UNIVERSITY OF PISA

MSc in Computer Engineering

Project for Internet of Things

Smart solar panel and HVAC system

Professors:	Student:
Prof. Giuseppe Anastasi	**Francesco Barcherini (645413)**
Prof.ssa Francesca Righetti	
Prof.ssa Carlo Vallati	

ACADEMIC YEAR 2024/2025

Contents

1 Introduction

The increasing demand for sustainable energy solutions can enhance the integration of renewable sources, such as solar power, into residential and industrial energy infrastructures. Yet, the power generated by photovoltaic panels is inherently variable, being highly dependent on dynamic environmental conditions such as solar irradiation, atmospheric clarity, and panel temperature. As such, to guarantee both energy efficiency and operational continuity, it becomes essential to develop intelligent systems capable of managing energy flows and consumption autonomously, responding to fluctuations in real time. The present work describes an integrated system in which the energy generated by solar panels is routed through relays, either to charge storage batteries via direct current (DC) or to be converted into alternating current (AC) for immediate use by the outer grid or by a connected load—here, specifically, a heating, ventilation, and air conditioning (HVAC) system.

2 Architecture and use-case

2.1 Components

The core of the system architecture revolves around two principal components: an energy management node and an HVAC control node, each equipped with a suite of sensors, actuators, and control logic designed to cooperate under both centralized and distributed regimes. Energy generated by the solar panels is first processed by an inverter, which converts from AC to DC power. The produced energy can be directed to three primary destinations: storage batteries, the public electric grid, or the domestic HVAC load. Decision-making about where the energy is sent is delegated to a set of two programmable relays, whose states are dynamically configured by the control logic based on multiple inputs, including the instantaneous demand from the HVAC system, the level of charge in the battery pack, and the real-time power output of the solar panels. Of course, the best strategy to reduce consumptions and to save money is to exploit firstly the power generated by the solar panel, then the energy stored in the batteries and, only if the two sources are not sufficient for the load demand, the electricity from the grid.

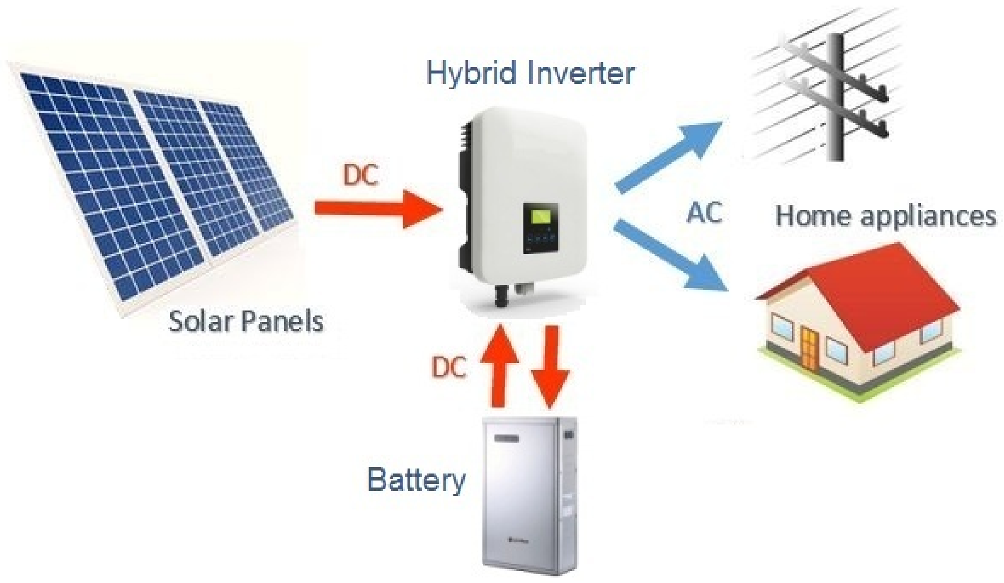


Figure 1: General architecture of a solar panel system with batteries and connection to house load and outer electric grid

