Application of Servers and Unix-like Systems for Sensor Control in Smart Homes

Presented by Weize Yuan

Supervisor: prof. Ing. Miroslav Husák, CSc.

Czech Technical University in Prague

Faculty of Electrical Engineering

2026

The Smart Home Dilemma: Cloud vs. Local Control

Cloud-based Smart Homes: Inherent Challenges

Privacy Risks

Data transmitted to and stored on vendor servers, vulnerable to breaches.

Vendor Lock-in

Limited to proprietary ecosystems, restricting device choice and interoperability.

Internet Dependency

Local control often relies on cloud connectivity, failing in outages.

Local-first Solution: Empowering Users

100% Local Data Processing

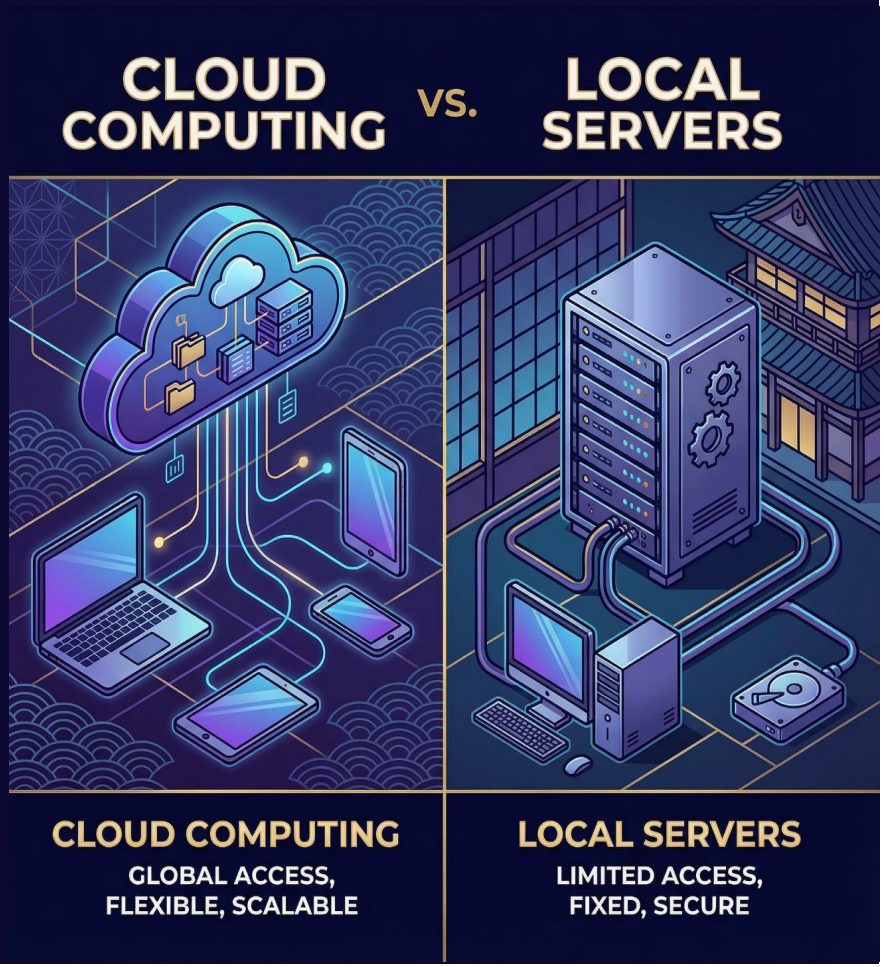
All sensitive information remains within the home network.

Open Standards (MQTT)

Ensuring broad compatibility and future-proofing the system.

Full Customization Control

Users have complete autonomy over system configuration and behavior.



Three-Tiered System Architecture

Our proposed solution is built upon a robust, three-tiered architecture ensuring efficiency, security, and scalability. All processing occurs entirely within the local infrastructure.

