощность вращение мотора B и уменьшить мощность мотора C.

Во второй ситуации, когда робот заезжает на черную линию, все наоборот.

Чем сильнее робот отклоняется от идеальной точки, тем быстрее ему надо к ней вернуться.

Математически это будет записано так:

$$H_b = H_{\text{базовая}} + (I_{\text{тек.}} - I_{\text{идеал.}}) * k \tag{2.2}$$

$$H_c = H_{\text{базовая}} - (I_{\text{тек.}} - I_{\text{идеал.}}) * k \tag{2.3}$$

где H_b и H_c — итоговые мощности моторов B и C соответственно, $H_{\text{базовая}}$ — базовая мощность моторов, определяющая скорость движения робота, $I_{\text{тек.}}$ — текущие показания датчика освещенности, $I_{\text{идеал.}}$ — рассчитанная идеальная точка, k —коэффициент пропорциональности.

2.5.4 Вывод символа

Перед выводом символа нужно проверить поставлена ли ручка на бумагу, например, при сканировании и получении информации ручка должна быть поднята, чтобы не испортить холст.

Нужно откалибровать положение манипулятора перед нанесением, то есть, чтобы символ выводился в клетке. Делается сканированием ячейки и перемещением манипулятора или положения робота.

После чего можно приступать к выводу.

Листинг 2.7 — Формирование алфавита

```
switch(digit)
  {
    case ONE:
      sequence="DDDDDDDDDDDDDD";
      startX=0;
      startY=6;
      break;
    case TWO:
      sequence="RRRRRRRRRRDDDDDDDLLLLLLLLLLLDDDDDDDRRRRRRRRRRR";
      startX=-4;
      startY=6;
      break;
    case THREE:
      sequence="RRRRRRRRRRDDDDDDDLLLLLLLLLRRRRRRRRRRDDDDDDDLLLLLLLLLL";
      startX=-4;
      startY=6;
      break;
    case FOUR:
      sequence="DDDDDDDDURRRRRRRRRLLLLUUUUUUUUDDDDDDDDDDDDDD";
      startX=-4;
      startY=6;
      break;
    case FIVE:
      sequence="LLLLLLLLLLRDDDDDDDURRRRRRRRDRRDDRDLLDDLLDLDLLLLLUULLUL";
      startX=4;
      startY=6;
      break;
    case SIX:
      sequence="LLLLLLLLLDDDDDDDDDDDDRRRRRRRRRRRRUUUUUUULLLLLLLLLL";
      startX=4;
      startY=6;
      break;
    case SEVEN:
      sequence="RRRRRRRRRRLLDLDLDDLDDLDDLDDLDDLDDLD";
      startX=-4;
```

```
36          startY=6;
37          break;
38      case EIGHT:
39          sequence="RRRRRRDDDDDDDLLLLLLLLRRRRRRRRRRDDDDDDLLLLLLUUUUUUUUUU";
40          startX=-3;
41          startY=6;
42          break;
43      case NINE:
44          sequence="LLLLLLLLLUUUUUUURRRRRRRRRRDDDDDDDDDDDDDDLLLLLLLLL";
45          startX=4;
46          startY=0;
47          break;
48    }
```

где $sequence$ — управляющие команды на двигатель манипулятора , $startX$ и $startY$ — начальное положение манипулятора в клетке.

Как только поступила команда печати символа:

а) опускается ручка,

б) берется нужная последовательность команд для манипулятора,

в) подается на вход двигателю манипулятора.

2.5.5 Получение данных с сенсора цвета

Сенсор цвета закреплен на манипуляторе, там же, где закреплена ручка.

Команды для передвижения сенсора будут такие же, как и для перемещения ручки, но будет обходимости делать сложные движения, как при рисовании.

Данные сенсор получает так же как и фотокамера на смартфоне, но при этом у нас не тратиться время на

а) конвертацию в черно-белый цвет,

б) выравнивание картинка,

в) распознавание.

На распознавание мы не тратим время, т.к. у нас задание распознано уже, нам необходимо понять есть ли "последовательность идеальных точек в клетке".

Конвертация нам не нужна, так мы и так получаем картинку в монохромном режим, мы берем лишь входной параметр $I_{\text{тек.}}$ — текущего показания датчика освещенности.

И выравнивание у нас отсутствует, так картинка задание лежит горизонтально и сенсор света смотрит на него перпендикулярно.

На выходе мы получаем интерфейс:

Листинг 2.8 — Интерфейс для вывода цифры

```
1  void writeDigit(int digit)
```

И у нас пропадает необходимость использовать какие-либо машинные команды для печати символа.

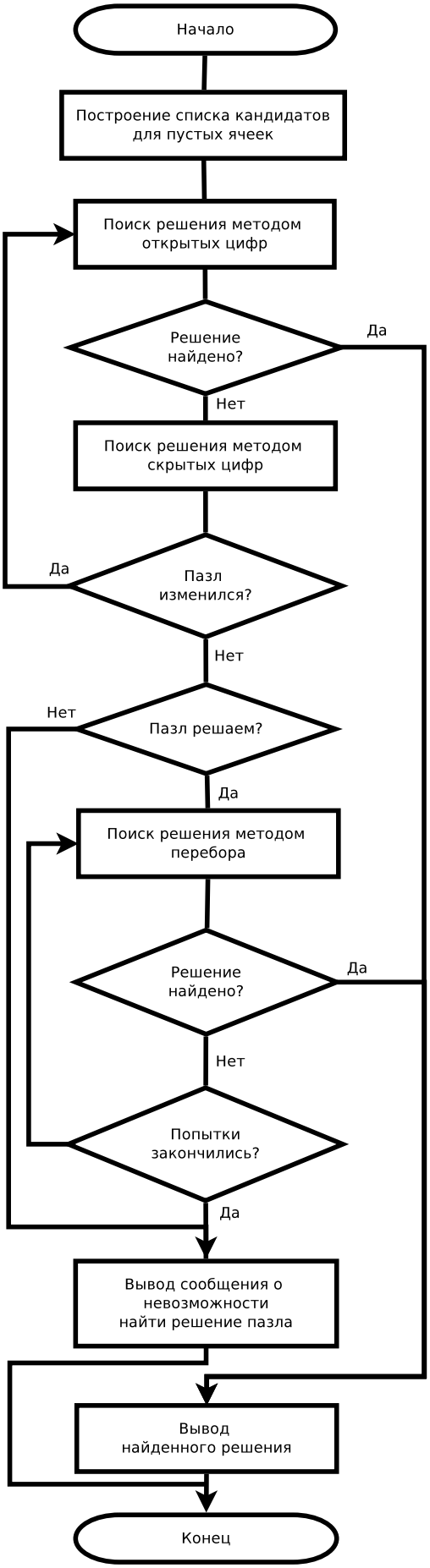

Рисунок 2.19 — Блок-схема алгоритма решения

3 Технологический раздел

В данном разделе описываются используемые при работе над проектом инструменты и технологии.

3.1 Выбор языка программирования

Для реализации системы выбран язык Java. Перечислим основные преимущества, говорящие в пользу его использования. Java является объектно-ориентированным языком. Это дает возможность при разработке приложений использовать технологию объектно-ориентированного программирования, которая позволяет сократить общее время разработки и писать повторно используемый код.

Java-приложения являются независимыми от платформы. Это достигается путем совмещения в языке свойств компилятора и интерпретатора следующим образом: классы программы компилируются во внутренний байт-код, который может интерпретирован виртуальной java-машиной (JVM, Java Virtual Machine).

Платформонезависимость байт-кода обеспечивается наличием виртуальных java-машин для всех основных платформ.

В комплект поставки Java (JDK, Java Developer Kit) входят стандартные классы, которые обладают достаточной функциональностью для быстрой разработки приложений.

Важным преимуществом является наличие большого количества библиотек, расширяющих возможности языка.

3.2 Используемые библиотеки и модули

3.2.1 Мобильное приложение

Android — операционная система для смартфонов, планшетных компьютеров, электронных книг, цифровых проигрывателей, наручных часов, игровых приставок, нетбуков, смартбуков, других устройств. Основана на ядре Linux и собственной реализации Java от Google.

Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit позволяет портировать библиотеки и компоненты приложений.

3.2.2 Android SDK

Приложения под операционную систему Android являются программами в нестандартном байт-коде для виртуальной машины Dalvik, для них был разработан формат установочных пакетов .APK.

Google предлагает для свободного скачивания инструментарий для разработки (Software Development Kit), который предназначен для x86-машин под операционными системами Linux, Mac OS X (10.4.8 или выше), Windows XP, Windows Vista и Windows 7. Для разработки требуется JDK 5 или более новый.

Разработку приложений для Android можно вести на языке Java (не ниже Java 1.5).

Существует плагин для Eclipse — Android Development Tools (ADT), предназначенный для Eclipse версий 3.3—3.7. Также существует плагин для IntelliJ IDEA, облегчающий разработку Android-приложений, и для среды разработки NetBeans IDE, который, начиная с версии NetBeans 7.0, перестал быть экспериментальным, хоть пока и не является официальным.

Кроме того, существует Motodev Studio for Android — комплексная среда разработки на базе Eclipse, позволяющая работать непосредственно с Google SDK.

В целом, **Andoroid SDK** предоставляет нам все необходимые средства для реализации заявленных задач.

3.2.3 ПО для контроллера робота

leJOS является заменой прошивки для программируемых блоков Lego Mindstorms.

В настоящее время она поддерживает LEGO RCX блок и Lejos NXJ поддерживает блок NXT. Она включает в себя виртуальную машину Java, что позволяет Lego Mindstorms роботам писать программное обеспечение на языке программирования Java.

Так как Lejos NXJ является проектом Java, она основывается на богатстве функциональных возможностей, присущей платформе Java. Существуют Lejos NXJ плагины для двух ведущих IDE для Java: Eclipse и NetBeans. Разработчики могут воспользоваться стандартной функциональности в IDE (автозавершения кода, рефакторинга и шаблонов для тестирования), а также такие возможности как: компиляция, сборка и выгрузки.

Lejos NXJ обеспечивает поддержку доступа к портам робота. Это позволяет получить доступ к стандартным датчикам и двигателям (ультразвуковой датчик расстояния, сенсорный датчик, звук и датчик света). Другие компании, такие как MindSensors и HiTechnic распространили этот базовый набор, предоставляя усовершенствованные датчики, исполнительные механизмы и мультиплексоры. Lejos NXJ включает Java API, для этих продуктов.

Воспользовавшись объектно-ориентированной структурой языка Java, разработчики Lejos NXJ смогли скрыть детали реализации датчиков. Это позволяет разработчику работать с абстракциями высокого уровня, не беспокоясь о деталях, например таких как шестнадцатеричная адресация аппаратных компонентов.

Проект включает в себя реализацию наиболее часто используемых контроллером обратной связи, ПИД-регулятора и алгоритма снижения фильтр шума Калмана. Lejos NXJ

также предоставляет библиотеки, которые поддерживают более абстрактные функции, такие как навигации, картографирования и поведения.

Листинг 3.1 — Пример кода для работы с двигателями

```
1  import lejos.nxt.Motor;
2  import lejos.nxt.Button;
3
4  public class Example {
5      public static void main(String[] args) {
6          Motor.A.forward();
7          Button.waitForPress();
8          Motor.A.backward();
9          Button.waitForPress();
10         System.exit(1);
11     }
12 }
```

Таким образом, мы можем писать на удобном нам языке (в данном случае Java) и работать с подключаемыми устройствами робота посредством API, просто делая импорт нужной библиотеки, которая отвечает за управляющие команды или данные сенсоров.

3.3 Система контроля версий

3.3.1 Git

В качестве системы контроля версий была выбрана CVS Git. Данный инструмент является очень популярным и удобным в работе. Не требует установки сервера при одиночной разработке.

Работа с подмодулями в данной системе упрощает разработку и интеграцию нескольких проектов параллельно, что является большим плюсом в работе, поскольку каждый компонент системы был оформлен как отдельный репозиторий git.

3.4 Среда разработки

Выбор описанных выше инструментов привел к естественному выбору IDE, поддерживающей как язык Java и работу с его зависимостями и библиотеками, так и обладающей интеграцией с CVS Git. Такой средой стала программа IntelliJ IDEA компании JetBrains.

3.4.1 IntelliJ IDEA

Данная среда разработки создана специально для разработки на языке Java и имеет отличные инструменты для отладки программ, запуска модульных и функциональных тестов с возможностью отображения покрытия кода тестами, эмулятор командой строки для быстрого запуска консольных команд, систему управления библиотеками Java, а так

же поддержка всевозможных Java фреймворков и библиотек и всех его основных функций.

Подсветка синтаксиса, автодополнение, инспекция кода и другие функции облегающие жизнь разработчика так же присутствуют в данной IDE, а прекрасное выполнения функции поиска классов и файлов проекта экономит значительное количество времени.

Так же кроме написания ПО на языке Java, IntelliJ IDEA поддерживает работу с другими языками, документами HTML, CSS-стилями и JavaScript-скриптами. Поддержка манифест-файлов для разного рода приложений, умение понимать, анлизировать и определять вставки исходного кода на другом языке облегчают процесс разработки.

Таким образом данная программа является идеальным решением для разработки на языке Java и в частности Java, Android и других производных приложений.

3.5 База данных

3.5.1 SQLite

SQLite - компактная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние.

Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер, то есть СУБД SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и СУБД становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу.

SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.

В комплекте поставки идёт также функциональная клиентская часть в виде исполняемого файла sqlite3, с помощью которого демонстрируется реализация функций основной библиотеки. Клиентская часть работает из командной строки, позволяет обращаться к файлу БД на основе типовых функций ОС.

Благодаря архитектуре СУБД возможно использовать SQLite как на встраиваемых системах, так и на выделенных машинах с гигабайтными массивами данных.

Таким образом SQLite идеально подходит для реляционного хранения данных на мобильном устройстве под управлением ОС Android схема которой представлена на рис 3.1

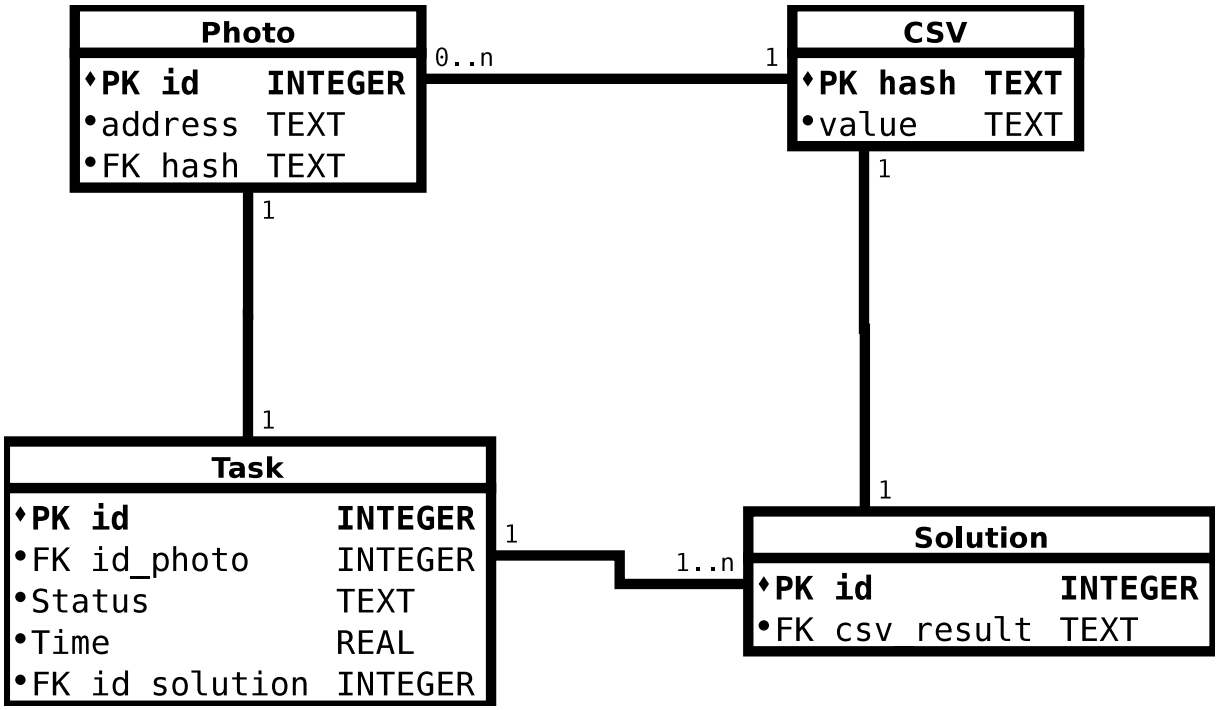

Рисунок 3.1 — ER диаграмма

4 Экспериментальный раздел

В данном разделе рассматриваются результаты скорости работы подсистемы распознования данных, решения, вывода.

