**PYTHON-BASED SMART HOME IOT MONITORING SYSTEM: BEHAVIORAL ANOMALY DETECTION IN IOT DEVICES**

**Overview**

This project is a simulation of a smart home IoT system built using Python. It includes a central controller/hub and simulated IoT devices like smart lights, thermostats, and security cameras. The goal is to investigate behavioral anomalies in IoT devices to determine if such anomalies can indicate potential device compromise or malfunction.

**Hypothesis (Behavior - B from ABCDE Model)**

**"Can deviations in smart device behavior (e.g., lights, thermostat, cameras) be detected through behavioral modeling, and do these anomalies indicate a potential compromise or malfunction?"**

**Objective**

To design and implement a system that models the normal behavior of IoT devices and detects deviations as behavioral anomalies, demonstrating the practicality and effectiveness of behavior-based monitoring for enhancing IoT security

**System Components**

**1. Devices (device.py)**

* Smart Light
* Thermostat
* Security Camera

Each device has basic attributes like state (ON/OFF), functional parameters (e.g., brightness or temperature), and methods for interaction.

**2. Controller (controller.py)**

* Acts as a hub that manages device interactions.
* Sends and receives commands to/from devices.
* Logs behavior for analysis.

**3. Monitor (monitor.py)**

* Logs and analyzes the behavior of devices.
* Compares current behavior against learned baseline.
* Flags anomalies.

**4. GUI (main.py)**

* Uses Tkinter for a simple dashboard.
* Allows users to toggle device states and adjust parameters.
* Shows real-time status and anomalies.

**System Design Overview:**

**1. System Architecture**

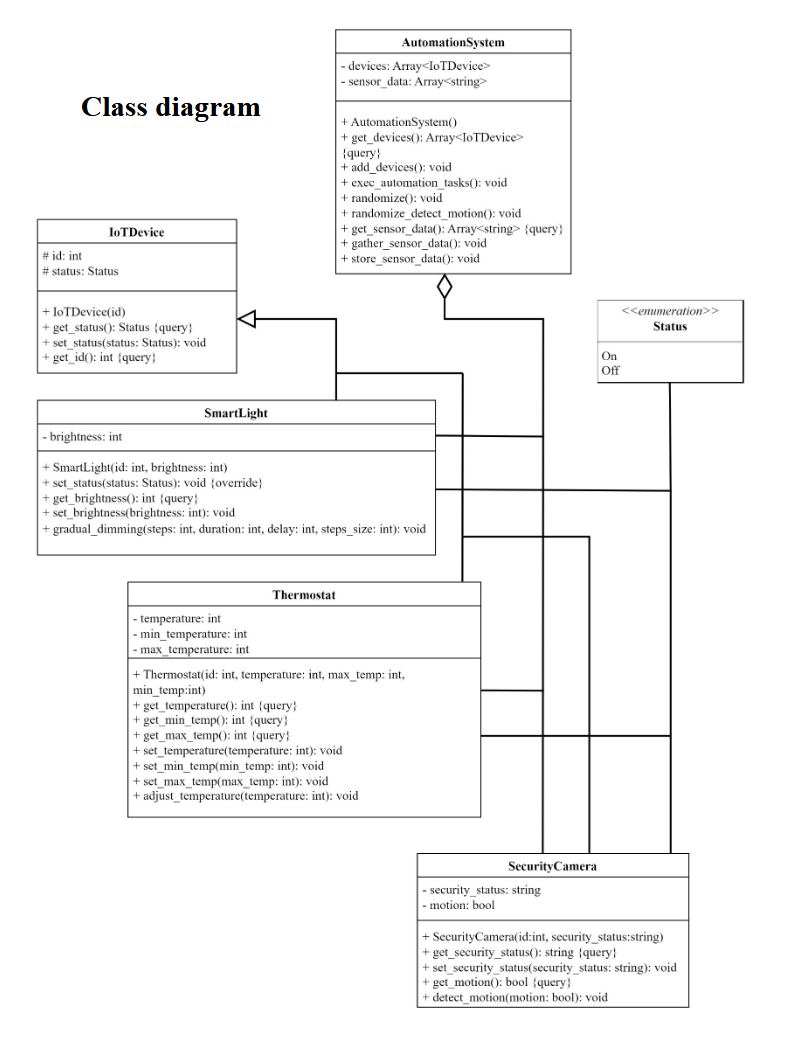
* **Devices**: Simulated smart home devices (light, thermostat, security camera).
* **Controller (Hub)**: Central system that issues commands and monitors behavior.
* **Dashboard GUI**: Visualization tool for status, control, and behavior logs.

