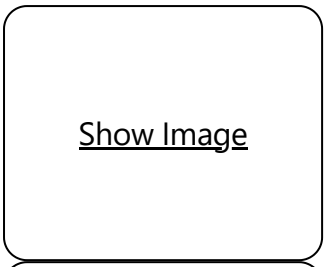
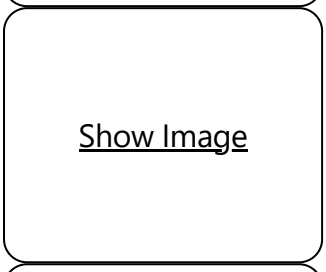
🚤 Autonomous Boat Drone





> Автономная лодка-дрон на базе Samsung S21 Ultra с GPS навигацией и дистанционным управлением

📋 Содержание

🎯 Описание проекта

Autonomous Boat Drone — это открытый проект автономной лодки-дрона, использующей Samsung S21 Ultra в качестве основного вычислительного модуля. Система обеспечивает автономную навигацию по запрограммированному маршруту с использованием GPS, автоматическое управление двигателем и рулем, а также передачу данных о местоположении в реальном времени.

📹 Демонстрация

> *Здесь будет видео с демонстрацией работы лодки-дрона*

🌟 Основные особенности

- **Автономная навигация** с использованием GPS высокой точности
- **PID-регулирование** курса для точного следования маршруту
- **Веб-интерфейс** для мониторинга и управления
- **Система безопасности** с множественными уровнями защиты
- **Модульная архитектура** для легкого расширения функционала

⚡ Основные функции

🧭 Навигация

- [x] GPS навигация с точностью до 1 метра
- [x] Автоматическое следование по маршруту
- [x] PID-регулирование курса
- [x] Избегание препятствий (планируется)
- [x] Автоматический возврат домой

🎮 Управление

- [x] Дистанционное управление через веб-интерфейс
- [x] Программирование маршрутов
- [x] Реальное время телеметрии
- [x] Управление скоростью и направлением
- [x] Экстренная остановка

