UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Master's Degree in Artificial Intelligence and Data Engineering

DomusCare

IoT-Based Intelligent Home Management System

Group:
Leonardo Ceccarelli
Luca Rotelli

ANNO ACCADEMICO 2024/2025

Contents

Chapter 1

Introduction

Energy consumption and home automation are two of the most pressing areas in which the Internet of Things (IoT) has proven to offer innovative and sustainable solutions. Modern smart homes aim to optimize power usage, increase user comfort, and enable intelligent energy management by integrating sensors, actuators, and cloud services within a low-power wireless network. However, designing a flexible, low-latency, and energy-efficient infrastructure still presents numerous challenges.
This project introduces a CoAP-based Smart Home IoT system designed to intelligently monitor and manage domestic energy consumption by leveraging a network of solar and power sensors, alongside smart plug actuators. The architecture is built using Contiki-NG and RPL Lite to ensure reliability in a Low Power and Lossy Network (LLN), with a border router coordinating traffic between sensor nodes and external systems. Real-time communication is achieved using the Constrained Application Protocol (CoAP), which enables lightweight observation and control of the smart devices.

Figure 1.1: Example of a smart home that uses internet-connected devices to automate and control various aspects of a home, from lighting and temperature to security and entertainment

The system is capable of collecting solar production data and household power consumption, making predictions about the most energy-efficient periods to operate devices. Based on this analysis, smart plug actuators are triggered to optimize energy usage. Sensor data and actuator status updates are forwarded to a cloud application and stored in a MySQL database. Visualization tools like Grafana provide users with insights into their energy footprint. Additionally, a remote control application allows users to issue commands based on real-time needs.

By combining CoAP observation, cloud analytics, and user input, the proposed framework enhances automation, reduces unnecessary energy waste, and fosters sustainable living. This project lays a foundation for scalable, intelligent home automation systems with the potential to adapt to varying household needs and energy availability patterns.

Chapter 2

Architecture

The system architecture is designed around a CoAP (Constrained Application Protocol) network to ensure efficient communication between sensor nodes and the central server. Each sensor node in the Low Power and Lossy Network is equipped with a sensing module that monitors the device's parameters (such as power) which they are attached to. These sensor nodes are programmed to send their readings to a CoAP server, which stores the data, and to an actuator node. The nodes use CoAP observe mechanism to enable the server to receive real-time updates from the sensors. This setup ensures low-power, reliable communication suitable for IoT (Internet of Things) environments. Additionally, the system includes a remote control application to manage the operational state of the devices. CoAP network is deployed using real sensors nRF52840 Dongle. Networks are connected to a border router that allows them external access.