**2. Behavior Model**

* Define “normal” behavior ranges and rules for each device.
* Monitor and log real-time behavior and compare against expected behavior.
* Flag deviations as **anomalies**.

**3. Anomaly Scenarios (Simulated)**

* Thermostat randomly switches temperature without command.
* Light toggles ON/OFF unusually frequently.
* Camera stops streaming when it should be active.



**Experimental Design**

**Experiment 1: Baseline Recording**

**Objective:** Establish normal operating range. **Method:** Run system without anomalies. **Expected Output:** Baseline data for lights, thermostat, and camera.

**Experiment 2: Behavior Anomaly Injection**

**Objective:** Test anomaly detection. **Method:** Randomly inject incorrect values/commands. **Expected Output:** Alerts raised for unusual activity.

**Experiment 3: False Positive Check**

**Objective:** Ensure threshold tolerance. **Method:** Normal variation within range. **Expected Output:** No anomalies falsely reported.

**Technologies Used:**

* **Language**: Python 3.3
* **Libraries**: tkinter (GUI), time, random, json
* **Structure**:
  + main.py – Runs GUI and simulator
  + device.py – Device logic
  + controller.py – Hub/controller logic
  + monitor.py – Behavior logging and anomaly detection
  + experiments/ – Contains anomaly injection scripts and test runners

**How to use the dashboard**

GUI allows monitoring of the smart home system through several text fields which are implemented on the dashboard. For example, to see the current status (“ON” or “OFF”) of all the devices it is enough to check the text box which goes right after the “randomize” button on the top. To check the current state of core feature of each device, for instance brightness of the light, it is enough to look at the text right after the button “Toggle ON/OFF”.

For controlling the smart home system there are located different buttons and scales. For example, to turn on the automation you can tap on the first button on the top of the page called “Automation ON/OFF”. To adjust different things to devices and change their statuses you can interact with buttons and scales which were created for each device. Hence, to change the brightness of the light you can pull the scale which is under the text “Living room light brightness”.

**Testing**

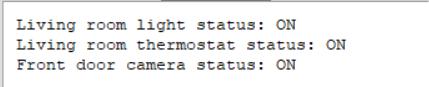
**1. Test toggling on and off the devices**

**Description:**

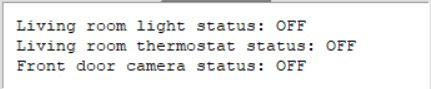
On the start all devices are turned off. Tap on three buttons “Toggle ON/OFF” for each device to turn on the devices. Tap on the buttons once again to turn off the devices.

**Result:**

After the first taps all three devices are turned on:



After the second taps all three devices are turned off:



