ПРИМЕНЕНИЕ LLM-МОДЕЛЕЙ НА ОДНОПЛАТНЫХ КОМПЬЮТЕРАХ ДЛЯ РЕАЛИЗАЦИИ АВТОНОМНОГО ПОЛЕТА БПЛА[1]

Анисимов Р. О.[2], Дворников А. Д.[3], Кулагин К. А.[4],
(*ФГБУН Институт проблем управления им. В.А. Трапезникова РАН, Москва*)
Титова С. А.[5], Петров К. В.[6]
(*Московский Политехнический Университет, Москва*)

Рассматривается применение больших языковых моделей (LLM) для управления беспилотными летательными аппаратами (БПЛА) с помощью естественно-языковых команд. Исследование направлено на решение ключевой проблемы – несоответствия между высокими вычислительными требованиями LLM и ограниченными ресурсами бортовых компьютеров. Основное внимание уделено оптимизации LLM для работы на энергоэффективных одноплатных компьютерах с нейропроцессорами, таких как OrangePi 5B на базе Rockchip RK3588S. В работе представлены результаты тестирования квантованных моделей Qwen2.5-Coder, демонстрирующих сохранение качества генерации кода при скорости обработки до 17,8 токенов/с. Разработан специализированный тест (бенчмарк) для оценки эффективности интеграции LLM в архитектуру автономного управления БПЛА и корректности генерации кода, включающий 125 сценариев. Результаты подтверждают возможность практического применения LLM в автономных системах управления дронами, хотя и выявляют типичные ошибки, связанные с обработкой данных датчиков и системами координат. Работа предлагает перспективное направление для развития интеллектуальных систем управления БПЛА с естественно-языковым интерфейсом (NLP). В рамках исследования были реализованы как научная – создание специализированного теста, так и технологическая новизна – проведение анализа производительности на одноплатных компьютерах, направленные на интеграцию LLM в архитектуру автономного управления БПЛА.

[1] *Исследование поддержано РНФ, грант № 24-29-00671, https://rscf.ru/project/24-29-00671/.*
[2] *Родион Олегович Анисимов, магистр* (rodion_anisimov@mail.ru).
[3] *Алексей Дмитриевич Дворников, магистр* (applskyp@gmail.com).
[4] *Константин Александрович Кулагин, н.с.* (kka8686@mail.ru).
[5] *Софья Алексеевна Титова, студент* (titovas63059@gmail.com).
[6] *Константин Владимирович Петров, студент* (r.92rab@gmail.com).

Ключевые слова: беспилотный летательный аппарат, большая языковая модель, квантование, одноплатный компьютер, автономное управление, естественно-языковой интерфейс.

1. Введение

Современные беспилотные летательные аппараты (БПЛА) находят применение в критически важных областях: от мониторинга инфраструктуры до экстренного реагирования. Однако существующие системы управления требуют либо ручного контроля оператора, либо сложного программирования автономных миссий.

Интеграция больших языковых моделей (англ. Large Language Models – LLM) [3] предлагает революционный подход – преобразование естественно-языковых команд в исполняемый код в реальном времени, что значительно расширяет доступность и гибкость управления БПЛА.

Современные одноплатные компьютеры (англ. Single-Board Computers – SBCs) [1] предлагают значительную вычислительную мощность при компактных размерах и энергоэффективности, что делает их идеальными для применения в БПЛА [15].

Существует несколько успешных проектов, направленных на интеграцию LLM в системы управления БПЛА, например:

1. DroneGPT [13]. В данном проекте применяется квантованная языковая модель GPTQ-4bit, реализованная на аппаратной платформе NVIDIA Jetson Xavier NX, оснащённой 384 CUDA-ядрами и 8 ГБ оперативной памяти. Система демонстрирует способность БПЛА интерпретировать голосовые команды и преобразовывать их в полетные задания. Например, команда «Облети здание и найди открытые окна» преобразуется в траекторию полета.
2. Autonomous Aerial Exploration System [7]. В этом исследовании используется модель TinyLlama, развернутая на одноплатном компьютере Raspberry Pi 5 с нейронным процессором Coral (4 TOPS). Система обеспечивает автономную навигацию в ранее неизвестной среде и обладает функцией генерации объяснений на естественном языке, поясняющих принятые решения.

