🌬️ ระบบ AI ตรวจวัดคุณภาพอากาศสำหรับ Jetson Nano

ระบบตรวจวัดและพยากรณ์คุณภาพอากาศด้วย AI ที่ทำงานบน NVIDIA Jetson Nano ระบบนี้เก็บข้อมูลเซ็นเซอร์แบบเรียลไทม์ ฝึกโมเดลแมชชีนเลิร์นนิง และให้การพยากรณ์คุณภาพอากาศที่แม่นยำพร้อมแดชบอร์ดเว็บแบบอินเทอร์แอคทีฟ

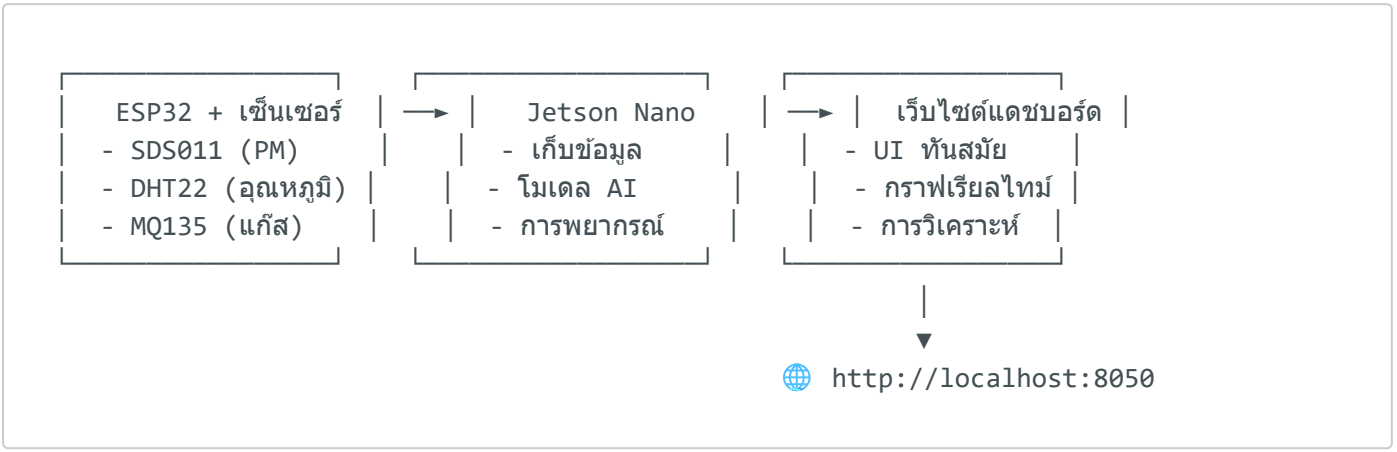
🎯 ภาพรวมโปรเจค

ระบบนี้รวม **เซ็นเซอร์ IoT**, **โมเดล AI/ML**, และ **การแสดงผลแบบเรียลไทม์** เพื่อสร้างโซลูชันการตรวจวัดคุณภาพอากาศที่ครบถ้วน ระบบจะเก็บข้อมูลสิ่งแวดล้อมอัตโนมัติ เรียนรู้รูปแบบ และพยากรณ์สภาพคุณภาพอากาศในอนาคต

ความสามารถหลัก

- 📊 **ตรวจวัดแบบเรียลไทม์**: PM2.5, PM10, อุณหภูมิ, ความชื้น, ระดับแก๊ส
- 🤖 **การพยากรณ์ด้วย AI**: พยากรณ์ 1-6 ชั่วโมงข้างหน้าด้วย LSTM หรือ Random Forest
- 🌐 **เว็บไซต์แดชบอร์ดสวยงาม**: UI ทันสมัยพร้อม real-time updates
- 📈 **กราฟแบบอินเทอร์แอคทีฟ**: แนวโน้ม 24 ชั่วโมง และเมตริกความแม่นยำ
- 🔄 **การเรียนรู้อัตโนมัติ**: โมเดลปรับปรุงตัวเองด้วยการฝึกต่อเนื่อง
- 💾 **การเก็บข้อมูลถาวร**: จัดเก็บ CSV พร้อมการบันทึกครบถ้วน
- 🧪 **โหมดทดสอบ**: เซ็นเซอร์จำลองสำหรับพัฒนาโดยไม่ต้องมีฮาร์ดแวร์
- 📱 **Responsive Design**: ใช้งานได้ทั้งคอมพิวเตอร์ แท็บเล็ต และมือถือ

