Conceptual design for an energy consumption meter simulator (workshop 2)

Students:

- Andres Jeronimo Ramirez
- David Santiago Ibañez
- Johan Danilo Trujillo

Course: Object-Oriented Programming — Semester 2025-II

1. Conceptual design updates

❖ Requirements documentation

1.1. Functional requirements

The system must have:

- Display consumption data to the user.
- Notify the user about consumption.
- Allows you to set consumption limits.
- Show expected device consumption.
- View consumption over the course of the month, either numerically or graphically.
- View a list of each device, sorted from highest to lowest, based on consumption, including the cost in Colombian pesos (COP) for each device.
- View a breakdown of each device in the list.
- Change the device name.
- Remotely turn off the desired socket.
- Allows you to view information about concepts and functions.
- When a device in the list is selected, open a detailed interface with a daily consumption graph (kWh vs. time).

- Integrate an analysis module that identifies consumption patterns, anomalies (e.g. devices consuming more than expected), and provides recommendations for energy saving.

1.2. Non functional requirements

Category	Requirement
Usability	The interface should be intuitive for new users or people with little experience.
Performance	Must support at least 20 devices stably
Reliability	It must maintain stable transitions even with rapid interaction.
Portability	The system should run on Windows, macOS, and Linux using Python.
Maintainability	Modular OOP design to ensure readable and modifiable classes.
Educational Value	The system must support clear visualization of signal flow for learning purposes.

❖ User stories

US-01: Real-Time Energy Dashboard for Homeowners	**Priority:** High	**Estimate:** 10-12 hours
As a homeowner, I want to view my current and past energy usage in a dashboard, so that I can identify which appliances use the most power and receive alerts if usage is unexpectedly high.		
Acceptance criteria: - The dashboard shows live voltage, current, and power values for each monitored appliance. - The homeowner can select a time range and see a historical graph of total energy consumption.		

- The web app sends a notification (email or on-screen) if an appliance's power exceeds a set threshold. - The interface displays an estimated cost of energy used in the current billing cycle.

US-02: Sensor Calibration and Diagnostics for Technicians	**Priority:** Medium-High	**Estimate:** 8-10 hours
As a technician, I want to calibrate and test the sensors and device, so that I can ensure accurate measurements and diagnose any issues.		
Acceptance criteria: - A "Calibration" section allows adjusting sensor calibration parameters (offset/gain) and saving them. - The technician can trigger a self-test or view raw sensor readings. - All calibration actions and measurements are logged with timestamps. - If any hardware error (e.g. sensor disconnected) is detected, an error alert is raised.		

US-03: Modular Code and Simulator for Developers	**Priority:** High	**Estimate:** 12-16 hours
As a developer, I want clear, modular classes and a simulator, so that I can develop and maintain the system easily and test features without the hardware.		

Acceptance criteria: - Code is organized into OOP classes (Sensor, NetworkManager, etc.) and fully documented. - A virtual simulator can feed fake sensor data into the system for testing. - The developer can deploy firmware updates (via OTA) and access debug logs. - Unit or integration tests exist for each module (e.g. data logging, alerting).

US-04: Family Member Shared Access Dashboard	**Priority:** Medium	**Estimate:** 6-8 hours
As a family member, I want to view the household's energy usage and alerts so that I can help monitor and reduce our home's energy consumption.		
Acceptance criteria: - The homeowner can invite family members with limited (read-only) access. - Shared accounts can view the dashboard and alerts but cannot modify system settings. - The system ensures proper permissions per role. - Family members can receive alert notifications if enabled by the homeowner.		

US-05: System Administration and OTA Management	**Priority:** High	**Estimate:** 10-12 hours
As a system administrator, I want to manage users, configurations, and firmware updates so that the network remains secure and up-to-date.		

Acceptance criteria: - Admin can add/remove users and assign roles (owner, family, technician). - Admin can trigger and monitor OTA firmware updates. - Admin can view system logs (connectivity, sensor errors, firmware status). - Admin can edit global thresholds and alert rules.

US-06: Energy Ranking and Cost Analysis	**Priority:** High	**Estimate:** 12-14 hours
As a homeowner, I want to see an ordered list of devices by energy consumption and cost in Colombian pesos, so that I can identify which devices use the most power and take actions to reduce my bill.		
Acceptance criteria: - The system displays a ranked list (high → low) of devices by daily energy usage (kWh). - Each device shows its consumption cost in COP using a configurable tariff. - When clicking a device, a detailed page opens showing a daily consumption graph. - A “Monthly Analysis” tab ranks devices by monthly usage and total cost. - The system highlights outliers or excessive consumption and suggests possible causes (e.g., standby losses, malfunction).		

❖ CRC Cards

4.1. Class: Device

Responsibilities:

- Store device data (name, power, usage time).
- Calculate individual energy consumption (kWh).

- Provide consumption history.
- Send data to the consumption manager.

Collaborators:

- Device manager
- Energy calculator
- UI

4.2. Class: Device manager

Responsibilities:

- Register devices.
- Delete and edit connected devices.
- Maintain a list of all devices.
- Request individual usage calculations.
- Detects devices with high usage.

Collaborators:

- Device
- Energy calculator
- UI

4.3. Class: Energy calculator

Responsibilities:

- Calculate consumption in kWh (energy = power × time).
- Convert kWh to Colombian pesos.
- Calculate total monthly consumption.
- Generate statistics (averages, peaks, trends).

