

# Dashboard DHT11 - Real-time IoT Dashboard

---

Dashboard web real-time untuk monitoring sensor DHT11 (suhu dan kelembapan) menggunakan MQTT dan WebSocket.

## Fitur

- **Real-time Updates via MQTT & WebSocket:** Data sensor dikirim langsung dari MQTT broker ke frontend tanpa polling REST API
- **Kontrol LED:** Kontrol LED secara real-time via MQTT
- **Grafik Real-time:** Visualisasi suhu dan kelembapan menggunakan Chart.js
- **Fallback Mode:** Simulator sensor jika MQTT broker tidak tersedia

## Persyaratan

- Python 3.7+
- MQTT Broker (misalnya Mosquitto) - *opsional, akan menggunakan simulator jika tidak ada*

## Instalasi

1. Install dependencies:

```
pip install -r requirements.txt
```

2. (Opsional) Install dan jalankan MQTT broker jika belum ada:

```
# Ubuntu/Debian
sudo apt-get install mosquitto mosquitto-clients

# atau menggunakan Docker
docker run -d -p 1883:1883 eclipse-mosquitto
```

## Cara Menggunakan

### Mode 1: Dengan MQTT Broker

1. Pastikan MQTT broker berjalan di **localhost:1883** (atau sesuaikan konfigurasi)
2. Jalankan dashboard:

```
python dashboard.py
```

3. Buka browser ke `http://localhost:5000`
4. Kirim data sensor dari perangkat IoT ke topik MQTT:
  - **Topik sensor:** `sensors/dht`
  - **Format JSON:** `{"temperature": 25.5, "humidity": 60.0}`
  - **Format CSV:** `25.5,60.0`
5. Kirim status LED dari perangkat:
  - **Topik LED status:** `sensors/led/state`
  - **Payload:** `ON` / `OFF` atau `1` / `0` atau JSON `{"led": true}`
6. Dashboard akan otomatis menerima dan menampilkan data secara real-time via WebSocket!

## Mode 2: Tanpa MQTT (Simulator)

Jika MQTT broker tidak tersedia, dashboard akan otomatis menggunakan simulator:

```
# Set environment variable untuk disable MQTT
USE_MQTT=0 python dashboard.py
```

## Konfigurasi MQTT

Sesuaikan konfigurasi MQTT menggunakan environment variables:

```
MQTT_BROKER=192.168.1.100 \
MQTT_PORT=1883 \
MQTT_TOPIC_SENSOR=sensors/dht \
MQTT_TOPIC_LED_STATE=sensors/led/state \
MQTT_TOPIC_LED_CMD=sensors/led/cmd \
python dashboard.py
```

PROF



## Arsitektur

Alur Data Real-time:

```
[Perangkat IoT/DHT11]
  ↓ publish
[MQTT Broker]
  ↓ subscribe
[Dashboard Backend (Flask-SocketIO)]
  ↓ emit via WebSocket
[Browser Frontend (Socket.IO Client)]
  ↓ update UI
[Tampilan Real-time + Chart]
```

## Topik MQTT:

- `sensors/dht` - Sensor mengirim data suhu & kelembapan
- `sensors/led/state` - Sensor mengirim status LED saat ini
- `sensors/led/cmd` - Dashboard mengirim perintah ON/OFF ke LED

## 🔧 Testing dengan Mosquitto

Simulasi data sensor:

```
# Kirim data suhu dan kelembapan (JSON)
mosquitto_pub -h localhost -t sensors/dht -m '{"temperature": 26.5,
"humidity": 65.0}'

# Kirim data format CSV
mosquitto_pub -h localhost -t sensors/dht -m '27.0,70.0'

# Simulasi LED state
mosquitto_pub -h localhost -t sensors/led/state -m 'ON'
```

Subscribe untuk melihat perintah LED dari dashboard:

```
mosquitto_sub -h localhost -t sensors/led/cmd
```

## 📚 Teknologi yang Digunakan

- **Backend:** Flask, Flask-SocketIO, paho-mqtt
- **Frontend:** Socket.IO Client, Chart.js
- **Protocol:** MQTT, WebSocket
- **Real-time:** Tanpa polling, push-based architecture

PROF

## 🔍 Troubleshooting

### MQTT connection failed?

- Pastikan MQTT broker berjalan
- Cek firewall dan port 1883
- Dashboard akan otomatis fallback ke mode simulator

### Data tidak muncul?

- Cek console browser untuk error WebSocket
- Cek terminal untuk log MQTT connection
- Pastikan format data sesuai (JSON atau CSV)



## Catatan

Aplikasi ini menggunakan **WebSocket** untuk komunikasi real-time, bukan polling REST API. Setiap kali data baru diterima dari MQTT broker, data langsung di-push ke semua client yang terhubung.