0. Контекст проекта

Мы строим распределённую LoRa-сеть, предоставляющую:

- **Чат** с end-to-end шифрованием
- Поток **телеметрии** (датчики, акторы)
- **Охранный режим** (alarm + автономное питание)
- Лёгкую **ИИ-надстройку** для rule-based автоматизации

В репозитории (ветка dev-experimental, см. архив) уже есть каркас: транспорт, маршрутизация, событийная шина, unit-тесты.
Ваша задача — доработать отдельный компонент и продемонстрировать инженерную культуру.

---

1. Основное задание (обязательное, ~70 % оценки)

PersistenceService: надёжное хранение состояния сети

1. **Спроектируйте** удобный public-интерфейс PersistenceService (файл уже есть, но пустой).
    - Должен сохранять и восстанавливать:
        - очередь исходящих, ещё не подтверждённых сообщений (Frame / Message)
        - статистику маршрутизации (RoutingTable2, успешные/неудачные попытки)
    - API не должен «протекать» на детали файловой системы/Flash.
2. **Реализуйте** (C++17):
    - Бинарный формат или JSON с версионированием.
    - Минимум два бэкенда – «POSIX-файл» (для CI/PC-сборок) и «Flash ESP32» (через SPIFFS или wear level FS, можно stub, см. README).
    - Потокобезопасность (могут писать/читать разные задачи FreeRTOS).
    - Защита от порчи (CRC32 или hash каждого блока).
3. **Unit-тесты** (GoogleTest) на PC-платформе:
    - save → load восстанавливает очередь сообщений и метрики 1:1;
    - повреждённый файл корректно детектируется и не ломает приложение.
4. **Сборка**: cmake .. && make && ctest (CI будет крутиться под Ubuntu 22.04, g++-13).

---

2. Мини-ревью чужого кода (~15 %)

Посмотрите два файла из репозитория:

- src/core/ThreadSafeQueue.h
- src/events/EventBus.h

Сформулируйте **минимум по три** потенциальные проблемы/улучшения в каждом (производительность, корректность, безопасность, стиль) и предложите исправления. Формат – code_review.md, кратко и по делу.

---

3. Доп-часть (опционально, +15 % к итоговой оценке)

Фрагментация крупных сообщений

LoRa payload ≤ 51 байта @SF12.
Реализуйте в MessageCodec автоматическую нарезку (encodeFragments) и сборку (reassemble) длинных сообщений:

- Номер фрагмента, общее количество, CRC каждого куска.
- Устойчивость к дублированию/повторному порядку.
- Тесты: передача 2-, 5-, 10-фрагментных сообщений в перемешанном порядке.

---

4. Что оцениваем

Критерий	**Вес**
Корректность, покрытие тестами	40%
Читаемость, архитектурная чистота	25%
СИ/CI, reproducible build	15%
Code-review-document	15%
(опция) Фрагментация	+15%

5. Как сдавать

1. Сделайте приватный Git-репо или передайте локальный zip файл.
2. Коммиты атомарные, внятные (git rebase -i приветствуется).
3. Pull Request / архив + SHA, дедлайн — 72 часа после получения задания.
4. В README опишите:
    - зависимости (кросс-компилятор, PlatformIO, Python-скрипты и т.п.)
    - как запускать тесты, пример вывода
    - если делали опцию (3) – как смоделировать.

---

6. Ресурсы

- API LoRa MAC – [Semtech AN1200.22]
- ESP-IDF FreeRTOS docs

- GoogleTest quick-start