2.2 Energy control system

A defining feature of this architecture is its support for environmentally prioritized operation through a so-called “green mode”. In this mode, the system is constrained to use only energy originating from renewable sources—that is, from the battery or directly from the solar panels—excluding grid electricity entirely. This is achieved by delegating to the HVAC node the responsibility of negotiating energy supply with the energy node, which serves both as a provider and a monitor. The HVAC unit, upon determining its operational state and computing the required power to maintain target climate conditions, queries the energy node for availability of renewable energy. If there is sufficient power, the HVAC proceeds with normal operation. If the supply is insufficient, fallback strategies are employed: the system may opt to switch from an energy-intensive cooling mode to a reduced-consumption ventilation mode, or suspend operation entirely, thus ensuring that no non-renewable energy is consumed.

The normal operational mode, in contrast, is governed by centralized decision-making. The energy node collects and processes real-time data to determine the most appropriate energy path. Priority is given to the direct use of solar power, followed by discharging the battery if solar production is inadequate, and finally sourcing from the grid only as a last resort. This decision logic takes into account the HVAC demand, the current battery state-of-charge, and the available instantaneous solar generation. The system is implemented so that, when the solar panel charges the batteries, does not overcharge them above the 90% of the capacity in order to avoid damages.

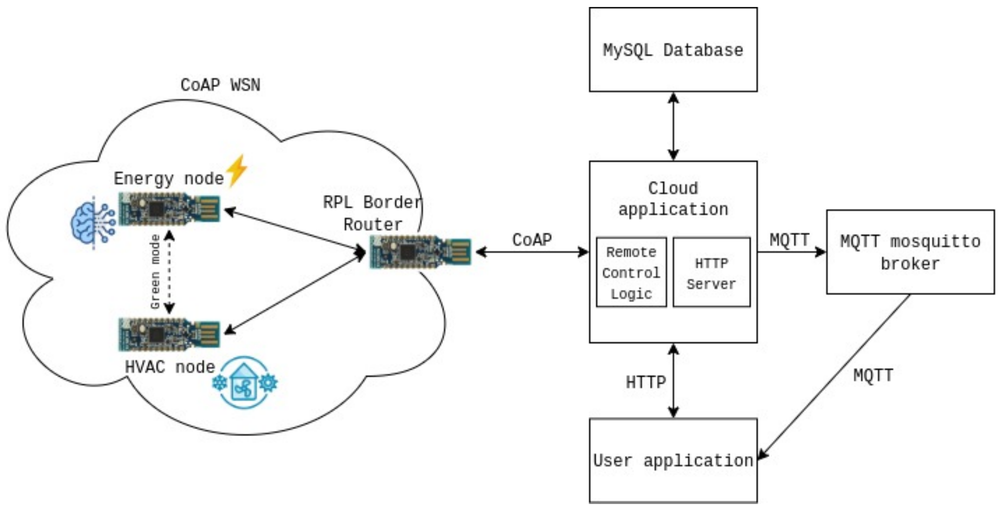


Figure 2: Architecture of the implemented system

2.3 IoT nodes

An additional level of control is introduced through a predictive maintenance and cleaning mechanism for the solar panels. Because surface soiling (such as dust accumulation) can significantly reduce photovoltaic efficiency, the system continuously compares predicted solar generation—computed via a machine learning model that incorporates weather data including irradiation levels, ambient and panel temperatures—with the actual power output measured. If a discrepancy beyond a predefined threshold is observed, an automatic cleaning cycle is initiated via an anti-dust actuator. This preemptive response is designed to restore panel efficiency without human intervention. Should the mismatch persist beyond the cycle, an alarm is triggered, alerting the central control system and the HVAC.