Application Layer : Home Assistant OS

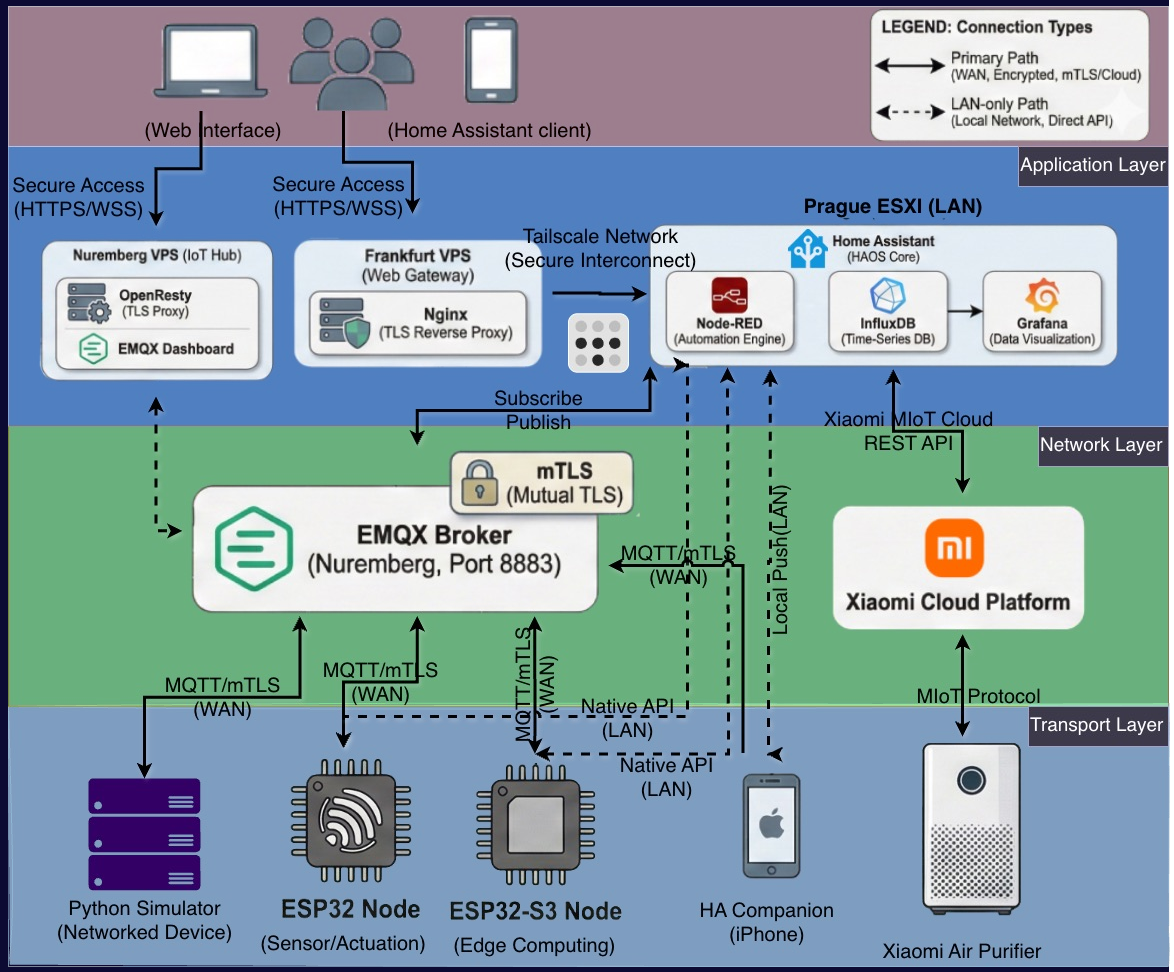
Serves as the central automation engine and provides a user-friendly interface for control and monitoring.