3. Emergency Response Drone [12]. Проект реализован на базе модели Phi-2, работающей на Jetson Nano с 128 CUDA-ядрами и 4 ГБ оперативной памяти. Дрон автономно оценивает ситуации при стихийных бедствиях, проявляя способность к пониманию визуального контекста и принятию решений на его основе.

Несмотря на достигнутый прогресс, существует ряд ограничений, препятствующих широкому внедрению LLM в системы управления БПЛА. Одним из ключевых факторов является высокое энергопотребление: даже оптимизированные модели требуют значительных вычислительных ресурсов. Так, применение языковой модели TinyLlama на платформе Jetson Xavier NX приводит к сокращению времени полёта дрона до 15–20 мин. по сравнению с 30–35 мин., характерными для традиционных систем [8]. Задержка обработки данных также представляет собой критический параметр в условиях динамического управления. Время обработки данных для квантованных моделей на одноплатных компьютерах составляет от 50 до 300 мс, что недостаточно для реализации оперативной реакции в реальном времени. А надёжность и безопасность эксплуатации осложняются вероятностной природой LLM, способной приводить к неоптимальным или непредсказуемым решениям. Для повышения устойчивости работы модели LLM требуется внедрение дублирующих систем и механизмов контроля, а также ограничений операционной области для предотвращения опасных ситуаций. Законодательные ограничения в ряде юрисдикций накладывают жёсткие требования на использование БПЛА, включая обязательство поддержания визуального контакта с оператором и запреты на эксплуатацию в определённых зонах.

Целью настоящей работы является разработка и экспериментальная верификация архитектуры автономного управления БПЛА, основанной на специально оптимизированных LLM с 2–3 млрд параметров и на энергоэффективных одноплатных компьютерах, оснащённых нейронным процессором.

В рамках исследования были реализованы как научная, так и технологическая новизна, направленные на интеграцию LLM в архитектуру автономного управления БПЛА. К научной новизне относится разработка специализированного теста

(бенчмарк, англ. benchmark) для оценки эффективности интеграции LLM в архитектуру автономного управления БПЛА, а также проведение экспериментальной оценки точности и производительности предложенной архитектуры. Технологическая новизна заключается в проведении анализа производительности на одноплатных компьютерах, таких как Jetson, Rockchip и Raspberry Pi, при выполнении вычислений с LLM, а также в сравнении различных методов оптимизации моделей, включая квантование, дистилляцию знаний и сжатия части параметров предобученной модели (прунинг нейронных сетей).

2. Постановка задачи

Требуется разработать архитектуру автономного управления БПЛА, работающую на основе автоматической генерации программного кода на языке Python и обработки естественного языка (англ. Natural Language Processing – NLP) по текстовому описанию миссии полета. Сгенерированный код должен корректно взаимодействовать с заданным API дрона, обеспечивая безопасное и точное выполнение миссий.

Важным требованием архитектуры автономного управления БПЛА является способность в текущей ситуации принимать решения с учётом контекстуальных факторов окружающей среды, что обеспечит адаптацию к непредвиденным обстоятельствам за счёт корректировки планов полёта при обнаружении препятствий или иных факторов. Кроме того, архитектура должна включать мультимодальный анализ, предусматривающий интеграцию данных с различных сенсорных устройств для комплексной оценки обстановки.

Пример генерации кода по описанию миссии полета. Инструкция: *«Взлет, установить высоту 1,5 м, лететь в точку (–1, 4) по одометрии. Искать ArUco-маркер с ID = 5 в бесконечном цикле. Если найдется, включить зеленый свет, иначе – красный. Обнулить одометрию и лететь в точку (0, 0) в глобальных координатах»*.

Ответ модели Qwen2.5-Coder-1.5B-Instruct (процессор Rockchip RK3588S):

```
from drone_control_api import Drone
import numpy as np
drone = Drone()
drone.connect(ip="trackingcam3.local", port="1233")
drone.takeoff()
drone.setHeight(1.5)
drone.gotoXYodom(-1, 4)

while True:
    arucos = drone.getArucos()
    for aruco in arucos["response"]:
        if aruco["id"] == 5:
            drone.setDiod(0, 255, 0)
            break
        else:
            drone.setDiod(255, 0, 0)
drone.setZeroOdomOpticflow()
drone.gotoXYdrone(0, 0)
```

Согласно данным исследования [18], эффективная архитектура автономного управления БПЛА предполагает разделение функциональных задач между LLM и традиционными алгоритмами. Большие языковые модели отвечают за стратегическое планирование, интерпретацию контекста и принятие сложных решений, тогда как классические алгоритмы реализуют тактическое управление и обеспечивают оперативную реакцию на непредвиденные ситуации. Поэтому необходимо применить гибридный режим функционирования системы, при котором часть вычислительных задач выполняется локально, а часть, при наличии стабильного соединения, передается на более мощные серверные ресурсы для обработки.