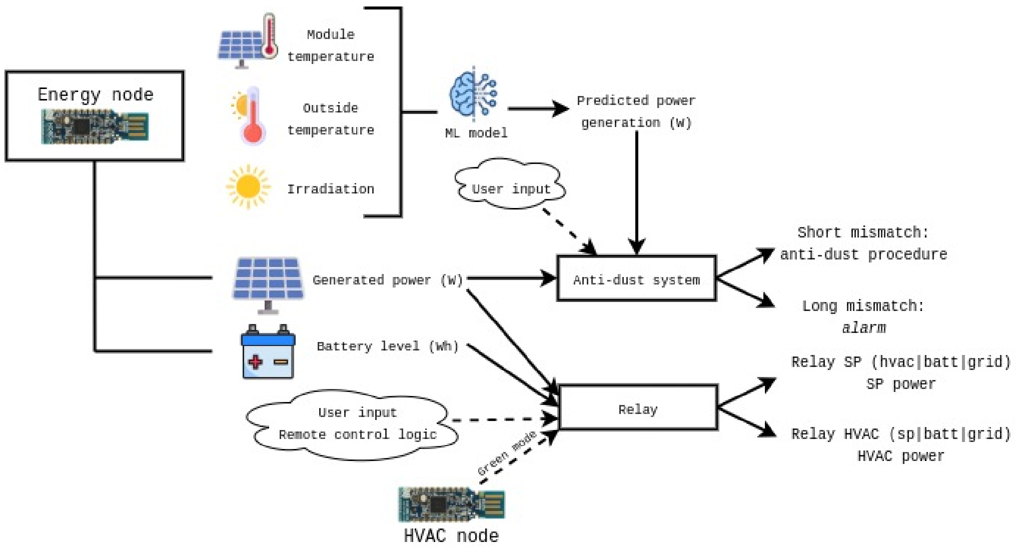


Figure 3: Architecture of the energy node

Sensor integration is extensive and supports the system's adaptability and robustness. The energy node monitors environmental parameters (external temperature, solar irradiation, panel temperature) and internal metrics (actual power output, battery charge level), while also keeping track of power flow through energy counters associated with each relay channel. The HVAC node, instead, focuses on internal climate control, measuring room temperature and adjusting operational parameters such as target temperature, current mode (cooling, heating, ventilation), and power consumption. In green mode, it leverages information received from the energy node (outside temperature and available energy from solar panels and from batteries) to compute the required power for achieving target conditions, based on the room temperature, the outside temperature, the target temperature, the status and the energy available.

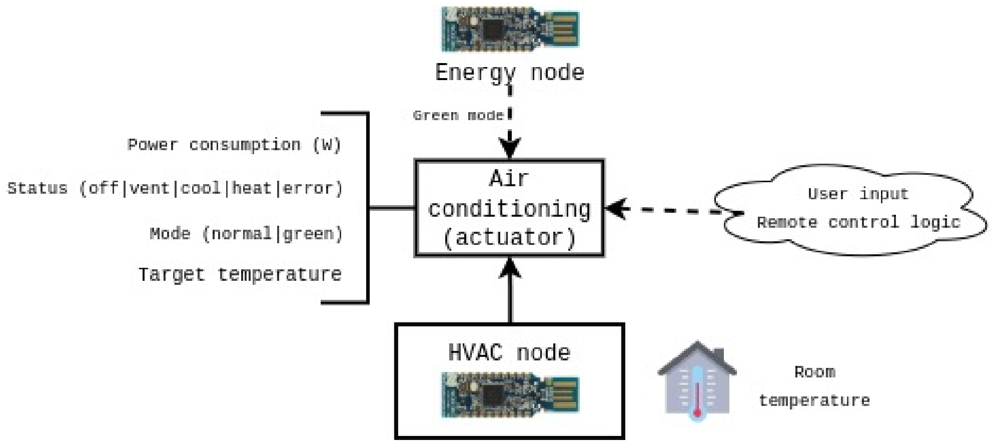


Figure 4: Architecture of the HVAC node

2.4 Cloud infrastructure

The architecture is designed to support both autonomous and user-driven control. While the default behavior relies on embedded intelligence and inter-node communication, manual inputs are facilitated through physical interface buttons that allow local users to reset error states or manually initiate cleaning cycles. Moreover, the HVAC node is capable of self-diagnosis and can raise error codes if it detects operational anomalies or unfulfillable energy requirements.