Эксперименты проводились на мобильном устройстве Google Nexus 4:

— процессор: Qualcomm APQ8064, 1500 МГц, 4 ядра;
— оперативная память: 2GB;
— фронтальная камера: 8 МПикс;
— ОС: Android 4.4.

Робот LEGO Mindstorm NXT 2 имеет следующие характеристики:

— основной процессор: Atmel ARM7 AT91SAM7S256, частота 48 МГц, 32-битный;
— сопроцессор: Atmel AVR ATmega48, частота 8 МГц, 8-битный;
— память: 16Кб ППЗУ, 512 байт ОЗУ процессора и 32Кб внешнего ОЗУ;
— 256 Кб FLASH, 64 Кб ОЗУ основного процессора, и 4Кб FLASH, 8 Кб ОЗУ сопроцессора.

4.1 Анализ результатов распознавания

В качестве примера для работы будет рассматривать время распознавания картинки с заданием.

Входными данными для задачи являются:

— фотография с заданием.

В качестве результата выполнения эксперимента получим время работы затраченное на выполнение каждого этапа из:

а) захват фото,
б) конвертация в черно-белый цвет,
в) определение угла поворота,
г) определение сетки,
д) распознавание цифр,
е) корректировка распознавания.

Результаты времени распознавания представлены в табл. 4.1

Самые длительными этапами являются процессы конвертации и определения сетки.

Операция конвертации является длительной из-за того, что пороги считаются для каждого пикселя, ведь алгоритм требует огромного количества чтений пикселя изображения и подсчета для области. И даже целочисленная форма, которая помогает ускорить этот алгоритм, все-равно оставляет эту операцию длительной.

Таблица 4.1 — Результаты времени распознавания

Операция	Время, мс
1) захват фото	2
2) конвертация в черно-белый цвет,	13
3) определение угла поворота,	1
4) определение сетки	13
5) распознавание цифр	7
6) корректировка распознавания	1

Определение сетки длительный алгоритм потому, что для каждого пикселя проводится 180 воображаемых линий с шагом в 1 градус, которые потом уйдут на «голосование».

В остальном обработка большого зашумленного изображения на смартфоне за 37 мс это хороший результат.

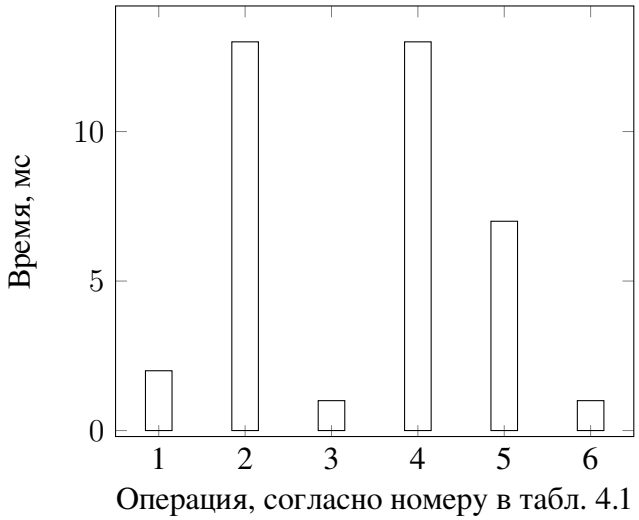

Рисунок 4.1 — Гистограмма времени операций

4.2 Анализ результатов решения

Сложность алгоритма определяется глубиной рекурсии для каждого кандидата.

При правильно поставленном задании не должно быть большое количество рекурсий, т.к. по правилам Судоку имеется единственное верное решение.

Но в реальной жизни под сложными пазлами подаются задачи, которые имееют большое ветвление и в реальности могут иметь несколько решений, эти задачи нам тоже решать.

Простой пазл - пазл который имеет для первого разбора не более 3 кандидатов.

Обычный пазл - пазл который имеет для первого разбора от 4 до 6 кандидатов.

Сложный пазл - пазл который имеет для первого более 6 кандидатов.

Сложности, оценки эффективности методов приведены из статьи [3] и [4].

Результаты решения представлены в табл. 4.2

Таблица 4.2 — Время решения пазлов

Сложность пазла	Время, мс
простой пазл	1
простой пазл	2
сложный пазл	18

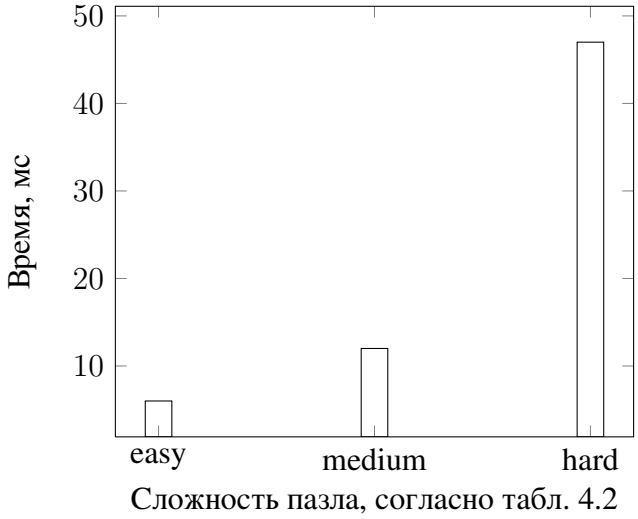

Рисунок 4.2 — Гистограмма времени операций

В начале расположены два быстрых метода, и только если им не удаётся решить пазл, управление переходит к решению методом перебора.

Только если методы открытых и скрытых пар не могут решить пазл, программа начинает перебирать значения. Также существует три попытки, после чего программа завершается. Между каждой попыткой, рекурсивная последовательность переставляется случайным образом с надеждой, что новая последовательность поможет решить пазл быстрее.

Если сложность пазла не очень высока и не доходит до метода перебора, то пазлы решаются в 2 или 8 раз быстрее.

Решение для легких пазлов и средних превышают всего на 2 мс на роботе с не очень мощным процессором и медленным ОЗУ решение такой же задачи на ПК, с характеристиками в десятки раз превышающими характеристики робота.

Но мы не можем отказаться от алгоритма перебора, т.к. он гарантировано приведет к решению.

4.3 Время результатов вывода решения

Вывод одного символа на лист бумаги занимает в среднем 4.8 сек при выводе решения на бумагу формата А4.

Для заполнения всей таблицы нам необходимо около 6 минут.

Столь длительный вывод объясняется тем, что управляющие двигатели манипулятора не позволяют работать с большей мощностью, иначе изображение получается не четким.

При выборе большего формата бумаги, можно увеличить мощность двигателей, но будет затрачиваться большего на рисование более крупного символа.

5 Организационно-экономический раздел

Организационно-экономическая часть процесса разработки программного продукта предусматривает выполнение следующих работ:

— формирование состава выполняемых работ и группировка их по стадиям разработки;
— расчет трудоемкости выполнения работ;
— установление профессионального состава и расчет количества исполнителей;
— определение продолжительности выполнения отдельных этапов разработки;
— построение календарного графика выполнения разработки;
— контроль выполнения календарного графика.

