Growatt Monitor

1. ระบบติดตามอุปกรณ์ Growatt - สถาปัตยกรรมระบบ

Growatt Devices Monitoring System Documentation

May 03, 2025

Version 1.0

2. ระบบติดตามอุปกรณ์ Growatt - สถาปัตยกรรมระบบ

2.1 1. ภาพรวมระบบ

ระบบติดตามอุปกรณ์ Growatt เป็นแอปพลิเคชันเว็บที่ออกแบบมาเพื่อติดตามและแสดงข้อมูลจากอุปกรณ์พลังงานแสงอาทิตย์ Growatt ระบบนี้ให้บริการการติดตามแบบเรียลไทม์ การวิเคราะห์ข้อมูลในอดีต เครื่องมือการแสดงผลข้อมูล และการวิเคราะห์เชิงคาดการณ์สำหรับการผลิตพลังงานแสงอาทิตย์ ระบบนี้ใช้สถาปัตยกรรมแอปพลิเคชันเว็บสมัยใหม่ที่มีการแยกส่วนความรับผิดชอบอย่างชัดเจนระหว่างส่วนแบ็กเอนด์ ฟรอนต์เอนด์ และส่วนประมวลผลข้อมูล

2.2 2. องค์ประกอบของสถาปัตยกรรม

2.2.1 2.1 สถาปัตยกรรมแบ็กเอนด์

2.1.1 เฟรมเวิร์กแอปพลิเคชันหลัก

- **เฟรมเวิร์ก Flask**: แอปพลิเคชันนี้สร้างขึ้นโดยใช้ Flask ซึ่งเป็นเฟรมเวิร์กเว็บ Python แบบเบาที่ทำหน้าที่เป็นพื้นฐานสำหรับการจัดการคำขอ HTTP การกำหนดเส้นทาง และการตอบสนอง
- **โครงสร้างบลูพรินท์แบบโมดูลาร์**: แอปพลิเคชันใช้ Flask Blueprints สำหรับการจัดระเบียบเส้นทางแบบโมดูลาร์:
- `main_routes.py` : เส้นทางแอปพลิเคชันหลักสำหรับหน้าเว็บ
- `api_routes.py` : จุดสิ้นสุด API สำหรับการดึงข้อมูล
- `data_routes.py` : จุดสิ้นสุดสำหรับการเก็บรวบรวมและการจัดการข้อมูล
- `prediction_routes.py` : บริการคาดการณ์ที่ใช้ ML

2.1.2 เลเยอร์การเข้าถึงข้อมูล

- **การรวม API**: โมดูล `core/growatt.py` ให้บริการไคลเอ็นต์สำหรับการโต้ตอบกับ API ของ Growatt การจัดการการตรวจสอบสิทธิ์ การดึงข้อมูล และการแปลงข้อมูล
- **เลเยอร์ฐานข้อมูล**: โมดูล `database.py` จัดการการเชื่อมต่อฐานข้อมูล การจัดการสคีมา และการเก็บข้อมูลโดยใช้ SQLite
- **เลเยอร์บริการ**: บริการเช่น `plant_service.py` ทำหน้าที่เป็นตัวกลางระหว่างเส้นทางเว็บและการจัดเก็บข้อมูล ซึ่งใช้ตรรกะทางธุรกิจ

2.1.3 ความสามารถด้าน ML

- **ตัวทำนายพลังงาน**: โมดูล `ml/energy_predictor.py` ใช้ความสามารถในการเรียนรู้ของเครื่องสำหรับการคาดการณ์การผลิตพลังงานโดยอิงจากข้อมูลในอดีต รูปแบบตามฤดูกาล และปัจจัยด้านสิ่งแวดล้อม
- **API การทำนาย**: เปิดให้บริการผ่านจุดสิ้นสุดเฉพาะที่อนุญาตให้ส่วนประกอบฟรอนต์เอนด์เข้าถึงการวิเคราะห์เชิงคาดการณ์

