ConvEnviTest - 환경 테스트 시스템

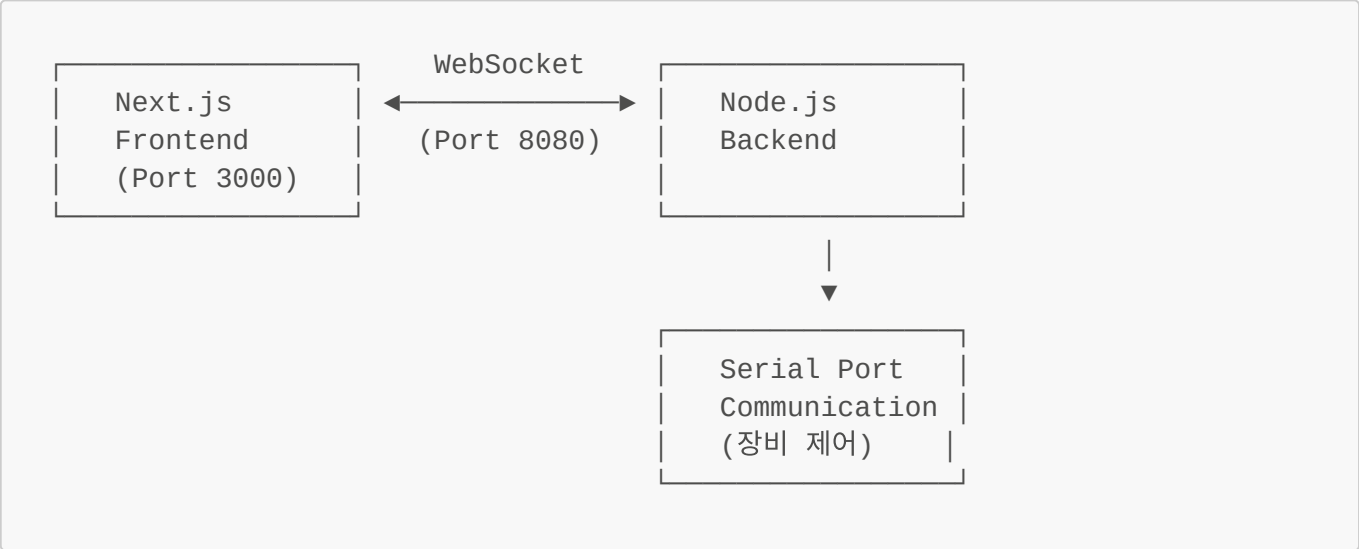
📋 프로젝트 개요

ConvEnviTest는 환경 테스트 장비를 제어하고 모니터링하는 웹 기반 시스템입니다. 이 시스템은 Next.js 프론트엔드와 Node.js 백엔드로 구성되어 있으며, WebSocket을 통해 실시간 통신을 지원합니다.

주요 기능

- 🔌 **릴레이 제어**: 10개 채널의 릴레이 ON/OFF 제어
- ⚡ **전압 설정 및 측정**: 각 채널별 전압 설정 및 실시간 측정
- 🌡️ **온도 제어**: 챔버 온도 설정 및 모니터링
- 🔄 **자동 테스트 프로세스**: 설정된 조건에 따른 자동 테스트 실행
- 📊 **실시간 데이터 모니터링**: 전압, 온도, 테스트 결과 실시간 표시
- 💾 **설정 저장**: 모든 설정값을 JSON 파일로 저장 및 복원

🏗️ 시스템 아키텍처



!)

🚀 설치 및 실행 가이드

사전 요구사항

- **Node.js**: 18.0.0 이상
- **npm** 또는 **pnpm**: 패키지 관리자
- **Git**: 버전 관리 시스템

1. 프로젝트 클론

```
# GitHub에서 프로젝트 클론
git clone <repository-url>
cd convEnviTest
```

2. 백엔드 서버 설치 및 실행

```
# 서버 디렉토리로 이동
cd server

# 의존성 설치
npm install

# 개발 모드로 서버 실행
npm run dev

# 또는 프로덕션 모드로 실행
npm start
```

백엔드 서버는 포트 8080에서 실행됩니다.

3. 프론트엔드 설치 및 실행

```
# 새 터미널을 열고 프로젝트 루트로 이동
cd nextjs

# 의존성 설치
npm install

# 개발 서버 실행
npm run dev
```

프론트엔드는 포트 3000에서 실행됩니다.

4. 브라우저에서 접속

웹 브라우저를 열고 다음 주소로 접속하세요:

```
http://localhost:3000
```

🐳 Docker를 사용한 설치 (선택사항)

Docker와 Docker Compose가 설치되어 있다면 더 간단하게 설치할 수 있습니다:

1. Docker 설치 확인

```
# Docker 버전 확인
docker --version
docker-compose --version
```

2. Docker Compose로 실행

```
# 프로젝트 루트 디렉토리에서
docker-compose up -d

# 로그 확인
docker-compose logs -f

# 서비스 중지
docker-compose down
```

3. 개별 서비스 실행

```
# 백엔드만 실행
docker-compose up backend

# 프론트엔드만 실행
docker-compose up frontend
```

주의: Docker를 사용할 때는 시리얼 포트 접근을 위해 `--privileged` 플래그가 필요할 수 있습니다.

