Domotic Circuit Simulator — Conceptual Design

Object-Oriented Programming – Semester 2025-II

Juan Diego Arévalo Bareño

October 2025

1. Requirements Documentation

Functional Requirements:

- The system must allow the user to add different domotic components to a virtual workspace.
- The user must be able to connect the components to each other using virtual wires to form a functional circuit.
- The Simulator must execute the circuit simulation, showing the behavior of the components as quickly as possible.
- The user must be able to interact with the components during simulation.
- The system must be able to save and load circuits designs created previously.
- The Interface must provide visual feedback that shows the status of the components.
- The user must have options to eliminate and edit components previously added to the circuit.
- The simulation must include options to stop, resume, and reset the simulation.
- The system must validate the connections to avoid errors or incompatible settings.
- The simulation must register basic information about the simulation, such as the component numbers and the state of each one.

Non-Functional Requirements:

- The interface must be intuitive and easy to use.
- The system must execute the simulation without notable delays.
- The system arquitecture must allow to add new components without modifying existing types.
- The simulation must be executed in different operative systems that support python.
- The system must avoid invalid connections.
- The code must follow the POO principles, with a clear structure.
- All the interface elements must be visible and understandable.

2. User Stories

User Storie 1: As a student, I want to add components such as lights, sensors, and switches to the workspace so that I can build a basic domotic circuit and understand how connections work.

Acceptance Criteria:

- The user can drag and drop components onto the workspace.
- Components are displayed with their names and visible positions.
- The system automatically saves the changes or allows manual saving.

User Story 2: As a user, I want to save my circuit designs and load them later so that I can continue working without losing progress.

Acceptance Criteria:

- The system allows the user to save the circuit into a local file.
- The user can load a previously saved circuit.
- Loaded components and connections are displayed correctly on the workspace.

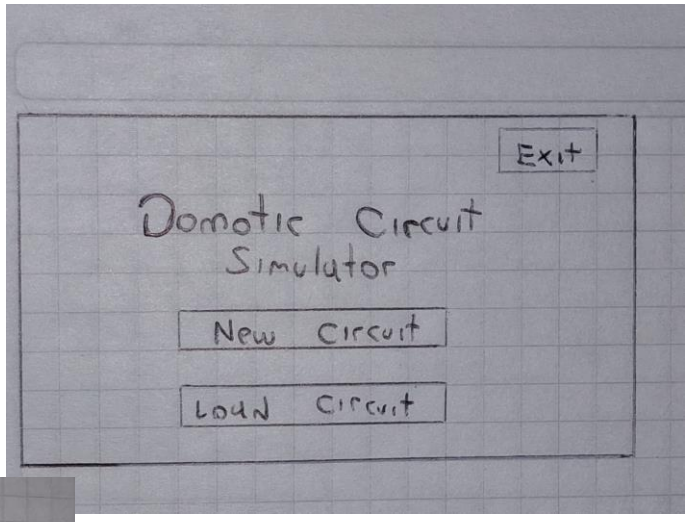
User Story 3: As an instructor, I want to open projects created by students so that I can review and evaluate their designs.

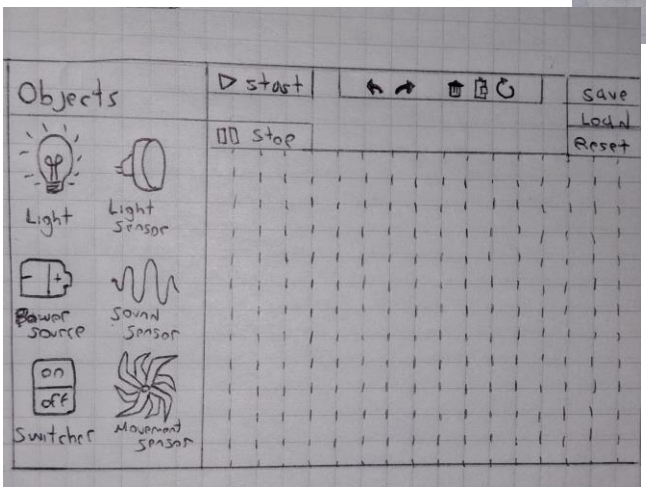
Acceptance Criteria:

- The instructor can access files created by students.
- The system loads components and connections correctly.
- The instructor can run the simulation without modifying the circuit design.

3. Mockups

Main Dashboard: The main screen includes only the essential options: *New Circuit*, *Load Circuit*, and *Exit,* to help users start quickly without distractions. The centered layout and clear labels make it easy to navigate for students who may not have previous experience with simulation tools.





Circuit Editor: This interface was designed as the core workspace of the simulator. On the left, a component panel displays icons for the domotic devices, allowing users to drag and drop them easily into the workspace. The toolbar on top provides access to key functions such as *Start*, *Stop*, *Undo*, *Redo*, *Delete*, *Save*, *Load*, and *Reset*, supporting efficiency and organization.

The grid-based workspace helps align and connect components clearly, simulating the physical layout of a real electronic circuit. The right side contains simple management options (*Save, Load, Reset*), ensuring that users can keep track of their progress.

4. CRC Cards

- **Class:** Components
  Responsibilities:
  ***** Store the basic attributes of all components.
  * Provide methods to connect or disconnect from other components.
  * Act as a base class for all specific component types.
  Collaborators: Circuit, Switch, Light, Sensor.

- **Class:** Switch

Responsibilities:

***** Change its state between ON and OFF.

* Send a signal to connected components when activated.
* Update its visual representation in the simulation.

Collaborators: Circuit, Component, Light, Simulator.

- **Class:** Lights

  Responsibilities:

  ***** Receive input signals from switches or sensors.

  * Change its state according to received signals.
  * Display visual feedback during simulation.

  Collaborators: Circuit, Sensor, Switch, Simulator.

- **Class:** Sensor

  Responsibilities:

  ***** Detect external conditions.

  * Send activation signals to other components.
  * Provide input data to the simulation loop.

  Collaborators: Circuit, Light, Switch, Simulator.

- **Class:** Circuit

  Responsibilities:

  ***** Store and manage all components within the circuit.

  * Handle the connections between components.
  * Validate the circuit structure before simulation.
  * Save and load circuit configurations.

  Collaborators: Components, Sensor, Light, Switch, Simulator.

- **Class:** Simulator

  Responsibilities:

  ***** Control the execution of the simulation

  * Update component states in real time.

* Display messages or feedback to the user.
* Manage the simulation clock or timing events.

Collaborators: Circuit, Components, Sensor, Light, User Interface.