2.2.2 2.2 สถาปัตยกรรมฟรอนต์เอนด์

2.2.1 เฟรมเวิร์ก UI

- **ระบบเทมเพลต**: ใช้ระบบเทมเพลต Jinja2 ของ Flask ด้วยวิธีการแบบองค์ประกอบ ซึ่งจัดระเบียบในไดเรกทอรี `templates/`
- **เลย์เอาต์พื้นฐาน**: `base.html` ให้เปลือกแอปพลิเคชันที่มีองค์ประกอบทั่วไปและพฤติกรรมที่ตอบสนอง
- **องค์ประกอบ**: องค์ประกอบ UI แบบโมดูลาร์ใน `templates/components/` เพื่อการนำกลับมาใช้ใหม่:
- องค์ประกอบทั่วไป (แถบนำทาง ส่วนท้าย)
- องค์ประกอบแดชบอร์ด (การ์ด เมตริก)
- การแสดงผลข้อมูล (แผนภูมิ แผนที่)

2.2.2 สถาปัตยกรรม JavaScript

- **เฟรมเวิร์ก Alpine.js**: เฟรมเวิร์ก JavaScript แบบเบาสำหรับพฤติกรรม UI แบบประกาศโดยไม่ต้องมีขั้นตอนการสร้าง
- **การจัดการองค์ประกอบ**: `component-loader.js` ให้การโหลดองค์ประกอบแบบไดนามิกพร้อมการแคช
- **การเรนเดอร์แผนภูมิ**: การรวมเข้ากับ Chart.js สำหรับการแสดงผลข้อมูลในรูปแบบต่างๆ
- **โมดูลการโต้ตอบ**:
- `map-interaction.js`: จัดการแผนที่แบบโต้ตอบของการติดตั้งพลังงานแสงอาทิตย์
- `alpine-extensions.js`: ส่วนขยายที่กำหนดเองเพื่อเพิ่มฟังก์ชันการทำงานของ Alpine.js
- องค์ประกอบเฉพาะแอป เช่น `plants-app.js` สำหรับการจัดการข้อมูลพืช

2.2.3 การออกแบบที่ตอบสนอง

- **เฟรมเวิร์ก TailwindCSS**: วิธีการ CSS แบบ utility-first สำหรับการจัดแต่งทรงที่สอดคล้องกัน
- **จุดพักที่ตอบสนอง**: กำหนดใน `base.html` และจัดการผ่าน JavaScript สำหรับเลย์เอาต์ที่ปรับเปลี่ยนได้
- **การตรวจจับอุปกรณ์**: วัตถุสถานะที่ตอบสนอง (`window.responsive`) ติดตามลักษณะของอุปกรณ์และเรียกการปรับเลย์เอาต์

2.2.3 2.3 สถาปัตยกรรมการไหลของข้อมูล

2.3.1 การเก็บรวบรวมข้อมูล

- **ตัวเก็บรวบรวมข้อมูล**: `data_collector.py` จัดการการดึงข้อมูลตามกำหนดเวลาจาก API ของ Growatt
- **การรวม Cron**: งานที่กำหนดเวลาไว้สำหรับการซิงโครไนซ์ข้อมูลเป็นประจำ
- **การเก็บรวบรวมด้วยตนเอง**: จุดสิ้นสุด API สำหรับการเก็บรวบรวมข้อมูลตามความต้องการ

2.3.2 การจัดเก็บข้อมูล

- **ฐานข้อมูล SQLite**: การจัดเก็บข้อมูลในเครื่องโดยใช้ SQLite สำหรับข้อมูลอุปกรณ์ สถิติพลังงาน และข้อมูลสภาพอากาศ

- **โครงสร้างตาราง**:
- Plants: ข้อมูลการติดตั้งพลังงานแสงอาทิตย์
- Devices: รายละเอียดอุปกรณ์แต่ละรายการที่เชื่อมโยงกับ Plants
- Energy Stats: สถิติการผลิตในช่วงเวลา
- Weather Data: สภาพแวดล้อม

2.3.3 การแคชข้อมูล

- **การแคชฝั่งเบราว์เซอร์**: ใช้ในส่วนประกอบฟรอนต์เอนด์โดยใช้ localStorage
- **การแคชฝั่งเซิร์ฟเวอร์**: การรวม Flask-Caching สำหรับการตอบสนอง API
- **การทำให้แคชเป็นโมฆะ**: ทั้งการหมดอายุตามเวลาและกลไกการรีเฟรชโดยชัดแจ้ง