Beyond the local coordination between the energy node and the HVAC node, the system architecture incorporates a higher-level communication and control framework that enables both remote monitoring and actuation. The core of the former operations lies on an IPv6-enabled CoAP Wireless Sensor Network (WSN), within which the sensor-actuator nodes communicate over potential low-power lossy links. These interactions are orchestrated via an RPL Border Router, which bridges the constrained CoAP network with the cloud infrastructure.

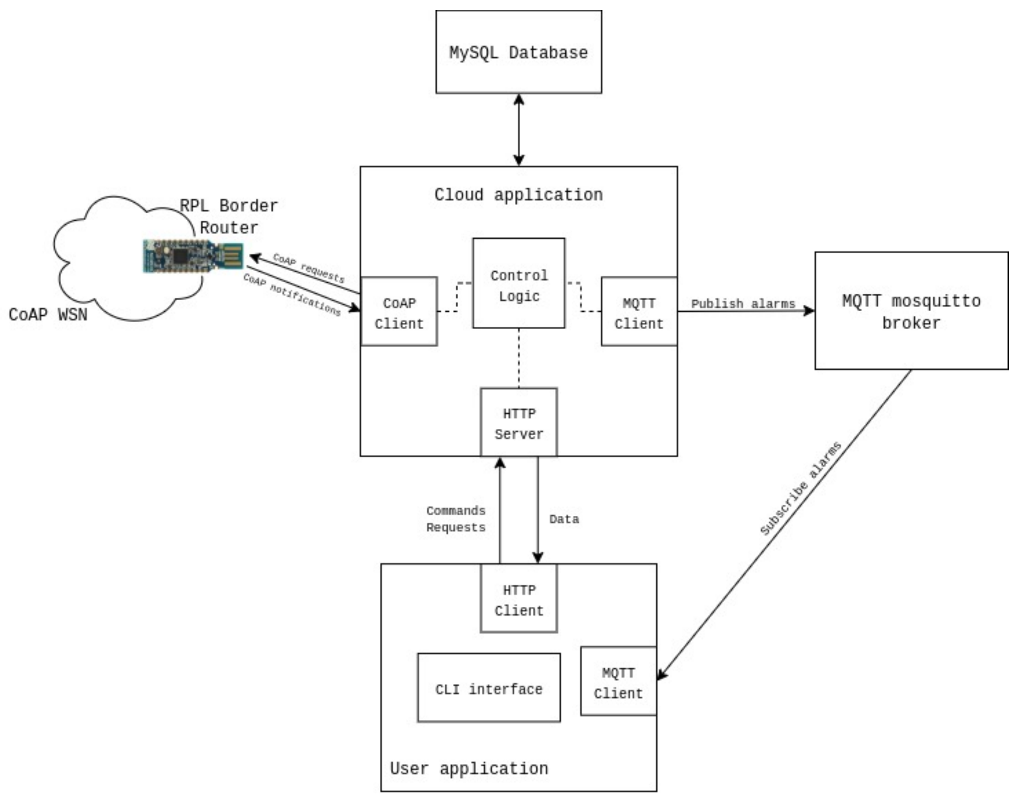


Figure 5: Architecture of the cloud infrastructure

The RPL Border Router serves as the gateway through which all data from the local CoAP-enabled nodes (energy and HVAC) is transmitted toward the cloud. Conversely, it also functions as the access point through which commands, configurations, or acknowledgments from the remote control infrastructure are relayed back into the local wireless sensor network. This bidirectional flow ensures that remote intelligence and control decisions are effectively deployed in real time to the physical environment.

At the cloud level there is the *cloud application*, which hosts two integrated services: a *remote control logic module* and an *HTTP server interface*.

The remote control logic is responsible for computing the next set of control actions for the actuators, based on the current system state. More precisely, it retrieves from the MySQL database the latest sensed data and the most recent actuator parameters, and on this basis determines, according to internal policies or optimization criteria, the updated configuration to be applied to the nodes. This enables dynamic system behavior in response to changes in environmental conditions or energy availability, implementing reactive control strategy.