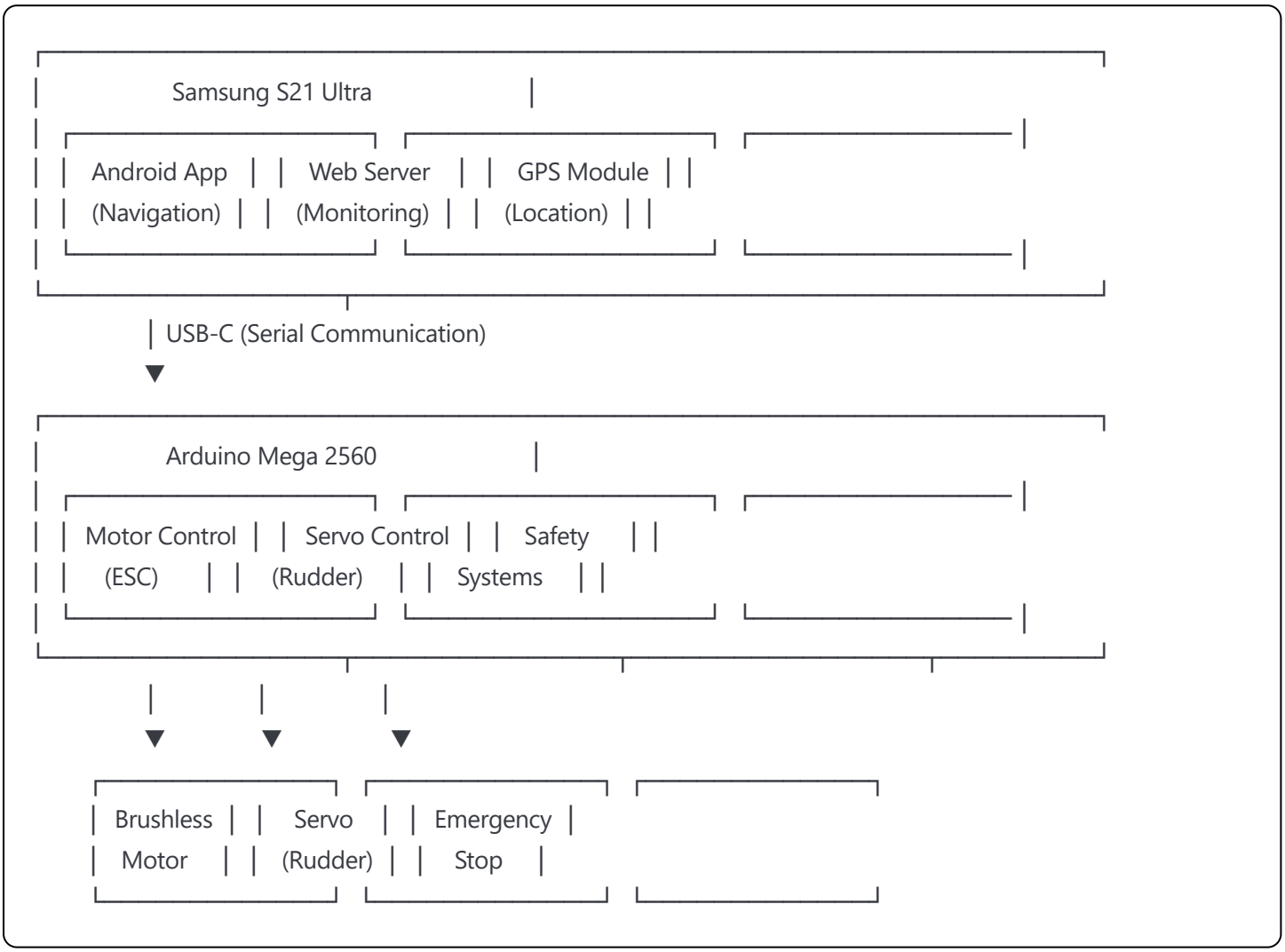
📡 Связь

- [x] 4G/5G подключение для удаленного мониторинга
- [x] Wi-Fi точка доступа для локального управления
- [x] Bluetooth для отладки
- [x] WebSocket для данных в реальном времени

🔒 Безопасность

- [x] Геозоны (запретные области)
- [x] Автоматическое отключение при потере связи
- [x] Мониторинг заряда батареи
- [x] Физическая кнопка экстренной остановки
- [x] Watchdog таймеры

🏗️ Архитектура системы



🔧 Аппаратные компоненты

🧮 Основные компоненты

Компонент	Модель	Цена	Описание
Главный модуль	Samsung S21 Ultra	-	GPS, вычисления, связь
Микроконтроллер	Arduino Mega 2560	$25	Управление моторами
USB Hub	Anker PowerExpand+ 7-in-1	$45	OTG + зарядка
Двигатель	Brushless Motor 2838	$60	Водонепроницаемый
ESC	60A Brushless ESC	$40	Контроллер мотора
Сервопривод	MG995 Servo	$15	Управление рулем
Аккумулятор	3S LiPo 5000mAh	$60	Основное питание
Powerbank	20000mAh USB-C PD	$50	Питание телефона

📡 Антенны и связь

Компонент	Модель	Цена	Описание
GPS Усилитель	GPS Signal Booster 28dB	$80	Улучшение GPS
4G Антенна	Directional 4G Antenna	$50	Дальняя связь
Wi-Fi Антенна	Yagi 2.4/5GHz	$40	Локальная связь

🛡️ Защита и корпус

Компонент	Модель	Цена	Описание
Корпус	Waterproof Box IP67	$80	Защита электроники
Разъемы	Waterproof Connectors	$30	Герметичные соединения
Поплавки	Foam Floats	$20	Дополнительная плавучесть

Общая стоимость: ~$595 (без Samsung S21 Ultra)