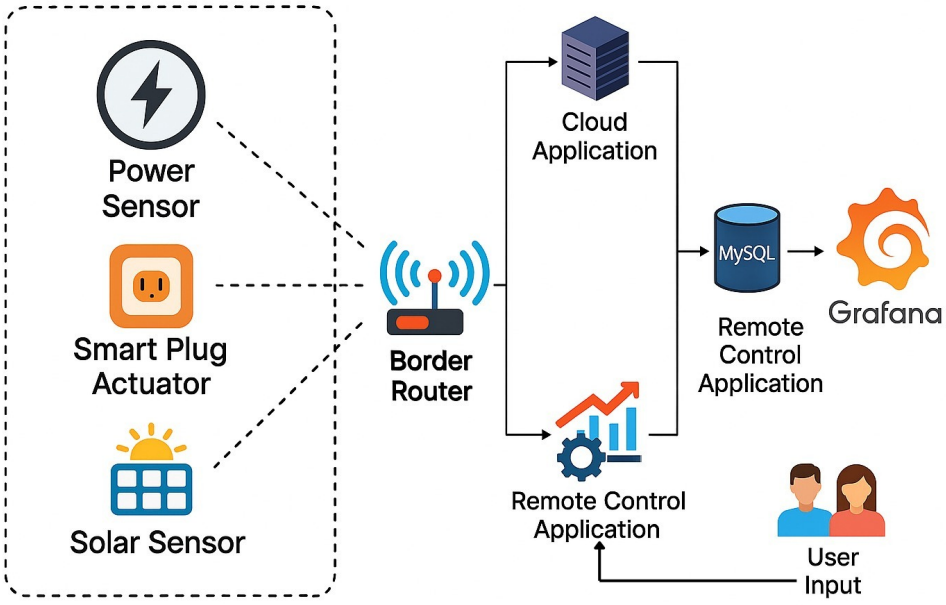


Figure 2.1: System architecture schema

2.0.1 CoAP Network

The smart home deployment consists of multiple sensors and smart actuators, specifically, the system includes the following:

- **Power Sensor:** Measures the energy drawn from the grid by connected home appliances. It enables real-time monitoring of consumption and supports energy optimization strategies.
- **Solar Sensor:** Measures energy production from photovoltaic panels, allowing the system to assess the availability of renewable energy and determine surplus periods.
- **Smart Plug Actuator:** Controls the on/off status of a home appliance. It receives commands based on energy availability and machine learning predictions to optimize activation windows within user-defined constraints.

The CoAP protocol was selected as the application-layer protocol due to the simplicity and limited computational resources of the IoT devices used in the project. Given the low volume and infrequent exchange of data, the lightweight design of CoAP provides an efficient and scalable solution without the need for the complexity introduced by MQTT's broker-based architecture. CoAP also enables direct communication between devices, which fits well with the decentralized nature of our system.

2.0.2 Actuator Node

The smart plug actuator plays a dual role. It locally processes the data received from sensors and integrates predictions from an embedded machine learning model. Based on this information, it autonomously determines the most energy-efficient moment to activate a household device—especially when surplus solar energy is available.
This decision logic supports:

- **Efficient Scheduling:** Using predicted values of future energy consumption and solar production, the system determines the ideal activation time within a predefined user time window.
- **Autonomous Control:** If an optimal surplus energy condition is detected, the actuator activates the smart plug and sends a notification to the cloud.
- **Fallback and Retry:** If no surplus condition is forecasted, the device postpones activation and reevaluates the situation after a fixed interval (e.g., 15 minutes).

2.1 Database

It is essential to store data collected with sensors, to be also able to analyze them through Grafana. The database is composed of three main tables:

2.1.1 data_solar Table

The `data_solar` table contains an entry for every sampling received by the solar panels' sensors. Each entry includes:

- a unique ID;
- the renewable energy input (solarpower attribute);
- the timestamp.

id	solarpower	timestamp
1	177.51	2025-05-24 18:18:36
2	165.45	2025-05-24 18:19:06
3	167.05	2025-05-24 18:19:36
4	161.65	2025-05-24 18:20:06
5	178.7	2025-05-24 18:20:36
6	168.72	2025-05-24 18:21:06
7	169.27	2025-05-24 18:21:36
8	178.68	2025-05-24 18:22:06
9	150.18	2025-05-24 18:22:36
10	162.93	2025-05-24 18:23:06
11	165.52	2025-05-24 18:23:36
12	176.98	2025-05-24 18:24:06
13	185.07	2025-05-24 18:24:36
14	184.39	2025-05-24 18:25:06
15	163.11	2025-05-24 18:25:36
16	174.77	2025-05-24 18:26:06
17	162.68	2025-05-24 18:26:37
18	150.62	2025-05-24 18:27:07
19	189.08	2025-05-24 18:27:37
20	153.34	2025-05-24 18:28:07
21	182.19	2025-05-24 18:28:37
22	192.03	2025-05-24 18:29:07
23	145.83	2025-05-24 18:29:37
24	176.83	2025-05-24 18:30:07
25	167.64	2025-05-24 18:30:37
26	144.98	2025-05-24 18:31:07
27	158.43	2025-05-24 18:31:37
28	168.3	2025-05-24 18:32:07
29	161.48	2025-05-24 18:32:37
30	193.38	2025-05-24 18:33:07
31	149.18	2025-05-24 18:33:37
32	156.54	2025-05-24 18:34:07
33	161.76	2025-05-24 18:34:37
34	177.89	2025-05-24 18:35:07
35	175.71	2025-05-24 18:35:37
36	179.65	2025-05-24 18:36:07
37	180.6	2025-05-24 18:36:37

