# Szenzor Szimulációs Rendszer

**Készítette:** Tóth Máté, Szedlár Márk
**Dátum:** 2025.12.8

## I. Felhasználói Dokumentáció

### 1. A feladat megfogalmazása

A szoftver célja egy környezeti szenzorokat (hőmérséklet és páratartalom) szimuláló rendszer megvalósítása. A program automatikusan generál mérési adatokat, amelyeket több formátumban (memória, JSON fájl, LiteDB adatbázis) tárol. A futás végén a program statisztikai adatokat jelenít meg a konzolon, például az átlaghőmérsékletet, valamint listázza az összes mérést érték szerint rendezve.

### 2. Bemenet és a várt kimenet leírása

**Bemenet:**
A program nem igényel felhasználói beavatkozást vagy kézi adatbevitelt futás közben. Az adatok bemenetét a Szenzor osztályban található véletlenszám-generátor (szimuláció) biztosítja:

- **Hőmérséklet:** 20 és 35 °C közötti véletlen egész szám.
- **Páratartalom:** 40 és 80 % közötti véletlen egész szám.

**Kimenet:**
A program háromféle kimenetet állít elő:

1. **Konzol kimenet:**
    - Valós idejű üzenetek az egyes mérések érkezésekor (pl. *"Mérés érkezett: ..."*).
    - Statisztikai számítás eredménye (Átlag hőmérséklet).
    - Rendezett lista a mért értékekről.
2. **Fájl kimenet (JSON):**
    - meresek.json: A program létrehoz (vagy felülír) egy szöveges fájlt, amely szabványos JSON formátumban tartalmazza az összes mérést.
3. **Adatbázis kimenet (LiteDB):**
    - SzenzorAdatok.db: A program egy beágyazott NoSQL adatbázisba menti az egyes méréseket tartós tárolás céljából.

### 3. Rövid kezelési útmutató

1. **Előfeltételek:** A futtatáshoz szükséges a .NET keretrendszer, valamint a program mellé másolt LiteDB.dll és Newtonsoft.Json.dll fájlok (vagy NuGet csomagok helyreállítása).
2. **Indítás:** Indítsa el a SensorApp.exe alkalmazást.
3. **Működés:** A program automatikusan elvégzi a szimulációt (5-5 mérést végez mindkét szenzorral). A folyamat közben a képernyőn megjelennek az adatok.

4. **Bezárás:** A futás végén a program megáll és vár egy billentyűleütésre. A kilépéshez nyomjon meg egy tetszőleges billentyűt.
5. **Adatok megtekintése:** A program mappájában létrejött meresek.json fájl bármilyen szövegszerkesztővel megnyitható.

# II. A projekt felépítése

1. A megoldás (Solution) két projektből áll egy console app és egy class libary-ból.

- **SensorLib (Class Library):** Ez tartalmazza az üzleti logikát és az adatmodelleket.
    - Meres.cs: Az adatmodell osztálya.
    - Szenzor.cs: A szenzor működését és az eseménykezelést megvalósító osztály.
    - SzenzorAdat.cs: Kiegészítő osztály (jelenleg minimális funkcionalitással).
- **SensorApp (Console Application):** A végrehajtható program.
    - Program.cs: A belépési pont (Main metódus). Itt történik a példányosítás, az eseményekre való feliratkozás, a fájlkezelés és a LINQ lekérdezések.

**Külső függőségek:**

- Newtonsoft.Json: JSON szerializációhoz.
- LiteDB: Beágyazott NoSQL adatbázis kezeléséhez.

# 2. Saját fejlesztésű osztályok leírása

## class Meres (SensorLib)

Ez az osztály reprezentál egyetlen mérési rekordot. POCO (Plain Old CLR Object) jellegű.

- **Tulajdonságok:** Id (int), SzenzorNev (string), Tipus (string), Ertek (double), Ido (DateTime).
- **Metódusok:** ToString() felüldefiniálva a könnyebb konzolos megjelenítés érdekében.

## class Szenzor (SensorLib)

A hardveres szenzort szimuláló osztály.

- **Esemény:** public event MeresKezelo MeresTortent. Ez teszi lehetővé, hogy a főprogram reagáljon, amikor a szenzor "mér". Observer tervezési minta alapja.
- **Metódus:** MeresSzimulalas(): Generál egy véletlenszámot a szenzor típusától függően (Hőmérséklet vagy Páratartalom), majd elsüti a MeresTortent eseményt.

## Program (SensorApp)

A vezérlő osztály.

- Létrehozza a Szenzor objektumokat.
- Lambda kifejezésekkel feliratkozik a MeresTortent eseményekre.
- Kezeli a LiteDatabase kapcsolatot a MentesLiteDB metóduson keresztül.

- LINQ lekérdezésekkel szűri (Where) és rendezi (OrderBy) az adatokat.

# 3. Választott adatszerkezetek leírása

1. **Lista (List<Meres>)**:
    - A memóriában történő átmeneti tárolásra szolgál. Dinamikus méretű tömb, amely lehetővé teszi az elemek könnyű hozzáadását (Add) és a LINQ lekérdezések hatékony futtatását.
2. **Esemény és Delegált (Delegate / Event)**:
    - A MeresKezelo delegált és a MeresTortent esemény biztosítja a laza csatolást a szenzor és a főprogram között. A szenzornak nem kell tudnia arról, hogy ki dolgozza fel az adatot (konzol, adatbázis, fájl), ő csak jelzi, hogy adat keletkezett.
3. **NoSQL Dokumentumtár (LiteDB)**:
    - A meresek kollekcióban tároljuk az objektumokat. Ez egy dokumentum-alapú tárolás, amely nem igényel szigorú sémát, és közvetlenül a C# objektumokat képezi le adatbázis rekordokká.
4. **JSON (JavaScript Object Notation)**:
    - Szabványos szöveges adatformátum az adatok hordozhatósága érdekében.
      A List<Meres> tartalmát szerializáljuk egyetlen JSON tömbbé.