Debreceni Egyetem

Informatikai Kar

Információ Technológia Tanszék

Otthon automatizáció Java-ban

Konzulens:
Vágner Anikó Szilvia
adjunktus

Szerző:
Lakatos István
programtervező informatikus

Debrecen, 2017.

Köszönetnyilvánítás

Tartalomjegyzék

1. fejezet

Bevezetés

1.1. Témaválasztás

Dolgozatom témaválasztása nem volt nehéz feladat számomra, mivel korábbi érdeklődés alapján már találkoztam az otthon „okosítás” vagy automatizálás témakörével. Rendkívül le tudtak foglalni a hardverközeli, mikrokontrollereket felhasználó projektjeim és azoknak kapcsán kezdtem el például különböző környezeti jellemző mérőeszközöket készíteni szabadidőmben. Onnantól kezdve pedig rövid út vezet házak utólagos felokosításának ötletéhez. Foglalkoztat még az is, hogy milyen előnyei lehetnek egy okos otthon rendszernek. Természetesen az elsődleges dolog e szempontból a spórolás lehetősége. Mindenki szeretne lefaragni a számlák összegéből például a fűtés kikapcsolásával, mikor nem vagyunk otthon, vagy a lámpák automatikus lekapcsolásával, ha éppen nem vagyunk a szobában. Másik vonzó tulajdonsága egy okos otthonnak, hogy rengeteg kényelmi funkciót tud nyújtani. Gomb nyomásra leengedhetjük a ház összes sötétítőjét vagy a rendszer meglocsolhatja helyettünk a növényeket. Tervem, hogy ezt az általam fejlesztett rendszert a saját otthonomban használjam jelentős számú mérőeszközöket felhasználva.

Annak a háttérében, hogy mindezt a Java nyelv segítségével tervezem megvalósítani, szintén egyszerű okok állnak. Tanulmányaim során legtöbbször a Java nyelvvel találkozhattam és így szerezhettem jelentősebb belelátást a működésébe és használatába. Így mivel szinte egyedül a Java nyelvhez kapcsolódóan van megfelelő tudásom ahhoz, hogy szakdolgozathoz illő nagyságú munkát készítsek, azt választottam alapnak. Másik fontos oka választásomnak, hogy szakmai gyakorlatom alatt is Java nyelvet használtam és így tudtam ismereteket szerezni több olyan keretrendszerről, amiket használni is szeretnék tapasztalatszerzés céljából, illetve pontosan beleilletek a bennem kialakuló

képbe, amit arról a rendszerről alkottam, mely ennek a dolgozatnak az alapját adja.

1.2. Célkitűzés

A dolgozatom során a fő cél egy olyan rendszer elkészítése, mely képes apró eszközök és egy központi egység segítségével automatizált feladatokat elvégezni. Ennek eléréséhez meg kellett tervezni az apró eszközök hardveres felépítését, fejleszteni egy központi szoftvert, ami képes az eszközöktől érkező adatok feldolgozására és azok alapján a felhasználó által megadott szabályok mentén feladatokat végrehajtására. Felmerült több érdekes kérdés is, melyekre kutatásaim során találtam válaszokat és próbáltam ezen válaszokat felhasználni a fejlesztés során.

A fejlesztés alatt végig úgy hoztam a döntéseket, hogy egy egyszerű háztartásban kell működjön a rendszer. Tehát nem volt célom az, hogy nagy számú felhasználók használják egyszerre vagy éppen több száz eszköz kapcsolódjon a központi rendszerhez. Részletesebben és feladathoz kapcsolódóan is meg fogom fogalmazni az elvárásokat, célokat a rendszerrel szemben, de ezt természetesen a későbbi fejezetekben teszem majd.

