Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Архитектура вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**РАСПОЗНАВАНИЕ СТРОЧНЫХ И ПРОПИСНЫХ БУКВ КИРИЛЛИЦЫ И ЦИФР НА МИКРОКОНТРОЛЛЕРЕ ARM**

Выполнил: студент гр.253504

Фроленко К.Ю.

Проверил: ассистент кафедры информатики Калиновская А.А.

Минск 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc178244987)

[1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ 5](#_Toc178244988)

[1.1 Структура и архитектура вычислительной системы 5](#_Toc178244989)

[1.2 История, версии и достоинства 6](#_Toc178244990)

[1.3 Обоснование выбора вычислительной системы 7](#_Toc178244991)

[1.4 Анализ выбранной вычислительной системы для написания программы 8](#_Toc178244992)

[2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 10](#_Toc178244993)

[2.1 Структура и архитектура платформы 10](#_Toc178244994)

[2.2 История, версии и достоинства 11](#_Toc178244995)

[2.3 Обоснование выбора платформы 12](#_Toc178244996)

[2.4 Анализ программного обеспечения для написания программы 13](#_Toc178244997)

[3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА 15](#_Toc178244998)

[3.1 Обоснование необходимости разработки 15](#_Toc178244999)

[3.2 Технологии программирования, используемые для решения поставленных задач 16](#_Toc178245000)

[3.3 Связь архитектуры вычислительной системы с разрабатываемым программным продуктом 17](#_Toc178245001)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc178245002)

ВВЕДЕНИЕ

В современном мире искусственный интеллект (ИИ) становится неотъемлемой частью различных сфер человеческой деятельности, начиная от автоматизации производственных процессов и заканчивая разработкой интеллектуальных бытовых устройств и систем развлечений. Одной из ключевых технологий ИИ является машинное зрение, и в частности распознавание рукописного текста. Эта область исследований играет важную роль в таких приложениях, как автоматизация ввода данных, создание интерактивных обучающих систем и совершенствование средств коммуникации, что делает ее крайне актуальной в условиях быстро развивающегося цифрового мира.

Распознавание рукописного текста представляет собой сложную задачу, требующую применения передовых методов машинного обучения и глубоких оптимизаций. Разнообразие почерков, стили написания символов и необходимость достижения высокой точности при ограниченных вычислительных ресурсах усиливают сложность задачи. Более того, скорость распознавания и эффективность системы играют ключевую роль в её успешном внедрении, особенно в контексте мобильных и встроенных устройств, где каждое вычисление влияет на производительность системы в целом.

Актуальность темы курсового проекта определяется растущей потребностью в автономных и энергоэффективных системах распознавания, которые могут работать на устройствах с ограниченными ресурсами. В условиях, когда доступ к облачным сервисам и мощным вычислительным системам ограничен или отсутствует, особенно в удаленных и малообеспеченных регионах, создание таких систем становится важной задачей. Внедрение локальных решений на базе микроконтроллеров позволяет значительно снизить задержки при обработке данных и повысить конфиденциальность, поскольку информация остается на устройстве, не покидая его пределов. Это особенно важно в контексте защиты данных и их безопасности.

Одной из основных сложностей при разработке таких систем является балансировка между сложностью модели и вычислительными возможностями микроконтроллера. Процессоры с архитектурой ARM, такие как Cortex-M7, хотя и обладают высокой производительностью для своего класса, существенно уступают по вычислительной мощности настольным компьютерам и облачным серверам. Это накладывает строгие ограничения на размеры нейронных сетей и объем доступной памяти, что делает задачу разработки системы еще более интересной и сложной.

Целью данной работы является разработка системы распознавания строчных и прописных букв кириллицы и цифр на микроконтроллере ARM, используя плату STM32F746G-DISCO.

Для достижения поставленной цели необходимо решить следующие задачи:

1 Изучить архитектуру вычислительной системы на основе микроконтроллера ARM Cortex-M7 и обосновать выбор платы STM32F746G-DISCO.

2 Проанализировать программную платформу TensorFlow Lite для микроконтроллеров, её структуру, историю развития и ключевые преимущества.

3 Обосновать необходимость разработки программного продукта и определить технологии программирования, необходимые для решения поставленных задач.

4 Разработать и оптимизировать модель нейронной сети для распознавания рукописных символов кириллицы и цифр.

5 Реализовать программное обеспечение на микроконтроллере STM32F746G-DISCO и провести тестирование системы распознавания.

Достижение поставленной цели позволит продемонстрировать возможности выполнения сложных задач машинного обучения на устройствах с ограниченными ресурсами и оценить эффективность использования современных технологий ИИ во встраиваемых системах. Более того, успешная реализация системы подчеркнет потенциал использования нейронных сетей в микроконтроллерах для решения практических задач.

Детальная постановка задачи заключается в создании эффективной и экономичной системы распознавания рукописного текста, способной работать в режиме реального времени на микроконтроллере с ограниченными вычислительными ресурсами. Это предполагает обучение модели на базе данных рукописных символов, оптимизацию модели для выполнения на ARM Cortex-M7 и разработку встроенного приложения для демонстрации возможностей системы.

1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

1.1 Структура и архитектура вычислительной системы

Микроконтроллеры семейства ARM Cortex-M, в частности модель Cortex-M7, являются высокопроизводительными 32-битными процессорами, специально разработанными для встраиваемых систем с ограниченными вычислительными ресурсами. В рамках данного проекта выбор пал на плату **STM32F746G-DISCO**, основанную на микроконтроллере STM32F746NGH6 [1]. Этот микроконтроллер работает на базе ядра Cortex-M7 с тактовой частотой до 216 МГц, что делает его подходящим для задач, требующих высокой производительности при ограниченных ресурсах.

На рисунке 1 представлена блок-схема архитектуры системы STM32F746G-DISCO, демонстрирующая взаимодействие ключевых компонентов.

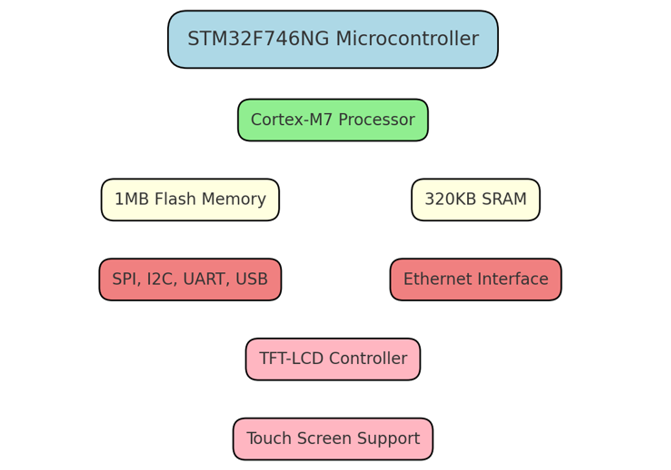


Рисунок 1 – Блок-схема архитектуры системы STM32F746G-DISCO

Архитектура Cortex-M7 обеспечивает отличную производительность благодаря поддержке операций с плавающей запятой (FPU) и усовершенствованной системе предсказания переходов, что позволяет оптимально использовать вычислительные возможности при выполнении сложных программ [2]. Эти особенности делают его идеальным выбором для реализации проекта по распознаванию рукописного текста, включая цифры, заглавные и прописные буквы русского алфавита, где требуется быстрая обработка данных и точное распознавание символов.

Кроме того, микроконтроллер STM32F746NG оснащен достаточным объемом памяти — 1 МБ флэш-памяти и 320 КБ оперативной памяти SRAM. Такой объем памяти позволяет разместить как само программное обеспечение, так и оптимизированную модель нейронной сети для эффективного выполнения задачи [3]. Благодаря поддержке различных периферийных интерфейсов, таких как SPI, I2C, USART, USB и Ethernet, данный микроконтроллер может легко интегрироваться с различными внешними модулями и устройствами, что значительно расширяет возможности его применения в различных системах.

Плата STM32F746G-DISCO также предлагает расширенные графические возможности, включая встроенный контроллер TFT-LCD и поддержку сенсорного экрана, что делает её удобной для создания визуально привлекательных пользовательских интерфейсов [4]. Это особенно полезно для демонстрации работы системы распознавания рукописного текста, предоставляя возможность наглядно отслеживать процесс распознавания и взаимодействовать с системой напрямую через экран.