🏗️ สถาปัตยกรรมระบบ



📋 ข้อกำหนดฮาร์ดแวร์

NVIDIA Jetson Nano

- **บอร์ด**: Jetson Nano Developer Kit (4GB)
- **OS**: Ubuntu 18.04 LTS (JetPack 4.6+)
- **หน่วยเก็บข้อมูล**: microSD card 64GB+ (Class 10)
- **แหล่งจ่ายไฟ**: 5V 4A barrel jack power supply

- **การเชื่อมต่อ**: Ethernet/WiFi, พอร์ต USB

ส่วนประกอบเซ็นเซอร์

ส่วนประกอบ	วัตถุประสงค์	อินเทอร์เฟซ	ความแม่นยำ
SDS011	อนุภาค PM2.5/PM10	UART	±10%/±15%
DHT22	อุณหภูมิ/ความชื้น	Digital	±0.5°C/±2%
MQ135	แก๊สคุณภาพอากาศ	Analog	เชิงคุณภาพ
ESP32	ตัวควบคุมเซ็นเซอร์	USB-Serial	-

การเชื่อมต่อสายไฟ

```
การจัดเรียงพิน ESP32:
├── เซ็นเซอร์อนุภาค SDS011
│   ├── RX → GPIO16 (ESP32 TX)
│   └── TX → GPIO17 (ESP32 RX)
├── อุณหภูมิ/ความชื้น DHT22
│   └── Data → GPIO4
├── เซ็นเซอร์แก๊ส MQ135
│   └── Analog → GPIO36 (A0)
└── พลังงานและ USB
    ├── 5V → SDS011 VCC
    ├── 3.3V → DHT22, MQ135 VCC
    └── USB → Jetson Nano
```

🔧 คู่มือการประกอบฮาร์ดแวร์

> 📖 **คู่มือการประกอบแบบละเอียด**: ASSEMBLY_GUIDE.md

สำหรับขั้นตอนการประกอบและเชื่อมต่อฮาร์ดแวร์อย่างละเอียด กรุณาดูที่ไฟล์ ASSEMBLY_GUIDE.md ซึ่งรวมถึง:

- 🔌 แผนผังการเชื่อมต่อแบบละเอียด
- 🛠️ ขั้นตอนการประกอบทีละขั้น
- 🔍 การตรวจสอบและทดสอบ
- 🚨 การแก้ปัญหาที่พบบ่อย
- 📦 Checklist การต่อสาย

🚀 คู่มือการติดตั้งซอฟต์แวร์

ขั้นตอนที่ 1: ตั้งค่า Jetson Nano

```
# 1. อัปเดตแพ็กเกจระบบ
sudo apt update && sudo apt upgrade -y

# 2. ติดตั้งเครื่องมือพัฒนา Python
sudo apt install python3-pip python3-venv python3-dev git -y
sudo apt install libhdf5-serial-dev hdf5-tools libhdf5-dev -y
sudo apt install libatlas-base-dev gfortran -y

# 3. โคลนโปรเจค
git clone <url-ของ-repository>
cd air_quality_ai

# 4. สร้างและเปิดใช้งาน virtual environment
python3 -m venv venv
source venv/bin/activate

# 5. ติดตั้ง dependencies ของ Python
pip install --upgrade pip setuptools wheel
pip install -r requirements.txt
```

ขั้นตอนที่ 2: การเขียนโปรแกรม ESP32

1. **ติดตั้ง Arduino IDE** (เวอร์ชัน 1.8.19+)
2. **เพิ่มการรองรับบอร์ด ESP32**:

   ```
   File → Preferences → Additional Board Manager URLs
   เพิ่ม: https://dl.espressif.com/dl/package_esp32_index.json
   Tools → Board → Boards Manager → ค้นหา "ESP32" → Install
   ```

