Szerverfarm Felügyeleti Rendszer Tervspecifikáció

Halász Olivér

2025.04.25

A programozás alapjai II. Nagy házi feladat

Budapesti Műszaki és Gazdaságtudományi Egyetem II. félév

1. Feladat leírása

Szerverfarm

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:

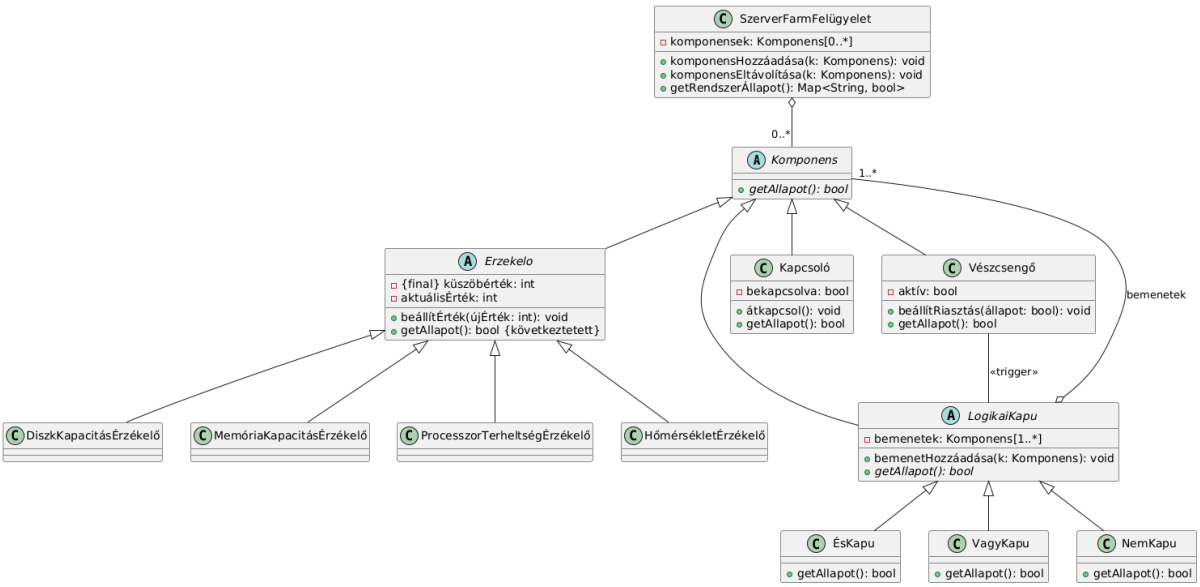

Figure 1: UML osztálydiagram a rendszerhez

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.

Szerverfarm Felügyeleti Rendszer Feladatspecifikáció

Halász Olivér

2025.04.25

A programozás alapjai II. Nagy házi feladat

Budapesti Műszaki és Gazdaságtudományi Egyetem II. félév

1 Feladat leírása

Szerverfarm

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 funkciói

A tervezendő rendszer célja egy olyan felügyeleti rendszer modellezése, amely képes:

- Különböző érzékelők állapotának monitorozására
- Az érzékelők jeleit logikai kapuk segítségével feldolgozni
- Előre definiált feltételek teljesülése esetén riasztást generálni
- Manuális kapcsolókkal beavatkozni a rendszer működésébe
- Tetszőleges komplexitású logikai hálózatot felépíteni a komponensek összekapcsolásával

3 Bemenetek

3.1 Érzékelők

Az alábbi érzékelő típusok állnak rendelkezésre:

- **Diszk kapacitás érzékelő**: A szabad/foglalt tárterület arányát figyeli százalékos formában. Bemenetként megadható egy küszöbérték, amely fölött (foglaltság esetén) az érzékelő jelzést ad.
- **Memória kapacitás érzékelő**: A szabad/foglalt memória arányát figyeli százalékos formában. Bemenetként megadható egy küszöbérték, amely fölött az érzékelő jelzést ad.
- **Processzor terheltség érzékelő**: A processzor terheltségét figyeli százalékos formában. Bemenetként megadható egy küszöbérték, amely fölött az érzékelő jelzést ad.
- **Szerverszoba hőmérséklet érzékelő**: A szerverszoba hőmérsékletét figyeli Celsius fokban. Bemenetként megadható egy felső hőmérsékleti küszöb, amely fölött az érzékelő jelzést ad.
- **Tűzjelző**: Bináris értéket szolgáltat (tűz észlelve/nincs tűz). Nem paraméterezhető.

Az érzékelők konfigurációs adatai (ahol értelmezett) bemenetként tekintendők, amelyeket a program indulásakor vagy futás közben lehet megadni.

3.2 Kapcsolók

A kapcsolók manuálisan állítható komponensek, amelyek be- vagy kikapcsolt állapotban lehetnek. Ezek állapotát a felhasználó állíthatja be, amely bemenetként szolgál a rendszer számára.

4 Kimenetek

4.1 Vészcsengő

A vészcsengő a rendszer elsődleges kimenete, amely aktív vagy inaktív állapotban lehet. A vészcsengő aktiválódik, ha a hozzá kapcsolt logikai komponens(ek) kimenete ezt indukálja.

4.2 Rendszer állapot kijelzése

A rendszer az alábbi információkat jeleníti meg kimenetként:

- Minden érzékelő aktuális értéke és állapota (jelez/nem jelez)
- Minden logikai kapu állapota (aktív/inaktív)
- Minden kapcsoló állapota (be/ki)
- A vészcsengő állapota (aktív/inaktív)

5 Logikai komponensek

A rendszer tartalmaz logikai kapukat, amelyek segítségével az érzékelők és kapcsolók jelei feldolgozhatók:

- **ÉS kapu**: Minden bemenete aktív kell legyen az aktiváláshoz
- **VAGY kapu**: Legalább egy bemenete aktív kell legyen az aktiváláshoz
- **NEM kapu**: Megfordítja a bemenet értékét

A logikai kapuk kimenetei más kapuk bemeneteiként szolgálhatnak, így tetszőleges komplexitású logikai hálózat építhető fel.

6 A program működésének feltételei

- A program futásához standard C++ fordító és futtatókörnyezet szükséges.
- A megoldás nem használhat STL tárolókat, a szükséges adatszerkezeteket manuálisan kell implementálni.
- A rendszernek képesnek kell lennie tetszőleges számú komponenst kezelni a memória korlátain belül.
- A komponensek összekapcsolásának módját egyértelműen kell definiálni a felhasználói interfészen keresztül.

7 A rendszer tesztelése

A rendszer funkcionalitását külön modulként fordított tesztprogramnak kell demonstrálnia. A tesztprogram az alábbi funkciókat kell biztosítsa:

- Különböző érzékelők létrehozása és konfigurálása
- Logikai kapuk létrehozása és bemenetekkel való összekapcsolása
- Kapcsolók létrehozása és állapotuk változtatása
- Vészcsengő létrehozása és logikai hálózathoz kapcsolása
- A teljes rendszer állapotának lekérdezése és megjelenítése
- Szimulált bemenetek generálása a rendszer működésének demonstrálásához

8 Formátum és felhasználói felület

A tesztprogram parancssoros interfésszel fog rendelkezni, amely az alábbi funkciókat biztosítja:

- Komponensek létrehozása és konfigurálása
- Komponensek összekapcsolása
- Érzékelők értékeinek manuális vagy szimulált módosítása
- Rendszerállapot lekérdezése és megjelenítése
- Tesztszcenáriók futtatása