3. Решение

3.1. АППАРАТНАЯ ПЛАТФОРМА

Для реализации автономного управления БПЛА с использованием LLM необходим одноплатный компьютер, который

должен обладать высокими вычислительными возможностями, включая встроенный нейронный процессор с производительностью 6 TOPS и поддержку современных методов квантования, таких как INT4/INT8 и W8A8, что обеспечивает выполнение вывода языковых моделей размером до 2 млрд параметров. Важным преимуществом является энергоэффективность платформы, т.е. потребляемая мощность при работе нейронного процессора должна составлять 5–8 Вт с пассивной системой охлаждения, которая минимизирует энергопотребление.

3.2. ОПТИМИЗАЦИЯ МОДЕЛИ LLM

Для работы на одноплатных компьютерах существуют специальные облегченные версии LLM:

1. TinyLlama – с 1,1 млрд параметров, что более чем в 100 раз меньше по сравнению с GPT-4, и которая способна работать на OrangePi.
2. Phi-2 – 2,7 млрд параметров, оптимизирована для встраиваемых систем.
3. MiniGPT – 350 млн параметров, идеальна для ограниченных ресурсов.
4. Llama2-7B-GPTQ – квантованная до 4 бит версия Llama 2, требует около 3,5 ГБ памяти.
5. Qwen2.5-Coder-0.5B-Instruct и Qwen2.5-Coder-1.5B-Instruct – высокая скорость вывода при сохранении качества генерации кода. Оптимизирована для работы на нейронных процессорах с квантованием до 4 бит.

При развертывании моделей LLM на ограниченных по ресурсам устройствах и в условиях необходимости быстрого отклика (токен/с) применяются методы оптимизации, направленные на уменьшение объёма вычислений и требований к памяти при минимальных потерях в качестве.

Рассмотрим основные подходы, используемые нами при проведении исследования:

1. **Квантование.** Предполагает снижение разрядности представления весов и активаций модели (например, с 32-битных чисел с плавающей точкой (FP32) до 8-битных целых (INT8) или даже 4-битных (INT4)). Такой подход позволяет су-

щественно сократить объём памяти и ускорить вычисления за счёт работы с меньшим числом бит на параметр.

Модель MiniLLM, квантованная до 4 бит, демонстрирует сокращение требований к памяти до 8 раз при снижении точности менее чем на 5%, что указывает на эффективность данного метода для практических применений [10].

Для формального описания процесса квантования применяется метод W8A8, при котором веса и активации моделируются 8-битными числами. Квантование весов и активаций можно записать как

$$(1)\quad W_q = round\left(\frac{W}{s_w}\right), \quad s_w = \frac{\max(|W|)}{127},$$

$$(2)\quad A_q = round\left(\frac{A}{s_a}\right), \quad s_a = \frac{\max(|A|)}{127},$$

где W и A – исходные веса и активации соответственно, s_w и s_a – масштабные коэффициенты. Де-квантование (обратное преобразование) выполняется по формулам

$$(3)\quad \tilde{W} = W_q \cdot s_w, \quad \tilde{A} = A_q \cdot s_a.$$

При вычислениях на компьютерах с нейронным процессором (NPU) матричное умножение весов и активаций производится в квантованном виде с последующим масштабированием:

$$(4)\quad NPU_{Op}(W_q, A_q) = MatMul(W_q, A_q) \cdot (s_w \cdot s_a).$$

Для повышения точности квантования в моделях Qwen2.5-Coder применяется групповое квантование (group-wise quantization), при котором веса разбиваются на группы, и для каждой группы рассчитывается свой масштабный коэффициент:

$$(5)\quad s_W^{(g)} = \frac{\max\left(|W_g|\right)}{127}, \quad g = 1,...,G,$$

где W_g – веса g-й группы, G – общее количество групп. Такая локализация масштабирования позволяет снизить ошибку квантования на 15–20% по сравнению с глобальным квантованием, что особенно важно для моделей с размером 0,5–1,5 млрд параметров, где каждый параметр оказывает значительное влияние на качество.

