Smart Home IoT Simulator Documentation

Table of Contents

Directory Structure

```
graphqlCopy code
smart_home_automation/
├── main.py                   # Entry point for running the dashboard GUI
├── system/                   # Defines the AutomationSystem class
│   ├── __init__.py           # Initializes the devices package
│   ├── automation_system.py  # Defines the AutomationSystem class
├── devices/                  # Contains device classes
│   ├── __init__.py           # Initializes the devices package
│   ├── SmartLight.py         # SmartLight class definition
│   ├── Thermostat.py         # Thermostat class definition
│   └── SecurityCamera.py     # SecurityCamera class definition
├── gui/                      # Contains GUI-related classes
│   ├── __init__.py           # Initializes the gui package
│   └── SmartHomeDashboard.py # SmartHomeDashboard GUI class definition
└── sensor_data.csv           # csv for Sensor Data Logging
```

Installation and Running the Simulator

Prerequisites

- Python 3.7 or higher
- Tkinter library (included with Python)

Running the Simulation

1. Navigate to the project's root directory.
2. Execute `python main.py` in the terminal.
3. The dashboard GUI will open, displaying controls for each smart device.

Class Documentation

SecurityCamera Class

The `SecurityCamera` class simulates a security camera with streaming and motion detection capabilities.

Constructor

```
pythonCopy code
def __init__(self, device_id, device_name, is_motion_detected=False, status='off', is_
streaming=False):
```

Initializes the SecurityCamera with IDs, names, and initial statuses.

Methods

- `get_status()`: Returns the camera's current status.
- `toggle_streaming()`: Toggles the camera's streaming state.
- `stop_streaming()`: Stops streaming if active.
- `detect_motion(motion)`: Updates motion detection status.
- `toggle_motion_detection(is_detected)`: Toggles the motion detection feature.
- `set_motion_detection(is_detected)`: Sets the motion detection status.

SmartLight Class

The `SmartLight` class models a smart light device that can be turned on/off and have adjustable brightness.

Constructor

```
pythonCopy code
def __init__(self, device_id, device_name, status='off', brightness=0):
```

Sets up a SmartLight with specified attributes.

Methods

- `get_status()`: Retrieves the light's status and brightness.
- `turn_on()`: Turns on the light, gradually increasing brightness.
- `turn_off()`: Turns off the light, setting brightness to 0.
- `set_brightness(level)`: Adjusts the light's brightness.
- `toggle_status()`: Switches the light's on/off state.

Thermostat Class

The `Thermostat` class represents a thermostat device controlling environmental temperature.

Constructor

```
pythonCopy code
def __init__(self, device_id, device_name, target_temperature=20):
```

Creates a thermostat with a target temperature.

Methods

- `get_status()`: Gets the current and target temperatures and status.
- `turn_on()`: Activates the thermostat.
- `turn_off()`: Deactivates the thermostat.
- `set_target_temperature(temperature)`: Sets a new target temperature.
- `update_temperature()`: Adjusts the current temperature toward the target.
- `randomize_temperature()`: Randomly alters the current temperature.
- `toggle_status()`: Toggles the thermostat's on/off status.

Dashboard Class

`Dashboard` is the GUI class for interacting with the Smart Home IoT Simulator.

Key Methods

- `create_widgets()`: Sets up the GUI components.
- `update_device_status_labels()`: Updates the device status labels.
- `create_status_indicator()`: Creates visual indicators for device status.
- `update_status_indicator()`: Changes the indicator color based on the status.
- `toggle_automation_status()`: Changes the automation system's status.
- `update_automation_status()`: Reflects the automation status on the GUI.

Usage Example

To launch the dashboard within the application, use the following code snippet:

```
pythonCopy code
root = tk.Tk()
system = AutomationSystem()
dashboard = Dashboard(root, system)
dashboard.run()
```

Main Application Script

Overview

The main script initializes the smart home system, adds devices, and starts the dashboard GUI for user interaction.

Script Workflow

1. Initialize the `AutomationSystem`.
2. Create `SmartLight`, `Thermostat`, and `SecurityCamera` instances.
3. Add devices to the system.
4. Set up the Tkinter root and start the dashboard.
5. Log the initial state of devices after the GUI begins.
6. Perform cleanup and resource deallocation after GUI closure.

Testing

- Confirm correct initialization and device addition.
- Verify the dashboard launches and displays device states accurately.
- Check that log messages are appropriate and the cleanup process is thorough.

Testing

Overview

Testing ensures that the smart home system and its components operate as designed, both individually and as part of the larger system.

