

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский
университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

Отчет по лабораторной работе №2

Дисциплина: «Проектирование программных комплексов»

Дисциплина: «Инженерия данных»

Выполнил: Седых М.Ю.
Группа: 6232-010402D

Самара 2025

1.1 Введение

Цель работы — реализовать пайплайн обработки видео с автоматическим распознаванием речи, переводом субтитров и их встраиванием в видео. Пайплайн принимает видео через Telegram-бота (файл или ссылка), извлекает аудио, генерирует английские субтитры с помощью Whisper, переводит их на русский с помощью LLM и встраивает в исходное видео.

1.2 Описание данных и источников

Входные данные:

Видеофайлы: MP4, AVI, MOV, MKV и другие форматы, поддерживаемые FFmpeg

Минимальная длительность: 1 минута

Язык речи: английский (EN)

Источник: Telegram Bot (файл или URL)

Выходные данные:

Видеофайл MP4 с встроенными русскими субтитрами

Формат субтитров: SRT (SubRip)

1.3 Архитектура и методы

1.3.1 Общая архитектура системы

Система состоит из четырех основных компонентов:

n8n — оркестрация workflow

video_processing — HTTP API для работы с видео (FFmpeg)

auto_subtitle — HTTP API для распознавания речи (Whisper)

vllm_server — сервер LLM для перевода (vLLM)

Все компоненты развернуты в Docker-контейнерах и взаимодействуют через HTTP API.

1.3.2 Технологический стек

Версии используемых инструментов:

n8n: latest (1.123.5)

Python: 3.11-slim

Flask: 3.0.0

Flask-CORS: 4.0.0

OpenAI Whisper: 20231117

PyTorch: >=2.0.0

TorchAudio: >=2.0.0

NumPy: >=1.24.0

FFmpeg: последняя версия из репозитория Debian

vLLM: latest (0.12.0)

Docker Compose: 3.8

1.3.3 Структура репозитория

Репозиторий содержит конфигурацию и код для запуска связки сервисов обработки видео и аудио.

`docker-compose.yml` описывает все сервисы и их взаимодействие. `Dockerfile` используется для сборки образов с поддержкой `n8n` и `Python`-сервисов. Файл `requirements.txt` фиксирует зависимости `Python`. `workflow.json` хранит экспортированный `workflow` из `n8n`.

`auto_subtitle_service.py` реализует `HTTP-API` для распознавания речи на базе `Whisper`. `video_processing_service.py` предоставляет `HTTP-API` для обработки видео через `FFmpeg`. `start_services.sh` запускает оба `Python`-сервиса. Каталог `temp` используется для временных файлов и создаётся автоматически.

1.3.4 Описание пайплайна

Пайплайн в `n8n` реализует автоматическую обработку входящих сообщений из `Telegram`. По расписанию выполняется опрос `Telegram API`, после чего проверяется наличие новых сообщений и сохраняются их метаданные. Контент сообщения анализируется, определяется его тип, и выполняется ветвление логики.

Для видео или ссылки видеофайл загружается, преобразуется в бинарный формат и передаётся в сервис обработки видео для извлечения аудио. Аудио направляется в сервис распознавания речи, где генерируются английские субтитры в формате `SRT`. Полученный текст субтитров передаётся

в LLM-сервис, где выполняется перевод с использованием OpenAI-совместимого API и выбранной языковой модели.

Переведённые субтитры преобразуются обратно в формат SRT, после чего встраиваются в исходное видео с помощью FFmpeg. Готовое видео отправляется пользователю в Telegram.

1.3.5 Реализация сервисов

`auto_subtitle_service.py` реализует HTTP API на Flask для распознавания речи с использованием Whisper. Сервис предоставляет эндпоинты для проверки работоспособности и для распознавания речи с возвратом результата в формате JSON или SRT. Модель Whisper загружается при первом запросе и кэшируется, используется английский язык распознавания, временные метки формируются в формате HH:MM:SS,mmm. Поддерживается выбор размера модели и устройства выполнения.

`video_processing_service.py` представляет собой HTTP API на Flask для обработки видео с помощью FFmpeg. Сервис выполняет извлечение аудио из видеофайлов с приведением к формату WAV 16 kHz mono PCM и встраивание SRT-субтитров в видео с кодированием H.264. Реализованы проверки входных файлов, базовая валидация SRT, контроль прав доступа и автоматическая очистка временных файлов.

`start_services.sh` — Bash-скрипт, запускающий оба Python-сервиса в одном контейнере на разных портах и обеспечивающий корректное завершение процессов при получении сигналов SIGTERM и SIGINT.

1.3.6 Конфигурация Docker

`docker-compose.yml` описывает запуск `n8n`, сервисов обработки видео и LLM. `n8n` работает на порту 5678 и зависит от `backend`-сервисов. `video_processing` публикует порты для Whisper и FFmpeg-API. `vllm_server` работает на GPU и предоставляет LLM-API. Используются `volume` для данных `n8n`, кэша моделей и временных файлов, сервисы объединены в `bridge`-сеть.