2. **Дистилляция знаний**. Это передача знаний от большой модели-учителя к компактной модели-ученику. Процесс обучения компактной модели происходит с использованием выходов предварительно обученной, более крупной модели-учителя. Такой подход позволяет уменьшить размер и вычислительную сложность модели при минимальных потерях в точности. Например, модель DistilLLM демонстрирует сжатие в 3–4 раза с сохранением до 95% первоначальной производительности [16].

Метод широко используется при переносе крупных моделей на бортовые вычислительные устройства, включая вычислительные модули на БПЛА [9, 11].

3. **Сжатие (прунинг).** Направлен на удаление наименее значимых параметров предобученной модели, что уменьшает её размер и ускоряет вычисления без заметного ухудшения качества. Вес w_i обнуляется, если его модуль меньше заданного порога θ:

$$\hat{w}_i = \begin{cases} 0, \text{ если } |w_i| < \theta, \\ w_i \text{ иначе.} \end{cases} \quad (6)$$

Метод SparseGPT позволяет удалять до 60% параметров с минимальной потерей точности [4].

4. Эксперимент

4.1. АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ ОДНОПЛАТНЫХ КОМПЬЮТЕРОВ И ОЦЕНКА КАЧЕСТВА РАБОТЫ МОДЕЛЕЙ СЕМЕЙСТВА QWEN2.5-CODER-INSTRUCT

В рамках эксперимента проверялась работоспособность системы интерпретации текстовых команд для управления дроном, а также корректность выполнения полётного задания в условиях ограниченного пространства и наличия препятствий.

Входными данными служило текстовое описание задачи, сформулированное на естественном языке:

«Взлететь на 2 м, пролететь 3 м вперёд, развернуться на 180 градусов и вернуться в точку старта».

Кроме того, учитывалась спецификация API дрона, включающая методы управления (взлёт, движение, повороты), си-

стему координат и ограничения по скорости, высоте и углам поворота. Также учитывался контекст окружения – начальное положение дрона, наличие препятствий и параметры используемых датчиков (их тип, точность и дальность).

В решении использовалось квантование W8A8 для семейства моделей Qwen2.5-Coder-Instruct. Основное внимание уделено сравнению производительности и качества моделей на тестах HumanEval [5] и Mostly Basic Python Problems (MBPP) [2], которые оценивают качество генерации кода на языке Python, что напрямую связано с целевой задачей.

HumanEval – это набор из 164 задач, включающих написание функций на Python. Задачи охватывают алгоритмические конструкции, работу со строками и структурами данных. Основная метрика – pass@1, отражающая вероятность того, что сгенерированный код успешно проходит все тесты с первой попытки. Этот тест позволяет оценить способность модели не только синтаксически корректно генерировать код, но и логически правильно решать поставленные задачи.

MBPP содержит 974 задачи базового уровня. Они охватывают более широкий спектр задач, включая работу с файлами, парсинг строк, базовые структуры данных и обработку информации. Также используется метрика pass@1. В отличие от HumanEval, MBPP позволяет дополнительно оценить обобщающую способность модели на типичные пользовательские задачи.

Модели Qwen2.5-Coder построены на трансформерной архитектуре [17] с несколькими ключевыми модификациями для эффективной работы на встраиваемых системах. В их основе лежат следующие компоненты:

Механизм многоголового внимания [17]:

$$(7)\quad Attention(Q,K,V) = softmax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V,$$

где Q, K, V – матрицы запроса, ключа и значения соответственно; d_k – размерность ключа.

Позиционные векторные представления (embedding) с вращением по методу позиционного кодирования (англ. Rotary Positional Embedding – RoPE) [14]:

(8) $\int RoPE(x_m, m) = x_m e^{im\Theta}$,

где x_m – входное представление для позиции m; Θ – параметр вращения.

Активация SwiGLU в полносвязном слое сети прямого распространения (англ. Feed-Forward Network – FFN):

(9) $SwiGLU(x, W, V, b, c) = Switch(xW + b) \otimes (xV + c)$,

где $\otimes$ – поэлементное умножение; Switch – функция активации $Switch(x) = x \cdot \sigma(\beta x)$ с σ – сигмоидой.

