Szerverfarm felügyeleti rendszer
Feladatspecifikáció

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