

Documentation

Python assignment IoT

Yehor Danilenko

HQ4EGB

Group 7

Task description

You will need to develop a Python-based IoT simulator for a smart home automation system. The simulator should emulate the behaviour of various IoT devices commonly found in a smart home, such as smart lights, thermostats, and security cameras. You will also create a central automation system that manages these devices and build a monitoring dashboard to visualise and control the smart home. This assignment will help you apply your Python programming skills, including OOP, data handling, real-time data monitoring, and graphical user interfaces (GUIs).

Part 1: IoT Device Emulation (25 %)

- **Device Classes:** Create Python classes for each type of IoT device you want to simulate, such as SmartLight, Thermostat, and SecurityCamera. Each class should have attributes like device ID, status (on/off), and relevant properties (e.g., temperature for thermostats, brightness for lights, and security status for cameras).
- **Device Behavior:** Implement methods for each device class that allow for turning devices on/off and changing their properties. Simulate realistic behavior, such as gradual dimming for lights or setting temperature ranges for thermostats.
- **Randomization:** Include a randomization mechanism to simulate changing device states and properties over time.

Part 2: Central Automation System (25 %)

- **Automation System Class:** Create a central automation system class, e.g., AutomationSystem, responsible for managing and controlling all devices. It should provide methods for discovering devices, adding them to the system, and executing automation tasks.
- **Simulation Loop:** Implement a simulation loop that runs periodically (e.g., every few seconds) to trigger automation rules, update device states, and simulate device behaviors.

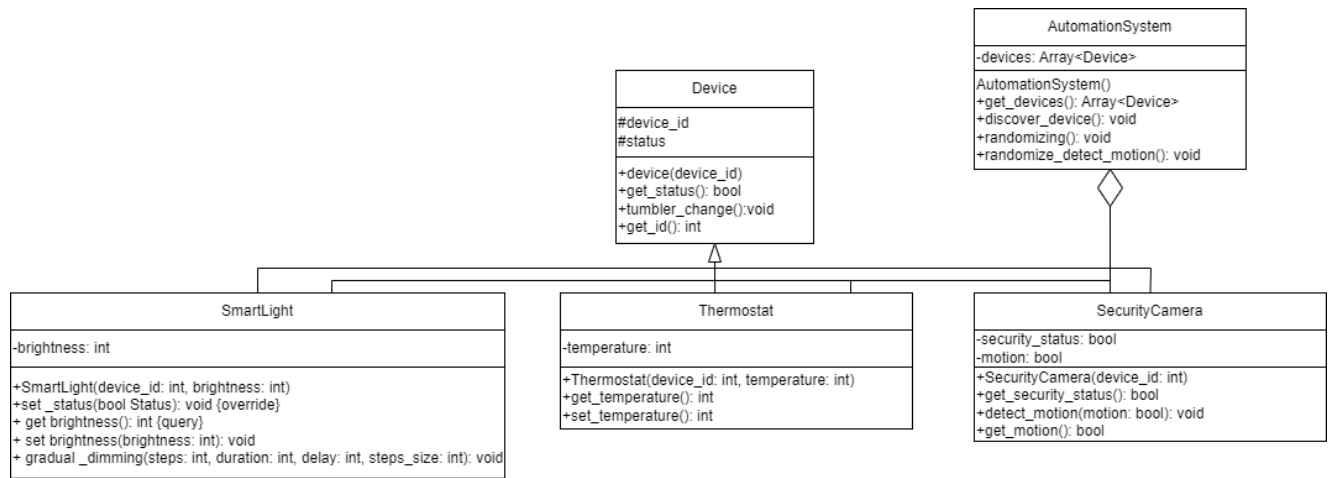
Part 3: Documentation (30%)

- **Documentation:** Provide clear documentation for your code, including class descriptions, method explanations, and instructions on how to run the simulation and use the dashboard.
- **Develop test cases** to ensure that the simulator and automation system behave as expected. Test various scenarios, such as different automation rules and user interactions

Part 4: Monitoring Dashboard (20%)

- **Graphical User Interface (GUI):** Create a GUI for monitoring and controlling the smart home system. You can use Python GUI libraries like Tkinter. The GUI should display the status and properties of each device, provide controls to interact with them, and visualize data.
- **Real-time Data Monitoring:** Display real-time data from the simulated devices on the dashboard. This includes temperature graphs for thermostats, motion detection status for cameras, and brightness levels for lights.
- **User Interaction:** Allow users to interact with devices through the GUI, such as toggling lights on/off, adjusting thermostat settings, and arming/disarming security cameras.

- UML class diagram



The short description of classes and explanations of methods

- Class Device

`__init__(self, device_id)`: Constructor method to initialize a Device object and set its status to "False".

`get_status(self)`: Return the current status of the device, representing the device status (True for on, False for off).

`tumbler_change(self)`: Toggles the status of the device from on to off or vice versa.