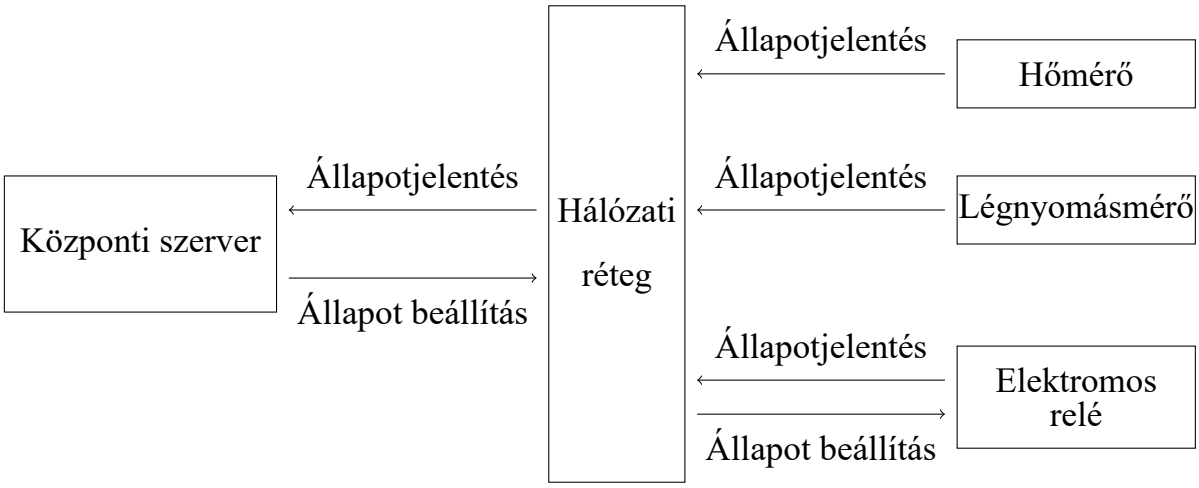
2. fejezet

Rendszer áttekintés

Ebben a fejezetben egy kezdeti képet szeretnék adni a teljes rendszer alapvető felépítéséről, illetve arról, hogy az egyes rendszer részek felé milyen elvárások vannak. Minden részt kifejtek ebben a fejezetben, de csak annyira, hogy átláthatóvá tegye a dolgozat többi részét. Későbbi fejezetekben viszont majd mélyrehatolóbban nézzük meg a rendszer alapjait.

Alapvetően három részre bontható a rendszer:

- Eszközök, amely magába foglalja az összes mérőeszközt és egyéb kisebb beágyazott rendszereket. Ezek egyszerű mikrokontrollereket használó rendszerek, amiknek annyi szerepe van, hogy vagy adatot szolgáltatnak, vagy valamilyen funkcionalitást végeznek el.
- Hálózati réteg, amely egy egyszerű közvetítőként funkcionál a központi szerver és az eszközök között. Lényegében átalakítja az üzeneteket a két rész között a megfelelő formátumra.
- Központi szerver, amely végzi az adatok feldolgozását és biztosít egy felületet a felhasználó számára, hogy láthassa a rendszer állapotát vagy aktuális adatokat.



2.1. ábra. A rendszer felépítése, a rétegek és eszközök egymás közötti kommunikációja

2.1. Eszközök

Mint már említve volt, ide tartoznak azok a hardverközeli rendszerek, amik vagy adatgyűjtő szerepet töltenek be, vagy valamilyen funkcionalitást képesek végrehajtani. Lehet egy ilyen eszköz például egy hőmérő, légnyomásmérő, földnedvességmérő vagy éppen egy relé.

Ezek az eszközök két csoportba sorolhatók:

- Érzékelők, másnéven *szenzorok* vagy
- Cselekvők, másnéven *aktorok*[1]

Valójában viszont ez a két csoport nem diszjunkt és később kiderül miért is nem, ha látjuk pontosan melyik mit jelent. Közös bennük, hogy mindegyikben van egy mikrokontroller és egy kommunikációért felelős modul. E mellett mindegyik eszköz tartalmaz még egy specifikus modult is, ami meghatározza, hogy mit is csinál az adott eszköz (pl.: hőmérő esetén egy hőmérséklet- és páratartalom érzékelő), illetve, hogy melyik csoportba tartozik.

