Raspberry Pi 5 터치 허브 개발기

Behelit
2025년 9월 20일 00:47

Raspberry Pi 5 터치 허브 개발기

개발 동기

Raspberry Pi 5(8GB)와 7인치 터치스크린으로 실용적인 스마트 대시보드를 만들었습니다. 시판의 스마트 디스플레이도 있지만, 자신의 요구에 완벽하게 맞춘 시스템을 직접 만들고 싶었다.

시스템 구성

하드웨어

- Raspberry Pi 5(8GB RAM)
- 7인치 터치스크린(800×480)
- CSI 카메라(적외선 대응)
- 각종 센서(온습도, 가스 등)
- **3D 프린트제 커스텀 프레임** (카본 파이버풍 마무리)

소프트웨어 구성

- **백엔드** : Python FastAPI
- **프런트 엔드** : 순수한 HTML/CSS/JavaScript
- **카메라 처리** : OpenCV
- **시스템 모니터링** : 시스템 모니터링 도구 연동
- **날씨 정보** : OpenWeatherMap API

하드웨어 설계 및 사용자 정의 프레임

3D 프린트 프레임 설계

실용성과 미관을 양립시키기 위해 3D 프린터로 전용 프레임을 설계·제작했다. 카본 파이버풍의 마무리로, 프로페셔널한 외관을 실현.

디자인 포인트:

- **방열 설계** : Raspberry Pi 5의 발열에 대응하는 통기성 중시의 구조
- **케이블 관리** : 각종 케이블을 스마트하게 수납하는 배선 경로
- **확장성** : 향후 센서 추가를 지원할 수 있는 모듈식 설계
- **조작성** : 터치스크린에 대한 액세스를 방해하지 않는 최적의 각도

```
# フレーム設計パラメータ（参考値）
FRAME_DIMENSIONS = {
    "width": 125,      # mm
    "height": 85,      # mm
    "depth": 45,       # mm
    "wall_thickness": 2.0,  # mm
    "ventilation_holes": 1.5,  # mm径
}
```

프레임 전면에 다음 요소를 배치:

- **카메라 렌즈 개구부** (중앙)
- **상태 LED 표시창** (상단)
- **센서 환기구** (측면)
- **각종 인디케이터** (코너부)

주요 기능

1. 3페이지 스와이프 UI

작은 화면에서도 사용하기 쉽도록 세 개의 화면을 가로 스 와이프로 전환하도록 설계했습니다.

- **페이지 1** : CSI 카메라 라이브 스트리밍
- **페이지 2** : 날씨 정보 + 시스템 모니터링 상태
- **페이지 3** : 음성 / 텍스트 명령 콘솔

2. 실시간 카메라 스트리밍

CSI 카메라에서 MJPEG 스트리밍 구현. 야간 적외선 촬영에도 대응하고 있다.

```python
@app.get("/cam/csi")
async def csi_camera_stream():
    """CSI camera MJPEG stream"""
    return StreamingResponse(
        mjpeg_generator(csi_cap),
        media_type="multipart/x-mixed-replace; boundary=frame"
    )

def mjpeg_generator(camera):
    """Generate MJPEG frames"""
    while True:
        success, frame = camera.read()
        if not success:
            continue

        _, buffer = cv2.imencode('.jpg', frame,
                                 [cv2.IMWRITE_JPEG_QUALITY, 80])
        frame_bytes = buffer.tobytes()

        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
```

3. 지능형 음성 명령 처리

자연 언어로 명령을 구문 분석하고 적절한 작업을 수행합니다.

```python
@app.post("/assistant/command")
async def process_command(request: CommandRequest):
    """Process voice/text commands"""
    text = request.text.lower().strip()

    # 天気コマンド
    if any(word in text for word in ["天気", "気温", "weather"]):
        weather_data = await get_weather_data()
        return {"type": "weather", "data": weather_data}

    # YouTube検索
    if "youtube" in text or "ユーチューブ" in text:
        query = text.replace("youtube", "").replace("ユーチューブ", "").strip()
        search_url = f"https://www.youtube.com/results?search_query={quote(query)}"
        return {"action": "navigate", "url": search_url}

    # ニュース
    if "ニュース" in text or "news" in text:
        return {"action": "navigate", "url": "https://news.google.com"}

    return {"type": "unknown", "message": "コマンドを理解できませんでした"}
```

4. 시스템 감시 연계

시스템 감시 툴과의 제휴로, 네트워크상의 서버나 서비스의 사활 감시를 표시.

```
@app.get("/uptime_summary")
async def get_uptime_summary():
    """Get formatted uptime status"""
    if not MONITORING_URL:
        return {"enabled": False}

    try:
        response = requests.get(f"{MONITORING_URL}/api/status-page/{STATUS_SLUG}",
                                timeout=5)
        data = response.json()

        monitors = []
        for group in data.get("publicGroupList", []):
            for monitor in group.get("monitorList", []):
                monitors.append({
                    "name": monitor.get("name", "Unknown"),
                    "status": monitor.get("status", 0),
                    "ping": monitor.get("ping"),
                    "time": monitor.get("lastReportTime")
                })

        return {"enabled": True, "list": monitors}
    except Exception as e:
        return {"enabled": True, "error": str(e)}
```

