Szerverfarm Felügyeleti Rendszer Tervspecifikáció

Halász Olivér

March 31, 2025

1. Feladat leírása

Tervezzen objektummodellt számítógépek üzemeltetését segítő felügyeleti rendszer működésének modellezésére! A modellben legyenek érzékelők (diszk kapacitás, memória kapacitás, processzor terheltség, szerverszoba hőmérséklet, tűzjelző, stb.), logikai kapuk (ÉS, VAGY, NEM), kapcsolók, és vészcsengő! Tetszőlegesen bonyolult modell legyen felépíthető a komponensek és a logikai kapuk egyszerű összekapcsolásával! Demonstrálja a működést külön modulként fordított tesztprogrammal! A megoldáshoz ne használjon STL tárolót!

2. A rendszer céljai

A tervezendő rendszer fő céljai:

- Érzékelők állapotának valós idejű monitorozása.
- Az érzékelők jeleinek logikai összevonása különböző kapuk segítségével.
- Riasztási állapot jelzése előre definiált feltételek teljesülésekor.
- Manuális beavatkozás lehetősége kapcsolók segítségével.
- Tetszőleges komplexitású logikai hálózat felépítése a komponensek összekapcsolásával.

3. Bemenetek és komponensek

A rendszer komponensei az alábbi típusokra oszthatók:

3.1. Sensors (Érzékelők)

Az érzékelők különböző típusai és funkciói:

- **DiskCapacitySensor:** A diszk foglaltságának százalékos értékét méri.
- **MemoryCapacitySensor:** A memória használtságát százalékban méri.
- **CpuLoadSensor:** A processzor terheltségét százalékos formában figyeli.

- **TemperatureSensor:** A szerverszoba hőmérsékletét Celsius fokban méri.
- **FireAlarm:** Bináris jelzést ad: tűz vagy nincs tűz.

Minden érzékelő küszöbérték alapján működik, amely meghatározza az aktuális állapotát.

3.2. Switches (Kapcsolók)

A kapcsolók manuálisan vezérelhetők (be- vagy kikapcsolt állapotban). Ezek állapotát a felhasználó állíthatja be.

3.3. Logical Gates (Logikai Kapuk)

A rendszer logikai kapui az érzékelők és kapcsolók jeleit dolgozzák fel:

- **ANDGate:** Akkor aktív, ha az összes bemenete aktív.
- **ORGate:** Akkor aktív, ha legalább egy bemenete aktív.
- **NOTGate:** A bemenet értékét megfordítja.

A kapuk bemenetei más komponensek kimenetei lehetnek.

3.4. Alarm (Vészcsengő)

A vészcsengő bináris állapotú kimeneti eszköz. Aktiválódik, ha a hozzá kapcsolt logikai komponens aktiválja.

4. UML Osztálydiagram

Az alábbi UML osztálydiagram szemlélteti a rendszer osztályait és azok kapcsolatát:

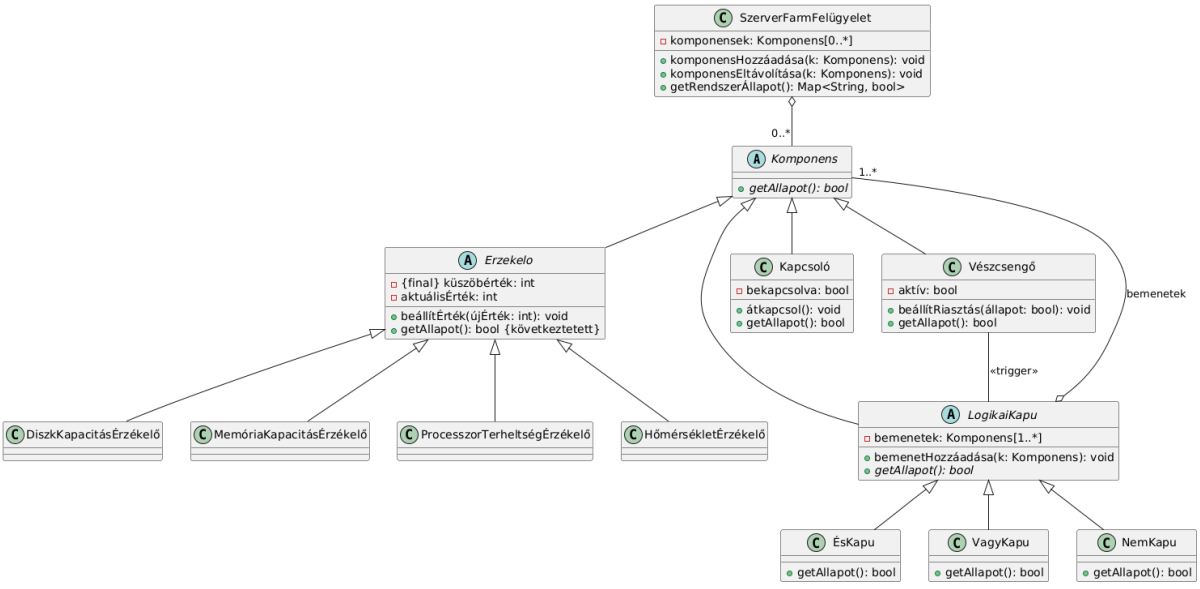

Figure 1: UML osztálydiagram a rendszerhez

5. Komponensek és metódusok leírása

5.1. Sensors

- Attributes: `threshold (int)`, `currentValue (int)`
- Methods:
  - `setValue(newValue: int): void` – Az aktuális érték beállítása.
  - `getState(): bool` – Az érzékelő állapotának lekérdezése (küszöbérték alapján).

5.2. Logical Gates

- Attributes: `inputs (Component[1..*])`
- Methods:
  - `addInput(c: Component): void` – Kapu bemenetének hozzáadása.
  - `getState(): bool` – Kapu aktuális állapotának lekérdezése.

5.3. Alarm

- Attributes: `active (bool)`
- Methods:
  - `setAlarm(state: bool): void` – Vészjelzés beállítása.
  - `getState(): bool` – A vészjelző állapotának lekérdezése.

5.4. Switch

- Attributes: `isOn (bool)`
- Methods:
  - `toggle(): void` – Kapcsoló állapotának változtatása.
  - `getState(): bool` – Kapcsoló állapotának lekérdezése.

6. Fontosabb algoritmusok leírása

6.1. Determining Sensor State (Érzékelők állapotának meghatározása)

- Input: Sensor `currentValue`, `threshold`
- Output: Logical state (`active`/`inactive`)
- Pszeudokód:

```
if currentValue >= threshold:
    state = true
else:
    state = false
```

6.2. Determining Logical Gate State (Logikai kapuk állapotának meghatározása)

- Input: Logical gate input states
- Output: Gate state (`active`/`inactive`)
- Pszeudokód (AND Gate):

```
state = true
for each input:
    if input state == false:
        state = false
        break
```

7. A rendszer tesztelése

A tesztprogram az alábbi funkciókat demonstrálja:

- Érzékelők létrehozása és állapotuk megjelenítése.
- Kapcsolók állapotának módosítása.
- Logikai kapuk összekapcsolása.
- Vészcsengő aktiválása és tesztelése.
- Teljes rendszer állapotának megjelenítése.

8. Következtetések

Az elkészített terv lehetővé teszi egy moduláris, jól strukturált felügyeleti rendszer fejlesztését, amely könnyen bővíthető és tesztelhető.