

**«Санкт-Петербургский государственный электротехнический университет
«ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)**

Направление	09.03.01 – Информатика и вычислительная техника
Профиль	Системы автоматизированного проектирования
Факультет	Компьютерных технологий и информатики
Кафедра	Систем автоматизированного проектирования

К защите допустить

Зав. кафедрой, к.т.н., доцент

Бутусов Д.Н.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
БАКАЛАВРА**

**Тема: РАЗРАБОТКА ПРОГРАММНО-АППАРАТНОГО КОМПЛЕКСА
ИГРОВОЙ КОНСОЛИ**

Студент	_____	Сухарев Л.А.
Руководитель	_____	Боброва Ю.О.
Консультант	_____	Павлова О.А.
Консультант по нормоконтролю	_____	Примакова Е.Е.

Санкт-Петербург

2024

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю

Зав. кафедрой САПР, к.т.н., доцент

_____ Бутусов Д.Н.

«_04_»_апреля_2024 г.

Студент Сухарев Л.А.

Группа 0302

Тема работы: Разработка программно-аппаратного комплекса игровой консоли

Место выполнения ВКР: кафедра САПР, СПбГЭТУ «ЛЭТИ»

Исходные данные (технические требования):

Игровая консоль на основе Arduino. Ввод с помощью потенциометра и кнопки, вывод — OLED-дисплей. Сбор статистики игровых сессий.

Содержание ВКР: аннотация, введение, литературный обзор, теоретические аспекты разработки, практические аспекты разработки, заключение, список использованных источников (литературы).

Перечень отчетных материалов: пояснительная записка, иллюстративный материал, материалы к презентации ВКР

Дополнительный раздел - Экономическое обоснование ВКР

Дата выдачи задания

Дата представления к защите

«_02_»_апреля_____2024г.

«_05_»_июня_____2024г.

Студент

Сухарев Л.А.

Руководитель к.т.н.

Боброва Ю.О.

Консультант

Павлова О.А.

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. кафедрой САПР, к.т.н., доцент

_____ Бутусов Д.Н.

«04»_апреля__2024_ г.

Студент Сухарев Л.А.

Группа 0302

Тема работы: Разработка программно-аппаратного комплекса игровой
консоли

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	04.04 -14.04
2	Анализ состояния проблемы	14.04-20.04
3	Описание теоретических аспектов разработки	20.04-27.04
4	Описание практических аспектов разработки	27.04-5.05
5	Выполнение дополнительного раздела «Экономическое обоснование ВКР»	5.05-10.05
6	Оформление пояснительной записки	10.05-15.05
7	Оформление иллюстративного материала	15.05-19.05
8	Прохождение предварительной защиты на кафедре	20.05.2024
9	Представление ВКР для проверки степени оригинальности пояснительной записки	22.05.2024
10	Представление ВКР к защите.	05.06.2024

Студент _____ Сухарев Л.А.

Руководитель к.т.н. _____ Боброва Ю.О.

Консультант _____ Павлова О.А.

РЕФЕРАТ

Пояснительная записка содержит 98 стр., 49 рис., 15 табл., 5 лист., 39 ист., 1 приложение.

ИГРА, ИГРОВАЯ КОНСОЛЬ, ARDUINO, МЕЛКАЯ МОТОРИКА

Целью данной работы является разработка игровой консоли на базе платы Arduino и с использованием языка Arduino. Данная консоль направлена на использование детьми, имеющими проблемы с мелкой моторикой для улучшения их состояния.

Введение описывает актуальность проблемы и раскрывает задачи работы.

В первой главе проводится детальный обзор проблемы и обосновывается выбор технологий, а именно Arduino, и производится сравнение с конкурентами.

Во второй главе описан процесс проектирования устройства и игры, показана общая концепция основных аппаратных и программных составляющих.

Третья глава посвящена реализации устройства и программированию игр, особенностям работы с Arduino, а также описанию основных алгоритмов и шагов реализации основных методов.

Четвертая глава описывает экономическое обоснование проделанной работы, проводится анализ конкурентов и вывод о коммерческих возможностях разработанного продукта.

Результатом работы является устройство и программное обеспечение к нему, составляющие игровую консоль.

ABSTRACT

The purpose of this work is to develop a game console based on an Arduino board and using the Arduino language. This console is aimed at using children with problems with fine motor skills to improve their condition.

The introduction describes the relevance of the problem and reveals the tasks of the work.

The first chapter provides a detailed overview of the problem and justifies the choice of technologies, namely Arduino, and compares it with competitors.

The second chapter describes the design process of the device and the game, shows the general concept of the main hardware and software components.

The third chapter is devoted to the implementation of the device and game programming, the features of working with Arduino, as well as a description of the basic algorithms and steps for implementing the basic methods.

The fourth chapter describes the economic justification of the work done, analyzes competitors and concludes about the commercial possibilities of the developed product.

The result of the work is the device and the software for it that make up the game console.

СОДЕРЖАНИЕ

Введение	6
1. Литературный обзор	10
1.1. Нарушение мелкой моторики у детей	10
1.2. Системы развития мелкой моторики с игровой частью	13
1.3. Использование Arduino для создания системы развития мелкой моторики	20
1.4. Аналоги. Сравнение с конкурентами	24
Выводы по главе	33
2. Теоретические аспекты разработки	34
2.1. Структурная схема устройства	34
2.2. Алгоритм работы устройства	35
2.3. Описание игры «Понг»	38
2.4. Описание игры «Скроллер»	46
Выводы по главе	52
3. Практические аспекты разработки	53
3.1. Аппаратная часть	53
3.2. Меню выбора игры	56
3.3. Программная реализация игры «Понг»	57
3.4. Программная реализация игры «Скроллер»	62
3.5. Сбор статистики	65
Вывод по главе	67
4. Экономическое обоснование дипломного проекта.	69
4.1. Расчет расходов на оплату труда согласно детализированному плану-графику работ	69
4.2. Расчет материальных затрат	72
4.3. Расчет амортизационных издержек	73
4.4. Расчет прочих прямых расходов	74
4.5. Расчет накладных расходов	75
4.6. Расчет полной себестоимости	76

4.7. Расчет себестоимости одного изделия	76
4.8. Анализ конкурентов	78
4.9. Выводы по главе	80
Заключение.....	81
Список использованных источников.....	82
Приложение А.....	86

ВВЕДЕНИЕ

Развитая мелкая моторика является важным навыком, без которого невозможно представить самостоятельную навигацию во взрослой жизни. Важно развивать этот навык с самого детства, ибо именно тогда закладываются его основы. Однако, врожденные или приобретенные заболевания могут усложнить эту задачу. К сожалению, единственным способом лечения нарушений мелкой моторики являются постоянные тренировки, поэтому важно, чтобы у пациентов, чаще всего являющимися детьми, сохранялась мотивация продолжать план лечения, несмотря на трудности. Важно постоянно проводить тренировку мелкой моторики и следить за изменениями, чтобы подобрать наилучший возможный курс лечения для пациента.

Нарушение мелкой моторики у детей, являющееся расстройством кистевого и пальцевого праксиса, сопровождающееся трудностями выполнения точных координированных движений, приводит к появлению у дошкольников проблем с удержанием предметов в руке (например, ложки, карандашей), с самостоятельным уходом за своей одеждой (застегивание пуговиц, шнуровка обуви), и с многими привычными детскими активностями и играми, такими как конструирование, рисование, лепка и т.д. [1]

Справиться с этими проблемами помогают устройства развития мелкой моторики с игровой частью. Однако существующие устройства имеют ряд недостатков, усложняющих их использование вне медицинских учреждений. Для того, чтобы предоставить альтернативу и позволить продолжение реабилитации в домашних условиях или же во время какой-либо поездки или ситуации, когда регулярное посещение медицинских заведений затруднено, было принято решение создать в рамках ВКР игровую консоль, направленную на развитие мелкой моторики.

Целью данной ВКР является создание устройства, содержащего несколько игр и возможность сбора статистики игровых сессий. Данное устройство должно

иметь небольшие размеры, должно быть удобно для удержания в руках в процессе игры, а также иметь простое, интуитивно понятное управление, поскольку направленно на использование детьми. Также необходимо создать функцию сбора статистики. По полученным данным можно будет, консультируясь со специалистом, наблюдать за прогрессом реабилитации, а также вносить в него корректировки при необходимости.

Задачами данной ВКР являются:

1. Сбор и анализ данных о проблемах мелкой моторики
2. Анализ существующих аппаратов с игровой частью, тренирующих мелкую моторику
3. Выявление недостатков конкурентов и концептуализация устройства
4. Изучение способов создания игровой консоли
5. Выбор игр для игровой консоли
6. Разработка аппаратной части игровой консоли
7. Разработка программной части игровой консоли
8. Реализация сбора статистики использования, направленной на изучение прогресса пациента

Для аппаратной части необходимо изучить способы создания игровых консолей и выбрать подходящую основу для создаваемого устройства. После этого необходимо изучить возможности ввода, вывода и обратной связи. После этого необходимо спроектировать игровую консоль, помня про требования к размерам и особенностям управления.

Для программной части необходимо разработать общий алгоритм работы консоли, позволяющий продолжительное использование без необходимости перезагрузок, выключения или ручного рестарта. Также необходимо разработать алгоритмы игр, при этом игры должны иметь простое интуитивно понятное управление, требующее навыков мелкой моторики. Однако важно также иметь ввиду целевую аудиторию консоли — дети с проблемами мелкой моторики. Проектируемые игры должны одновременно быть простыми для понимания и

управления, и достаточно увлекательными, дабы достичь постоянной мотивации использовать устройство. Также важно предусмотреть возможность увеличения уровня сложности игр со временем, корректируя ее под конкретные нужды и состояние пациента, особенно во время продолжительной реабилитации.

Программная часть тесно связана с аппаратной частью, важно добиться высокой точности реакции программы на объекты управления, поскольку любые ошибки будут негативно влиять как на мотивацию продолжать использовать проектируемое устройство, так и на общий процесс реабилитации.

Программная часть также должна включать сбор статистических данных, полученных в процессе игровых сессий. В базовой версии прототипа планируется сохранение данных на персональный компьютер, однако также важно оставить возможность для улучшений, таких как автономная работа устройства с сохранением данных либо на внешний носитель информации, подключенный к устройству, либо удаленное сохранение данных с использованием Bluetooth модуля.

Объектом исследования являются игровые консоли и устройства с игровой частью, направленные на развитие мелкой моторики и имеющие возможность прямой интеграции в процесс реабилитации.

Предметом исследования является конкретная реализация аппаратной и программной части игровой консоли.

Практическим применением прибора, разработанного в результате данной работы, является, в основном, использование в рамках реабилитационных программ вне медицинских учреждений (т.е. в домашних условиях, в условиях поездки и т.д.) для развития навыков мелкой моторики. Возможно использование в рамках развлекательных целей пользователями без заболеваний, связанных с проблемами мелкой моторики. Устройство также предусматривает сбор статистики игровых сессий, эти данные могут использоваться для наблюдения за процессом реабилитации, для изучения собственных возможностей и прогресса,

или же для соревновательных целей при использовании устройства более чем одним человеком.

1. ЛИТЕРАТУРНЫЙ ОБЗОР

1.1. Нарушение мелкой моторики у детей

Для начала рассмотрим, что такое проблемы с мелкой моторикой и причины их возникновения, дабы лучше понять, с чем ведется работа. Данное расстройство имеет код МКБ-10 F82 — Специфические расстройства развития моторной функции. Под названием «мелкая моторика» понимается разнообразие самых различных движений кистей рук: от захвата и удержания относительно крупных объектов, например, кружки до точных манипуляций, таких как рисование, письмо, манипуляции с мелкими объектами, такими как бусины в бисероплетении. В детском возрасте мелкая моторика особенно важна, так как, помимо того, что это одно из основных средств познания мира, мелкая моторика является базой для развития высших психических функций: памяти, мыслительных процессов, внимания, речевых навыков. [1]

Для точных тонких движений кисти и пальцев рук необходима согласованная работа нервной, костно-мышечной и зрительной систем. При проблемах или повреждениях одной из этих систем происходит развитие нарушений мелкой моторики. У детей расстройства ручного праксиса чаще всего вызываются следующими факторами:

- Перинатальная патология. Главная роль здесь отводится гипоксически-ишемическим повреждениям ЦНС. Последствиями перинатальной энцефалопатии в старшем возрасте могут быть минимальная мозговая дисфункция, ДЦП, эпилепсия, ЗПРР, олигофрения. Неблагоприятные последствия для последующего моторного развития имеют родовые травмы головы и ШОП.
- Недоношенность. От 30 до 50% детей, рожденных с малым гестационным возрастом, имеют отставание в физическом и психомоторном развитии. Это связано с более частыми отклонениями в неврологическом статусе,

дефектами зрения, двигательными нарушениями у недоношенных детей по сравнению с рожденными в срок.

- Нейроинфекции. Последствием энцефалитов, церебеллитов, менингитов может выступать атаксия у детей. Поражение мозжечка всегда сопровождается нарушением координации произвольных движений.
- Травмы головы. Двигательные и неврологические расстройства у детей нередко сохраняются в отдаленном периоде после перенесенной ЧМТ. Чаще всего они выражаются в снижении скорости движений, нарушении координации, реже возникают парезы и параличи конечностей.
- Травмы и патологии верхних конечностей. Следствием переломов, ожогов рук, оперативных вмешательств могут становиться контрактуры суставов, ограничивающие движение кистей и пальцев. Мелкомоторные функции также ограничены у детей с врожденными аномалиями верхних конечностей – синдактилией, клинодактилией, эктродактилией.
- Наследственные заболевания. Нарушения мелкой моторики отмечаются у значительного числа детей, имеющих генетические патологии. Как правило, это синдромы, ассоциированные с умственной отсталостью и физическим недоразвитием (Дауна, Ангельмана, Корнелии де Ланге, Мартина-Белл и мн. др.).
- Нарушения зрения. Планирование точных движений невозможно без хорошей ориентировки в пространстве. Если у ребенка существенно снижена функция зрительного анализатора, имеется косоглазие или амблиопия, то формирование мелкой моторики также значительно страдает. [1]

Причиной также могут являться психические и психологические особенности. Так, например, нарушение моторики обнаруживается у детей с СДВГ, аутизмом, синдромом Аспергера, речевой патологией (алалией, заиканием, дизартрией). Моторная недостаточность у взрослых чаще бывает связана с перенесенным инсультом, прогрессирующей болезнью Паркинсона,

спиноцеребеллярной атаксией, последствиями черепно-мозговых травм, опухолями мозга, однако эти заболевания не являются болезнью исключительно взрослых людей, и могут также служить причиной нарушений мелкой моторики у детей.

Развитие мелкой моторики в основном приходится на дошкольный возраст и заканчивается к 6-7 годам. Старшие дошкольники должны хорошо владеть тонкими манипулятивными действиями, в т. ч. с карандашом, кисточкой для рисования и пр. Развитость моторики характеризуется такими качествами, как скорость, сила, точность, ловкость. Эти навыки коррелируют с возможностью выполнения бытовых действий, предметной и конструктивной деятельностью, освоением навыка письма.

В основе механизма нарушения мелкой моторики лежат расстройства центрального или периферического звена двигательного анализатора. Стойкие расстройства пальцевого и кистевого праксиса возникают в результате мышечной дистонии, парезов, гиперкинезов, функциональной неполноценности руки, дефектов зрения. При этом страдает не только двигательная сфера, но и речевое, сенсорное, интеллектуальное развитие ребенка.

С учетом этиопатогенетических механизмов различают 3 варианта расстройства тонкой моторики:

- Нарушение зарождения нервного импульса – в этом случае сигнал, необходимый для выполнения действия, в нейронах коры не возникает. Такая ситуация наблюдается при очаговых поражениях головного мозга: инсульте, опухолях, травмах.
- Нарушение нейротрансмиссии – сбой происходит на уровне нейронной цепи, когда из-за отсутствия связи между нервными клетками возникший импульс не может достичь своей цели. Данный механизм типичен для нейродегенеративных заболеваний, в частности, болезни Паркинсона.
- Нарушение приема импульса – нервный сигнал не воспринимается органами движения в силу нарушения периферической иннервации, из-за

чего не возникает адекватного двигательного ответа. Отмечается при ДЦП, травматическом повреждении конечностей.

Дети с проблемами мелкой моторики встречаются множество трудностей в повседневной жизни: они имеют проблемы в самообслуживании, одевании и раздевании, приеме пищи. Таким детям сложно застегивать пуговицы и кнопки, шнуровать ботинки, завязывать банты, правильно держать вилку и ложку. Также отмечается несформированность продуктивной деятельности. Дети не любят такие занятия, как лепка из глины и пластилина, раскрашивание, аппликация, собирание конструктора, т. к. их неловкость не позволяет достичь ожидаемого результата. В процессе рисования они неправильно держат кисточку, не регулируют силу нажима на карандаш, не соблюдают масштаб и границы строки, листа. Характерно плохое владение ножницами при вырезании. При этом грубых двигательных расстройств у ребенка может не быть.

Практически всегда дети с неразвитой мелкой моторикой испытывают речевые проблемы. Они могут быть выражены в различной форме и степени: от функциональной дислалии и стертой дизартрии до заикания и ОНР разного уровня. Также у детей с нарушением моторики затруднено формирование графомоторных навыков, необходимых для перехода к письму.

Следствием низкого уровня сформированности мелкой моторики становятся школьные трудности. Дети имеют неаккуратный, неразборчивый почерк, что приводит к проблемам с выполнением письменных заданий. Несовершенства устной речи часто приводят к развитию дисграфии. Характерна плохая успеваемость по трудовому обучению, рисованию. Все это приводит к снижению познавательной активности, мотивации к обучению. [1]

1.2. Системы развития мелкой моторики с игровой частью

Основным способом коррекции проблем с мелкой моторикой является практика, выполнение действий, требующих точности, аккуратности. Наиболее

распространенными методами является выполнение пальчиковой гимнастики, массажа рук, ручной труд (шнуровку, штрихование, лепку, аппликацию). Однако важно отметить, что поскольку работа ведется с детьми, задача реабилитации после заболеваний и решение проблем мелкой моторики усложняется необходимостью заинтересовать пациентов в выполнении лечебных упражнений, удержанием внимания и необходимостью какого-то результата или же психологического чувства награды после выполненной работы.

Справится с этими особенностями работы с детьми помогают системы развития мелкой моторики с игровой частью. Рассмотрим самые популярные из них.

1.2.1. Реабилитационная перчатка «Аника»

«Аника» –тренажер с биологической обратной связью, служащий для восстановления мелкой моторики и координации движений. Перчатка используется при реабилитации пациентов с повреждениями головного и спинного мозга, при восстановлении моторики рук после перенесенных операций и травм. Лечение производится в легкой игровой форме, что позволяет поддерживать у пациента высокий уровень мотивации. (см. рис. 1.1)

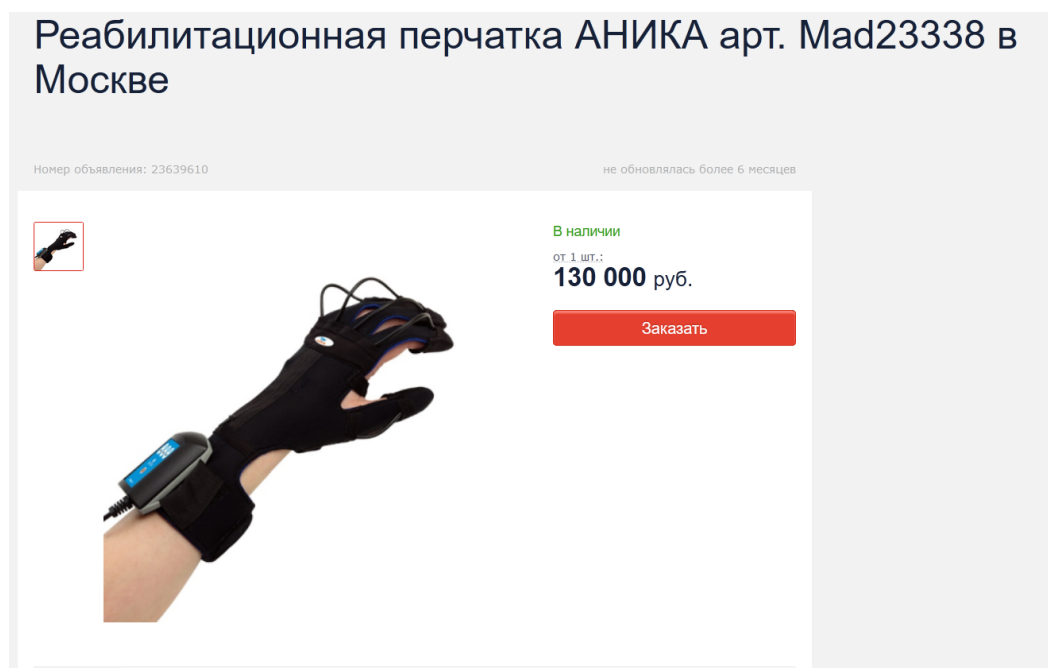


Рисунок 1.1 — Реабилитационная перчатка «Аника»

Принципы и этапы работы

Комплекс «Аника» включает в себя перчатку и программное обеспечение, устанавливаемое на ПК. На тыльной стороне кисти, предплечье и дистальных фалангах пальцев имеются датчики, фиксирующие движения и передающие информацию о них в программу. (см. рис. 1.2)



Рисунок 1.2 — Перчатка на руке пациента

Первый этап – диагностика

Изучается объем пассивных и активных движений пальцев, запястья и предплечья. Итоговые результаты исследования сохраняются. Программа анализирует эту информацию и выявляет возможности пациента.

Второй этап – реабилитация

На основе полученных данных предлагаются упражнения разной сложности. Комплекс использует зрительную обратную связь: весь процесс отражается на мониторе в реальном времени, поэтому пациент может следить за правильностью выполнения задач. При этом врач в режиме реального времени получает объективную и точную информацию обо всех действиях пациента. Это позволяет гибко настраивать программу и подбирать наиболее подходящий для конкретного случая комплекс упражнений. Упражнения проходят в

вовлекающем игровом формате – это обеспечивает необходимую мотивацию к многократному повторению целевых движений и повышает эффективность реабилитации.

Третий этап – контроль, корректировка реабилитационного процесса

Информация о процессе реабилитации сохраняется в программе, поэтому врач может использовать ее, чтобы контролировать лечение и при необходимости оперативно вносить в него корректировки. Предусмотрена возможность удаленного взаимодействия со специалистом медицинского учреждения. В этом случае программа сохраняет информацию на сервер статистики, которым в любое время может воспользоваться врач.