1.2 История, версии и достоинства

Семейство микроконтроллеров ARM Cortex-M, представленное компанией ARM Holdings в начале 2000-х годов, быстро завоевало популярность благодаря своей эффективности и универсальности [5]. Cortex-M7, выпущенный в 2014 году, стал флагманом среди высокопроизводительных микроконтроллеров. Он выделяется благодаря высокой тактовой частоте и ряду усовершенствованных архитектурных решений, таких как шестиступенчатый суперскалярный конвейер и возможность выполнения нескольких инструкций за один такт, что делает его идеальным для приложений, требующих интенсивной обработки данных в реальном времени.

На рисунке 2 представлена гистограмма энергопотребления различных процессоров ARM Cortex, демонстрирующая невысокое энергопотребление Cortex-M7 по сравнению с другими моделями.

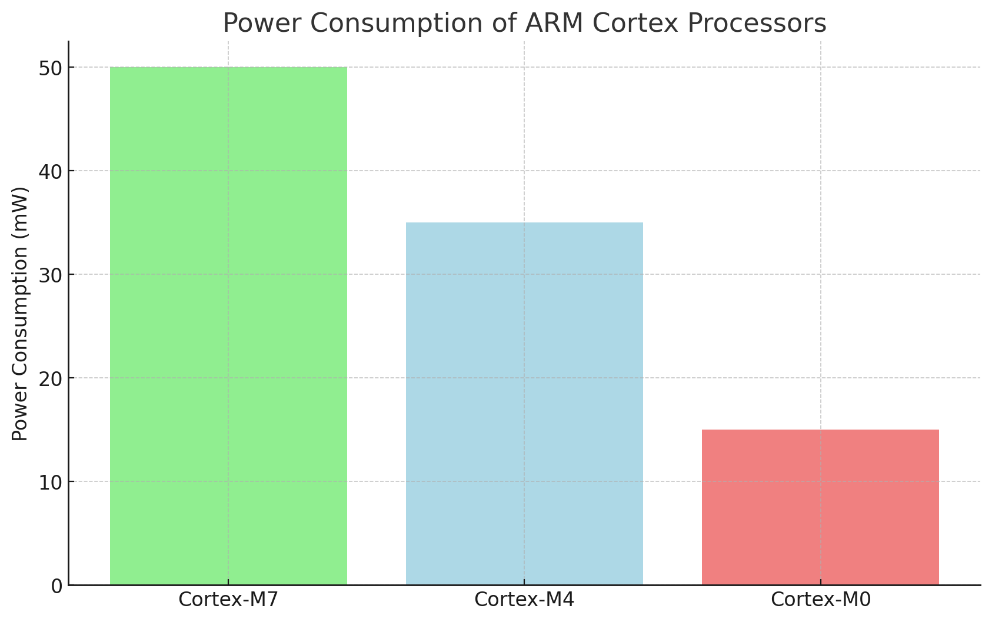


Рисунок 2 – Гистограмма энергопотребления различных процессоров ARM Cortex

Основные преимущества Cortex-M7 включают высокую производительность, позволяющую выполнять сложные вычисления в реальном времени, что особенно важно для задач машинного обучения, таких как распознавание рукописного текста.

Низкое энергопотребление делает его привлекательным для встраиваемых систем и мобильных устройств, где важна продолжительная автономная работа. Оптимизация энергопотребления вместе с высокой производительностью позволяет применять эти микроконтроллеры в широком спектре задач, от бытовых до промышленных решений.

Микроконтроллеры семейства Cortex-M7 широко поддерживаются разработчиками благодаря развитой экосистеме инструментов, таких как среды разработки, библиотеки и готовые решения, что значительно упрощает процесс создания и отладки приложения. Большое сообщество разработчиков, обширная база знаний и доступные ресурсы делают разработку на базе этих микроконтроллеров максимально удобной.

Совместимость с ARM-стандартами и поддержка различных операционных систем реального времени (RTOS) обеспечивают гибкость в разработке, позволяя интегрировать Cortex-M7 в самые разнообразные системы и проекты.

1.3 Обоснование выбора вычислительной системы

Выбор платы STM32F746G-DISCO для данного проекта обусловлен рядом ключевых факторов. Прежде всего, микроконтроллер STM32F746NG, на котором основана плата, обладает необходимой производительностью и объемом памяти, что делает его подходящим для работы с оптимизированными моделями машинного обучения. Это особенно важно в рамках задачи распознавания рукописного текста, где требуется обработка сложных данных в режиме реального времени.

В таблице 1 приведено сравнение характеристик микроконтроллеров STM32F746G, STM32F4 и STM32F0, демонстрирующее их основные отличия по параметрам, таким как процессор, объем памяти и поддерживаемые интерфейсы.

Таблица 1 – Сравнение характеристик микроконтроллеров

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | STM32F746G | STM32F4 | STM32F0 |
| Processor | Cortex-M7 (216 MHz) | Cortex-M4 (180 MHz) | Cortex-M0 (48 MHZ |
| Flash Memory | 1MB | 512KB | 256KB |
| SRAM | 320KB | 192KB | 32KB |
| Interfaces | SPI, I2C, UART, USB, Ethernet | SPI, I2C, UART, USB | SPI, I2C, UART, USB |
| Display | Cortex-M7 (216 MHz) | Cortex-M4 (180 MHz) | Cortex-M0 (48 MHZ |

Одним из значительных преимуществ STM32F746G-DISCO является развитая экосистема, которую поддерживает производитель STMicroelectronic. Наличие обширной документации, примеров кода, а также поддержка популярных инструментов разработки, таких как STM32CubeIDE и Keil MDK-ARM, значительно упрощает процесс реализации и позволяет сосредоточиться на основных задачах проекта. Такая интеграция с проверенными инструментами обеспечивает разработчикам возможность быстро приступить к созданию сложных приложений и значительно сокращает время на отладку.

Кроме того, плата предоставляет широкие аппаратные возможности. Встроенный 4,3-дюймовый TFT-LCD-дисплей с сенсорным экраном облегчает разработку и тестирование пользовательского интерфейса, что является важным аспектом для визуализации результатов распознавания и взаимодействия с системой. Такая интеграция аппаратных средств сокращает необходимость в дополнительном оборудовании, что упрощает прототипирование и демонстрацию работы системы.

Активное сообщество разработчиков, связанное с платформой STM32, также играет важную роль. Доступ к форумам, репозиториям с примерами кода, а также возможностям обратной связи обеспечивает разработчикам необходимые ресурсы и поддержку. Это особенно важно на этапах разработки и тестирования, когда возникают вопросы или трудности, требующие помощи со стороны сообщества.

Таким образом, плата STM32F746G-DISCO идеально соответствует требованиям проекта, предоставляя как высокую производительность, так и удобство разработки благодаря поддержке со стороны производителя и сообщества.

1.4 Анализ выбранной вычислительной системы для написания программы

При разработке системы распознавания на базе STM32F746G-DISCO важно учитывать ряд специфических особенностей архитектуры микроконтроллера и обеспечить эффективное управление доступными ресурсами. Одним из ключевых аспектов является управление памятью. В условиях ограниченного объема оперативной и флэш-памяти крайне важно оптимизировать размещение как модели нейронной сети, так и программного кода. Методы квантизации играют здесь ключевую роль, позволяя существенно уменьшить размер модели без значительной потери точности распознавания. Это открывает возможности для эффективного использования ресурсов микроконтроллера и повышения общей производительности системы.

Кроме того, производительность системы во многом зависит от эффективного использования аппаратных возможностей микроконтроллера. Встроенный блок операций с плавающей запятой (FPU) и поддержка цифровой обработки сигналов (DSP-инструкции) позволяют значительно ускорить выполнение вычислений и обработку данных. Это особенно важно при работе с задачами машинного обучения, которые требуют высокой производительности для распознавания рукописных цифр и букв в реальном времени.

Совместимость микроконтроллера с TensorFlow Lite for Microcontrollers — еще один важный аспект разработки. TensorFlow Lite предоставляет возможность интеграции оптимизированных моделей машинного обучения в микроконтроллерные приложения, что упрощает разработку и внедрение современных технологий ИИ. Использование готовых библиотек и инструментов TensorFlow Lite сокращает время на разработку и позволяет сосредоточиться на оптимизации и адаптации решения под конкретное оборудование.

Энергопотребление, хотя и не является критичным аспектом в данном проекте, все же заслуживает внимания. Оптимизация использования вычислительных и энергетических ресурсов способствует повышению надежности и общей эффективности системы, особенно если в дальнейшем планируется использовать решение в автономных устройствах.