2.3 3. คุณสมบัติและการใช้งานที่สำคัญ

2.3.1 3.1 แดชบอร์ดและการติดตาม

- **การอัปเดตแบบเรียลไทม์**: การรีเฟรชข้อมูลแบบไดนามิกสำหรับค่าการผลิตปัจจุบัน
- **ภาพรวมสถานะระบบ**: แสดงเมตริกสำคัญ การแจ้งเตือน และตัวบ่งชี้ประสิทธิภาพ
- **การติดตามสถานะอุปกรณ์**: การตรวจสอบสถานะออนไลน์/ออฟไลน์ของอุปกรณ์ที่เชื่อมต่อ

2.3.2 3.2 การแสดงผลข้อมูล

- **แผนภูมิผลผลิตพลังงาน**: แผนภูมิที่ปรับแต่งได้สำหรับการดูข้อมูลการผลิตในช่วงเวลาต่างๆ
- **การแสดงผลการไหลของพลังงาน**:
  การแสดงผลแบบโต้ตอบของการไหลของพลังงานระหว่างส่วนประกอบ
- **แผนที่พลังงานแสงอาทิตย์ประเทศไทย**: การแสดงผลทางภูมิศาสตร์ของตำแหน่ง Plants พร้อมตัวบ่งชี้สถานะ
- **การวิเคราะห์ประสิทธิภาพ**: แผนภูมิสำหรับประสิทธิภาพ การกระจาย และเมตริกสำคัญอื่นๆ

2.3.3 3.3 การวิเคราะห์เชิงคาดการณ์

- **การคาดการณ์การผลิตพลังงาน**: การคาดการณ์การผลิตพลังงานในอนาคตโดยใช้ ML
- **การปรับตามฤดูกาล**: การคำนึงถึงความแปรปรวนตามฤดูกาลในประมาณการการผลิต
- **การวิเคราะห์อัตราส่วนประสิทธิภาพ**:
  การเปรียบเทียบผลลัพธ์จริงกับผลลัพธ์ทางทฤษฎีสำหรับการประเมินระบบ

2.3.4 3.4 การรวมสภาพอากาศ

- **การเชื่อมโยงข้อมูลสภาพอากาศ**: การรวมข้อมูลสภาพอากาศกับการผลิตพลังงาน
- **การแสดงผลสภาพ**:
  แผนภูมิที่แสดงความสัมพันธ์ระหว่างสภาพอากาศและประสิทธิภาพของระบบ
- **ข้อมูลสภาพอากาศในอดีต**: เก็บไว้พร้อมกับข้อมูลการผลิตเพื่อการวิเคราะห์

2.4 4. การไหลของการโต้ตอบระบบ

2.4.1 4.1 การไหลของการตรวจสอบสิทธิ์

1. ผู้ใช้เข้าถึงแอปพลิเคชัน
2. แอปพลิเคชันตรวจสอบเซสชันที่มีอยู่
3. หากไม่ได้รับการตรวจสอบสิทธิ์ จะมีการแสดงแบบฟอร์มเข้าสู่ระบบ
4. ส่งข้อมูลรับรองไปยัง API ของ Growatt ผ่านจุดสิ้นสุดที่ปลอดภัย
5. เมื่อสำเร็จ เซสชันจะถูกสร้างขึ้นและผู้ใช้จะถูกเปลี่ยนเส้นทางไปยังแดชบอร์ด

2.4.2 4.2 การไหลของการดึงข้อมูล

1. ส่วนประกอบฟรอนต์เอนด์ร้องขอข้อมูลผ่านจุดสิ้นสุด API
2. เส้นทางฝั่งเซิร์ฟเวอร์จัดการคำขอและตรวจสอบการตรวจสอบสิทธิ์
3. เลเยอร์บริการดึงข้อมูลจากฐานข้อมูลในเครื่องหรือ API ภายนอก
4. ข้อมูลถูกจัดรูปแบบและส่งคืนเป็น JSON

5. ส่วนประกอบฟรอนต์เอนด์แสดงการแสดงผลตามข้อมูลที่ได้รับ

2.4.3 4.3 การไหลของการทำนาย

1. ผู้ใช้ร้องขอการทำนายผ่าน UI
2. ฟรอนต์เอนด์ส่งคำขอไปยังจุดสิ้นสุดการทำนาย
3. โมเดล ML ประมวลผลคำขอด้วยพารามิเตอร์ที่เหมาะสม
4. ผลลัพธ์การทำนายส่งคืนไปยังฟรอนต์เอนด์
5. ผลลัพธ์แสดงในแผนภูมิพร้อมตัวบ่งชี้ความมั่นใจ