Простота и удобства для врача и пациента

Программное обеспечение «Аники» можно скачать из облака и установить на любом компьютере. Интерфейс удобен и понятен, а настройки очень точны: можно включать и выключать разные пальцы; выбирать игры, которые формируют строго определенные двигательные паттерны; регулировать сложность выполнения заданий. Регулируя настройки и выбирая новые, более сложные игры, мы демонстрируем пациенту его успехи, что позволяет поддерживать высокий уровень мотивации и стремление к новым достижениям. Это чрезвычайно важно при работе с детьми или с людьми, перенесшими инсульт.

Важные особенности

Реабилитационную перчатку «Аника» можно использовать в лечебных и санаторно-курортных заведениях.

Как стационарное оборудование или как часть программы мобильной реабилитации. [2]

1.2.2. Орторент МОТОРИКА

Орторент МОТОРИКА – это тренажер для реабилитации верхних конечностей, восстановления мелкой моторики рук при помощи интерактивных программ. (см. рис. 1.3)



Рисунок 1.3 — Орторент МОТОРИКА

- Клинически доказанный эффект при реабилитации верхних конечностей;
- Обеспечивает имитацию повседневных движений верхних конечностей, маскируя интенсивную реабилитацию под мотивационными играми;
- Удобен в использовании и способен адаптироваться под задачи каждого пациента, не требуя помощи третьих лиц.

Тренажер для реабилитации рук Орторент МОТОРИКА предназначен для тех, кто перенес инсульт и другие неврологические заболевания, а также для людей, страдающих от костно-мышечных повреждений и сниженной функции верхних конечностей.

Преимущества Орторент МОТОРИКА:

- Тренажер для развития мелкой моторики не уступает в качестве и эффективности.
- Реабилитация происходит в формате компьютерной игры с симуляцией реальных жизненных ситуаций. В игре доступен широкий выбор эффективных и увлекательных упражнений с различным уровнем сложности.

Аппарат Орторент МОТОРИКА предлагает 12 мотивационных программ в базовом комплекте с возможностью дальнейшего пополнения. (см. рис. 1.4)



Рисунок 1.4 — Пример мотивационных программ

- Джойстик обладает высокой чувствительностью. Он способен уловить даже небольшое по силе движение кисти, облегчая выполнение упражнений.
- Прибор имитирует движения рук и плеч, которые встречаются в повседневной жизни: от самых крупных и простых до мелкой моторики. Всего таких движений 6, и они представлены на рис. 1.5.

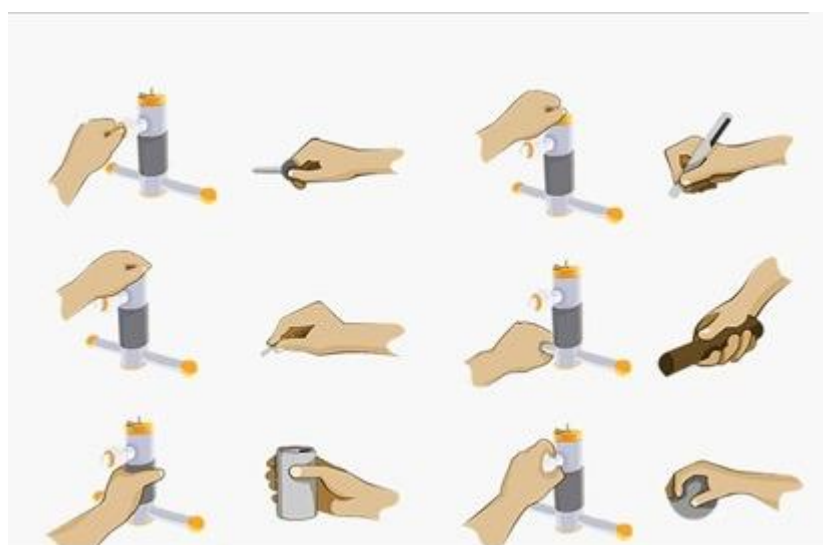


Рисунок 1.5 — основные движения рук и плеч

- Возможность ведения статистики занятий для каждого пациента, что позволяет автоматически отслеживать функциональные улучшения и вносить коррективы в процесс реабилитации;

- Возможность осуществлять удаленный контроль за пациентом при наличии выхода в Интернет;
- Plug n Play за 5 минут – для установки Орторент МОТОРИКА не требуется специальной технической подготовки;
- Тренажер Орторент МОТОРИКА удобен в использовании и способен адаптироваться под задачи каждого пациента, не требуя помощи третьих лиц. Оборудование для развития мелкой моторики можно легко установить на стол с помощью специального крепления.
- Диагностика функции руки. Программное обеспечение точно регистрирует движение верхних конечностей, позволяя врачу определить координацию пациента и прогресс восстановления.

Таким образом, выполняя необходимые упражнения, пациент добивается успеха не только в игре, но и в самом процессе реабилитации.

Интересный игровой процесс позволяет отвлечься от проблемы пациента и повысить качество реабилитации.

Обеспечивая имитацию повседневных движений верхних конечностей, маскируя интенсивную реабилитацию под мотивационными играми, терапия на тренажере для восстановления руки Орторент МОТОРИКА позволяет значительно улучшить функциональные возможности верхних конечностей.

Важно заметить, что данные тренажеры имеют довольно большую стоимость и предназначены в первую очередь для использования в специализированных клиниках, поскольку имеют слишком высокую цену для частной покупки для домашнего использования. Однако у многих пациентов может не быть возможности постоянно посещать больницу, или же может иметься проблема страха перед врачами и стерильной обстановкой больницы у детей. Поэтому целью данной работы является создание прототипа устройства с доступной ценой, подходящего для домашнего использования, и являющегося портативным устройством, подходящим в том числе и для путешествий. В результате будет получено устройство, позволяющее пациенту продолжать

занятия по улучшению мелкой моторике вне зависимости от возможности постоянного присутствия в реабилитационном центре, что делает реабилитацию более доступной и простой, ставящих меньше ограничений на семьи, столкнувшихся с этой проблемой.[3]

1.3. Использование Arduino для создания системы развития мелкой моторики

Основой для разрабатываемого устройства была выбрана плата Arduino.

Arduino – это официальные платы и программы, выпускающиеся под одной торговой маркой. Arduino включает в себя и железо (платы) и софт (среда разработки) и является, с точки зрения использования, платформой для разработки электронных устройств.

Важная особенность семейства Arduino это то, что они представляют собой отладочные платы – печатные платы, на которой стоит микроконтроллер (далее МК), который и программируется пользователем. Микроконтроллеры бывают разных типов, в более ранних моделях Arduino используются AVR (UNO, Nano, Mega, Leonardo), в новых — более мощные ARM Cortex, подходящие для более требовательных проектов.

Для дальнейшей работы нам больше всего интересны Arduino-платы, поэтому, если иначе не указано, под Arduino будет пониматься в первую очередь плата с микроконтроллером. Рассмотрим общее строение Arduino.

Arduino — это небольшая управляющая плата с собственным процессором и памятью. Помимо них на плате есть пара десятков контактов, к которым можно подключать всевозможные компоненты: светодиоды, датчики, моторы, чайники, роутеры, магнитные дверные замки и так далее, что демонстрирует рисунок 1.6.[4]

В процессор Ардуино можно загрузить программу, которая будет управлять всеми этими устройствами по заданному алгоритму. Таким образом

можно создать бесконечное количество уникальных полезных гаджетов, сделанных своими руками и по собственной задумке.[5]

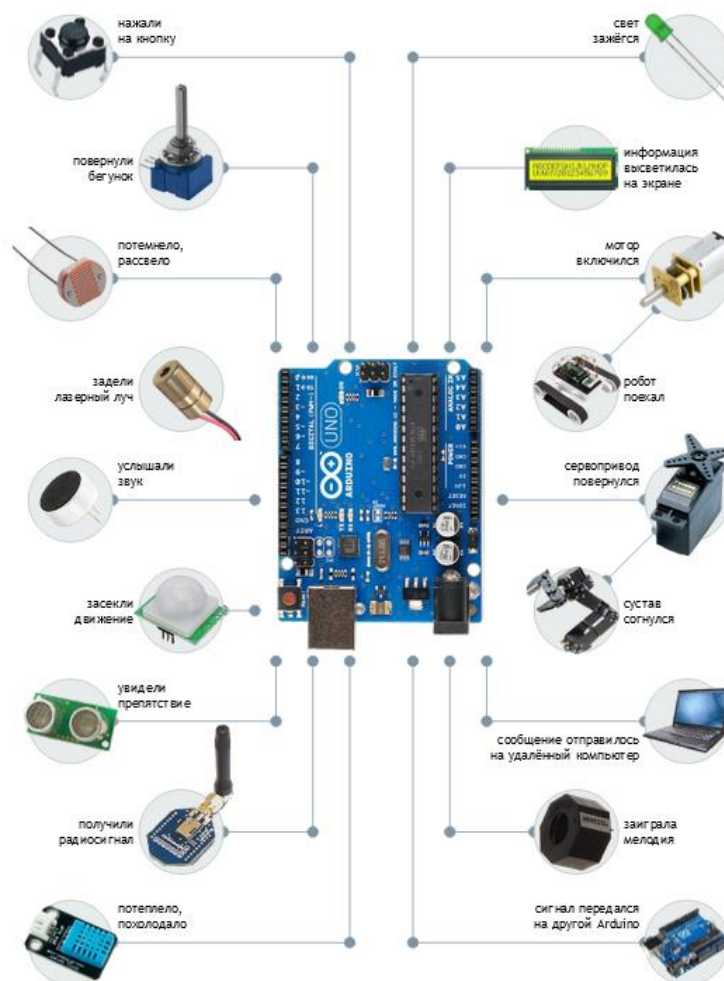


Рисунок 1.6 — Arduino и возможные подключаемые компоненты

Аппаратная часть

Для того, чтобы собрать электронное устройство на базе МК необходимо:

- Печатная плата (ибо МК имеет весьма малые размеры, и паяние МК может привести к проблемам)
- Тактирование МК (нужно подключить тактовый генератор)
- Необходимая обвязка (фильтры питания, кнопки перезагрузки, некоторые МК требует подключения резисторов к определенным пинам и так далее)

- Компоненты схемы (расположить на плате или предусмотреть штекеры для используемых компонентов: кнопок, экрана, и так далее в зависимости от целей электронного устройства)
- Стабильное питание схемы
- Прошивка (загрузка происходит с помощью программатора)

Важной особенностью Arduino является то, что разработчики объединили всё необходимое на одной плате: уже настроенный микроконтроллер и всё необходимое для его работы, стабилизатор напряжения, и самое главное — программатор, он тоже расположен на плате и для загрузки прошивки достаточно просто подключить USB кабель! Ноги МК выведены на рейку с пинами (стандартный шаг 2.54 мм), что позволяет работать с платой на брэдборде (макетная плата) и быстро подключать к ней любые компоненты. Изначально сложную задачу упростили до электронного «конструктора», что и привело к такой популярности Arduino.[4]

Принцип бутерброда

Ещё одной отличительной особенностью Arduino является наличие плат расширения под названием Shield. Эти «шилды» ставятся поверх ардуино подобно слоям бутерброда и дают ей новые дополнительные возможности. Например, существуют платы расширения для подключения к локальной сети и интернету (Ethernet Shield), для управления мощными моторами (Motor Shield), для получения координат и времени со спутников GPS (приёмник GPS/ГЛОНАСС) и многие другие.[5]

Программная часть

Для того, чтобы запрограммировать МК нужно:

- Прошивка (написанная при помощи любого текстового редактора)
- Компиляция прошивки
- Загрузка прошивки в МК

Для выполнения всех этих условий Arduino использует собственное IDE — интегрированную среду разработки Arduino IDE, представляющую собой

текстовый редактор, подходящий для компиляции и загрузки кода. Также эта IDE включает менеджер библиотек и даже поддержку неофициальных плат. Благодаря этому процесс прошивки значительно упрощается и автоматизируется — пользователю не нужно выполнять настройку вручную, и процесс сводится к одному щелчку по кнопке «загрузить». Стоит также отметить, что Arduino IDE поддерживает разные операционные системы, включающие Windows, Mac OS и Linux.

К программной части также относится язык Arduino, однако его можно назвать скорее встроенной библиотекой. Так как все Arduino-совместимые платы имеют одинаковый набор функций, то у пользователя есть возможность переносить проект практически без изменений с одной платы на другую. Стоит так же упомянуть библиотеки, во много раз упрощающие работу с модулями и в целом аппаратной частью. Для Arduino-среды существует около 5000 библиотек, которые охватывают все Arduino-модули и некоторые микросхемы. Также среди библиотек можно найти полезные методы и алгоритмы обработки данных, и множество интересных решений для самых разных задач.

Простота и удобство разработки в совокупности с огромным количеством плат на разных МК и набором библиотек на все случаи жизни сделало Arduino самой простой и удобной платформой для изучения робототехники и создания прототипов электронных устройств.

Программирование

Язык программирования Arduino официально называется «Arduino Wiring», однако многие называют его «упрощённый C++», «разновидность C++», поскольку он и правда построен на этом языке, со знакомым для C++ синтаксисом, операторами, инструментами и библиотеками, однако обладает некоторыми особенностями:

- Из-за среды Arduino IDE слегка меняется стандартный вид программы на C++, делая его более простым для понимания, что упрощает работу новичкам. В то же время программу все же можно оформить и как

обычную программу на C++, что позволяет более опытным пользователям легко перенести полученные ими ранее знания и применить их к программированию Arduino.

- Arduino IDE автоматически подключает в код библиотеку `Arduino.h`, которая содержит базовый набор функций для работы с МК, а также некоторые константы и математические функции, которые пришли из открытого фреймворка Wiring.
- В AVR Arduino используется компилятор `avr-gcc`, в котором нет стандартных для компьютерной разработки `std::` библиотек. Но зато есть свои библиотеки, ориентированные на работу с микроконтроллером.
- Arduino IDE также включает встроенные библиотеки для работы с интерфейсами связи и памятью, а также стандартные библиотеки для работы с компонентами, такими как дисплеи, шаговые моторы, сервоприводы и многими другими.
- Arduino IDE позволяет писать напрямую на ассемблере, контролируя каждый такт работы МК.
- Компилятор в Arduino IDE настроен на максимальное прощение ошибок и «всеядность», и иногда его называют обучающей платформой.

Arduino — это сердце конструктора, который в конечном итоге может быть собран как и в простую конструкцию для обучения детей робототехники, так и в сложный специфический контролер для конкретной реальной задачи: все ограничено, в первую очередь, знаниями и ресурсами пользователя, поскольку ограничения программы довольно гибкие, особенно благодаря существованию множества разных плат с МК и расширений для них. [4]

1.4. Аналоги. Сравнение с конкурентами

Среди плат для разработки можно выделить две основные категории: платы на микроконтроллере и одноплатные компьютеры.

Основным отличием является то, что, в то время как микроконтроллеры могут одновременно исполнять всего одну задачу, одноплатные компьютеры исполняют программы в рамках операционной системы (которой чаще всего является Linux) и обладают большей производительностью и широкими мультимедийными возможностями.

Стоит также упомянуть существование гибридных платформ, то есть платформ, в которых на одной плате расположен микроконтроллер и процессор. Принцип работы с такими платформами обычно заключается в разделении задач таким образом, то более сложные задачи, такие как выход в сеть, обработка медиа и прочие, выполняет более мощный процессор, а микроконтроллер берет на себя функции точного управления приводами, реле, сенсорами и остальной периферией. Пользователь может самостоятельно создать гибридную платформу, взяв по одной плате из каждого семейства. Это возможно, потому что у всех этих плат найдутся общие интерфейсы, через которые и можно будет организовать их взаимодействие.

В обоих семействах можно найти специализированные платы, сильно выделяющиеся среди прочих какой-либо особенностью, однако усредненные микроконтроллеры и одноплатные компьютеры сравнить все же можно, результаты этого сравнения объединены в табл. 1.

Таблица 1 — Сравнение микроконтроллеров и одноплатных компьютеров

	Микроконтроллер	Одноплатный компьютер
Производительность	1 ядро, десятки-сотни МГц, десятки КБ оперативки, десятки-сотни КБ постоянной памяти.	1 или более ядер, сотни-тысячи МГц, сотни МБ оперативки, гигабайты постоянной памяти.
Многозадачность	Нет. Но можно эмулировать.	Да. Управляется ОС.
Удобство работы с интернетом	☆☆☆ Обычно нужны дополнительные модули и глубокое знание протоколов.	★★★ Легко подключается из коробки, сетевой модуль обычно уже на борту.
Длительность работы от батареек	★★★	☆☆☆

	Потребляет единицы-десятки мА. Возможны недели работы от батареек.	Потребляет сотни-тысячи мА. Заряда большого аккумулятора хватит от силы на десяток часов.
Скорость реакции в проектах, критичных к времени	★★★ 100% контроль над временем и длительностью подачи сигналов.	☆☆☆ Из-за многозадачности критический процесс может проспать своё время.
Выбор языков программирования	☆☆☆ Ограниченный. Чаще C/C++.	★★★ Python, JavaScript, Bash и десятки других: любые доступные в ОС.
Возможности для работы с видео, компьютерным зрением	☆☆☆ Не хватает мощности.	★★★ OpenCV, аппаратные видеокodeки, HDMI-выход.
Возможности для работы со звуком	★★★ На мощных микроконтроллерах возможен синтез звука. Для работы с MP3/OGG/WAV нужны дополнительные модули.	★★★ Поддержка MP3/OGG/WAV на уровне ОС. Аудиовыход HDMI и/или разъём 3,5 мм.

Благодаря этому сравнению легко можно понять, что сравнивать между собой платы разных семейств для выделения преимуществ и недостатков отдельных плат дело непродуктивное, поскольку эти семейства сильно отличаются между собой. Поэтому далее будет приведено сравнение плюсов и минусов отдельных плат при сравнении только с платами своего семейства.

Сравнение микроконтроллеров

Для сравнения возьмем за основу плату Arduino Uno, поскольку именно платы этого семейства дали невероятный пинок развитию хобби-электроники во всём мире. Разные компании выпускают модули, сенсоры, платформы, дополнения с шильдами «Arduino compatible», «Designed for Arduino» и т.д. За этими словами стоит электронная и программная совместимость в первую очередь с Arduino Uno, а уж затем со всем остальным.

Как правило, при достаточном труде, знаниях и возможностях, а также доступу к нужным компонентам, можно добиться соединения и слаженной работы между собой каких угодно частей. Однако в целях сосредоточения на выбранном проекте, а на борьбе с электроникой и ее ограничения, мы будем

сравнивать любую плату на микроконтроллере именно с Arduino Uno. Сравнение различных моделей микроконтроллеров приведено в табл. 2. [6-17]

Таблица 2 — Сравнение микроконтроллеров

Модель	Микроконтроллер	Количество цифровых входов/выходов	ШИМ поддержка	Количество аналоговых входов	Количество контактов для аппаратного	Объём Flash-памяти (кБ)	Объём SRAM-памяти (кБ)	Объём EEPROM-памяти (кБ)	Тактовая частота (МГц)	Количество аппаратных serial-портов	USB-разъём	Габариты, мм	Цена, руб
Arduino Uno	ATmega328p	20	6	6	2	32	2	1	16	1	Type B	69×53	2940
Arduino Leonardo	ATmega32u4	20	7	12	5	32	2,5	1	16	1	micro-USB	69×53	3190
Iskra Neo	ATmega32U4	20	7	12	5	32	2,5	1	16	1	micro-USB	69×53	2130
Arduino Pro Mini	ATmega328P-AU	14	6	8	2	16	1	0.512	16	2	-	33x18	345
Iskra Mini	ATmega328	20	6	8	2	32	2	1	16	2	-	33×20	1040
Arduino Micro	ATmega32u4	20	7	12	5	32	2,5	1	16	1	micro-USB	48x18	2390
Arduino Mega 2560	ATmega2560	70	15	16	6	256	8	4	16	4	Type B	102x53	4840
Arduino Due	AT91SAM3X8E	54	12	12	0-54	512	64	-	84	4	Micro-B	102x53	5940
Iskra JS	STM32F405RG	26	22	12	2	1024	192	1	168	2	micro-USB	69x53	2590
Strela	ATmega32u4	11	4	8	2	32	2.5	1	16	2	Type B	105x79	3890
STM32 Nucleo F401RE	STM32F401	81	10	10	3	512	96	1	84	4	mini-USB	83x70	3990
Teensy 3.2	MK20DX256	34	12	21	6	256	64	2	72	4	USB	35x17	3040

Также каждая плата обладает, помимо основных технических особенностей, некоторыми характеристиками, которые в целом можно разделить на их плюсы и минусы среди соседей по семейству (см. табл. 3).