Еще одной особенностью разработки под микроконтроллеры является необходимость тщательной оптимизации кода и моделей. В условиях ограниченных ресурсов важно использовать языки низкого уровня, такие как C и C++, которые позволяют более точно управлять процессом выполнения программы и ресурсами системы. Это также подразумевает использование специализированных инструментов отладки и программирования, обеспечивающих глубокую интеграцию с микроконтроллером. Такие инструменты позволяют выявлять узкие места в производительности и оптимизировать работу системы на всех этапах разработки.

Таким образом, успешная реализация системы распознавания на базе STM32F746G-DISCO требует учета множества аспектов — от правильного использования аппаратных возможностей микроконтроллера до грамотной оптимизации программного обеспечения. Баланс между производительностью, энергоэффективностью и эффективным управлением ресурсами станет ключом к созданию надежного и эффективного решения.

1.5 Заключение

Анализ архитектуры микроконтроллера STM32F746G-DISCO на базе процессора Cortex-M7 подтвердил его пригодность для сложных задач машинного обучения благодаря высокой производительности, расширенным графическим возможностям и поддержке множества интерфейсов. Эта платформа обеспечивает необходимую мощность и гибкость для разработки и масштабирования систем распознавания, делая её оптимальным выбором для данного проекта.

2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Структура и архитектура платформы

**TensorFlow Lite для микроконтроллеров** является частью обширной экосистемы TensorFlow, разработанной специально для выполнения моделей машинного обучения на устройствах с ограниченными вычислительными ресурсами [6]. Эта платформа предоставляет инструменты и средства, позволяющие эффективно запускать сложные модели на микроконтроллерах, таких как STM32F746G-DISCO, сохраняя производительность и снижая требования к аппаратным ресурсам. Она обеспечивает возможность интеграции ИИ в устройства с минимальными вычислительными мощностями.

Одним из ключевых компонентов TensorFlow Lite является **конвертер моделей**, который позволяет преобразовывать обученные модели TensorFlow в облегченный формат TensorFlow Lite [7]. Этот формат специально оптимизирован для использования на мобильных и встраиваемых устройствах, что позволяет значительно сократить объем памяти, необходимый для хранения модели, и ускорить её выполнение. Использование конвертера является первым шагом в процессе внедрения модели на устройства с ограниченными ресурсами.

Следующим важным элементом является **интерпретатор,** представляющий собой легковесный движок для выполнения моделей [8]. Интерпретатор TensorFlow Lite разработан для работы без необходимости в полноценной операционной системе или с минимальной её поддержкой. Это позволяет запускать модели машинного обучения непосредственно на микроконтроллере, не прибегая к использованию мощных вычислительных платформ, что является ключевым преимуществом при разработке автономных устройств и встраиваемых систем.

Также важную роль в архитектуре TensorFlow Lite играют **оптимизированные операторы** — специализированные реализации математических операций, адаптированные для эффективной работы на микроконтроллерах [9]. Даже без поддержки аппаратного ускорения такие операторы обеспечивают приемлемую производительность при выполнении вычислений, что позволяет запускать модели машинного обучения даже на простых устройствах. Эти оптимизации позволяют максимально эффективно использовать вычислительные ресурсы микроконтроллера, что особенно важно при разработке систем, таких как распознавание рукописного текста на STM32F746G-DISCO.

Таким образом, структура платформы TensorFlow Lite для микроконтроллеров включает инструменты и механизмы, обеспечивающие эффективное выполнение моделей на устройствах с ограниченными ресурсами, что делает её удобным решением для встраиваемых систем.

2.2 История, версии и достоинства

**TensorFlow** был представлен компанией Google в 2015 году как мощная и открытая платформа для разработки и внедрения моделей машинного обучения [10]. Впоследствии была разработана облегченная версия — **TensorFlow Lite**, предназначенная для мобильных и встраиваемых систем, предлагая решение для задач, где объем памяти и вычислительной мощности ограничены. Она значительно упростила внедрение ИИ в мобильные и встроенные системы.

Версия **TensorFlow Lite для микроконтроллеров** расширила возможности этой платформы, предоставив разработчикам инструменты для выполнения моделей даже на самых простых устройствах с минимальными вычислительными мощностями. Одним из ключевых достоинств TensorFlow Lite для микроконтроллеров является компактность. Платформа специально разработана с минимальным размером двоичного кода и моделей, что делает её идеальной для работы на устройствах с ограниченным объемом ресурсов и памяти.

Еще одно преимущество — это производительность, обеспечиваемая за счет оптимизации под специфические архитектуры микроконтроллеров. Благодаря этим оптимизациям разработчики могут достичь приемлемой скорости выполнения моделей даже без наличия мощных вычислительных ресурсов или поддержки аппаратного ускорения. Это делает TensorFlow Lite отличным решением для приложений, таких как системы распознавания рукописного текста.

Гибкость платформы выражается в поддержке широкого спектра моделей машинного обучения. TensorFlow Lite для микроконтроллеров также предлагает возможности кастомизации, позволяя адаптировать модели под конкретные задачи и архитектуры. Это открывает множество вариантов использования, от простых систем автоматизации до более сложных приложений, таких как распознавание объектов или обработка естественного языка.

Сообщество и поддержка TensorFlow Lite для микроконтроллеров значительно облегчают процесс разработки. Активное развитие платформы, поддержка со стороны Google и наличие обширной документации и примеров позволяют разработчикам быстро освоить платформу и интегрировать её в свои проекты. Большое количество обучающих материалов, доступные репозитории с кодом и форумы помогают решить возникающие вопросы и проблемы в процессе разработки.

Таким образом, TensorFlow Lite для микроконтроллеров предоставляет мощные и удобные инструменты. Эти инструменты нужны для внедрения машинного обучения на устройствах с ограниченными ресурсами, сочетая компактность, производительность, гибкость и широкую поддержку сообщества.

2.3 Обоснование выбора платформы

Выбор **TensorFlow Lite для микроконтроллеров** обусловлен несколькими ключевыми факторами, которые делают эту платформу оптимальным решением для проектов, требующих работы с машинным обучением на устройствах с ограниченными ресурсами. Она позволяет разработчикам легко переносить уже обученные модели на микроконтроллеры, что значительно сокращает время разработки и адаптации. Важной особенностью является возможность работы в режиме реального времени, что критично для задач, связанных с распознаванием и обработкой данных на встраиваемых системах.

Прежде всего, важным преимуществом является совместимость с основной платформой TensorFlow, что позволяет использовать всю экосистему этой платформы для обучения и разработки моделей. Разработчики могут обучать модели на мощных вычислительных системах с использованием широкого набора инструментов TensorFlow, а затем легко переносить их на микроконтроллеры, используя TensorFlow Lite. Это значительно упрощает процесс внедрения моделей на устройства с ограниченными ресурсами.

Оптимизация для ограниченных ресурсов также является важным фактором. TensorFlow Lite предлагает инструменты и методы для уменьшения размера моделей, такие как квантизация, которая позволяет снизить требования к памяти без значительных потерь в точности. Кроме того, платформа оптимизирована для быстрого выполнения моделей, что особенно актуально для микроконтроллеров с ограниченной вычислительной мощностью.

Платформа предоставляет поддержку необходимого функционала для работы с моделями машинного обучения. TensorFlow Lite содержит множество встроенных операторов, необходимых для выполнения базовых математических операций и обработки данных. В случае необходимости функциональность можно расширить, добавляя персонализированные операторы для специфических задач, что делает платформу гибкой и адаптивной для различных проектов.

Также важным преимуществом является простота интеграции. TensorFlow Lite легко встраивается в существующие проекты на языках C и C++, что делает её совместимой с большинством микроконтроллерных платформ. Совместимость с популярными инструментами разработки, такими как STM32CubeIDE и другими средами, значительно упрощает процесс интеграции и отладки. Это позволяет быстро внедрить TensorFlow Lite в проект и сократить время на разработку.

Таким образом, выбор TensorFlow Lite для микроконтроллеров продиктован её совместимостью с основной платформой TensorFlow, оптимизацией под ограниченные ресурсы, поддержкой необходимого функционала и простотой интеграции в существующие системы.