5. 반응형 터치 인터페이스

800×480의 소화면에 최적화된 터치 조작을 실장.

```
// 改良されたタッチスワイプ処理
let startX = 0, startY = 0, moved = false;

function onTouchStart(e) {
    if (isTransitioning) return;
    const touch = e.touches[0];
    startX = touch.clientX;
    startY = touch.clientY;
    moved = false;
}

function onTouchMove(e) {
    if (isTransitioning) return;
```

```
        const touch = e.touches[0];
        const deltaX = touch.clientX - startX;
        const deltaY = touch.clientY - startY;

        // 横方向の動きが優勢な場合、縦スクロールを防ぐ
        if (Math.abs(deltaX) > Math.abs(deltaY) && Math.abs(deltaX) > 10) {
            e.preventDefault();
            moved = true;
        }
    }

    function onTouchEnd(e) {
        if (isTransitioning || !moved) return;
        const touch = e.changedTouches[0];
        const deltaX = touch.clientX - startX;

        if (Math.abs(deltaX) > 80) {
            if (deltaX < 0) {
                goPage(currentPage + 1); // 次のページ
            } else {
                goPage(currentPage - 1); // 前のページ
            }
        }
    }
```

기술적 고안

1. 환경 변수에 의한 유연 설정

모든 설정을 환경 변수로 제어할 수 있게 하고, 코드의 재컴파일 없이 동작을 조정 가능하게 했다.

```
# .env設定例
WIDTH = int(os.getenv("WIDTH", "1280"))
HEIGHT = int(os.getenv("HEIGHT", "720"))
WEATHER_API_KEY = os.getenv("WEATHER_API_KEY", "")
MONITORING_URL = os.getenv("MONITORING_URL", "")
CITY_NAME = os.getenv("CITY_NAME", "Tokyo")
```

2. 오프라인 우선 아키텍처

기본 기능은 모두 로컬에서 동작하고 외부 API는 보조적인 역할만. 네트워크가 끊어져도 기본 조작은 계속 가능.

3. 경량 실장

외부 라이브러리에 대한 의존성을 최소화하고 Raspberry Pi 5의 성능을 최대한 활용.

```
// 外部ライブラリなしの純粋なJavaScript実装
const slider = document.getElementById('slider');
let currentPage = 0;

function goPage(pageNum) {
    const maxPage = 2;
    pageNum = Math.max(0, Math.min(maxPage, pageNum));
    currentPage = pageNum;

    slider.style.transform = `translateX(-${currentPage * 100}svw)`;
    updateIndicators();
}
```

MVP 개발 프로세스

처음부터 완벽한 시스템을 목표로 하지 않고 단계적으로 기능을 추가해 갔다.

1단계: 기본 인프라

- FastAPI 서버 구축
- 단일 페이지 HTML 표시
- CSI 카메라 스트리밍
- **3D 프린트 프레임 설계·제작**

2단계: UI 개선

- 3페이지 스와이프 인터페이스
- 터치 조작 최적화
- 반응형 디자인
- **프레임 임베디드 및 케이블 정리**

3단계: 향상된 기능

- 날씨 API 연계

- 시스템 모니터링 연계
- 음성 명령 처리
- 오버레이 네비게이션

공연

- **시작 시간** : 약 15초
- **메모리 사용량** : 유휴 시 800MB, 피크 시 1.5GB
- **응답 시간** : 로컬 처리 시 평균 50ms 이하
- **스트리밍** : 720p @ 15fps 안정 동작

실제 사용감

좋은 점

- 터치 조작이 직관적이고 가족도 쉽게 사용할 수 있습니다.
- 로컬 처리로 응답이 빠름
- 맞춤형 자유도가 높음
- 개인정보 보호
- **3D 프린트 프레임으로 완전히 맞춤화된 외관**
- **컴팩트하고 놓을 장소를 가리지 않는 디자인**

개선점

- 초기 설정에 기술 지식 필요
- UI 디자인을 더욱 세련할 수 있습니다.
- 음성인식 정밀도 향상 여지
- **3D 프린트의 적층 흔적이 눈에 띄는 부분이 있다**
- **프레임 무게로 약간 무거워졌다.**

향후 전망

1. **로컬 LLM 통합** : 보다 자연스러운 음성 대화
2. **IoT 장치 연동** : 스마트 홈 제어

3. **데이터 시각화** : 센서 데이터 그래프 표시
4. **멀티 디바이스 동기화** : 다중 Raspberry Pi 연동

기술 사양

- **플랫폼** : Raspberry Pi OS (64-bit)
- **언어** : Python 3.11 + JavaScript ES2020
- **프레임워크** : FastAPI, OpenCV
- **화면 크기** : 800×480 최적화
- **메모리 요구사항** : 최소 4GB, 권장 8GB

요약

Raspberry Pi 5의 성능 향상으로 실용적인 스마트 홈 허브를 자작하는 것이 현실적으로 되었다. 특히 8GB 메모리는 복수의 서비스를 동시 실행하기에 충분한 여유가 있다.

기제품으로는 실현할 수 없는, 자신만의 기능이나 인터페이스를 실현할 수 있는 것이 자작의 매력이다. MVP부터 시작하여 단계적으로 기능을 추가해 가는 접근법으로 무리없이 실용적인 시스템을 구축할 수 있었다.

이 프로젝트를 통해 Raspberry Pi가 단순한 취미 용도에서 실용적인 일상 도구로 진화한 것을 실감하고 있다. 기술적 장애물은 있지만 그만큼 만족도가 높은 시스템을 구축할 수 있다.

똑같이 Raspberry Pi에서 실용적인 것을 만들고 싶은 분의 참고가 되면 다행이다.