Таблица 3 — Плюсы и минусы разных моделей микроконтроллеров

Модель	Заметные плюсы	Заметные минусы
Arduino Uno	<p>Тонны документации, уроков и готовых библиотек, огромное сообщество, работа из простой в освоении среды <u>Arduino IDE</u> с языком Arduino C++. Всё это просто не даст вам возможности сказать «не осилил».</p> <p>Родное напряжение в 5 вольт, которое является стандартом де-факто, и колодки для установки <u>плат расширения</u>, аналоговые входы, всевозможные аппаратные интерфейсы позволяют подключить практически любую периферию, <u>сенсоры</u> и исполнительные устройства.</p>	
Arduino Leonardo	<p>Та же Arduino Uno, но с другим, слегка улучшенным микроконтроллером. Большее количество аналоговых входов (12 против 6) для сенсоров, больше каналов ШИМ (7 против 6), больше пинов с аппаратным прерыванием (5 против 2), отдельные независимые Serial-интерфейсы для USB и UART.</p> <p>Arduino Leonardo может притворяться клавиатурой или мышью (HID-устройством) для компьютера. Это позволяет легко сделать своё собственное устройство ввода.</p>	<p>Из-за небольших отличий распиновки от Arduino Uno возможна несовместимость с некоторыми платами расширения. Такие случаи, однако, редки, и в нашем магазине мы явно их прописываем.</p>
Iskra Neo	<p>Та же Arduino Leonardo, но произведённая нами, в России.</p> <p>Заметно дешевле оригинала</p>	
Arduino Pro Mini	<p>Та же Arduino Uno, но в другом форм-факторе.</p> <p>Компактная. Всего 30×18 мм.</p>	<p>Из-за форм-фактора нельзя без ухищрений устанавливать платы расширения Arduino. Предполагается соединение с дополнительными модулями</p>

Модель	Заметные плюсы	Заметные минусы
		проводами и/или через <u>макетную плату</u> . На плате нет USB-порта, поэтому прошивать нужно через отдельный преобразователь USB-Serial
Iskra Mini	Та же Arduino Mini, но произведённая нами, в России. Есть в варианте с распаянными колодками и с незапаянными отверстиями	Заметно дороже оригинала.
Arduino Micro	Та же Arduino Leonardo, но в другом форм-факторе. Компактная. Всего 48×18 мм.	Из-за форм-фактора нельзя без ухищрений устанавливать платы расширения Arduino. Предполагается соединение с дополнительными модулями проводами и/или через <u>макетную плату</u> .
Arduino Mega 2560	В разы больше памяти: 256 КБ постоянной и 8 КБ оперативной. В разы больше портов: 60 из них 16 аналоговых и 15 с ШИМ.	Немного длиннее базовой Arduino Uno: 101×53 мм против 69×53 мм.
Arduino Due	Родным напряжением для платы является 3,3 В, а не традиционные 5 В. Необходимо следить, чтобы выбираемая периферия поддерживала работу с этим уровнем или ставить преобразователи уровней напряжения.	
Iskra JS	Плата на ядре Espruino: её программируют на JavaScript. JavaScript — язык высокого уровня. Программы писать проще, они компактнее и выразительнее. Особенно, если речь идёт	Другой язык программирования, нежели у большинства плат Arduino Несмотря на то, что родной уровень для платы — 3,3 вольта, пины толерантны к 5 вольтам:

Модель	Заметные плюсы	Заметные минусы
	<p>о многочисленных строковых операциях, массивах данных, веб-интерфейсе.</p> <p>Мощный микроконтроллер Cortex-M4 на 168 МГц, 1 МБ флеш, 192 КБ оперативной памяти, десятки портов с ШИМ и аналоговых входов, 2 аналоговых выхода, по несколько I²C, SPI, UART — всё это даёт подключить и одновременно работать с самыми разнообразными сенсорами и модулями.</p>	<p>подключение пятивольтовой периферии тривиально.</p> <p>Из-за другой среды и экосистемы для программирования может не существовать готовой библиотеки для выбранной периферии. Её придётся реализовать самостоятельно.</p>
Strela	<p>Встроенный драйвер для двух двигателей, 4 разъёма для сервоприводов, 4 кнопки и 4 светодиода свободного назначения, зуммер, слоты для ЖК-экрана и модуля беспроводной связи.</p> <p>Мощный регулятор питания позволяет без ухищрений использовать множество различных аккумуляторов.</p> <p>11 входов-выходов выведены в виде трёхконтактных разъёмов для лёгкого подключения дополнительных датчиков и модулей. ЖК-экран, кнопки и светодиоды подключены через расширитель портов, поэтому они не занимают входы-выходы общего назначения.</p>	<p>На плате не предусмотрены колодки для установки плат расширения Arduino.</p> <p>Из-за изменённой нумерации контактов (в сравнении с базовой Arduino Leonardo) необходимо использовать немного другие функции для работы с пинами платы. Они предоставлены в одноимённой библиотеке.</p>
STM32 Nucleo F401RE	<p>Процессор на 84 МГц, 512 КБ постоянной и 96 КБ оперативной памяти. 50 портов ввода-вывода, из которых 16 аналоговых и 29 с ШИМ. Родной уровень напряжения — 3,3 В, но все пины толерантны к 5 В, поэтому проблем электронной совместимости с Arduino-периферией возникнуть не должно.</p> <p>Колодки для плат расширения по конфигурации совпадают с Arduino Uno, поэтому на Nucleo можно поставить множество плат расширения от Arduino.</p>	<p>Из-за другой среды и экосистемы для программирования может не существовать готовой библиотеки для выбранной периферии. Её придётся реализовать самостоятельно.</p>

Модель	Заметные плюсы	Заметные минусы
	На плате не выведен отдельный SPI-разъём. Платы расширения Arduino, которые используют SPI через ICSP-разъём, без ухищрений не будут работать.	
Teensy 3.2	<p>Меньше Arduino Micro (35×17 мм), но почти столь же мощная, как Nucleo: процессор 72 МГц, 256 КБ постоянной и 64 КБ оперативной памяти, 34 порта ввода-вывода, из которых 21 могут быть аналоговыми, а 12 поддерживают ШИМ.</p> <p>Teensy 3.2 очень энергоэффективна. У неё нет регулятора напряжения, но входным может являться любое от 3,3 до 5,5 В. Это же напряжение и будет логическим уровнем. В режиме сна плата потребляет всего 0,25 мА, что даёт возможность работать от аккумулятора несколько месяцев.</p> <p>Встроенный контроллер шины CAN позволяет создавать сеть из Due или взаимодействовать с автомобильной электроникой. Два канала ЦАП позволяют синтезировать стереозвук с разрешением в 4,88 Гц.</p>	<p>Плата поставляется с нераспаянными контактами. Вам предстоит самостоятельно впаять <u>штырьковые соединители</u> или проводки.</p> <p>Из-за большой разницы в архитектуре с классическими Arduino не все библиотеки для сторонней периферии могут работать из коробки.</p> <p>Рабочее напряжение равно входному, поэтому плавёт по мере разряда батарейки. Это может оказаться важным при выборе периферии, если она рассчитана на какой-то конкретный вольтаж.</p>

Сравнение одноплатных компьютеров

Законодателем моды среди одноплатных компьютеров является Raspberry Pi. Эта платформа в своё время перевернула представление о возможностях, габаритах и стоимости полноценного компьютера для людей, желающих заняться электроникой как хобби.

Опять же, для каждого проекта может лучше подойти тот или иной одноплатный компьютер, однако, для того чтобы все же узнать что-то об одноплатных компьютерах, и посмотреть на их особенности, сравним две самые

популярные модели - Raspberry Pi 3 Model B и BeagleBone Black (см. табл. 4).
[18]

Таблица 4 — Сравнение одноплатных компьютеров

Модель	Особенности	Плюсы	Минусы
Raspberry Pi 3 Model B	Один из самых популярных одноплатников. Четыре ядра по 1200 МГц, 1 ГБ оперативной памяти и полноценный Linux, основанный на Debian, помогут решить множество задач, требовательных к вычислительным ресурсам. Среди них можно выделить компьютерное зрение, обработку звука в реальном времени, создание веб-сервисов.	Тонны документации, уроков и готовых библиотек, огромное сообщество. Всё это просто не даст вам возможности сказать «не осилил». Привычные порты HDMI, 3,5 мм аудио, 4 USB помогут с лёгкостью подключить монитор, колонки, клавиатуру, мышь и другие USB-устройства. Модули BLE и Wi-Fi на борту помогут соединить компьютер с другими устройствами без проводов.	На плате нет АЦП, поэтому подключение аналоговых сенсоров возможно только с помощью внешних, дополнительных компонентов. Предоставляется лишь 1 аппаратный ШИМ-канал, что усложняет работу с периферией, которая управляется ШИМ'ом.
BeagleBone Black	Микрокомпьютер, схожий с Raspberry Pi, который даёт больше благ, привычных для микроконтроллерных плат. Отличный выбор для проектов интернета вещей, когда необходимо управляться с множеством сенсоров и исполнительных устройств.	Мощная среда для разработки Cloud9 IDE. Вы просто заходите на BeagleBone через браузер и программируете на любимом языке, будь то Python, JavaScript (Node.js), Bash или любой другой язык Linux. Результат можно проверить мгновенно, а если что-то не заработало, использовать встроенный в среду полноценный отладчик. На борту уже установлена флеш-память eMMC на 4 ГБ с операционной системой Linux. Память	Диковинный разъём microHDMI для подключения монитора. Для передачи звука используется он же. Вычислительная мощность скромнее, чем у Raspberry Pi: 1 ядро на 400 МГц и 512 МБ оперативной памяти.

Модель	Особенности	Плюсы	Минусы
		<p>может быть увеличена внешней microSD-картой.</p> <p>Широкие возможности по подключению периферии. 8 ШИМ-выходов и 7 аналоговых входов. Возможны аппаратные прерывания.</p>	

Выводы по главе

В результате данной главы была изучена проблематика выбранной предметной области. Было выяснено, что причиной проблем с мелкой моторикой могут являться самые различные наследственные и приобретенные заболевания, и единственным методом реабилитации является постоянная тренировка конечностей.

Были изучены конкурирующие продукты в области развивающих мелку моторику аппаратов с игровой частью и выяснены их основные преимущества и недостатки. В результате было выявлено направление разработки игровой консоли в рамках данной ВКР: хотя конкуренты имеют большой выбор игр и обладают широкой функциональностью, индивидуальная покупка этих устройств затрудняется высокой стоимостью, а их размеры и требования к установке ещё больше затрудняют использование в домашних условиях. В результате было решено создать игровую консоль небольших размеров, направленное на использование вне медицинских учреждений.

Основой игровой консоли были выбраны платы Arduino. Была изучена теория работы с этими платами, а также проведено сравнение между платами с микроконтроллером и одноплатных компьютеров по различным характеристикам. В результате было решено использование платы Arduino Uno как основу для игровой консоли.

2. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ РАЗРАБОТКИ

2.1. Структурная схема устройства

Целью данной работы является разработка портативного устройства, помогающего развивать мелкую моторику, главными особенностями которого станут небольшой размер, позволяющий домашнее использование и использование во время путешествий, поездок, и в ситуациях, когда у пациента нет возможности постоянно посещать реабилитационные центры. Так как устройство предназначено для управления либо самим пациентом, либо опекунами, не обязательно имеющими надлежащее медицинское образование, оно также должно быть достаточно просто и понятно в управлении.

Как было указано раньше, поскольку устройство нацелено на реабилитацию, в первую очередь, детей, испытывающих трудности с мелкой моторикой, то важной задачей является заинтересовать ребенка процессом реабилитации. Для решения этой проблемы было решено ввести игровой метод, что делает создаваемое устройство по сути своей игровой консолью.

Разберемся в строении этой консоли.

Основными частями создаваемого прототипа являются: устройство ввода, устройство вывода, контроллер и система, позволяющая производить мониторинг результатов, что позволит анализировать прогресс пациента. (см. рис. 2.1)

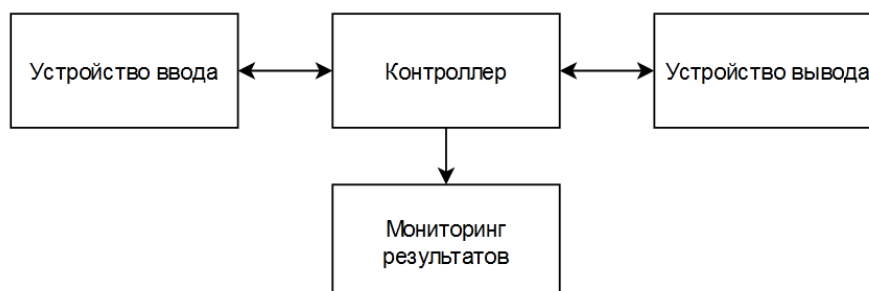


Рисунок 2.1 — Структура проектируемого устройства.

Устройство ввода

Ввод осуществляется посредством реостатов и кнопок, имеющих небольшие размеры, что требует использование мелкой моторики, тренируя ее в процессе игры. Управление осуществляется с помощью кнопок и подкручиванием реостатов, нажатие кнопки или поворот реостата изменяют положение контролируемого пользователем объекта на экране (Устройство вывода)

Устройство вывода

Устройством вывода является экран, отображающий всю необходимую для управления устройством информацию. На данном экране отображается текущая игра, счет и объекты, управляемые пользователем.

Мониторинг результатов

По ходу игры контроллер записывает результат текущей игры, который позже может использоваться чтобы оценить наличие или отсутствие прогресса в развитии мелкой моторики. Данные записываются в отдельный файл, подходящий для анализа процесса реабилитации.

Контроллер

Контроллером является плата Arduino, которая, согласно коду, контролирует все остальные компоненты и выполняет их связь между собой. Так, например, контроллер обеспечивает адекватную реакцию игры на пользовательский ввод (нажатие кнопки, подкрутку реостата), а также собирает и посылает с помощью устройства для мониторинга результатов статистику игры.

2.2. Алгоритм работы устройства

Поскольку устройство создается для работы с детьми, важно, чтобы оно имело простое и понятное управление. Таким образом основные этапы алгоритма работы устройства должны быть легко делимы между собой,

управление должно иметь понятный интуитивный интерфейс, доступный для понимания без сложных дополнительных инструкций.

Основной цикл можно разделить на несколько этапов — старт, инициализация, выбор игры, сбор статистики. (см. рис. 2.2)

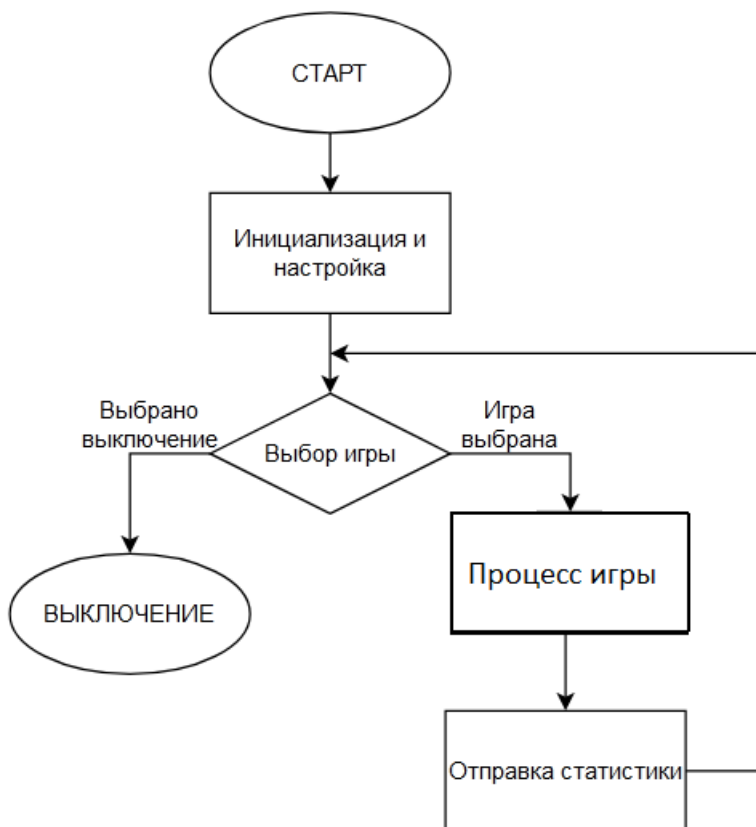


Рисунок 2.2 — Основные этапы работы программы

На старте происходит загрузка программного обеспечения устройства и проверка корректной работы всех составляющих, в особенности наличие удачного соединения для отсылки собранной статистики игр. В случае, если возникла какая-то проблема или же не удалось установить соединение, устройство должно выдавать сообщение об ошибке, содержащее текстовое описание проблемы.

Выбор игры — суть этого этапа понятна из названия. На экран должен выводиться список доступных игр. Важно учесть ограничения размера экрана и вывести игры так, чтобы понятно было, что каждая игра из себя представляет,

какая игра будет выбрана при переходе к следующему шагу. При этом важно не загружать экран лишней информацией, не вызвать информационного перегруза, сделать дизайн выбора меню простым и не вызывающим сложностей для понимания, для навигации и так далее.

В процессе самой игры будет запускаться часть программы, отвечающая за конкретную игру, и вестись сбор статистики текущего игрового сеанса. На данном этапе критически важна стабильность работы программы и устройства, так как малейшие ошибки и задержки не только повлияют на собранную статистику, что может привести к ошибкам анализа процесса восстановления пациента, но также эти ошибки, если замечены пользователями, негативно повлияют на мотивацию продолжать работать с устройством, что плохо не только для имиджа разработчика, но и для главной цели создания устройства — помощи в реабилитации и развитии мелкой моторики. Поэтому на данном этапе работы программы важно обеспечить отсутствие задержек, потерь памяти, и провести тщательное тестирование, гарантирующее, что даже при самом нелогичном с точки зрения разработчика входном сигнале от пользователя устройство работает как задумывалось, и корректно отвечает на подаваемые пользователем команды.

После завершения игры необходимо, помимо выведения результата игры, отправить собранную статистику в выбранное хранилище. Планируется создание отдельного файла на персональном компьютере, содержащий информацию о времени игрового сеанса и результате (выигрыш или проигрыш, набранное количество очков). Эти данные позже можно будет использовать для анализа прогресса развития мелкой моторики: улучшается ли время и количество выигранных игровых сессий. Важно обеспечить корректную передачу данных без потерь и искажений, так как любые неточности поставят под угрозу саму цель сбора данных, делая их непригодным для анализа.

После завершения отправки данных игровая консоль должна вернуться обратно к меню выбора игры, таким образом создавая цикл «выбор игры —

процесс игры – отправка статистики – выбор игры...». Важно отметить, что при входе в новый виток цикла не должно оставаться данных из предыдущих циклов, поскольку это может привести к забиванию памяти, что вызовет ошибки в работе устройства при длительном использовании. Также «старые» неудаленные данные могут вызвать непредсказуемое поведение сбора статистики, ставя под угрозу точность и правдивость отправляемых результатов.

2.3. Описание игры «Понг»

Обзор игры

Pong — одна из ранних аркадных видеоигр (см. рис. 2.3). Это простой спортивный симулятор настольного тенниса. теннисная спортивная игра с использованием простой двумерной графики, разработанная и выпущенная фирмой Atari в 1972 году. Pong называют первой в истории коммерчески успешной видеоигрой, а с её именем связывают появление индустрии интерактивных развлечений.

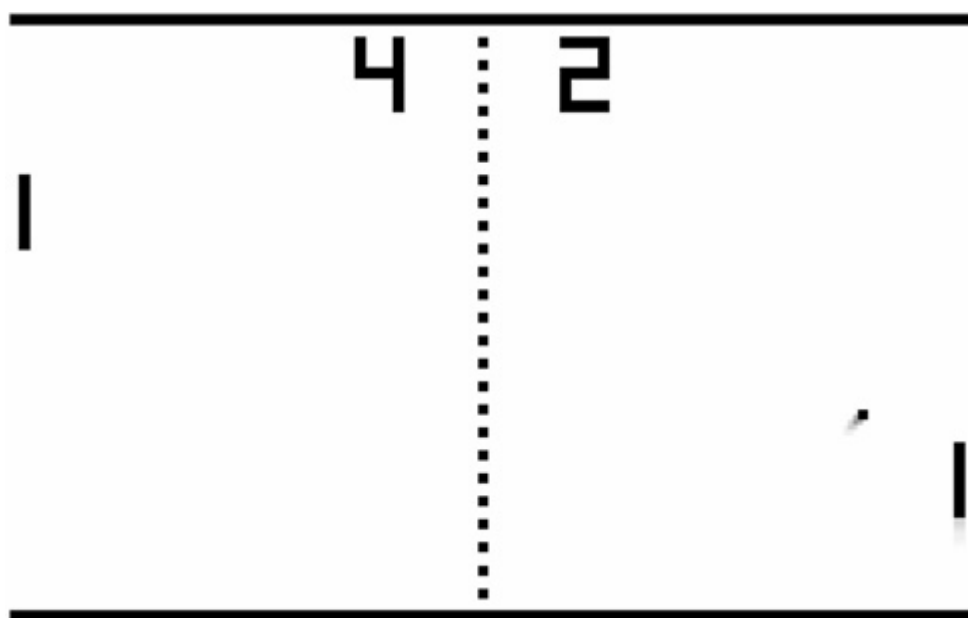


Рисунок 2.3 — Пример игры «Понг»

Идею пинг-понга для создания игры предложил Нолан Бушнелл своему сотруднику, программисту Аллану Алькорну. В то время у Аллана не было

опыта разработки игр, и Pong стал для него тренировочным проектом. Идея пинг-понга для видеоигр в то время уже была реализована в Magnavox Odyssey, и это привело к иску против Atari.

Игровой процесс оригинальной игры Понг

Небольшой квадратик, заменяющий пинг-понговый мячик, движется по экрану по линейной траектории. Если он ударяется о периметр игрового поля, то его траектория изменяется в зависимости от угла столкновения. Если шарик отбивается ракеткой игрока, то его движение дополнительно зависит от скорости и направления движения ракетки. Управление ракетками в Pong осуществляется с помощью paddle-контроллера, управляющего посредством поворота ручки вокруг своей оси. Периметр игрового поля обозначен краями экрана, а мячик не может покинуть поле через верхний или нижний край. В верхней части поля отображаются очки игроков, у каждого на своей половине экрана.

Игровой процесс состоит в том, что игроки передвигают свои ракетки вертикально для защиты своих ворот. В начале каждого раунда мячик подаётся одному из игроков, и раунд продолжается до тех пор, пока один из игроков не заработает очко. Это происходит тогда, когда его противник не может отбить мячик. Со временем игры скорость движения мячика постепенно увеличивается, и так игра усложняется. Особенностью игры является то, что ракетки не могут дойти до самого верха экрана и отбить мячик, если он туда попадает — в этом случае противнику засчитывается очко.

Первоначальная версия Pong разработана для двух игроков, когда каждый управляет своей ракеткой с помощью своего контроллера. В более поздних версиях игры стал доступен однопользовательский режим, когда одна из ракеток управляется компьютерным игроком. [19-20]

Причины выбора

Понг является простой с точки зрения правил и управления игрой, однако, как показала многолетняя практика, эта игра обладает увлекательным процессом, что позволит, с помощью несложного управления, требующего

мелкой моторики, удерживать внимание пациента на задаче использования разрабатываемого продукта в целях реабилитации и тренировки мелких точных движений рук. Для понимания процесса игры не нужно дополнительно изучать какие-либо сложные правила и особенности, и даже если человек встречается с Понг впервые при использовании разработанного в данной работе устройства, на полное понимание игрового процесса и перехода к стабильному использованию устройства в предназначенных целях не потребуется много времени.

Правила игры

Для начала определимся с общими обозначениями. Игровым полем является пространство, на котором происходит игра. Шариком называется движущийся по всему игровому полю объект шарообразного вида, на взаимодействии которого с другими объектами игрового поля и строится игра. Планками называются движущиеся объекты прямоугольного вида, которых на игровом поле два — одна контролируется игроком-пользователем, другая — противником. Количество очков показывает, насколько близки к победе пользователь и противник, и явно визуально отображается на игровом поле. Игровым раундом называется время свободного перемещения шарика и планок до того момента, как не произойдет событие, ведущее к изменению очков. Изменение очков происходит при касании шарика горизонтальных сторон экрана — «ворот». Игровой сессией называется время от начала игры до достижения условия выигрыша, которым, в свою очередь, является достижение одной из сторон установленного количества очков.

