

Félévi beadandó feladat

Okos szoba szenzorhálózat

Készítette: Teker Balázs és Nyiredi Balázs

Neptun-kód: FIVCQC, A0SPYC

I. Felhasználói Dokumentáció

1. A Feladat Megfogalmazása

A beadando_version5 egy konzolalkalmazás, amely egy szimulált okosszoba környezeti adatait (hőmérséklet, páratartalom, légnyomás) figyeli és kezeli. A program egy 24 órás időszakot szimulál, percenkénti lépésekben, és a generált adatokat egy SQLite adatbázisba menti. Ezt követően a program betölti az adatokat az adatbázisból, elemezi azokat LINQ lekérdezésekkel, és az eredményeket egy JSON fájlba írja. A szimuláció a C# Delegált és Eseménykezelés mechanizmusait használja a szenzor adatok szimulált változásának és a kritikus állapotok jelzésének kezelésére.

2. Bemenet és Várt Kimenet Leírása

Bemenet:

- Külső bemenet: Nincs. A program nem igényel felhasználói bemenetet vagy külső bemeneti fájlt.
- Belső bemenet: A szimuláció kezdőértékei véletlenszerűen generálódnak és minden kezdőérték egy intervallumon generálódik. A szimulált változásokat (percenként) a C# Random osztály segítségével generálja a Okosszoba osztály.

Várt Kimenet:

A program futása két fő kimenetet eredményez:

1. Konzol kimenet:
 - a. Fázisok jelzése (adatgenerálás, elemzés).
 - b. Az első 5 szimulációs lépés mérési adatai kiíródnak.
 - c. Kritikus páratartalom elérése esetén figyelemfelhívás (piros színnel).
 - d. 3 darab LINQ elemzés

Fájl kimenet (Mért adatok tárolása):

- e. SQLite adatbázis: Tartalmazza a teljes 1440 percnyi (24 óra) mérési adatot.
- f. JSON fájl: Tartalmazza az adatbázisból betöltött összes mérési eredményt

3. Rövid Kezelési Útmutató

1. Telepítés: Győződjön meg róla, hogy a .NET 6.0 vagy újabb Runtime telepítve van a rendszeren.
2. Futtatás: Futtassa a fordított .exe fájlt a parancssorból, vagy indítsa el az IDE-ből (Visual Studio, VS Code, Jetbrains Rider):
3. Működés:
 - a. A program automatikusan futtatja a szimulációt (, generálja az adatokat és eltárolja azokat a szoba_adatok.db adatbázisban).
 - b. Ezt követően betölti az adatokat az adatbázisból.
 - c. Elvégzi a LINQ elemzéseket, az eredményeket a konzolra írja.
 - d. Létrehozza a SzobaMeresek.json fájlt a futtatható .exe melletti mappában.
4. Kilépés: A futás befejeztével a program megvárja, amíg egy gombot megnyom a felhasználó.

II. Fejlesztői Dokumentáció

Adatkezelő DLL, Class1.cs

1. A projekt felépítése és szerepe

A Adatkezelő névtér tartalmazza az intelligens szobát szimuláló és az érzékelő adatokat kezelő logikát. Ez a komponens biztosítja az alapot az adatok generálásához és tárolásához, amelyeket a konzol alkalmazás (felhasználói interfész) a későbbiekben felhasznál és kiír.

Fő komponens: Okosszoba osztály.

Szerepe: Szimulálja egy szoba környezeti paramétereinek (hőmérséklet, légnyomás, páratartalom) időbeli változását, kezeli az adatokat, és eseményt generál kritikus helyzetek esetén.

2. Adattagok (A mért paraméterek)

Az osztály tárolja a szoba aktuális állapotát, valamint az összes korábbi mérési pontot:

- Aktuális Adatok (Property-k):
 - Ido (Idő/Lépésszám)
 - Homerseklet (Hőmérséklet)
 - Legnyomas (Légnymás)
 - Paratartalom (Páratartalom)
- Tárolók (Listák):
 - Mindegyik paraméterhez tartozik egy List<T> (dinamikus lista): IdoLista, HomersekletLista, stb.
 - Szerepük: Ezek a listák tárolják az összes mérési pontot, ami a szimuláció során keletkezik (idősoros adatok).

3. Konstruktor és Kezdőértékek

- Okosszoba(...): Ez a metódus fut le az osztály létrehozásakor.
- Bemenet: Négy kezdeti értéket (Idő, Hőmérséklet, Légnymás, Páratartalom) vár.
- Feladata: Beállítja az aktuális property-keket, és hozzáadja ezeket a kezdőértékeket az adattároló listákhoz is.

4. Konstansok és Véletlenszerűség (Szimulációs Beállítások)

Ezek az értékek szabályozzák, hogy mennyit változhatnak az adatok két mérés között:

- _r (Random): Ez generál véletlenszerű változásokat a szimulációhoz.
- Max. Változások: MAX_HOMERSEKLET_VALTOZAS,
MAX_LEGNYOMAS_VALTOZAS, MAX_PARATARTALOM_VALTOZAS.

Szerepük: Ezek határozzák meg a maximális mértéket, amivel egy-egy paraméter növekedhet vagy csökkenhet a következő szimulációs lépésben.

Kritikus Páratartalom: MAX_PARATARTALOM (80.0). Ez a küszöb, ami felett a program figyelmeztetést ad.