📁 프로젝트 구조

```
convEnviTest/
├── server/                          # 백엔드 서버
│   ├── backend-websocket-server.js  # 메인 서버 파일
│   ├── package.json                 # 서버 의존성
│   ├── *.js                        # 가ㄱ종 모듈 파일들
│   └── *.json                      # 설정 저장 파일들
├── nextjs/                          # 프론트엔드 애플리케이션
│   ├── pages/                       # Next.js 페이지
│   ├── components/                   # React 컴포넌트들
│   ├── styles/                      # CSS 스타일 파일들
│   └── package.json                 # 프론트엔드 의존성
└── README.md                        # 이 파일
```

🔧 주요 컴포넌트 설명

백엔드 모듈

- **`backend-websocket-server.js`**: 메인 WebSocket 서버
- **`ReadChamber.js`**: 챔버 온도 읽기
- **`ReadVolt.js`**: 전압 측정
- **`SetVolt.js`**: 전압 설정

- `SelectDevice.js`: 릴레이 제어
- `RunTestProcess.js`: 자동 테스트 프로세스

프론트엔드 컴포넌트

- `PowerSwitch.tsx`: 전원 스위치 제어
- `RelayOnOff.tsx`: 릴레이 ON/OFF 제어
- `SetVolt.tsx`: 전압 설정 패널
- `ReadVolt.tsx`: 전압 측정 표시
- `ReadChamber.tsx`: 온도 모니터링
- `TestProcess.tsx`: 테스트 프로세스 제어
- `SystemSet.tsx`: 시스템 설정

⚙ 설정 파일

백엔드 서버는 다음 JSON 파일들을 통해 설정을 관리합니다:

- `delay_settings.json`: 딜레이 설정
- `device_states.json`: 장비 상태
- `high_temp_settings.json`: 고온 설정
- `low_temp_settings.json`: 저온 설정
- `product_input.json`: 제품 정보
- `usb_port_settings.json`: USB 포트 설정
- `out_volt_settings.json`: 출력 전압 설정
- `channel_voltages.json`: 채널 전압 설정

🔌 하드웨어 연결

시리얼 포트 설정

시스템은 시리얼 포트를 통해 장비와 통신합니다. 다음 명령어로 사용 가능한 포트를 확인하세요:

```
# Linux/Mac
ls /dev/tty*

# Windows
# 장치 관리자에서 COM 포트 확인
```

릴레이 명령어

시스템은 Modbus RTU 프로토콜을 사용하여 릴레이를 제어합니다:

```
#1 RELAY ON: 010600010100D99A   OFF: 010600010200D96A
#2 RELAY ON: 010600020100299A   OFF: 010600020200296A
#3 RELAY ON: 010600030100785A   OFF: 01060003020078AA
#4 RELAY ON: 010600040100C99B   OFF: 010600040200C96B
#5 RELAY ON: 010600050100985B   OFF: 01060005020098AB
#6 RELAY ON: 020600010100D9A9   OFF: 020600010200D959
```

```
#7 RELAY ON: 020600020100029A9   OFF: 0206000202002959
#8 RELAY ON: 0206000301007869   OFF: 0206000302007899
#9 RELAY ON: 020600040100C9A8   OFF: 020600040200C958
#10 RELAY ON: 0206000501009868  OFF: 0206000502009898
```

🎯 사용 방법

1. 시스템 시작

1. 백엔드 서버를 먼저 실행
2. 프론트엔드 애플리케이션 실행
3. 웹 브라우저에서 `http://localhost:3000` 접속

2. 기본 설정

1. **USB 포트 선택**: 장비가 연결된 COM 포트 선택
2. **시스템 설정**: 제품 정보 및 기본 설정 입력
3. **전압 설정**: 각 채널별 전압값 설정
4. **온도 설정**: 고온/저온 설정값 입력

3. 테스트 실행

1. **Power Switch**: 전원 스위치를 ON으로 설정
2. **자동 프로세스**: 설정된 조건에 따라 자동 테스트 실행
3. **모니터링**: 실시간으로 전압, 온도, 테스트 결과 확인
4. **결과 확인**: 테스트 완료 후 결과 데이터 확인

4. 수동 제어

- **릴레이 제어**: 각 채널별 릴레이 수동 ON/OFF
- **전압 설정**: 실시간 전압값 설정
- **온도 제어**: 챔버 온도 수동 제어

🔧 문제 해결

일반적인 문제들

1. WebSocket 연결 실패

```
문제: 프론트엔드에서 백엔드 서버에 연결할 수 없음
해결: 백엔드 서버가 실행 중인지 확인 (포트 8080)
```

2. 시리얼 포트 접근 오류

```
문제: 장비와 통신할 수 없음
해결:
```

```
- 올바른 COM 포트 선택
- 장비 전원 확인
- 케이블 연결 상태 확인
```

3. 권한 오류 (Linux/Mac)

```
# 시리얼 포트 권한 설정
sudo chmod 666 /dev/ttyUSB0
```

로그 확인

백엔드 서버 콘솔에서 실시간 로그를 확인할 수 있습니다:

- WebSocket 연결 상태
- 시리얼 통신 상태
- 테스트 프로세스 진행 상황
- 오류 메시지

📝 개발 가이드

새로운 기능 추가

1. 백엔드: `server/` 디렉토리에 새 모듈 추가
2. 프론트엔드: `nextjs/components/` 디렉토리에 새 컴포넌트 추가
3. WebSocket 메시지 처리: `backend-websocket-server.js`에 메시지 핸들러 추가

설정 파일 수정

모든 설정은 JSON 파일로 저장되며, 서버 재시작 없이 실시간으로 적용됩니다.

📄 라이선스

이 프로젝트는 ISC 라이선스 하에 배포됩니다.

🤝 기여하기

버그 리포트나 기능 요청은 이슈 트래커를 통해 제출해 주세요.

주의사항: 이 시스템은 산업용 장비 제어용으로 설계되었습니다. 사용 전에 모든 안전 규정을 준수하고, 적절한 훈련을 받은 후 사용하시기 바랍니다.