

TDK-dolgozat

**Sándor Hunor
Szabó Tamás**

Androidos okosház felügyelő és vezérlő

Szerzők:

Sándor Hunor

Petru Maior Egyetem, Marosvásárhely, Mérnöki Kar, Villamosmérnöki és Számítástechnika Tanszék, Automatizálás és Alkalmazott Informatika szak, IV. év

Szabó Tamás

Petru Maior Egyetem, Marosvásárhely, Mérnöki Kar, Villamosmérnöki és Számítástechnika Tanszék, Automatizálás és Alkalmazott Informatika szak, IV. Év

Témavezető:

dr. Haller Piroska egyetemi docens,

Petru Maior Egyetem, Marosvásárhely, Mérnöki Kar, Villamosmérnöki és Számítástechnika Tanszék

Tartalomjegyzék

1.	Bevezető	4
2.	Általános felépítés és működés	5
3.	Funkcionalitások	7
3.1	Konfigurációs szerver	7
3.2	ESS szerver	8
3.3	Multimédia rendszer	9
3.4	Végrehajtó eszközök.....	10
4.	Android kliens alkalmazás	10
4.1	Grafikus felület.....	10
4.2	Sajátos funkciók	12
4.3	Több szál szerepe	13
4.4	Öndiagnosztizáló funkció.....	15
4.5	Különös megoldások a megbízhatóság növelésére.....	15
4.6	Vezeték nélküli kapcsolatok.....	16
4.7	Teljesítmény paraméterek.....	17
5.	Következtetések.....	18
6.	Egyéni hozzájárulás	19
7.	Jövőbeli tervek.....	20
8.	Könyvészet.....	21
9.	Függelékek.....	22

1. Bevezető

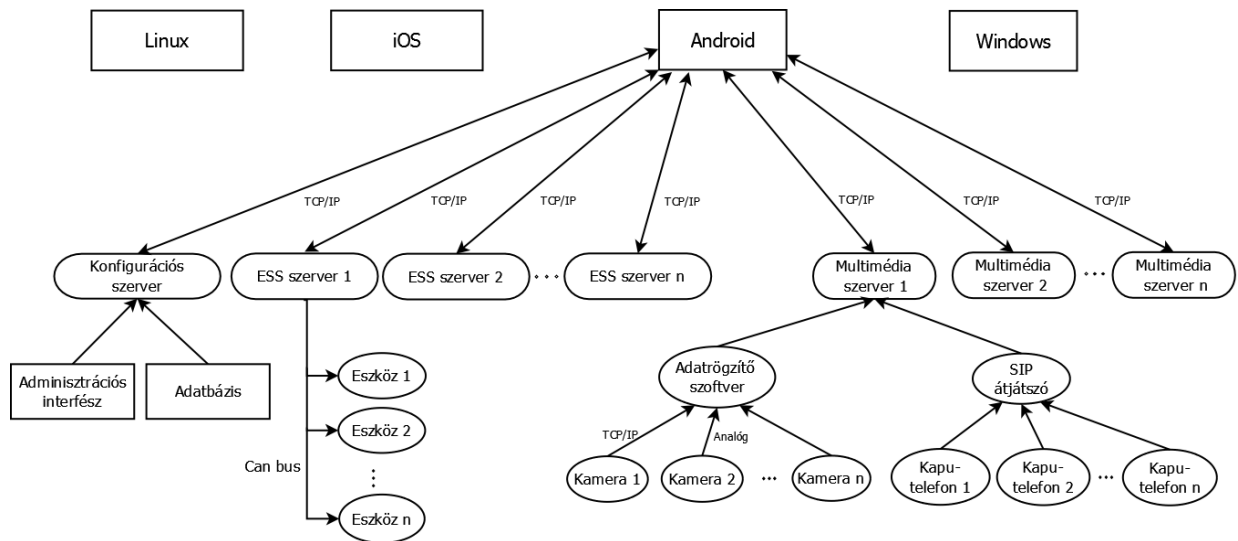
Napjainkban az emberiség egyre nagyobb hangsúlyt fektet önmaga kényelmének megteremtésére, ennek következtében újszerű igények alakulnak ki. Továbbá az ember szereti szemmel tartani és biztonságban tudni tulajdonait. Ezeknek az igényeknek a kielégítését tűzi ki célul ezen projekt egy olyan Androidos mobil alkalmazás fejlesztésével, mely lehetővé teszi fizikailag osztott épületegyüttesek (lakóházak, intézmények, gyárak, stb.) központosított távfelügyeletét és vezérlését egy egyszerű és intuitív grafikus felületen keresztül.

Az alkalmazásnak biztosítania kell az épületekben felszerelt különböző fizikai eszközök (kapcsolók, érzékelők, kamerák, fűtőtestek, stb.) állapotainak valós idejű megjelenítését és módosítását. Mindez érdekében az alkalmazás egy több szerverből álló rendszerrel áll állandó kapcsolatban, melyek minden szükséges információt biztosítanak és továbbítják a beavatkozó parancsokat az automatizált eszközök felé. A rendszer teljes mértékben integrálja a különböző csatolt eszközöket és egységes elérést biztosít a felhasználónak.

A teljesítménykövetelményeknek való megfelelés érdekében az alkalmazás több szálon fut, ezáltal kamatoztatva az osztott architektúra előnyeit: párhuzamos adatfeldolgozás, valós idejű információmegjelenítés és állandó öndiagnosztika.