5.1 Формирование состава выполняемых работ и группировка их по стадиям разработки

Разработку программного продукта можно разделить на следующие стадии:

— техническое задание;
— расчет трудоемкости выполнения работ;
— эскизный проект;
— технический проект;
— рабочий проект;
— внедрение.

Допускается объединение технического и рабочего проекта в технорабочий проект.

Планирование длительности этапов и содержания проекта осуществляется в соответствии с ЕСПД ГОСТ 34.603–92 и распределяет работы по этапам. как показано в таблице 5.1.

Таблица 5.1 — Распределение работ проекта по этапам

Основные стадии	№	Содержание работы
1. Техническое задание	1	Постановка задачи
	2	Выбор средств проектирования и разработки
2. Эскизный проект	3	Разработка структуры системы
	4	Разработка алгоритмов описания моделей и моделирования
	5	Разработка вспомогательных алгоритмов
	6	Разработка пользовательского интерфейса
2. Технорабочий проект	7	Реализация алгоритмов описания моделей и моделирования
	8	Реализация вспомогательных алгоритмов
	9	Реализация пользовательского интерфейса
	10	Отладка программного продукта
	11	Исправление ошибок и недочетов
	12	Разработка документации к системе
	13	Итоговое тестирование системы
4. Внедрение	14	Установка и настройка программного продукта

5.2 Расчет трудоемкость выполнения работ

Трудоемкость разработки программной продукции заывисит от ряда факторов, основными из которых являются следующие:

— степень новизны разрабатываемого программного продукта;

— сложность алгоритма его функционирования;

— объем используемой информации, вид ее представления и способ обработки;

— уровень используемого алгоритмического языка программирования.

Разрабатываемый программный продукт можно отнести:

— по степени новизны — к категории В. Разрботка программной продукции имеющей аналоги.

— по степени сложности алгоритма функционирования — к 1-ой группе (программная продукиця реализующая моделирующие алгоритмы).

Трудоемкость разработки программного продукта $\tau_{\text{ПП}}$ может быть определена как сумма величин трудоемкости выполнения отдельных стадий разработки ПП из выражения 5.1.

$$\tau_{\text{ПП}} = \tau_{\text{ТЗ}} + \tau_{\text{ЭП}} + \tau_{\text{ТП}} + \tau_{\text{РП}} + \tau_{\text{В}} \tag{5.1}$$

, где

$\tau_{\text{ТЗ}}$ — трудоемкость разработки технического задания; $\tau_{\text{ЭП}}$ — трудоемкость разработки эскизного проекта; $\tau_{\text{ТП}}$ — трудоемкость разработки технического проекта; $\tau_{\text{РП}}$ — трудоемкость разработки рабочего проекта; $\tau_{\text{В}}$ — трудоемкость внедрения.

Трудоемкость разработки технического задания рассчитывается по формуле 5.2

$$\tau_{\text{ТЗ}} = T_{\text{ЗРЗ}} + T_{\text{ЗРП}} \tag{5.2}$$

, где

$T_{\text{ЗРЗ}}$ — затраты времени разработчика постановки задач на разработку ТЗ, чел.-дни; $T_{\text{ЗРП}}$ — затраты времени разработчика ПО на раззработку ТЗ, чел.-дни.

Значения величин $T_{\text{ЗРЗ}}$ и $T_{\text{ЗРЗ}}$ рассчитываются по формулам 5.3 и 5.4.

$$T_{\text{ЗРЗ}} = t_{\text{З}} \cdot K_{\text{ЗРЗ}} \tag{5.3}$$

$$T_{\text{ЗРП}} = t_{\text{З}} \cdot K_{\text{ЗРП}} \tag{5.4}$$

, где

$t_{\text{З}}$ — норма времени на разработку ТЗ на ПП в зависимости от его функционального назначенияя и стпени новизны, чел.-дни; $K_{\text{ЗРЗ}}$ — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчикм ТЗ; $K_{\text{ЗРЗ}}$ — коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разраьотчиком ПО на стадии ТЗ.

$t_{\text{З}} = 24$ чел.-дн. (управление НИР)

$K_{\text{ЗРЗ}} = 0.65$ (совместная разработка)

$K_{\text{ЗРП}} = 0.35$ (совместная разработка)

$\tau_{\text{ТЗ}} = 24 \cdot 0.65 + 24 \cdot 0.35 = 24$ чел.-дн.

Аналогично рассчитывается трудоемкость эскизного проекта $\tau_{\text{ЭП}}$:

$\tau_{\text{ЭП}} = 70 \cdot 0.5 + 70 \cdot 0.5 = 70$ чел.-дн.

Трудоемкость разработки технического проекта $\tau_{\text{ТП}}$ зависит от функционального назначения ПП, количества разновидностей форм входной и выходной информации и определяется как сумма времени, затраченного разрабьотчикм постановки задач и разработчиком программного обеспечения по формуле 5.5.

$$\tau_{\text{ТП}} = (t_{\text{ТРЗ}} + t_{\text{ТРП}}) \cdot K_{\text{В}} \cdot K_{\text{Р}} \tag{5.5}$$

, где

$t_{\text{ТРЗ}}$ и $t_{\text{ТРП}}$ — норма времени, затрачивваемого на разработку ТП разрабьотчиком постановки задач и разработчиком программного обуспечения соответственно, чел.-дни; $K_{\text{В}}$ — коэффициент учета вида используемой информации, $K_{\text{Р}}$ — коэффициент учета режима обработки информации.

Значение коэффициента $K_{\text{В}}$ определяется из выражения:

$$K_{\text{В}} = \frac{K_{\text{П}} \cdot n_{\text{П}} + K_{\text{НС}} \cdot n_{\text{НС}+K_{\text{Б}} \cdot n_{\text{Б}}}}{n_{\text{П}} + n_{\text{НС}} + n_{\text{Б}}} \tag{5.6}$$

где $K_{\text{П}}$, $K_{\text{НС}}$, $K_{\text{Б}}$ — значения коэффициентов учета вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно; $n_{\text{П}}$, $n_{\text{НС}}$, $n_{\text{Б}}$ — количество наборов данных переменной, нормативно-справочной информации и баз данных соответственно.

$K_{\text{П}} = 1$, $K_{\text{НС}}$ - 0.72, $K_{\text{Б=2.08}}$

$K_{\text{В}} = \frac{1 \cdot 3 + 0.72 \cdot 1 + 2.08 \cdot 0}{3+1+0} = 0.505$

$K_{\text{Р}} = 1.26$

$\tau_{\text{ТП}} = (33 + 10) \cdot 0.505 \cdot 1.26 = 28$, чел.-дн.

Трудоемкость разработки рабочего проекта $\tau_{\text{РП}}$ зависит от функционального назначения ПП, количества разновидностей форм входной информации, сложности алгоритма функционирования, сложности контроля информации, степени использования готовых программных модулей, уровня алгоритмического языка программирования и определяется по формуле:

$$\tau_{\text{РП}} = K_{\text{К}} \cdot K_{\text{Р}} \cdot K_{\text{Я}} \cdot K_{\text{З}} \cdot K_{\text{ИА}} \cdot (t_{\text{РРЗ}} + t_{\text{РРП}}) \tag{5.7}$$

где $K_{\text{К}}$ — коэффициент учета сложности контроля информации; $K_{\text{Я}}$ — коэффициент учета уровня используемого алгоритмического языка программирования; $K_{\text{З}}$ — коэффициент учета степени использования готовых программных модулей; $K_{\text{ИА}}$ — коэффициент учета вида используемой информации и сложности алгоритма ПП.