Введем подробное обозначение игрового поля. Игровым полем является все отображаемое пространство на экране, и границами игрового поля служат границы экрана. При этом вертикальные и горизонтальные границы экрана обладают разным значением с точки зрения правил. Также стоит отметить, что в середине игрового поля находится вертикальная линия, визуальное делящая игровое поле на две зоны: зону игрока и зону противника. Левая часть поля

является зоной игрока-пользователя устройства, в то время как правая сторона является полем противника.

Рассмотрим вертикальные границы (см. рис. 2.4). Они являются непроходимыми «стенами», при контакте с которыми мячик отталкивается, в соответствии с заданными правилами. Мячик не может выходить за вертикальные границы, и касание вертикальных границ не влияет на счет и не засчитывается как плюс или минус очко любой из сторон.

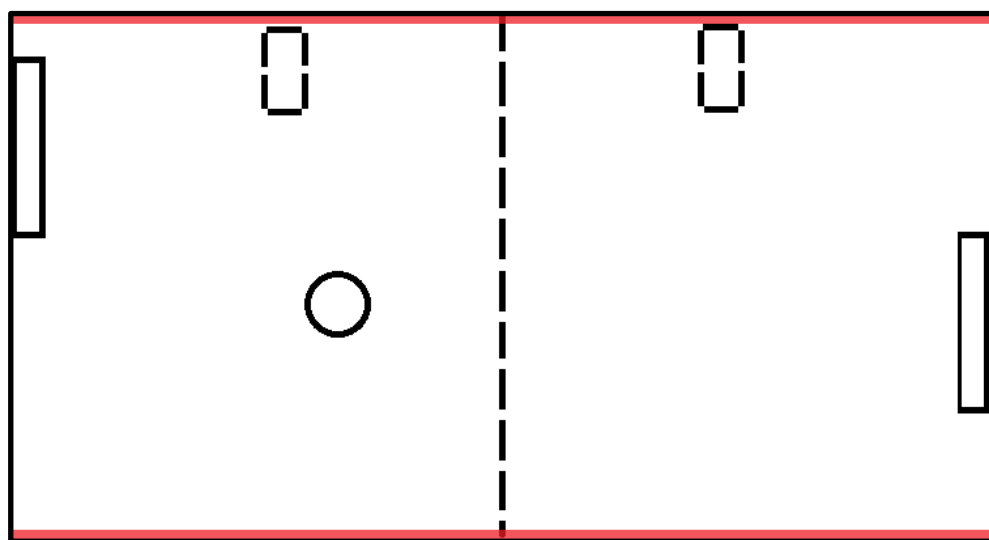


Рисунок 2.4 — Вертикальные границы игрового поля игры «Понг»

Рассмотрим вертикальные границы поля (см. рис. 2.5). В отличие от горизонтальных границ, при касании вертикальных шарик не отталкивается, так как касание вертикальные границы являются своего рода «воротами», которые защищают движущиеся планки. При касании горизонтальной границы происходит завершение раунда и перерасчёт очков: при касании «ворот» противника засчитывается плюс одно очко игроку, при касании «ворот» игрока — плюс одно очко противнику.

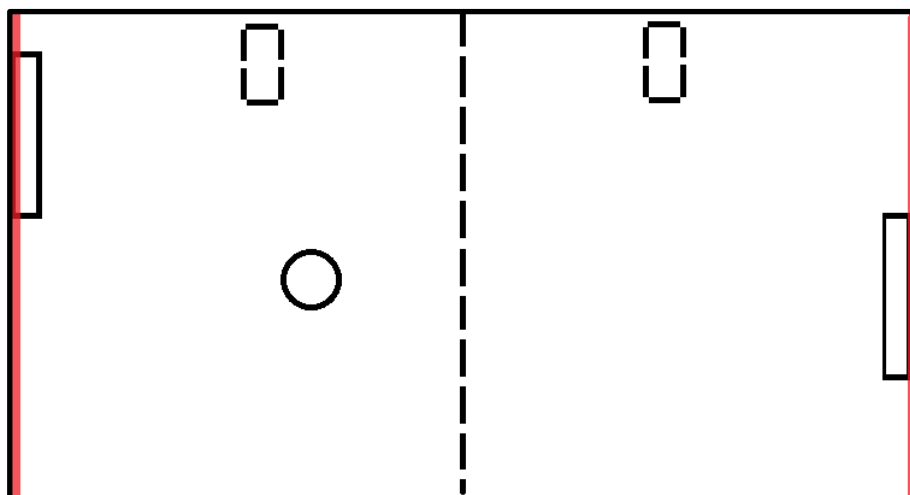


Рисунок 2.5 — Горизонтальные границы игрового поля игры «Понг»

Перейдем к такому важному элементу игры, как подсчет очков. Очки отображаются в верхней части экрана как два числа с левой и правой стороны от линии середины, соответствуя полю игрока и полю противника (см. рис. 2.6). Очко добавляется к стороне, противоположной той, чьей вертикальной границы поля коснулся шарик, после чего происходит переход на новый раунд и шарик возвращается на свое положение в начале игры. Стороны начинают с равным количеством очков (ноль очков), и игра заканчивается, когда какая-либо из сторон набирает установленное для победы количество очков.

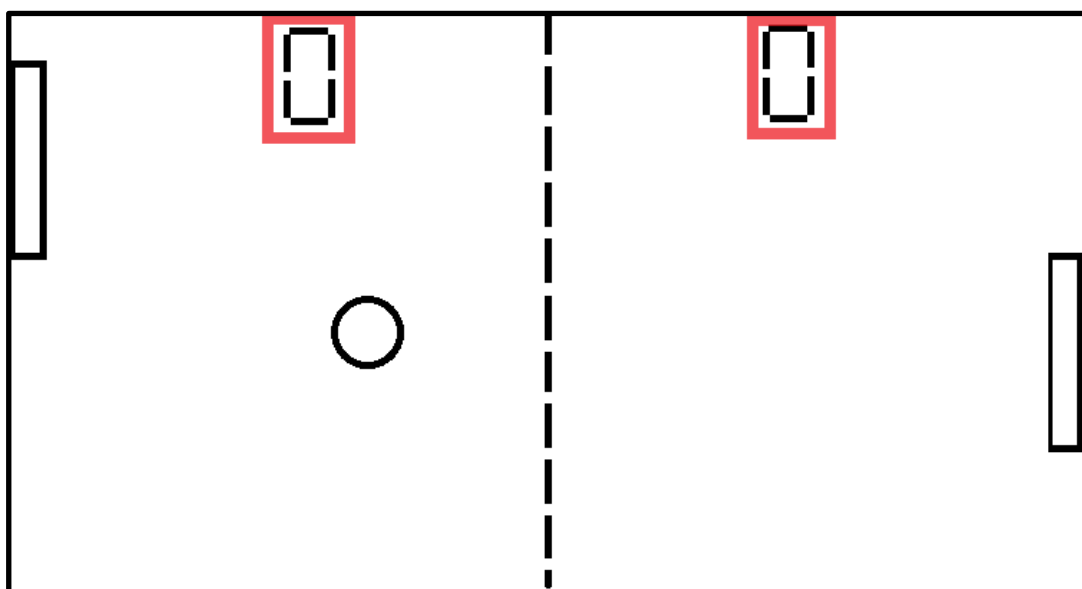


Рисунок 2.6 — Отображение очков при игре «Понг»

Движущиеся элементы. Движущимися элементами в игре является шарик и планки, возможные направления движения которых представлены на рисунке 2.7. Все элементы движутся по прямой линии, однако важно отметить, что шарик может двигаться в восьми направлениях, в то время как планки – только в двух. Ни один из элементов не может функционально выходить за границы игрового поля. Планка на левой стороне экрана контролируется игроком-пользователем, в то время как планка на правой стороне экрана контролируется программой. В данной работе планируется реализовать движение планки согласно установленному алгоритму, однако для конечной реализации продукта (при выходе за стадию прототипа) возможно развитие передвижения планки противника, например, путем разработки нескольких разных алгоритмов движения планки противника для разных уровней сложности, подключение искусственного интеллекта, подстраивающегося под действия игрока, и выполняющего наилучшее решение (в соответствии с заданным уровнем сложности) в любой момент игры, а также возможность управления планкой на правой стороне другим игроком, либо посредством добавления дополнительного комплекта контроллеров управления, либо по сети.

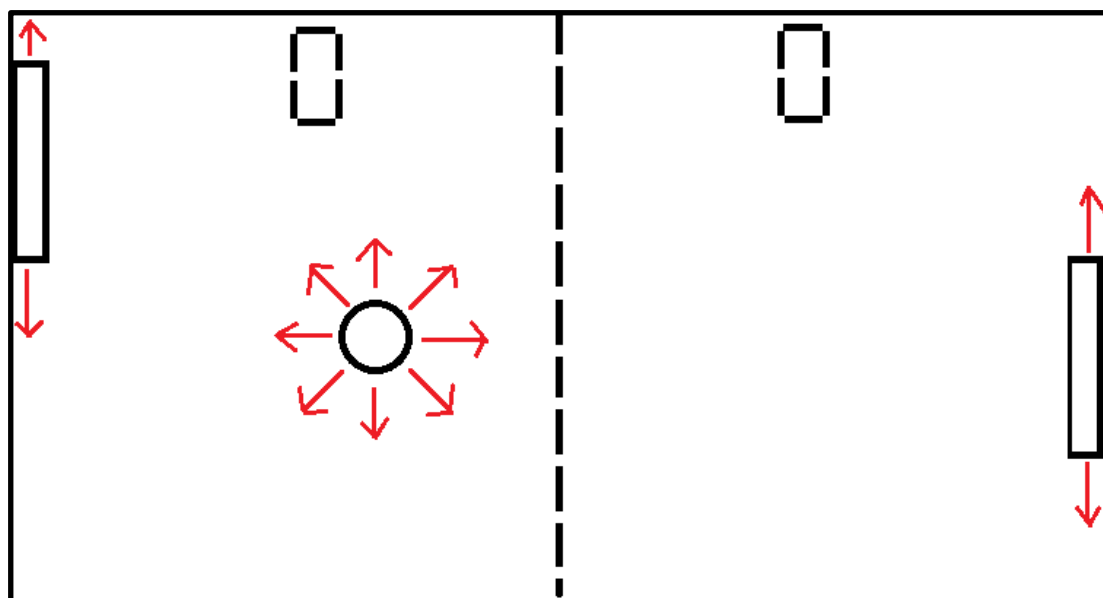


Рисунок 2.7 — Движущиеся объекты при игре «Понг»

Важно также отметить правила передвижения шарика при столкновении с препятствием (горизонтальной стеной или планкой). Шарик должен отталкиваться под тем же углом, под которым произошло столкновение (угол вычисляется от перпендикуляра поверхности до линии траектории шарика, для визуального примера см. рис. 2.8). Важно заметить, что при столкновении шарика под углом, или при столкновении с движущейся планкой происходит увеличение скорости, не настолько значительное, чтобы резко увеличить скорость игры, но достаточное, чтобы игра оставалась интересной. ИИ, управляющий планкой противника, не учитывает эту особенность, так что данный ход позволяет легко победить противника, однако это требует более точного контроля моторики. Таким образом эта особенность служит мотиватором для использования более тонких действий управления, что положительно влияет на главную задачу устройства: улучшение и тренировка мелкой моторики.



Рисунок 2.8 — Правила отражения мячика от планки или вертикальной границы игрового поля

Алгоритм игры

Была составлена блок-схема (см. рис. 2.9), объясняющая процесс работы игры и основные этапы, которые были описаны выше, такие как столкновение шарика с разными объектами и завершение игрового раунда при касании шарика горизонтальной границы, что приводит к обновлению счета, а также завершение игры при достижении какой-либо из сторон установленного количества очков. Данная схема позволяет установить общий алгоритм работы созданной для игры

программы, более детальное описание которого можно найти в третьей главе данной работы.

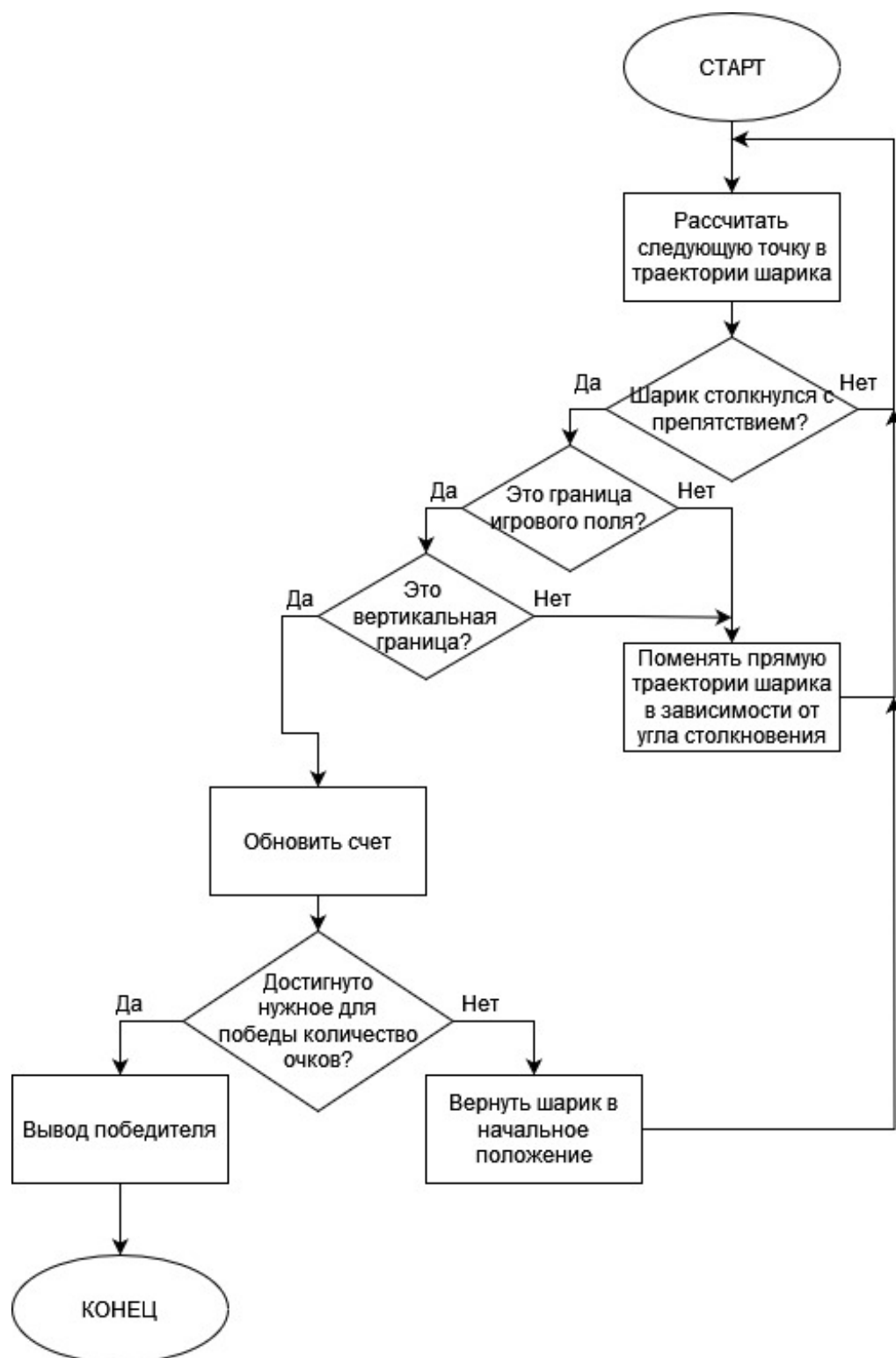


Рисунок 2.9 — Блок-схема концепта работы игры «Понг»

2.4. Описание игры «Скроллер»

Обзор игры

Сколлерами, или же сайд-скроллерами называются игры, наблюдение за игровым процессом осуществляется с боку. Главной особенностью, как видно из названия, является постоянное перемещение по двумерному игровому миру. Эти игры используют технологию скроллинга компьютерных дисплеев. Переход от игр с одним игровым экраном или резкой сменой игровых экранов (англ. flip-screen graphics) к играм, графика которых основана на скроллинге, произошел в течение золотого века аркадных игр и третьего поколения игровых приставок.

Одной из ответвлений скроллеров являются автоскроллер-игры. Это игры, в которых передвижение игрового мира происходит автоматически, и игрок должен подстраиваться под меняющийся мир, используя тот участок игрового пространства, который находится на экране в текущий момент. Данный подход позволяет реализовать множество разных игр: гонки, платформеры, шутеры и т.д..

Важно отметить, что существуют бесконечные скроллеры, где новый участок игрового мира генерируется случайным образом или же следуя какому-то алгоритму. Такие игры не имеют какой-либо конечной цели, кроме как сбор как можно большего числа очков. За что начисляются эти очки зависит от конкретной игры. [21-22]

Причины выбора

Поскольку скроллеры ориентированы на двумерный игровой мир, они не требуют такой сложной аппаратной мощности и требований к игровому устройству как трехмерные игры, а значит не займут много памяти, что позволит реализовать игру на существующих компонентах. Для реализации был выбран бесконечный автоскроллер, имеющий ограниченное управление. Благодаря этому достигается желаемое равновесие между простотой управления и увлекательностью игры. Также, благодаря системе очков и отсутствию верхнего

лимита будет достигнуто сохранение мотивации в течении игры — нет более сложного и интересного противника, чем ты сам.

Правила игры

Сутью игры является управление космическим кораблем, уклоняющимся от астероидов. Примерная графика игры представлена на рисунке 2.10.

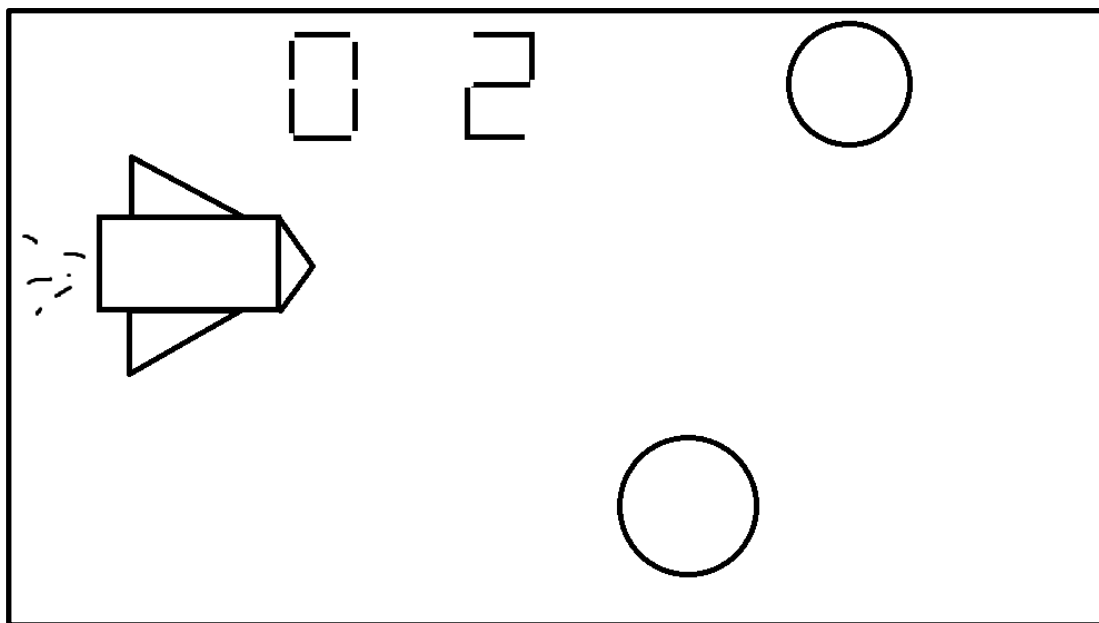


Рисунок 2.10 — Концептуальный вид игры «Скроллер»

Основные обозначения схожи с принятыми в игре «Понг» - игровым полем называется пространство, на котором происходит игра. Астероидами называются шарообразные объекты на экране. Целью игры является избежание столкновения с ними. Кораблем называется похожий на упрощенный космический корабль объект на экране, управление кораблем осуществляется игроком. Очки начисляются за каждый астероид, который удалось успешно обойти. Избежать столкновения возможно вовремя передвинув корабль вверх или вниз. Также на экране присутствует счетчик скорости, показывающий, на какой скорости корабль движется через космос, или же, переходя на более технический взгляд на игру, с какой скоростью появляются и двигаются на игрока астероиды. Игровой сессией называется время от начала до конца игрового процесса. Игровой процесс завершается при столкновении игрока с астероидом.

Аналогично игре «Понг» игровым полем является все отображаемое пространство на экране (см. рис. 2.11). При важно отметить две зоны: зона I является пространством, в котором может перемещаться игрок. Зоной II является зона появления астероидов.

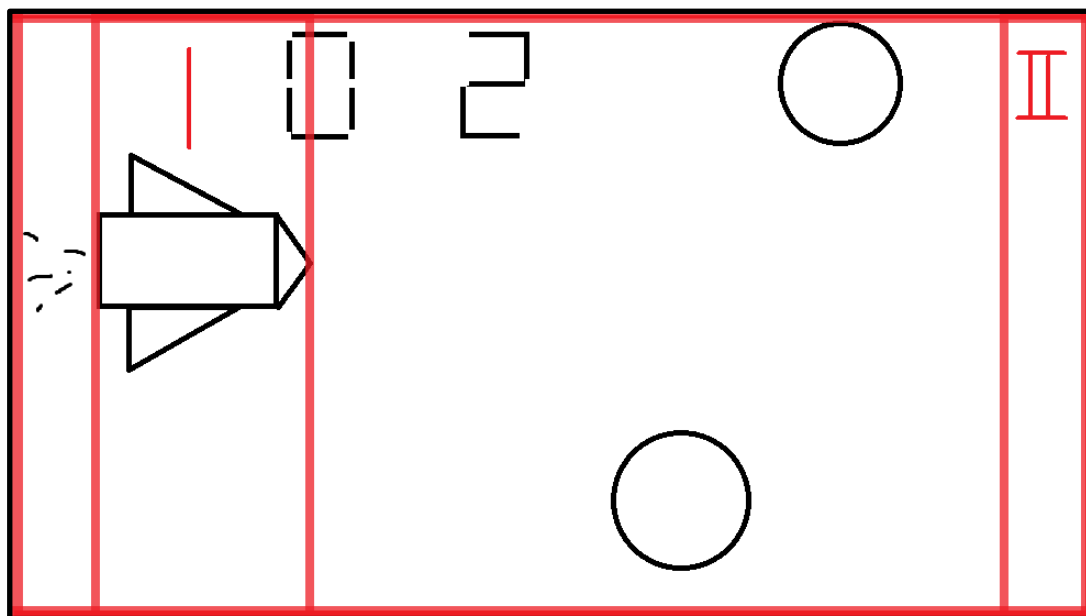


Рисунок 2.11 — Игровое поле при игре «Скроллер»

Рассмотрим цифры вверху экрана (см. рис. 2.12). Первая цифра – это количество очков. Начисление очков происходит в процессе игры и увеличивается на один с каждым астероидом, столкновение с которым удалось избежать. Астероид считается удачно обойденным, когда исчезает за левой границей экрана. Второе число показывает текущую скорость корабля. В базовой реализации игры со течением игрового процесса скорость увеличивается при увеличении количества очков, набранного игроком. Однако возможна реализация с постоянной скоростью, что упрощает игровой процесс, приводя к тому, что продолжительность игры напрямую зависит от точности управления, убирая влияния скорости восприятия, играющей роль при большой скорости игры. Данный режим может быть полезен для мониторинга процесса развития мелкой моторики, однако может быть не так интересен, как увеличивающаяся скорость, и приведет к снижению мотивации использования устройства. При конечной реализации продукта возможно, будет полезным включить режим

постоянной скорости как возможную опцию игры, однако в данной реализации будет использоваться увеличивающаяся со временем скорость.