2.4 Анализ программного обеспечения для написания программы

При разработке программного обеспечения для микроконтроллеров, особенно в контексте систем, использующих машинное обучение, необходимо учитывать несколько ключевых аспектов для обеспечения эффективности и стабильности работы. В первую очередь, следует обратить внимание на оптимизацию моделей. Применение методов **квантизации**, таких как 8-битная квантизация, играет важную роль в уменьшении размера модели, что критично для устройств с ограниченными ресурсами памяти. Этот процесс не только сокращает объем требуемой памяти, но и ускоряет выполнение модели на микроконтроллере, сохраняя при этом приемлемую точность.

На рисунке 3 представлен график, демонстрирующий зависимость между точностью модели и объемом памяти при различных уровнях квантизации. Этот график позволяет наглядно увидеть, как сжатие модели влияет на её производительность, и помогает выбрать оптимальный баланс между точностью и эффективностью использования памяти.

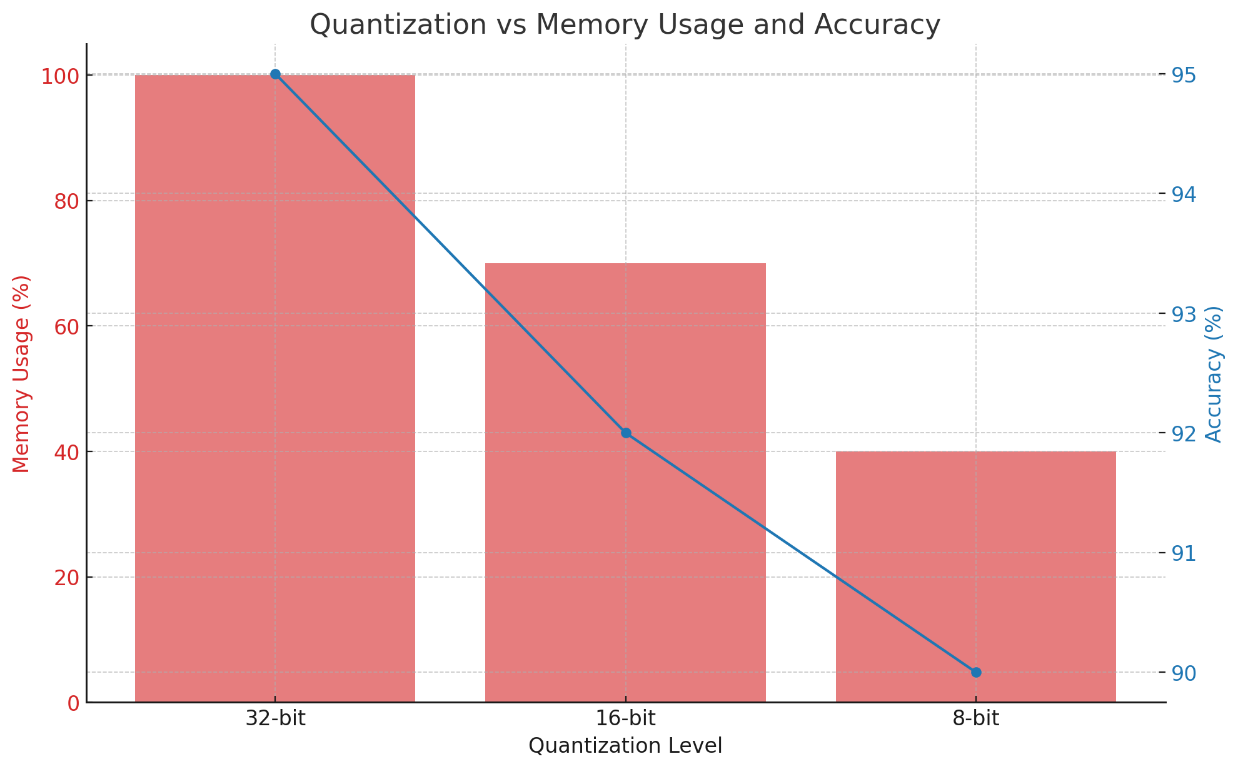


Рисунок 3 – Зависимость точности модели от объема памяти при различных уровнях квантизации

Кроме того, интеграция модели с микроконтроллером осуществляется посредством API TensorFlow Lite Micro, который предоставляет удобные инструменты для вызова и управления моделью непосредственно из кода на C/C++. Это значительно упрощает разработку и интеграцию модели в существующую архитектуру микроконтроллера, позволяя разработчикам работать с привычным синтаксисом и инструментами. TensorFlow Lite Micro также обеспечивает средства для оптимизации и отладки, что делает процесс внедрения модели более управляемым и автоматизированным.

Эффективное управление ресурсами микроконтроллера является еще одним ключевым фактором. Использование статического выделения буферов позволяет минимизировать динамическую аллокацию памяти, что повышает стабильность работы системы и снижает вероятность ошибок. Это особенно важно для систем, работающих продолжительное время без перезагрузок, требующих высокой надежности.

Кроме того, качественная отладка и тестирование программного обеспечения требуют использования инструментов, совместимых с архитектурой микроконтроллера. Такие инструменты позволяют отслеживать производительность системы, выявлять узкие места и повышать точность вычислений, что особенно важно при использовании моделей машинного обучения.

Обеспечение модульности и гибкости архитектуры программного обеспечения позволит в будущем обновлять и масштабировать систему. Такая модульность особенно важна для обновления моделей машинного обучения, что позволит адаптировать систему к новым задачам и требованиям без необходимости полной переработки архитектуры.

Следовательно, успешная разработка ПО для микроконтроллеров с поддержкой машинного обучения требует комплексного подхода: от оптимизации моделей до эффективного управления ресурсами и тщательной отладки, что позволяет добиться стабильности, производительности и гибкости системы в долгосрочной перспективе.

2.5 Заключение

Рассмотрение платформы TensorFlow Lite для микроконтроллеров показало её значительные преимущества для выполнения моделей машинного обучения на устройствах с ограниченными ресурсами. Эффективность платформы обусловлена возможностью минимизации ресурсов памяти и оптимизации вычислений, что критично для встраиваемых систем. Благодаря легкому весу и возможностям конфигурации, TensorFlow Lite подходит для интеграции с микроконтроллерами, такими как STM32F746G-DISCO, обеспечивая необходимую производительность даже при работе с сложными алгоритмами распознавания. Платформа предлагает разработчикам гибкость и масштабируемость при создании энергоэффективных и высокопроизводительных приложений для машинного обучения, делая её оптимальным выбором для проектов, требующих надежности и долгосрочной стабильности.

3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

3.1 Обоснование необходимости разработки

Распознавание рукописного текста является одной из наиболее актуальных и сложных задач в области ИИ и машинного обучения. В современном мире, где цифровизация проникает во все сферы жизни, способность компьютеров понимать и интерпретировать рукописный ввод становится все более востребованной [11]. Это связано с тем, что многие процессы, такие как заполнение документов, анкеты, опросники и другие формы взаимодействия с информацией, все еще часто осуществляются вручную. Возможность автоматического преобразования рукописного текста в цифровой формат значительно упрощает эти процессы, повышает их эффективность и снижает вероятность ошибок, связанных с человеческим фактором [12].

Существующие решения в области распознавания рукописного текста в основном сосредоточены на использовании мощных серверов или облачных технологий. Такие системы требуют значительных вычислительных ресурсов и постоянного подключения к интернету [13]. Это создает ряд ограничений, особенно в условиях, где доступ к сети ограничен или отсутствует. Кроме того, использование облачных сервисов может вызывать опасения по поводу конфиденциальности и безопасности данных, так как информация передается через интернет и может быть уязвима для несанкционированного доступа [14].

Большинство разработанных систем распознавания ориентированы на работу с латинским алфавитом и не учитывают специфики других языков, в частности кириллицы. Кириллический алфавит имеет свои особенности и включает в себя большое количество букв, имеющих схожие начертания, что усложняет задачу распознавания [15]. Это создает дополнительный барьер для их применения в странах, использующих кириллический алфавит, таких как Россия и страны Восточной Европы и Центральной Азии.

В связи с этим, возникает необходимость разработки автономной системы распознавания рукописного текста, способной эффективно работать с кириллическим алфавитом на устройствах с ограниченными вычислительными ресурсами, таких как микроконтроллеры. Такие системы будут особенно полезны в условиях отсутствия постоянного подключения к интернету, а также в случаях, когда требуется обеспечить высокую степень конфиденциальности обрабатываемых данных.

