ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Ανάκτηση Δεδομένων από

The Things Network

Αναλυτικός Οδηγός για MQTT, HTTP & WebHooks

Μέθοδοι Ανάκτησης Δεδομένων από TTN

MQTT

Real-time
Pub/Sub
Scalable

HTTP API

REST Calls
Polling
Stateless

WebHooks

Push-based
HTTP POST
Custom endpoints

Προτείνεται: MQTT για real-time applications, HTTP API για batch processing

MQTT - Ρυθμίσεις Σύνδεσης

Connection Parameters

- Server: eu1.cloud.thethings.network
- Port: 1883 (plain) / 8883 (TLS)
- Username: [Application_ID]
- Password: [API_Key]
- Client ID: [optional_unique_id]
- Keep Alive: 60 seconds

API Key Setup στο TTN Console:

1. Applications → [Your App] → API keys
2. Add API key → Set name
3. Rights: 'Read application traffic'
4. Create key και save immediately!

MQTT Topic Structure

- `v3/{app_id}/devices/{dev_id}/up`
- `v3/{app_id}/devices/+/up`
- `v3/{app_id}/devices/{dev_id}/down/queued`
- `v3/{app_id}/devices/{dev_id}/join`
- `v3/{app_id}/devices/{dev_id}/location/solved`

MQTT Python Implementation

Installation & Basic Setup:

```
pip install paho-mqtt
import paho.mqtt.client as mqtt
import json
import base64
import struct
from datetime import datetime
```

Connection & Subscription:

```
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to TTN MQTT")
        client.subscribe("v3/+/devices/+/up")
    else:
        print(f"Failed to connect, return code {rc}")

client = mqtt.Client()
client.username_pw_set("YOUR_APP_ID", "YOUR_API_KEY")
client.on_connect = on_connect
client.connect("eu1.cloud.thethings.network", 1883, 60)
```

Message Processing Function:

```
def on_message(client, userdata, msg):
    try:
        data = json.loads(msg.payload.decode())
        payload =
base64.b64decode(data['uplink_message']['frm_pa
yload'])
        device_id =
data['end_device_ids']['device_id']
        print(f"Received data from device:
{device_id}")
    except Exception as e:
        print(f"Error processing message: {e}")
```

TTN HTTP API - REST Integration

Βασικά API Endpoints:

- `GET /api/v3/applications/{app_id}/devices`
- `GET /api/v3/as/applications/{app_id}/devices/{dev_id}/packages/storage`
- `POST /api/v3/as/applications/{app_id}/devices/{dev_id}/down/push`

Authentication Headers:

```
Authorization: Bearer [API_KEY]
Content-Type: application/json
```

Python Requests Example:

```
import requests

headers = {
    'Authorization': 'Bearer YOUR_API_KEY',
    'Content-Type': 'application/json'
}

url =
'https://eu1.cloud.thethings.network/api/v3/as/applicatio
ns/YOUR_APP_ID/packages/storage/uplink_message'
response = requests.get(url, headers=headers,
params={'limit': 10})

if response.status_code == 200:
    data = response.json()
    print("Success:", data)
```

Payload Decoding Strategies

TTN Payload Formatter (JavaScript):

```
function decodeUplink(input) {
  var bytes = input.bytes;

  // Temperature (2 bytes, signed, /100)
  var temp = ((bytes[0] << 8) | bytes[1]);
  if (temp > 32767) temp -= 65536;
  temp = temp / 100.0;

  // Humidity (2 bytes, unsigned, /100)
  var humidity = ((bytes[2] << 8) | bytes[3]) / 100.0;

  return {
    data: {
      temperature: temp,
      humidity: humidity
    }
  };
}
```

Python Client-Side Decoding:

```
import struct

def decode_payload(payload_bytes):
    if len(payload_bytes) < 6:
        raise ValueError("Payload too
short")

    # Unpack big-endian signed 16-bit
integers
    temp_raw, hum_raw, light_raw =
struct.unpack('>hhh', payload_bytes[:6])

    return {
        'temperature': temp_raw / 100.0,
        'humidity': hum_raw / 100.0,
        'light_level': light_raw / 100.0
    }

# Usage
sensor_data = decode_payload(payload_bytes)
print(f"Temperature:
{sensor_data['temperature']} C")
```

Best Practices: Validate payload length - Handle signed/unsigned numbers - Use consistent endianness

Data Storage & Logging

Storage Options Comparison:

CSV Files

Προς: Simple, Human readable
Κατά: Limited scalability
Χρήση: Prototyping, small datasets

SQL Database

Προς: ACID, Complex queries
Κατά: Setup overhead
Χρήση: Production apps

InfluxDB

Προς: Time-series optimized
Κατά: Learning curve
Χρήση: IoT analytics

Data Storage & Logging

CSV Logging Implementation:

```
import csv
import os
from datetime import datetime

def log_to_csv(device_id, sensor_data, signal_quality):
    filename = f"sensor_data_{device_id}.csv"
    file_exists = os.path.isfile(filename)

    with open(filename, 'a', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)

        # Write header if new file
        if not file_exists:
            writer.writerow(['timestamp', 'device_id', 'temperature',
                             'humidity', 'light_level', 'rssi', 'snr'])

        # Write data row
        writer.writerow([
            datetime.now().isoformat(),
            device_id,
            sensor_data['temperature'],
            sensor_data['humidity'],
            sensor_data['light_level'],
            signal_quality['rssi'],
            signal_quality['snr']
        ])
```

Error Handling & Debugging

Συχνά Προβλήματα & Λύσεις:

- Authentication failures → Έλεγχος API key permissions
- Connection timeouts → Verify network connectivity
- Payload decode errors → Validate payload format & length
- Missing messages → Check device transmission intervals
- JSON parse errors → Handle malformed messages gracefully
- Memory leaks → Implement proper resource cleanup

Logging & Debugging Setup:

```
import logging
import sys

# Configure comprehensive logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler('ttn_client.log', encoding='utf-8'),
        logging.StreamHandler(sys.stdout)
    ]
)

logger = logging.getLogger('TTNClient')

def safe_process_message(msg):
    try:
        # Message processing code here
        data = json.loads(msg.payload.decode())
        logger.info(f"Successfully processed message from {data.get
('device_id', 'unknown')}")
    except json.JSONDecodeError as e:
        logger.error(f"JSON decode error: {e}")
    except Exception as e:
        logger.error(f"Unexpected error: {e}")
```