3. **ติดตั้งไลบรารีที่จำเป็น**:
   - ArduinoJson (โดย Benoit Blanchon)
   - DHT sensor library (โดย Adafruit)
   - EspSoftwareSerial
4. **อัปโหลดโค้ดเซ็นเซอร์**:

```
- เปิด: esp32_sensor_code.ino
- บอร์ด: "ESP32 Dev Module"
- พอร์ต: เลือกพอร์ต ESP32 ของคุณ
- ความเร็วอัปโหลด: 115200
- คลิก Upload
```

ขั้นตอนที่ 3: การประกอบฮาร์ดแวร์

1. **เชื่อมต่อเซ็นเซอร์** ตามแผนภาพการเดินสาย
2. **การเชื่อมต่อพลังงาน**:
   - SDS011: 5V (ต้องการแรงดันสูงกว่า)
   - DHT22, MQ135: 3.3V
3. **การเชื่อมต่อ USB**: ESP32 ไปยัง Jetson Nano
4. **ตรวจสอบ**: ตรวจสอบว่ามี `/dev/ttyUSB0` หรือ `/dev/ttyACM0`

ขั้นตอนที่ 4: การกำหนดค่า

ระบบจะตรวจจับแพลตฟอร์มอัตโนมัติ แต่คุณสามารถปรับแต่งการตั้งค่าใน `config.py`:

```
# พอร์ตอนุกรม (ตรวจจับอัตโนมัติตาม OS)
# Windows: COM3, Linux: /dev/ttyUSB0

# การตั้งค่าการเก็บข้อมูล
DATA_COLLECTION_INTERVAL = 5  # วินาทีระหว่างการอ่านค่า

# การตั้งค่าแดชบอร์ด
DASHBOARD_HOST = '0.0.0.0'     # อนุญาตการเข้าถึงผ่านเครือข่าย
DASHBOARD_PORT = 8050          # พอร์ตเว็บอินเทอร์เฟซ

# การตั้งค่าโมเดล AI
MIN_DATA_FOR_LSTM = 10000      # เปลี่ยนเป็น LSTM เมื่อมีข้อมูลเพียงพอ
SEQUENCE_LENGTH = 60           # ขั้นเวลาสำหรับการพยากรณ์
```

🎮 คำแนะนำการใช้งาน

🌐 เริ่มต้นเว็บไซต์แดชบอร์ด (แนะนำ)

วิธีที่ง่ายที่สุดในการดูผลลัพธ์:

```
# เปิดใช้งาน environment
source venv/bin/activate

# รันเว็บไซต์แดชบอร์ด (เปิดเบราว์เซอร์อัตโนมัติ)
python start_website.py
```

เข้าถึงเว็บไซต์ที่:

- 🏠 **Local**: http://localhost:8050
- 🌐 **Network**: http://[jetson-ip]:8050

เมนูเริ่มต้นด่วน

```
# รันเมนูแบบอินเทอร์แอคทีฟ
python run_system.py
```

ตัวเลือกเมนู:

1. 🔍 **ตรวจสอบระบบ** - ตรวจสอบ dependencies และฮาร์ดแวร์
2. 🚀 **เริ่มระบบเต็มรูปแบบ** - ตรวจวัดครบถ้วนด้วยเซ็นเซอร์จริง
3. 🧪 **โหมดเซ็นเซอร์จำลอง** - ทดสอบโดยไม่ต้องมีฮาร์ดแวร์
4. 📊 **แดชบอร์ดเท่านั้น** - ดูข้อมูลที่มีอยู่
5. 🔧 **ทดสอบส่วนประกอบ** - ทดสอบโมดูลแต่ละตัว
6. 🧹 **การบำรุงรักษา** - ทำความสะอาดข้อมูลและตรวจสุขภาพระบบ

คำสั่งโดยตรง

```
# เว็บไซต์แดชบอร์ดอย่างเดียว (แนะนำ)
python start_website.py

# ระบบเต็มรูปแบบด้วยเซ็นเซอร์จริง
python main.py

# โหมดทดสอบด้วยข้อมูลจำลอง
python main.py --mock

# แดชบอร์ดเท่านั้น (ดูข้อมูลที่มีอยู่)
python main.py --dashboard-only
```

```
# รันงานบำรุงรักษา
python main.py --maintenance
```