Network Layer : EMQX MQTT Broker

Handles secure message routing with mTLS authentication (ECDSA P-256) and Quality of Service (QoS) guarantees.

Transport Layer : ESP32/ESP32-S3 Sensor Nodes

Edge computing capabilities, such as fall detection, run directly on the device. Supports integration of multiple sensors.



Security Architecture:

Our security design prioritizes robust protection within embedded system constraints.

Cryptographic Choices

- ECDSA P-256 for certificate signatures
- X25519 for secure key exchange
- TLS 1.3 support enabled for enhanced security
- Mutual authentication (client and server verification)

```
esp32:
  board: esp32-s3-devkitc-1
  framework:
    type: esp-idf
    sdkconfig_options:
      # certificate signatures: ECDSA P-256
      CONFIG_MBEDTLS_ECP_DP_SECP256R1_ENABLED: "y"
      CONFIG_MBEDTLS_KEY_EXCHANGE_ECDHE_ECDSA_ENABLED: "y"
      CONFIG_MBEDTLS_ECDSA_DETERMINISTIC: "y"
      # key exchange: X25519
      CONFIG_MBEDTLS_ECP_DP_CURVE25519_ENABLED: "y"
      CONFIG_MBEDTLS_KEY_EXCHANGE_ECDHE_RSA_ENABLED: "y"
      # TLS 1.3 (prioritizes X25519)
      CONFIG_MBEDTLS_SSL_PROTO_TLS1_3: "y"

```

Engineering Rationale

ECDSA P-256

Excellent performance

TLS 1.3

Provides forward secrecy and reduced handshake latency for faster, more secure connections.

Mutual TLS

Each device uses a unique certificate, allowing targeted revocation without system-wide impact.

This design balances security standards with embedded constraints and performance requirements.

System Integration: MQTT Architecture Design

Our MQTT topic design ensures scalability and seamless integration with Home Assistant.

1

homeassistant/{domain}/{device_id}/{config/state}
A standardized topic design pattern for consistent communication.

2

Example Topics

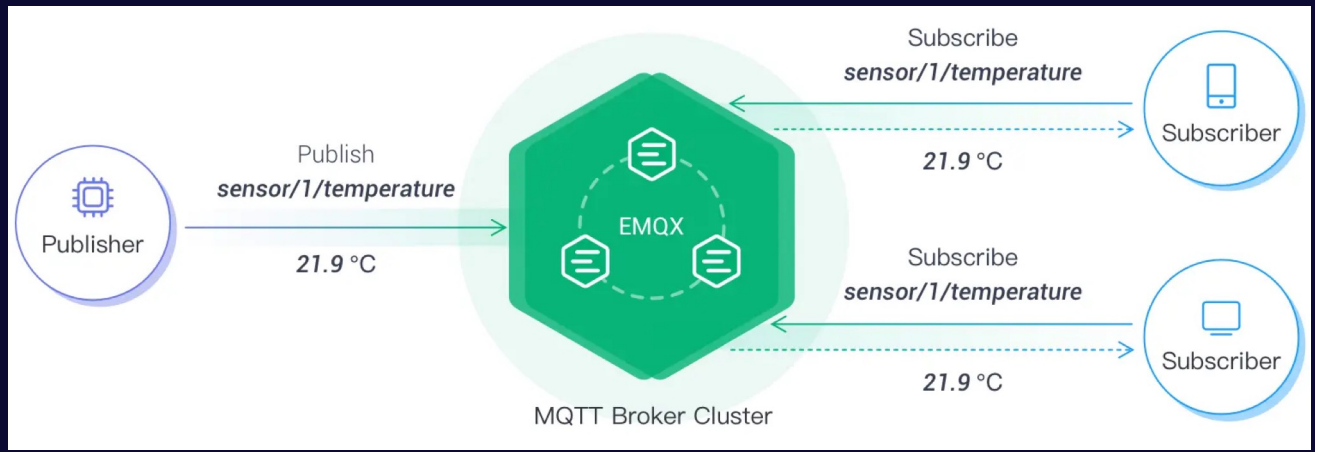
- `homeassistant/sensor/esp32s3_temp/state`
- `homeassistant/binary_sensor/fall_alert/state`