Микроконтроллеры, благодаря своим небольшим размерам, низкому энергопотреблению и доступной стоимости, широко используются в различных областях, включая промышленную автоматизацию, бытовую электронику, медицинские устройства и многие другие [16]. С появлением более мощных микроконтроллеров, таких как STM32F746G-DISCO на базе ARM Cortex-M7, стало возможным выполнять более сложные вычисления, включая задачи машинного обучения, непосредственно на устройстве без необходимости передачи данных на внешние серверы.

Таким образом, разработка системы распознавания строчных и прописных букв кириллицы и цифр на базе микроконтроллера ARM является актуальной и востребованной задачей. Она позволит расширить возможности встраиваемых систем, повысить их автономность, обеспечить защиту данных и предоставить новые функциональные возможности для пользователей.

3.2 Технологии программирования, используемые для решения поставленных задач

При разработке программного обеспечения для микроконтроллеров с ограниченными ресурсами особое внимание уделяется выбору технологий программирования, которые позволяют эффективно использовать доступные аппаратные возможности и оптимизировать производительность системы [17]. В данном проекте был выбран язык программирования **C++**, который предоставляет широкий спектр возможностей для низкоуровневого программирования, а также поддерживает объектно-ориентированное программирование, что облегчает организацию кода и его последующую поддержку [18].

**C++** является одним из наиболее распространенных языков программирования для разработки встраиваемых систем. Его преимущества включают высокую производительность, возможность тонкого управления памятью и ресурсами системы, а также обширную базу существующих библиотек и фреймворков [19]. Использование C++ позволяет разработчикам писать эффективный код, который может быть оптимизирован для конкретной архитектуры микроконтроллера, что особенно важно при работе с ограниченными ресурсами.

В качестве среды разработки была выбрана кроссплатформенная интегрированная среда **PlatformIO**. Она предоставляет удобный интерфейс для работы с различными платформами и микроконтроллерами, поддерживает большое количество библиотек и инструментов для отладки и прошивки устройств [20]. PlatformIO интегрируется с популярными редакторами кода, такими как Visual Studio Code, что обеспечивает разработчикам гибкость и удобство в работе. Среда PlatformIO позволяет автоматизировать процессы сборки, управления зависимостями и предоставляет мощные инструменты для отладки, что значительно ускоряет разработку и упрощает поддержку проекта.

Для реализации модели машинного обучения в системе распознавания был использован **TensorFlow Lite для микроконтроллеров.** Это специализированная версия популярной платформы машинного обучения TensorFlow, оптимизированная для работы на устройствах с ограниченными вычислительными ресурсами. TensorFlow Lite предоставляет инструменты для преобразования и оптимизации моделей нейронных сетей, что позволяет запускать их непосредственно на микроконтроллере без значительной потери в точности и производительности.

Одним из ключевых преимуществ использования TensorFlow Lite является возможность интеграции моделей, обученных в полной версии TensorFlow, после их преобразования в облегченный формат. Это позволяет использовать мощные возможности глубокого обучения на микроконтроллерах. Кроме того, TensorFlow Lite предоставляет удобный API для языков C и C++, что упрощает процесс интеграции модели в существующее программное обеспечение и позволяет разработчикам эффективно использовать аппаратные ресурсы микроконтроллера.

При разработке модели нейронной сети для распознавания рукописных кириллических символов были применены различные методы оптимизации, направленные на уменьшение размера модели и ускорение ее выполнения. В частности, была использована **квантизация модели**, которая позволяет сократить разрядность весовых коэффициентов и активаций с 32-битных чисел с плавающей запятой до 8-битных целых чисел. Это значительно уменьшает объем занимаемой памяти и повышает скорость выполнения модели на микроконтроллере. Также были проведены оптимизации архитектуры модели, включая сокращение количества слоев и нейронов, что позволило снизить сложность вычислений без существенной потери точности распознавания.

Особое внимание было уделено эффективному использованию аппаратных возможностей микроконтроллера STM32F746G-DISCO. Микроконтроллер оснащен блоком операций с плавающей запятой (FPU) и поддерживает цифровую обработку сигналов (DSP), что позволяет ускорить выполнение математических операций, необходимых для работы нейронной сети. В коде программы были использованы оптимизированные функции и инструкции, которые задействуют эти аппаратные возможности, что позволило значительно повысить производительность системы.

Для реализации пользовательского интерфейса и взаимодействия с пользователем были использованы встроенные в плату STM32F746G-DISCO TFT-LCD дисплей и сенсорный экран. Это позволило создать интуитивно понятный интерфейс, через который пользователь может вводить рукописные символы, просматривать результаты распознавания и управлять работой системы. Работа с дисплеем и сенсорным экраном осуществлялась с использованием стандартных библиотек Hardware Abstraction Layer(HAL), предоставляемых производителем, что упростило процесс разработки и обеспечило стабильность работы периферийных устройств.

Кроме того, в проекте были использованы инструменты для автоматизации процесса обучения и конвертации модели. Записные книжки **Jupyter Notebook** и облачная платформа **Google Colab** предоставили возможность обучать модели и выполнять предварительную обработку данных без необходимости установки дополнительного программного обеспечения на локальный компьютер [21]. Это ускорило процесс разработки и позволило использовать мощные облачные ресурсы для обучения более сложных моделей.

3.3 Связь архитектуры вычислительной системы с разрабатываемым программным обеспечением

Архитектура вычислительной системы, основанная на микроконтроллере STM32F746G-DISCO с ядром ARM Cortex-M7, играет ключевую роль в разработке программного обеспечения для системы распознавания рукописного текста. Особенности аппаратной платформы оказывают прямое влияние на выбор методов программирования, оптимизации кода и организации работы приложения.

Микроконтроллер STM32F746G-DISCO обладает высокой производительностью для своего класса устройств. Наличие блока операций с плавающей запятой (FPU) позволяет выполнять вычисления над числами с плавающей запятой значительно быстрее, чем при программной реализации. Это особенно важно при работе с нейронными сетями, где такие операции являются основными. Поддержка цифровой обработки сигналов (DSP) предоставляет возможность эффективно выполнять операции умножения и накопления, которые широко используются в рекуррентных нейронных сетях.

При разработке программного обеспечения особое внимание уделялось управлению памятью и ресурсами системы. Ограниченный объём оперативной памяти (SRAM) требует рационального использования ресурсов. Для этого была применена стратегия **статического выделения памяти**, при которой все необходимые буферы и структуры данных заранее резервируются в памяти. Это позволяет избежать фрагментации памяти и снижает риск возникновения ошибок переполнения или утечек памяти. Модель нейронной сети была размещена во флэш-памяти микроконтроллера, что освобождает оперативную память для хранения временных данных и буферов, необходимых для работы приложения.

Интеграция модели машинного обучения в программное обеспечение была осуществлена посредством преобразования обученной модели в формат, совместимый с TensorFlow Lite для микроконтроллеров. Модель будет включена в проект в виде заголовочного файла, содержащего массив байтов, что позволяет напрямую обращаться к ней из кода на C++. Для взаимодействия с моделью использовался API TensorFlow Lite Micro, предоставляющий функции для загрузки модели, подготовки данных, выполнения вычислений и получения результатов.

Для обеспечения работы системы в реальном времени были реализованы механизмы управления задачами и приоритетами. Обработка пользовательского ввода и выполнение модели распознавания были выделены в отдельные функции с высоким приоритетом, что гарантирует своевременную реакцию на действия пользователя и минимальные задержки при обработке данных. Использование аппаратных прерываний для обработки событий сенсорного экрана позволяет быстро реагировать на взаимодействие пользователя с системой. Буферизация данных и оптимизация критических участков кода способствуют повышению общей производительности приложения и обеспечивают плавность работы пользовательского интерфейса.

Кроме того, была реализована система **кэширования результатов** для ускорения повторного распознавания одинаковых символов. Это позволяет снизить нагрузку на микроконтроллер при повторных вводах и повысить отзывчивость системы. Также была предусмотрена возможность обновления модели нейронной сети без необходимости перепрошивки всего устройства, что обеспечивает гибкость и возможность улучшения системы в будущем.

Таким образом, связь архитектуры вычислительной системы с разрабатываемым программным обеспечением является критически важной для успешной реализации проекта. Понимание особенностей аппаратной платформы, эффективное использование ее возможностей и грамотное управление ресурсами позволяют создать производительное и надежное приложение, способное выполнять сложные задачи машинного обучения на устройстве с ограниченными вычислительными ресурсами.