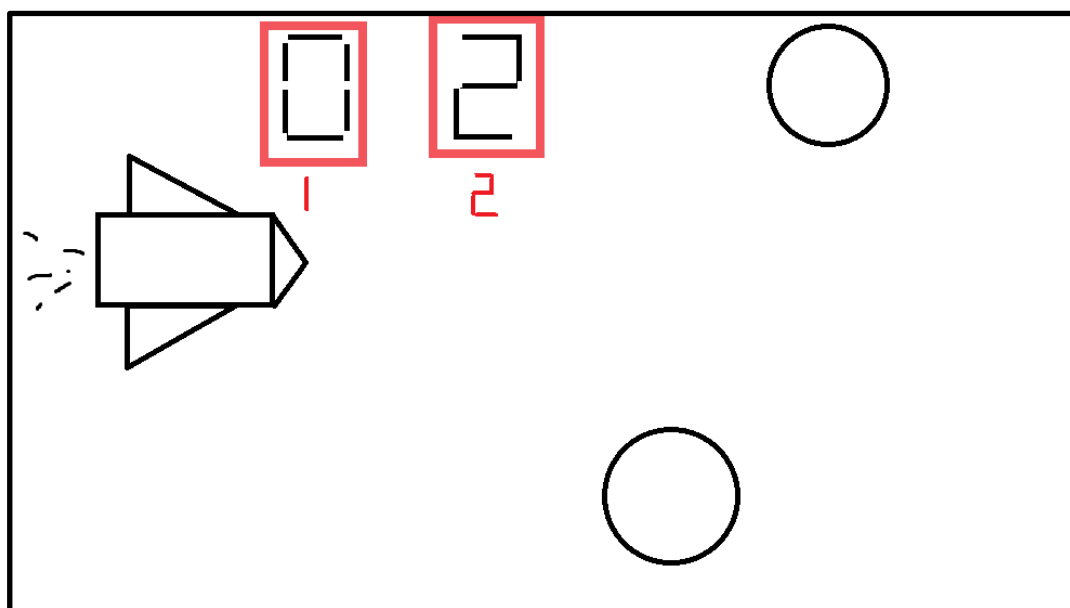


Рисунок 2.12 — Отображение очков и скорости при игре «Скроллер»

Рассмотрим движущиеся элементы в игре (см. рис. 2.13). Ими являются корабль и астероиды. Корабль может двигаться вверх или вниз в границах игрового поля, в то время как астероиды движутся только в одном направлении в горизонтальной плоскости, приближаясь к кораблю. Постоянное движение астероидов создает ощущение меняющегося игрового пространства, таким образом создавая иллюзию, что корабль движется мимо астероидов, которые остаются на месте. Дополнительным элементом, позволяющим поддерживать эту иллюзию, являются «следы топлива», остающиеся за кораблем и колеблющиеся, следуя алгоритму.

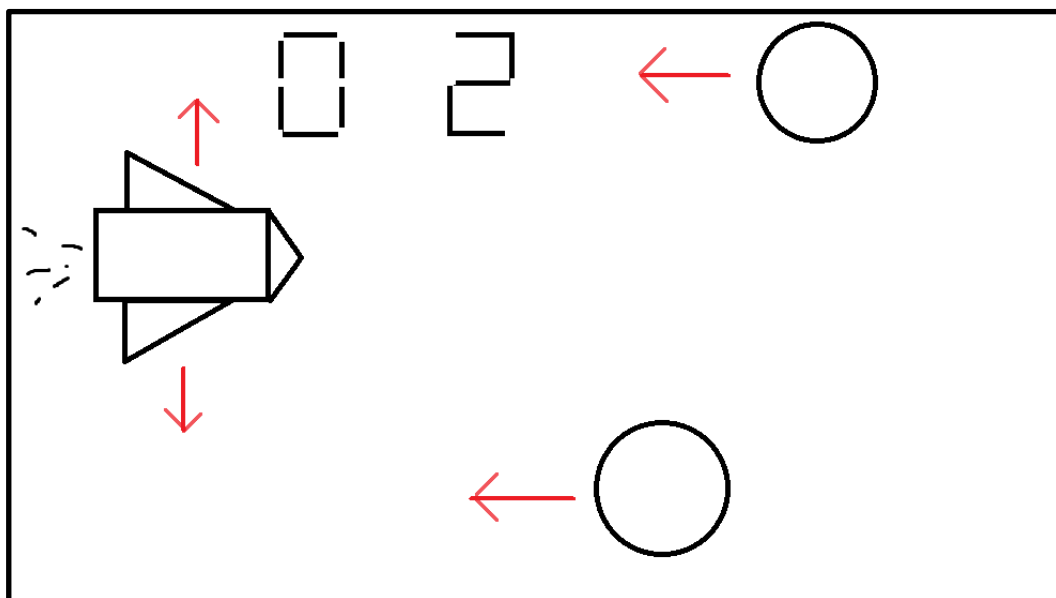


Рисунок 2.13 — Движущиеся объекты и их возможные направления движения в игре «Скроллер»

Завершение игры происходит при столкновении корабля с астероидом, после чего появляется экран, сообщающий о завершении игры и показывающий финальный счет. Пример момента завершения игры показан на рисунке 2.14, где красным выделен астероид, столкновение с которым и привело к завершению.

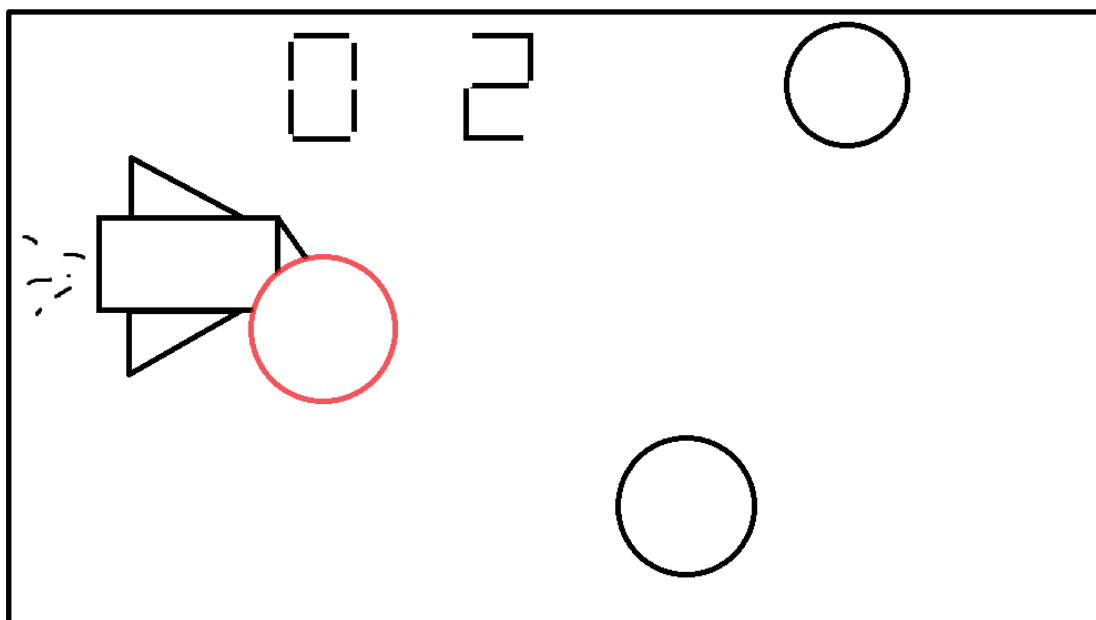


Рисунок 2.14 — Пример момента завершения игры «Скроллер»

Алгоритм игры

Была составлена блок-схема (см. рис. 2.15), объясняющая процесс работы игры и основные этапы, которые были описаны выше, такие как перемещение астероидов, увеличение количества очков и столкновение астероидов с кораблем, что приводит к завершению игры. Данная схема позволяет установить общий алгоритм работы созданной для игры программы, более детальное описание которого можно найти в третьей главе данной работы.

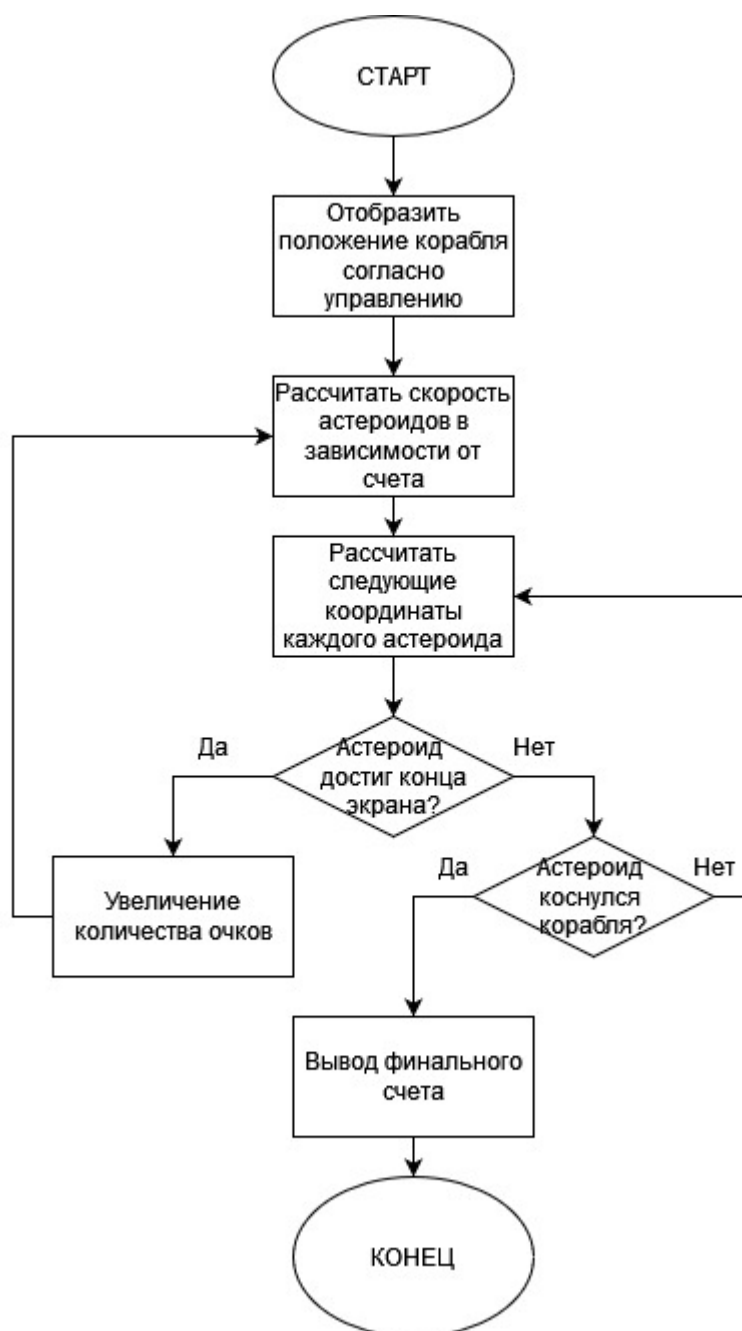


Рисунок 2.15 — Блок-схема концепта работы игры «Скроллер»

Выводы по главе

В ходе данной главы была выделена структура устройства. Были выяснены основные компоненты, необходимые для его создания, и их назначения: плата Arduino для связи всех компонентов и реализации программной части, потенциометр и кнопка как устройство ввода и OLED экран как устройство вывода.

Были составлены основные алгоритмы для работы устройства, включая общий алгоритм работы устройства и особенности программируемых игр.

В результате данной работы была закончена теоретическая подготовка к проектированию и программированию устройства.

3. ПРАКТИЧЕСКИЕ АСПЕКТЫ РАЗРАБОТКИ

3.1. Аппаратная часть

Схема устройства приведена на рисунке 3.1. Для реализации данного устройства необходимо, в первую очередь, плата Arduino Nano (1), кнопка (2) и потенциометр (3), являющиеся средствами управления, и экран OLED Display (4) и провода. Это позволит построить прототип устройства, приведенный на рисунке 3.2. В виду временной нехватки компонентов кнопка была реализована в виде двух проводов, замыкание которых имитирует работу кнопки.

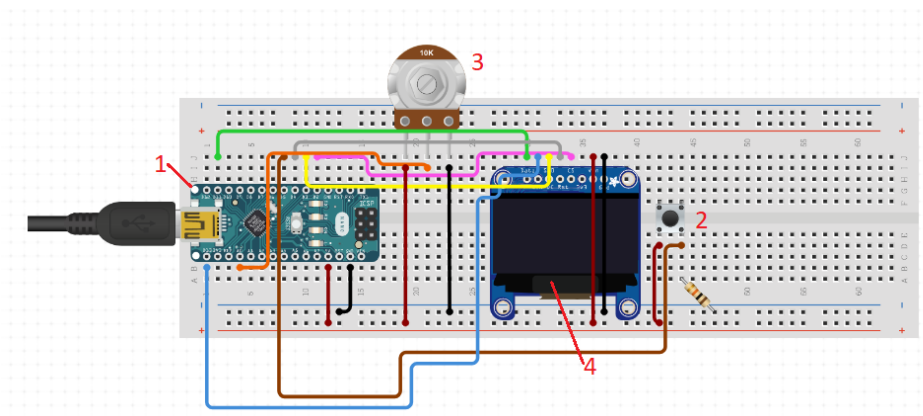


Рисунок 3.1 — Схема подключения компонентов

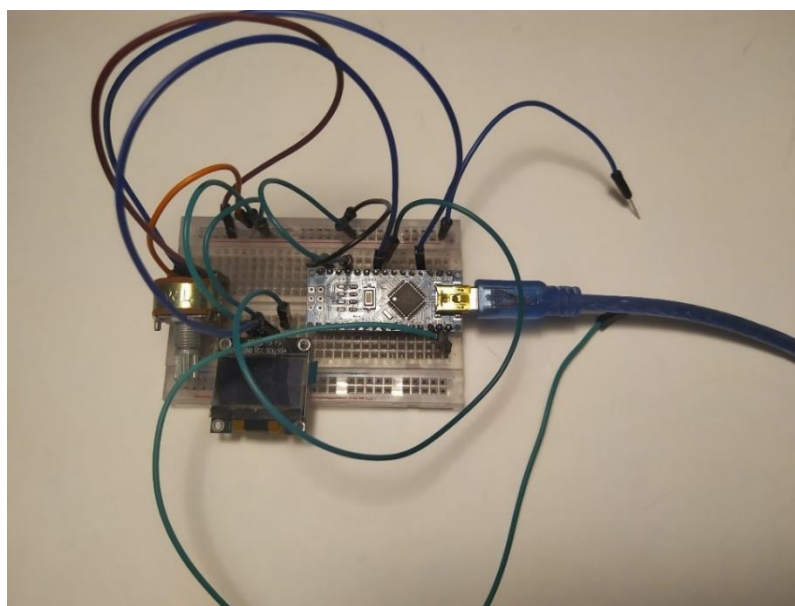


Рисунок 3.2 — Собранный прототип устройства

Используя данную схему соединений, можно в дальнейшем спроектировать разводку платы (см. рис. 3.3), провести трассировку и спроектировать печатную плату для устройства (см. рис. 3.4-3.5)

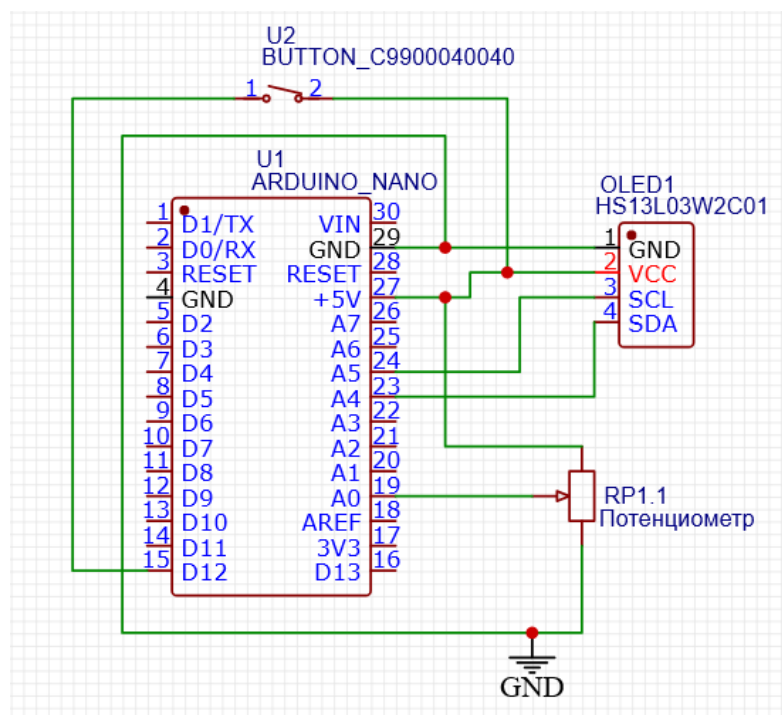


Рисунок 3.3 — Проектирование разводки платы, схема подключений

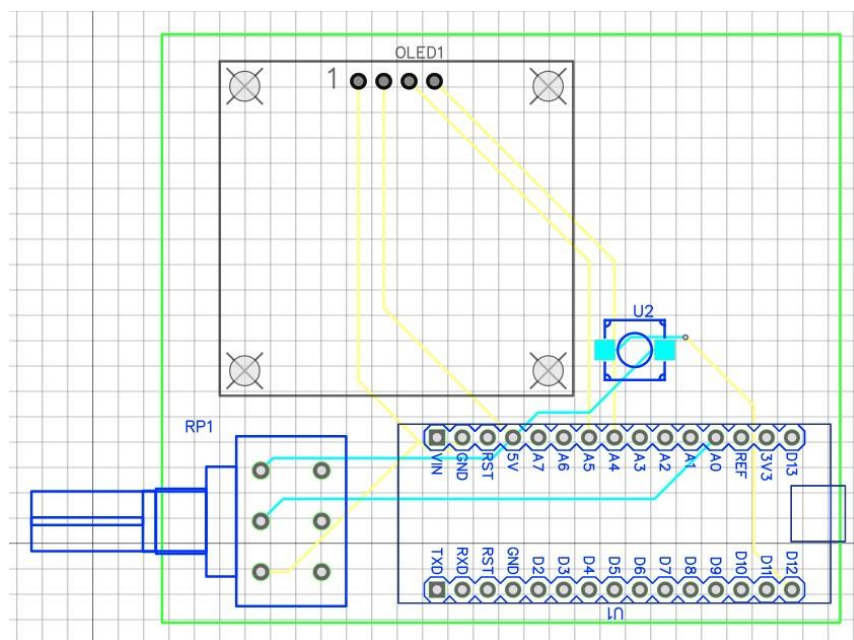


Рисунок 3.4 — Схема печатной платы после трассировки

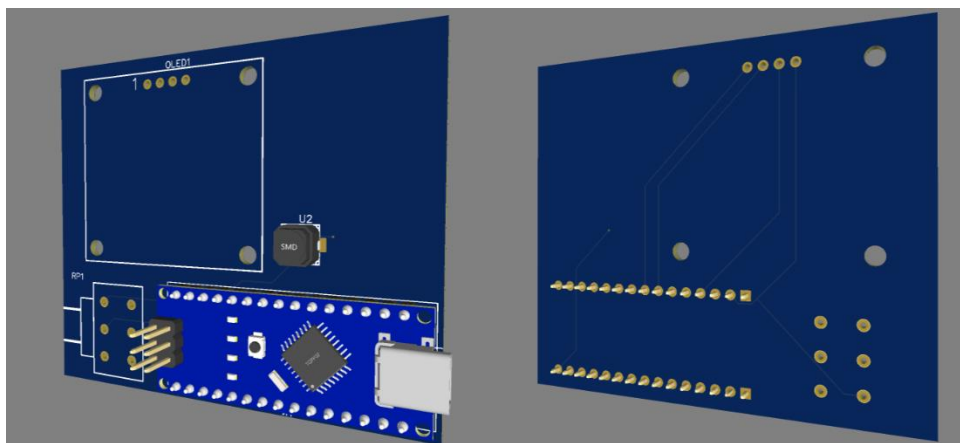


Рисунок 3.5 — 3D модель спроектированной печатной платы

Также для дальнейшего развития проекта возможно добавление автономного режима питания от аккумулятора. Передача данных в таком случае будет реализована с помощью использования Bluetooth-модуля. Схема соединения компонентов приведена на рисунке 3.6. Аналогично с рисунком 22 плата Arduino Nano обозначена 1, кнопка – 2, потенциометр – 3, экран OLED Display – 4. Bluetooth-модуль обозначен цифрой 5, а аккумулятор – 6.

