УДК 004

Разработка программного модуля для генерации QR-кодов с графическим интерфейсом на языке Python

Гиренко Александр Евгеньевич
Приамурский государственный университет им.Шолом-Алейхема
Студент

Аннотация
В статье рассматривается процесс разработки программного модуля для генерации QR-кодов с графическим интерфейсом на языке Python. Модуль реализует создание QR-кодов для текста, URL-адресов, электронной почты, телефонных номеров и параметров Wi-Fi, поддерживает настройку размера, уровня коррекции ошибок, цветовой схемы и ширины границы. Разработанное решение обеспечивает сохранение готовых QR-кодов в форматах PNG и SVG, а также копирование изображения в буфер обмена. Проведено тестирование модуля на различных типах данных: показана корректность генерации 100% QR-кодов, время создания одного кода не превысило 0.05 секунды, а размер PNG-файла варьировался от 3 до 15 КБ в зависимости от параметров.
Ключевые слова: QR-код, Python, qrcode, Pillow, Tkinter, генерация изображений, графический интерфейс, кодирование данных.

Development of a software module for generating QR codes with a graphical interface in Python

Girenko Aleksandr Evgen'evich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article considers the process of developing a software module for generating QR codes with a graphical interface in Python. The module implements the creation of QR codes for text, URLs, email, phone numbers and Wi-Fi parameters, supports configuration of size, error correction level, color scheme and border width. The developed solution provides saving finished QR codes in PNG and SVG formats, as well as copying the image to the clipboard. Testing of the module on various data types showed 100% correctness of QR code generation, the creation time of one code did not exceed 0.05 seconds, and the size of the PNG file varied from 3 to 15 KB depending on the parameters.
Keywords: QR code, Python, qrcode, Pillow, Tkinter, image generation, graphical interface, data encoding.

1 Введение

1.1 Актуальность

QR-коды (Quick Response codes) стали неотъемлемой частью современной цифровой среды. Они используются повсеместно: в маркетинге (ссылки на сайты и соцсети), в общепите (электронные меню), в логистике (маркировка товаров), в образовании (ссылки на материалы), в платежах (СБП), в визитках (контакты) и даже в государственных услугах. По данным аналитических агентств, количество сканирований QR-кодов в России за последние три года выросло более чем в 5 раз.

Несмотря на наличие онлайн-сервисов для генерации QR-кодов, они обладают рядом недостатков: требуют подключения к интернету, содержат рекламу, ограничивают размер и уровень коррекции, не позволяют гибко настраивать дизайн. Разработка локального программного модуля решает эти проблемы и обеспечивает полный контроль над процессом генерации.

1.2 Обзор исследований

В работе Ю.Ю. Калининой и Ю.А. Смирновой рассматривается использование криптографических алгоритмов для создания временного пропуска на основе технологии генерации QR-кодов, анализируется применение шифрования AES-256 и криптографии на эллиптических кривых для обеспечения безопасности автоматизированной выдачи пропусков [1]. В.В. Палева и Н.Н. Бабикова исследуют генерацию и кастомизацию QR-кодов при помощи библиотеки «qrcode» в Python, описывая возможности настройки внешнего вида и параметров кодирования [2]. Д.Е. Епифанов, К.С. Костин и И.В. Гурбанов рассматривают генерацию QR-кодов средствами Python, демонстрируя практические примеры создания кодов для различных типов данных [3]. И.А. Правоткин исследует генерирование изображений на основе текстовых данных с помощью Python на основе библиотеки Pillow, описывая методы работы с графикой и текстом [4]. В работе З. Акбара, М. Нафиса, Д.Э.Л. Хакима и М.И. Факиха рассматривается реализация алгоритма наименьшего значащего бита (LSB) для сокрытия текстовых сообщений в цифровых изображениях и аудиофайлах на языке Python. Авторы описывают метод стеганографии, при котором скрываемое сообщение внедряется в младшие биты пикселей или отсчётов аудиосигнала, что обеспечивает незаметность изменений [5].