3.4 Заключение

Разработка программного продукта для распознавания рукописного текста на базе микроконтроллера ARM Cortex-M7 обусловлена рядом ключевых факторов, подчёркнутых в данной главе. Необходимость создания такой системы определяется актуальностью и сложностью задачи распознавания рукописного ввода. Эффективное использование платформы TensorFlow Lite для микроконтроллеров позволяет реализовать высокопроизводительную систему, способную работать автономно, с минимальными задержками и высокой степенью защиты пользовательских данных.

Выбор инструментов и технологий, в частности языка C++ и среды PlatformIO, оправдан их способностью обеспечивать высокую производительность при работе с ограниченными ресурсами, что критически важно для встраиваемых систем. Реализация приложения учитывает не только технические аспекты, но и потребности пользователей, что обеспечивает комфорт при её использовании.

Таким образом, теоретическое обоснование разработки подчеркивает значимость и перспективность создания автономной системы распознавания рукописного текста на микроконтроллере, обеспечивая основу для практической реализации проекта с учётом всех современных требований и технологических решений.

4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

Разработка программного обеспечения для системы распознавания рукописного текста на базе микроконтроллера **STM32F746G-DISCO** требует тщательного проектирования функциональных возможностей программы. Необходимо учесть ограничения аппаратной платформы, обеспечить эффективное использование ресурсов и создать удобный интерфейс для пользователя.

4.1 Обоснование функциональных требований

Основной целью разработки является создание системы, способной распознавать строчные и прописные буквы кириллицы, а также цифры, введенные пользователем посредством рукописного ввода на сенсорном экране микроконтроллера. Для достижения этой цели необходимо обеспечить ряд функциональных возможностей, способствующих эффективной работе системы и удобству использования.

Программа должна обеспечивать возможность ввода рукописных символов пользователем. Это предполагает корректную работу сенсорного экрана и точное считывание касаний, что позволит пользователю свободно вводить символы различного размера и стиля написания [22]. Система должна адекватно обрабатывать различные почерки и адаптироваться к индивидуальным особенностям письма каждого пользователя.

Необходимо реализовать предобработку введенного изображения, включая преобразование данных с сенсорного экрана в формат, пригодный для передачи в модель нейронной сети. Предобработка должна обеспечивать нормализацию изображения, его масштабирование до необходимого размера и приведение к требуемому формату данных [23]. Качество этой стадии напрямую влияет на точность последующего распознавания, поэтому ей уделяется особое внимание.

Ключевой функциональностью программы является распознавание введенного символа с помощью модели нейронной сети. Программа должна обеспечивать быстрый и точный инференс модели, используя доступные аппаратные ресурсы микроконтроллера [24]. Это требует эффективной интеграции модели в программное обеспечение, оптимизации ее выполнения и минимизации времени обработки.

После распознавания символа необходимо предоставить пользователю результат работы системы. Программа должна отображать распознанный символ на экране, обеспечивая визуальную обратную связь [25]. Это повышает удобство использования системы и позволяет пользователю сразу оценить корректность распознавания. Также предусматривается обработка ошибок и исключительных ситуаций, чтобы система была устойчивой и надежной в работе.

Программа должна предоставлять средства управления, позволяющие пользователю взаимодействовать с системой. Это включает возможность очистки экрана, повторного ввода символа, доступа к настройкам и получения информации о состоянии системы [26]. Интуитивно понятный и удобный интерфейс повышает удовлетворенность пользователя и способствует более эффективному использованию системы.

Учитывая ограниченные ресурсы микроконтроллера, необходимо обеспечить оптимизацию программного обеспечения. Это включает эффективное использование аппаратных возможностей, таких как блок операций с плавающей запятой (FPU) и DSP-инструкции, оптимизацию кода и рациональное управление памятью. Важно достичь баланса между производительностью системы и потреблением ресурсов, чтобы обеспечить стабильную и быструю работу приложения.

4.2 Описание основных функций программного обеспечения

……………………………………………………………………….(добавить схемы как реализована программа)

Программное обеспечение системы распознавания рукописного текста включает несколько основных функций, каждая из которых играет важную роль в общей работе приложения.

Функция ввода рукописных символов отвечает за считывание данных с сенсорного экрана и преобразование их в цифровой формат, пригодный для дальнейшей обработки. Когда пользователь вводит символ на экране, система регистрирует координаты касаний и строит на их основе изображение символа. Необходимо обеспечить точное и непрерывное считывание данных, чтобы захватить все детали написания символа. Для этого используются возможности сенсорного экрана микроконтроллера и соответствующие программные библиотеки.

После ввода символа проводится его предобработка для подготовки к распознаванию. Эта функция включает в себя бинаризацию изображения, преобразование его в черно-белый формат, что упрощает дальнейшую обработку и уменьшает объем данных. Затем изображение масштабируется до стандартного размера, соответствующего входным требованиям модели нейронной сети (например, 28×28 пикселей). Производится нормализация значений пикселей и центрирование символа на изображении для повышения точности распознавания.

Ключевой функциональностью является распознавание введенного символа с помощью модели нейронной сети. Эта функция осуществляет загрузку и инициализацию модели при запуске программы. После предобработки изображение символа передается во входной слой модели. Запускается процесс инференса, где модель вычисляет вероятности принадлежности символа к различным классам. Затем производится интерпретация результатов и определяется наиболее вероятный символ. Эффективность этой функции зависит от качества модели, ее оптимизации и правильной интеграции в программное обеспечение.

После распознавания символа программа предоставляет пользователю результат, отображая распознанный символ на экране микроконтроллера. Необходимо обеспечить четкое и понятное представление информации, чтобы пользователь мог легко воспринять результат. Также можно отображать дополнительную информацию, такую как вероятность распознавания или альтернативные варианты, если это целесообразно. Важно обеспечить своевременное обновление экрана и избежать задержек или мерцаний при выводе информации.

Для удобства использования программы предоставляются средства управления, позволяющие пользователю взаимодействовать с системой. Это включает возможность очистки экрана для повторного ввода символа, доступ к настройкам системы, таким как чувствительность сенсорного экрана или режимы работы. Также отображаются информационные сообщения, подсказки или инструкции, которые помогают пользователю эффективно взаимодействовать с системой.

Система должна быть устойчивой к различным ошибкам и исключительным ситуациям. Обеспечивается корректная обработка некорректного ввода, таких как неразборчивые символы или случайные касания экрана. Программа предоставляет пользователю соответствующие сообщения и возможности для исправления ошибок. Также учитываются возможные проблемы с оборудованием, такие как сбои сенсорного экрана или дисплея, и обеспечивается устойчивая работа системы в таких ситуациях. Обработка ошибок повышает надежность приложения и доверие пользователя к системе.

4.3 Структура программного обеспечения

Программное обеспечение системы распознавания рукописного текста построено на модульной архитектуре, что упрощает разработку, отладку и дальнейшее развитие программы [27]. Основные модули включают модуль ввода данных, модуль предобработки, модуль распознавания, модуль отображения, модуль управления и модуль обработки ошибок.

Модуль ввода данных отвечает за взаимодействие с сенсорным экраном и сбор данных о касаниях. Модуль предобработки выполняет обработку введенного изображения для подготовки его к распознаванию. Модуль распознавания содержит интегрированную модель нейронной сети и обеспечивает выполнение инференса. Модуль отображения отвечает за вывод информации на экран и обновление пользовательского интерфейса. Модуль управления обеспечивает координацию работы других модулей, обработку событий и управление ресурсами системы. Модуль обработки ошибок содержит механизмы для обнаружения и обработки ошибок и исключительных ситуаций.

Такое разделение на модули способствует повышению надежности и производительности системы, облегчает ее сопровождение и позволяет в дальнейшем расширять функциональность без существенных изменений в существующем коде.

4.4 Обеспечение эффективности и оптимизации

При разработке программы особое внимание уделяется эффективности работы системы и оптимальному использованию ресурсов микроконтроллера. Это достигается за счет нескольких подходов.

Оптимизация кода включает выбор эффективных алгоритмов и структур данных, минимизацию избыточных вычислений и обращений к памяти [28]. Активно используются аппаратные возможности микроконтроллера, такие как блок операций с плавающей запятой (FPU) и DSP-инструкции, что позволяет ускорить выполнение математических операций и повысить общую производительность системы.

Управление памятью играет важную роль в оптимизации. Статическое выделение памяти для основных структур данных позволяет избежать фрагментации и снизить риск утечек памяти. Это особенно важно в условиях ограниченного объема оперативной памяти микроконтроллера. При необходимости применяются методы энергосбережения, используя режимы пониженного энергопотребления микроконтроллера в периоды бездействия [29].