🔌 Схема подключения

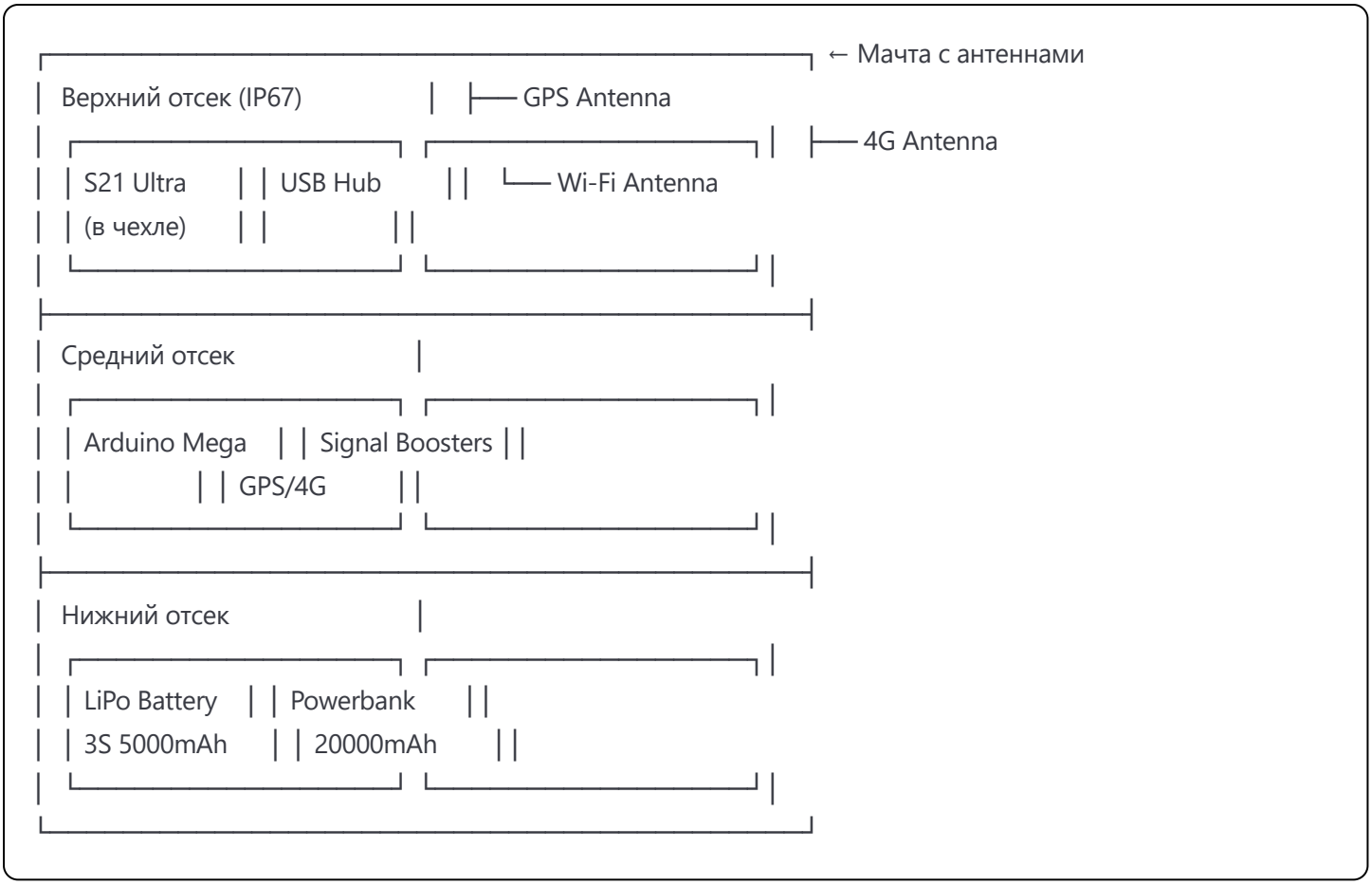
Электрическая схема

```
Samsung S21 Ultra
  │
  ├── USB-C Hub ──┬── USB-A → Arduino Mega
  │               ├── USB-C → Power Delivery (65W)
  │               └── USB-A → Wi-Fi Adapter (optional)
  │
  └── Внешние антенны (беспроводные усилители):
      ├── GPS Booster → GPS Antenna
      ├── 4G Booster → 4G Antenna
      └── Wi-Fi Extender → Wi-Fi Antenna

Arduino Mega 2560
  ├── Pin 9 (PWM) → ESC Signal Wire
  ├── Pin 10 (PWM) → Servo Signal Wire
  ├── Pin 2 (Digital) → Emergency Stop Button
  ├── Pin 13 (Digital) → Status LED
  ├── Pin A0 (Analog) → Battery Voltage Monitor
  ├── 5V → Servo Power (+)
  └── GND → Common Ground

Power Distribution
  ├── 3S LiPo 11.1V → ESC → Brushless Motor
  ├── 5V BEC → Arduino + Servo
  ├── 12V → GPS/4G Boosters
  └── USB-C PD → Samsung S21 Ultra
```