Dockerfile реализует multi-stage сборку: на первом этапе собираются Python-сервисы с FFmpeg и Whisper, на втором — образ n8n с установленным FFmpeg и docker-cli.

1.3.7 Обработка ошибок

В workflow используются проверки наличия данных, try-catch в пользовательском коде и резервная модель при сбоях LLM. В сервисах выполняется валидация входных файлов, проверка SRT и очистка временных данных с возвратом структурированных ошибок. На уровне Docker применяются healthcheck, управляемые зависимости сервисов и автоматический перезапуск контейнеров.

1.4 Эксперименты и результаты

Система запускается через docker-compose с настройкой переменных для Telegram, n8n, Whisper и LLM. Пайплайн обрабатывает видео и ссылки, генерирует английские субтитры, переводит на русский и встраивает их в видео, временные файлы удаляются автоматически.

Время обработки видео 1–5 минут: аудио 5–15 с, субтитры 30–120 с, перевод 10–30 с, встраивание 20–60 с. Используется 2–4 GB RAM для Whisper, 8–12 GB для vLLM, GPU требуется для vLLM, дисковое место ~1–2 GB.

1.5 Обсуждение

Система модульная, гибкая, с валидацией на всех этапах и воспроизводимой через Docker. Ограничения: требуется GPU, долгий первый запуск моделей. Возможные улучшения: мультиязычное распознавание, кэширование, параллельная обработка, мониторинг, поддержка других форматов субтитров. Проверки качества включают SRT, LLM-ответы, размер файлов, права доступа и структуру бинарных данных.

1.6 Выводы

Реализован стабильный Docker-пайплайн для обработки видео: распознавание речи, перевод и встраивание субтитров, с интеграцией n8n, FFmpeg, Whisper и vLLM, поддержкой ошибок и видео >1 минуты, полный цикл от Telegram до результата.

1.7 Ссылки на документацию

n8n: <https://docs.n8n.io/>

OpenAI Whisper: <https://github.com/openai/whisper>

vLLM: <https://docs.vllm.ai/>

FFmpeg: <https://ffmpeg.org/documentation.html>

Docker Compose: <https://docs.docker.com/compose/>

Flask: <https://flask.palletsprojects.com/>

Telegram Bot API: <https://core.telegram.org/bots/api>

```
PS R:\Python\ID\lr2> docker compose up -d
time="2025-12-14T22:06:34+04:00" level=warning
time="2025-12-14T22:06:34+04:00" level=warning
avoid potential confusion"
[+] Running 3/3
✓ Container vllm_server      Running
✓ Container video_processing Healthy
✓ Container n8n              Running
```

