УДК 004

Разработка браузерного QR-генератора и декодера в парадигме Zero-Dependency методами промпт-инжиниринга

Пайдершев Алексей Константинович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассматривается процесс создания браузерного приложения для генерации и декодирования QR-кодов, реализованного в парадигме Zero-Dependency. С помощью большой языковой модели Qwen Studio и методологии итеративного промпт-инжиниринга разработан инструмент, опирающийся на стандарт ISO/IEC 18004. Приложение включает в себя прямой алгоритм кодирования (арифметика конечных полей Галуа GF(256), коды Рида-Соломона, BCH-кодирование, маскирование) и обратный алгоритм компьютерного зрения (бинаризация методом Отсу, поиск матрицы, считывание данных и проверка синдромов). Особое внимание в работе уделено анализу процесса отладки. Эксперимент показал, что нейросеть безупречно справляется с линейной математикой помехоустойчивого кодирования, однако испытывает фундаментальные трудности при отладке пространственной логики (двумерных матриц) исключительно через текстовые описания, что выявляет границы применимости генеративного подхода в задачах, требующих визуального контроля координат.
Ключевые слова: Qwen Studio, нейросети, промпт-инжиниринг, вайб-кодинг, Zero-Dependency.

Development of a browser-based QR generator and decoder in the Zero-Dependency paradigm using prompt engineering methods

Paysershev Alexey Konstantinovich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article examines the process of creating a browser-based application for generating and decoding QR codes, implemented in the Zero-Dependency paradigm. Using the large language model Qwen Studio and the iterative prompt engineering methodology, a tool was developed that is based on the ISO/IEC 18004 standard. The application includes a direct encoding algorithm (arithmetic of finite Galois fields GF(256), Reed–Solomon codes, BCH encoding, masking) and a reverse algorithm of computer vision (Otsu's method for binarization, matrix search, data reading, and syndrome checking). Special attention in the work is paid to the analysis of the debugging process. The experiment showed that the neural network copes

flawlessly with the linear mathematics of error-correcting coding, but it encounters fundamental difficulties when debugging spatial logic (two-dimensional matrices) solely through textual descriptions, which reveals the limits of the generative approach in tasks requiring visual control of coordinates.

Keywords: Qwen Studio, neural networks, prompt engineering, vibe coding, Zero-Dependency.

1 Введение

1.1 Актуальность

Технология QR-кодов, стандартизированная в ISO/IEC 18004, прочно вошла в повседневную цифровую среду, обеспечивая быструю передачу данных, бесконтактные платежи и авторизацию. За внешней простотой квадратной матрицы скрывается сложный математический аппарат: помехоустойчивое кодирование, операции в конечных полях, применение масок и строгая геометрическая раскладка модулей.

Создание программ для работы с QR-кодами без привлечения сторонних фреймворков и библиотек требует от разработчика глубокого понимания как абстрактной алгебры, так и алгоритмов компьютерного зрения. В связи с развитием генеративных моделей возникает закономерный вопрос: способна ли большая языковая модель автономно реализовать подобный двунаправленный конвейер и способна ли она эффективно отлаживать пространственную логику исключительно через текстовые инструкции?

1.2 Обзор исследований

Д. С. Пащенко подтверждает, что внедрение ИИ-инструментов становится ключевым фактором конкурентоспособности ИТ-компаний, позволяя существенно ускорять процессы разработки и автоматизировать рутинные задачи [1]. Фундаментальную возможность создания сложных интерактивных приложений исключительно на базе нативных веб-технологий без внешних зависимостей обосновывают С. В. Рогатин и В. А. Антошкин [2]. Применимость генеративных моделей в сложных предметных областях и креативных индустриях подтверждает Д. Р. Мухаметов, исследующий влияние искусственного интеллекта на формирование политических образов [3]. Однако автоматическая генерация кода требует строгого контроля, что И. В. Елманов доказывает, предлагая методологию Spec Driven Development для надежной валидации программ в парадигме вайб-кодинга [4]. Практическую способность больших языковых моделей эффективно работать со сложной структурой веб-приложений демонстрирует А. М. Титеев, оптимизируя с их помощью создание автотестов и анализ HTML-разметки [5].

1.3 Цель исследования

Цель работы – оценить способность большой языковой модели (на примере Qwen Studio) спроектировать двунаправленный инструмент для работы с QR-кодами в рамках парадигмы Zero-Dependency, а также

проанализировать пределы автоматической отладки пространственных алгоритмов методами промпт-инжиниринга.