2.5 5. ความกังวลที่ครอบคลุม

2.5.1 5.1 การออกแบบที่ตอบสนอง

- เลย์เอาต์ที่ปรับเปลี่ยนได้สำหรับอุปกรณ์มือถือ แท็บเล็ต และเดสก์ท็อป
- การเรนเดอร์องค์ประกอบตามจุดพัก
- การโต้ตอบที่เป็นมิตรต่อการสัมผัสสำหรับผู้ใช้มือถือ

2.5.2 5.2 การจัดการข้อผิดพลาด

- การลดระดับอย่างสง่างามเมื่อ API ไม่พร้อมใช้งาน
- ข้อความแสดงข้อผิดพลาดที่เป็นมิตรต่อผู้ใช้
- กลไกการลองใหม่อัตโนมัติพร้อมการถอยกลับแบบเอ็กซ์โพเนนเชียล

2.5.3 5.3 การเพิ่มประสิทธิภาพประสิทธิภาพ

- การแคชข้อมูลในหลายระดับ
- การโหลดองค์ประกอบแบบ Lazy
- คำขอ API ที่มีการควบคุม
- การประมวลผลข้อมูลฝั่งไคลเอ็นต์เมื่อเหมาะสม

2.5.4 5.4 มาตรการรักษาความปลอดภัย

- การไหลของการตรวจสอบสิทธิ์ที่ปลอดภัย
- การจัดการเซสชัน
- การกำหนดค่า CORS
- การล้างผลลัพธ์ข้อผิดพลาดในโปรดักชัน

2.6 6. สถาปัตยกรรมการปรับใช้

2.6.1 6.1 สภาพแวดล้อมการพัฒนา

- เซิร์ฟเวอร์การพัฒนา Flask ในเครื่อง
- การรีโหลดสดสำหรับการทำซ้ำอย่างรวดเร็ว
- ฐานข้อมูลในเครื่องสำหรับการทดสอบ

2.6.2 6.2 การปรับใช้โปรดักชัน

- การทำคอนเทนเนอร์ด้วย Docker และ docker-compose
- Nginx เป็น reverse proxy พร้อมการยกเลิก SSL
- Gunicorn เป็นเซิร์ฟเวอร์ WSGI สำหรับแอปพลิเคชัน Flask
- ปริมาณงานที่คงอยู่สำหรับฐานข้อมูลและบันทึก

2.7 7. การรวมภายนอก

2.7.1 7.1 API ของ Growatt

- กลไกการตรวจสอบสิทธิ์
- จุดสิ้นสุดการดึงข้อมูล
- อินเทอร์เฟซคำสั่งสำหรับการควบคุมอุปกรณ์

2.7.2 7.2 บริการสภาพอากาศ

- การรวมเข้ากับผู้ให้บริการข้อมูลสภาพอากาศ
- การเชื่อมโยงข้อมูลสภาพอากาศในอดีต
- การพยากรณ์สภาพอากาศเพื่อเพิ่มการคาดการณ์

2.8 8. การพิจารณาสถาปัตยกรรมในอนาคต

2.8.1 8.1 การปรับปรุงความสามารถในการปรับขนาด

- การย้ายไปยัง PostgreSQL สำหรับการติดตั้งขนาดใหญ่
- Redis สำหรับการปรับปรุงการแคชและการจัดการเซสชัน
- การแยกส่วนไมโครเซอร์วิสสำหรับส่วนประกอบเฉพาะ

2.8.2 8.2 การขยายคุณสมบัติ

- แอปพลิเคชันมือถือพร้อมการแจ้งเตือนแบบพุช
- การตรวจจับความผิดปกติขั้นสูง
- การรวมเข้ากับระบบสมาร์ทโฮม
- ความสามารถ ML ที่ขยายสำหรับการคาดการณ์การบำรุงรักษา

2.9 9. สรุปเทคโนโลยีที่ใช้

- **แบ็กเอนด์**: Python, Flask, SQLite
- **ฟรอนต์เอนด์**: HTML, Alpine.js, TailwindCSS, Chart.js
- **การแสดงผลข้อมูล**: Chart.js, แผนที่ SVG แบบโต้ตอบ
- **การเรียนรู้ของเครื่อง**: NumPy, โมเดลการทำนายที่กำหนดเอง
- **การปรับใช้**: Docker, Nginx, Gunicorn

© 2025 BORING9.DEV. All rights reserved.