Test Cases

- **Widget Creation**: Confirm GUI widgets are created properly.
- **Device Status Update**: Check if device statuses are updated accurately in the GUI.
- **User Interaction**: Verify GUI controls trigger the correct system responses.
- **Event Logging**: Ensure log messages are displayed in the GUI's log area.

Testing Smart Devices

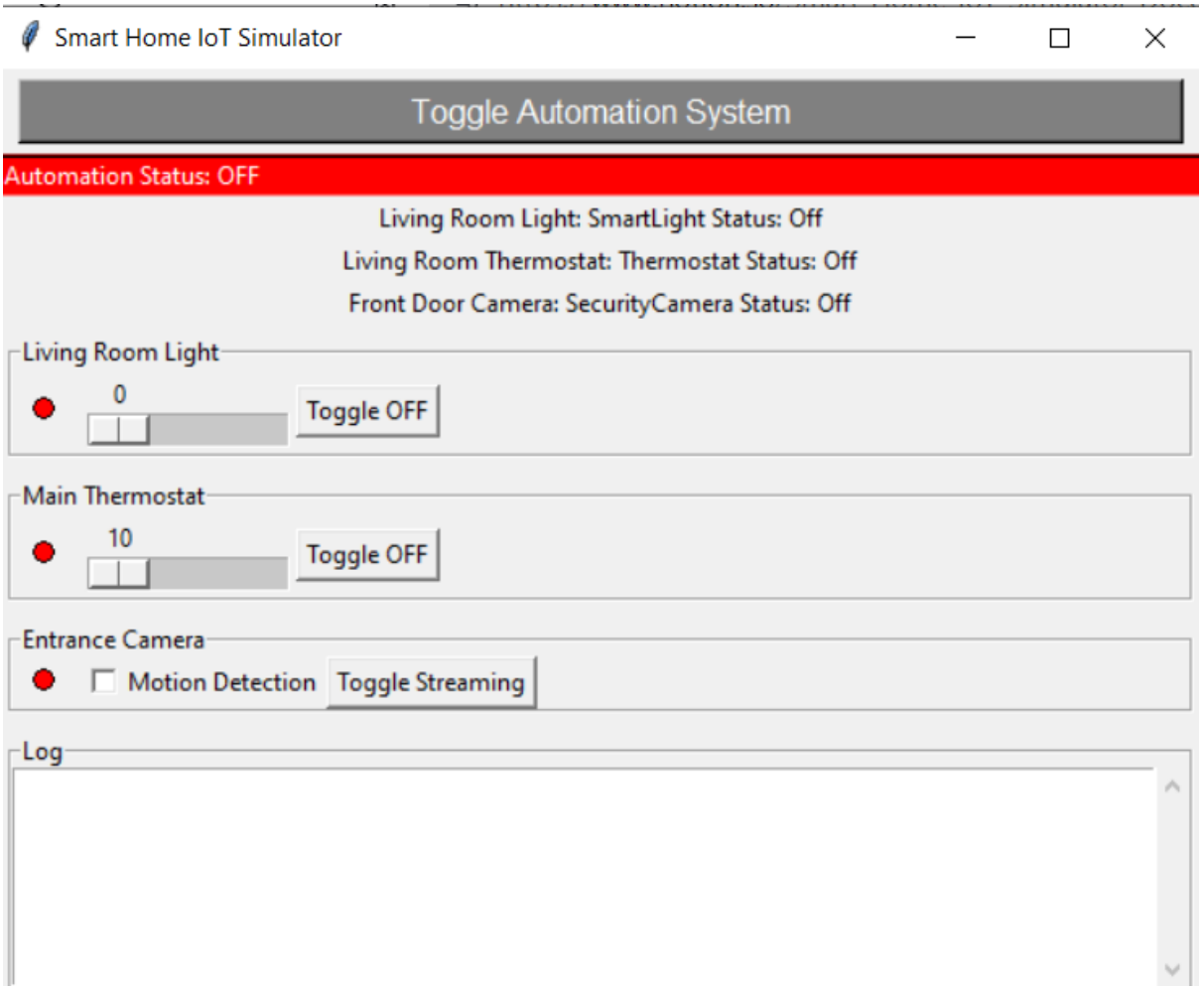
- **SmartLight**: Test brightness adjustments and on/off toggling.
- **Thermostat**: Verify temperature settings and on/off functionality.
- **SecurityCamera**: Check motion detection toggling and streaming functions.

