ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ

SputnikSim

Тренажёр анализа и коррекции орбит спутников

Версия	1.0
Дата	20.02.2026
Команда	RUDnik
Репозиторий	github.com/SkillichSE/SputnikSim

1. Общие сведения

1.1 Назначение документа

Настоящий документ описывает архитектуру, функциональные возможности и структуру программного продукта SputnikSim версии 1.0.

Документ предназначен для:

- разработчиков, осуществляющих сопровождение и развитие продукта
- технических специалистов, выполняющих развёртывание и настройку
- операторов, использующих систему в учебных и исследовательских целях

1.2 Назначение программного продукта

SputnikSim — десктопное приложение на Python, предназначенное для обучения работе с орбитальными параметрами спутников. Система обеспечивает:

- загрузку актуальных TLE-данных любого спутника из каталога NORAD/Celestrak
- автоматический анализ орбитальных аномалий с помощью нейросети
- интерактивную коррекцию параметров орбиты (команда delta)
- симуляцию движения спутника с ускоренным временем
- отображение текущего положения спутника на карте мира
- ведение журнала сессии оператора

1.3 Область применения

- Учебные заведения, готовящие специалистов по управлению космическими аппаратами
- Молодёжные космические лаборатории и кружки
- Индивидуальное изучение основ баллистики и управления спутниками

2. Архитектура системы

2.1 Общая архитектура

Архитектурный стиль: монолитное десктопное приложение с модульной структурой. Система состоит из следующих модулей:

Модуль	Назначение
main.py (GUI)	Главное окно, обработка команд оператора, управление состоянием
tle_loader.py	Загрузка и парсинг TLE-данных с Celestrak, работа с dataset
sgp4_core.py	Расчёт орбиты методом SGP4, преобразование координат TEME→ECEF→ lat/lon
parse_tle.py (AI)	Парсинг TLE, классификация типа орбиты
train_model.py (AI)	Обучение нейросети-автоэнкодера
generate.py (AI)	Генерация текстового отчёта по результатам анализа
main.py (AI)	Инференс модели, детектирование аномалий, формирование заключения

2.2 Технологический стек

Компонент	Технология
Язык программирования	Python 3.8+
GUI	PyQt5 + Qt Designer (.ui файл)
Асинхронность	qasync + asyncio
Нейросеть	PyTorch 2.0+
Орбитальная механика	sgp4 (Python binding)
Обработка данных	NumPy, scikit-learn
Источник данных	Celestrak NORAD (HTTP, формат TLE)
Хранение данных	Файловая система (data.json, session.log, AI/model/)

3. Функциональные возможности

3.1 Консольный интерфейс

Взаимодействие с системой осуществляется через встроенную консоль. Ниже приведены доступные команды:

Команда	Синтаксис	Описание
sat	sat <N> | sat <название>	Загрузить спутник по индексу или имени из каталога
delta	delta	Режим ввода коррекций орбитальных параметров
simulate	simulate <часы> <шаг_мин>	Симуляция движения спутника с отображением на карте
live	live	Включить/выключить обновление позиции в реальном времени (каждые 30 с)
help	help	Вывод списка доступных команд

3.2 Загрузка спутника

При выполнении команды sat система:

1. Обращается к локальному кэшу TLE-данных (data.json)
2. Извлекает три строки TLE для выбранного спутника
3. Заполняет таблицу параметров в панели TLE Data
4. Записывает TLE в файл AI/data/test.txt
5. Запускает нейросетевой анализ в фоновом потоке
6. Отображает текущую позицию спутника на карте

3.3 Коррекция орбитальных параметров (delta)

Команда delta запускает пошаговый ввод дельта-значений для следующих параметров:

- Inclination - наклонение орбиты (градусы)
- RAAN - прямое восхождение восходящего узла (градусы)
- Eccentricity - эксцентриситет
- Argument of Perigee - аргумент перигея (градусы)
- Mean Motion - среднее движение (обороты/сутки)

После ввода всех значений система пересчитывает TLE, обновляет таблицу параметров и запускает повторный нейросетевой анализ для оценки изменений.

3.4 Симуляция

Команда simulate выполняет распространение орбиты методом SGP4 на заданный интервал времени с заданным шагом. Результат — перемещение маркера спутника на карте в финальную расчётную точку. Журналируется количество вычисленных точек траектории.

3.5 Отображение на карте

Положение спутника отображается точечным маркером на проекционной карте мира. Координаты вычисляются в системе TEME с последующим преобразованием в ECEF и географические координаты (широта/долгота). Применяется проекция Меркатора.