`get_id(self)`: Returns the id of the device.

- Class Thermostat

`__init__(self, device_id)`: Constructor method to initialize a Thermostat object with a given id and temperature.

`get_temperature(self)`: Return the current temperature set on the thermostat.

`set_temperature(self, temperature)`: Sets the temperature on the thermostat. If the thermostat is currently off, it is turned on (status changes to True). Sets the temperature of the thermostat to the provided value.

- Class SmartLight

`__init__(self, device_id)`: Constructor method to initialize a SmartLight object with a given id and brightness.

`get_brightness(self)`: Return the current brightness of the smart light.

`set_status(self, status)`: Boolean value representing the new status of the light (True for on, False for off).

`set_brightness(self, brightness)`: Sets the brightness of the smart light.

`gradual_dimming(self, steps, duration, delay, step_size)`: Method to gradually dim the light by reducing its brightness in a series of steps over a specified duration. It also turns off the light after dimming to zero.

- Class SecurityCamera

`__init__(self, device_id)`: Constructor method to initialize a SecurityCamera object.

`get_security_status(self)`: Return the security status of the camera.

`get_motion(self)`: Return the motion detection status of the camera.

`detect_motion(self, motion)`: Updates the motion detection status of the camera.

- Class AutomationSystem

`__init__(self)`: Constructor method to initialize an AutomationSystem object

`get_devices(self)`: Return the list of discovered devices.

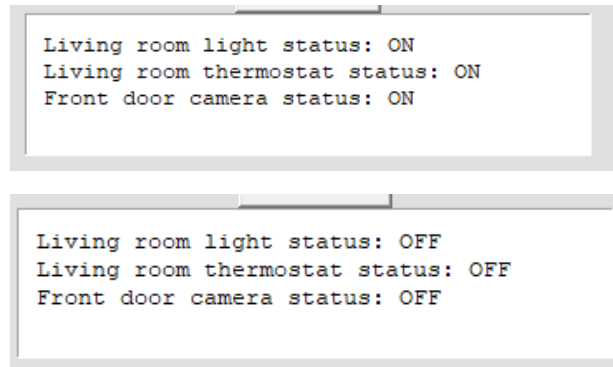
`discover_device(self)`: Discovers and adds predefined devices (SmartLight, Thermostat, SecurityCamera) to the system.

`randomizing(self)`: Randomizes the parameters of the discovered devices.

`randomize_detect_motion(self)`: Randomizes the motion detection status specifically for SecurityCamera devices.

Tested cases

Test toggling on and off the devices:



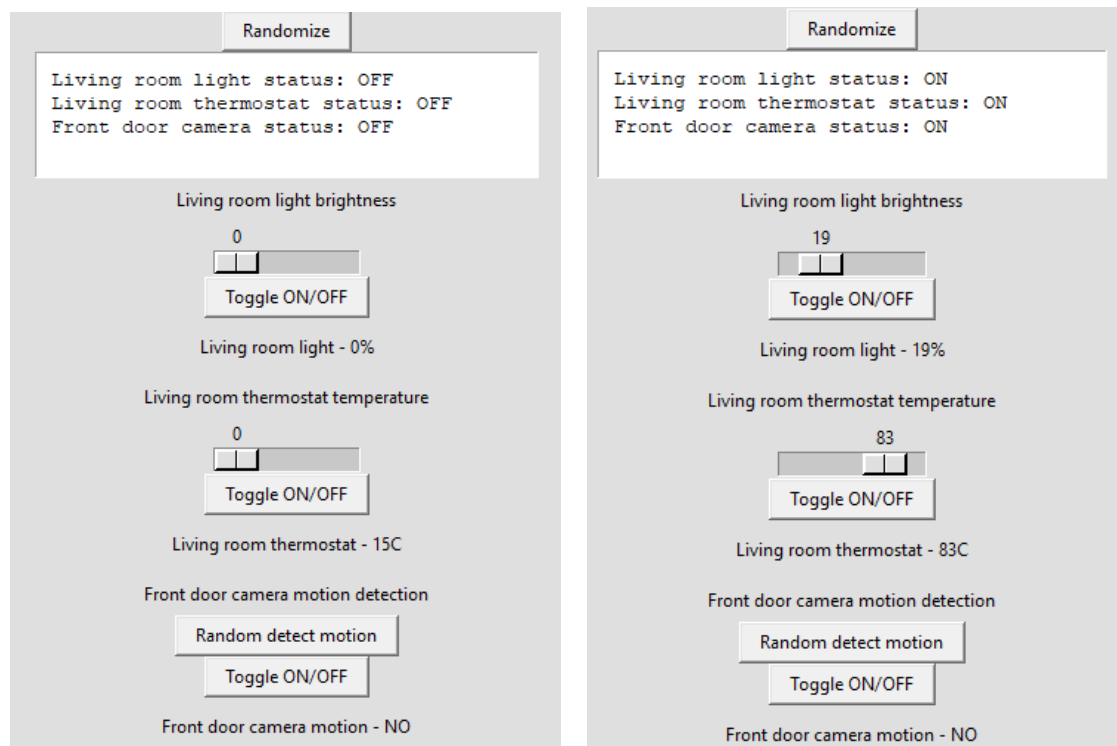
The first screenshot shows the status of three devices: Living room light status: ON, Living room thermostat status: ON, and Front door camera status: ON. The second screenshot shows the same three devices, but all are now OFF.

```
Living room light status: ON
Living room thermostat status: ON
Front door camera status: ON
```

```
Living room light status: OFF
Living room thermostat status: OFF
Front door camera status: OFF
```