Рисунок 1 - Статус всех Docker-контейнеров системы

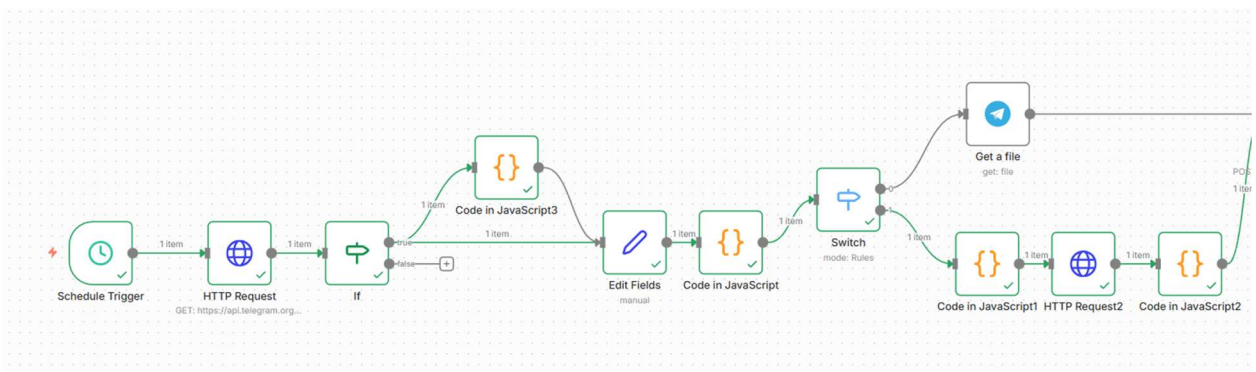


Рисунок 2 - Успешное выполнение пайплайном загрузки видео

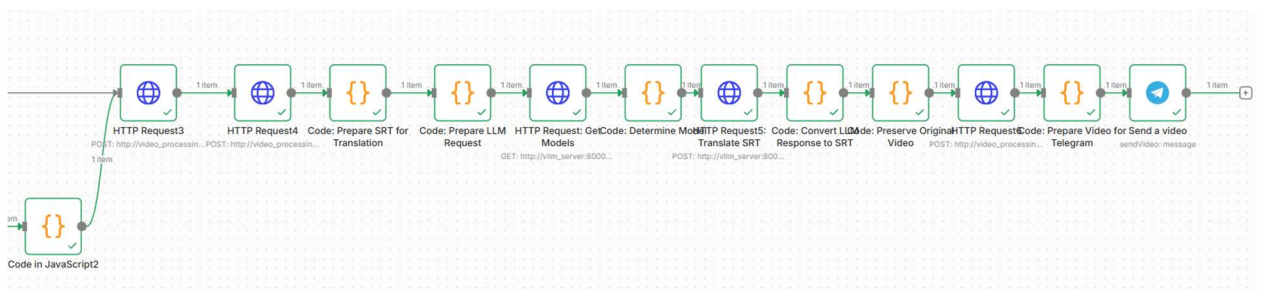


Рисунок 3 - Успешное выполнение пайплайном обработки видео



Рисунок 6 - Получение обработанного видео с русскими субтитрами
(отправка файлом)

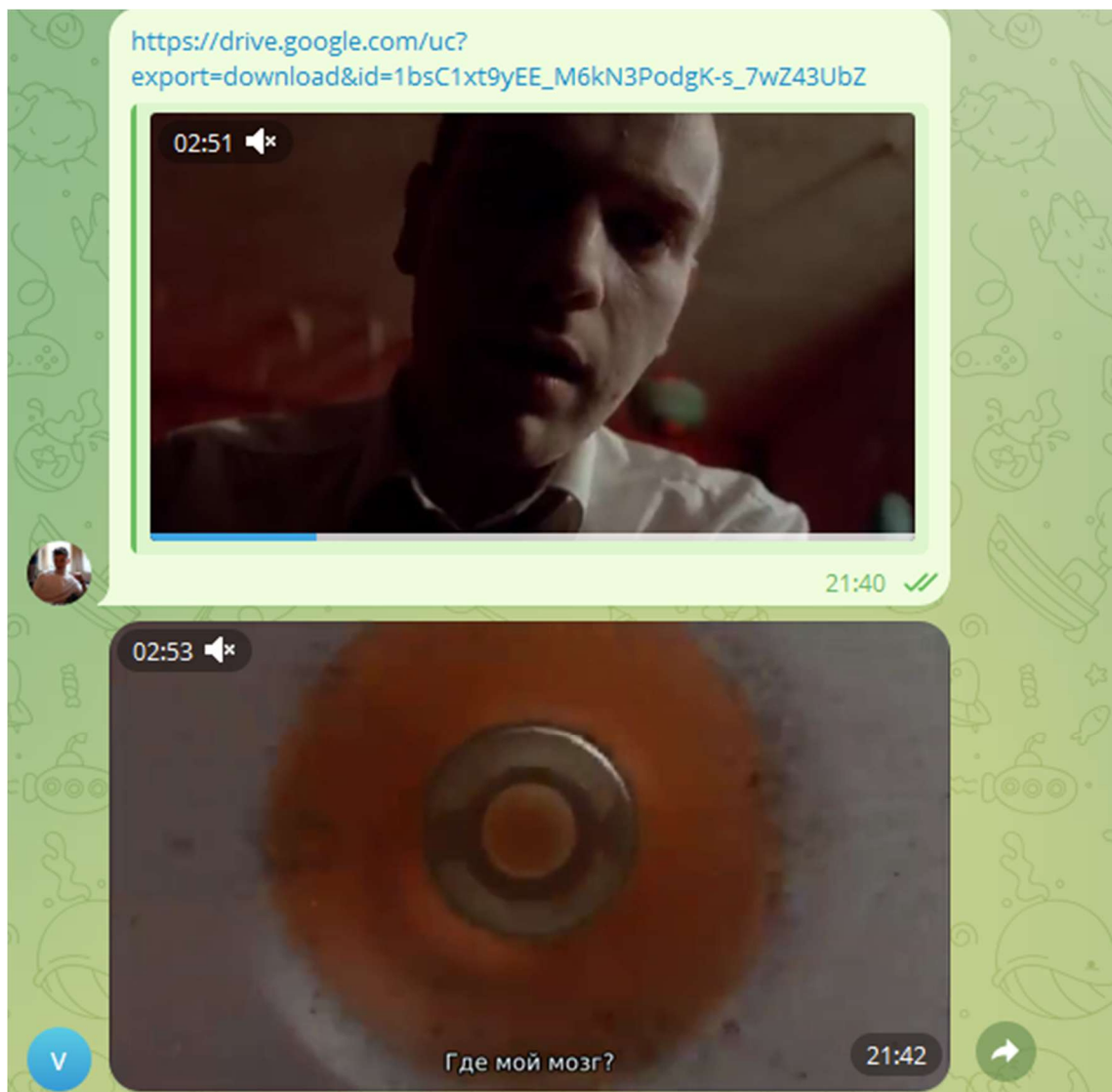


Рисунок 7 - Получение обработанного видео с русскими субтитрами
(отправка ссылкой)