Такая архитектура обеспечивает хорошее соотношение между качеством и вычислительной эффективностью, особенно в условиях низкоразрядного квантования и работы на нейронных процессорах [6].

В таблице 1 представлены результаты тестирования на двух аппаратных платформах одноплатных компьютеров – Nvidia RTX4090 и Rockchip RK3588S – с использованием двух языковых моделей Qwen2.5-Coder-1.5B-Instruct (полная модель) и Qwen2.5-Coder-0.5B-Instruct (компактный вариант). Оценка производилась на тестах HumanEval и MBPP по входным данным в виде текстового описания задачи, указанного выше по тексту.

Таблица 1. Результаты тестов для аппаратных платформ и языковых моделей Qwen2.5-Coder

Модель	Платформа	HumanEval (pass@1)	MBPP (pass@1)	Скорость (токен/с)
Qwen2.5-Coder-1.5B-Instruct	Nvidia RTX4090	0,630	0,510	
Qwen2.5-Coder-1.5B-Instruct	Rockchip RK3588S	0,597	0,492	9,60
Qwen2.5-Coder-0.5B-Instruct	Nvidia RTX4090	0,463	0,336	
Qwen2.5-Coder-0.5B-Instruct	Rockchip RK3588S	0,420	0,302	17,84

Как видно из таблицы 1, при квантовании в формате W8A8 и запуске на компьютерах с нейронными процессорами каче-

ство моделей не падает, а скорость обработки (токен/с) значительно увеличивается. Модель Qwen2.5-Coder-1.5B-Instruct демонстрирует более высокие показатели качества на тестах HumanEval (pass@1 = 0,63) и MBPP (pass@1 = 0,51) при выполнении на платформе Nvidia RTX4090 по сравнению с результатами на платформе Rockchip RK3588S (0,597 и 0,492 соответственно), при этом скорость генерации на Rockchip составляет 9,6 токен/с. Модель меньшего размера – Qwen2.5-Coder-0.5B-Instruct – характеризуется снижением качества на обеих платформах, однако обеспечивает существенно более высокую скорость генерации на платформе Rockchip (17,84 токен/с), что свидетельствует о ее пригодности для применения в условиях ограниченных вычислительных ресурсов и необходимости ускоренного вывода результатов.

4.2. РАЗРАБОТКА СПЕЦИАЛИЗИРОВАННОГО ТЕСТА ДЛЯ ОЦЕНКИ ЭФФЕКТИВНОСТИ ИНТЕГРАЦИИ LLM В АРХИТЕКТУРУ АВТОНОМНОГО УПРАВЛЕНИЯ БПЛА

По аналогии с тестами HumanEval и MBPP, использующимися для оценки качества генерируемого языковой моделью кода, мы разработали специализированный тест для оценки способности языковых моделей генерировать корректный код управления беспилотными летательными аппаратами. Тест основан на реальном API для управления дроном и оценивает:

1) корректность подключения и инициализации дрона;
2) правильность работы с системами координат;
3) правильность обработки данных от датчиков;
4) точность выполнения навигационных команд;
5) безопасность и надежность сгенерированного кода.

Оценка эффективности интеграции LLM в архитектуру автономного управления БПЛА производится по метрике pass@1. Для оценки качества генерации кода используется 125 тестовых примеров, равномерно распределенных по 5 категориям, представленным в таблице 2. Общий результат вычисляется как среднее значение pass@1 по всем категориям с весовыми коэффициентами, учитывающими сложность задач.

Таблица 2. Распределение тестовых примеров по категориям

Категория	Описание	Количество примеров
Базовые операции	Тестирование основных команд: подключение, взлёт, посадка, управление высотой	25
Движение по координатам	Проверка работы с глобальной и локальной системами координат (методы gotoXYdrone, gotoXYodom)	25
Движение со скоростью и повороты	Проверка работы с глобальной и локальной системами координат (методы setVelXY, setVelXYYaw, setYaw)	25
Работа с датчиками	Интеграция с лидаром, камерой, ArUco-маркерами, оптическим потоком	25
Дополнительные сценарии	Комплексные задания (поиск маркеров, облёт препятствий, патрулирование)	25

4.2.1. ПРИМЕРЫ ТЕСТОВЫХ ЗАДАНИЙ

Для оценки функциональности разработанного теста сформированы тестовые задания, сгруппированные по ключевым категориям.