2 Материалы и методы

Методология исследования базируется на гибридном подходе, объединяющем структурированный промпт-инжиниринг и отказ от традиционных методов веб-разработки в пользу процедурной генерации. В качестве жесткого архитектурного ограничения выступает применение исключительно нативных HTML, CSS, JavaScript и Canvas API. Всё приложение должно размещаться в едином файле index.html.

В качестве платформы для написания кода применяется среда Qwen Studio. Ключевой особенностью методологии является итеративный подход: первоначальная генерация рассматривается как прототип, который последовательно дорабатывается через серию уточняющих промптов.

Бинарный критерий верификации: сгенерированный код должен считываться стандартными сканерами смартфона, а декодер обязан безошибочно распознавать изображения, созданные генератором.

3 Результаты

Первый этап эксперимента – создание математического ядра приложения. Промпт сфокусирован на строгой реализации стандарта ISO/IEC 18004, чтобы исключить любые упрощения или использование сторонних решений.

Промпт для генерации: «Действуй как Senior Frontend разработчик и специалист по помехоустойчивому кодированию. Твоя задача – написать полный, готовый к запуску код веб-приложения «QR-генератор». Приложение должно быть реализовано в одном файле и работать локально в браузере. Никаких сторонних библиотек – только нативные HTML, CSS и JavaScript. Все вычисления должны выполняться процедурно.

Архитектурные требования:

– Реализуй арифметику в GF(256) с полиномом 0x11D.

– Реализуй коды Рида-Соломона для коррекции ошибок.

– Реализуй BCH-кодирование для формат-информации (полином 0x537) и версии (0x1F25).

– Поддержи версии от 1 до 10 и 4 уровня коррекции (L, M, Q, H).

– Реализуй зигзагообразное размещение данных и выбор оптимальной маски по 4 правилам штрафов.

Требования к интерфейсу:

– Тёмная тема. Слева панель управления: поле ввода текста, выбор уровня коррекции, кнопки скачивания в PNG и SVG. Справа предпросмотр QR-кода.

Выведи полный код в одном блоке, чтобы я мог сохранить его как index.html и сразу запустить».

Результат генерации: https://disk.yandex.ru/d/TFVwyPlPUqGBKA

Нейросеть сгенерировала полностью рабочий код с первой попытки. Модель корректно реализовала вычисление синдромов Рида-Соломона и геометрию размещения поисковых узоров. Важно отметить, что ИИ успешно справился с алгоритмом выбора маски, перебирая 8 вариантов и вычисляя штрафы по четырем правилам спецификации для минимизации количества артефактов. Сгенерированное приложение получило единый интерфейс. Основная область занята предпросмотром сгенерированного QR-кода (рис. 1).

