Project 2: Home Assistant Mock Device

Project 2: Home Assistant Mock Device Integration - Report

Project Overview

This project creates a mock IoT device that integrates with Home Assistant using the MQTT Discovery protocol. The device simulates a weather station with temperature, humidity, and pressure sensors, automatically registering itself in Home Assistant.

Implementation

Mock Device Architecture

The mock device implements the Home Assistant MQTT Discovery specification: 1. Publishes discovery configuration messages to special topics 2. Sends sensor data to state topics 3. Uses random values to simulate real sensor readings

Code Structure

mock_device.py - Main Implementation:

```
import json
import time
import random
import paho.mqtt.client as mqtt
```

```
DEVICE_ID = "mock_sensor_01"
DEVICE_NAME = "Mock Weather Station"
DISCOVERY_PREFIX = "homeassistant"

SENSORS = [
    {
        "name": "Temperature",
        "unique_id": f"{DEVICE_ID}_temp",
        "device_class": "temperature",
        "unit_of_measurement": "°C",
        "state_topic": f"home/{DEVICE_ID}/temperature",
    },
    # ... humidity and pressure sensors
]

def publish_discovery_config(client):
    """Publish MQTT discovery config for Home Assistant"""
    device_info = {
        "identifiers": [DEVICE_ID],
        "name": DEVICE_NAME,
        "model": "Virtual Sensor v1.0",
        "manufacturer": "IoT Lab",
    }

    for sensor in SENSORS:
        config_topic = f"{DISCOVERY_PREFIX}/sensor/
        {sensor['unique_id']}/config"
        config_payload = {
            "name": sensor["name"],
            "unique_id": sensor["unique_id"],
            "device_class": sensor["device_class"],
            "state_topic": sensor["state_topic"],
            "device": device_info,
        }
        client.publish(config_topic, json.dumps(config_payload),
        retain=True)

def generate_sensor_data():
    """Generate random sensor values"""
    return {
        "temperature": round(random.uniform(18.0, 28.0), 1),
        "humidity": round(random.uniform(40.0, 80.0), 1),
        "pressure": round(random.uniform(1000, 1025), 1),
```

```
    }
```

MQTT Discovery Protocol

Discovery Topics

The device publishes configuration to these topics: - `homeassistant/sensor/mock_sensor_01_temp/config` - `homeassistant/sensor/mock_sensor_01_humidity/config` - `homeassistant/sensor/mock_sensor_01_pressure/config`

Configuration Message Format

```
{
  "name": "Temperature",
  "unique_id": "mock_sensor_01_temp",
  "device_class": "temperature",
  "unit_of_measurement": "°C",
  "state_topic": "home/mock_sensor_01/temperature",
  "value_template": "{{ value_json.value }}",
  "device": {
    "identifiers": ["mock_sensor_01"],
    "name": "Mock Weather Station",
    "model": "Virtual Sensor v1.0",
    "manufacturer": "IoT Lab"
  }
}
```

State Topics

Sensor data is published to: - `home/mock_sensor_01/temperature` - `home/mock_sensor_01/humidity` - `home/mock_sensor_01/pressure`

Running and Testing

Test Output

```
python mock_device.py
```

Console Output:

```
Connecting to MQTT broker...
Connected to MQTT broker at localhost:1883
Published discovery config for Temperature
Published discovery config for Humidity
Published discovery config for Pressure

Starting sensor data publishing (Ctrl+C to stop)...
--------------------------------------------------
Published temperature: 25.5
Published humidity: 54.7
Published pressure: 1010.3
--------------------------------------------------
```

MQTT Message Flow

Discovery Messages (received by subscriber):

[homeassistant/sensor/mock_sensor_01_temp/config]

```
{
  "name": "Temperature",
  "unique_id": "mock_sensor_01_temp",
  "device_class": "temperature",
  "unit_of_measurement": "°C",
  "state_topic": "home/mock_sensor_01/temperature",
  "value_template": "{{ value_json.value }}",
  "device": {
    "identifiers": ["mock_sensor_01"],
    "name": "Mock Weather Station",
    "model": "Virtual Sensor v1.0",
    "manufacturer": "IoT Lab"
  }
}
```

Sensor Data Messages:

[home/mock_sensor_01/temperature]

```
{
  "value": 25.5
}
```

[home/mock_sensor_01/humidity]

```
{
```

```
  "value": 54.7
}

[home/mock_sensor_01/pressure]
{
  "value": 1010.3
}
```

The device successfully publishes discovery configurations and sensor data every 5 seconds. When integrated with Home Assistant, it automatically appears as a device with three sensor entities.

Technologies Used

- **Python 3** - Implementation language
- **paho-mqtt** - MQTT client library
- **Eclipse Mosquitto** - MQTT broker (Docker)
- **Home Assistant MQTT Discovery** - Auto-discovery protocol

How It Works

Step 1: Connection

Device connects to MQTT broker at startup.

Step 2: Discovery

Publishes retained configuration messages to `homeassistant/sensor/*/config` topics. Home Assistant subscribes to these topics and automatically creates entities.

Step 3: Data Publishing

Every 5 seconds: - Generates random sensor values - Publishes to state topics in JSON format - Home Assistant updates entity states

Step 4: Device Registration

Home Assistant groups all three sensors under one device using the shared

`device.identifiers` field.

Integration with Home Assistant

When running with Home Assistant: 1. Device appears in Settings → Devices & Services → MQTT 2. Shows as “Mock Weather Station” device 3. Three sensor entities created automatically: - Temperature (°C) - Humidity (%) - Pressure (hPa) 4. Values update in real-time on the dashboard 5. Historical data can be viewed in charts

Conclusion

This project demonstrates: 1. **MQTT Discovery Protocol** - Automatic device registration without manual configuration 2. **IoT Device Simulation** - Mock sensor data generation 3. **Home Assistant Integration** - Proper device and entity structure 4. **Real-time Updates** - Continuous sensor data streaming

Key Learning Points

- MQTT Discovery enables zero-configuration device integration
- Retained messages ensure configuration persists after Home Assistant restarts
- Device grouping improves organization in Home Assistant UI
- JSON value templates allow flexible data parsing
- The pattern can be extended to real hardware sensors (ESP32, Arduino, etc.)

Future Enhancements

- Add control capabilities (switches, buttons)
- Implement availability tracking
- Add device diagnostics (battery, signal strength)
- Support for more sensor types
- Configuration via MQTT commands