Test adjusting values and turning on caused by adjusting values:

On the start all devices are turned off. Tap on “Randomize” to randomize values for the devices what has to adjust random values and turn on the devices.



The left screenshot shows the interface after the 'Randomize' button was pressed, with all devices still OFF. The right screenshot shows the same interface, but the 'Randomize' button has been pressed again, and the devices are now ON with randomized values.

Left Screenshot (All OFF):

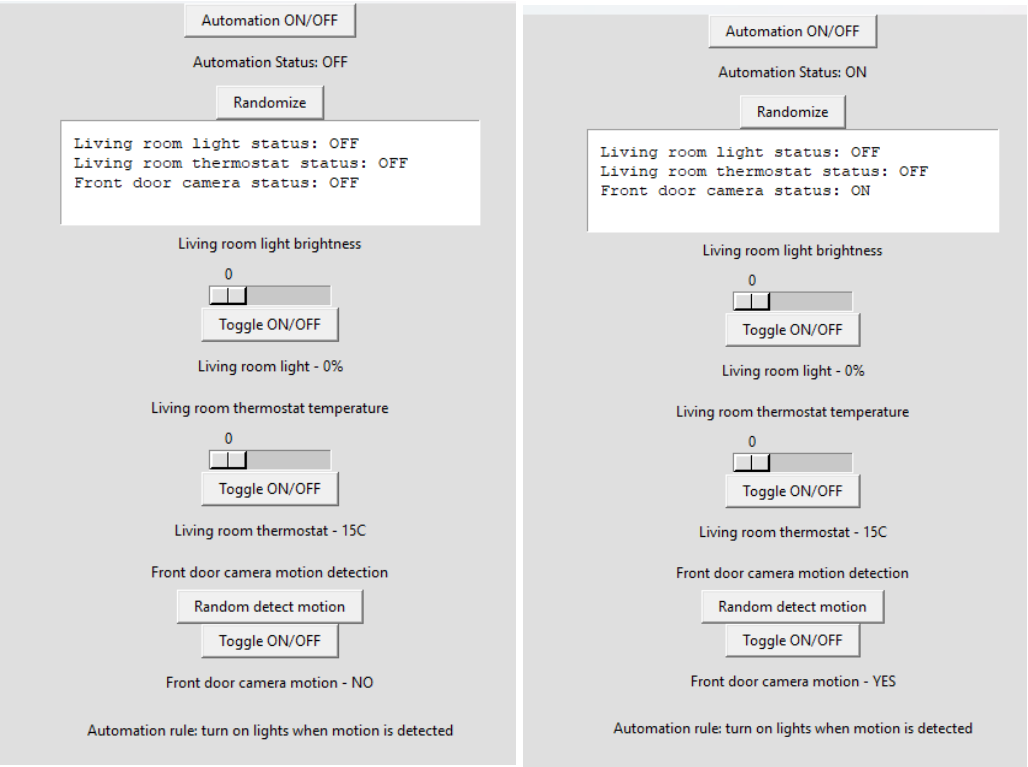
- Living room light status: OFF
- Living room thermostat status: OFF
- Front door camera status: OFF
- Living room light brightness: 0
- Living room light - 0%
- Living room thermostat temperature: 0
- Living room thermostat - 15C
- Front door camera motion detection: Random detect motion
- Front door camera motion - NO

Right Screenshot (All ON):

- Living room light status: ON
- Living room thermostat status: ON
- Front door camera status: ON
- Living room light brightness: 19
- Living room light - 19%
- Living room thermostat temperature: 83
- Living room thermostat - 83C
- Front door camera motion detection: Random detect motion
- Front door camera motion - NO

Test automation

On the start all devices are turned off. Tap on “Automation ON/OFF” to turn on the automation which turns on the lights if the motion is detected by the camera. Tap multiple times on “Random detect motion” until the motion will be randomly detected



Test gradual dimming for the light brightness.

On the start all devices are turned off. Toggle brightness bigger than zero for light what has to cause adjusting new brightness and turning on the light. Tap on “Toggle ON/OFF”. It has to cause gradual dimming for the light brightness what means that brightness will gradually fall during some time until hitting 0% what is supposed to cause turning off the light.