Тестовые задания включают сценарии, предусматривающие взлёт, полёт к заданным точкам с координатами (например, (2, 0) и (2, 2)), установку высоты на 1,7 м, зависание на 3 с и последующую посадку.

Движение по координатам проверяет способность управления дроном с использованием различных систем координат. В задачи входит полёт до точек (3, 0) и (3, 3) с применением одометрии и абсолютных координат, а также возврат в исходную точку. Дополнительно рассматривается движение назад на 1 м в абсолютных координатах и влево на 1 м по одометрии.

Движение со скоростью включает выполнение поворотов на заданные углы (например, 0,1 радиан против часовой стрелки), а также перемещение вперёд со скоростью 1 м/с и влево со скоростью 0,25 м/с с последующими остановками.

Работа с датчиками оценивается на основе задач по обнаружению ближайших препятствий с использованием лидаров и остановке на безопасном расстоянии в 1 м от них, а также идентификации ArUco-маркеров с *ID* = 5 с выводом их координат.

Дополнительные сценарии включают более сложные манёвры, такие как облёт квадрата со стороной 3 м с паузами в углах и движение по спирали с постепенным увеличением радиуса и поворотом.

4.2.2. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ МОДЕЛЕЙ И АНАЛИЗ ОШИБОК

Для проведения экспериментов с генерацией кода управления БПЛА были использованы несколько версий языковых моделей Qwen2.5-Coder: Qwen2.5-Coder-3B-Instruct, Qwen2.5-Coder-1.5B-Instruct и Qwen2.5-Coder-0.5B-Instruct. Каждая из этих моделей отличается числом параметров, что позволяет оценить влияние модели на корректность выполненных заданий по генерируемому коду с первого запуска. Результаты специализированного теста семейства моделей Qwen2.5-Coder-Instruct представлены в таблице 3.

Таблица 3. Сравнение моделей Qwen на специализированном тесте

Модель	Параметры (кол-во)	Метрика (pass@1)	Скорость (токен/с)	Память (ГБ)	Оценка эффективности
Qwen2.5-Coder-3B-Instruct	3 млрд	0,81	7,2	4,8	1,215
Qwen2.5-Coder-1.5B-Instruct	1,5 млрд	0,76	9,6	3,2	2,28
Qwen2.5-Coder-0.5B-Instruct	500 млн	0,55	17,8	1,5	6,53

Тестирование проводилось с использованием встроенного нейронного процессора на Rockchip RK3588S и квантования в формате W8A8.

Основные типы ошибок в сгенерированном коде:

1) путаница в системах координат;

2) ошибки работы с датчиками, некорректная обработка возвращаемой информации.

Модель Qwen2.5-Coder-3B-Instruct с наибольшим количеством параметров (3 млрд) демонстрирует лучшую точность на тесте (pass@1 = 0,81), однако уступает в скорости генерации (7,2 токен/с) и требует больше памяти (4,8 ГБ) по сравнению с меньшими моделями. Модель среднего размера (1,5 млрд параметров) обеспечивает хороший баланс между качеством (0,76) и скоростью (9,6 токен/с), а самая компактная модель (500 млн параметров) работает быстрее всего (17,8 токен/с) при минимальном потреблении памяти (1,5 ГБ), но с заметно меньшей точностью (0,55).

Оценка эффективности показывает количество генерации (pass@1 × скорость) на единицу потребляемой памяти. Таким образом, специализированный тест способствует более обоснованному выбору моделей с оптимальным соотношением производительности и точности для реальных сценариев эксплуатации БПЛА.

В виде схемы отображается процесс реализации операций автономного управления БПЛА с применением LLM, см. рис. 1.

Оператор формулирует задачу, например, *«Взлететь, установить высоту 2 м»*. Запрос может вводиться текстом в чат-интерфейсе или голосом через микрофон. Если запрос продиктован голосом, модуль STT преобразует аудио в текстовое сообщение (промпт – prompt), сохраняя смысл и ключевые параметры. При вводе с клавиатуры этот блок пропускается. Большая языковая модель LLM принимает текстовое сообщение, анализирует требования и генерирует исполняемый код или миссию для дрона. Сгенерированный код передаётся на борт дрона по выбранному каналу связи – Wi-Fi, LTE или радиомодему с MAVLink-мостом.