Simultaneously, the cloud application performs the task of continuously receiving, processing, and storing data sent by the sensor nodes acting as CoAP client. Upon arrival, each measurement is validated and archived in the MySQL database along with the timestamp, ensuring data persistence and historical traceability. The stored dataset supports not only control decision-making but also data analysis.

The cloud application exposes an HTTP server, allowing user applications to connect through a standard web interface. Via this interface, users can visualize in real time the values of sensors (e.g. room temperature, solar panel output, irradiation) and actuators (e.g. relay status, HVAC operational state), and also push updates to configurable parameters. For example, users can set the target temperature of the HVAC system, select its operational mode (normal or green), or directly instruct the energy node on how to route generated power—whether to the battery, grid, or directly to the load.

Complementing this interaction loop, the cloud application integrates an MQTT broker to handle asynchronous alarm management and event propagation. When the energy node raises a signal indicating an *anti-dust operation* or detects a persistent mismatch between predicted and actual power output, the cloud publishes messages to an `antiDust` topic. Similarly, if the HVAC node encounters a fault or an unresolvable operational issue, it sends an error signal which results in the publication of a message on the `hvac` topic. The user application, acting as an MQTT client, subscribes to these topics. This allows it to receive alarms and warnings in real time and immediately alert the user, so that he can autonomously initiate mitigation procedures.

Overall, the system represents a holistic integration of renewable generation, predictive analytics, intelligent control, and fail-safe mechanisms, aimed at optimizing energy usage in HVAC applications while maximizing sustainability and system autonomy.

3 Implementation

3.1 Sensors and actuators

For simmplicity there are three nodes: the RPL Border Router, the Energy Node, and the HVAC Node. The application could be easily expanded to control a system with many HVAC and energy nodes, or with sensors separated from the rest of the devices.

The operativity of the **RPL Border Router** is visually confirmed by the activation of a blue LED.

The **Energy Node**, whose activation is signaled by a green LED, integrates a set of environmental and electrical sensors. These include a weather sensor unit capable of measuring three key physical quantities: the ambient external temperature `outTemp` (expressed in °C), the temperature of the solar panel module `modTemp` (also in °C), and the solar irradiation `irr` (in kWh/m^2). In addition, the node constantly monitors the instantaneous power output `gen_power` (in watts) generated by the solar panel, and the current charge level of the battery `batter_level` (in watt-hours). Sensor values are transmitted using the CoAP observing mechanism, which allows for efficient periodic notifications. While weather and generation data are pushed at regular intervals, updates to the battery level are event-driven and transmitted only when a change is detected.

The Energy Node also controls some actuators. The relay system managing the routing and sourcing of electrical power is represented by four functional parameters: the parameter `r_sp` determines the destination of the power produced by the solar panel (selecting among house, battery, or the electric grid); the parameter `r_h` sets the source of energy powering the house (again among solar panel, battery, or grid); while `p_sp` and `p_h` represent the effective power currently flowing from the panel and consumed by the HVAC system, respectively.

An additional actuator is the **antiDust** mechanism, which can operate in three distinct modes: `off`, `on`, and `alarm`. This system is governed by a predictive maintenance logic: at each prediction interval, the system estimates the expected solar panel output using a machine learning model trained on weather data. If the measured generation power remains consistently below a predefined threshold compared to the predicted value for more than n consecutive iterations, the antiDust cycle is triggered. This activates a cleaning operation (signaled by a blinking yellow LED) for a limited number of seconds. Should the discrepancy persist even after the cleaning attempt, the node transitions to the `alarm` state. Manual intervention is permitted through physical interaction: a short button press allows the user to force a transition from `dust` to `clean`, whereas a long press of three seconds toggles the state between `off` and `alarm`.

The **HVAC Node**, which indicates its activation with a red LED, is equipped with an internal room temperature sensor `roomTemp`. It also implements several actuator `settings`: `conditioner_power`, which quantifies the power currently consumed by the HVAC unit; the `status`, which can be *off*, *vent*, *cool*, *heat*, or *error*; the `mode`, which determines whether the node operates in *normal* or *green* mode; and the `targetTemp`, the user-defined temperature that the system aims to maintain.