การทดสอบส่วนประกอบ

```
# ทดสอบการสื่อสารเซ็นเซอร์
python sensor_interface.py

# ทดสอบการบันทึกข้อมูล
python data_logger.py

# ทดสอบโมเดล AI
python ml_models.py

# ทดสอบระบบพยากรณ์
python prediction_system.py

# รันแดชบอร์ดแยก
python dashboard.py
```

🌐 เว็บไซต์แดชบอร์ด

เข้าถึงเว็บไซต์ที่: **http://localhost:8050** หรือ **http://[jetson-ip]:8050**

🎨 ฟีเจอร์เว็บไซต์

การออกแบบทันสมัย:

- 🎨 **UI สวยงาม**: พื้นหลัง gradient และ glass effect
- 📱 **Responsive Design**: ใช้งานได้ทุกอุปกรณ์
- ⚡ **Real-time Updates**: อัปเดตทุก 30 วินาที
- 🎯 **Interactive Charts**: กราฟแบบโต้ตอบได้

📊 ส่วนหลักของเว็บไซต์

1. 🔄 **System Status**
    - สถานะการเชื่อมต่อเซ็นเซอร์
    - ประเภทโมเดล AI ที่ใช้งาน

- จำนวนข้อมูลที่เก็บรวบรวม

2. 📊 **Current Readings**
   - ค่าเซ็นเซอร์แบบเรียลไทม์ในการ์ดสวยงาม
   - ตัวบ่งชี้สีตามระดับความปลอดภัย
   - การแจ้งเตือนเมื่อเกินค่ามาตรฐาน
3. 📈 **Real-time Trends**
   - กราฟแนวโน้ม 24 ชั่วโมงแบบอินเทอร์แอคทีฟ
   - PM2.5/PM10 พร้อมเส้นเกณฑ์ความปลอดภัย
   - อุณหภูมิและความชื้นในแกนรอง
4. 🔮 **AI Predictions**
   - การพยากรณ์ 1, 3, 6 ชั่วโมงข้างหน้า
   - แสดงประเภทโมเดลที่ใช้
   - ตัวบ่งชี้ความเชื่อมั่น
5. 🎯 **Prediction Accuracy**
   - กราฟความแม่นยำของโมเดล
   - Mean Absolute Error (MAE)
   - ประสิทธิภาพตามช่วงเวลา
6. ⚖️ **Historical vs Predicted**
   - เปรียบเทียบค่าพยากรณ์กับค่าจริง
   - การติดตามประสิทธิภาพโมเดล
   - การวิเคราะห์แนวโน้ม

🤖 ระบบโมเดล AI

การเลือกโมเดลอัจฉริยะ

ระบบจะเลือกโมเดลที่ดีที่สุดอัตโนมัติตามข้อมูลที่มี:

```
# ตรรกะการตัดสินใจ
if จำนวนข้อมูล >= 10,000 and tensorflow_available:
```

```
    โมเดล = "LSTM Neural Network"
    # ดีกว่าสำหรับรูปแบบซับซ้อน แนวโน้มระยะยาว
else:
    โมเดล = "Random Forest"
    # ฝึกเร็วกว่า ดีสำหรับข้อมูลจำกัด
```

LSTM Neural Network (ชุดข้อมูลขนาดใหญ่)

- **สถาปัตยกรรม**: LSTM 2 ชั้นพร้อม dropout regularization
- **อินพุต**: 60 ขั้นเวลา (ช่วง 5 นาที)
- **เอาต์พุต**: การพยากรณ์หลายช่วงเวลา (1ชม, 3ชม, 6ชม)
- **การฝึก**: ฝึกใหม่อัตโนมัติทุก 24 ชั่วโมง
- **คุณสมบัติ**: จับความสัมพันธ์เชิงเวลาที่ซับซ้อน

Random Forest (ชุดข้อมูลขนาดเล็ก)