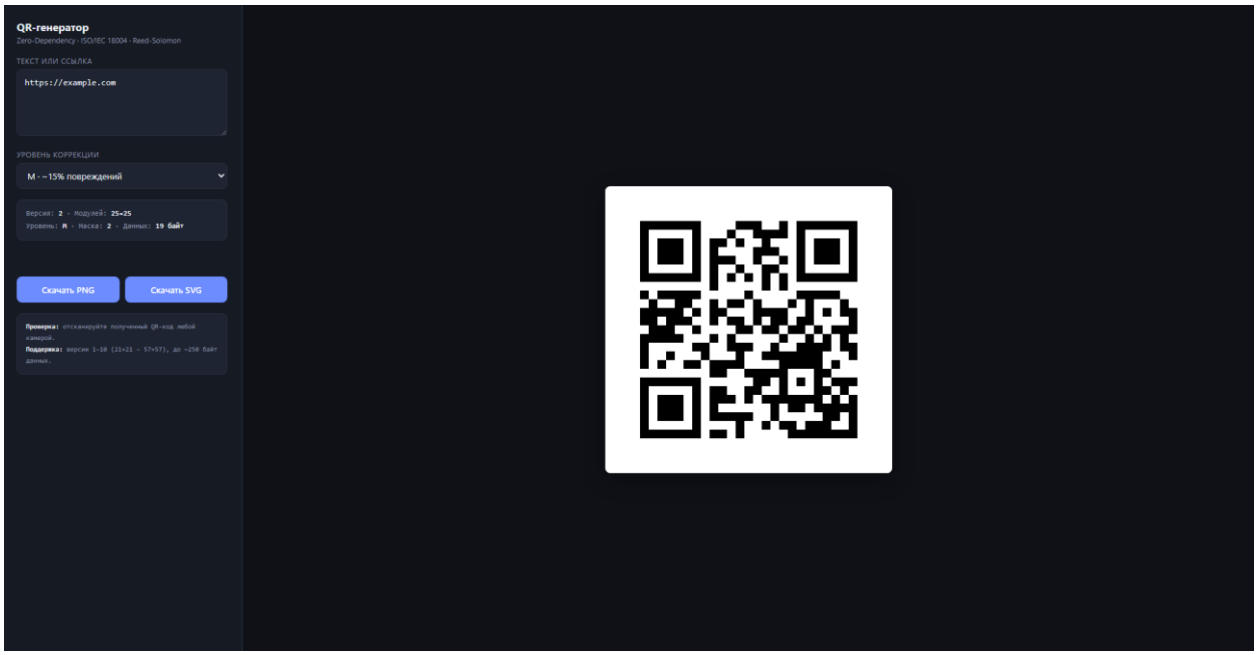


Рисунок 1. Интерфейс приложения

Левая боковая панель приложения содержит поле ввода текста или ссылки, выпадающий список для выбора уровня коррекции (L, M, Q, H) и блок служебной информации. В этом блоке динамически отображаются выбранная версия QR-кода, его размер в модулях (например, 21×21), количество байт данных и примененная маска (рис. 2).

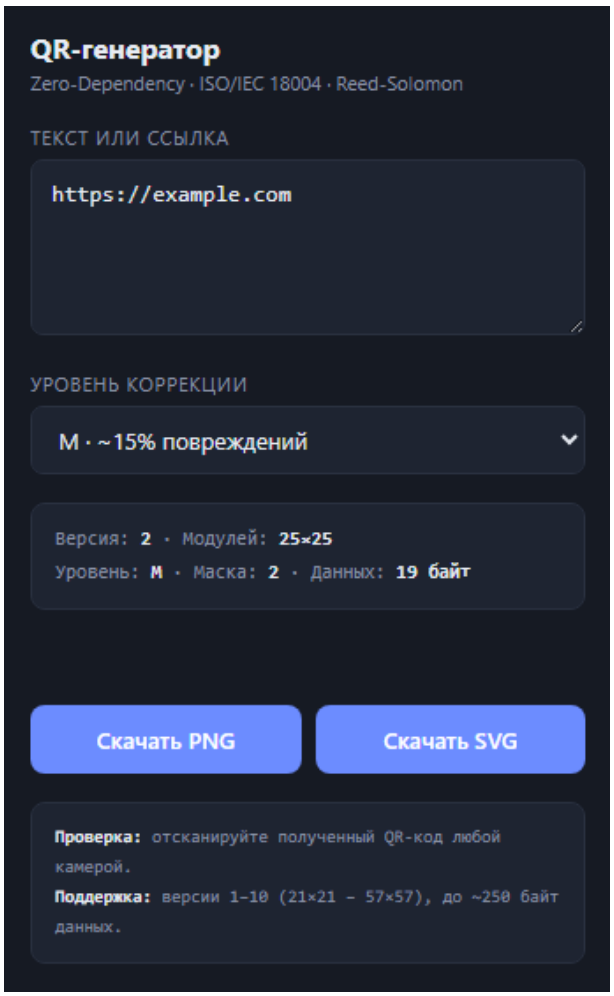


Рисунок 2. Боковая панель

Генератор успешно создает QR-коды, которые безошибочно распознаются сторонними сканерами (рис. 3).