Схема корпуса



⚙️ Установка и настройка

📋 Предварительные требования

- **Android устройство**: Samsung S21 Ultra (или совместимое)
- **Android версия**: 11.0 или выше
- **Arduino IDE**: версия 1.8.19 или выше
- **USB OTG поддержка**: обязательно
- **Разрешения Android**: GPS, Camera, Storage, Network

📱 Установка Android приложения

1. **Клонирование репозитория**

```bash
git clone https://github.com/username/autonomous-boat-drone.git
cd autonomous-boat-drone
```

2. **Установка зависимостей**

```bash
```

```
# Android Studio
./gradlew build

# Или установка APK
adb install app/build/outputs/apk/debug/boat-autopilot-debug.apk
```

3. **Настройка разрешений**

```xml
<!-- AndroidManifest.xml уже настроен -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.USB_PERMISSION" />
<uses-permission android:name="android.permission.INTERNET" />
```

🔧 Программирование Arduino

1. **Установка библиотек**

```cpp
// В Arduino IDE: Sketch → Include Library → Manage Libraries
// Установить: Servo, SoftwareSerial
```

2. **Загрузка прошивки**

```bash
# Откройте arduino/boat_controller/boat_controller.ino
# Выберите: Tools → Board → Arduino Mega 2560
# Выберите: Tools → Port → /dev/ttyUSB0 (или COM порт)
# Нажмите: Upload
```

3. **Калибровка ESC**

```cpp
// Первый запуск - автоматическая калибровка
// Следуйте инструкциям в Serial Monitor
```

🌐 Настройка веб-сервера

1. **Настройка конфигурации**

```javascript
```

```
// src/config/config.js
const CONFIG = {
    SERVER_URL: 'https://your-domain.com',
    WEBSOCKET_URL: 'wss://your-domain.com/ws',
    API_KEY: 'your-api-key'
};
```

2. **Деплой веб-интерфейса**

```
bash

cd web-interface
npm install
npm run build
npm run deploy
```

🚀 Использование

📍 Первый запуск

1. **Включение системы**

1. Подключите все компоненты согласно схеме
2. Включите питание (аккумулятор → Arduino → телефон)
3. Запустите приложение на телефоне
4. Дождитесь инициализации GPS (2-5 минут)

2. **Калибровка компаса**

1. В приложении: Settings → Calibrate Compass
2. Поверните лодку на 360° медленно
3. Дождитесь сообщения "Calibration Complete"

3. **Тестирование связи**

1. Проверьте статус: USB ✅, GPS ✅, 4G ✅
2. Выполните команду "Test Motors"
3. Проверьте отклик руля и мотора

🗺️ Программирование маршрута

1. **Создание маршрута**

```
json
```

```
{
  "route_name": "Test Route",
  "waypoints": [
    {"lat": 55.7558, "lon": 37.6176, "speed": 50},
    {"lat": 55.7539, "lon": 37.6208, "speed": 30}
  ],
  "safety": {
    "max_distance": 1000,
    "return_battery": 20
  }
}
```

2. **Загрузка маршрута**

```
bash

# Через веб-интерфейс
1. Откройте http://boat-monitor.local
2. Waypoints → Load Route → Выберите файл

# Или через приложение
1. Menu → Routes → Import → Выберите JSON файл
```

🎮 Управление

Автоматический режим

```
java

// Запуск автопилота
boatController.startAutopilot();

// Остановка
boatController.stopAutopilot();

// Экстренная остановка
boatController.emergencyStop();
```

Ручное управление

```
java
```

```
// Управление мотором (0-100%)
boatController.setMotorSpeed(75);

// Управление рулем (-45° до +45°)
boatController.setRudderAngle(15);

// Получение телеметрии
Telemetry data = boatController.getTelemetry();
```

📊 Мониторинг

Веб-интерфейс

- **URL**: `http://boat-ip:8080` или `https://your-domain.com`
- **Функции**: Карта в реальном времени, телеметрия, управление
- **Совместимость**: Chrome, Firefox, Safari, Edge

Мобильное приложение

- **Локальный мониторинг**: Все данные на экране
- **Push уведомления**: Критические события
- **Офлайн режим**: Сохранение данных при потере связи