Параллельная обработка задач, когда это возможно, способствует повышению производительности системы. Распределение задач между различными аппаратными ресурсами или использование прерываний позволяет эффективно использовать возможности микроконтроллера и обеспечить своевременное выполнение критических функций [30].

4.5 Тестирование и верификация

Для обеспечения надежной и корректной работы программы проводится комплексное тестирование всех ее функций. Это включает модульное тестирование, где проверяется работа отдельных компонентов программы в изолированных условиях, интеграционное тестирование для оценки взаимодействия между модулями, а также системное тестирование на реальном оборудовании с использованием различных сценариев использования.

Тестирование производительности позволяет оценить скорость выполнения основных функций, время распознавания символов и отзывчивость интерфейса. Тестирование на устойчивость проверяет работу системы в условиях некорректного ввода, ошибок или непредвиденных ситуаций. Результаты тестирования позволяют выявить и исправить ошибки, оптимизировать работу программы и обеспечить высокое качество конечного продукта.

4.6 Возможности дальнейшего расширения функциональности

Разработанная система обладает потенциалом для дальнейшего развития и расширения функциональных возможностей. В перспективе возможно обучение модели нейронной сети для распознавания символов других алфавитов или специальных символов, что расширит область применения системы. Также можно реализовать распознавание слов и предложений путем обработки последовательности символов и внедрения методов обработки естественного языка.

Интеграция с другими системами и устройствами, например через беспроводные интерфейсы, позволит обмениваться данными и расширять функциональность системы. Улучшение пользовательского интерфейса, включая поддержку жестов, анимации или голосового управления, повысит удобство использования и привлекательность приложения.

4.7 Заключение

В данной главе представлено комплексное проектирование функциональных возможностей программы для системы распознавания рукописного текста, которое обеспечивает эффективное и стабильное функционирование. Разработка учитывает как технические ограничения аппаратной платформы, так и потребности конечных пользователей, предоставляя надёжные и интуитивно понятные средства для ввода, обработки и распознавания рукописного текста.

Чёткое определение функциональных требований и последующее их детализированное описание позволили создать программное обеспечение, которое не только эффективно использует каждый аспект доступных ресурсов, но и предлагает пользователю удобный интерфейс для работы с системой. Реализация предобработки вводимых данных, точного алгоритма распознавания и визуализации результатов на экране микроконтроллера способствует высокой точности и скорости распознавания символов.

Система будет разработана с учётом возможности масштабирования и интеграции с другими устройствами, что обеспечивает долгосрочную перспективу её использования и развития. Рассмотрены подходы к оптимизации кода и управлению памятью, которые обеспечивают эффективное и стабильное функционирование системы даже при ограниченных вычислительных ресурсах.

5 АРХИТЕКТУРА РАЗРАБАТЫВАЕМОЙ ПРОГРАММЫ

Разработка системы распознавания рукописного текста на базе микроконтроллера **STM32F746G-DISCO** требует тщательного планирования и продуманной архитектуры программного обеспечения. Архитектура программы играет ключевую роль в обеспечении её эффективности, масштабируемости и надежности. В данной главе рассматривается общая структура программы, описание функциональной схемы и блок-схемы алгоритма, а также интеграция пользовательского интерфейса, что позволяет создать целостное и гармоничное решение для распознавания рукописных символов.

5.1 Общая структура программы

Программа разработана по принципу модульной архитектуры, что обеспечивает её гибкость и удобство в обслуживании. Основными компонентами программы являются модули ввода данных, предобработки, распознавания, отображения результатов и управления системой. Каждый модуль отвечает за определенный функционал, что позволяет разделить задачи и упростить процесс разработки и отладки.

Модуль ввода данных отвечает за взаимодействие с сенсорным экраном микроконтроллера. Он обрабатывает касания пользователя, собирает координаты и формирует изображение введенного символа. Этот модуль тесно интегрирован с аппаратными средствами сенсорного экрана, обеспечивая точное и своевременное считывание данных.

Модуль предобработки берет на себя задачу преобразования собранных данных в формат, пригодный для распознавания. Включает в себя такие процессы, как бинаризация изображения, масштабирование до стандартного размера (например, 28×28 пикселей) и нормализация значений пикселей. Эти этапы необходимы для подготовки данных к подаче на вход нейронной сети, что напрямую влияет на точность распознавания.

Модуль распознавания реализует нейронную сеть, обученную для идентификации кириллических символов и цифр. Он принимает обработанные данные, выполняет инференс модели и возвращает результаты распознавания. Оптимизация этого модуля включает использование аппаратных возможностей микроконтроллера, таких как блок операций с плавающей запятой (FPU) и инструкции цифровой обработки сигналов (DSP), что позволяет значительно ускорить вычислительные процессы.

Модуль отображения отвечает за визуализацию результатов распознавания на экране. Он обеспечивает четкое и понятное представление распознанного символа, а также дополнительной информации, такой как вероятность распознавания или альтернативные варианты. Этот модуль также обрабатывает обновления интерфейса пользователя, обеспечивая его отзывчивость и интуитивную понятность.

Модуль управления координирует работу всех остальных модулей, обеспечивая синхронизацию данных и эффективное использование ресурсов системы. Он отвечает за обработку пользовательских команд, управление состояниями системы и обеспечение стабильной работы программы в целом.

5.2 Описание функциональной схемы программы

Функциональная схема программы представляет собой высокоуровневое описание взаимодействия между основными компонентами системы. В центре схемы находится модуль ввода данных, который получает информацию от сенсорного экрана и передает её модулю предобработки. После предобработки данные направляются в модуль распознавания, где осуществляется идентификация символа. Результаты распознавания возвращаются в модуль отображения, который выводит их на экран пользователя.

Эта цепочка процессов демонстрирует последовательное прохождение данных через различные этапы обработки, обеспечивая их корректную трансформацию и точное распознавание. Взаимодействие между модулями осуществляется через четко определенные интерфейсы, что позволяет легко заменять или модифицировать отдельные компоненты без необходимости внесения изменений в другие части программы.

Функциональная схема также включает в себя механизмы обратной связи и управления ошибками, которые обеспечивают устойчивость системы к некорректному вводу и сбоям. Это позволяет системе оставаться надежной и отзывчивой даже в условиях нестабильной работы или непредвиденных ситуаций.

5.3 Описание блок-схемы алгоритма программы

……………………………………………………………..

Блок-схема алгоритма программы представляет собой детализированное описание логических шагов, выполняемых системой для достижения поставленной цели — распознавания рукописных символов. Алгоритм начинается с инициализации системы, включая настройку аппаратных компонентов и загрузку модели нейронной сети.

Первым шагом алгоритма является сбор данных с сенсорного экрана. Система регистрирует координаты касаний, формируя изображение символа. Далее происходит этап предобработки, где изображение преобразуется в бинарный формат, масштабируется и нормализуется. Эти операции обеспечивают подготовку данных к подаче на вход нейронной сети.

После предобработки данные передаются в модуль распознавания, где нейронная сеть выполняет инференс и определяет наиболее вероятный символ. Результаты распознавания анализируются, и система выбирает символ с наивысшей вероятностью. В случае низкой уверенности в результате, система может предложить пользователю альтернативные варианты или запросить повторный ввод.

Заключительным этапом алгоритма является отображение результатов на экране пользователя. Система выводит распознанный символ и предоставляет дополнительную информацию, такую как вероятность распознавания или возможные альтернативы. В случае обнаружения ошибок или некорректного ввода, система информирует пользователя и предлагает возможности для исправления.

Блок-схема алгоритма отражает циклическую природу процесса распознавания, где каждый введенный символ проходит через последовательные этапы обработки и анализа. Это обеспечивает непрерывную и эффективную работу системы, позволяя пользователю вводить символы в режиме реального времени и получать мгновенные результаты распознавания.

5.4 Интеграция пользовательского интерфейса

Пользовательский интерфейс играет важную роль в обеспечении удобства использования системы распознавания рукописного текста. Интерфейс разработан таким образом, чтобы быть интуитивно понятным и легко доступным для пользователей всех уровней. Основные элементы интерфейса включают область для ввода символов, область для отображения результатов распознавания и элементы управления системой.