Одноплатный компьютер – Jetson, Raspberry Pi, Orange Pi – принимает код и при необходимости компилирует или интерпретирует его, запускает в контейнере или в виртуальном окружении (реализация гибридного режима), конвертирует высоко-

уровневые действия («лети к точке X») в низкоуровневые приказы для полётного контроллера.

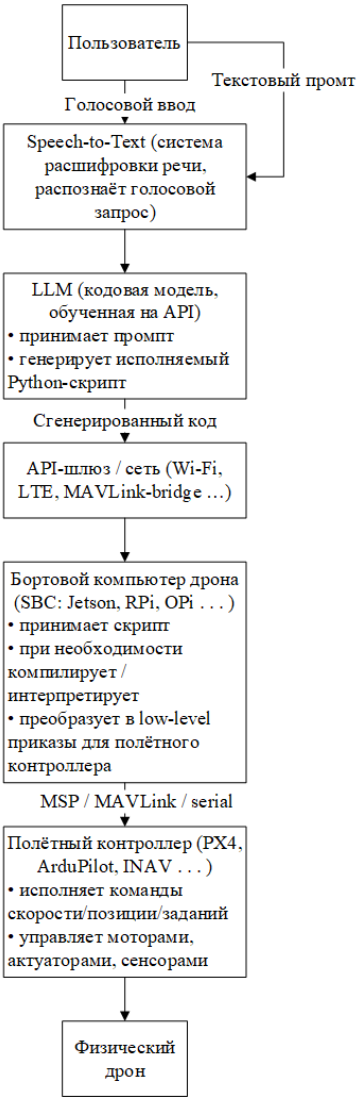

Рис. 1. Схема автономного управления БПЛА с применением LLM

PX4, ArduPilot, INAV или другой автопилот получает команды и генерирует сигналы для моторов, управляет стабилизацией, использует данные инерциального измерительного блока, GPS, барометра. Далее физический дрон выполняет манёвры, собирает телеметрию (позиция, батарея, видео). Эти данные могут возвращаться в LLM для уточнения следующих сообщений.

5. Выводы

Результаты показывают, что использование компьютеров с нейронным процессором на Rockchip RK3588S позволяет достичь лучшей скорости 17,8 токен/с. Это особенно важно для задач, требующих быстрого принятия решений, таких как автономный полет БПЛА. Все варианты моделей (Qwen2.5-Coder-3B-Instruct, Qwen2.5-Coder-1.5B-Instruct, Qwen2.5-Coder-0.5B-Instruct) демонстрируют стабильное качество на разработанном тесте, что подтверждает их пригодность для использования в системах с ограниченными ресурсами. В данной работе представлен специализированный тест для оценки способности языковых моделей генерировать корректный код управления БПЛА. Разработанный тест полностью адаптирован для проверки сгенерированного кода полетных заданий БПЛА.

Литература

1. ARIZA J.A., BAEZ H. *Understanding the role of single-board computers in engineering and computer science education: a systematic literature review* // Computer Applications in Engineering Education. – 2022. – Vol. 30(1). – P. 304–329. – DOI: 10.1002/cae.22439.
2. AUSTIN J., OZI A., DRAZNER S. *Program synthesis with large language models* // Journal of Artificial Intelligence Research. – 2021. – Vol. 71. – P. 355–420.
3. BROWN T.B. et al. *Language models are few-shot learners* // arXiv:2005.14165. – 2020. – 75 p.