$K_{\text{К}} = 1, K_{\text{Р}} = 1.44, K_{\text{Я}} = 1, K_{\text{З}} = 0.7, t_{\text{РРЗ}} = 9$ чел.-дн., $t_{\text{РРП}} = 54$ чел.-дн., $K_{\text{П}} = 1.2, K_{\text{НС}} = 0.65, K_{\text{Б}} = 0.54$

$K_{\text{ИА}} = \frac{1.2 \cdot 3 + 0.65 \cdot 1 + 0.54 \cdot 0}{3+1+0} = 1.06$

$\tau_{\text{РП}} = 1 \cdot 1.44 \cdot 1 \cdot 0.7 \cdot 1.06 \cdot (9 + 54) = 67$

Так как при разработке ПП стадии «Технический проект» и «Рабочий проект» объеденины в стадию «Техно-рабочий проект», то трудоемокость ее выполнения $\tau_{\text{ТРП}}$ определяется по формуле:

$$\tau_{\text{ТРП}} = 0.85 \cdot (\tau_{\text{ТП}} + \tau_{\text{РП}}) \tag{5.8}$$

$\tau_{\text{ТРП}} = 0.85 \cdot (28 + 67) = 91$

Трудоемкость выполнения стадии внедрения $\tau_{\text{В}}$ может быть расчитана по формуле:

$$\tau_{\text{В}=(t_{\text{ВРЗ}}+t_{\text{ВРП}}) \cdot K_{\text{К}} \cdot K_{\text{Р}} \cdot K_{\text{З}}} \tag{5.9}$$

где $t_{\text{ВРЗ}}$, $t_{\text{ВРП}}$ — норма времени, затрачиваемого разработчиком постановки задач и разработчиком ПО соответственно на выполнение процедур внедрения ПП, чел.-дни.

$\tau_{\text{В}} = (10 + 11) \cdot 1 \cdot 1.26 \cdot 0.7 = 19$ чел.-дн.

Подставив полученные данные в формулу 5.1 получим:

$\tau_{\text{ПП}} = 24 + 70 + 91 + 19 = 204$ чел.-дн.

Таблица 5.2 — Распределение трудоемкости по стадиям разработки проекта

Этап	Трудоемкость этапа, чел.-дн.	№	Содержание работы	Трудоемкость, чел.-дн.
1	24	1	Постановка задачи, разработка ТЗ	20
		2	Выбор средств проектирования и разработки	4
2	70	3	Разработка структуры системы	15
		4	Разработка алгоритмов описания модели и моделирования	30
		5	Разработка вспомогательных алгоритмов	15
		6	Разработка пользовательского интерфейса	10
3	91	7	Реализация алгоритмов описания модели и моделирования	23
		8	Реализация вспомогательных алгоритмов	16
		9	Реализация пользовательского интерфейса	10
		10	Отладка программного продукта	12
		11	Исправление ошибок и недочетов	15
		12	Разработка документации к системе	8
		13	Итоговое тестирование системы	7
4	19	14	установка и настройка ПП	19
			Итого:	204

5.3 Расчет количества исполнителей

Средняя численность исполнителей при реализации проекта разработки и внедрения ПО определяется соотношением:

$$N = \frac{Q_{\text{Р}}}{F} \tag{5.10}$$

где $Q_{\text{Р}}$ — затраты труда на выполнение проекта (разработка и внедрение ПО); F — фонд рабочего времени.

Величина фонда рабочего времени определяется соотношением:

$$F = T \cdot F_M \tag{5.11}$$

где T — фвремя выполнения проекта в месяцах, равное 4 месяцам; F_M — фонд времени в текущем месяце, который рассчитывается из учета числа дней в году, числа выходных и праздничных дней:

$$F_M = \frac{t_{\text{р}} \cdot (D_{\text{К}} - D_{\text{В}} - D_{\text{П}})}{12} \tag{5.12}$$

где $t_{\text{р}}$ — продолжительность рабочего дня; $D_{\text{К}}$ — общее число дней в году; $D_{\text{В}}$ — число выходных дней в году; $D_{\text{П}}$ — число праздничных дней в году.

$F_M = \frac{8 \cdot (365 - 103 - 10)}{12} = 168$

$F = 4 \cdot 168 = 672$

$N = \frac{204 \cdot 8}{672} = 3$ — число исполнителей проекта.

5.4 Календарный план-график

Таблица 5.3 — Планирование разработки

Стадия разработки	Трудоемкость	Должность исполнителя	Распределение трудоемкости	Численность
Техническое задание	24	Ведущий программист	18(75%)	1
		Программист 1	6(25%)	1
Эскизный проект	70	Ведущий программист	26(37%)	1
		Программист 1	27(39%)	1
		Программист 2	17(24%)	1
Технорабочий проект	91	Ведущий программист	36(40%)	1
		Программист 1	32(35%)	1
		Программист 2	23(25%)	1
Внедрение	19	Ведущий программист	8(42%)	1
		Программист 2	11(58%)	1

Таблица 5.4 — Календарный ленточный график работ

Стадия разработки	Должность исполнителя	Трудоемкость			
Техническое задание	Ведущий программист	18			
	Программист 1	6			
Эскизный проект	Ведущий программист		26		
	Программист 1		27		
	Программист 2		7	7	
Технорабочий проект	Ведущий программист			36	
	Программист 1			32	
	Программист 2				23
Внедрение	Ведущий программист				9
	Программист 1				10

Из таблицы 5.4 видно, что благодаря параллельной работе ведущего программиста и программистов можно добиться сокращения сроков разработки с 204 дней до $18 + 27 + 36 + 11 = 92$ дней, т.е. в 2.2 раза.

5.5 Расчет затрат на разработку ПП

Затраты на выполнение проекта сосстоят из затрат на заработную плату исполнителям, затрат на закупку или аренду оборудования, затрат на организацию рабочих мест и затрат на накладные расходы. В таблице 5.5 приведены затраты на заработную плату и страховые взносы во внебюджетные фонды. Суммарно эти отчисления для организаций, осуществляющих деятельность в области информационных технологий, составляют

14%: в Пенсионный фонд (ПФ) — 8%, в Фонд социального стразования (ФСС), Федеральный и территориальный фонды обязательного медицинского страхования (ФФОМС и ТФОМС) — по 2%. Для ведущего программиста предполагается ставка 2500 рублей за полный рабочий день, для первого программиста — 1500 рублей, для второго — 1000 рублей.

Таблица 5.5 — Затраты на зарплату и отчисления во внебюджетные фонды

	Февраль					
Исполниель	Рабочих дней	Зарплата	ПФ	ФСС	ФФОМС	ТФОМС
Ведущий программист	20	50000	4000	1000	1000	1000
Программист 1	20	30000	2400	600	600	600
Программист 2	19	19000	1520	380	380	380
Итого:						112860
	Март					
Исполниель	Рабочих дней	Зарплата	ПФ	ФСС	ФФОМС	ТФОМС
Ведущий программист	22	55000	4400	1100	1100	1100
Программист 1	22	33000	2640	660	660	660
Программист 2	6	6000	480	120	120	120
Итого:						107160
	Апрель					
Исполниель	Рабочих дней	Зарплата	ПФ	ФСС	ФФОМС	ТФОМС
Ведущий программист	21	52500	4200	1050	1050	1050
Программист 1	21	31500	2520	630	630	630
Программист 2	10	10000	800	200	200	200
Итого:						107160
	Май					
Исполниель	Рабочих дней	Зарплата	ПФ	ФСС	ФФОМС	ТФОМС
Ведущий программист	20	50000	4000	1000	1000	1000
Программист 1	20	30000	2400	600	600	600
Программист 2	19	19000	1520	380	380	380
Итого:						112860
	Июнь					
Исполниель	Рабочих дней	Зарплата	ПФ	ФСС	ФФОМС	ТФОМС
Ведущий программист	16	40000	3200	800	800	800
Программист 1	15	22500	1800	450	450	450
Программист 2	3	3000	240	60	60	60
Итого:						74670
Общая сумма:						446310

Расходы на материалы, необходимые для разработки ПП указаны в таблице 5.6.

При разработке понадобится 3 компьютера (по одному на человека), один принтер, компьютерная мебель (3 комплекта), аксесуары для компьютеров (3 компьютера). Стоимость одной ПЭВМ составляет 35000 рублей. Месячная норма амортизации $K = 4\%$ (Срок службы — 25 месяцев). Срок службы составляет 3 года, месячная амортизация

Таблица 5.6 — Материальные затраты

№	Наименование материала	Единица измерения	Кол-во	Цена за единицу, руб.	Сумма, руб.
1	Бумага А4	Пачка 500 л.	1	150	150
2	Картридж для лазерного принтера Brother HL-2030R	Шт.	2	1400	2800
				Итого:	2950

$K = 2.78\%$. Срок службы компьютерной мебели: столы — 5 лет, стулья — 2 года, месячные нормы амортизации — $K = 1.67\%$ и $K = 4.17\%$ соответственно. компьютерные аксесуары имеют срок службы 2 года, норма амортизации $K = 4.17\%$.