Design Benefits

- **Scalability:** Effortless addition of new devices without schema changes.
- **Home Assistant Auto-Discovery:** Devices publish configuration once, HA creates entities automatically.
- **Namespace Isolation:** Prevents device conflicts.
- **QoS Support:** Critical alerts use QoS 1, sensor data uses QoS 0 for reliability.

My Work

- Designed topic hierarchy for multi-device scaling.
- Implemented HA discovery payloads (JSON schemas).
- Configured QoS policies per message type.



Hardware Integration: ESP32-S3 Sensor Nodes

Our ESP32-S3 nodes integrate various sensors via I2C Bus and ADC for comprehensive environmental monitoring.

Integrated Sensors

- **MPU6050 (0x68):** 6-axis IMU (accelerometer + gyroscope)
- **BMP180 (0x77):** Temperature and atmospheric pressure
- **ADC (GPIO6):** Acoustic sensor (KY-037)
- **MQ-2:** Gas detection (smoke, propane, methane)
- **TCS34725:** RGB color sensing

Edge Computing: Fall Detection

The fall detection algorithm runs locally on the device, ensuring immediate response and data privacy.

- Thresholds: 2.4 G acceleration + 240°/s angular velocity
- Reference: Huynh et al., J. Sensors, 2015
- Sensitivity: 96.3%, Specificity: 96.2%

The firmware, comprising 266 lines within the ESPHome framework, is detailed in `esphome/esp32s3.yaml`.

Fall Detection Algorithm: Edge Implementation

The fall detection algorithm is executed directly on the ESP32-S3, leveraging edge computing benefits.

01

Data Sampling

Accelerometer and gyroscope sampled.

02

Magnitude Calculation

Resultant acceleration magnitude (G-force) and angular velocity magnitude calculated.

03

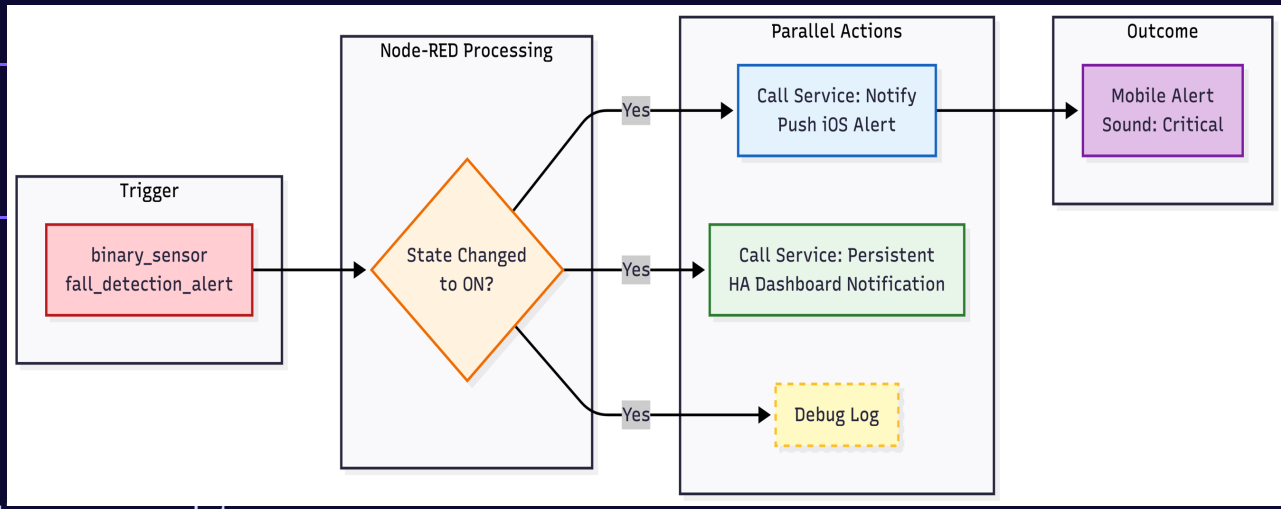
Trigger Condition

Alert triggered if SMV > 2.4 G AND ω > 240 deg/s.