🛡️ Безопасность

🚨 Системы безопасности

Аппаратная защита

- **Emergency Stop Button**: Физическая кнопка мгновенной остановки
- **Watchdog Timer**: Автоматический перезапуск при зависании
- **Voltage Monitor**: Мониторинг напряжения батарей
- **Tilt Sensor**: Обнаружение переворачивания

Программная защита

- **Geofencing**: Виртуальные границы
- **Connection Timeout**: Остановка при потере связи
- **Battery Monitor**: Автоматический возврат при разряде
- **Speed Limiter**: Ограничение максимальной скорости

Протоколы безопасности

```java
// Проверки безопасности
public class SafetyController {
    public boolean checkSafety() {
        return checkBattery() &&
               checkConnection() &&
               checkGeofence() &&
               checkWeather();
    }

    public void emergencyProcedure() {
        stopMotor();
        centerRudder();
        sendDistressSignal();
        logEmergency();
    }
}
```

⚖️ Юридические требования

Россия

- **ГИМС регистрация**: Если мощность > 10 л.с.
- **Права на управление**: Для судов > 5 л.с.
- **Страхование**: Рекомендуется
- **Радиочастоты**: Лицензия для передатчиков

Международные воды

- **IMO regulations**: Следование международным правилам
- **AIS transponder**: Для судов > 300 тонн (не применимо)
- **Navigation lights**: Обязательно в темное время

📋 Безопасное использование

1. **Перед запуском**
   - ✅ Проверить прогноз погоды
   - ✅ Убедиться в заряде всех батарей
   - ✅ Проверить связь с берегом
   - ✅ Уведомить службы о планах
2. **Во время работы**
   - ✅ Постоянный мониторинг

- ✅ Готовность к экстренному вмешательству
- ✅ Соблюдение геозон
- ✅ Мониторинг погодных условий

3. **После использования**
   - ✅ Безопасная парковка/швартовка
   - ✅ Отключение всех систем
   - ✅ Извлечение и зарядка батарей
   - ✅ Анализ логов для улучшений

📡 API и протоколы

REST API

Телеметрия

```http
GET /api/v1/telemetry
Content-Type: application/json

{
  "timestamp": 1690123456789,
  "location": {
    "latitude": 55.7558,
    "longitude": 37.6176,
    "altitude": 150.0,
    "accuracy": 3.0
  },
  "navigation": {
    "speed": 2.5,
    "heading": 45.0,
    "target_heading": 47.0
  },
  "systems": {
    "motor_speed": 75,
    "rudder_angle": 5,
    "battery_level": 85,
    "connection_strength": 4
  }
}
```

Управление

```http
```

```
POST /api/v1/control/motor
Content-Type: application/json

{
  "speed": 80,
  "duration": 10000
}
```

```http
POST /api/v1/control/rudder
Content-Type: application/json

{
  "angle": 15,
  "smooth": true
}
```

Маршруты

```http
POST /api/v1/routes
Content-Type: application/json

{
  "name": "Harbor Tour",
  "waypoints": [
    {"lat": 55.7558, "lon": 37.6176, "speed": 50, "action": "navigate"},
    {"lat": 55.7539, "lon": 37.6208, "speed": 30, "action": "photo"}
  ],
  "safety_settings": {
    "max_distance_from_home": 1000,
    "low_battery_return": 20,
    "max_speed": 100
  }
}
```

WebSocket Protocol

Подключение

```javascript
```

```javascript
const ws = new WebSocket('wss://boat.example.com/ws');

ws.onopen = function() {
    console.log('Connected to boat');

    // Аутентификация
    ws.send(JSON.stringify({
        type: 'auth',
        token: 'your-jwt-token'
    }));
};
```

Сообщения

```javascript
// Real-time телеметрия
{
  "type": "telemetry",
  "data": {
    "location": [55.7558, 37.6176],
    "speed": 2.5,
    "heading": 45.0,
    "battery": 85
  }
}

// Команды управления
{
  "type": "command",
  "action": "set_motor_speed",
  "params": {"speed": 75}
}

// Системные события
{
  "type": "event",
  "level": "warning",
  "message": "Low battery detected",
  "timestamp": 1690123456789
}
```

Serial Protocol (USB)

Команды Arduino

```
# Формат: COMMAND:PARAMETER\n

MOTOR:75        # Установить скорость мотора 75%
RUDDER:15       # Повернуть руль на 15°
ARM             # Взвести систему
DISARM          # Разоружить систему
STATUS          # Запросить статус
PING            # Проверка связи
```