Затраты на амортизацию приведены в таблице 5.7

Таблица 5.7 — Амортизационные отчисления

Наименование оборудования	Балансовая цена, руб.	Кол-во, шт.	К, %	Время использования, мес.	Сумма отчислений, руб.
Компьютер	35000	3	4	4	16800
Принтер	2600	1	2.78	4	290
Стол	3000	3	1.67	4	600
Стул	1000	3	4.17	4	500
Клавиатура	500	3	4.17	4	350
Мышь	500	3	4.17	4	350
				Итого:	18790

Таке необходимо учесть арендную полату за помещение, которая составляет 14000 рублей за месяц. За 4 месяца на аренду помещения уйдет 56000 рублей.

Общие затраты на разработку ПП составят $446310 + 2950 + 18790 + 56000 = 524050$ рублей.

5.6 Расчет экономической эффективности

Основными показателями экономической эффективности является чистый дисконтированный доход (ЧДД) и сок окупаемости вложенных средств.

Чистый дисконтированный доход определяется по формуле:

$$\text{ЧДД} = \sum_{t=0}^{T} R_t - \text{З}_t \cdot \frac{1}{(1+E)^t} \tag{5.13}$$

где T — горизонт рачета по месяцам; t — период расчета; R_t — результат достигнутый на шаге (стоимость)t; З_t — затраты; E — приемлимая для инвестора норма прибыли на вложенный капитал.

Коэффициент E установим равным ставке рефинансирования ЦБ РФ — 8% годовых (0.67% в месяц). В результате анализа рынка программной продукции, аналогичной

разрабатываемой, планируется продажа 2 единц в 1-й месяц и 3 единицы в остальные. Планируемая цена ПП составляет 60000 рублей. Предполагаемые накладные расходы, связанные с реализацией составят 2000 рублей в месяц.

В таблице 5.8 приведен расчет ЧДД по месяцам работы над проектом, а на Рисунке 5.1 изображен график ЧДД (начерчен до момента, когда ЧДД принимает положительные значения).

Таблица 5.8 — Расчет ЧДД

Месяц	Текущие затраты, руб.	Затраты с начала года, руб.	Текущий доход, руб.	Доход с начала года, руб.	ЧДД, руб.
1	66451	66451	0	0	-66451
2	126201	192652	0	0	-192652
3	126201	318853	0	0	-318853
4	131901	450754	0	0	-450754
5	73296	524050	60000	60000	-464050
6	2000	526050	180000	240000	-286050
7	2000	528050	180000	420000	-108050
7	2000	530050	180000	600000	69950

Срок окупаемости проекта — 8 месяцев.

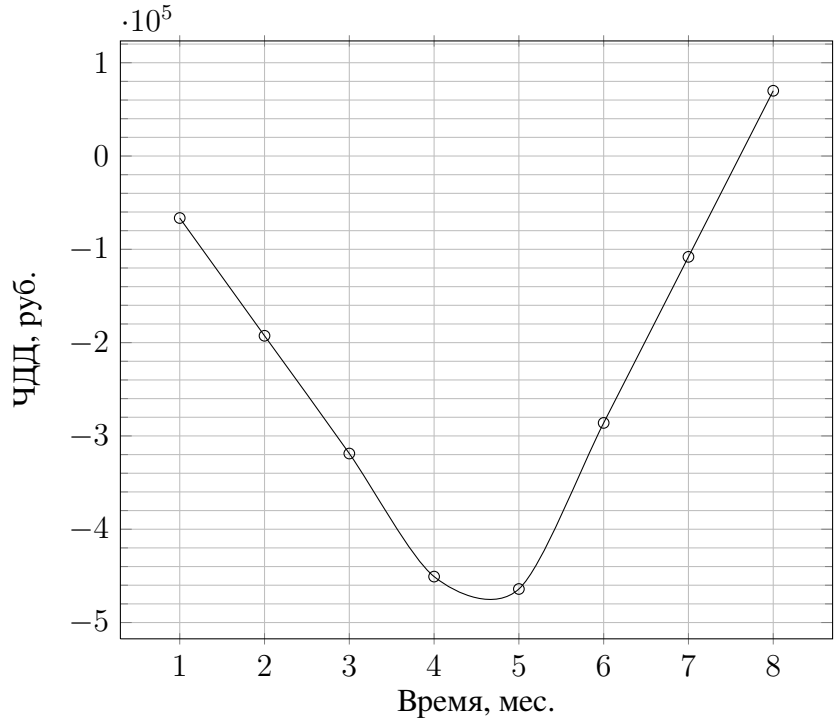

Рисунок 5.1 — График изменения ЧДД

5.7 Выводы

По результатам расчета трудоемкости выполнения работ, затрат на выполнение работ и прибыли от реализации программного продукта был найден чистый дисконтиро-

ванный доход по месяцам разработки и реализации программного продукта. На основе полученных данных можно сделать вывод, что проект эффективен и срок его окупаемости составляет 8 месяцев с начала разработки.

6 Охрана труда и экология

6.1 Оценка условий труда на рабочем месте пользователя ПЭВМ

Разработка ПО требует длительного взаимодействия с вычислительными системами. Работа с ПЭВМ связана с рядом вредных и опасных факторов, таких как статическое электричество, рентгеновское излучение, электромагнитные поля, блики отраженный свет, ультрафиолетовое излучение. При длительном воздействии на организм эти факторы негативно влияют на здоровье человека.

6.1.1 Требования к организации рабочих мест

При организации рабочего места должны учитываться следующие требования:

— достаточное рабочее пространство, позволяющее человеку осуществлять необходимые движения и перемещения при эксплуатации и техническом обслуживании;

— оптимальное размещение оборудования;

— необходимое естественное и искусственное освещение;

— наличие необходимых средств защиты работающего персонала от действия вредных и опасных производственных факторов.

Важную роль играет планировка рабочего места, которая должна способствовать удобству выполнения работ, экономии электроэнергии и времени оператора, удобству обслуживания ЭВМ и отвечать правилам охраны труда. В соответствии с ГОСТ Р 50923-96 «Дисплеи. Рабочее место оператора. Общие эргономические требования к производственной среде. Методы измерения» к организации рабочих мест предъявляются следующие ниже требования. Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и сим- волов. Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики. Поверхность рабочего стола должна иметь коэффициент отражения 0,5-0,7. Регулируемая высота рабочей поверхности должна изменяться в пределах от 680 до 800 мм. Механизмы для регулирования высоты рабочей поверхности стола должны быть легко досягаемы в положении сидя, иметь простое управление и надежную фиксацию. Высота рабочей поверхности при нерегулируемой высоте должна составлять 725 мм. Поверхность рабочего стола должна иметь коэффициент отражения 0,5 - 0,7. Рабочая поверхность стола должна иметь следующие размеры: глубина не менее 600 мм; ширина не менее 1200 мм.

Рабочая поверхность стола не должна иметь острых углов и краев. Конструкция рабочего стула (кресла) должна обеспечивать поддержание рациональной рабочей позы

при работе на ПЭВМ позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула (кресла) следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ. Рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию. Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений. При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора), должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м.

6.1.2 Параметры микроклимата

Параметры микроклимата могут меняться в широких пределах, в то время как необходимым условием жизнедеятельности человека является поддержание постоянства температуры тела благодаря терморегуляции, т.е. способности организма регулировать отдачу тепла в окружающую среду. Принцип нормирования микроклимата - создание оптимальных условий для теплообмена тела человека с окружающей средой. Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться определенные параметры микроклимата. Нормы, установленные СанПиН 2.2.2/2.4.1340-03 для категории работ 1а приведены в таблице 6.1. Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения.