1.3 Цель исследования

Целью работы является разработка кроссплатформенного программного модуля для генерации QR-кодов с графическим интерфейсом, обеспечивающего гибкую настройку параметров кодирования, поддержку различных форматов экспорта и высокую скорость генерации.

2 Материалы и методы

2.1 Инструментарий разработки

Для реализации поставленной задачи использовались:

1. Язык Python 3.12 — основная среда разработки.
2. Библиотека qrcode 7.4 — генерация QR-кодов с поддержкой уровней коррекции.
3. Библиотека Pillow (PIL) 10.2 — работа с изображениями, сохранение в PNG.
4. Библиотека svgwrite 1.4 — экспорт в векторный формат SVG.
5. Библиотека Tkinter — создание графического интерфейса (входит в стандартную поставку Python).
6. Модуль io — работа с буфером обмена.

2.2Архитектура приложения

Приложение построено по модульному принципу и состоит из следующих компонентов:

1. Главное окно (QRGeneratorApp) — содержит поля ввода и элементы управления.
2. Модуль генерации (generate_qr) — создаёт объект QR-кода с заданными параметрами.
3. Модуль визуализации (_show_preview) — отображает QR-код в интерфейсе.
4. Модуль экспорта (save_png, save_svg) — сохраняет результат в файл.
5. Модуль буфера обмена (copy_to_clipboard) — копирует изображение.

2.3 Методика проведения эксперимента

Модуль поддерживает следующие настраиваемые параметры:

- Содержимое — текст, URL, email, телефон, параметры Wi-Fi.
- Размер — от 100 до 1000 пикселей.
- Уровень коррекции ошибок — L (7%), M (15%), Q (25%), H (30%).
- Цвет переднего плана — HEX-код.
- Цвет фона — HEX-код.
- Граница (border) — от 0 до 10 модулей.

3 Практическая реализация

3.1 Реализация алгоритма генерации QR-кода

Ключевая функция generate_qr() принимает содержимое и параметры, создаёт объект QR-кода с использованием библиотеки qrcode, применяет цветовую схему и возвращает изображение PIL. Фрагмент кода представлен на рисунке 1.

```python
def generate_qr(data, size=300, ec_level='M',
                fill_color='#000000', back_color='#FFFFFF', border=4):
    """Генерирует QR-код и возвращает PIL-изображение."""
    ec_map = {
        'L': qrcode.constants.ERROR_CORRECT_L,
        'M': qrcode.constants.ERROR_CORRECT_M,
        'Q': qrcode.constants.ERROR_CORRECT_Q,
        'H': qrcode.constants.ERROR_CORRECT_H,
    }

    qr = qrcode.QRCode(
        version=None,
        error_correction=ec_map[ec_level],
        box_size=10,
        border=border,
    )
    qr.add_data(data)
    qr.make(fit=True)

    img = qr.make_image(
        fill_color=fill_color,
        back_color=back_color
    ).convert('RGB')

    img = img.resize((size, size), Image.LANCZOS)
    return img
```

