Smart Home IoT Automation Simulator Documentation

Introduction

The Smart Home IoT Automation Simulator is a Python-based project designed to simulate the behavior of various IoT devices commonly found in a smart home. This documentation provides a comprehensive guide on the code structure, classes, methods, and instructions for running the simulation and using the monitoring dashboard.

Table of Contents

Classes and Methods

Device Class

Attributes:

- `device_id`: Unique identifier for the device.
- `is_on`: Current status of the light (on/off).

Methods:

1. `turn_on()`: Turns on the smart light.
2. `turn_off()`: Turns off the smart light.
3. `get_is_on()`: Returns is_on attribute.
4. `get_device_id()`: Returns device_id attribute.

SmartLight Class

Attributes:

- `brightness`: Brightness level of the light.

Methods:

1. `set_brightness(brightness)`: Sets the brightness of the smartlight to the specified value.
2. `get_brightness()`: Returns brightness attribute.

Thermostat Class

Attributes:

- `temperature`: Current temperature setting of the thermostat.

Methods:

1. `set_temperature(temperature)`: Sets the temperature of the thermostat to the specified value.
2. `get_temperature()`: Returns temperature attribute

SecurityCamera Class

Attributes:

- `motion_detected`: Holds whether motion is detected or not.

Methods:

1. `detect_motion()`: Set True to motion_detected
2. `not_detect_motion()`: Set False to motion_detected
3. `set_motion_detected()`: Sets the motion_detected of the security camera to the specified value.
4. `get_motion_detected()`: Returns motion_detected attibute

AutomationSystem Class

Attributes:

- `devices[]`: List of all connected IoT devices.
- `stop_threads`: Flag to terminate thread
- `automation`: Flag to activate/deactivate automation

Methods:

1. `get_devices()`: Returns devices[] attribute
2. `discover_devices(device)`: Discovers and adds new devices to the system.
3. `automation_on()`: Set True to automation attribute
4. `automation_off()`: Set False to automation attribute
5. `automatic_lighting()`: Executes automatic lighting
6. `store_sensor_data()`: Output various statuses to external files

Running the Simulation

To run the simulation, follow these steps:

1. Ensure you have Python installed on your system.
2. Clone the repository and navigate to the project directory.
3. Run the `main.py` script to start the simulation.

Monitoring Dashboard

The monitoring dashboard is created using Tkinter and provides a user-friendly interface to interact with and monitor the smart home system.

Features:

- `Automation ON/OFF` button: Toggle whether automation is activated or not. The text below will indicate whether it is On or Off. In this application, Brightness is forced to 100% when the camera detects motion. However, Light must be On.
- `Text field at the top`: Displays On/Off of various devices and On/Off of Random Detect Motion. Constantly updated as changes are made
- `Scale for Living Room Light Brightness`:User can adjust brightness directly. The status is displayed in the text field at the bottom.
- `Toggle for Living Room Light`:Turn the lights on and off. The status is displayed in the text field at the top.
- `Scale for Living Room Thermostat`:User can adjust temperature directly. The status is displayed in the text field at the bottom.
- `Toggle for living Room Thermostat`:Turn the thermostat on and off. The status is displayed in the text field at the top.
- `Random Detect Motion ON/OFF` Activate/deactivate Random detect motion. The status is displayed in the upper text field. Random detect motion randomly determines if the camera detects motion once every 5 seconds when the camera is on. Whether motion is detected or not is indicated by the True/False value of motion_detected in the lower text field.

The status is displayed in the upper text field.

- `Toggle for Front Door Camera`:Turn the SecurityCamera on and off.
- `Text field at the bottom`:The current date and time, brightness, temperature, and motion_detected are displayed every three seconds.

Test Cases

Ensure the simulator and automation system behave as expected by running various test scenarios:

1. **Scenario 1: Turning On/Off Devices**
    - Turn on and off each type of device and verify their status.
2. **Scenario 2: Adjusting Device Properties**
    - Adjust brightness for smart lights.
    - Set different temperatures for thermostats.
3. **Scenario 3: Automation Rules**
    - Test automation rules (lights turning on when motion is detected).
4. **Scenario 4: User Interaction**
    - Interact with devices through the GUI and ensure proper functionality.
5. **Scenario 5: Randomization Mechanism**
    - Verify that motion_detected states change randomly over time.
