Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Okosotthon rendszerek konfigurációja

Jankó András I. évf, MSc mérnökinformatikus szakos hallgató
Konzulens: dr. Ráth István, MIT
Kritikus rendszerek specializáció
Önálló laboratórium 1. összefoglaló
2018/19. I. félév

Okosotthon alatt mindenki mást ért. Az én értelmezésemben egy okosotthon nem attól lesz igazán okos, hogy vezérelni tudjuk benne az eszközeinket (pl. lámpák, fűtés), hanem képes észlelni, hogy használni szeretnénk őket, vagy szükségünk lesz rájuk, és ennek megfelelően cselekedni. Azonban, hogy magától képes legyen reagálni a környezeti változtatásokra, szükség van egy mögöttes automatizációs logikára. A probléma, hogy ezt a logikát el is kell készíteni, és sajnos minden piacon lévő okosotthon rendszerben más. A félév során ebben a témakörben kutattam, hogy milyen megoldást lehetne találni erre a problémára.

A félév során több okosotthon rendszer működését kipróbáltam, hogy nagyobb rálátásom legyen arra, hogy mi okoz problémákat, melyek azok a részek amik ismétlődnek, és esetleg kiválthatók lennének valami mással. Arra jutottam, hogy három fő része van egy rendszer konfigurációnak, ahol redundáns elemek vannak.

Az első ilyen az eszközök felvétele. Itt mindig meg kell adni az eszköz kijelzett nevét, egy egyedi azonosítót, amivel a rendszerben hivatkozhatunk rá, kommunikációs paramétereket, pl MQTT-n melyik csatornán érhető el, és az ott érkező adatokat hogyan értelmezze/konvertálja, és opcionálisan csoportokba rendezni. Ez két okból is szükséges, egyrészt a megjelenítő felületek akár tudnak egy automatikus megjelenítést készíteni, például ha szobákba rendezzük az eszközeinket.

A másik nagyon hasonló érsz az automatizációs szabályok. Sajnos itt minden egyes rendszerhez ismernünk kell a megfelelő szintaxist ahhoz, hogy el tudjunk kezdeni dolgozni vele. Általában egy okosotthon szabálynak három része van. Először megadjuk a trigger feltételeket, vagy másnéven event-eket, amik elindítják a szabály kiértékelését. Ebből többfajta is lehet, és logikai vagy kapcsolatban állnak egymással. Második részben a szűrő feltételeket adjuk meg, és a szabály csak akkor lép át a következő részbe, ha minden feltétel igaz volt, tehát logikai és van közöttük. Ezek után az akció részben tudjuk megadni, hogyan reagáljunk, például eszközöknek küldhetünk parancsot. Ami itt kifejezett problémát okoz, hogy nincs a szabály szerkesztéshez szintaxis ellenőrzés szerkesztés időben, csak futás időben.

A harmadik hasonló rész a felhasználói felületek. Általában itt megadjuk, hogy az eszközeink milyen hierarchiában legyenek a megjelenítésnél, és az egyes típusú eszközökhöz, milyen megjelenítési típus tartozzon, vagy a szenzorok eredményeit mekkora pontossággal jelenítse meg (pl. hőmérő).

Arra a következtetésre jutottam, hogy a három részhez a mögöttes logikát, meg lehetne fogalmazni eggyel magasabb szinten is, nem csak az okosotthon rendszerek specifikus nyelvén. Ebből a magasabb szinten megfogalmazott logikából egyértelműen le lehetne generálni az adott okosotthon nyelvén is a rendszer konfigurációját. Ezzel kiküszöbölhetnénk a szintaxis hibákat, és még validációt is vezethetünk a rendszerbe, hogy például csak olyan eszközökre lehessen hivatkozni egy automatizációs szabályban, amelyeket léteznek, tehát definiáltuk a konfigurációs részben.