3.6 Журналирование сессии

Все действия оператора записываются в файл session.log. Формат записи:

```
[HH:MM:SS] > команда
[HH:MM:SS] событие
```

Логирование происходит при каждой сессии программы.

4. Нейросеть

4.1 Архитектура модели

Нейросеть реализована на фреймворке PyTorch и представляет собой автоэнкодер с двумя выходными головами.

Параметр	Значение
Входная размерность	14 признаков (6 орбитальных + 8 признаков типа орбиты)
Латентное пространство	48 нейронов
Выходная голова 1	Декодер - восстановление входного вектора (реконструкция)
Выходная голова 2	Классификатор - бинарное предсказание (норма / аномалия)
Объём кода	~750 строк (train_model.py)
Размер модели на диске	~2 МБ

4.2 Обучающий набор данных

Источник	Количество	Тип
Синтетические TLE (GEO, MEO, LEO, SSO, HEO и др.)	12 000	Нормальные
Реальные TLE - действующие спутники (Celestrak)	872	Нормальные
Реальные TLE - космический мусор (Celestrak)	1 297	Аномальные
Итого	12 872	

4.3 Принцип детектирования аномалий

Автоэнкодер обучен восстанавливать параметры нормальных орбит. При подаче аномального объекта (космический мусор, спутник с отклонениями) ошибка реконструкции превышает пороговое значение, характерное для данного типа орбиты. Порог определяется как 95-й перцентиль ошибки реконструкции на обучающей выборке для каждого класса орбит.

4.4 Метрики точности

Метод	Accuracy	Precision	Recall	F1
Классификатор	91.3%	91.3%	94.0%	92.6%
Детектор аномалий (реконструкция)	81.8%	81.4%	89.0%	85.0%
Комбинированный	91.3%	91.3%	94.0%	92.6%

Валидация выполнена на 516 размеченных объектах: 216 нормальных спутников (активные, МКС, GPS) и 300 объектов космического мусора (Cosmos-1408, Fengyun-1C, Iridium-33, Cosmos-2251). Можно проверить запустив скрипт:

```
python evaluate_accuracy.py
```

4.5 Комбинированное решение

Итоговый вердикт системы формируется взвешенным голосованием: классификатор имеет вес 2, детектор аномалий - вес 1. Аномалия объявляется при суммарном счёте не менее 2. Это исключает ложное срабатывание от детектора реконструкции при нормальном классификаторе.

4.6 Параметры обучения

- Эпох: до 150 (с ранней остановкой при отсутствии улучшения val loss за 15 эпох)
- Оптимизатор: Adam, начальная LR = 1e-3, планировщик ReduceLROnPlateau
- Время обучения: ~10 минут на GPU (CUDA), ~30-60 минут на CPU
- Время инференса на одном спутнике: 1-3 секунды на CPU

5. Структура проекта

Путь	Назначение
main.py	Главное приложение (GUI)
main.ui	Файл интерфейса Qt Designer
tle_loader.py	Загрузка и обработка TLE
sgp4_core.py	Расчёт орбиты SGP4
data.json	Локальный кэш TLE-данных (создаётся при запуске)
session.log	Журнал действий оператора
map.jpg	Карта мира для отображения позиции
AI/main.py	Инференс нейросети
AI/train_model.py	Обучение модели
AI/generate.py	Генерация текстового отчёта
AI/parse_tle.py	Парсинг TLE и классификация орбиты
AI/evaluate_accuracy.py	Оценка точности модели
AI/data/test.txt	TLE текущего спутника для анализа
AI/model/tle_model_best.pth	Веса лучшей модели
AI/model/normalizer.pkl	Параметры нормализатора

6. Развёртывание

6.1 Системные требования

Компонент	Требование
Операционная система	Windows 10+ / Ubuntu 20.04+
Python	3.8 и выше
RAM	4 ГБ и более
CPU	4 ядра и более, поддержка AVX
GPU (опционально)	CUDA-совместимая карта для ускорения обучения
Сеть	Интернет-соединение для загрузки TLE с Celestrak
Место на диске	~2 ГБ

6.2 Зависимости

Библиотека	Назначение
PyQt5	Графический интерфейс
qasync	Интеграция asyncio с Qt event loop
torch	Нейросетевой фреймворк
numpy	Математические операции
scikit-learn	Нормализация данных
sgp4	Расчёт орбиты по методу SGP4
requests	Загрузка TLE с Celestrak

6.3 Установка на Windows

1) Клонировать репозиторий: git clone https://github.com/SkillichSE/SputnikSim или скачать архивом

2) Запустить App_win.bat - скрипт автоматически создаст виртуальное окружение, установит зависимости и запустит программу