Ответы Arduino

```
# Формат: STATUS:PARAMETER или ERROR:MESSAGE\n

OK:MOTOR:75      # Скорость мотора установлена
OK:RUDDER:15     # Руль повернут
ERROR:EMERGENCY_STOP_ACTIVE  # Ошибка: экстренная остановка
PONG             # Ответ на PING
WARNING:LOW_BATTERY:10.5V    # Предупреждение
```

🌐 Веб-интерфейс

📊 Возможности интерфейса

- **Real-time карта** с положением лодки
- **Телеметрия** в реальном времени
- **Управление маршрутами** через drag & drop
- **Мониторинг систем** (батарея, связь, моторы)
- **Журнал событий** с фильтрацией
- **Настройки безопасности**

🎮 Панель управления

Основные элементы

```
html
```

```
<!-- Статус подключения -->
<div class="connection-status connected">🟢 Подключено</div>

<!-- Телеметрия -->
<div class="telemetry-panel">
  <div class="metric">
    <label>Широта</label>
    <span id="latitude">55.7558</span>
  </div>
  <div class="metric">
    <label>Скорость</label>
    <span id="speed">2.5 км/ч</span>
  </div>
</div>

<!-- Управление -->
<div class="control-panel">
  <button onclick="startAutopilot()">▶️ Запуск</button>
  <button onclick="emergencyStop()">🚨 Стоп</button>
</div>
```

Интерактивная карта

javascript

```
// Leaflet.js карта
const map = L.map('map').setView([55.7558, 37.6176], 13);

// Маркер лодки
const boatMarker = L.marker([55.7558, 37.6176], {
    icon: L.divIcon({
        html: '🚤',
        className: 'boat-marker'
    })
}).addTo(map);

// Маршрут
const route = L.polyline(waypoints, {
    color: 'blue',
    weight: 3
}).addTo(map);
```

📱 Адаптивный дизайн

Брейкпоинты

```css
/* Desktop */
@media (min-width: 1024px) {
  .dashboard { grid-template-columns: 1fr 1fr 1fr; }
}

/* Tablet */
@media (max-width: 1023px) {
  .dashboard { grid-template-columns: 1fr 1fr; }
}

/* Mobile */
@media (max-width: 767px) {
  .dashboard { grid-template-columns: 1fr; }
  .map { height: 250px; }
}
```

👥 Разработка

🛠️ Настройка среды разработки

1. **Клонирование репозитория**

```bash
git clone https://github.com/username/autonomous-boat-drone.git
cd autonomous-boat-drone
```

2. **Android разработка**

```bash
# Android Studio
./gradlew assembleDebug

# Установка на устройство
adb install -r app/build/outputs/apk/debug/app-debug.apk
```

3. **Arduino разработка**

```bash
```

```
# Установка библиотек
arduino-cli lib install "Servo"
arduino-cli lib install "SoftwareSerial"

# Компиляция
arduino-cli compile --fqbn arduino:avr:mega arduino/boat_controller

# Загрузка
arduino-cli upload -p /dev/ttyUSB0 --fqbn arduino:avr:mega arduino/boat_controller
```

4. **Веб-интерфейс**

```bash
cd web-interface
npm install
npm run dev  # Development server
npm run build  # Production build
```

📁 Структура проекта

```
autonomous-boat-drone/
├── android/                 # Android приложение
│   ├── app/src/main/java/   # Исходный код Java
│   ├── app/src/main/res/    # Ресурсы (layout, strings)
│   └── build.gradle         # Gradle конфигурация
├── arduino/                 # Arduino прошивка
│   ├── boat_controller/     # Основной скетч
│   ├── libraries/           # Дополнительные библиотеки
│   └── tests/               # Тесты hardware
├── web-interface/           # Веб-интерфейс
│   ├── src/                 # Исходный код (HTML/CSS/JS)
│   ├── public/              # Статические файлы
│   └── dist/                # Собранная версия
├── server/                  # Backend сервер
│   ├── api/                 # REST API endpoints
│   ├── websocket/           # WebSocket handlers
│   └── database/            # Database schemas
├── docs/                    # Документация
│   ├── hardware/            # Схемы и спецификации
│   ├── software/            # Техническая документация
│   └── user-manual/         # Руководство пользователя
├── tests/                   # Автоматические тесты
│   ├── unit/                # Модульные тесты
│   ├── integration/         # Интеграционные тесты
│   └── hardware/            # Тесты железа
├── tools/                   # Вспомогательные инструменты
│   ├── calibration/         # Калибровка сенсоров
│   ├── simulation/          # Симуляторы
│   └── monitoring/          # Мониторинг и логирование
├── README.md                # Основная документация
├── LICENSE                  # Лицензия
├── CONTRIBUTING.md          # Гайд для контрибьюторов
└── CHANGELOG.md             # История изменений
```