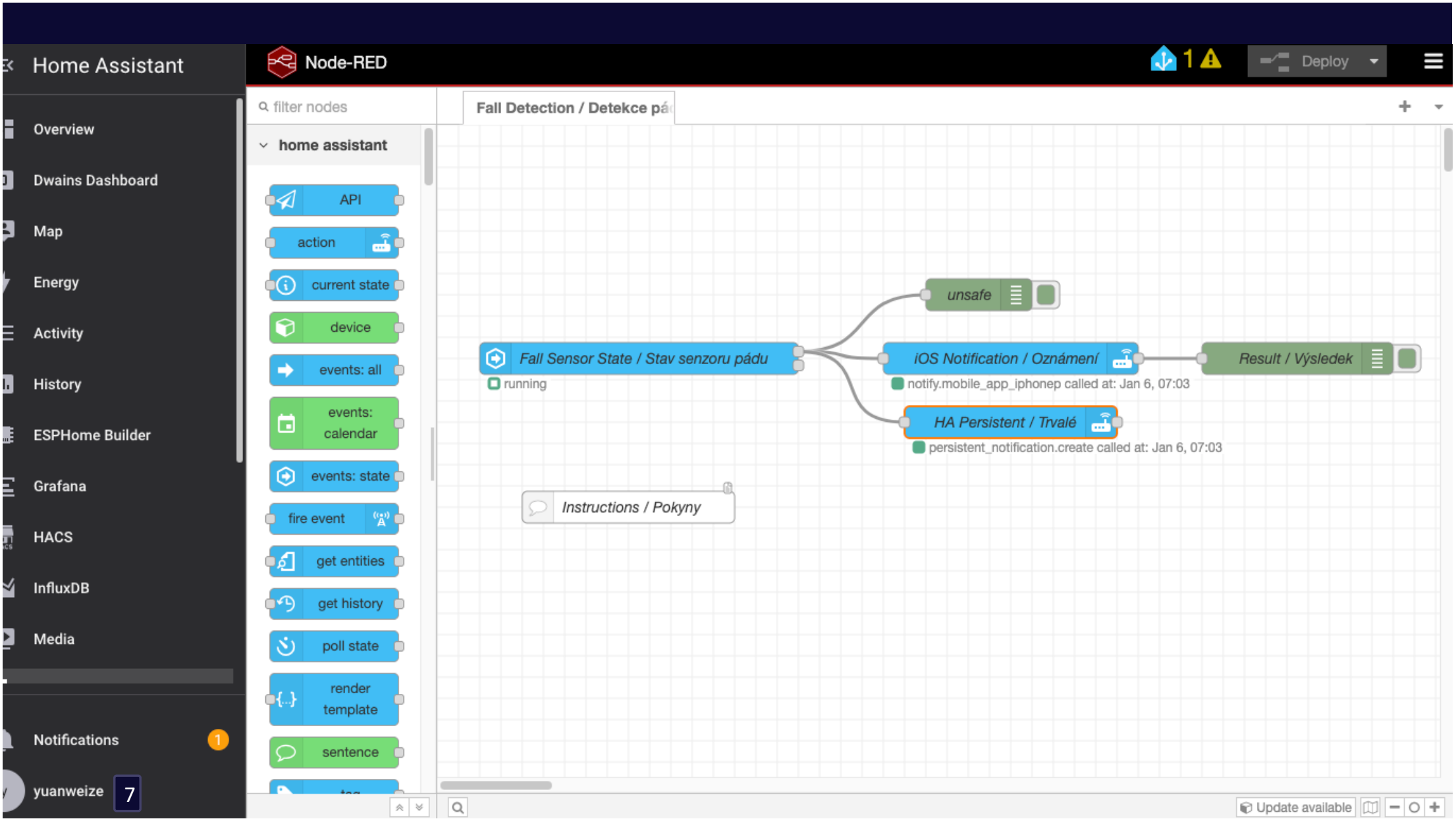
04

Alert Hold

Alert maintained for 5 seconds to ensure notification.



> **Key Point: Processing happens ON DEVICE**
>
> - No cloud latency for critical events.
> - Privacy-preserving: Raw data remains local.
> - Network efficiency: Only alerts are sent, not continuous raw data.


Home Assistant
Overview
Dwains Dashboard
Map
Energy
Activity
History
ESPHome Builder
Grafana
HACS
InfluxDB
Media
Notifications
yuanweize
Node-RED
Deploy
filter nodes
Fall Detection / Detekce pá
home assistant
API
action
current state
device
events: all
events: calendar
events: state
fire event
get entities
get history
poll state
render template
sentence
unsafe
Fall Sensor State / Stav senzoru pádu
running
iOS Notification / Oznámení
notify.mobile_app_iphonep called at: Jan 6, 07:03
Result / Výsledek
HA Persistent / Trvalé
persistent_notification.create called at: Jan 6, 07:03
Instructions / Pokyny
Update available

Scalability Testing

The Python simulator was used to evaluate system scalability by progressively increasing the number of simulated devices. Each test configuration was monitored for approximately 5 minutes to capture steady-state resource utilization.

Test Methodology

- Progressive device scaling from 100 to 1,000 concurrent simulated devices
- 5-minute monitoring period per configuration to capture steady-state metrics
- Monitoring data collected from EMQX Dashboard, broker system metrics, and Home Assistant

Performance Results

Devices	Messages/sec	Broker CPU (%)	HA CPU (%)
100	~229	3.72	1.1
1,000	~1,618	4.03	1.4

Platform Analysis

Comparative Matrix: Control vs. Convenience