3) При первом запуске установка занимает 2–5 минут

6.4 Установка на Ubuntu

1) Клонировать репозиторий или распаковать архив

2) В терминале перейти в папку проекта: cd SputnikSim

3) Выдать права и запустить скрипт: chmod +x App_linux.sh && ./App_linux.sh

4) Скрипт создаст виртуальное окружение, установит зависимости и запустит программу

6.5 Обучение модели (опционально)

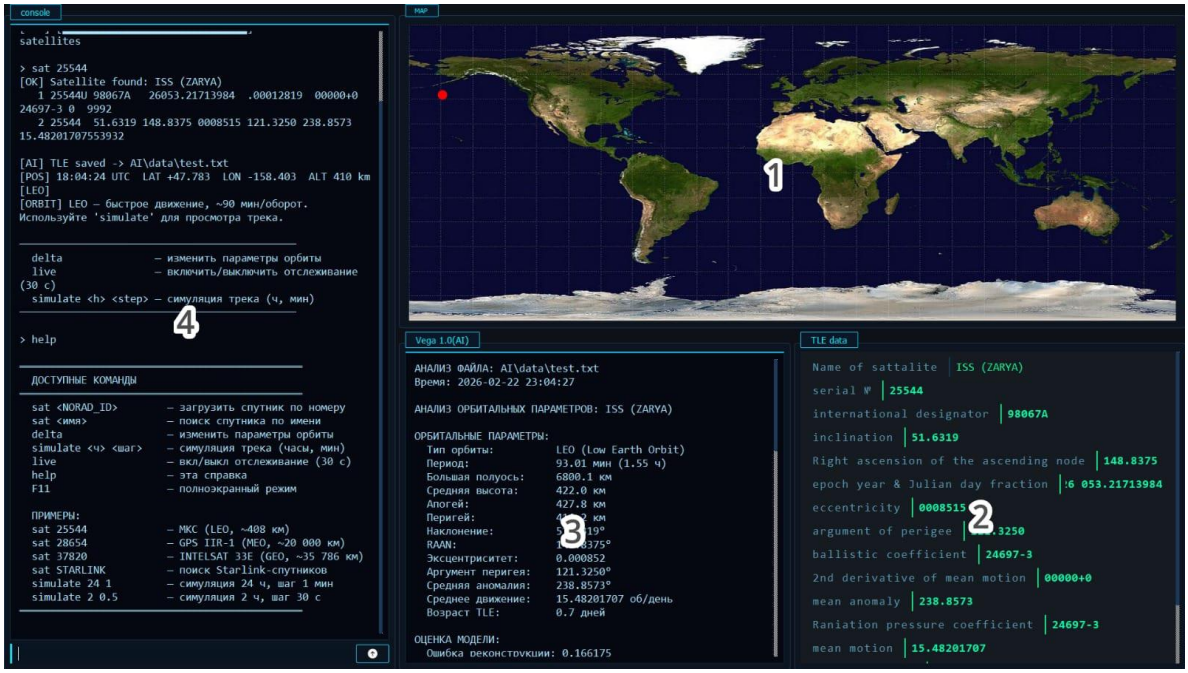
Файл AI/model/tle_model_best.pth для нейросети уже готов к работе, но для наглядности можно заново выполнить обучение:

```
cd AI
python train_model.py
```

Обучение занимает ~10 минут на GPU или ~30-60 минут на CPU. Выполняется однократно.

7. Дизайн

Сделан в QT designer



1) Карта для отслеживания положения спутника
2) TLE данные для вывода информации о спутнике
3) Окно обратной связи с ИИ ассистентом Vega 1.0
4) Консоль для ввода команд и взаимодействия с программой

Интерактивные кнопки

1) Очистка консоли
2) Сброс введенных delta коррекций
3) Обновление данных Celestak
4) Отключение/включение звука в программе

8. Ограничения

- Требуется сетевое соединение для обновления базы TLE при каждом запуске программы
- Система предназначена для образовательных целей и не рассчитана на использование в реальных операциях управления спутниками
- Звуковые уведомления доступны только на Windows (winsound); на Linux звук отсутствует
- Шрифт Consolas может отображаться как системный моноширинный на Ubuntu
- Модель обучена на открытых данных NORAD/Celestrak; точность на секретных или нестандартных объектах не гарантируется

9. О команде

Имя	Роль	Область ответственности
Хлызов Максим	Капитан, физик	Орбитальная физика
Гончар Фёдор	Frontend	GUI, интеграция модулей
Волочай Даниил	Backend и AI	Обучение и внедрение AI
Андин Артём	Маркетинг и документация	Документация, продвижение