5. Funkciók (Mérés és Módosítás)

HomersekletBeAllitas() / LegnyomasBeAllitas() / ParatartalomBeAllitas():

- Feladata: Mindegyik metódus véletlenszerűen növeli vagy csökkenti a hozzá tartozó aktuális értéket a maximális változási határon belül.
- Fontos: Az új, frissített értéket mindenhol metódus azonnal hozzáadja a saját listájához.

Eseménykezelés (ParatartalomBeAllitas()-ban):

- Ha a páratartalom eléri a MAX_PARATARTALOM értéket (80.0), akkor kiváltja a KritikusParatartalomElreve nevű eseményt.
- Szerepe: Ez lehetővé teszi a főprogram (a Konzol App) számára, hogy azonnal reagáljon erre a kritikus helyzetre.

6. Delegált és Ciklus (Együttműködés)

Valtozas (Delegált): Ez egyfajta "műveleti lista", ami egyszerre tud több metódust is tárolni és végrehajtani.

Delegalt() Metódus:

- Létrehozza a Valtozas delegáltat.
- Hozzáadja mindenhol beállító metódust (HomersekletBeAllitas, LegnyomasBeAllitas, ParatartalomBeAllitas).
- Hívja a delegáltat (del()), aminek hatására mindenhol metódus egyszerre fut le, szimulálva egy teljes mérési ciklust.

A Konzol Alkalmazás, program.cs

1. Projekt feladata

A program feladata az, hogy szimuláljon egy 24 órás (1440 perces) méréssorozatot, elmentse az adatokat egy adatbázisba, majd elemezze azokat.

2. Fő Struktúra és Függőségek

Függőségek (Importok): A program a saját DLL-jén (Adatkezelő), valamint több külső könyvtáron is múlik:

- Newtonsoft.Json: JSON fájlok írásához.
- Microsoft.Data.Sqlite: Az SQLite adatbázis kezeléséhez.
- System.Linq: Összetett adatelemzésekhez (LINQ lekérdezések).

Fő Metódus (Main): Ez indítja a programot, és két nagy fázisra osztja a munkát, hibakezeléssel ellátva (try-catch blokk).

3. MertAdat Osztály (Adattároló)

- Szerep: Ez az osztály csak az adatok szállítására szolgál a memóriában (Data Transfer Object, DTO). A főprogram ezt használja, amikor kiolvassa az adatbázisból az adatokat, és mielőtt JSON fájlba írná azokat.
- Adattagok: Ugyanazok a mérési adatok, mint a DLL-ben (Homerseklet, Paratartalom, stb...), plusz az adatbázis által generált Ido (lépésszám) és Mentesideje (pontos idő).

4. I. Fázis: AdatokatGeneralEsMent() (Szimuláció és Mentés)

Ez a metódus a szimuláció elindításáért és az adatok perzisztens tárolásáért felel.

- Adatbázis Inicializálás: Meghívja a SqliteAdatkezelő.InitializeDatabase() metódust, ami törli a korábbi adatokat, és létrehozza a tiszta, új adatbázis-táblát.
- Szimuláció Indítása: Létrehozza az Okosszoba osztály egy példányát (a DLL-ből), véletlenszerű kezdőértékekkel (pl. 20-23 °C).
- Eseményre Feliratkozás: Kritikus pont! Feliratkozik a DLL-ből érkező eseményre és ez azt jelenti, hogy ha a DLL-ben a páratartalom kritikus szintet ér el, a főprogram futtatja a Szoba_KritikusParatartalomElreve metódust.

Szimulációs Ciklus: Egy for ciklus fut 1440-szer (24 óra * 60 perc):

- Meghívja a DLL-ből a szoba.Delegalt() metódust (ez generálja az új értékeket).

- Meghívja az SqliteAdatkezelő.AdatBeszuras() metódust (ez menti az új értékeket az adatbázisba).

5. Eseménykezelő: Szoba KritikusParatartalomElérve()

- Szerep: Ez a metódus csak akkor fut le, ha a DLL-ben a páratartalom elérte a 80%-os küszöböt.
- Művelet: Kiír egy feltűnő (piros színű) FIGYELMEZTETÉST a konzolra az aktuális páratartalommal.

6. II. Fázis: AdatokatElemezAdatbazisbol() (Betöltés és Elemzés)

Ez a fázis a feldolgozott adatok kinyerését és elemzését végzi.

- Adatok Betöltése: Meghívja a SqliteAdatkezelő.AdatokatBetolt() metódust, amely a teljes adatbázis-tartalmat beolvassa a memóriába (List<MertAdat>).
- LINQ Elemzések: Három fő elemzést végez az adathalmazon:
 1. Óránkénti átlagok: GroupBy parancsokkal összevonja a 60 percenkénti adatokat, és kiszámítja azok átlagát (Average).
 2. Kritikus időtartam: Where feltétellel megszűri a 75% feletti páratartalmú méréseket, és megszámolja (Count), hogy összesen hány percen át tartott ez az állapot.
 3. Korreláció elemzése: Megkeresi azt az időpontot, ahol a legmagasabb Hőmérséklethez a legalacsonyabb Légnagyomás társul (OrderByDescending + ThenBy).
- JSON Export: A teljes betöltött adathalmazt a Newtonsoft.Json könyvtár segítségével szépen formázott (Formatting.Indented) JSON fájlba (SzobaMeresek.json) írja.

7. Segítő Osztály: SqliteAdatkezelő

- Szerep: Ez az osztály kizárolag az adatbázis-műveletekért felelős (DB Helper).
- InitializeDatabase(): Adatbázis tábla létrehozása.
- AdatBeszuras(): Egy sor adat beszúrása.
- AdatokatBetolt(): Az összes adat lekérdezése és MertAdat listává alakítása.