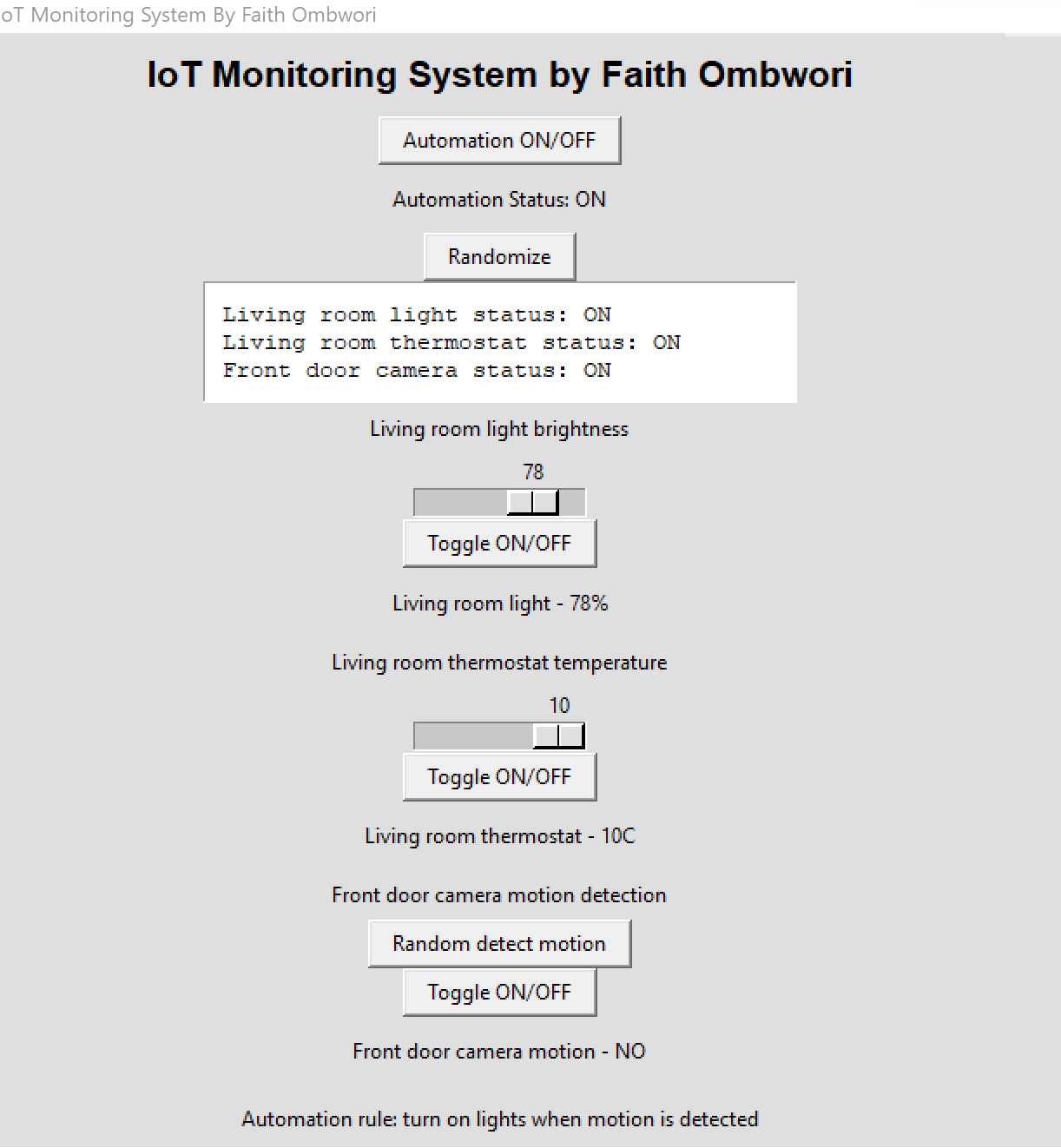
**2. Test adjusting values and turning on caused by adjusting values**

**Description:**

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 lights. Toggle thermostat temperature what has to cause adjusting new temperature and turning on the thermostat. Tap on “Random detect motion” what has to cause adjusting new motion detection status and turning on the camera.

**Result:**

All three devices are turned on. All values are adjusted.



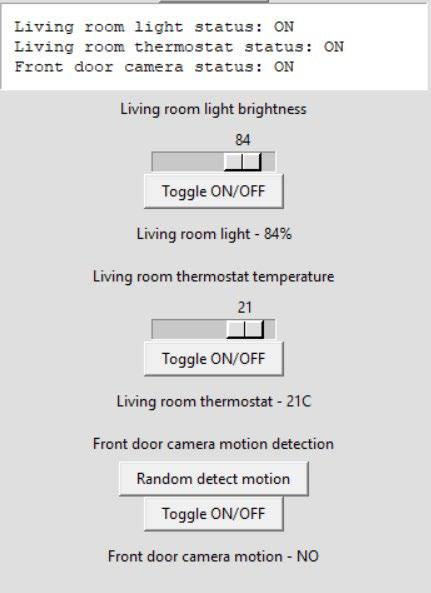
**3. Test randomizing**

**Description:**

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.