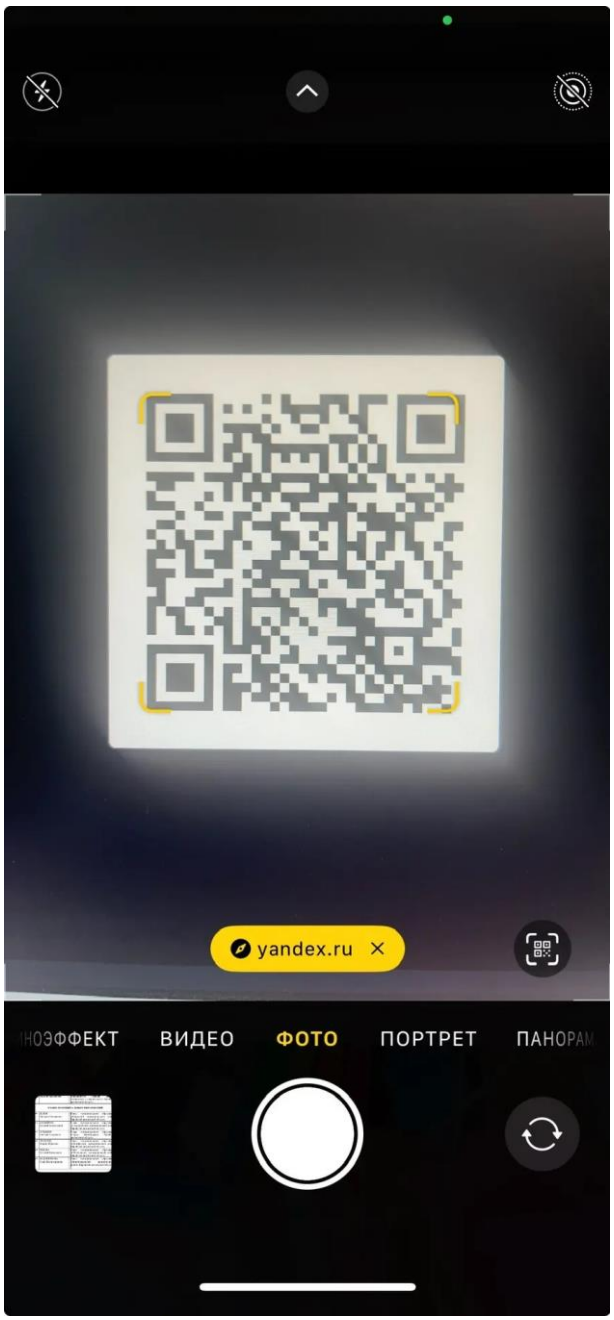


Рисунок 3. Считывание QR-кода камерой смартфона

Второй этап предполагает решение обратной задачи – чтение QR-кода с изображения. Это потребовало реализации алгоритмов компьютерного зрения и обратного де-чередования данных, а также добавления полезной пользовательской функции.

Промпт для доработки: «Добавь следующее:

– Режим декодера QR-кодов из загруженных изображений.

– Бинаризацию изображения методом Отсу для перевода в черно-белый вид.

– Поиск координат QR-кода на изображении, вычисление размера модуля и версии.

– Функцию readFormat для чтения 15-битной формат-информации из двух копий с BCH-коррекцией.

– Обратное чтение данных зигзагом с размаскированием, дечередование блоков и проверку синдромов Рида-Соломона.

– Если синдромы не равны нулю, выводить понятное сообщение об ошибке о повреждении.

– Возможность превращать распознанные QR-коды (если это ссылки) в кликабельные URL.

Требования к интерфейсу: переключатель режимов «Генератор» и «Декодер» вверху. В режиме «Декодер» – drop-зона для загрузки картинки и панель результата».

Результат генерации: https://disk.yandex.ru/d/tHEZAdPiQsscow

Нейросеть успешно реализовала модуль бинаризации, алгоритм поиска ограничивающей рамки и функцию обратного чтения битов. Появилась возможность распознавать ссылки и выводить их в виде кликабельных кнопок. Однако при интеграции сгенерированного кода возникла проблема: декодер успешно находил сетку, но при проверке синдромов Рида-Соломона сообщал о повреждении данных (рис. 4).

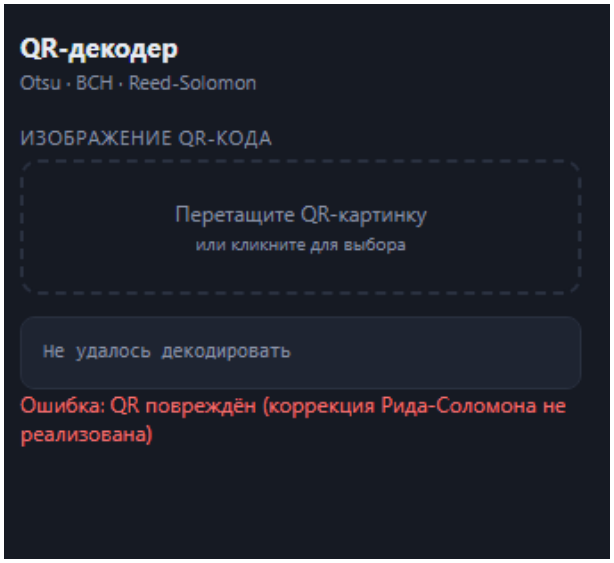


Рисунок 4. Ошибка считывания QR-кода

На этапе тестирования декодера приложение выдавало ошибку «QR повреждён». Выдвинуто предположение, что проблема кроется в функции чтения формат-информации (readFormat), которая отвечает за определение уровня коррекции и маски. Если координаты чтения не совпадают с координатами записи в генераторе, маска определяется неверно, и данные не расшифровываются.