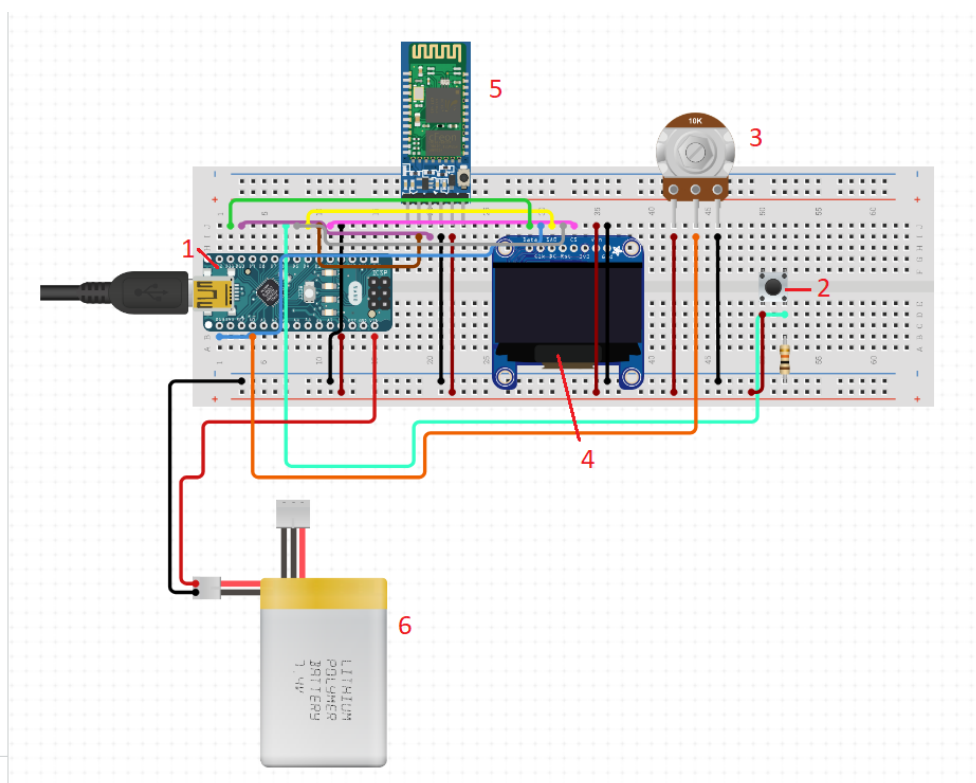


Рисунок 3.6 — Схема подключения компонентов для улучшенного варианта устройства

3.2. Меню выбора игры

При включении устройства появляется меню выбора игры, где, используя потенциометр для навигации и кнопку для подтверждения выбора, выбирается игра. После этого устройство переходит к выполнению алгоритма выбранной игры. (см. рис. 3.7)

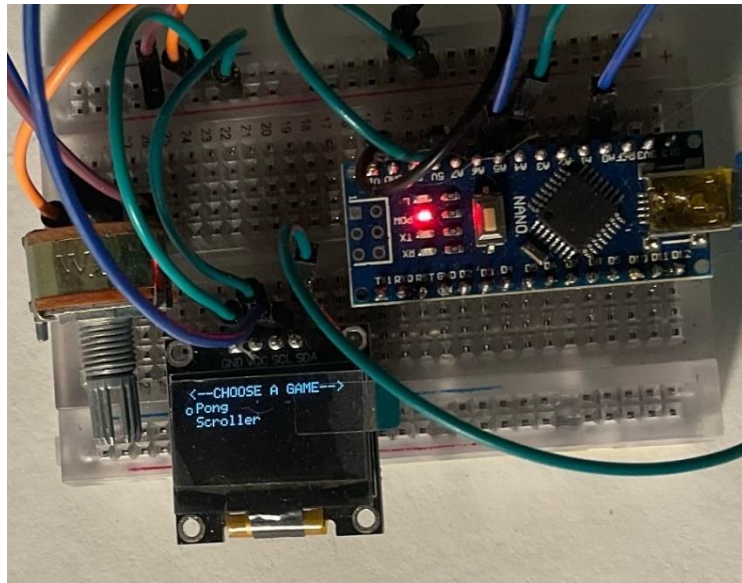


Рисунок 3.7 — Отображение меню на прототипе

Для отображения меню используются функции `print` (для отображения текста и названий игр) и `drawCircle` (для отображения кружка, символизирующего, какая игра будет выбрана при нажатии кнопки в данный момент) библиотеки `OLED_I2C`. Отрисовка меню осуществляется внутри основной петли работы устройства при условии, что игра не была выбрана, то есть кнопка не была нажата. (см. листинг 3.1)

Листинг 3.1 — Функции, отвечающие за работу меню

```
void drawMenu(){
    myOLED.clrScr();
    myOLED.print("<--CHOOSE A GAME-->", 5,0);
    myOLED.print("Pong",10,10);
    myOLED.print("Scroller",10,20);}
void loop() {
    while (gamenotchosen)
    {
```

```

    drawMenu();
    position = analogRead(A0);
    if (position<=512) {
        myOLED.drawCircle(5, 15, 2);
    } else {
        myOLED.drawCircle(5, 25, 2);
    }
    ...

```

3.3. Программная реализация игры «Понг»

Для начала нужно убедиться, что подключены необходимые для работы игры библиотеки. Основной библиотекой для работы является OLED_I2C, позволяющая управлять OLED экраном. Данная библиотека имеет преимущество перед такими библиотеками как Adafruit_SSD1306, поскольку не приводит к задержкам при постоянном изменении изображения и позволяет более точно присваивать значения пикселям экрана, что приводит к более плавно выглядящей игре.

После этого зададим необходимые для игры переменные. resolution отвечает за размер игрового поля и должно быть равно масштабам дисплея. ball отвечает за координаты игрового мячика. WALL_WIDTH отвечает за размер (видимый) горизонтальных стен, PADDLE_WIDTH – за размер планки, BALL_SIZE – размер мячика, SPEED – скорость движения мячика. Данные переменные являются константами, определяющими общую сложность и вид игры. При необходимости, через изменение этих параметров можно добиться увеличения или уменьшения сложности.

Далее идут переменные, задающие, собственно, начальный счет игрока, начальный счет противника (управляемого компьютером), позиция планки игрока и противника соответственно. Также введены сокращения для направления движения мяча и задана переменная, позволяющая определить, в процессе ли игра, или закончена. (см. листинг 3.2)

Листинг 3.2. — Задание переменных для игры Понг

```
int resolution[2] = { 128, 64 }, ball[2] = { 20, (resolution[1] / 2) };

const int PIXEL_SIZE = 10, WALL_WIDTH = 2, PADDLE_WIDTH = 4, BALL_SIZE = 5, SPEED = 2;
int playerScore = 0, aiScore = 0, playerPos = 0, aiPos = 0;
char ballDirectionHori = 'R', ballDirectionVerti = 'S';
boolean inProgress = true;
```

Рассмотрим функции, позволяющие отобразить или стереть мячик. Это довольно простые функции, получающие на вход координаты и рисующие, с помощью библиотеки `OLED_I2C` на указанных координатах окружность, которая соответствует игровому мячику. Библиотека имеет функции как для отрисовки, так и для удаления окружности, однако для более удобной читаемости кода было решено занести их в отдельные функции, присвоив имена, переводящиеся как «Нарисовать мячик» и «Стереть мячик» соответственно. Передвижение мячика на экране реализуется чередованием этих двух функций — при начале движения мячик «стирается» с места, где он был, и рисуется на новом месте. (см. листинг 3.3.)

Листинг 3.2 — Функции, отвечающие за передвижение мячика

```
void drawBall(int x, int y) {
    myOLED.drawCircle(x, y, BALL_SIZE);
}

void eraseBall(int x, int y) {
    myOLED.clrCircle(x, y, BALL_SIZE);
}
```

Функции для отображения и удаления плашек игрока и противника работают аналогичным образом, однако полагаются не на встроенную функцию библиотеки, а на пользовательскую. В листинге 3.3 представлены функции для планки игрока, в Приложении А можно найти функцию для планки противника.

Листинг 3.3 — Функции, отвечающие за передвижение планки

```
void erasePlayerPaddle(int row) {
    erasePixel(0, row - (PADDLE_WIDTH * 2), PADDLE_WIDTH);
    erasePixel(0, row - PADDLE_WIDTH, PADDLE_WIDTH);
    erasePixel(0, row, PADDLE_WIDTH);
}
```

```

    erasePixel(0, row + PADDLE_WIDTH, PADDLE_WIDTH);
    erasePixel(0, row + (PADDLE_WIDTH + 2), PADDLE_WIDTH);
}

void drawPlayerPaddle(int row) {
    drawPixel(0, row - (PADDLE_WIDTH * 2), PADDLE_WIDTH);
    drawPixel(0, row - PADDLE_WIDTH, PADDLE_WIDTH);
    drawPixel(0, row, PADDLE_WIDTH);
    drawPixel(0, row + PADDLE_WIDTH, PADDLE_WIDTH);
    drawPixel(0, row + (PADDLE_WIDTH + 2), PADDLE_WIDTH);
}

void drawPixel(int posX, int posY, int dimensions) {
    // draw group of pixels
    for (int x = 0; x < dimensions; ++x) {
        for (int y = 0; y < dimensions; ++y) {
            myOLED.setPixel((posX + x), (posY + y));
        }
    }
}

void erasePixel(int posX, int posY, int dimensions) {
    // erase group of pixels
    for (int x = 0; x < dimensions; ++x) {
        for (int y = 0; y < dimensions; ++y) {
            myOLED.clrPixel((posX + x), (posY + y));
        }
    }
}

```

Важной функцией является функция передвижения планки противника. В данной работе реализовано простое поведение: если мячик по оси Y находится выше планки, планка двигается вверх, если ниже – вниз. Одним из вариантов усложнения игры может быть ускорение планки, управляемой компьютером. Для этого, там, где в коде используется ++aiPos и –aiPos нужно изменить это на aiPos+=2, aiPos-=2 или больше, в зависимости от требуемой скорости. [23] (см. Приложение А)

Вид игры на прототипе предоставлен на рисунке 3.8.

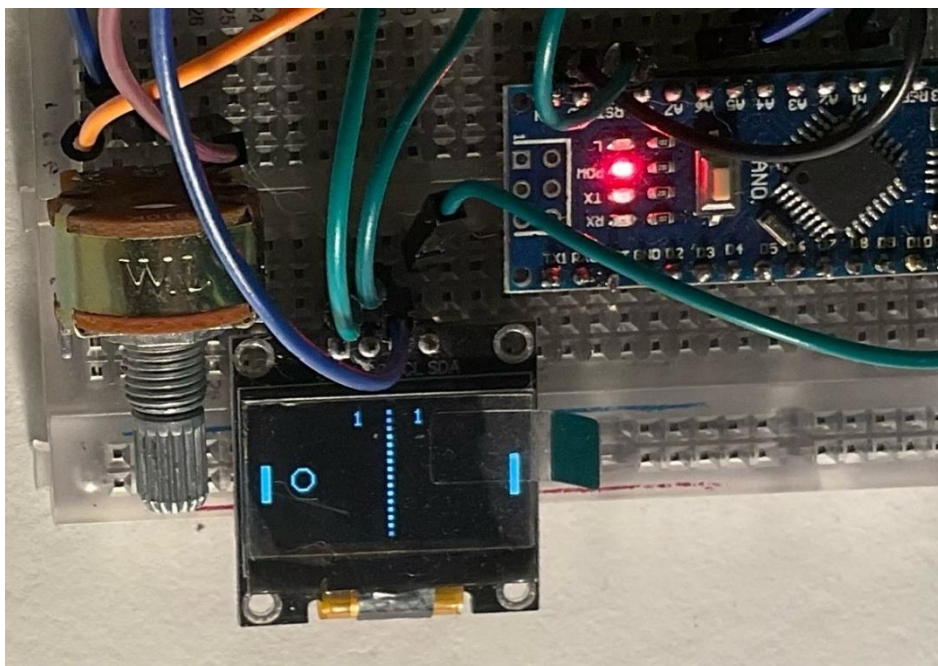


Рисунок 3.8 — Вид игры «Скроллер» на прототипе

Блок-схема, описывающая работу алгоритма, предоставлена на рисунке 3.9.

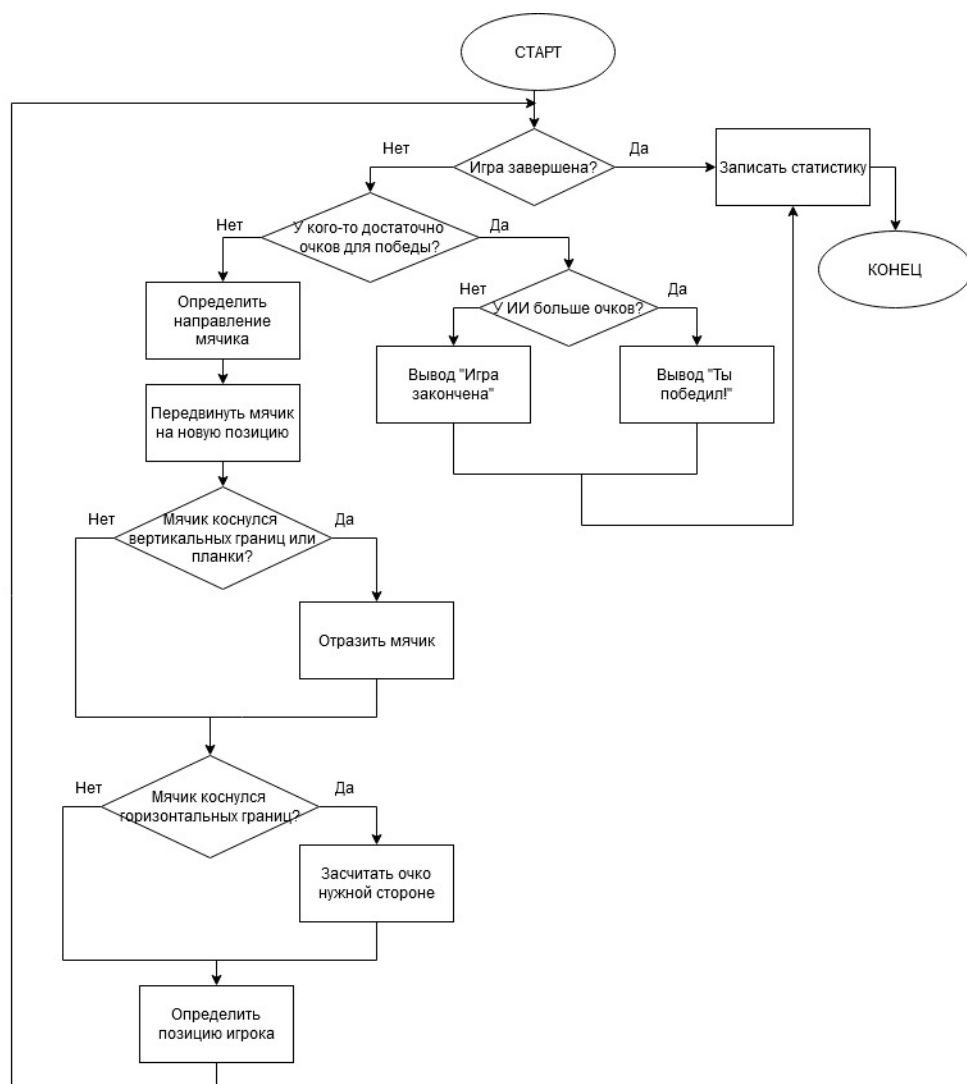


Рисунок 3.9 — Блок-схема игры «Понг»

Процесс игры заключен в цикл, выход из которого совершается при достижении какой-либо стороной нужного для победы количества очков. Это нужно для корректной работы постоянного движения мячика и платформ. В начале цикла проверяется количество очков каждой из сторон, и происходит обработка победы одной из сторон в случае достижения этой стороной нужного количества очков, после чего записывается статистика игры, и функция, выполняющая игру, завершается.

В случае продолжение процесса игры происходит расчет направления движения мячика, в соответствии с которым определяется позиция мячика на данном ветке цикла, согласно установленной скорости движения. После этого проверяется, столкнулся ли мячик на новой позиции с вертикальной границей

или планкой (игрока или противника). Если это произошло, активируется функция отражения, рассчитывающая конечные координаты мячика на данном витке цикла после отражения от преграды. Также проверяется, не столкнулся ли мячик с горизонтальными границами. В случае, если это произошло, происходит перерасчет очков и стороне, противоположной той, чьей горизонтальной границы коснулся мячик, засчитывается очко. Также в цикле постоянно считывается информация с устройства ввода, которая определяет положение игрока на текущем ветке цикла. После этого цикл повторяется.

3.4. Программная реализация игры «Скроллер»

Одной из основных функций является функция отображения корабля. Данная функция получает на вход координаты x и y , вокруг которых, с помощью графических примитивов — прямоугольника и прямых линий — и строится изображение, стилистически напоминающее космический корабль, или же ракету. Благодаря обработке данной функцией не только координаты y , но и x , возможно усложнение управления при последующим развитии проектируемого устройства, путем добавления кнопок, ещё одного реостата или иного устройства ввода, которое будет отвечать за перемещение корабля по горизонтали. (см. листинг 3.4)

Листинг 3.4 — Функция отображения корабля

```
void drawRocket(int x,int y){
    myOLED.drawRect(x,y,x+rocketLenght,y+rocketHight);
    myOLED.drawLine(x+rocketLenght,y+rocketHight,x+rocketLenght+5,y+rocketHight/2);
    myOLED.drawLine(x+rocketLenght,y,x+rocketLenght+5,y+rocketHight/2);
    myOLED.drawLine(x+5,y+rocketHight,x-2,y+8); // Down
    myOLED.drawLine(x,y+rocketHight,x-2,y+8);
    myOLED.drawLine(x+5,y,x-2,y-4); // Up
    myOLED.drawLine(x,y,x-2,y-4);
}
```


Функция отрисовки пламени создает пиксели в случайных позициях в указанном диапазоне координат за кораблем, таким образом создавая иллюзию, что корабль постоянно движется. (см. листинг 3.5)

Листинг 3.5 — Функция отображения пламени

```
void drawFlame(int x,int y){  
    for(int i = 0;i<= intensFire; i++){  
        pixX = random(x-flameR,x);  
        pixY = random(y-2,y+flameR);  
        myOLED.setPixel(pixX,pixY);  
    }  
}
```

Для отрисовки астероидов используется стандартная функция библиотеки OLED_I2C. Перемещение астероидов происходит в основном цикле игры (см. Приложение А), путем вычитания из координаты x установленного значения скорости. Таким образом происходит постоянное движение к левой стороне экрана. [24]

Вид игры на прототипе предоставлен на рисунке 3.10

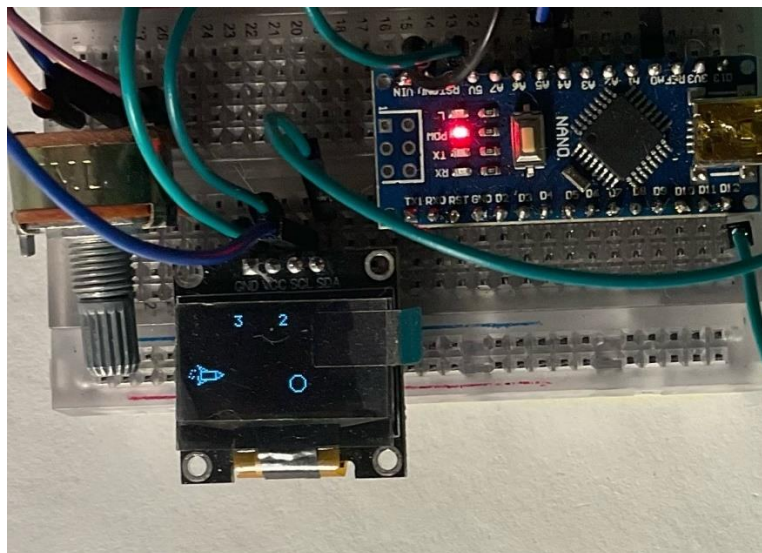


Рисунок 3.10 — Вид игры «Скроллер» на прототипе

Блок-схема, описывающая работу алгоритма, предоставлена на рисунке 3.11

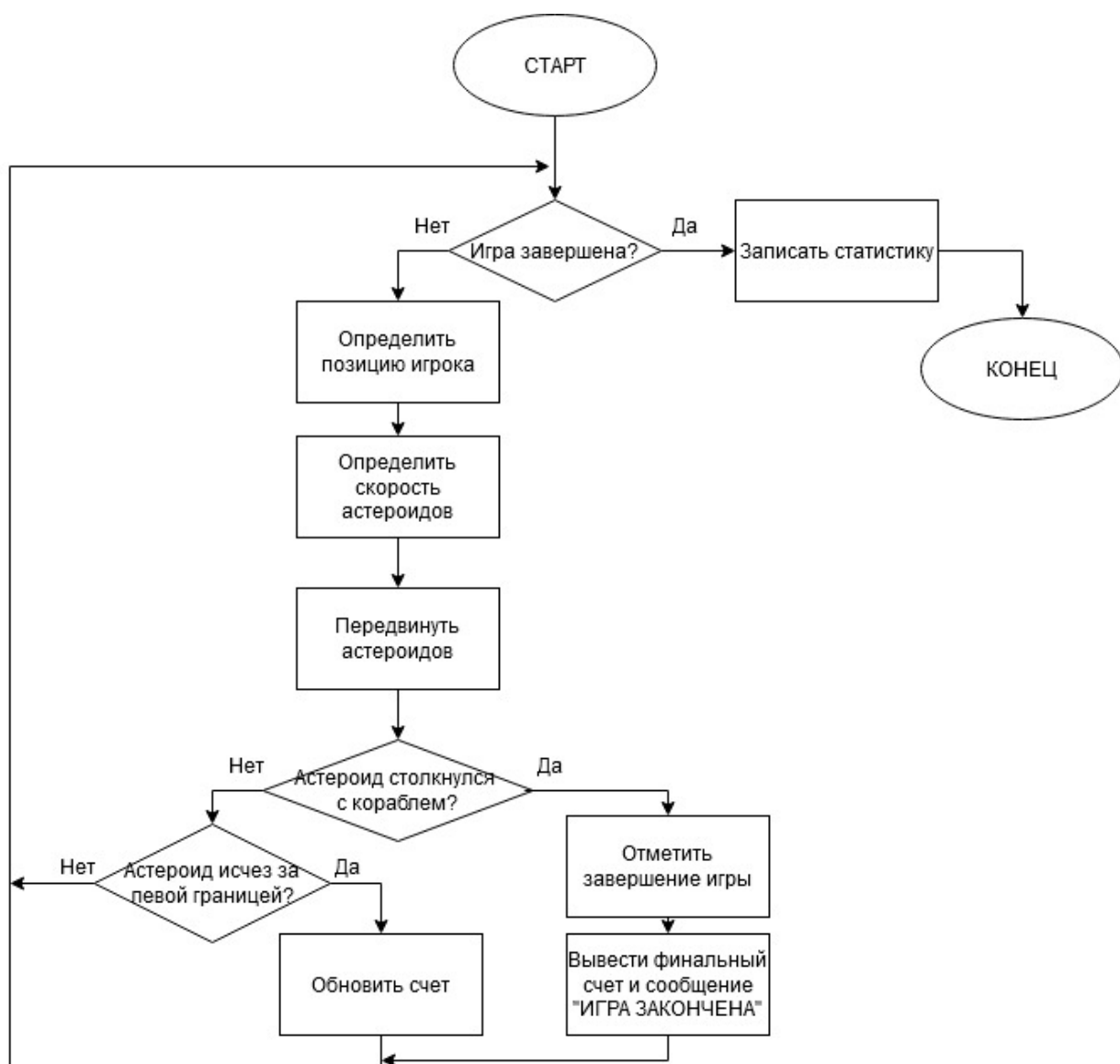


Рисунок 3.11 — Блок-схема игры «Скроллер»

Для достижения нормального отображения постоянного движения астероидов и корректной реакции программы на перемещение корабля, контролируемое пользователем, основной процесс игры заключен в цикл, выход из которого символизирует окончание игры, после чего происходит сбор статистики и завершение функции, отвечающей за игровой процесс.