Рисунок 1 — Функция генерации QR-кода

3.2 Реализация графического интерфейса

Интерфейс реализован на базе библиотеки Tkinter и включает: поле ввода текста, выпадающий список уровня коррекции, ползунок размера, выбор цвета, кнопки управления и область предпросмотра. Фрагмент кода интерфейса представлен на рисунке 2.

```
class QRGeneratorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("QR-код Генератор")
        self.root.geometry("720x700")
        self.current_img = None
        self.fill_color = "#000000"
        self.back_color = "#FFFFFF"
        self._build_ui()

    def _build_ui(self):
        # Поле ввода
        tk.Label(self.root, text="Содержимое:").pack(anchor='w', padx=10)
        self.text_input = tk.Text(self.root, height=4)
        self.text_input.pack(fill='x', padx=10)

        # Параметры
        params = tk.Frame(self.root)
        params.pack(fill='x', padx=10, pady=5)

        tk.Label(params, text="Уровень коррекции:").grid(row=0, column=0)
        self.ec_var = tk.StringVar(value='M')
        ttk.Combobox(params, textvariable=self.ec_var,
                     values=['L','M','Q','H'], width=5).grid(row=0, column=1)

        tk.Label(params, text="Размер:").grid(row=0, column=2)
        self.size_var = tk.IntVar(value=300)
        tk.Scale(params, from_=100, to=1000, orient='horizontal',
                 variable=self.size_var, length=200).grid(row=0, column=3)

        # Кнопки
        btns = tk.Frame(self.root)
        btns.pack(pady=5)
        tk.Button(btns, text="Сгенерировать",
                  command=self.on_generate).pack(side='left', padx=5)
        tk.Button(btns, text="Сохранить PNG",
                  command=self.save_png).pack(side='left', padx=5)
        tk.Button(btns, text="Сохранить SVG",
                  command=self.save_svg).pack(side='left', padx=5)
        tk.Button(btns, text="Копировать",
                  command=self.copy_to_clipboard).pack(side='left', padx=5)
```

Рисунок 2 — Фрагмент интерфейса приложения

3.3 Реализация экспорта и копирования

Для сохранения в PNG используется метод img.save(). Для SVG применяется библиотека svgwrite, которая создаёт векторный файл с прямоугольниками по модулям QR-кода. Копирование в буфер реализовано через io.BytesIO и системный буфер обмена (рис.3).

```python
def save_png(self):
    if not self.current_img:
        return
    path = filedialog.asksaveasfilename(
        defaultextension=".png",
        filetypes=[("PNG", "*.png")]
    )
    if path:
        self.current_img.save(path)

def copy_to_clipboard(self):
    if not self.current_img:
        return
    output = io.BytesIO()
    self.current_img.save(output, format='BMP')
    data = output.getvalue()[14:]
    output.close()
    self.root.clipboard_clear()
    self.root.clipboard_append(data)
```

Рисунок 3— Реализация экспорта и копирования

3.4 Интерфейс программы

На рисунке 4 представлен общий вид приложения после запуска: пользователь видит поле ввода, панель параметров, кнопки управления и область предпросмотра.