Промпт для отладки: «Приложение выдает ошибку "QR повреждён". Проанализируй симметричность функций placeFormat (генератор) и readFormat (декодер).

Вероятно, при чтении формат-информации перепутаны строки и столбцы (транспонирование координат), из-за чего маска определяется неверно. Проверь соответствие индексов матрицы при записи и чтении битов.

Напиши функцию readFormat так, чтобы координаты чтения битов в точности совпадали с координатами записи из функции placeFormat».

Результат генерации: https://disk.yandex.ru/d/CwiERM7C96hg9A

Нейросеть предложила новый код, однако ошибка сохранилась. Модель изменила координаты, но снова допустила транспонирование.

Промпт для повторной отладки: «Ошибка сохраняется. Ты снова перепутал координаты строки и столбца в функции readFormat. Напиши функцию readFormat так, чтобы координаты чтения битов в точности совпадали с координатами записи из функции placeFormat. Обрати внимание на индексы matrix[row][col]».

Результат генерации: https://disk.yandex.ru/d/m1hccfoDr27UGA

Несмотря на несколько итераций, языковая модель не смогла корректно сопоставить пространственные координаты чтения и записи. Из-за сложности визуализации двумерной матрицы исключительно в текстовом виде нейросеть циклически допускала одни и те же ошибки в индексации. В связи с этим дальнейшая отладка декодера признана нецелесообразной в рамках данного эксперимента.

4 Вывод

Проведенное исследование демонстрирует, что большие языковые модели способны успешно решать задачи, связанные с жестким математическим моделированием и помехоустойчивым кодированием. Эксперимент по созданию двунаправленного QR-инструмента выявил несколько важных закономерностей.

ИИ безупречно справился с линейной алгеброй. Реализация конечных полей Галуа, кодов Рида-Соломона и BCH-полиномов выполнена безошибочно с первого раза. Это подтверждает способность модели понимать абстрактные математические спецификации и транслировать их в код.

Этап создания декодера показал ограничения метода при работе с пространственной логикой. Двумерные матрицы, транспонирование координат и визуальная геометрия требуют отладки, которую сложно осуществить через текстовые промпты. Нейросеть теряет контекст координат «row» и «co»l при переходе от записи к чтению, что приводит к циклическим ошибкам, не устраняемым вербальными инструкциями.

Бинарный критерий верификации (считывание кода сканером) позволяет мгновенно оценить результат. Генератор полностью прошел этот тест, тогда как декодер продемонстрировал, что парадигма вайб-кодинга имеет пределы. В задачах, требующих сложной пространственной верификации и визуального контроля сеток, роль человека-разработчика остается критически важной для ручного аудита координатной логики.

Библиографический список

1. Пащенко Д. С. Использование ИИ-инструментов в конкурентном развитии ИТ-компаний: прикладное исследование 2026 //Информатика. Экономика.

Управление-Informatics. Economics. Management. 2026. Т. 5. №. 3. С. 2012-2024. URL: https://oajiem.com/index.php/24/article/view/297

2. Рогатин С. В., Антошкин В. А. Разработка компьютерных игр по технологии html5 //Информатика и прикладная математика. 2015. №. 21. С.12-16. URL: https://elibrary.ru/item.asp?id=25030492
3. Мухаметов Д. Р. Политический вайб-маркетинг: возможности искусственного интеллекта для продвижения образа будущего //Известия Тульского государственного университета. Гуманитарные науки. 2025. №. 3. С. 27-33. URL: https://cyberleninka.ru/article/n/politicheskiy-vayb-marketing-vozmozhnosti-iskusstvennogo-intellekta-dlya-prodvizheniya-obraza-buduschego
4. Елманов И. В. Инженерный подход к ии: использование spec driven development для валидации vibe coding //Вестник науки. 2025. Т. 4. №. 12 (93). С. 716-723. URL: https://cyberleninka.ru/article/n/inzhenernyy-podhod-kii-ispolzovanie-spec-driven-development-dlya-validatsii-vibe-coding
5. Титеев А. М., Оптимизация процесса создания автотестов веб-приложений с использованием LLM и структурного анализа HTML //Моделирование и анализ данных. 2025. Т. 15. №4. С. 87–103. URL: https://psyjournals.ru/journals/mda/archive/2025_n4/Titeev