Figure 2.2: Example rows of the `data_solar` table

2.1.2 data_power Table

The `data_power` table contains an entry for every sampling received by the devices' sensors. Each entry includes:

- a unique ID;
- the overall power consumed by the smart home (power attribute);
- the timestamp.

This table is updated during the registration phase and whenever a node's status is modified via the remote control application.

id	timestamp	power
1	2025-05-24 18:18:41	140.67
2	2025-05-24 18:19:11	174.07
3	2025-05-24 18:19:41	167.53
4	2025-05-24 18:20:11	199.28
5	2025-05-24 18:20:41	156.58
6	2025-05-24 18:21:11	166.07
7	2025-05-24 18:21:41	184.81
8	2025-05-24 18:22:11	145.11
9	2025-05-24 18:22:41	148.47
10	2025-05-24 18:23:11	150.61
11	2025-05-24 18:23:41	146.52
12	2025-05-24 18:24:11	186.78
13	2025-05-24 18:24:41	181.11
14	2025-05-24 18:25:11	196.32
15	2025-05-24 18:25:41	156.67
16	2025-05-24 18:26:11	172.2
17	2025-05-24 18:26:41	170.28
18	2025-05-24 18:27:11	156.44
19	2025-05-24 18:27:41	142.81
20	2025-05-24 18:28:11	162.74
21	2025-05-24 18:28:41	175.91
22	2025-05-24 18:29:11	157.94
23	2025-05-24 18:29:41	148.39
24	2025-05-24 18:30:11	152.3
25	2025-05-24 18:30:42	180.36
26	2025-05-24 18:31:12	161.76
27	2025-05-24 18:31:42	155.94
28	2025-05-24 18:32:12	161.16
29	2025-05-24 18:32:42	140.57
30	2025-05-24 18:33:12	199.85
31	2025-05-24 18:33:42	191.89
32	2025-05-24 18:34:12	204.3
33	2025-05-24 18:34:42	207.71
34	2025-05-24 18:35:12	156.12
35	2025-05-24 18:35:42	175.98
36	2025-05-24 18:36:12	209.08

Figure 2.3: Example rows of the `data_power` table

2.1.3 Dongle Table

The `nodes` table contains an entry for each sensor and actuator. Each entry includes:

- the IP address;
- the type of the node (`actuator` for smart plugs' actuators, `solar` for solar panels' sensors and `power` for the devices' sensors).

ip_address	type
('fd00::f6ce:3616:3304:68e1', 5683)	actuator
('fd00::f6ce:3655:9297:3060', 5683)	actuator
('fd00::f6ce:366c:f0fd:f7e5', 5683)	actuator
('fd00::f6ce:3696:9ac4:1103', 5683)	power
('fd00::f6ce:36f6:9a11:bab3', 5683)	solar

Figure 2.4: Example rows of the `dongle` table

2.1.4 Devices Table

The `nodes` table contains an entry for every smart devices. Each entry includes:

- the IP address;
- the device's name;
- its state (ON, OFF, Pronto);
- its consumption espressed in Kwh;
- the timestamp associated to the device's last activation;
- the duration of the set program of the device.