2.1.1. Szenzorok

Magától értetődően a szenzorok felelősek az adatok szolgáltatásáért. Működését tekintve a mikrokontroller vezérli az adatgyűjtés és küldés folyamatát. Az idő nagy részében alvó állapotban van energiatakarékossági szempontok miatt, viszont fix időközönként felébred. Mikor felébred lekéri a saját specifikus moduljától a jelenlegi mért állapotot, majd azt megpróbálja közvetíteni a központba. Ezt addig próbálja, amíg nem sikerül, vagyis a központ vissza nem küld egy automatikus jelzést, hogy megérkezett az adat. Szenzorok közé sorolhatók például a sok féle mérőeszközök vagy akár egy mozgásérzékelő is.

2.1.2. Aktorok

Az aktorok a rendszer olyan részei, amik segítségével a rendszer képes változtatásokat végbe vinni a fizikai világban. Hasonló a helyzet a szenzorok esetéhez, hiszen ugyanúgy a mikrokontroller kezel mindent. Ugyanúgy alvó állapotban van az eszköz az idő nagy részében, viszont jelen esetben nem fix időközönként ébred fel, hanem amikor

[1] Ezt az elnevezést magam adtam, mivel nem találkoztam kutatásaim során más megfelelő elnevezéssel.

üzenetet kap a központtól. Ezek az üzenetek állapot beállítási parancsok, vagyis utasítják az eszközt, hogy állítson át vagy cselekedjen valamit. Ha megtörtént a parancs feldolgozása, az eszköz egyből elküldi a központi rendszernek az új állapotát, állapotjelentés formájában, akárcsak a szenzorok, majd megint alvó állapotba kerül a következő parancsig. Aktor lehet például egy elektromos relé vagy egy termosztát is.

Tehát láthatjuk, hogy miért is nem szétválasztható a két csoport. Az aktorokra tekinthetünk olyan szenzorokként, amik képesek még egyéb funkcionalitást is végezni. Így tekintve a szenzorok csoportja tágabb és magába foglalja az aktorok csoportját.

2.2. Hálózati réteg

A hálózati réteg léte elsőnek nem látszik logikusnak. Feleslegesnek tűnhet egy plusz szereplőt bevonni a központi szerver és az eszközök közé. A probléma, ami mégis megköveteli azt, hogy legyen egy köztes réteg, az egy eszközökhöz kötődő döntésből fakad.

A hardveres tervezés alatt találkoztam több vezetéknélküli kommunikációs modullal is. Volt olyan, ami WiFi-t használ a kommunikációhoz, volt ami csak simán rádiófrekvenciás modul volt. Bizonyos okok miatt, amit majd a 3.1.2. alfejezetben fogok leírni, a rádiófrekvenciás modulok mellett döntöttem. Így viszont szükség van egy központi eszközre, ami egy ugyanolyan kommunikációs modullal kell rendelkezzen, mint a többi eszköz. Ahhoz, hogy ezt a modult használni tudjuk, viszont alacsonyabb szintű eszközökhöz kell folyamodni, mint amit a Java nyújtani tud. Pont ezért a hálózati réteg egésze C++-ban íródott.

Lényegében a hálózati réteg tartja fent a kapcsolatot az eszközökkel és alakítja át az üzeneteket a fogadó félnek megfelelő formátumra. Ez az alakítás azért szükséges, mert a kommunikáció a hálózati réteg és a központi szerver között egy TCP kapcsolaton keresztül történik JSON objektumok küldésével, viszont a hálózati réteg és az eszközök között C-beli `struct`-okat küldünk rádiófrekvenciás jelek segítségével.

2.3. Központi szerver

A legnagyobb funkcionalitást természetesen a központi szerver végzi, hiszen ott lesznek feldolgozva az eszközöktől érkező állapotjelentések és a szerver küldhet állapot beállítási parancsokat is.