**Result:**

All three devices are turned on. Light received random brightness. Thermostat received random minimum temperature, maximum temperature and current temperature. Camera received random motion detection.



**4. Test automation**

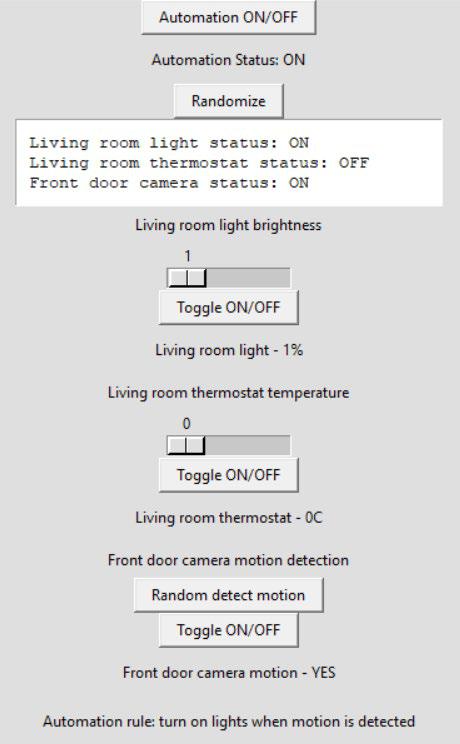
**Description:**

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 what has to cause turning on the lights and adjusting 1% brightness.

Light cannot be turned off while motion is detected. Tap on “Toggle ON/OFF” for the light to try turning it off what has to do nothing.

**Result:**

Camera and light are turned on, light has 1% brightness. While motion is detected turning off the light is not possible.



**5. Test gradual dimming for the light brightness.**

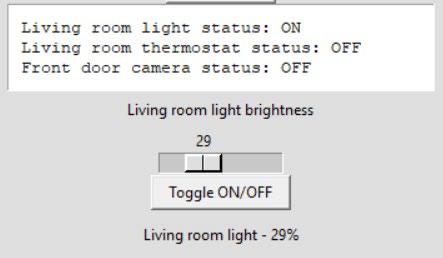
**Description:**

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.

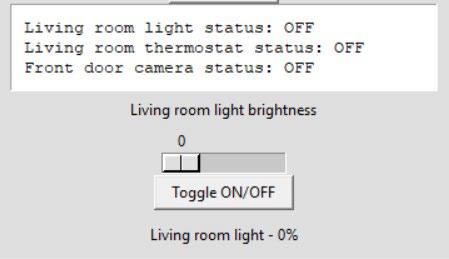
**Result:**

After toggling the light brightness bigger than zero light turned on. Tap on “Toggle ON/OFF” caused gradual dimming where the brightness felt to 0 and light was turned off.

After toggling the light brightness bigger than zero the light is turned on:



After the gradual dimming caused by tap on “Toggle ON/OFF” light is turned off:



**Security and Distributed System Features Demonstrated:**

* Behavioral **anomaly detection** for compromised devices.
* Logging and **event tracking** for accountability.
* GUI for **monitoring system status**.
* **Code modularity** for scalability to more devices.

**Conclusion:**

Behavior-based anomaly detection is a viable approach for identifying threats in smart home environments. Through simple rule-based monitoring and a behavioral profile per device, we demonstrated the ability to detect unusual behavior indicating potential compromise, misconfiguration, or failure.  
Future improvements could include:

* Machine learning for dynamic behavior profiling.
* Integration with external threat databases.
* Network-level behavior monitoring

README

## How to Run

1. Install Python 3.x.
2. Run `python main.py`.
3. Use GUI to interact and view logs.
4. Run tests in `experiments/` to see anomaly detection in action.

REFERENCES

<https://www.researchgate.net/publication/383312427_IoT_Virtual_Simulator>

<https://ieeexplore.ieee.org/document/9716199>

Network Simulation Tools. (n.d.). IoT Network Simulator. Retrieved June 4, 2025, from <https://networksimulationtools.com/iot-network-simulator/>