ДПП 2025

Проект представляет собой комплексное решение для управления мобильным ровером с возможностью анализа минералов. Система включает в себя несколько компонентов, каждый из которых выполняет свои функции:

- **Пункт управления**: десктопный интерфейс для управления ровером, построенный на базе Electron. В эту часть входит HTML/CSS/JS (index.html, main.js, preload.js, package.json), а также Python-скрипт для анализа камней (yaan.py), который интегрирован через asar-архив.
- **Ровер**: на Raspberry Pi работает ROS-пакет brorover_server, автоматически запускающийся через launch файл как сервис в systemd, обеспечивающий связь с остальными компонентами системы, обработку одометрии и передачу команд управления.
- **Контроллер**: программное обеспечение для Arduino Micro (dpp2025_controller.ino), которое считывает данные с физического контроллера (джойстики, кнопка, потенциометр) и отправляет их на ПК через serial порт.
- **Модуль**: код для CAN-модуля на ровере (dpp2025_can.ino), предназначенный для обмена данными по шине CAN, управления приводами и мониторинга состояния.

> [!NOTE] Подключение происходит через websocket, что гораздо стабильнее, чем стандартная отправка запросов по HTTP и экспорт ROS-топиков.

Структура проекта

```
/DPP2025
│
├── /pc
│   ├── index.html
│   ├── main.js
│   ├── preload.js
│   ├── package.json
│   └── yaan.py
│
├── /rover
│   └── server.py
│
├── /controller
│   └── dpp2025_controller.ino
│
└── /module
    └── dpp2025_can.ino
```

Компоненты

1. **Пункт управления**

   Пользователь запускает десктопное приложение с элементами управления, индикаторами, картой одометрии, и видео-потоками с камеры ровера и станции. Из приложения происходит

обработка сигналов с контроллера и отправка команд роверу. При необходимости можно запустить анализ камней.

2. **Ровер**

   На ровере запущен сервер, который обрабатывает входящие команды, осуществляет сбор телеметрии (одометрия) и передаёт данные в графический интерфейс, а также выполняет обмен информацией с модулем CAN.

3. **Контроллер**

   Программа для Arduino Micro считывает аналоговые и цифровые сигналы, формирует структуру JSON и передаёт данные через последовательный порт на ПК для дальнейшей обработки.

4. **Модуль**

   Код отвечает за обмен данными через шину CAN, управление приводами и сбор данных о состоянии механических узлов на ровере.

Установка и запуск

Требования

- **ПК (Пункт управления)**:
  - Node.js
  - Electron
  - Electron Forge
  - Python 3
- **Ровер**:
  - ROS
  - Python 3
  - socketio
  - python-can
- **Контроллер и Модуль**:
  - Arduino IDE для компиляции и загрузки прошивок на Arduino Micro
  - Драйверы для работы с CAN-модулем

Инструкции для запуска

0. Перед началом работы склонируйте репозиторий на ПК и на ровер.

1. **Пункт управления (ПК)**

   - Откройте терминал и перейдите в папку `/pc`.
   - Выполните команду `npm install` для установки необходимых npm-пакетов.
   - Для тестового запуска запустите `npm start`. Приложение откроется с управлением и видео-потоками.
   - Скрипт `yaan.py` вызывается через интерфейс (кнопка "Запустить анализ") и запакован в asar-архив при сборке приложения.

- Для сборки приложения выполните.

2. **Ровер (ROS-пакет)**
   - Разверните проект на устройстве, где установлен ROS.
   - Запустите скрипт в рамках ROS-пакета:

     ```
     roslaunch brorover_server server.launch
     ```

   - Убедитесь, что настроены правильные топики для передачи одометрии и управления.
   - Добавьте скрипт в автозапуск через systemd или иначе.
3. **Контроллер (Arduino Micro)**
   - Откройте файл `dpp2025_controller.ino` в Arduino IDE.
   - Проверьте настройки последовательного порта и, при необходимости, скорректируйте их.
   - Скомпилируйте и загрузите прошивку на Arduino Micro.
4. **Модуль (STM32)**
   - Откройте файл `dpp2025_can.ino` в Arduino IDE.
   - Проверьте параметры и настройки CAN-интерфейса.
   - Скомпилируйте и загрузите прошивку на модуль VBCore.

Подробная документация для каждого компонента доступна в папках соответствующих компонентов.