В случае продолжении игры происходит считывание информации с устройства ввода, в соответствии с которой определяется положение корабля, управляемого игроком. После этого определяется скорость астероидов в соответствии с набранными очками и установленным коэффициентом

увеличения скорости. Учитывая рассчитанную скорость и предыдущее положение астероида (если таковое имеется), рассчитываются координаты астероида на данном витке цикла. Путем удаления астероида с его предыдущего места (если таковое имеется) и отрисовки его на координатах текущего цикла происходит передвижение астероида.

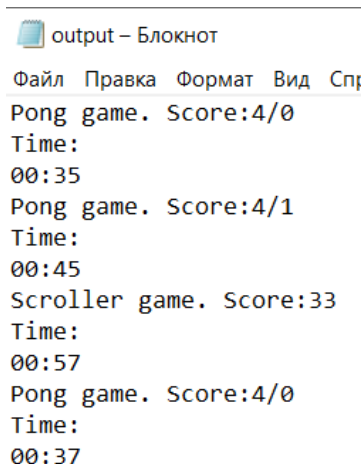
После передвижения астероида происходит сравнение координат корабля с координатами астероида (с учетом размеров астероида и корабля). При пересечении границ корабля с астероидом происходит столкновение, которое приводит к завершению игры. Выводиться сообщение о завершении игры и финальный счет, а также отмечается достижение условий завершения игры, что послужит сигналом выхода из цикла на следующем витке. В случае, если столкновения не произошло, проверяется, исчез ли астероид полностью за левой границей (то есть координата x меньше, чем $0 - \text{радиус_астероида}$). Если астероид полностью скрылся за левой границей, игроку добавляется очко и происходит обновление счета. После этого цикл повторяется.

3.5. Сбор статистики

Сбор статистики с Arduino осуществляется с помощью вывода на серийный монитор с помощью команд библиотеки Serial. Однако Arduino IDE и плата Arduino без дополнительных модулей не обладает функционалом сохранения данных в отдельный файл. Поскольку разрабатываемое устройство является лишь прототипом, направленным в первую очередь на демонстрацию возможностей, то было решено прибегнуть к самому простому способу: использования дополнительной программы записи с порта.

С помощью команд `Serial.print` и `Serial.println` выводятся данные об игровой сессии: ее продолжительность и итоговый счет. Используя программу CoolTerm была установлена запись информации, которую подает на порт Arduino, в текстовый файл. При дальнейшей разработке возможно автоматизация сбора

статистики путем установки дополнительных модулей: модуль с накопителем для офлайн сбора информации, и/или Bluetooth модуль для удаленного хранения данных. Пример, демонстрирующий вид статистики, предоставлен на рисунке 3.12.



```
output - Блокнот
Файл  Правка  Формат  Вид  Справка
Pong game. Score:4/0
Time:
00:35
Pong game. Score:4/1
Time:
00:45
Scroller game. Score:33
Time:
00:57
Pong game. Score:4/0
Time:
00:37
```

Рисунок 3.12 — Пример статистики

Для сохранения статистики в текстовый файл необходимо после установки соединения вручную (Connection-Capture Text/Binary File) начать сбор данных в файл. Однако программа также позволяет настроить автоматический сбор статистики при запуске сценария. Для были использованы настройки, указанные на рисунке 3.13. Благодаря им в указанной папке создается файл с указанным именем, в который позже добавляются все полученные сведения. Данный файл можно без труда открыть любым текстовым редактором.

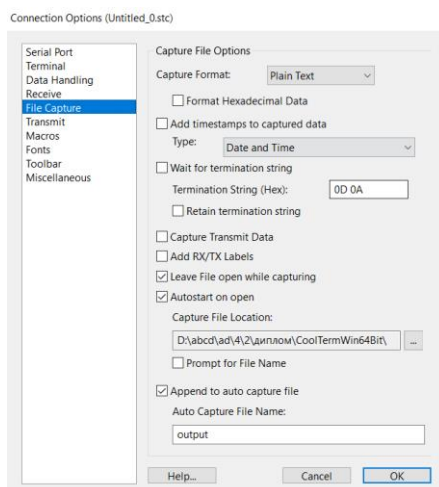


Рисунок 3.13 — Настройки для автоматического получения статистики

В ходе тестирования были собраны данные для 10 попыток в игры «Понг» и «Скроллер». Полученные данные приведены в таблице 3.1 и 3.2.

Таблица 3.1 – Статистика игры «Понг»

№ Попытки	1	2	3	4	5	6	7	8	9	10
Время (мин:сек)	00:38	00:43	00:37	00:47	00:42	00:35	00:45	00:37	00:35	00:37
Счет	4/2	4/0	4/0	4/1	4/1	4/0	4/1	4/0	4/0	4/0

Таблица 3.2 – Статистика игры «Скроллер»

№ Попытки	1	2	3	4	5	6	7	8	9	10
Время (мин:сек)	00:57	01:24	00:36	00:40	01:17	00:44	00:43	00:47	00:53	00:53
Счет	33	66	14	17	55	20	19	22	27	27

В результате анализа полученной статистики можно сделать вывод о том, что игровые сессии редко занимают больше минуты, и что для человека без проблем мелкой моторики не составляет труда быстро разобраться в управлении и добиться победы. По полученным данным видно, что после нескольких попыток с разным результатом в конце появляется некоторое среднее значение времени и счета, связанное с навыками тестирующего. По этому значению можно судить о развитии навыков мелкой моторики, а также о скорости реакции. Эти данные можно использовать чтобы корректировать общий уровень сложности при необходимости. Появление этого среднего значения обуславливается привыканием тестирующего к управлению и правилам игры.

Вывод по главе

В результате данного раздела было получен работающий прототип устройства. Было сконструирована и собрана аппаратная составляющая игровой

консоли, включающая в себя плату Arduino, потенциометр, кнопку, OLED экран и соединительные провода. Данный прототип предусматривает работу при подключении к персональному компьютеру. Также была разработана схема для варианта прототипа, работающего автономно, получающего питание от аккумулятора и посылающего статистику с помощью Bluetooth модуля.

Для собранного прототипа была разработаны и запрограммированы основные алгоритмы управления и игр, включая меню выбора игр, игру «Понг» и «Скроллер». Также был разработан способ сбора статистики без подключения дополнительных модулей.

Была собрана статистика нескольких игровых сессий и проведен анализ полученных данных. По ним было выяснено, что для человека без проблем с мелкой моторикой не составляет труда быстро привыкнуть к управлению и правилам игры и последовательно одерживать победу в игре «Понг» и стабильно набирать очки в игре «Скроллер». Также было выяснено, что по прошествию примерно 7 игровых сессий можно заметить усреднение значений счета и времени. По этим значениям можно судить о навыках использующего устройства, и они могут быть использованы для анализа прогресса пациента при последовательном использовании консоли.

4. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ДИПЛОМНОГО ПРОЕКТА.

Дипломный проект посвящен разработке программной и аппаратной части игровой консоли, направленной на развитие навыков мелкой моторики. Для управления аппаратной частью используется плата Arduino Nano, устройством ввода являются потенциометр и кнопка, вывод осуществляется с помощью OLED экрана. Также в устройстве реализован сбор статистики, осуществляющийся путем загрузки данных на подключенный к устройству компьютер. Программная часть разработана в программе Arduino IDE, для оформления документации использовались программы EasyIDE и Tinkercad, а также circuito.io. Поскольку программная часть реализована с использованием исключительно бесплатных программ, цена их использования указана не будет, поскольку она не повлияет на конечную цену изделия, так как является нулевой.

4.1. Расчет расходов на оплату труда согласно детализированному план-графику работ

По профессии младший программист-инженер по данным сайта spb.hh.ru [25] на май 2024 года заработная плата вакансий находится в диапазоне от 45000 до 78000 рублей. Учитывая, что проект выполняет начинающий специалист без прежнего опыта работы, а также для удобства расчетов примем ставку равной 45000 рублей.

Ставка доцента кафедры согласно сайтам etu.ru [26] и spb.hh.ru [27] находится в диапазоне от 36200 до 60000 рублей по связанной специальности. Тогда среднее между крайними границами диапазона – 48100 рублей.

Размер заработной платы в день – отношение оклада к количеству рабочих дней в месяце (21 день).

Для расчета расходов сведены в таблицу 4.1 фактические временные затраты на разработку системы и рассчитанная ставка.

Таблица 4.1 – Временные затраты на разработку и ставка исполнителя

№	Наименование работ	Исполнитель	Трудоемкость, чел/день	Ставка, руб/день
1	Аппаратное исполнение системы	Инженер-программист	2,3	2142,86
2	Разработка алгоритма управления и пользовательского интерфейса	Инженер-программист	35	2142,86
		Научный руководитель	3	2290,48
3	Проверка работы системы и выявленные закономерности	Инженер-программист	2	2142,86
		Научный руководитель	2	2290,48
4	Технико-экономическое обоснование дипломного проекта	Инженер-программист	5	2142,86
5	Оформление пояснительной записки	Инженер-программист	5	2142,86
6	Оформление иллюстративного материала	Инженер-программист	3	2142,86

Расходы на основную заработную плату представлены в таблице 4.2 и были определены по формулам:

$$Z_{\text{осн.з./пл}} = \sum_{i=1}^k T_i \cdot C_i, \quad (1)$$

где $Z_{\text{осн.з./пл}}$ - расходы на основную заработную плату исполнителей (руб.); k – количество исполнителей; T_i - время, затраченное i -м исполнителем на проведение исследования (дни); C_i - ставка i -го исполнителя (руб./день).

Таблица 4.3 – Расчет основной заработной платы

Исполнитель	Ставка (руб./день)	Общее время (день)	Основная заработная плата (руб.)
Инженер-программист	2142,86	52,3	112071,59
Научный руководитель	2290,48	5	11452,40
Итого			123523,99

Расходы на дополнительную заработную плату были определены по формуле:

$$З_{\text{доп.з/пл}} = З_{\text{осн.з/пл}} \cdot \frac{Н_{\text{доп}}}{100},$$

(2)

где $Н_{\text{доп}}$ – норматив дополнительной заработной платы (принят 8,3%).

Полученные данные сведены в таблицу 4.3.

Таблица 4.3 – Расчет заработной платы

Исполнитель	Основная заработная плата (руб.)	Дополнительная заработная плата (руб.)	Совокупная заработная плата (руб.)
Инженер-программист	112071,59	9301,94	121373,53
Научный руководитель	11452,40	950,55	12402,95
Итого	123523,99	10252,49	133776,48

Отчисления во внебюджетные фонды были определены по формуле:

$$З_{\text{соц}} = (З_{\text{осн.з/пл}} + З_{\text{доп.з/пл}}) \cdot \frac{Н_{\text{соц}}}{100},$$

(3)

где $H_{\text{соц}}$ – отчисления на страховые взносы (30,2%).

Таким образом, $Z_{\text{соц}} = (123523,99 + 10252,49) \cdot \frac{30,2}{100} = 40\,400,50$ рублей.

4.2. Расчет материальных затрат

Общие затраты на материалы были рассчитаны по формуле:

$$Z_M = \sum_{i=1}^L G_i \cdot C_i \left(1 + \frac{H_{\text{т.з.}}}{100}\right), \quad (4)$$

где G – норма расхода на единицу продукции, C – цена приобретения единицы материала, $H_{\text{т.з.}}$ – норма транспортно-заготовительных расходов (составляет 10%), L – индекс вида сырья.

Затраты на сырье и материалы сведены в таблицу 4.4. По данным сайта ozon.ru, стоимость офисной бумаги формата А4 составляет 391 рублей [28], а картриджа для принтера – 2375 рублей [29].

Затраты на комплектующие изделия сведены в таблицу 4.5.

Таблица 4.4 – Затраты на сырье и материалы

Изделие	Материал	Тип	Норма расхода на изделие, ед.	Цена за единицу, руб	Сумма на изделие, руб
Бумага офисная	бумага	А4, 500 листов	0,12	391	46,92
Картридж для принтера	чернила	черный	1	2375	2375,00
Итого					2421,92
Транспортно-заготовительные расходы					0
Итого с учетом транспортно-заготовительных расходов					2421,92

Таблица 4.5 – Затраты на покупные комплектующие

Наименование	Норма расхода на единицу продукции, шт.	Цена, руб./шт.	Сумма на единицу продукции, руб.
Микроконтроллер Arduino NANO V3.0 TYPE-C	1	613 [30]	613
Набор соединительных проводов папа-папа 65 штук	~15	280 [31]	65
Кабель usb type c	1	256 [32]	256
Потенциометр	1	165 [33]	165
Тактовая кнопка (10шт)	1	204 [34]	20
OLED дисплей 0.96" 128x64, I2C	1	333 [35]	333
Итого			1452

Затраты на покупные комплектующие составили 1452 рублей, на сырье и материалы – 2421,92 рублей.

4.3. Расчет амортизационных издержек

При разработке программной части игровой консоли был использован компьютер. Стоимость ноутбука по данным сайте ozon.ru [36] – 15703 рубля.

Годовая норма амортизации определена линейным методом как отношение единицы к сроку полезного использования изделия. Согласно [37] персональный компьютер относится ко второй группе со сроком полезного использования от 2 до 3 лет, примем срок три года, считая, что оборудование надежное и используется бережно.

Амортизационные отчисления за год использования были определены как:

$$A_i = C_{п.н.i} \cdot \frac{H_{ai}}{100}, \quad (5)$$

где $C_{п.н.i}$ – первоначальная стоимость средства; H_{ai} – годовая норма амортизации.

Тогда величина амортизационных отчислений за период использования оборудования была получена как:

$$A_{iВКР} = A_i \cdot \frac{T_{iВКР}}{12}, \quad (6)$$

где $T_{iВКР}$ – время использования оборудования при работе над ВКР.

Персональный компьютер использовался все время работы над проектом – 53,3 дня.

Полученные данные сведены в таблицу 4.6.

Таблица 4.6 – Амортизационные отчисления

Оборудование	Амортизационные отчисления за год (руб.)	Время использования в проекте (мес.)	Амортизационные отчисления (руб.)
Ноутбук ASUS S15	5 234,33	1,7767	774,99
Итого			774,99

Амортизационные отчисления составили 774,99 рублей

4.4. Расчет прочих прямых расходов

В работе над проектом была использована передача данных с микросхемы на персональный компьютер посредством Wi-Fi сети. Время использования составило 2 месяца, абонентская плата за тарифный план «Тарифище» оператора

МТС по данным сайта mts.ru [38] составила 550 рублей. Данные сведены в таблицу 4.7

Таблица 4.7 – Прочие прямые расходы

Наименование	Время использования, округление до большего целого (мес.)	Стоимость, руб.	Общая стоимость, руб.
Тариф МТС Проще	2	550	1100
Итого			1100

Wi-Fi соединение было использовано при подготовке дипломного проекта, а также задействовано непосредственно во время работы системы при передаче данных с микроконтроллера к персональному компьютеру.

4.5. Расчет накладных расходов

Накладные расходы представляют собой дополнительные затраты, не включающиеся в прямые расходы. Наибольшей статьей расходов в проекте являются расходы на заработную плату.

Согласно формуле:

$$H = \frac{\sum_{i=1}^2 \left(z_{\text{доп.з.пл}} + z_{\text{осн.з.пл}} \right) \cdot N_{\text{нр}}}{100}, \quad (7)$$

где $N_{\text{нр}}$ – норматив накладных расходов, составляющий 20%; i – номер специалиста.

Тогда:

$$H = \frac{(9301,94 + 112071,59) \cdot 20 + (950,55 + 11452,40) \cdot 20}{100}$$

Тогда величина накладных расходов составляет: 26755,29 рублей.

4.6. Расчет полной себестоимости

Для удобства определения себестоимости проекта все полученные ранее данные сведены в таблицу 4.8.

Таблица 4.8 – Смета затрат на ВКР

№	Наименование статьи	Сумма, (руб.)	Структура себестоимости, (%)
1	Материальные затраты	3873,92	1,87
2	Заработная плата	133776,48	64,73
3	Отчисления во внебюджетные фонды	40400,50	19,55
4	Издержки на амортизацию	774,99	0,37
5	Накладные расходы	26755,29	12,95
6	Прочие прямые затраты	1100,00	0,53
Итого		206681,18	100,00

Себестоимость проекта составляет 206681,18 рублей.

4.7. Расчет себестоимости одного изделия

При расчете стоимости единицы изделия материальные затраты не изменяются. При условии, что код программы уже разработан, а все комплектующие имеются в наличии, на сборку системы работнику потребуется 1 день. Тогда затраты заработная плата специалиста составляет 2142,86 рублей. Согласно формуле (2), расходы на дополнительную заработную плату:

$$З_{\text{доп.з/пл}} = 2142,86 \cdot \frac{8,3}{100} = 177,86 \text{ руб.}$$

Согласно формуле (3), отчисления во внебюджетные фонды:

$$З_{\text{соц}} = (2142,86 + 177,85) \cdot \frac{30,2}{100} = 700,86 \text{ руб.}$$

Тогда общие расходы на заработную плату составляют:

$$З_{\text{общ.}} = З_{\text{осн.з/пл}} + З_{\text{доп.з/пл}},$$

(8)

$$З_{\text{общ.}} = 2142,86 + 177,85 = 2320,72 \text{ руб.}$$

Накладные расходы рассчитаны согласно формуле (7):

$$Н = \frac{(177,85 + 2142,86) \cdot 20}{100}$$

И составляют 464,14 рублей.

Согласно таблице 4.6, амортизационные отчисления составляют 774,99 руб. Калькуляция себестоимости прибора приведена в таблице 4.9. [39]

Таблица 4.9 – Калькуляция себестоимости прибора

№	Наименование статьи	Сумма, (руб.)	Структура себестоимости, (%)
1	Материальные затраты	3873,92	47,62
2	Заработная плата	2320,72	28,53
3	Отчисления во внебюджетные фонды	700,86	8,62
4	Издержки на амортизацию	774,99	9,53
5	Накладные расходы	464,14	5,70
Итого		8134,63	100,00

Стоимость изготовления единицы изделия составляет 8134,63 рублей.

4.8. Анализ конкурентов

Орторент МОТОРИКА – это тренажер для реабилитации верхних конечностей, восстановления мелкой моторики рук при помощи интерактивных программ. Цена — от 1150000 рублей. [2]

Преимущества Орторент МОТОРИКА:

- Реабилитация происходит в формате компьютерной игры с симуляцией реальных жизненных ситуаций. В игре доступен широкий выбор эффективных и увлекательных упражнений с различным уровнем сложности.
- Аппарат Орторент МОТОРИКА предлагает 12 мотивационных программ в базовом комплекте с возможностью дальнейшего пополнения.
- Прибор имитирует движения рук и плеч, которые встречаются в повседневной жизни: от самых крупных и простых до мелкой моторики. Всего таких движений 6
- Возможность ведения статистики занятий для каждого пациента, что позволяет автоматически отслеживать функциональные улучшения и вносить коррективы в процесс реабилитации;
- Возможность осуществлять удаленный контроль за пациентом при наличии выхода в Интернет;
- Plug n Play за 5 минут – для установки Орторент МОТОРИКА не требуется специальной технической подготовки;
- Диагностика функции руки. Программное обеспечение точно регистрирует движение верхних конечностей, позволяя врачу определить координацию пациента и прогресс восстановления.

Минусы:

- Высокая цена
- Отсутствие возможности портативного использования

- Необходимо постоянное физическое место для установки и использования

Реабилитационная перчатка АНИКА — это изделие для восстановления мелкой моторики и координации с оценкой функциональных возможностей при помощи биологической обратной связи. (1 комплект = перчатка + датчик+ПО).
Цена — от 130000 рублей. [3]

Плюсы:

- Реабилитационная перчатка позволяет проводить реабилитацию пациента в игровой форме, получая объективные данные о производимых пациентом действиях в реальном времени, и адаптировать программу реабилитации в соответствии с реальными возможностями конкретного пациента, подобрав комплекс интерактивных упражнений для восстановления мелкой моторики и координации движений.
- Датчики положения регистрируют движение кисти в 3-х измерениях
- Возможность отработки захвата и вращения в локте
- Расширенный функционал: регистрация движения локтя и кисти в пространстве
- Пациент может выполнять упражнения из любого удобного положения — лежа, сидя, стоя.
- В конце занятия проводится оценка диапазона движений и сравнение с результатами предыдущих занятий.

Минусы:

- Высокая цена
- Отсутствие возможности портативного использования
- Необходимо постоянное физическое место для установки и использования

После анализа конкурентов можно сделать вывод, что существующие устройства направлены в основном на использование в лечебных учреждениях, поскольку обладают довольно высокой ценой для индивидуальной покупки, в то время как разрабатываемое устройство направлено на домашнее использование и имеет цену, доступную для покупки человеком со средним заработком. Также приборы конкурентов требуют использования в определенном месте, тогда как разрабатываемый продукт имеет портативный потенциал. У существующих конкурентов имеется сильное преимущество — они развивают моторику всей руки, включая движения от локтя, тогда как устройство, созданное в результате данной ВКР, фокусируется только на движениях кисти и пальцев. Также программный аппарат конкурентов предлагает большой выбор игр-программ для реабилитации.

4.9. Выводы по главе

При анализе экономической составляющей дипломного проекта была получена себестоимость готового продукта, равная 206681,18 при этом расходы на заработную плату составляют 64,73% стоимости проекта, отчисления во внебюджетные фонды – 19,55%, оставшиеся расходы равны 28 630,28 рублей. Стоимость изготовления единицы изделия при массовом производстве и условии готовности всех программ – 8134,63 рублей.

Представленные на рынке системы управления микроклиматом со схожими возможностями стоят не менее 100000 рублей в версии и хотя имеют более расширенный функционал, ограничены использованием в определенном месте установки.

Для личного использования пациентами вне лечебных учреждений, во время домашнего использования или поездок, выполнение такого проекта является экономически выгодным.