This table is updated during the registration phase whenever a node's status is modified via the remote control application and by the smart plug when it makes an autonomous decision.

ip_address	nome	stato	consumo_kwh	timestamp_attivazione	durata
('fd00::f6ce:3655:9297:3060', 5683)	Asciugatrice	0	2	2025-05-21 15:03:30	60
('fd00::f6ce:3616:3304:68e1', 5683)	Lavastoviglie	0	2.3	2025-05-21 15:03:07	60
('fd00::f6ce:366c:f0fd:f7e5', 5683)	Lavatrice	0	1.3	2025-05-21 15:03:06	60

Figure 2.5: Example records of the `Devices` table

2.1.5 Data Encoding

In resource-constrained IoT scenarios, the efficiency of the chosen protocol is crucial. Since we do not have constraints that force us to use XML (there are no other devices already deployed to interact with using XML), we opted to use JSON, which offers a simpler and more lightweight syntax compared to XML. As a result, JSON generally yields smaller data payloads since it utilizes concise syntax without the inclusion of verbose tags and attributes found in XML. This characteristic leads to reduced bandwidth usage and reduce the resource used by the sensor for computing the packet. Additionally JSON is human-readable, which makes debugging easier and allows developers to quickly interpret and modify data without complex tools.

Chapter 3

Deployment and Execution

The deployment and operation of the smart home system are based on a CoAP (Constrained Application Protocol) architecture, enabling seamless communication between energy sensors, a smart plug actuator, and a remote control application. The design ensures efficient data exchange, scalability, and low power consumption, making it suitable for embedded IoT environments.

3.0.1 Node Registration

To be reachable by the cloud application and remote controller, each device in the system—whether sensor or actuator—must register with a CoAP Registration Server. Upon startup, each node sends a registration message including its type (e.g., "solar", "power", "actuator"), and a payload containing its IP address and operational status. This information is stored by the server and made accessible to the actuator and external clients.

3.0.2 CoAP Network

Sensors

The system includes two main sensors:

- **Solar Sensor:** Exposes an observable resource at the resource path `/valore` (implemented in `res-solar.c`).
- **Power Sensor:** Exposes an observable resource at the resource path `/valore` (implemented in `res-power.c`).

Each sensor acts as a CoAP server. Data is simulated periodically (the sampling interval can be changed via button press), generating realistic values using a Gaussian distribution based on historical metrics. Some of the payload formats used are:

- Data: `{"t:" "solar" / "power", "value:" valoreCampionato}`
- SensorRegister: `{"t:" "solar" / "power"}`
- ActuatorRegister : `{"t:" "actuator", "n:" nome, "c:" consumo, "d:" durata}`

Smart Plug Actuator

The actuator operates both as a CoAP server and a client:

- **As server:** It exposes a resource `/res-smartplug`, used to publish activation events;
- **As client:** When the corresponding device is set to 'pronto', it requests data from the sensor (whose addresses were obtained during the registration phase). If no surplus condition is forecasted, the device postpones activation, sets a 15-minute timer, and reevaluates the situation once the timer expires.

The actuator, when a device is set to "pronto", calculates the energy that will be produced in the next hour and checks if the energy will be enough to start the device's program. If so, it then activates the device in question. To differentiate the actuators based on the device they are associated with, we used distinct LED colors: blue for the dryer, red for the dishwasher, and yellow for the washing machine.

3.0.3 LEDs and Buttons

In both the `solar.c` and `power.c` files, we implemented an interactive mechanism for adjusting the sensor sampling rate using the onboard button and providing visual feedback through the device's LEDs. Specifically, three different sampling intervals are defined, each associated with a distinct LED color: green, blue, and red. When the user presses the button, the application cycles through the available sampling intervals, updating the current interval and changing the LED color accordingly to reflect the new setting. This approach allows users to intuitively control the data acquisition frequency in real time, directly from the hardware, without requiring any external interface. The logic is implemented using Contiki-NG's event-driven process model, where button events trigger the update of both the sampling interval and the LED state.