- **ต้นไม้**: 100 estimators พร้อมความลึกสูงสุด 20
- **คุณสมบัติ**: ลำดับเวลาแบบแบน
- **ความเร็ว**: ฝึกและพยากรณ์เร็ว
- **ความแข็งแกร่ง**: จัดการข้อมูลที่ขาดหายได้ดี

ช่วงการพยากรณ์

- **1 ชั่วโมง**: ความแม่นยำสูง การวางแผนทันที
- **3 ชั่วโมง**: การพยากรณ์ระยะกลาง
- **6 ชั่วโมง**: การพยากรณ์แนวโน้มระยะยาว

📁 โครงสร้างโปรเจค

```
air_quality_ai/
├── 🐍 ไฟล์ Python หลัก
│   ├── main.py                  # ตัวประสานงานแอปพลิเคชันหลัก
│   ├── config.py                # การกำหนดค่าระบบ
│   ├── sensor_interface.py      # ชั้นการสื่อสาร ESP32
│   ├── data_logger.py           # การจัดการข้อมูล CSV
│   ├── ml_models.py             # การใช้งานโมเดล AI
│   ├── prediction_system.py     # เครื่องมือพยากรณ์
│   ├── dashboard.py             # เซิร์ฟเวอร์เว็บแดชบอร์ด
```

```
│   └── run_system.py          # ตัวเปิดแบบอินเทอร์แอคทีฟ
│
├── 🔧 ฮาร์ดแวร์และการกำหนดค่า
│   ├── esp32_sensor_code.ino  # โค้ด Arduino สำหรับ ESP32
│   ├── requirements.txt       # dependencies ของ Python
│   └── README.md              # เอกสารนี้
│
├── 📊 การจัดเก็บข้อมูล
│   ├── data/
│   │   ├── air_quality_data.csv    # การอ่านเซ็นเซอร์ดิบ
│   │   ├── predictions.csv         # ผลการพยากรณ์ AI
│   │   └── accuracy_log.csv        # เมตริกประสิทธิภาพโมเดล
│   │
│   ├── models/                     # โมเดล AI ที่ฝึกแล้ว
│   │   ├── lstm_model.h5          # โมเดล LSTM ของ TensorFlow
│   │   ├── random_forest_model.joblib # โมเดล RF ของ Scikit-learn
│   │   └── scaler.joblib          # การปรับมาตรฐานข้อมูล
│   │
│   └── logs/                       # ไฟล์บันทึกระบบ
```

🔧 คู่มือแก้ปัญหา

ปัญหาทั่วไปและวิธีแก้ไข

1. 🔌 ปัญหาการเชื่อมต่อเซ็นเซอร์

```
# ตรวจสอบการตรวจจับ ESP32
ls /dev/ttyUSB* /dev/ttyACM*

# แก้ไขสิทธิ์
sudo usermod -a -G dialout $USER
sudo chmod 666 /dev/ttyUSB0

# ทดสอบการสื่อสาร
python sensor_interface.py
```

2. 🧠 ปัญหา TensorFlow บน Jetson Nano

```
# ติดตั้ง TensorFlow สำหรับ Jetson
pip3 install --pre --extra-index-url
https://developer.download.nvidia.com/compute/redist/jp/v461 tensorflow

# หากเกิดปัญหาหน่วยความจำ
sudo systemctl disable nvzramconfig
sudo fallocate -l 4G /swapfile
```

```
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

3. 🌐 ปัญหาการเข้าถึงแดชบอร์ด