Collaborators:

- Device
- Device manager

4.4. Class: Consumption history

Responsibilities:

- Save daily/weekly/monthly usage.
- Allow historical analysis.
- Export data for reporting.
- Notify of usage changes.

Collaborators:

- Device
- Device manager
- UI

4.5. Class: Concept info

Responsibilities:

- Store educational definitions (voltage, current, power, resistance).
- Show real-life examples.
- Provide optional diagrams or images.
- Act as a study guide.

Collaborators:

- UI
- Settings

4.6. Class: User settings

Responsibilities:

- Save user preferences.
- Configure notifications (consumption limits).
- Change currency and units (optional).
- Control language and interface.

Collaborators:

- UI

4.7. Class: Notification manager

Responsibilities:

- Send alerts when a device consumes too much.
- Notify of monthly increases.
- Show savings tips.

Collaborators:

- Device manager
- Consumption history

- UI

4.8. Class: UI (user interface)

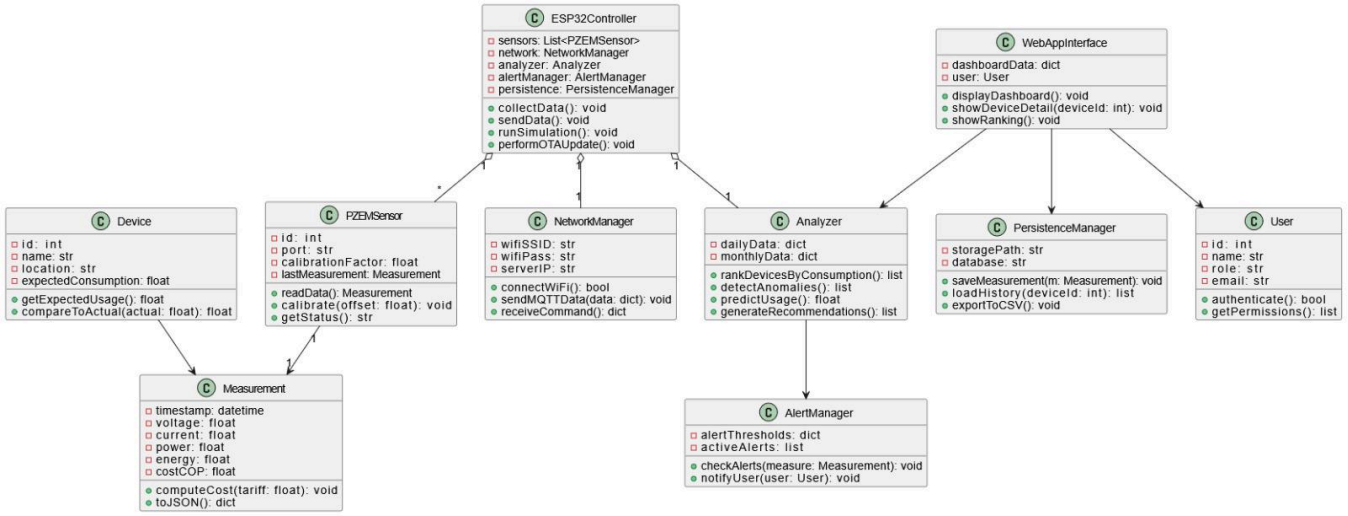
Responsibilities:

- Display monthly usage.
- Display monthly cost in pesos.
- Render list of devices.
- Pressing a device → opens details.
- Display educational information (Concept Info).

Collaborators:

- Device manager
- Energy calculator
- Concept info
- Notification manager

2. Technical Design (UML):



3. Implementation Plan for OOP concepts:

ESP-32 Class: We have an ESP32, a very important parent class in our circuit. It will receive data from the sensors, analyze it, send it to the application, and make the simulation work. It will have private encapsulation.

PZEM Sensor class: Another parent class will be our current, voltage, and power sensor. It will be able to receive the data and send it to the ESP32 so it can analyze it and send it to the app interface. It will have private encapsulation.

Device class: This will be the parent class for different child classes that we will create for several home devices, such as fridges, televisions, air conditioners, etc. It will also need the measurements and general data from the circuit. It will have public encapsulation.

Measurement class: This is a child class that will take different characteristics from the PZEM sensor or meter, and it will also need data from the device to show the information. It will have protected encapsulation.

Network Manager class: A child class that will interpret internet data for the ESP32 and the device. It will have protected encapsulation.

Web App Interface class: This is a parent class that will receive data from the ESP32 through a child class called the Analyzer. It will also be the parent class for storage, database, and user information. It will have public encapsulation.

Analyzer class: A child class of the ESP32 and the graphical interface. It will analyze the data from the meter, which comes from the ESP32. It will also work as a parent class for an alert manager. It will have private encapsulation.

Alert Manager class: Based on the analysis of different devices and measurements, it will send alerts depending on the problem or energy consumption. It will be a child class with public encapsulation.

Persistence Manager class: This is a child class of the graphical interface. It will save the app data and load the user's history. It will have protected encapsulation.

User class: A child class that represents the person using the application. The user can connect different home devices to the simulator and the implemented circuit through the app. It will have public encapsulation.

4. Initial Python code snippets:

(The code is located in the "Workshop_2" folder)

This code will use simulated data that can be changed in main.py, you can change the device name, type, usage hours, and wattage consumption.

It has a main class called "medidor.py", from which the power meter, time and cost of kWh in Colombian pesos are inherited.