3.0.4 Remote Control Application

Figure 3.1: CLI initial display

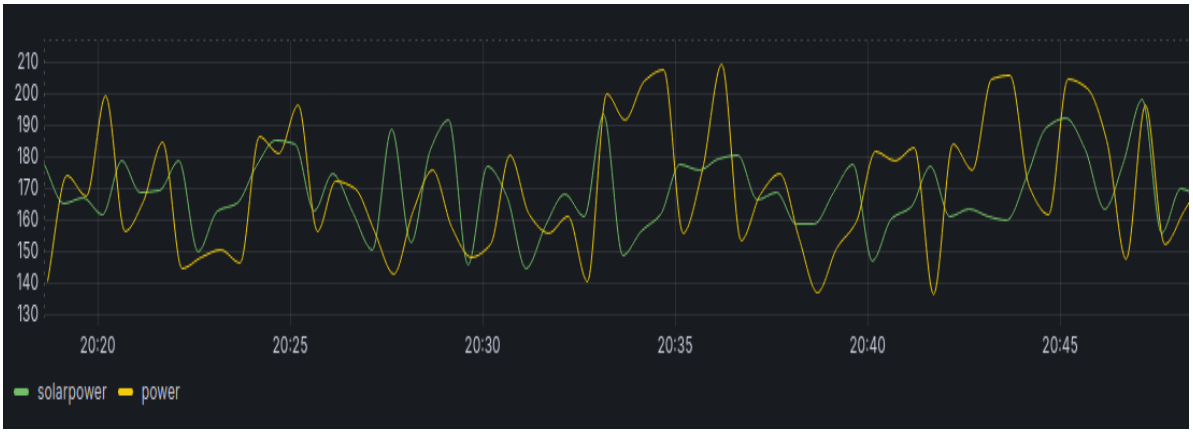
The command-line interface of the remote control application allows users to interact with the system in real-time. It supports the following commands:

- **"Mostra dispositivi":** Show the content of the devices' table;
- **"Cambia stato dispositivo":** Change the status of a specific device into one of the three possible states;
- **"Rimuovi Dispositivo":** Remove a specificic device from the corresponding table;
- **"Esci":** Terminates the session.

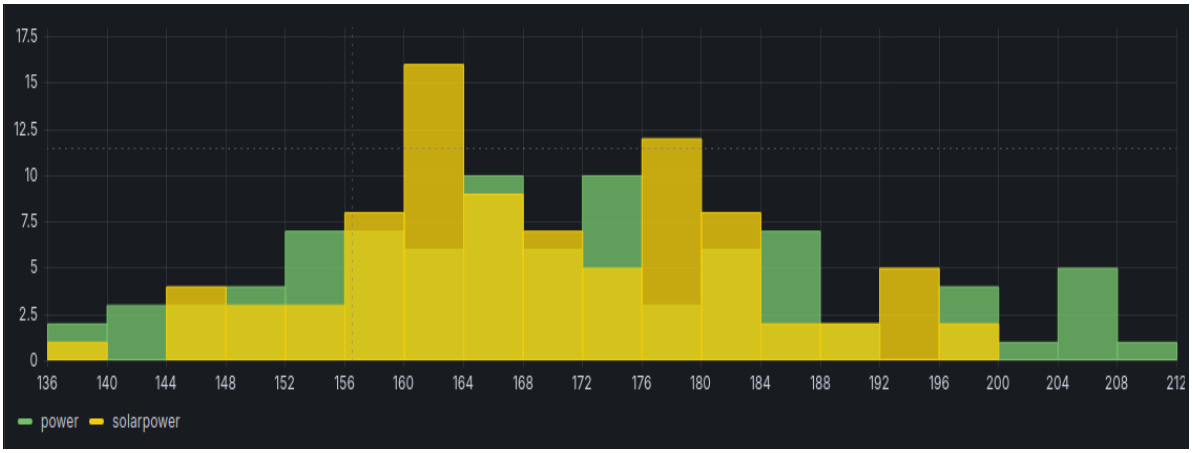
Chapter 4

Grafana

Grafana dashboard is used to compose some analytics based on the values that are stored on the database.