```
# ตรวจสอบความพร้อมใช้งานพอร์ต
sudo netstat -tlnp | grep 8050

# ทดสอบการเข้าถึงภายใน
curl http://localhost:8050

# ตรวจสอบไฟร์วอลล์ (หากเปิดใช้งาน)
sudo ufw allow 8050
```

4. 💾 การจัดการหน่วยความจำ

```
# ตรวจสอบการใช้หน่วยความจำ
free -h
sudo tegrastats

# ล้างแคชระบบ
sudo sync && sudo sysctl vm.drop_caches=3

# ปรับให้เหมาะกับ Jetson Nano
sudo jetson_clocks  # โหมดประสิทธิภาพสูงสุด
```

การปรับเทียบและบำรุงรักษาเซ็นเซอร์

เซ็นเซอร์อนุภาค SDS011

- **การอุ่นเครื่อง**: 30 วินาทีหลังเปิดไฟ
- **ความแม่นยำ**: ±10% (PM2.5), ±15% (PM10)
- **อายุการใช้งาน**: ~8000 ชั่วโมงการทำงานต่อเนื่อง
- **การบำรุงรักษา**: ทำความสะอาดพัดลมทุก 6 เดือน

DHT22 อุณหภูมิ/ความชื้น

- **ความแม่นยำ**: ±0.5°C, ±2-5% RH
- **การตอบสนอง**: อัปเดตทุก 2 วินาที
- **ช่วง**: -40°C ถึง 80°C, 0-100% RH

- **การปรับเทียบ**: มักปรับเทียบจากโรงงานแล้ว

เซ็นเซอร์แก๊ส MQ135

- **การอุ่นเครื่อง**: 24-48 ชั่วโมงสำหรับการอ่านที่เสถียร
- **ความไว**: CO2, NH3, NOx, แอลกอฮอล์, ควัน
- **การปรับเทียบ**: ค่าพื้นฐานในสภาพแวดล้อมอากาศสะอาด
- **อายุการใช้งาน**: 2-5 ปีขึ้นอยู่กับการใช้งาน

📈 การปรับปรุงประสิทธิภาพ

การปรับปรุงความแม่นยำ

1. **คุณภาพข้อมูล**
   - เก็บข้อมูลอย่างน้อย 1 สัปดาห์ก่อนเชื่อถือการพยากรณ์
   - ตรวจสอบให้แน่ใจว่าเซ็นเซอร์ปรับเทียบอย่างถูกต้อง
   - วางเซ็นเซอร์ห่างจากกระแสลมโดยตรง
2. **การปรับแต่งโมเดล**
   - เพิ่ม `SEQUENCE_LENGTH` เพื่อบริบทมากขึ้น
   - ปรับ `PREDICTION_HORIZONS` ตามความต้องการ
   - ฝึกโมเดลใหม่บ่อยขึ้นในสภาพที่เปลี่ยนแปลง
3. **ปัจจัยสิ่งแวดล้อม**
   - คำนึงถึงการเปลี่ยนแปลงตามฤดูกาล
   - พิจารณารูปแบบสภาพอากาศท้องถิ่น
   - ตรวจสอบการดริฟต์ของเซ็นเซอร์ตามเวลา

ประสิทธิภาพระบบ

1. **การปรับให้เหมาะกับ Jetson Nano**

```
# เปิดใช้งานโหมดประสิทธิภาพสูงสุด
sudo jetson_clocks

# ตรวจสอบการใช้งาน GPU
```

```
sudo tegrastats

# ปรับหน่วยความจำ
echo 'vm.swappiness=10' | sudo tee -a /etc/sysctl.conf
```

2. **ประสิทธิภาพโมเดล**
    - ใช้ Random Forest สำหรับการพยากรณ์ที่เร็วขึ้น
    - ลดความซับซ้อนของ LSTM สำหรับหน่วยความจำจำกัด
    - ใช้ model quantization สำหรับการปรับใช้

🔒 ความปลอดภัยและความเป็นส่วนตัว

การป้องกันข้อมูล

- **การจัดเก็บภายใน**: ข้อมูลทั้งหมดอยู่ใน Jetson Nano
- **ไม่มีคลาวด์**: ไม่มีการส่งข้อมูลภายนอกโดยค่าเริ่มต้น
- **การเข้ารหัส**: พิจารณาเข้ารหัสไฟล์ข้อมูลที่ละเอียดอ่อน
- **การควบคุมการเข้าถึง**: ใช้การยืนยันตัวตนสำหรับการผลิต

ความปลอดภัยเครือข่าย

```
# การกำหนดค่าไฟร์วอลล์
sudo ufw enable
sudo ufw allow 8050/tcp  # การเข้าถึงแดชบอร์ด
sudo ufw deny 22/tcp     # ปิด SSH หากไม่จำเป็น