Mint említve volt már a központi szerver TCP kapcsolaton keresztül kommunikál a hálózati réteggel. Az onnan érkező üzeneteket a rendszer például a megfelelő formában elmenti az adatbázisban vagy az üzenet hatására elindulhat

A központi szervert is tovább tudjuk bontani több darabra funkcionalitása alapján:

- Flow-rendszer
- Statisztika
- Kezelőpanel

2.3.1. Flow-rendszer

A *Flow-rendszer* onnan kapta a nevét, hogy a felhasználó által megadható szabályokat „flow”-knak neveztem el. A név a beérkező adatok folyamából és azoknak folyamatos feldolgozásából jön. Minden *flow*-nak van feltétele és hatása. Ez a hatás akkor fog bekövetkezni, ha a feltétel teljesül. Részletesebben a 5.3. alfejezetben lesz szó a *flow*-król.

A *Flow-rendszer* biztosítja az új szabályok létrehozásának, régebbiek módosításának lehetőségét és elmenti az így létrejött változásokat. Másik lényeges feladata ennek a rendszernek még, hogy be is tartassa ezeket a szabályokat. A szabályok betartásáért felelős mechanizmust szintén a 5.3. alfejezetben fogom taglalni. Röviden összefoglalva, ha új adat érkezik, a rendszer megnézi melyik szabályokat érintheti az esemény és kiértékeli azokat. Tulajdonképpen ez a rendszer felelős a szerver legfontosabb és legbonyolultabb funkcióiért.

2.3.2. Statisztika

A rendszer használata során természetesen szeretnék néha visszanézni régebbi méréseket. Lehet ennek sok oka, például szeretnénk megnézni mennyivel volt hidegebb az előző hónapban, vagy leellenőrizhetjük mennyit volt bekapcsolva a fűtés. Tehát elég könnyen adja magát az elvárás, hogy tudjunk statisztikákat nézni a múltbeli adatokról.

Ehhez viszont el kell tárolni minden adatot, hogy később is tudjunk valamit mutatni a felhasználónak. Az adatbázis-rendszer kiválasztásánál ez volt az egyik legnagyobb szempont. A választás a MongoDB-re jutott, viszont a döntés mögött álló érveket majd a 5.1. alfejezetben fogom ismertetni. Mindemellett ahhoz, hogy az a sok adat, amit elmentenénk ne foglaljon sok tárhelyet, nem külön mentődnek el, ahogy beérkeznek a szerverhez, hanem percenként átlagolva kerül az adatbázisba. Ezzel igaz, hogy egy

kicsi pontosságot vesztünk, mikor visszanézzük a statisztikákat, viszont ha jobban belegondolunk nincs is szükségünk arra, hogy például 10 másodpercenkénti részekre lebontva lássuk a múlt heti hőmérsékletet. Ami azt illeti még a percenkénti lebontás is túl aprónak tűnik bizonyos helyzetekben, de természetesen a lementett adatok segítségével elő tudunk állítani akár napi vagy havi átlagokat is. Mindezt grafikonokon és idő-soros diagrammokon mutatva, a felhasználó könnyedén tud következtetések levonni magának.

2.3.3. Kezelőpanel

A kezelőpanel szolgáltatja a felhasználó számára a valós idejű adatokat. Itt jelennek meg a beérkező állapotjelentések az eszközöktől legelőször, illetve aktorok esetében itt lesz lehetőségünk kézzel átállítani az eszköz állapotát, vagyis állapot beállító parancsokat küldeni.

Lehetővé válik így, hogy a felhasználó akár a munkahelyéről is megnézhesse hogy áll az otthona, esetleg nem hagyta-e nyitva az ajtót reggel. Ez a pár másopercenként frissülő kezelő felület minden eszköztípushoz külön mini megjelenítő modulokat rendel, aminek köszönhetően akár pár pillanat alatt megtalálhatjuk a számunkra fontos információkat.

3. fejezet

Eszközök

Az előző fejezetben megtudtuk, mit csinálnak az eszközeink. Szintén megismertük, hogy mit is jelent a *szenzor* és az *aktor* fogalma, illetve, hogy miben különböznek. Ebben a fejezetben ecsetlve lesz, milyen hardverek lettek kiválasztva az eszközök összeállítására, illetve meg lesznek indokolva ezen választások is.