Evaluation Criterion	Self-Hosted System	Commercial Platforms (Xiaomi/Alexa/Google)
Data Storage Location	Local Infrastructure(Secure)	Cloud-Dependent (Vulnerable)
Privacy Control	Complete Ownership	Limited User Control
Offline Functionality	Full Operation	Partial/Minimal
Vendor Independence	Open Source(No Lock-in)	Proprietary Ecosystem (Complete Lock-in)
Initial Setup Complexity	Medium Engineering Effort	Low (Plug & Play)

Strategic Trade-off Acknowledgment

While self-hosting requires more initial setup, it guarantees total data ownership and zero vendor lock-in. Unlike commercial platforms, this open-source approach ensures long-term stability regardless of external business changes.

Core Contributions & Future Improvements

Hardware + Firmware Engineering

Developed ESP32/ESP32-S3 firmware, multi-sensor integration, and edge-based fall detection.

Security Architecture Design

Implemented a robust PKI with ECDSA P-256, mTLS, and automated certificate generation.

System Integration Engineering

Designed scalable MQTT topic hierarchy, Home Assistant auto-discovery, and a Python MQTT simulator.

Technical Roadmap

- On-device activity recognition using TFLite Micro.
- 3D print shell
- 5G/ 4G module
- Long-term reliability testing (6+ months uptime).
- Energy optimization with deep sleep modes for battery-powered nodes.

Research Extensions

- Exploring federated learning for privacy-preserving ML.
- Implementing time-series databases for advanced analytics.
- Kubernetes deployment for multi-home management solutions.

Thank You

Questions?

Weize Yuan
yuanweiz@fel.cvut.cz

GitHub: https://github.com/yuanweize/SmartHome_Server/