Таблица 6.1 — Параметры микроклимата

Период год	Температура, °С		Влажность, %		Скорость воздуха, м/с	
	Оптим.	Допуст.	Оптим.	Допуст.	Оптим.	Допуст.
Холодный	22–24	21–25	40–60	75	0.1	0.1
Теплый	23–25	22–28	40–60	55 при 28°С	0.1	0.1

Вредным фактором при работе с ЭВМ является также запыленность помещения. Этот фактор усугубляется влиянием на частицы пыли электростатических полей персональных компьютеров.

Для устранения несоответствия параметров указанным нормам проектом предусмотренно использование системы кондиционирования как наиболее эффективного и автоматически функционирующего средства.

Нормы установленные содержания в воздухе положительных и отрицательной ионов, установленные СанПиН 2.2.4.1294–03, приведены в таблице 6.2.

Таблица 6.2 — Уровни ионизации воздуха при работе на ПЭВМ

Уровни	Число ионов в кубометре воздуха	
	n^+	n^-
Минимально необходимое	400	600
Оптимальное	1500–3000	3000–5000
Максимально допустимое	50000	50000

Для обеспечения требуемых уровней предусмотренно использование системы ионизации Сапфир-4А.

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше $19.5\frac{\text{м}^3}{\text{человека}}$ с учетом максимального числа одновременно работающих в смену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры, приведены в таблице 6.3.

Таблица 6.3 — Уровни ионизации воздуха при работе на ПЭВМ

Уровни	Число ионов в кубометре воздуха
Объем до 20м^3 на человека	Не менее 30
20.40м^3 на человека	Не менее 20
Более 40м^3 на человека	Естественная вентиляция

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ в зависимости от времени года и суток, чередование труда и отдыха), так и технические средства (вентиляция, кондиционирование воздуха, отопительная система).

6.1.3 Шум и вибрации

Уровень шума на рабочем месте программиста не должен превышать 50 дБА, а уровень вибрации не должен превышать норм установленных СанПиН 2.2.2.542–96 (см. таблицу 6.4).

При разработке ПО внутренними источниками шума являются вентиляторы, а также принтеры и другие перефферийные устройства ЭВМ. Внешние источники шума — прежде всего, шум с улицы и из соседних помещений. Постоянные внешние источники шума, превышающего нормы, отсутствуют.

Для устранения превышения нормы проектом предусмотрено применение звукопоглощающих материалов для облицовки стен и потолка помещения, в котором осуществляется работа с вычислительной техникой.

Таблица 6.4 — Допустимые нормы вибрации на раочих местах с ПЭВМ

Среднегеометрические частоты октавных полос, Гц	Допустимые значения по виброскорости	
	м/с	дБ
2	4.5×10	79
4	2.2×10	73
8	1.1×10	67
16	1.1×10	67
31.5	1.1×10	67
63	1.1×10	67
Корректированные значения и их уровни в дБ	2.0×10	72

6.1.4 Освещение

Наиболее важным условием эффективной работы программистов и пользователей является соблюдение оптимальных параметров системы освещения в рабочих помещениях.

Естественное освещение осуществляется через светопроемы, ориентированные в основном на север и северо-восток (для исключения попадания прямых солнечных лучей на экраны компьютеров) и обеспечивает коэффициент естественной освещенности (КЕО) не ниже 1.5%.

В качестве искусственного освещения проектом предусмотрено использование системы общего освещения. в соответствии с СанПин 2.2.2/2.4.1340–03 освещенность на поверхности рабочего стола должна находиться в пределах 300–500 лк. Разрешается использование светильников местного освещения для работы в документами (при этом светильники не должны создавать блики на поверхности экрана).

Правильное расположение рабочих мест относительно источников освещения, отсутствие зеркальных поверхностей и использование матовых материалов ограничивает прямую (от источников освещения) и отраженную (от рабочих поверхностей) блескость. При этом яркость светящихся поверхностей не превышает $200\frac{\text{кд}}{\text{м}^2}$, яркость бликов на экране ПЭВМ не превышает $40\frac{\text{кд}}{\text{м}^2}$, и яркость потолка не превышает $200\frac{\text{кд}}{\text{м}^2}$.

В соответствии с СанПинН 2.2.2/2.4.1340–03 проектом предусмотрено использование люминесцентных ламп типа ЛБ в качестве источников света при искусственном освещении. В светильниках допускается применение ламп накаливания. Применение газоразрядных ламп в светильниках общего и местного освещения обеспечивает коэффициент пульсации не более 5%.

Таким образом, проектом обеспечиваются оптимальные условия освещения рабочего помещения.

6.1.5 Рентгеновское излучение

В соответствии с СанПиН 2.2.2/2.4.1340-03 проектом предусмотрено использование ПЭВМ, конструкция которого обеспечивает мощность экспозиционной дозы рентгеновского излучения в любой точке на расстоянии 0.5 м. от экрана и корпуса не более 0.1 мбэр/час (100 мкР/час). Результаты сравнения норм излучения приведены в таблице 6.5.

Таблица 6.5 — Сравнение норм рентгеновского излучения в различных стандартах

	Допустимое значение мкР/час, не более
СанПиН 2.2.2/2.4.1340-03	100
TCO-99	500
MPR II	500

Как видно из таблицы, стандарты MPR II и TCO–99 предъявляют менее жесткие требования к рентгеновскому излучению, чем СанПиН. Но при соблюдении оптимального расстояния между пользователем и монитором дозы рентгеновского излучения не опасны для большинства людей.

6.1.6 Неионизирующие электромагнитные излучения

Допустимые значения параметров неионизирующих излучений в соответствии с СанПин 2.2.2/2.4.1340-03 приведены в таблицах 6.6 и 6.7.

Таблица 6.6 — Предельно допустимые значения напряженности электрического поля

Диапазон частот	Допустимые значения
5 Гц – 2 кГц	25 В/м
2 – 400 кГц	2.5 В/м

Таблица 6.7 — Предельно допустимые значения плотности магнитного потока

Диапазон частот	Допустимые значения
5 Гц – 2 кГц	250 нТл
2 – 400 кГц	5 нТл

Величина поверхностного электрического потенциала не должна превышать 500 В.

Мониторы, используемые в настоящее время, удовлетворяют более жестким нормам MPR II, а значит и СанПиН.

6.1.7 Визуальные параметры

Неправильный выбор визуальных эргономических параметров приводит к ухудшению здоровья пользователей, быстрой утомляемости, раздражительности. В связи с

этим, проектом предусмотрено, что конструкция вычислительной системы и ее эргономические параметры обеспечивают комфортное и надежное считывание информации. Требования к визуальным параметрам, их внешнему виду, дизайну, возможности настройки представлены в СанПиН 2.2.2/2.4.1340–03. Визуальные эргономические параметры монитора и пределы из изменений приведены в таблице **??**.

Таблица 6.8 — Визуальные эргономические параметры ВДТ и пределы из изменений

Наименование параметров	Пределы значений параметров	
	не менее	не более
Яркость экрана (фона), $\frac{\text{кд}}{\text{м}^2}$ (измеренная в темноте)	35	120
Внешняя освещенность экрана, лк	100	250
Угловой размер экрана, угл.мин.	16	60

Для выполнения этих требований проектом предусмотренно использование современных мониторов, имеющих достаточно широкий набор регулируемых параметров. В частности, для удобного считывания информации реализована возможность настройки положения монитора по горизонтали и вертикали. Мониторы оснащены специальными устройствами и средствами настройки ширины, высоты, яркости, контраста и разрешения изображения. кроме того, в современных мониторах зерно изображения имеет размер в пределах 0.27 мм, что обеспечивает высокую четкость и непрерывность изображения. Наконец, на поверхность дисплея нанесено матовое покрытие, чтобы избавиться от солнечных бликов.