4. CHEN K., WILLIAMS D., TAYLOR M. *SparseGPT: pruning large language models for embedded applications* // Journal of Artificial Intelligence Research. – 2024. – Vol. 55. – P. 213–229.
5. CHEN M., TWIGHT B., LEWIS M. *Evaluating large language models trained on code* // arXiv:2107.03374. – 2021.
6. DETTMERS T., ZETTLEMOYER L., LEWIS M. *LLM.int8(): 8-bit matrix multiplication for transformers at scale* // arXiv:2208.07339. – 2022.
7. JOHNSON T., LEE M., CARTER P. *Autonomous aerial exploration using compact language models. MIT computer science and artificial intelligence laboratory* // Technical Report MIT-CSAIL-TR-2024-03. – 2024.
8. KIM S., PAK D., LEE J. *Energy consumption analysis of LLM-powered UAVs* // Journal of Unmanned Aerial Systems. – 2024. – Vol. 12, No. 3. – P. 178–192.
9. KIM Y., PARK J., SHIN J. *PQK: practical quantization and pruning for on-device vision transformers* // arXiv:2106.14681. – 2021.
10. LEE D., SMITH R., PARKER A. *MiniLLM: extreme quantization for large language models on resource-constrained platforms* // Proc. of the Conf. on Machine Learning and Systems. – 2024. – P. 1243–1255.
11. LIU D., LI H., SHEN Y., et al. *KeepEdge: lightweight real-time detection for UAV applications* // IEEE Trans. on Industrial Informatics. – 2023. – Vol. 19(4). – P. 2873–2885.
12. MULLER A., SCHMIDT K., WEBER R. *Emergency response drone system with on-board decision making* // ETH Zurich Robotics and Perception Group Research Reports. – 2023.
13. RAMIREZ M., AL-HADITI F., NEIM S. *DroneGPT: natural language control for unmanned aerial vehicles* // Stanford AI Lab Technical Report, SAIL-TR-2023-04. – 2023.
14. SU J. et al. *RoFormer: enhanced transformer with rotary position embedding* // arXiv:2104.09864. – 2021.
15. TING L., JOHNSON M., CHEN K. *Performance evaluation of edge computing platforms for AI applications in UAVs* // IEEE Trans. on Robotics and Automation. – 2023. – Vol. 38, No. 2. – P. 112–127.

16. VAN X., BROWN T., ANDERSON J. *DistilLLM: knowledge distillation methods for efficient deployment of language models in UAVs* // Int. Conf. on Robotics and Automation (ICRA). – 2023. – P. 775–782.
17. VASWANI A. et al. *Attention is all you need* // 31st Conf. on Neural Information Processing Systems (NIPS–2017), Long Beach, CA, USA. – arXiv:1706.03762v7. – 2 Aug 2023. – P. 1–15.
18. ZHANG Y., LI W., MARTINEZ K. *Hybrid LLM-traditional architectures for autonomous UAV navigation* // Robotics and Autonomous Systems Journal. – 2024. – Vol. 152. – P. 45–62.

THE USE OF LLM MODELS ON SINGLE-BOARD COMPUTERS FOR THE IMPLEMENTATION OF AUTONOMOUS UAV FLIGHT

Rodion Anisimov, V.A. Trapeznikov Institute of Control Sciences of RAS, Moscow, master (rodion_anisimov@mail.ru).
Alexey Dvornikov, V.A. Trapeznikov Institute of Control Sciences of RAS, Moscow, master (applskyp@gmail.com).
Konstantin Kulagin, V.A. Trapeznikov Institute of Control Sciences of RAS, Moscow, Research Associate (kka8686@mail.ru).
Sofya Titova, Moscow Polytechnic University, student (titovas63059@gmail.com).
Konstantin Petrov, Moscow Polytechnic University, student (r.92rab@gmail.com).

Abstract: This paper examines the use of large language models (LLM) to control unmanned aerial vehicles (UAVs) using natural language commands. The research is aimed at solving a key problem – the discrepancy between the high computing requirements of LLM and the limited resources of on-board computers. The main focus is on optimizing LLM for operation on energy-efficient single-board computers with neuroprocessors, such as OrangePi 5B based on Rockchip RK3588S. The paper presents the results of testing Qwen2.5-Coder quantized models, demonstrating the preservation of code generation quality at processing speeds of up to 17.8 tokens/s. A specialized test (benchmark) was developed to evaluate the effectiveness of LLM integration into UAV autonomous control architecture and the correctness of code generation, including 125 scenarios. The results confirm the feasibility of LLM in autonomous drone control systems, although they reveal typical errors associated with sensor data processing and coordinate systems. The study proposes a promising direction for the development of intelligent UAV control systems with a

natural language interface (NLP). The study included both a scientific approach (development of a specialized test) and a technological innovation (performance analysis on single-board computers) aimed at integrating LLM into UAV autonomous control architecture.

Keywords: unmanned aerial vehicle, large language model, quantization, single-board computers, autonomous control, natural language processing.

УДК 004.93 + 004.8
ББК 39.52

Статья представлена к публикации
членом редакционной коллегии Н.А. Коргиным.

Поступила в редакцию 13.05.2025.
Опубликована 30.09.2025.