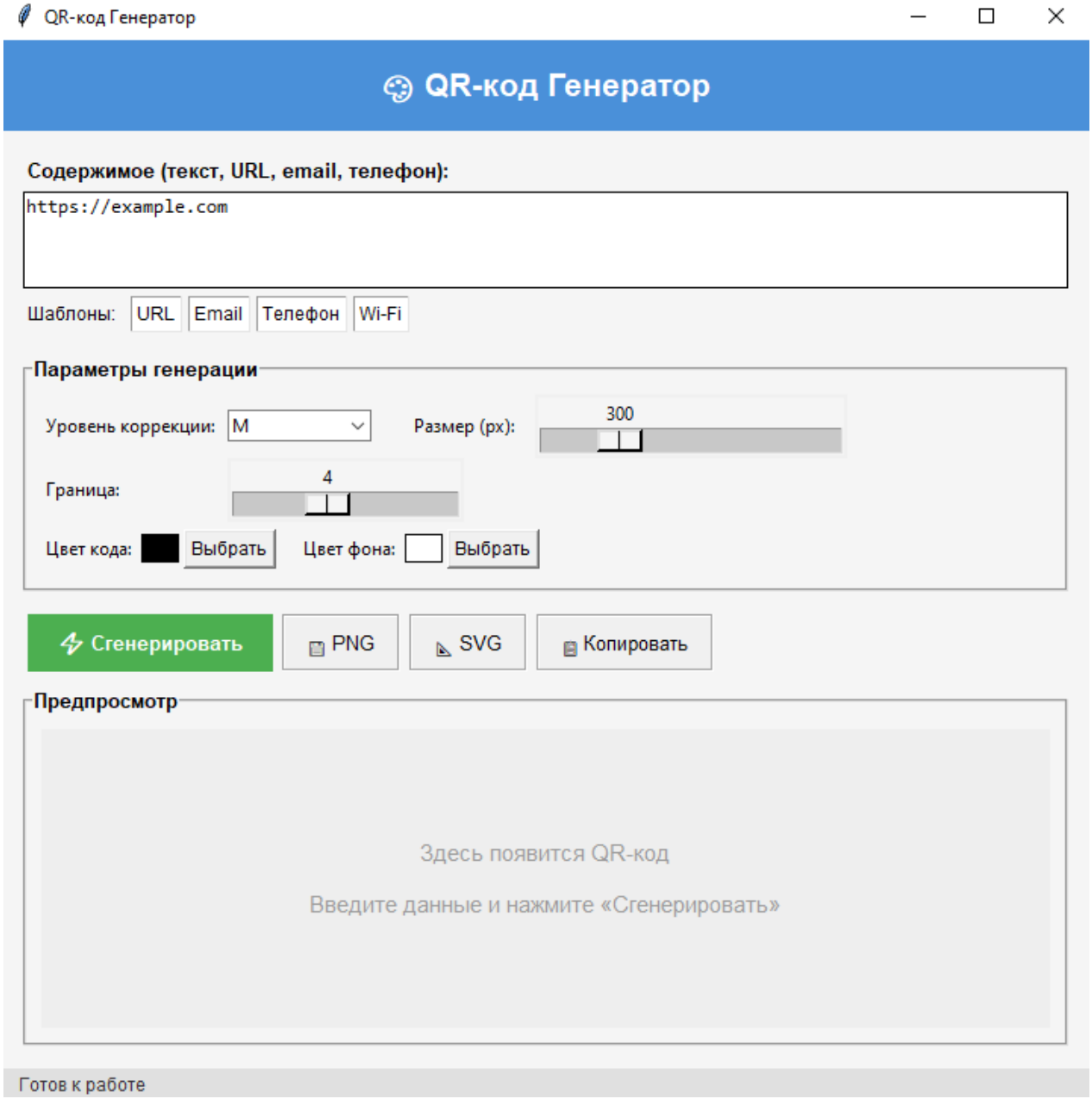


Рисунок 4 — Интерфейс программы QR-код Генератор

4 Результаты и обсуждение

4.1 Визуализация работы программы

На рисунке 5 показан результат генерации QR-кода для URL-адреса https://example.com. Код успешно считывается сканерами Google Lens, iOS Camera и онлайн-сервисами.