Ugye tudjuk szintén az előző fejezetből, hogy minden eszköz „agya" egy mikrokontroller. Ez a kicsi és alacsony teljesítményű processzor felelős az adatgyűjtés folyamatának koordinálásáért. Mivel nincs kiterjedt tudásom a mikrokontrollerek programozásának terén, számomra a legkönnyebb volt az **Arduino** fejlesztői környezetek mellett letenni a voksom.

Másik fontos alkotóelem, a vezetéknélküli modul. Ebben a választásban már természetesen megszorítás volt, hogy a kiválasztott modul kompatibilis legyen Arduinokal. Habár nem volt egyszerű döntés, egy **nRF24L01+** adó-vevő integrált áramkört felhasználó modult választottam.

Ahogy korábban említettem harmadik része az eszközeinknek változó, még pedig az határozza meg, hogy mit szeretnénk elérni az adott szenzorral vagy aktorral. A fejlesztés elején kiválasztottam pár eszköztípust, amit el szerettem volna készíteni és támogatni a kész rendszerben. A teljesség igénye nélkül be fogok mutatni pár szenzor és aktor modult, mint például a hőmérséklet mérésére szolgáló **DHT22** hő- és légnedvesség mérő szenzor modult.

Mindezek után kitérek arra, hogy a fent említett hardvereket hogyan kell összekötni, hogy egy működő eszközt kaphassunk. Szerencsémre az összerakási folyamat nem igényelt jelentős villamosmérnöki tudást, elég volt néhány kábellel összekötni a megfelelő ki- és bemeneteket.

