Task

Developing a Python-based IoT simulator for a smart home automation system. The simulator should emulate the behavior of various IoT devices commonly found in a smart home, such as smart lights, thermostats, and security cameras. Central automation system manages these devices and build a monitoring dashboard to visualize and control the smart home. Used skills: Python programming skills, including OOP, data handling, real-time data monitoring, and graphical user interfaces (GUIs).

IoT Device Classes Documentation

`IoTDevice` Class

Overview

The `IoTDevice` class represents a generic IoT device with basic functionality such as turning on, turning off, and toggling its status.

Constructor

def __init__(self, device_id: str)

- device_id (str): A unique identifier for the IoT device.

Methods

- turn_on() Turns on the IoT device.
- turn_off() Turns off the IoT device.
- toggle() Toggles the status of the IoT device between on and off.

SmartLight Class Overview

The SmartLight class is a subclass of IoTDevice and extends its functionality to include brightness control.

Constructor

def __init__(self, device_id: str, brightness: int = 50)

- device_id (str): A unique identifier for the smart light.
- brightness (int): Initial brightness level, default is 50.

Methods

- set_brightness(brightness: int) Sets the brightness level of the smart light.
- gradual_dimming() Gradually dims the smart light in 10% increments.
- toggle() Toggles the status of the smart light.

Thermostat Class Overview

The Thermostat class is a subclass of IoTDevice and represents a thermostat with temperature control functionality.

Constructor

def __init__(self, device_id: str, temperature: int = 20)

- device_id (str): A unique identifier for the thermostat.
- temperature (int): Initial temperature setting, default is 20°C.

Methods

- set_temperature(temperature: int) Sets the temperature of the thermostat.
- temperature (int): The temperature setting (10°C to 30°C).
- set_temperature_range(lower_limit: int, upper_limit: int) Sets the temperature range of the thermostat.
- lower_limit (int): The lower temperature limit (10°C to 30°C).
- upper_limit (int): The upper temperature limit (10°C to 30°C).
- toggle() Toggles the status of the thermostat.

SecurityCamera Class Overview

The SecurityCamera class is a subclass of IoTDevice and represents a security camera with motion detection.

Constructor

def __init__(self, device_id: str)

- device_id (str): A unique identifier for the security camera.
- Attributes motion (bool): Represents the motion detection status.

Methods

- toggleMotion() Toggles the motion detection status of the security camera.
- toggle() Toggles the status of the security camera.

Automation System Documentation

Overview

The `AutomationSystem` class represents an automation system that manages IoT devices, handles device discovery, and simulates random events within a specified interval. The system also has the capability to save sensor data to a file.

Class: `AutomationSystem`

Constructor

def __init__(self)

Initializes the AutomationSystem with an empty list of devices, an empty list for sensor data, a threading lock for synchronization, a threading event to control the automation loop, and sets the automation state to enabled by default.

Methods

- toggle_automation(enable: bool = True): Toggles the state of automation.
- enable (bool): If True, enables automation; if False, disables automation.
- discover_device(device: IoTDevice): Discovers a new IoT device and adds it to the system.
- device (IoTDevice): The IoT device to be discovered.
- add_device(device: IoTDevice): Adds an IoT device to the system if it's not already present.
- device (IoTDevice): The IoT device to be added.

- simulate_randomization(): Simulates random events for IoT devices, such as motion detection for security cameras and turning on lights.

Motion events trigger sensor data recording and may turn on lights.

- save_sensor_data_to_file(filename: str = "sensor_data.json"): Saves the recorded sensor data to a JSON file.
- filename (str): The name of the file to save the sensor data. Default is "sensor_data.json".

Function: automation_loop(system: AutomationSystem, interval: int = 5):

A separate thread function that runs the automation loop at regular intervals.

- system (AutomationSystem): The AutomationSystem instance to run the loop on.
- interval (int): The time interval between automation loop iterations. Default is 5 seconds.

Automation System GUI Documentation

Overview

The `AutomationGUI` class represents a graphical user interface (GUI) for interacting with an `AutomationSystem`. It allows users to toggle automation, view the status of IoT devices, and control SmartLights, Thermostats, and SecurityCameras.

Class: `AutomationGUI`

Constructor

def __init__(self, automation_system: AutomationSystem)

Initializes the GUI with a reference to an AutomationSystem.

Creates the main window and GUI components.

Starts the automation loop in a separate thread.

Methods

- create_widgets(): Creates various GUI components, including buttons, labels, text boxes, and sliders, for controlling and monitoring IoT devices.
- add_device_control(device: IoTDevice): Creates a control frame for a specific IoT device and adds it to the GUI.
- device (IoTDevice): The IoT device for which the control frame is created.
- update_after_release(event, device, label): Updates the GUI after releasing a slider (brightness or temperature).

- event: The event triggered by releasing the slider.
- device (IoTDevice): The IoT device associated with the slider.
- label: The label to be updated with the current value.
- toggle_automation(): Toggles the state of automation in the AutomationSystem.

- Updates the GUI and logs the automation state.
- toggle_device(device: IoTDevice)
- Toggles the state of a specific IoT device.

- Updates the GUI and logs the device status.
- change_brightness(device: SmartLight, value: float)
- Changes the brightness of a SmartLight device.

- Updates the GUI and logs the brightness change.
- change_temperature(device: Thermostat, value: float): Changes the temperature of a Thermostat device.

- Updates the GUI and logs the temperature change.
- update_gui_periodic(): Periodically updates the GUI elements. Schedules the next update.

- update_gui(log_entry: str): Updates the GUI with the provided log entry or general information. Updates the automation status label, device status text box, and log text box. Scrolls the log text box to the bottom.