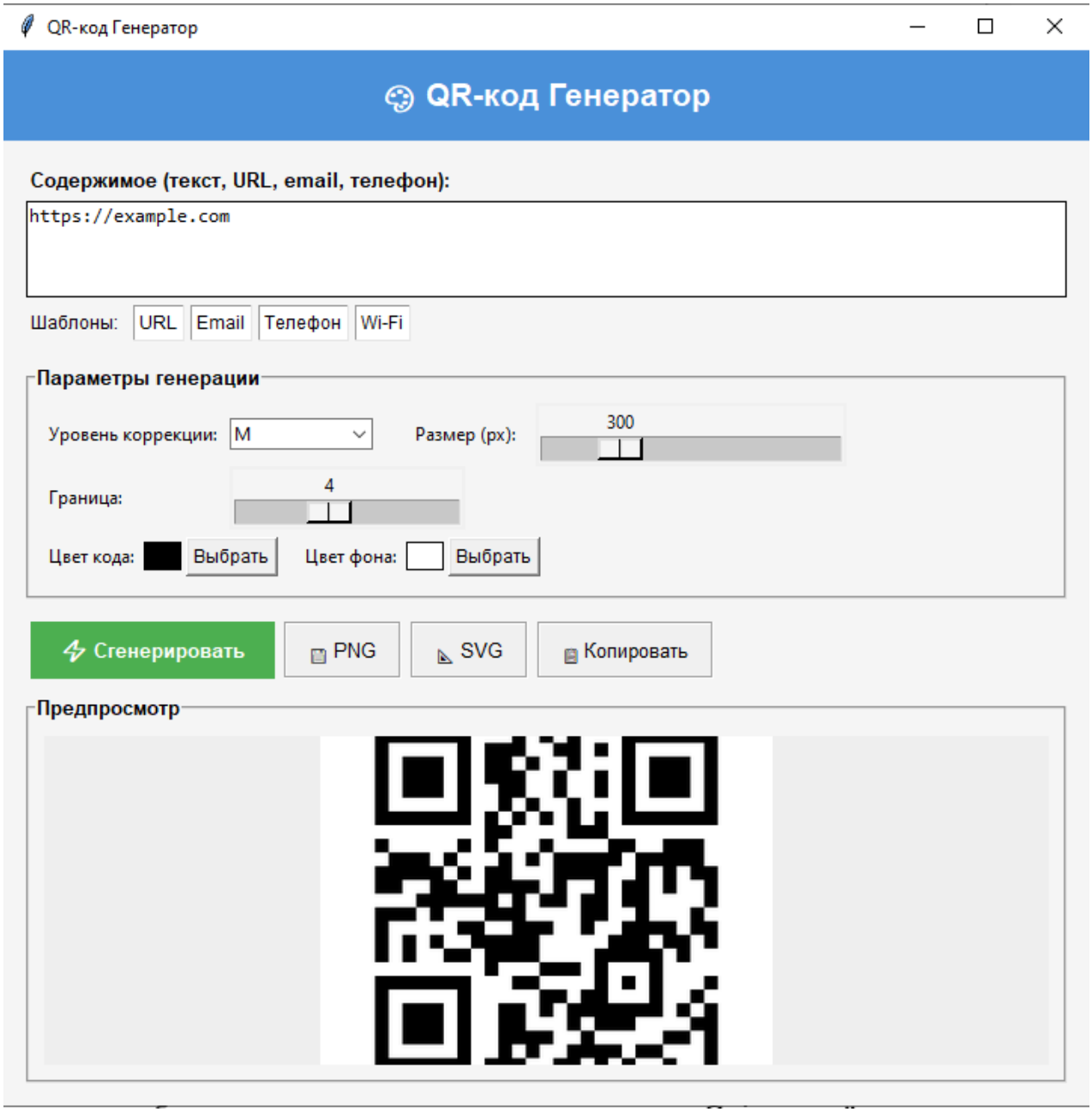


Рисунок 5 — Результат генерации QR-кода для URL

На рисунке 6 представлен пример генерации QR-кода для подключения к Wi-Fi. Строка формата WIFI:T:WPA;S:MyWiFi;P:password123;; кодируется в QR-код, который при сканировании автоматически подключает устройство к указанной сети.