3.1. Hardver választás

3.1.1. Arduino

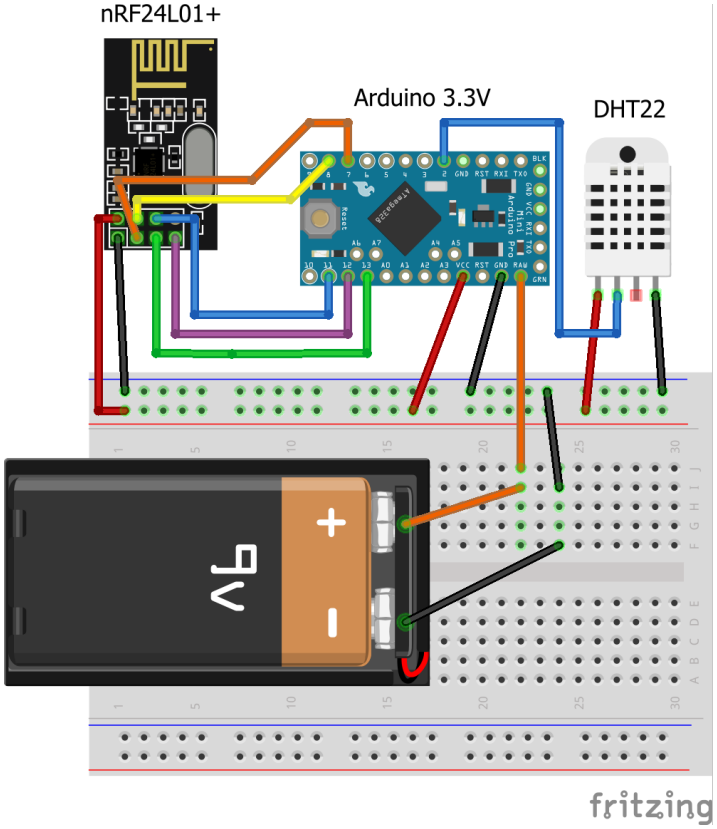
3.1.2. Vezetéknélküli modulok

3.1.3. Eszköz specifikus hardver

3.2. Felépítés

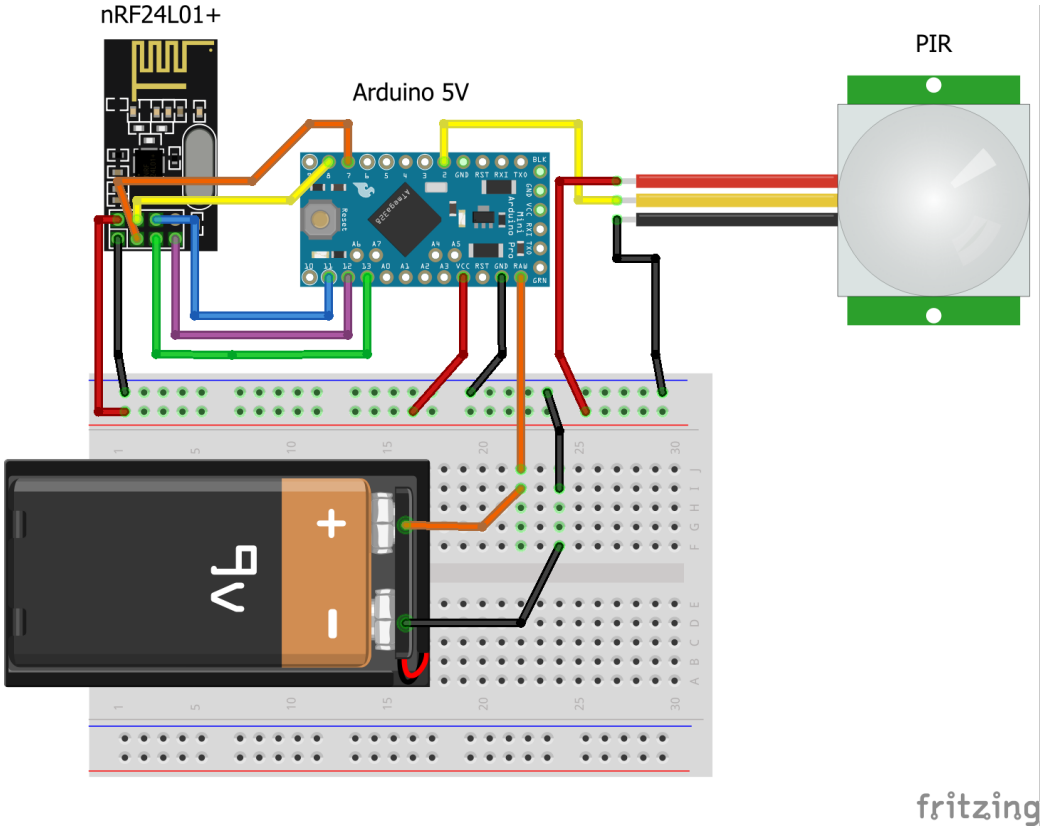
3.2.1. Példák kész eszközökre

- Hő- és páratartalommérő szenzor



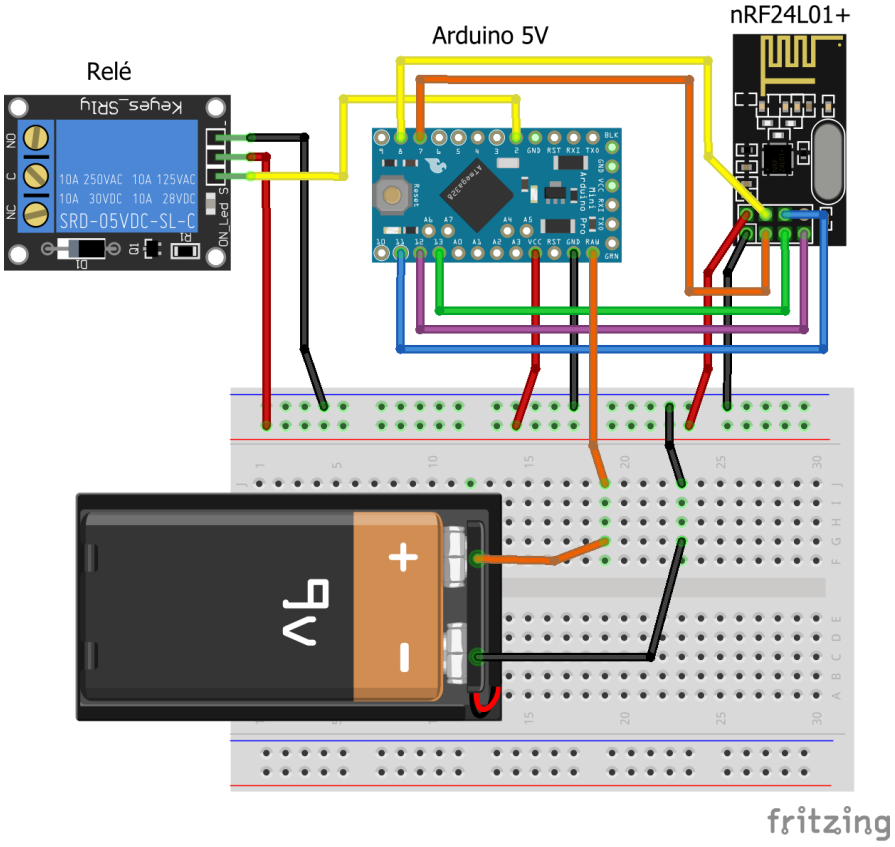
3.1. ábra. Hő- és páratartalommérő szenzor

