# Smart Home IoT Automation Simulator

# Szerző

Madár Bálint MRPBLG

[Mrpblg@inf.elte.hu](mailto:Mrpblg@inf.elte.hu)

# Felhasználói dokumentáció

## Feladat

Develop 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. You will also create a central automation system that manages these.

Devices and build a monitoring dashboard to visualize and control the smart home.

## Környezet

A program Python3 nyelven készült, Tkinter grafikus felülettel.

## Használat

A program egy grafikus felületen keresztül kezelhető, ahol a felhasználó interaktívan kapcsolhatja be vagy ki az eszközöket, állíthatja be a hőmérsékletet vagy a világítás fényerejét.

A program diagrammon mutatja az eszközök aktuális állapotát, illetve lehetőség van random szimulációra.

## A program indítása

python3 Smart Home IoT Automation Simulator.py

## A program bemenete

A felhasználó interaktív bemeneteket ad meg a grafikus felületen keresztül.

## Egy lehetséges párbeszéd

Felhasználó: Kapcsolja be a nappali termosztátot.

Program: A GUI jelzi, hogy a nappali termosztát bekapcsolva.

A screenshot of a computer

Description automatically generated

## Egy lehetséges kimenet, az első két feladatra

Simulation Time: 0.00 seconds

Device: Thermostat\_01, Type: Thermostat, Status: ON, Temperature: 25

Simulation Time: 1.01 seconds

Device: Thermostat\_01, Type: Thermostat, Status: ON, Temperature: 20

Simulation Time: 2.01 seconds

Device: Thermostat\_01, Type: Thermostat, Status: ON, Temperature: 19

Duration: 3 Interval: 1

Simulation Time: 0.00 seconds

Device: Camera\_01, Type: SecurityCamera, Status: ON, Security Status: MOTIONDETECTION

Simulation Time: 1.00 seconds

Device: Camera\_01, Type: SecurityCamera, Status: ON, Security Status: MOTIONDETECTION

Simulation Time: 2.01 seconds

Device: Camera\_01, Type: SecurityCamera, Status: ON, Security Status: MOTIONDETECTION

#PART 2

Device Part 2 Light\_02 added.

Device Part 2 Thermo\_2 added.

Device Part 2 Camera\_02 added.

Simulation Duration: 10 seconds, Interval: 1 second(s)

Simulation Time: 0.00 seconds

SmartLight Part 2 Light\_02 - Status: ON, Brightness: 66, Last Updated: 2023-10-31 11:20:30.234372

Thermostat Part 2 Thermo\_2 - Status: ON, Temperature: 30, Last Updated: 2023-10-31 11:20:30.234482

Security Part 2 Camera\_02 - Status: ON, Temperature: SecurityStatus.MOTIONDETECTION, Last Updated: 2023-10-31 11:20:30.234501

Simulation Time: 1.00 seconds

SmartLight Part 2 Light\_02 - Status: ON, Brightness: 81, Last Updated: 2023-10-31 11:20:31.236493

Thermostat Part 2 Thermo\_2 - Status: ON, Temperature: 23, Last Updated: 2023-10-31 11:20:31.236573

Security Part 2 Camera\_02 - Status: ON, Temperature: SecurityStatus.MOTIONDETECTION, Last Updated: 2023-10-31 11:20:31.236611

Simulation Time: 2.01 seconds

SmartLight Part 2 Light\_02 - Status: ON, Brightness: 50, Last Updated: 2023-10-31 11:20:32.239723

Thermostat Part 2 Thermo\_2 - Status: ON, Temperature: 25, Last Updated: 2023-10-31 11:20:32.239808

Security Part 2 Camera\_02 - Status: ON, Temperature: SecurityStatus.MOTIONDETECTION, Last Updated: 2023-10-31 11:20:32.239843

## 

## Hibalehetőségek

* Hibás bemeneti értékek, a csúszka nem megfelelően regisztrálja a bemenetet.
* A felhasználó hibásan konfigurálja az eszközöket.
* Téves adatbevitel, például nem megfelelő hőmérséklet vagy fényerő beállítása.
* Memória kezelési hiba

## 

# Fejlesztői dokumentáció

## Specifikáció és környezet

Tkinter grafikus felület

Eszközök: okos világítás, termosztát, biztonsági kamera

Környezet python 3.8+, Tkinter, Matplotlib (grafikonok megjelenítéséhez)

## Fontos Típusok, Változók

SmartLight, Thermostat, SecurityCamera: Eszköz osztályok

DeviceStatus, SecurityStatus: Enum típusok az eszközök állapotának kezelésére

Programfelépítés

Az alkalmazás egyetlen Python fájlból áll, amely tartalmazza az összes osztályt és a grafikus felületet.

## Osztályok és függvények

**SmartLight**:

Okoslámpa kezelése.

\_\_init\_\_(self, device\_id): Konstruktor, beállítja az eszköz azonosítóját és kezdeti fényerejét.

set\_brightness(self, brightness): Beállítja a fényerőt.

**Thermostat:**

Feladata: Termosztát kezelése.

Metódusok:

\_\_init\_\_(self, device\_id): Konstruktor, beállítja az eszköz azonosítóját és kezdeti hőmérsékletét.

set\_temperature(self, temperature): Beállítja a hőmérsékletet.

**SecurityCamera:**

Feladata: Biztonsági kamera kezelése.

Metódusok:

\_\_init\_\_(self, device\_id): Konstruktor, beállítja az eszköz azonosítóját és a biztonsági állapotát.

trigger\_alarm(self): Riasztást vált ki.

reset\_alarm(self): Visszaállítja a normál állapotot.

Függvények és Leírásuk:

toggle\_light\_gui(): Kapcsolja a lámpát.

update\_light\_label(): Frissíti a lámpa fényerejének címkéjét a felhasználói felületen.

toggle\_thermostat(): Kapcsolja a termosztátot.

update\_thermostat\_label(): Frissíti a termosztát hőmérsékletének címkéjét.

toggle\_camera(): Kapcsolja a kamerát.

toggle\_camera\_mode(): Vált a kamera módok között.

update\_camera\_mode\_button(): Frissíti a kamera mód gombot.

toggle\_cam\_onoff(): Bekapcsolja vagy kikapcsolja a kamerát.

update\_status\_label(): Frissíti az eszközök állapotának címkéjét.

update\_camera\_onoff\_button(): Frissíti a kamera be-/kikapcsolás gombot.

monitor\_random\_data(): Véletlenszerű adatokat generál és megjelenít grafikonokon.

monitor\_real\_time\_data(smart\_light, thermostat, security\_camera): Valós idejű adatokat jelenít meg grafikonokon a különböző eszközökről.

## Megoldás

A programot moduláris felépítéssel készült, ahol minden eszköz egy osztályban van implementálva. A grafikus felület a Tkinter könyvtárat használja.

Fontos típusok, változók

Programfelépítés

## Tesztelés

l

## Fejlesztési lehetőségek

* További eszközök hozzáadása a grafikus felületen
* Okoseszközök elvétele a grafikus felületről
* Okoseszköz vásárlás a grafikus felületen
* További funkciók az eszközökről, például elő videó