# เปลี่ยนพอร์ตแดชบอร์ดเริ่มต้น
# แก้ไข config.py: DASHBOARD_PORT = 8051
```

📚 การอ้างอิง API

API อินเทอร์เฟซเซ็นเซอร์

```
from sensor_interface import get_sensor_interface
```

```python
# เริ่มต้นเซ็นเซอร์ (ตรวจจับจำลองกับจริงอัตโนมัติ)
sensor = get_sensor_interface(mock=False)

# เชื่อมต่อและอ่านข้อมูล
if sensor.connect():
    data = sensor.read_sensor_data()
    # คืนค่า: {
    #   'timestamp': '2024-01-01T12:00:00',
    #   'pm25': 12.5, 'pm10': 18.3,
    #   'temperature': 25.4, 'humidity': 65.2,
    #   'gas_level': 150
    # }
```

API ตัวบันทึกข้อมูล

```python
from data_logger import DataLogger

logger = DataLogger()

# บันทึกข้อมูลเซ็นเซอร์
success = logger.log_data(sensor_data)

# ดึงข้อมูล
recent_data = logger.get_latest_data(n_rows=100)
date_range_data = logger.get_data_range('2024-01-01', '2024-01-02')
stats = logger.get_statistics()
```

API ระบบพยากรณ์

```python
from prediction_system import PredictionSystem

predictor = PredictionSystem()

# เริ่มบริการพยากรณ์อัตโนมัติ
predictor.start_prediction_service()

# การพยากรณ์ด้วยตนเอง
predictions = predictor.force_prediction()
# คืนค่า: {
#   '1h': {'pm25': 15.2, 'pm10': 22.1, ...},
#   '3h': {'pm25': 18.5, 'pm10': 25.3, ...},
#   '6h': {'pm25': 21.1, 'pm10': 28.7, ...}
# }

# รับเมตริกความแม่นยำ
accuracy = predictor.get_accuracy_summary(days=7)
```

การบำรุงรักษาและอัปเดต

ตารางการบำรุงรักษาปกติ

รายวัน:

- ตรวจสอบแดชบอร์ดหาความผิดปกติ
- ตรวจสอบบันทึกระบบหาข้อผิดพลาด

รายสัปดาห์:

- ทำความสะอาดตัวเซ็นเซอร์
- ตรวจสอบความต่อเนื่องของการเก็บข้อมูล
- ทบทวนความแม่นยำการพยากรณ์

รายเดือน:

- ปรับเทียบเซ็นเซอร์หากจำเป็น
- อัปเดต dependencies ซอฟต์แวร์
- สำรองข้อมูลไฟล์
- วิเคราะห์แนวโน้มระยะยาว

รายไตรมาส:

- ทำความสะอาดเซ็นเซอร์อย่างละเอียด
- ทบทวนและปรับพารามิเตอร์โมเดล
- ปรับประสิทธิภาพระบบ
- ตรวจสอบฮาร์ดแวร์

การอัปเดตระบบ

```
# อัปเดตแพ็กเกจ Python
source venv/bin/activate
pip install --upgrade -r requirements.txt

# อัปเดตแพ็กเกจระบบ
sudo apt update && sudo apt upgrade -y

# สำรองก่อนอัปเดต
cp -r data/ "data_backup_$(date +%Y%m%d_%H%M%S)/"
cp -r models/ "models_backup_$(date +%Y%m%d_%H%M%S)/"
```