In `normal` mode, the HVAC node behaves passively, merely executing commands received from the central control logic or from the user interface. However, when switched to `green` mode, it assumes an autonomous role. At each cycle, it initiates a request to the Energy Node to retrieve `outTemp`, `gen_power`, and `battery_level`. With this data, the HVAC computes the *needed_power* required to satisfy its current settings.

The node checks whether `gen_power` is sufficient to supply the computed `needed_power`. If so, it configures the relay accordingly and proceeds. If solar energy is inadequate, it evaluates whether the energy demand for the current time interval (converted to equivalent DC energy) can be fulfilled by the battery. If this second condition is met, it again sets the relay to use the battery as the power source. Failing both conditions, and if the current operational mode is `cool`, the node attempts to downgrade to the `vent` mode—a low, fixed consumption alternative—and repeats the feasibility checks. If even this fallback proves unfeasible due to insufficient power, the node sets the dissipated power to zero and enters a standby state, awaiting the next iteration for a possible reactivation.

It is possible to toggle the error status of the HVAC node with a three seconds press of the button, simulating the human intervention.

3.2 Communication protocols and data encoding

Within the WSN the chosen protocol is CoAP due to its lightweight design, minimal code footprint, and the inherent support for the *observe* mechanism, which enables efficient periodic or event-driven updates from sensors to subscribers. Furthermore, using a uniform CoAP-based stack across all nodes simplifies the development and maintenance of the software. Sensor values are distributed using CoAP observe notifications.

In contrast, the cloud environment, which benefits from far fewer resource constraints, employs HTTP as the primary protocol for interactions with user applications. HTTP guarantees broad compatibility with existing web frameworks.

To further support timely event notification and asynchronous communication—particularly in response to fault conditions, alarm signals, and anti-dust activations—the system integrates the MQTT protocol because it is a lightweight publish/subscribe protocol that ensures low-latency message dissemination with minimal overhead for its ease of implementation on *Mosquitto*. When the energy node detects a dust-related degradation or an alarm condition, or when the HVAC node reports an internal fault, the cloud application publishes structured messages to the corresponding MQTT topics (e.g., `antiDust` or `hvac_alarm`).

In terms of payload formatting, CoAP messages that include data—such as `POST` requests to actuators—use a simple `key=value` format. This format is intentionally selected to reduce parsing complexity and to keep message size small. On the other hand, responses from CoAP endpoints are structured in `JSON` format. JSON is preferred for its flexibility, human readability, and ability to encapsulate complex hierarchical data, which is especially valuable when dealing with multi-parameter actuator settings. The JSON structure used is partially aligned with the SenML (Sensor Measurement Lists) specification, employing fields such as `n` for the sensor name and `v` for the corresponding value. However, the current implementation does not yet fully conform to the SenML standard, and further refinement would be necessary to achieve complete interoperability.

In the MySQL database sensor data are stored in a single relational table to allow unified querying and aggregation across different sensor types. Each record includes an identifier, the sensor name, the measured value, and a timestamp. Actuator states are stored in a complementary structure, with each entry consisting of a unique identifier, the state parameters and a timestamp to enable temporal analysis and historical state reconstruction.

3.3 Solar power prediction model

To support the preventive maintenance anti-dust procedure, the system integrates a machine learning model that forecasts the expected solar power generation based on current weather conditions. The model operates at regular intervals and produces a real-time estimation of photovoltaic output, which is then compared to the actual energy measured by the energy node. Underperfomances between predicted and observed values are used as indicators of potential panel inefficiencies, such as dust accumulation.

The prediction model was trained using a supervised learning approach with regression. The dataset employed consists of historical measurements collected over several weeks of ambient temperature, module (panel) temperature, and solar irradiation as input features. The corresponding ground truth is the actual generated power (in watts) recorded by the panel over the same intervals.

After exploratory analysis and feature scaling, the model was trained using root mean squared error as the loss function, and hyperparameters such as the number of layers were optimized via cross-validation.

Once trained, the model was exported and integrated into the energy node with `emlearn` to reduce its footprint.