🧪 Тестирование

Unit тесты

bash

```
# Android тесты
./gradlew test

# Arduino тесты (с помощью ArduinoUnit)
arduino-cli compile --verify arduino/tests/

# JavaScript тесты
npm test
```

Интеграционные тесты

```bash
# Полный цикл тестирования
./scripts/integration-test.sh

# Симуляция Hardware in the Loop
./tools/simulation/hil-test.py
```

Hardware тесты

```bash
# Тест моторов и сервоприводов
./tests/hardware/motor-test.sh

# Тест GPS точности
./tests/hardware/gps-test.sh

# Тест дальности связи
./tests/hardware/range-test.sh
```

📝 Стиль кода

Java (Android)

```java
```

```java
// Google Java Style Guide
public class BoatController {
    private static final int MOTOR_MAX_SPEED = 100;

    public void setMotorSpeed(int speed) {
        if (speed < 0 || speed > MOTOR_MAX_SPEED) {
            throw new IllegalArgumentException("Invalid speed");
        }
        // Implementation
    }
}
```

C++ (Arduino)

```cpp
// Arduino Style Guide
#define MOTOR_PIN 9
#define RUDDER_PIN 10

void setMotorSpeed(int speed) {
  speed = constrain(speed, 0, 100);
  // Implementation
}
```

JavaScript (Web)

```javascript
// StandardJS Style
const BoatMonitor = {
  async updateTelemetry() {
    try {
      const data = await fetch('/api/telemetry')
      this.displayData(await data.json())
    } catch (error) {
      console.error('Telemetry error:', error)
    }
  }
}
```

🔄 CI/CD Pipeline

```yaml
```

```yaml
# .github/workflows/main.yml
name: CI/CD Pipeline

on: [push, pull_request]

jobs:
  android-build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Android SDK
        uses: android-actions/setup-android@v2
      - name: Build APK
        run: ./gradlew assembleDebug
      - name: Run tests
        run: ./gradlew test

  arduino-build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Arduino CLI
        uses: arduino/setup-arduino-cli@v1
      - name: Compile sketches
        run: arduino-cli compile --fqbn arduino:avr:mega

  web-build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'
      - name: Install dependencies
        run: npm ci
      - name: Build
        run: npm run build
      - name: Deploy
        if: github.ref == 'refs/heads/main'
        run: npm run deploy
```

🤝 Contributing

Мы приветствуем вклад в развитие проекта! Пожалуйста, ознакомьтесь с CONTRIBUTING.md для получения подробной информации.

📋 Как внести свой вклад

1. **Fork** репозитория
2. Создайте **feature branch** (`git checkout -b feature/amazing-feature`)
3. **Commit** ваши изменения (`git commit -m 'Add amazing feature'`)
4. **Push** в branch (`git push origin feature/amazing-feature`)
5. Откройте **Pull Request**

🐛 Баг репорты

Используйте GitHub Issues для сообщения об ошибках. Включите:

- Описание проблемы
- Шаги для воспроизведения
- Ожидаемое поведение
- Логи и скриншоты
- Версия ПО и железа

💡 Feature запросы

Мы открыты для новых идей! Создайте Issue с тегом `enhancement` и опишите:

- Функциональность которую вы хотите
- Зачем она нужна
- Как это должно работать

📄 Лицензия

📞 Контакты

- **Project Lead**: Your Name
- **Project Website**: https://autonomous-boat-drone.github.io
- **Documentation**: https://docs.autonomous-boat-drone.com
- **Support**: GitHub Discussions

🙏 Благодарности

- Arduino Community за отличную платформу
- Android Open Source Project за мобильную ОС
- Leaflet за картографическую библиотеку
- Всем контрибьюторам проекта! 🎉

⭐ **Поставьте звезду**, если этот проект был полезен для вас!

Show Image