```
# ทดสอบหลังอัปเดต
python run_system.py  # เลือกตัวเลือก 1 สำหรับตรวจสอบระบบ
```

🤝 การมีส่วนร่วม

เรายินดีรับการมีส่วนร่วม! วิธีเริ่มต้น:

1. **Fork** repository
2. **สร้าง** feature branch: `git checkout -b feature/คุณสมบัติเจ๋งๆ`
3. **ทำ** การเปลี่ยนแปลงพร้อมเอกสารที่เหมาะสม
4. **ทดสอบ** อย่างละเอียดบน Jetson Nano
5. **Commit** ด้วยข้อความที่ชัดเจน: `git commit -m 'เพิ่มคุณสมบัติเจ๋งๆ'`
6. **Push** ไปยัง branch: `git push origin feature/คุณสมบัติเจ๋งๆ`
7. **ส่ง** Pull Request

แนวทางการพัฒนา

- ปฏิบัติตามคู่มือสไตล์ PEP 8 Python
- เพิ่ม docstrings ให้ทุกฟังก์ชัน
- รวมการจัดการข้อผิดพลาดและการบันทึก
- ทดสอบทั้งเซ็นเซอร์จำลองและจริง
- อัปเดตเอกสารสำหรับคุณสมบัติใหม่

📄 ใบอนุญาต

โปรเจคนี้ได้รับอนุญาตภายใต้ MIT License - ดูไฟล์ LICENSE สำหรับรายละเอียด

🆘 การสนับสนุนและชุมชน

การขอความช่วยเหลือ

1. **ตรวจสอบเอกสาร**: ทบทวน README นี้และความคิดเห็นในโค้ด
2. **บันทึกระบบ**: ตรวจสอบไดเรกทอรี `logs/` สำหรับรายละเอียดข้อผิดพลาด

3. **ฮาร์ดแวร์**: ตรวจสอบการเชื่อมต่อและแหล่งจ่ายไฟทั้งหมด
4. **GitHub Issues**: สร้างรายงานปัญหาที่ละเอียด
5. **ชุมชน**: เข้าร่วมการอภิปรายและแบ่งปันประสบการณ์

การรายงานปัญหา

เมื่อรายงานปัญหา โปรดรวม:

- รุ่น Jetson Nano และเวอร์ชัน JetPack
- เวอร์ชัน Python และแพ็กเกจที่ติดตั้ง
- ข้อความข้อผิดพลาดและบันทึกที่สมบูรณ์
- ขั้นตอนในการทำซ้ำปัญหา
- รายละเอียดการกำหนดค่าฮาร์ดแวร์

🌟 กิตติกรรมประกาศ

- **NVIDIA** สำหรับแพลตฟอร์ม Jetson Nano และการสนับสนุน CUDA
- **ทีม TensorFlow** สำหรับความเข้ากันได้ ARM64
- **ชุมชน Arduino** สำหรับไลบรารี ESP32
- **ผู้ผลิตเซ็นเซอร์** สำหรับเอกสารที่ละเอียด
- **ผู้มีส่วนร่วมโอเพนซอร์ส** ที่ทำให้สิ่งนี้เป็นไปได้

🎯 การอ้างอิงด่วน

คำสั่งที่จำเป็น

```
# เริ่มระบบ
python run_system.py

# โหมดทดสอบ
python main.py --mock

# แดชบอร์ดเท่านั้น
python main.py --dashboard-only

# ตรวจสอบระบบ
python run_system.py  # ตัวเลือก 1
```

```
# ดูบันทึก
tail -f logs/*.log
```

📁 ไฟล์สำคัญ

- `start_website.py` - เริ่มเว็บไซต์แดชบอร์ด
- `ASSEMBLY_GUIDE.md` - คู่มือการประกอบฮาร์ดแวร์
- `data/air_quality_data.csv` - ข้อมูลเซ็นเซอร์ดิบ
- `data/predictions.csv` - การพยากรณ์ AI
- `logs/` - บันทึกระบบ
- `config.py` - การตั้งค่าระบบ

🌐 URL เว็บไซต์

- **Local**: http://localhost:8050
- **Network**: http://[jetson-ip]:8050

🚀 Quick Start

```
# วิธีเร็วที่สุด - รันเว็บไซต์
python start_website.py

# หรือทดสอบด้วยข้อมูลจำลอง
python main.py --mock
```

🌏 ขอให้การตรวจวัดเป็นไปด้วยดี! สร้างด้วย ❤️ เพื่ออากาศที่สะอาดและสุขภาพที่ดีขึ้น

ระบบนี้ช่วยให้คุณเข้าใจและพยากรณ์รูปแบบคุณภาพอากาศ มีส่วนร่วมในสภาพแวดล้อมการใช้ชีวิตที่ดีต่อสุขภาพและการตระหนักรู้ด้านสิ่งแวดล้อม