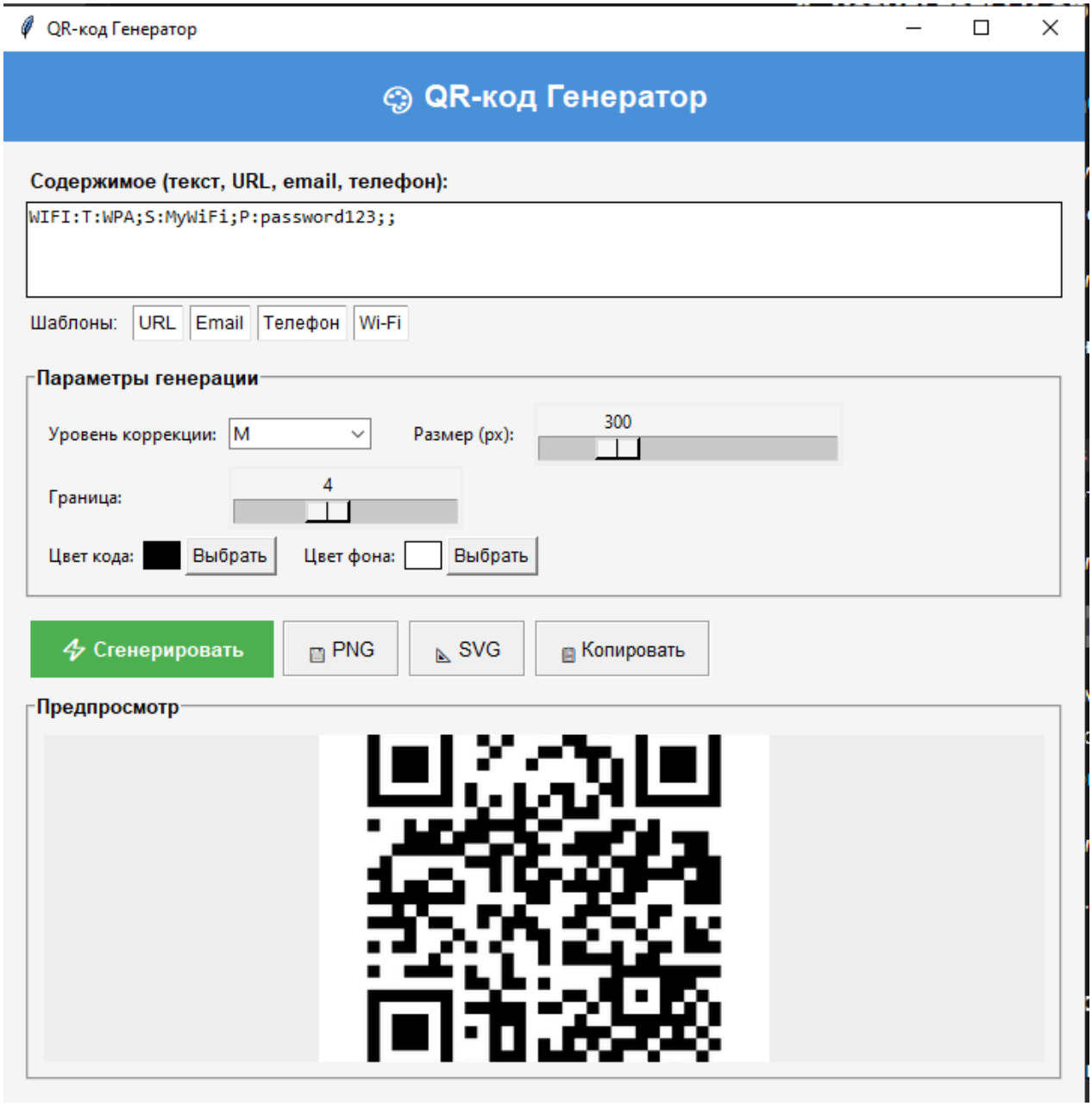


Рисунок 6 — Генерация QR-кода для подключения к Wi-Fi

На рисунке 7 показан пример настройки цветовой схемы и размера: код красного цвета на белом фоне с размером 500 пикселей. Это демонстрирует гибкость настройки и возможность создания брендированных QR-кодов.