6.2 Расчет искусственного освещения

При расчете освещенности от светильников общего равномерного освещения наиболее часто применяют метод расчета по световому потоку. При расчете освещения по этому методу необходимое количество светильников для освещения рабочего места рассчитывается по формуле:

$$N = \frac{E_{min} \cdot S \cdot K}{F_{\text{Л}} \cdot \mathbf{З} \cdot z \cdot h} \tag{6.1}$$

где E_{min} — нормируемая минимальная освещенность; S — площадь помещения, м^2; $F_{\text{Л}}$ — световой поток лампы, лк; K — коэффициент запаса; z — коэффициент неравномерности освещения (для люминесцентных ламп —1.1); h — коэффициент использования светового потока в долях единицы.

E_{min} определяется на основании нормативного документа СНиП23–05–95. В соответствии с произведенным выбором в предыдущем разделе, для работы программиста $E_{min} = 300$ лк (общее освещение).

Работы, производятся в помещении, требуют различения цветных объектов при невысоких требованиях к цветоразличению, поэтому в качестве источника освещения бы-

ла выбрана лампа люминесцентная холодно-белая (ЛХБ), 1940 лк, 30 Вт. В помещениях общественных и жилых зданий с нормальными условиями среды: К=1.4.

Для люминесцентных ламп коэффициент неравномерности освещения Z=1.1.

Коэффициент использования h зависит от типа светильника, от коэффициентов отражения потолка $\rho_{\text{П}}$, стен $\rho_{\text{С}}$, расчетной поверхности $\rho_{\text{Р}}$ и индекса помещения.

Высота подвеса над рабочей поверхностью Нр=3 м. Размеры помещения А=3.5 м, В=3 м. Определим индекс помещения по формуле:

$$\phi = \frac{A \cdot B}{H_P \cdot (A + B)} = \frac{3.5 \cdot 3}{s \cdot (3.5 + 3)} = 0.54 \tag{6.2}$$

Для светлого фона примем:$\rho_{\text{П}} = 70$ $\rho_{\text{С}} = 50$ $\rho_{\text{Р}} = 10$. h = 59%.

Освещение проектируется при помощи светильников ОДОР с минимальной освещенностью $E_{min} = 300$ лк, Р=40 Вт. Число ламп в ОДОР равно 2. Необходимое число светильников для данной комнаты:

$$N = \frac{300 \cdot 9 \cdot 1.4}{1940 \cdot 0.59 \cdot 1.1 \cdot 2} = 2\text{шт} \tag{6.3}$$

Общее количество ламп $n = 2 \times 2 = 4$ шт. Длина светильника ОДОР=1.26 м. Поскольку длина помещения 3 м, то светильники помещаются в два ряда. Суммарная мощность светильников: $30 \cdot 4 = 160$ Вт. Сумарный световой поток: $1940 \cdot 4 = 7760$ лм.

6.3 Режим труда

При работе с персональным компьютером очень важную роль играет соблюдение правильного режима труда и отдыха. В противном случае у программиста отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

При несоответствии фактических условий труда требованиям санитарных правил и норм время регламентированных перерывов следует увеличить на 30%. В соответствии со СанПиН 2.2.2 546-96 все виды трудовой деятельности, связанные с использованием компьютера, разделяются на три группы:

а) группа А: работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом;

б) группа Б: работа по вводу информации;

в) группа В: творческая работа в режиме диалога с ЭВМ.

Режим труда и отдыха должен зависеть от характера работы: при вводе данных, редактировании программ, чтении информации с экрана непрерывная продолжительность работы с монитором не должна превышать 4 часов. При 8 часовом рабочем дне, через

каждый час работы необходимо проводить перерыв 5-10 минут, а каждые два часа перерыв в 15 мин.

Эффективность перерывов повышается при сочетании с производственной гимнастикой или организации специального помещения для отдыха персонала с удобной мягкой мебелью, аквариумом, зеленой зоной и т.п.

6.4 Требования к пожарной безопасности

Основным источником пожаров в рабочих помещениях с большим количеством вычислительной техники является короткое замыкание, которое может приводить к значительному нагреву электропроводки и, как следствие, к возгоранию. Противопожарная защита такого рода помещений обеспечивается путем применения следующих мер:

а) средства автоматической пожарной сигнализации и пожаротушения;

б) оснащение помещения индивидуальными средствами пожаротушения (при наличии в помещении средств вычислительной техники применяются углекислотные огнетушители);

в) применение негорючих материалов для основных строительных конструкций и материалов, в том числе облицовочных;

г) использование средств противодымной защиты.

6.5 Утилизация оргтехники и расходных материалов

Текущие положения в законодательстве обязывают юридических лиц следовать четкой процедуре утилизации отработавшего свой моральный или технический ресурс оборудования и комплектующих к ним. Вся компьютерная и оргтехника должна утилизироваться в соответствии с «Методикой проведения работ по комплексной утилизации вторичных драгоценных метал- лов из отработанных средств вычислительной техники», утвержденной Государственным Комитетом РФ по телекоммуникациям (от 19 октября 1999 г). Благодаря комплексной системе утилизации сводятся к минимуму неперерабатываемые отходы, а основные материалы (пластмассы, цветные и черные металлы) и ценные компоненты (драгоценные металлы и др.) возвращаются в производство. В вышеуказанном акте описана технология, по которой проводится уничтожение описываемого вида оборудования:

а) Выполнить снятие оборудования с бухгалтерского баланса фирмы.

б) Предоставить оборудование, подлежащее списанию, организации, у которой есть подобающие свидетельства: лицензию, выданную Ростехнадзором, позволяющую вести деятельность по уничтожению, применению, складированию, транспортировке опасных отходов, с описанием таких вариантов отходов, как отходы оргтехники, компьютерная тех-

ника, вышедшая из употребления; сертификат о постановке на особый учет Пробирной палаты.

в) Приобрести полный комплект актов о том, что была сделана легальная утилизация картриджей.

Вышеописанная процедура утилизации распространяется как на офисное оборудование (принтеры, факсы, сканеры, копиры и т.д.), так и на ПЭВМ, сервера и комплектующие к ним (жесткие диски, блоки памяти и т.д.). К устройствам хранения данных (жесткие диски, RAID-массивы, флэш-накопители и т.д.) дополнительно предъявляются организационные требования: информация на утилизируемых устройствах должна быть надежно уничтожена во избежание ее получения третьими лицами. Основными расходными материалами являются картриджи для принтеров и копиров, блоки с порошком для ксероксов, а также офисная бумага. Использованная офисная бумага принимается для переработки некоторыми частными фирмами, оказывающими услуги утилизации офисной техники. Законодательно данный процесс не регламентирован. Процесс утилизации картриджей схож с описанным выше. Здесь стоит отметить, что в отличие от утилизации офисной техники, данный процесс менее затратный ввиду того, что большинство производителей офисной техники и комплектующих к ней предоставляют бесплатные услуги по сбору и утилизации отработанных картриджей, а также предоставляют скидки на покупку оборудования при участии в программах утилизации.

6.6 Выводы

В данном разделе были выявлены и рассмотрены основные опасные факторы, с которыми сталкивается сотрудник при работе на ПЭВМ, изложены основные требования к рабочему месту разработчика программного обеспечения. Условия труда, удовлетворяющие приведенным требованиям, должны обеспечи- вать комфортную работу. Комфортные и безопасные условия труда - один из основных факторов, влияющих на производительность людей работающих с ПЭВМ. Выполнение са- нитарных норм и правил позволит создать на рабочем месте оптимальные условия труда, гарантирующие хорошую работоспособность и производительность работника. При утилизации компонентов ПЭВМ, оргтехники и расходных материалов со- гласно действующему законодательству можно избежать нанесения урона окружающей среде и повысить степень повторного использования материалов в производстве.

Заключение

В результате проделанной работы стало ясно, что ничего не ясно...

Список использованных источников

1. *Group, The LEGO*. Appendix 1-LEGO MINDSTORMS NXT Communication protocol / The LEGO Group. — 2006.

2. *Group, The LEGO*. Appendix 3-LEGO MINDSTORMS NXT ARM7 Bluetooth Interface specification / The LEGO Group. — 2006.

3. *Подоматко, Павел*. Методы решения судоку / Павел Подоматко // *ТМ*. — 2013.

4. *Trabesinger, Andreas*. Search algorithms: Generality found / Andreas Trabesinger // *Nature Physics*. — 2007.