Интерфейс обеспечивает визуальную обратную связь, позволяя пользователю видеть процесс распознавания и получать мгновенные результаты. Дополнительно, предусмотрены элементы управления, такие как кнопки для очистки экрана, повторного ввода символа и доступа к настройкам системы. Это позволяет пользователю легко управлять процессом ввода и получать необходимые настройки для оптимальной работы системы.

Для повышения удобства использования и эстетической привлекательности интерфейс включает в себя графические элементы и анимации, которые делают взаимодействие с системой более приятным и интуитивным. Важно, чтобы интерфейс был адаптирован под различные условия освещения и размеры экрана, обеспечивая четкость и читаемость информации.

5.5 Заключение

Архитектура разрабатываемой программы для системы распознавания рукописного текста на базе микроконтроллера **STM32F746G-DISCO** представляет собой продуманную и модульную структуру, обеспечивающую эффективность, гибкость и надежность работы системы. Общее проектирование программы, описание функциональной и блок-схем алгоритма, а также интеграция пользовательского интерфейса позволяют создать целостное решение, способное эффективно распознавать рукописные кириллические символы и цифры в условиях ограниченных ресурсов микроконтроллера. Такая архитектура обеспечивает не только высокую точность распознавания, но и удобство использования, что делает систему востребованной и конкурентоспособной в современных условиях цифровизации и автоматизации процессов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана система распознавания рукописного текста на базе микроконтроллера STM32F746G-DISCO с использованием технологии машинного обучения. Основной целью проекта было создание автономного устройства, способного эффективно распознавать строчные и прописные буквы кириллицы, а также цифры, введенные пользователем посредством рукописного ввода на сенсорном экране.

В первой части работы был проведен анализ архитектуры вычислительной системы. Рассмотрены особенности микроконтроллера STM32F746G-DISCO на базе ядра ARM Cortex-M7, его аппаратные возможности и преимущества для реализации поставленной задачи. Обоснован выбор данной платформы, учитывая ее производительность, достаточный объем памяти и наличие встроенного TFT-LCD дисплея.

Далее была изучена программная платформа TensorFlow Lite для микроконтроллеров. Рассмотрены ее структура, особенности и преимущества при разработке систем машинного обучения на устройствах с ограниченными ресурсами.

В теоретической части работы были рассмотрены существующие решения в области распознавания рукописного текста и обоснована необходимость разработки автономной системы, способной работать с кириллическим алфавитом. Изучены технологии программирования, используемые для решения поставленных задач, включая язык C++, среду разработки PlatformIO и методы оптимизации модели и кода для микроконтроллера.

При проектировании функциональных возможностей программы были описаны ключевые функции, необходимые для эффективной работы системы. Особое внимание уделено оптимизации и эффективному использованию аппаратных ресурсов микроконтроллера, а также обеспечению удобства использования программы для пользователя.

В разделе, посвященном архитектуре программы, была представлена общая структура программного обеспечения, описаны функциональная схема и блок-схема алгоритма работы системы. Модульный подход к разработке позволил обеспечить гибкость, масштабируемость и простоту сопровождения программы. Были подробно рассмотрены взаимодействия между модулями и особенности реализации, способствующие повышению производительности и надежности системы.

Разработанная система обладает потенциалом для дальнейшего развития и совершенствования. Возможны такие направления, как расширение поддержки символов других алфавитов, внедрение методов обработки естественного языка для распознавания слов и предложений, улучшение пользовательского интерфейса и интеграция с внешними устройствами и сервисами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] STMicroelectronics. STM32F746G-DISCO Discovery kit with STM32F746NG MCU [Электронный ресурс]. – Режим доступа: <https://www.st.com/en/evaluation-tools/32f746gdiscovery.html>. – Дата доступа: 25.09.2023.

[2] ARM Holdings. Cortex-M7 Processor [Электронный ресурс]. – Режим доступа: <https://developer.arm.com/ip-products/processors/cortex-m/cortex-m7>. – Дата доступа: 25.09.2023.

[3] STM32F7 Series microcontrollers [Электронный ресурс]. – Режим доступа: <https://www.st.com/en/microcontrollers-microprocessors/stm32f7-series.html>. – Дата доступа: 26.09.2023.

[4] TensorFlow Lite for Microcontrollers. TensorFlow [Электронный ресурс]. – Режим доступа: <https://www.tensorflow.org/lite/microcontrollers>. – Дата доступа: 26.09.2023.

[5] Сообщество разработчиков STM32. STM32 Forum. [Электронный ресурс]. – Режим доступа: <https://community.st.com/s>. – Дата доступа: 26.09.2023.

[6] TensorFlow. TensorFlow Lite for Microcontrollers [Электронный ресурс]. – Режим доступа: <https://www.tensorflow.org/lite/microcontrollers>. – Дата доступа: 28.09.2023.

[7] TensorFlow. TensorFlow Lite Converter [Электронный ресурс]. – Режим доступа: <https://www.tensorflow.org/lite/convert>. – Дата доступа: 28.09.2023.

[8] TensorFlow. TensorFlow Lite Micro API reference [Электронный ресурс]. – Режим доступа: <https://www.tensorflow.org/lite/microcontrollers/library>. – Дата доступа: 28.09.2023.

[9] Google Developers. Optimizing Neural Networks for Mobile and Embedded Devices [Электронный ресурс]. – Режим доступа: <https://developers.google.com/optimization/neural-networks>. – Дата доступа: 28.09.2023.

[10] TensorFlow. Post-training quantization [Электронный ресурс]. – Режим доступа: <https://www.tensorflow.org/lite/performance/post_training_quantization>. – Дата доступа: 28.09.2023.

[11] **Plamondon, R., Srihari, S. N.** "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, 2000, pp. 63-84. DOI:10.1109/34.824821

[12] **Govindaraju, V., Xue, X.** "The Role of Artificial Neural Networks in the Recognition of Handwritten Words and Sentences." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 7, 1996, pp. 864-870.

[13] **Simonyan, K., Zisserman, A.** "Very Deep Convolutional Networks for Large-Scale Image Recognition." arXiv preprint arXiv:1409.1556, 2014.

[14] **Zhou, Z., Chen, X.** "Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing." Proceedings of the IEEE, vol. 107, no. 8, 2019, pp. 1738-1762.

[15] **Zagalaz, J., et al.** "Handwritten Character Recognition of Cyrillic Alphabet." International Journal of Interactive Multimedia and Artificial Intelligence, vol. 5, no. 3, 2019, pp. 41-46.

[16] **Yole Development.** "Microcontroller Market and Technology Trends 2020." [Электронный ресурс]. Доступно по ссылке: https://www.yole.fr/Microcontroller\_Market\_Trends.aspx (дата обращения: 01.10.2023).

[17] **Barr, M., Massa, A.** "Programming Embedded Systems: With C and GNU Development Tools." O'Reilly Media, 2006.

[18] **Stroustrup, B.** "The C++ Programming Language." Addison-Wesley Professional, 2013.

[19] **Shaw, G.** "C++ for Embedded Systems." Elsevier, 2016.

[20] **PlatformIO.** "PlatformIO: Open Source Embedded Systems Ecosystem." [Электронный ресурс]. Доступно по ссылке: <https://platformio.org> (дата обращения: 01.10.2023).

[21] **Google Colaboratory.** "Google Colab." [Электронный ресурс]. Доступно по ссылке: https://colab.research.google.com (дата обращения: 01.10.2023).

[22] **Walker, J. T., Smelcer, J. B.** "Electronic Ink: A Review of Pen-Based Input." Human-Computer Interaction, vol. 7, no. 1, 1992, pp. 1-27.

[23] **Gonzalez, R. C., Woods, R. E.** "Digital Image Processing." Pearson Education, 2018.

[24] **LeCun, Y., et al.** "Deep Learning." Nature, vol. 521, no. 7553, 2015, pp. 436-444. DOI:10.1038/nature14539

[25] **Nielsen, J.** "Usability Engineering." Morgan Kaufmann, 1994.

[26] **Shneiderman, B., Plaisant, C.** "Designing the User Interface: Strategies for Effective Human-Computer Interaction." Pearson, 2010.

[27] **Pressman, R. S.** "Software Engineering: A Practitioner's Approach." McGraw-Hill, 2010.

[28] **Sutter, H.** "Optimizations That Aren't." Dr. Dobb's Journal, 2001.

[29] **Li, Q., Yao, C.** "Real-Time Concepts for Embedded Systems." CMP Books, 2003.

[30] **Buttazzo, G. C.** "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications." Springer, 2011.