3.4 Physical data generation

The system generates synthetic yet physically plausible data.

The battery level, for instance, is dynamically updated at each iteration by computing the net charging rate, defined as the difference between the power provided by the solar panel (when routed to the battery) and the power consumed by the HVAC system (when it draws energy from the battery).

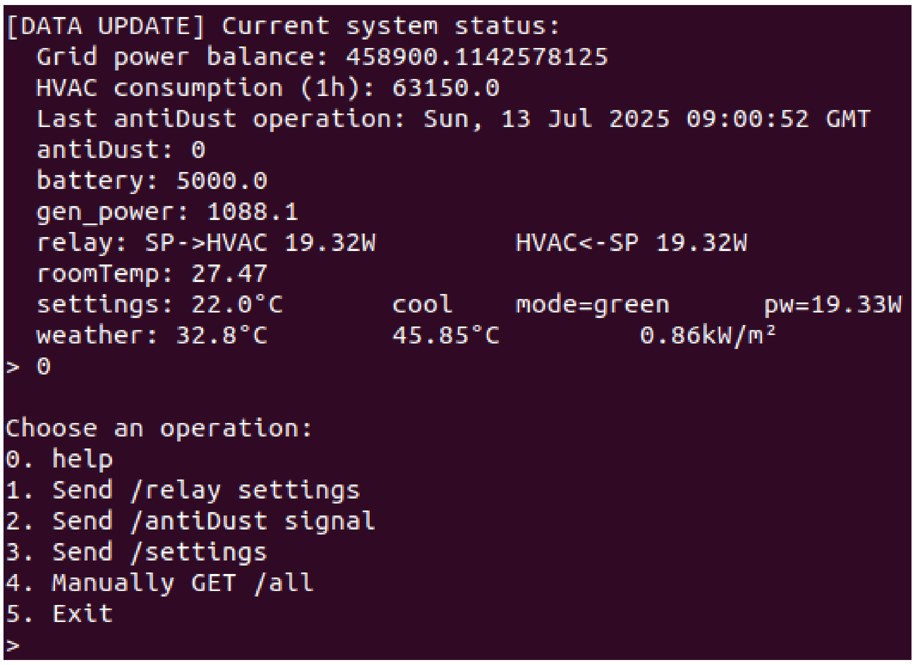
Weather variables such as ambient temperature, module temperature, and solar irradiation are simulated with bounded randomness: at each timestep, their values are perturbed with small random offsets relative to their previous values.

The generated power of the solar panel (`gen_power`) is also modeled to mimic realistic conditions. Under normal operation, it follows a profile statistically similar to that predicted by the machine learning model. However, when the system enters a `dust` mode—triggered manually via a button—the generation drops significantly, simulating the effect of panel soiling.

The indoor temperature (`roomTemp`) is modeled as a dynamic process driven by the interaction between external and internal thermal effects. Specifically, the outside temperature exerts a continuous influence, tending to bring the indoor temperature toward equilibrium. Simultaneously, if the HVAC system is in heating or cooling mode, it contributes additively to increasing or decreasing the room temperature, respectively.

3.5 User interface

The user interacts with the cloud application with a CLI interface. He receives periodic updates about the status of the system, some statistics such as the total power consumed in the last hour or the net between the energy produced and consumed from the grid or the last anti-dust procedure time.



```
[DATA UPDATE] Current system status:
  Grid power balance: 458900.1142578125
  HVAC consumption (1h): 63150.0
  Last antiDust operation: Sun, 13 Jul 2025 09:00:52 GMT
  antiDust: 0
  battery: 5000.0
  gen_power: 1088.1
  relay: SP->HVAC 19.32W         HVAC<-SP 19.32W
  roomTemp: 27.47
  settings: 22.0°C       cool    mode=green      pw=19.33W
  weather: 32.8°C        45.85°C         0.86kW/m²
> 0

Choose an operation:
0. help
1. Send /relay settings
2. Send /antiDust signal
3. Send /settings
4. Manually GET /all
5. Exit
>
```

Figure 6: User interface - CLI