Additional Notes

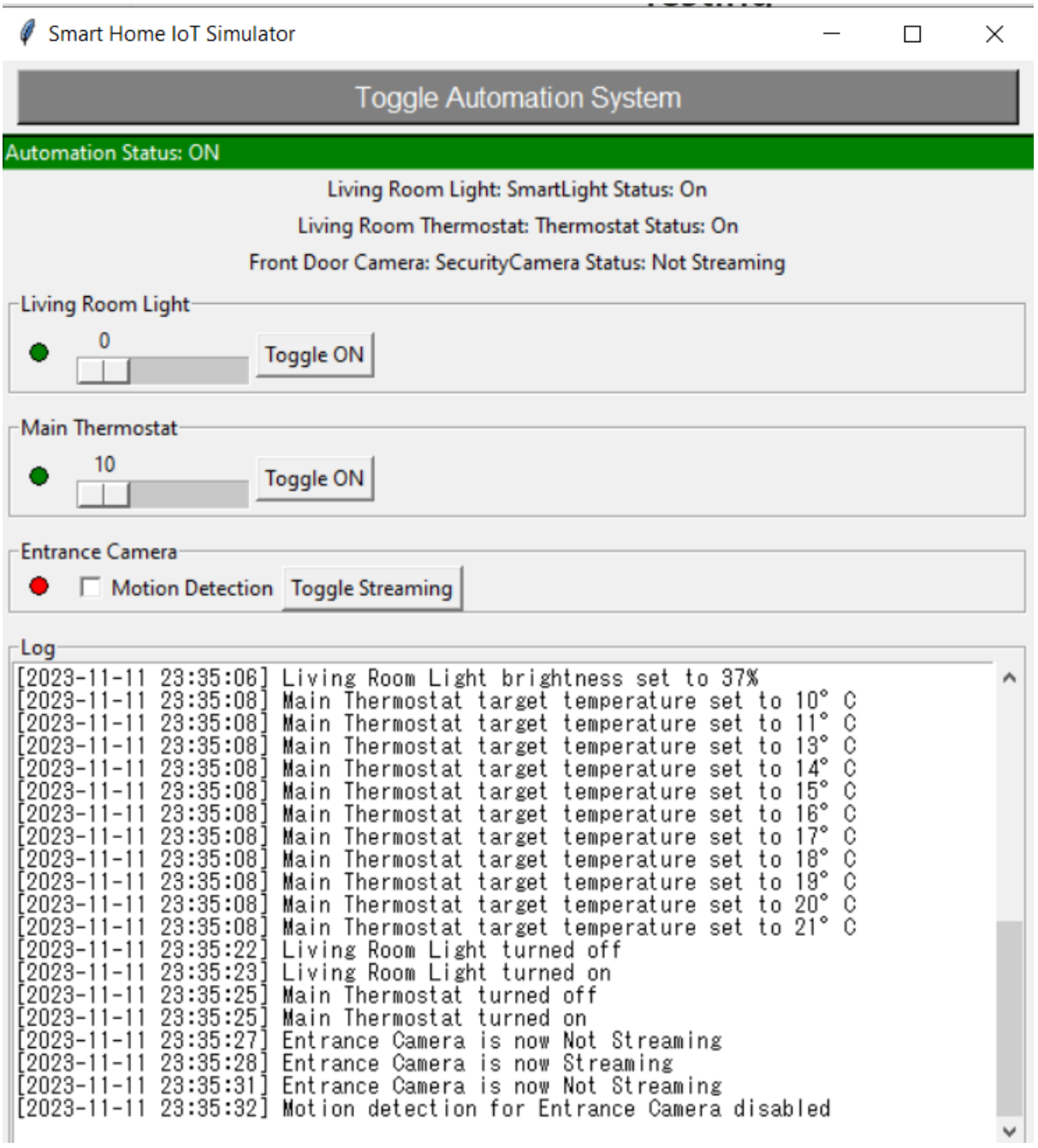
- Updates to GUI elements must occur on the main thread due to Tkinter's threading limitations.
- The `log_event` method should handle cases where the GUI may be closed or not available.
- Testing can be done manually or with automated scripts, using a framework like `unittest` for non-GUI components.

GUI

Initial


Smart Home IoT Simulator
Toggle Automation System
Automation Status: OFF
Living Room Light: SmartLight Status: Off
Living Room Thermostat: Thermostat Status: Off
Front Door Camera: SecurityCamera Status: Off
Living Room Light
0
Toggle OFF
Main Thermostat
10
Toggle OFF
Entrance Camera
Motion Detection
Toggle Streaming
Log

In process



Data Output

The Smart Home IoT Simulator includes functionality to record and output data to a `.csv` file when certain events occur, such as motion detection. This document outlines the implementation and usage of this feature.

Data Output Feature

Sensor Data Logging

When the system detects motion through a `SecurityCamera` device, it logs the event data to a CSV file named `sensor_data.csv`. The data includes the timestamp of the event, the device ID, the device name, and the details of the event (motion detected).