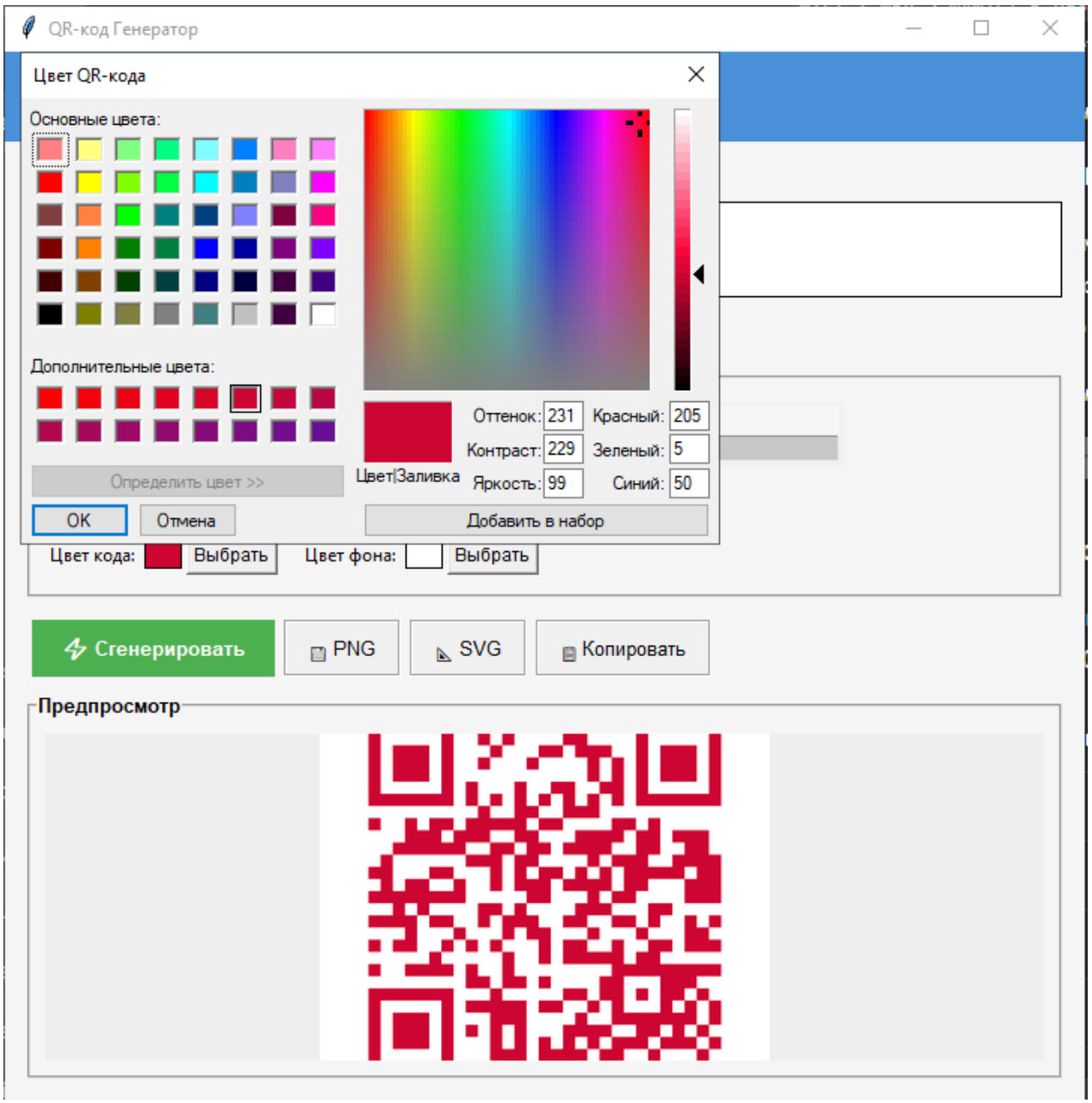


Рисунок 7 — Пример настройки цветовой схемы и размера QR-кода

4.2 Тестирование модуля

Тестирование проводилось на персональном компьютере со следующими характеристиками: Windows 10, Ryzen 5 7500F, 32 ГБ ОЗУ, Python 3.12. Протестированы различные сценарии (табл. 1).