In this section, we compare the most recent values measured by the sensors to see at which time of the day we produce more energy than we consume.



Chapter 5

Machine Learning

The machine learning system integrated into the smart home project provides a foundation for energy-aware decision-making by forecasting future solar power production and power consumption. This approach enhances energy efficiency by allowing the smart plug actuator to schedule its operation during optimal periods, based on predicted surplus energy.
The system utilizes two separate regression models built with TensorFlow's Keras API. The first model predicts the solar power generation in the near future, while the second forecasts household energy consumption. These predictions are essential for enabling the smart plug to make informed decisions regarding when to activate or defer tasks based on energy availability.
The dataset[1] used includes timestamped telemetry data collected from the deployed power and solar sensors.Preprocessing begins by converting timestamp information into time-related features: hour, minute, day, and month. Additionally, two new target columns are generated to represent the future values of both solar power and consumption—each shifted one hour ahead, assuming 15-minute sampling intervals. Any resulting missing values are removed to maintain data integrity.

Each model shares a similar neural architecture:

- An input layer explicitly defined by the dimensionality of the feature set (5 features).
- Two dense layers with ReLU activations, followed by dropout layers to reduce overfitting.
- A single output neuron for continuous value prediction.

The models are compiled with the `mean squared error (MSE)` loss function and the `Adam` optimizer, with `mean absolute error (MAE)` as a key evaluation metric. Training is performed over 50 epochs with an 75/25 split between training and validation data.
To ensure compatibility with the embedded environment of the smart plug (based on nRF52840 Dongle), the trained models are converted into a lightweight, deployable format using the `emlearn` library.
Upon evaluation, both models demonstrated satisfactory performance in terms of MAE on the test set, providing accurate and generalizable predictions for both production and consumption. The dual-model strategy allows the system to forecast both energy supply and demand, creating a smart decision-making framework for energy optimization. This machine learning component is pivotal in reducing energy waste and enabling smart, autonomous behavior within the smart home infrastructure.

[1] Smart Grid Real-Time Load Monitoring Dataset(kaggle.com)

Chapter 6

Use Case

6.1 Smart Home Energy Optimization Use Case

In the context of smart home automation, energy optimization plays a central role in reducing costs and improving energy efficiency. This use case leverages a network of constrained IoT devices to intelligently manage household appliances based on real-time power production and consumption. The system is designed to activate smart devices during optimal energy availability periods, so when solar energy production is sufficient to cover the appliance's energy demand.

6.1.1 Scenario

In a residential smart home environment, the system integrates power sensors and solar sensors to monitor both energy consumption and local energy production. These sensors send data to a central actuator node (Smart Plug), which controls the operation of household appliances. The Smart Plug is responsible for analyzing the incoming data and determining the most efficient time to activate the appliance, such as a washing machine, based on predicted energy surplus.

6.1.2 Implementation

1. **Data Monitoring:**
    - The power sensor continuously measures household energy consumption.
    - The solar sensor monitors the power generated from the home's solar panels.
    - Data is transmitted to the Smart Plug actuator through a CoAP-based communication network.
2. **Energy Prediction and Scheduling:**
    - A machine learning model running locally on the actuator predicts future energy production and consumption.
    - Based on these predictions, the system schedules the activation of the appliance at a time when solar power exceeds consumption.
3. **Intelligent Actuation:**
    - When optimal conditions are met, the Smart Plug activates the appliance autonomously.
    - A timer ensures that the task completes within a defined time limit, respecting user constraints.

6.1.3 Conclusion

By combining sensor data, predictive algorithms, and intelligent actuation, the smart home system enhances energy efficiency and promotes sustainable power usage. This proactive strategy reduces reliance on the grid, lowers electricity bills, and contributes to environmentally conscious energy consumption. The system represents a step forward in home automation, bringing intelligent control and decision-making to everyday appliances.