ЗАКЛЮЧЕНИЕ

В рамках ВКР было спроектирована и разработана игровая консоль на базе платы Arduino.

Было выбрано целевая аудитория: дети с проблемами мелкой моторики, изучена проблематика выбранной предметной области, что конкретно представляют собой проблемы мелкой моторики, каковы их причины, и каковы способы лечения данных проблем.

Были изучены платы с микроконтроллером и одноплатные компьютеры, проведено их сравнение, в результате которого самым выгодным оказалось создание устройства на базе Arduino. Были изучены особенности программирования для Arduino и проектирования устройств на базе Arduino, в результате чего был создан прототип игровой консоли.

Была выбраны две игры для реализации, созданы их алгоритмы и реализованы основные методы на языке программирования Arduino C. Было достигнут режим работы устройства без проблем и неожиданных прерываний, заключающийся в воспроизведении выбранной в меню игры и возвращение в меню после завершения игрового сеанса. Во время тестирования было отредактировано управление и решены проблемы с оптимизацией.

После создания для разработанного устройства была настроена технология обратной связи, включающая в себя сбор статистической информации о прошедших игровых сессиях. Было выяснено, что после нескольких игровых сессий в каждой игре достигается некоторое среднее значение времени и счета, по которому можно судить о навыках использующего устройства, и которые можно использовать для мониторинга прогресса при постоянном использовании устройства.

У созданного устройства имеется потенциал дальнейшего развития, как в аппаратном, так и в программном плане. Данное устройство также является выгодной альтернативой конкурентам с экономической точки зрения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Нарушение мелкой моторики у детей. [Электронный ресурс] URL: <https://volgograd.medsir.ru/spravochnik-zabolevaniy/narushenie-melkoy-motoriki-u-detey/> (дата обращения: 03.04.2024)
2. Реабилитационная перчатка АНИКА. [Электронный ресурс] URL: <https://moskva.fis.ru/product/23639610-reabilitacionnaya-perchatka-anika-art-mad23338> (дата обращения: 03.04.2024)
3. Система активной реабилитации верх. конечностей и развития мелкой моторики. [Электронный ресурс] URL: <https://www.nt-med.ru/catalog/trenazhery-dlja-reabilitacii/trenazhery-dlya-razvitiya-melkoj-motoriki/sistema-dlya-aktivnoj-reabilitatsii-verhnih-konechnostej-i-razvitiya-melkoj-motoriki-ortorent-motorika> (дата обращения: 03.04.2024)
4. Вводный урок об Arduino. [Электронный ресурс] URL: <https://alexgyver.ru/lessons/about-arduino/> (дата обращения: 28.02.2024)
5. Обзор Arduino. [Электронный ресурс] URL: <https://amperka.ru/page/what-is-arduino> (дата обращения: 28.02.2024)
6. Обзор продукта arduino-uno. [Электронный ресурс] URL: <https://amperka.ru/product/arduino-uno> (дата обращения: 28.02.2024)
7. Обзор продукта arduino-leonardo. [Электронный ресурс] URL: <https://amperka.ru/product/arduino-leonardo> (дата обращения: 28.02.2024)
8. Обзор продукта iskra-neo. [Электронный ресурс] URL: <https://amperka.ru/product/iskra-neo> (дата обращения: 28.02.2024)
9. Обзор продукта PRO Mini 5V. [Электронный ресурс] URL: https://amperkot.ru/spb/catalog/plata_pro_mini_5v_16mhz_arduinovmestimaya-23865992.html (дата обращения: 28.02.2024)
10. Обзор продукта iskra-mini. [Электронный ресурс] URL: <https://amperka.ru/product/iskra-mini> (дата обращения: 28.02.2024)

11. Обзор продукта arduino-micro. [Электронный ресурс] URL: <https://amperka.ru/product/arduino-micro> (дата обращения: 28.02.2024)
12. Обзор продукта arduino-mega-2560. [Электронный ресурс] URL: <https://amperka.ru/product/arduino-mega-2560> (дата обращения: 28.02.2024)
13. Обзор продукта arduino-due. [Электронный ресурс] URL: <https://amperka.ru/product/arduino-due> (дата обращения: 28.02.2024)
14. Обзор продукта iskra-js. [Электронный ресурс] URL: <https://amperka.ru/product/iskra-js> (дата обращения: 28.02.2024)
15. Обзор продукта strela. [Электронный ресурс] URL: <https://amperka.ru/product/strela> (дата обращения: 28.02.2024)
16. Обзор продукта stm32-nucleo-f401re. [Электронный ресурс] URL: <https://amperka.ru/product/stm32-nucleo-f401re> (дата обращения: 28.02.2024)
17. Обзор продукта teensy-32. [Электронный ресурс] URL: <https://amperka.ru/product/teensy-32> (дата обращения: 28.02.2024)
18. Arduino или Raspberry Pi: как выбрать контроллер для проекта. [Электронный ресурс] URL: <https://amperka.ru/page/development-board-guide> (дата обращения: 5.03.2024)
19. Pong - первая коммерчески успешная игра. [Электронный ресурс] URL: https://dzen.ru/a/ZA3bLMu0HWlibP_1 (дата обращения: 5.03.2024)
20. Pong (игра). [Электронный ресурс] URL: [https://ru.wikipedia.org/wiki/Pong_\(игра\)](https://ru.wikipedia.org/wiki/Pong_(игра)) (дата обращения: 5.03.2024)
21. Сайд-скроллер. [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/Сайд-скроллер> (дата обращения: 5.03.2024)
22. Термин Scroller. [Электронный ресурс] URL: <https://zoom.cnews.ru/games/likbez/item/scroller> (дата обращения: 5.03.2024)
23. Arduino retro gaming с OLED-дисплеем [Электронный ресурс] URL: <https://gadgetshelp.com/diy/arduino-retro-gaming-s-oled-displeem> (дата обращения: 7.02.2024)

24. The Rocket game on Arduino with OLED display [Электронный ресурс] URL: <https://alexnoyanov.wixsite.com/techblog/post/the-rocket-game-on-arduino-with-oled-display> (дата обращения: 24.04.2024)
25. Работа инженером-программистом в Санкт-Петербурге [Электронный ресурс] – URL: https://spb.hh.ru/search/vacancy?area=2&search_field=name&search_field=company_name&search_field=description&enable_snippets=false&experience=noExperience&text=%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82&only_with_salary=true (дата обращения: 12.05.2024)
26. Приказ № ОД/0539 от 27.09.2019 «...об изменении размеров минимальных должностных окладов...» [Электронный ресурс] – URL: https://etu.ru/sveden/files/Prikaz_OD-0539_27.09.2019.pdf (дата обращения: 12.05.2024)
27. Работа доцентом в Санкт-Петербурге [Электронный ресурс] – URL: <https://spb.hh.ru/vacancies/dotsent> (дата обращения: 12.05.2024)
28. Бумага офисная "Svetocopy" [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/7969279/> (дата обращения: 12.05.2024)
29. Картридж для струйного принтера HP 122, черный [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/257110801> (дата обращения: 12.05.2024)
30. Микроконтроллер Arduino NANO V3.0 TYPE-C черный [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/1519887659> (дата обращения: 12.05.2024)
31. Набор соединительных проводов папа-папа 65 штук [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/1298538134> (дата обращения: 12.05.2024)
32. Кабель usb type c [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/201361145> (дата обращения: 12.05.2024)

33. Потенциометр GSMIN WH148 B1K (1 кОм) переменный резистор 15мм 3-pin [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/1341065825> (дата обращения: 12.05.2024)
34. Тактовая кнопка 6х6х5мм (10шт) [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/913790904> (дата обращения: 12.05.2024)
35. OLED дисплей 0.96" 128х64, I2C, Белый [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/832387328> (дата обращения: 12.05.2024)
36. Digma C4403 Ноутбук [Электронный ресурс] – URL: <https://www.ozon.ru/context/detail/id/1456814576> (дата обращения: 12.05.2024)
37. Постановление Правительства РФ от 01.01.2002 N 1 (ред. от 18.11.2022) "О Классификации основных средств, включаемых в амортизационные группы" [Электронный ресурс] – URL: https://www.consultant.ru/document/cons_doc_LAW_34710/9db3ac4dcd3b010eb67c60af3d823fe6ba73d749/ (дата обращения: 12.05.2024)
38. МТС Проще [Электронный ресурс] – URL: https://spb.mts.ru/personal/mobilnaya-svyaz/tarifi/vse-tarifi/mts_proshche (дата обращения: 12.05.2024)
39. Алексеева О. Г. Экономическое обоснование ВКР. СПб.: Издательство СПбГЭТУ «ЛЭТИ», 2023

ПРИЛОЖЕНИЕ А

Код программы

Файл main_program

```
#include <OLED_I2C.h>

unsigned long time1=0, time2=0;
int position=0;
byte lastButtonState = LOW;
byte buttonState = LOW;
int buttonPin = 12;
int ledPin = 3;
bool gamenotchosen = true;
OLED myOLED(SDA, SCL, 8);

int resolution[2] = { 128, 64 }, ball[2] = { 20, (resolution[1] / 2) };
const int PIXEL_SIZE = 10, WALL_WIDTH = 2, PADDLE_WIDTH = 4, BALL_SIZE = 5,
SPEED = 2;
int playerScore = 0, aiScore = 0, playerPos = 0, aiPos = 0;
char ballDirectionHori = 'R', ballDirectionVerti = 'S';
boolean inProgress = true;

const int rocketLenght = 10; // Rocket body Length
const int rocketHight = 5; // Rocket body Hight
const int maxX = 32; // Maximum X-value
const int maxY = 128; // Maximum Y-value
int y; // Rocket Y-coordinate
int x = 8; // Rocket X-coordinate
int pixX; // Pixel coordinate X
int pixY; // Pixel coordinate Y
int flameR = 8; // Flame radius
int potValue;
int intensFire = 10; // Fire pixel concentration
int astR = 5; // Asteroid radius
int minastR = 2;
int maxastR = 10;
const int astPixConcent = 20; // Pixels concentration inside the asteroid
int rOut = 5;
int astX ;
int astY ;
int astSpd = 2; // Asteroid speed
int score = 0; // Game score
int record = 0; // Game record
int pixExpX; // Explosion coordinate X
int pixExpY; // Explsion coordinate Y
int explosionIntetsivity = 10;
int explosionR = 15; // Radius of rocket explosion
bool gameMode = false;
```



```

int potPin = A0;          // Potenciometer pin

extern uint8_t SmallFont[];
void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
    Serial.begin(9600);
    myOLED.begin();
    myOLED.setFont(SmallFont);
    analogWrite(A1,1023);
    analogWrite(A2,0);
    potValue = analogRead(potPin);
    byte lastButtonState = digitalRead(buttonPin);
    //drawMenu();// put your setup code here, to run once:

}

void TimePrint(unsigned long time)
{
    time=time/250;
    if (time/60%60<10) { Serial.print ("0"); }
    Serial.print ((time/60)%60);
    Serial.print (":");
    if (time%60<10) { Serial.print ("0"); }
    Serial.println (time%60);
}

void pong(){
    myOLED.clrScr();
    playerScore = 0; aiScore = 0; playerPos = 0; aiPos = 0;
    inProgress = true;
    time1=millis();
    pong_game();
    TimePrint(millis()-time1);
    playerScore = 0; aiScore = 0; playerPos = 0; aiPos = 0;
    inProgress = true;
    gamenotchosen=true;

}

void scroller(){
    myOLED.clrScr();
    gameMode=true;
    time1=millis();
    scroller_game();
    time2=millis();
    TimePrint(time2-time1);
    gamenotchosen=true;
}

void drawMenu(){

```

```

myOLED.clrScr();
myOLED.print("<--CHOOSE A GAME-->", 5,0);
myOLED.print("Pong",10,10);
myOLED.print("Scroller",10,20);
}
void loop() {
  while (gamenotchosen)
  {
    drawMenu();
    position = analogRead(A0);
    if (position<=512) {
      myOLED.drawCircle(5, 15, 2);
    } else {
      myOLED.drawCircle(5, 25, 2);
    }
    buttonState=digitalRead(buttonPin);
    if (buttonState==lastButtonState) {
      lastButtonState=buttonState;
      if (position<=512) {

        gamenotchosen=false;
        pong();
      } else {

        gamenotchosen=false;
        scroller();
      }
    }
  }
  // put your main code here, to run repeatedly:
  myOLED.update();
}
delay(1000);
}

```

Файл pong_game

```

void pong_game() {
  while (inProgress) {
    if (aiScore > 3 || playerScore > 3) { // check game state
      inProgress = false;
    }
    if (inProgress) {
      eraseScore();
      eraseBall(ball[0], ball[1]);
      if (ballDirectionVerti == 'U') { // move ball up diagonally
        ball[1] = ball[1] - SPEED;

```

```

    }
    if (ballDirectionVerti == 'D') { // move ball down diagonally
        ball[1] = ball[1] + SPEED;
    }
    if (ball[1] <= 0) { // bounce the ball off the top
        ballDirectionVerti = 'D';
    }
    if (ball[1] >= resolution[1]) { // bounce the ball off the bottom
        ballDirectionVerti = 'U';
    }
    if (ballDirectionHori == 'R') {
        ball[0] = ball[0] + SPEED; // move
ball
        if (ball[0] >= (resolution[0] - 6)) { // ball is
at the AI edge of the screen
            if ((aiPos + 6) >= ball[1] && (aiPos - 6) <= ball[1]) { // ball
hits AI paddle IT WAS +-12
                if (ball[1] > (aiPos + 4)) { // deflect
ball down
                    ballDirectionVerti = 'D';
                } else if (ball[1] < (aiPos - 4)) { // deflect ball up
                    ballDirectionVerti = 'U';
                } else { // deflect ball straight
                    ballDirectionVerti = 'S';
                } // change ball direction
                ballDirectionHori = 'L';
            } else { // GOAL!
                ball[0] = 6; // move ball to other side of screen
                ballDirectionVerti = 'S';
                // reset ball to straight travel
                ball[1] = resolution[1] / 2; // move ball to middle of screen
                ++playerScore; // increase player score
            }
        }
    }
    if (ballDirectionHori == 'L') {
        ball[0] = ball[0] -
SPEED; // move ball
        if (ball[0] <= 6)
{ // ball is at the player
edge of the screen
            if ((playerPos + 15) >= ball[1] && (playerPos - 15) <= ball[1])
{ // ball hits player paddle
                if (ball[1] > (playerPos + 4))
{ // deflect ball down
                    ballDirectionVerti = 'D';
                } else if (ball[1] < (playerPos - 4)) { // deflect ball up
                    ballDirectionVerti = 'U';
                } else { // deflect ball straight

```

```

        ballDirectionVerti = 'S';
    } // change ball direction
    ballDirectionHori = 'R';
} else {
    ball[0] = resolution[0] - 6; // move ball to other side of screen
    ballDirectionVerti = 'S';    // reset ball to straight travel
    ball[1] = resolution[1] / 2; // move ball to middle of screen
    ++aiScore;                    // increase AI score
}
}
}
drawBall(ball[0], ball[1]);
erasePlayerPaddle(playerPos);
playerPos = analogRead(A0); // read player potentiometer
playerPos = map(playerPos, 0, 1023, 6, 54); // convert value from 0 -
1023 to 8 - 54
drawPlayerPaddle(playerPos);
moveAi();
drawNet();
drawScore();
} else { // somebody has won

    myOLED.clrScr();

    // figure out who
    if (aiScore > playerScore) {
        myOLED.print("GAME OVER!", 0, 0);
    } else if (playerScore > aiScore) {
        myOLED.print("YOU WIN!", 0, 0);
    }
    Serial.print ("Pong game. Score:");
    Serial.print(playerScore);
    Serial.print("/");
    Serial.println(aiScore);
    Serial.println("Time:");
}

myOLED.update();
}
delay(2000);}

void moveAi() {
    // move the AI paddle
    eraseAiPaddle(aiPos);
    if (ball[1] > aiPos) {
        ++aiPos;
    } else if (ball[1] < aiPos) {

```

```

        --aiPos;
    }
    drawAiPaddle(aiPos);
}

void drawScore() {
    // draw AI and player scores
    myOLED.printNumI(playerScore, 45, 0);
    myOLED.printNumI(aiScore, 75, 0);
}

void eraseScore() {
    // erase AI and player scores
    myOLED.invertText(true);
    myOLED.printNumI(playerScore, 45, 0);
    myOLED.printNumI(aiScore, 75, 0);
    myOLED.invertText(false);
}

void drawNet() {
    for (int i = 0; i < (resolution[1] / WALL_WIDTH); ++i) {
        drawPixel(((resolution[0] / 2) - 1), i * (WALL_WIDTH) + (WALL_WIDTH *
i), WALL_WIDTH);
    }
}

void drawPixel(int posX, int posY, int dimensions) {
    // draw group of pixels
    for (int x = 0; x < dimensions; ++x) {
        for (int y = 0; y < dimensions; ++y) {
            myOLED.setPixel((posX + x), (posY + y));
        }
    }
}

void erasePixel(int posX, int posY, int dimensions) {
    // erase group of pixels
    for (int x = 0; x < dimensions; ++x) {
        for (int y = 0; y < dimensions; ++y) {
            myOLED.clrPixel((posX + x), (posY + y));
        }
    }
}

void erasePlayerPaddle(int row) {
    erasePixel(0, row - (PADDLE_WIDTH * 2), PADDLE_WIDTH);
    erasePixel(0, row - PADDLE_WIDTH, PADDLE_WIDTH);
    erasePixel(0, row, PADDLE_WIDTH);
    erasePixel(0, row + PADDLE_WIDTH, PADDLE_WIDTH);
}

```

```

    erasePixel(0, row + (PADDLE_WIDTH + 2), PADDLE_WIDTH);
}

void drawPlayerPaddle(int row) {
    drawPixel(0, row - (PADDLE_WIDTH * 2), PADDLE_WIDTH);
    drawPixel(0, row - PADDLE_WIDTH, PADDLE_WIDTH);
    drawPixel(0, row, PADDLE_WIDTH);
    drawPixel(0, row + PADDLE_WIDTH, PADDLE_WIDTH);
    drawPixel(0, row + (PADDLE_WIDTH + 2), PADDLE_WIDTH);
}

void drawAiPaddle(int row) {
    int column = resolution[0] - PADDLE_WIDTH;
    drawPixel(column, row - (PADDLE_WIDTH * 2), PADDLE_WIDTH);
    drawPixel(column, row - PADDLE_WIDTH, PADDLE_WIDTH);
    drawPixel(column, row, PADDLE_WIDTH);
    drawPixel(column, row + PADDLE_WIDTH, PADDLE_WIDTH);
    drawPixel(column, row + (PADDLE_WIDTH * 2), PADDLE_WIDTH);
}

void eraseAiPaddle(int row) {
    int column = resolution[0] - PADDLE_WIDTH;
    erasePixel(column, row - (PADDLE_WIDTH * 2), PADDLE_WIDTH);
    erasePixel(column, row - PADDLE_WIDTH, PADDLE_WIDTH);
    erasePixel(column, row, PADDLE_WIDTH);
    erasePixel(column, row + PADDLE_WIDTH, PADDLE_WIDTH);
    erasePixel(column, row + (PADDLE_WIDTH * 2), PADDLE_WIDTH);
}

void drawBall(int x, int y) {
    myOLED.drawCircle(x, y, BALL_SIZE);
}

void eraseBall(int x, int y) {
    myOLED.clrCircle(x, y, BALL_SIZE);
}

```

Файл scroller_game

```

// Function for drawing rocket:
void drawRocket(int x,int y){
    myOLED.drawRect(x,y,x+rocketLenght,y+rocketHeight);
    myOLED.drawLine(x+rocketLenght,y+rocketHeight,x+rocketLenght+5,y+rocketHeight/2);
    myOLED.drawLine(x+rocketLenght,y,x+rocketLenght+5,y+rocketHeight/2);
    myOLED.drawLine(x+5,y+rocketHeight,x-2,y+8); // Down
    myOLED.drawLine(x,y+rocketHeight,x-2,y+8);
}

```

```

    myOLED.drawLine(x+5,y,x-2,y-4); // Up
    myOLED.drawLine(x,y,x-2,y-4);
}
// Drawing rocket flame:
void drawFlame(int x,int y){
    for(int i = 0;i<= intensFire; i++){
        pixX = random(x-flameR,x);
        pixY = random(y-2,y+flameR);
        myOLED.setPixel(pixX,pixY);
    }
}

void Asteroid(int x,int y){
    myOLED.drawCircle(x,y,astR);
}

void Explosion(int x, int y){
    for(int i = 0; i < explosionIntetsivity; i++){
        //myOLED.drawCircle(x,y,i);
        pixExpX = random(x,x+explosionR);
        pixExpY = random(y-explosionR,y+explosionR);
        myOLED.setPixel(pixExpX,pixExpY);
        myOLED.setPixel(pixExpX+1,pixExpY);
        myOLED.setPixel(pixExpX,pixExpY+1);
        myOLED.update();
    }
}

void scroller_game() {
    // put your main code here, to run repeatedly:

    while(gameMode){ // Playing game or printing menu:
        myOLED.clrScr();
        potValue = analogRead(potPin);
        y = map(potValue,0,1023,4,64-rocketHight-4);

        drawRocket(x,y); // Drawing rocket
        drawFlame(x,y);

        astSpd = map(score,0,100,2,10 ); // Asteroids speed increasing

        // Moving the Asteroid:
        if(astX > 0-astR){
            Asteroid(astX,astY);
            astX = astX-astSpd;
        }else{

```

```

        astY = random(astR, 64-astR);
        astX = 128+astR;
        score++;
    }
    // Asteroid with rocket:
    if(astX-astR < x+rocketLenght+5 && astY < y+rocketHight+5 && astY >
y-rocketHight-5){
        myOLED.clrScr();
        drawRocket(x,y);
        for(int i = astX; i > 0; i--){
            Asteroid(i,astY);
            myOLED.update();
            myOLED.clrCircle(i, astY, astR);
        }
        Explosion(x,y);
        astY = random(astR, 64-astR);
        astX = 128+astR;
        delay(500);
        myOLED.print("GAME OVER ", 30, 20);
        myOLED.print("YOUR SCORE: ", 30, 30);
        myOLED.printNumI(score, 100, 30);
        Serial.print ("Scroller game. Score:");
        Serial.println(score);
        Serial.println("Time:");
        score = 0;
        myOLED.update();
        delay(2000);
        gameMode = false;           // Exiting the game to the menu
    }

    myOLED.printNumI(score, 30, 2); // Drawing score
    myOLED.printNumI(astSpd, 60,2); // Drawing asteroids speed
    myOLED.update();
}
}

```