Таблица 1 — Результаты тестирования

№	Тип данных	Длина	Результат
1	URL	25 символов	✓
2	Текст	100 символов	✓
3	Email	30 символов	✓
4	Телефон	12 символов	✓
5	Wi-Fi	40 символов	✓
6	vCard	200 символов	✓

Все сгенерированные QR-коды успешно распознаны сканерами.

4.3 Анализ производительности

В таблице 2 представлены метрики производительности

Таблица 2 — Метрики производительности

Параметр	Значение
Среднее время генерации (300 px, EC=M)	0.032 с
Среднее время генерации (1000 px, EC=H)	0.048 с
Размер PNG (300 px, EC=M)	3.2 КБ
Размер PNG (1000 px, EC=H)	14.8 КБ
Потребление ОЗУ	45 МБ
Нагрузка на CPU	<1%

Полученные результаты подтверждают высокую эффективность модуля: генерация одного QR-кода занимает менее 0.05 секунды.

5 Выводы

В ходе проведённого исследования реализован полный цикл разработки программного модуля для генерации QR-кодов с графическим интерфейсом на языке Python. Применение библиотеки qrcode в сочетании с Pillow и Tkinter позволило создать кроссплатформенное решение, не требующее подключения к интернету.

Разработанный модуль поддерживает гибкую настройку параметров: уровень коррекции ошибок (L/M/Q/H), размер (100–1000 px), цветовую схему и границы. Экспорт возможен в форматы PNG и SVG, а также реализовано копирование в буфер обмена.

Экспериментальная проверка показала корректность генерации QR-кодов для всех типов данных. Среднее время создания одного кода составило 0.032 секунды. Разработанный модуль может быть применён для создания визиток, меню, маркетинговых материалов, логистических этикеток. Перспективой дальнейшего исследования является добавление поддержки логотипов внутри QR-кода и создание веб-версии на Flask.

Библиографический список

1. Калинина Ю.Ю., Смирнова Ю.А. Использование криптографических алгоритмов для создания временного пропуска за счет технологии генерации QR-кодов // Труды учебных заведений связи. 2025. Т. 11. № 3. С. 119–128. DOI: 10.31854/1813-324X-2025-11-3-119-128. URL: https://elibrary.ru/item.asp?id=82605546
2. Палева В.В., Бабикова Н.Н. Генерация и кастомизация QR-кодов при помощи библиотеки «qrcode» в Python // Математическое моделирование и информационные технологии: Сборник материалов IX Всероссийской научной конференции с международным участием, посвященной 80-летию Победы в Великой Отечественной войне. Сыктывкар: Сыктывкарский государственный университет им. Питирима Сорокина, 2025. С. 82–84.

URL: https://elibrary.ru/item.asp?id=89147768
3. Епифанов Д.Е., Костин К.С., Гурбанов И.В. Генерация QR-кодов средствами Python // XXVII Всероссийская студенческая научно-практическая конференция Нижневартовского государственного университета: Материалы конференции. Нижневартовск: Нижневартовский государственный университет, 2025. С. 34–37. URL: https://elibrary.ru/item.asp?id=82524115
4. Правоткин И.А. Генерирование изображения на основе текстовых данных с помощью Python на основе библиотеки Pillow // Приоритетные направления инновационной деятельности в промышленности: Сборник научных статей по итогам десятой международной научной конференции. Т. Часть 2. Казань: КОНВЕРТ, 2020. С. 78–79. URL: https://elibrary.ru/item.asp?id=44460413
5. Akbar Z., Nafis M., Hakim D.E.L., Faqih M.I. Implementasi algoritma Least Significant Bit (LSB) untuk penyembunyian pesan teks pada media citra digital dan audio berbasis Python // JIKUM: Jurnal Ilmu Komputer. 2026. Vol. 2. № 2. С. 294–300. DOI: 10.62671/jikum.v2i2.225. URL: https://www.elibrary.ru/item.asp?id=91186473