- Mozgásérzékelő szenzor



3.2. ábra. Mozgásérzékelő szenzor

- Elektromos relé aktor



3.3. ábra. Elektromos relé aktor

3.3. Szoftver működés

3.3.1. Alvó mód

4. fejezet

Hálózati réteg

4.1. Mesh hálózat

4.2. Üzenet küldés

4.3. Szoftver működés

4.3.1. Boost

5. fejezet

Központi rendszer

5.1. Adatbázis

5.1.1. Adatbázis kiválasztása

5.1.2. Tárolási séma

5.2. Keretrendszerek

5.2.1. Spring Boot

5.2.2. Vaadin

5.3. Flow rendszer

Ezek az úgynevezett „flow”-k olyanok akárcsak egy-egy szabály. Olyan szabályok melyeknek van egy feltétele és egy hatása. Egy flow akkor lép életbe, ha a hozzátartozó feltétel a rendszer éppen aktuális állapota mellett teljesül. Ekkor a flow hatása végrehajtódik a rendszer által.

A feltétel része lehet egyes eszközök állapotára vonatkozó megszorítások (pl.: 20°C-nál nagyobb a hőmérsékletet mutat a nappaliban elhelyezett hőmérő), a felhasználó vagy külső rendszer által indított kérés az alkalmazáshoz (pl.: gomb nyomás a kezelőfelületen vagy HTTP kérés egy bizonyos címen), egyéb rendszer állapot feltétel (pl.: időponthoz kötődő feltétel) és ezeknek logikai ÉS-sel összekötött kombinációja. A

hatása egy flow-nak állhat eszközök állapotának módosításából, más rendszerhez történő kérésből és egyéb segéd akciókból (pl.: késleltetés). Ezeknek a „hatás elemeknek” egymás után történő végrehajtása adja az adott flow hatását.

A rendszer célja az, hogy a fenti flow-k segítségével a felhasználó szabályokat/feladatokat tud leírni, amelyeket a rendszer majd végrehajt. Így tehát lehetséges bizonyos házkörüli dolgok automatizálása.

Pár példa a rendszer használatára:

- ha adott szobában nincs érzékelt mozgás, akkor lekapcsolódik a villany
- ha több hőmérő is alacsony értéket mutat, akkor automatikusan fentebb megy a fűtés
- ha egy virág földje kiszáradna, akkor víz engedődik a virág alatt
- ha reggeli időpont van, akkor beindul a kávéfőző
- ha a levegő szén-monoxid tartalma átlép egy határt, akkor elindul egy jelző berendezés
- ha besötétedik és van mozgás, akkor a sötétítők leengednek és felkapcsol egy villany

5.3.1. Flow kiértékelés

6. fejezet

Teljesítmény

6.1. Hatótáv

6.2. Eszköz szám

6.3. Áteresztőképesség

7. fejezet

Biztonság

7.1. Megbízott eszközök

7.2. Titkosítás

7.3. Problémák

8. fejezet

Összefoglalás

Irodalomjegyzék