Eredményképpen egy olyan stabil és felhasználóbarát alkalmazás jött létre, amely a felhasználó számára teljes mértékben testreszabható és amely bármely piacon kapható Androidos telefonon vagy tableten futtatható.

2. Általános felépítés és működés



1. Ábra Általános felépítés

Az okosház felügyelő és vezérlő rendszer egy összetett osztott rendszer, amelyben központi helyet foglal el az Android alkalmazás, mivel ez valósítja meg a különböző komponensek központosítását.

A rendszer összessége épületek, épületrendszerek vagy akár fizikailag osztott létesítmények állandó megfigyelését, automatizálását biztosítja. Lehetővé teszi a riasztó rendszer, térfigyelő rendszer, világító rendszer, fűtő és klimatizáló rendszer, nyílászárók, érzékelők és más kényelmi eszközök távmegfigyelését és távvezérlését.

Továbbá a felhasználó intelligens funkciókat állíthat be, ezáltal a rendszer automatikus módban is működhet, külső beavatkozás nélkül. Ezek a funkciók lehetnek szkriptek, amelyek bizonyos körülményeket figyelnek a rendszerben, és azok teljesülése esetén előre meghatározott eseménysorozatokat váltanak ki. További automata funkció a vészjelzések kezelése, amely hasonlóan a szkriptekhez, meghatározott paramétereket figyel a rendszerben, és a feltételek teljesülése esetén figyelmeztető jelzést küld a felhasználó felé.

Egy egyszerű példa szkriptek használatára a napszak és évszak függvényében való hőmérsékleti referencia változtatása a fűtőrendszerben, viszont ennél merőben bonyolultabb feltételek és kiváltott események is beállíthatóak. Jó példa vészjelzésekre a fizikai rendszert

tápláló elektromos ellátás figyelése, és riasztás abban az esetben, ha megszűnik az akkumulátorok töltése vagy azok feszültsége bizonyos határ alá esik.

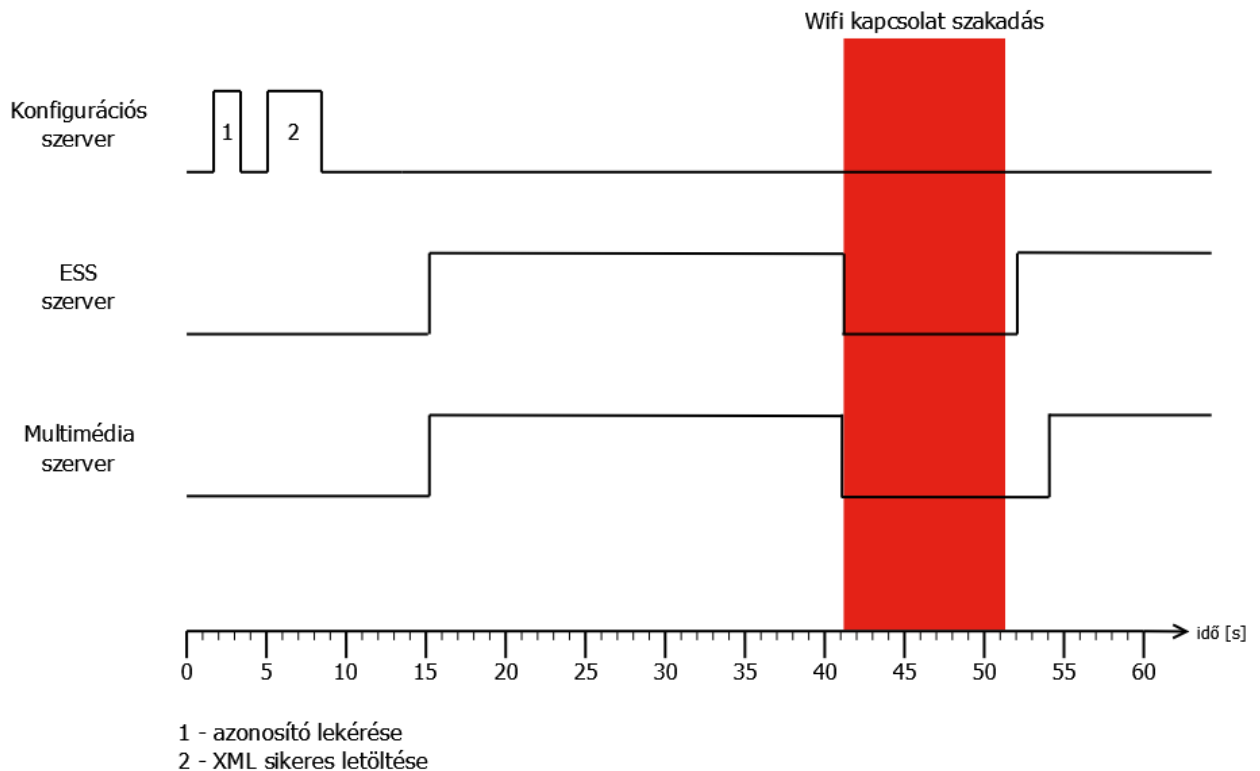
Fontos követelmény egy ilyen jellegű rendszer esetén a jogok, hozzáférések kialakítása. Ez a felhasználó azonosításán alapszik, amely függvényében a felhasználó csak a számára elérhető rendszerkomponensekhez férhet hozzá. Így egy rendszerhez akár több kliensalkalmazás is csatlakozhat egyidőben, amelyek között akár audio kommunikáció is létesíthető.

A rendszert alkotó részek egyik fontos kategóriáját a fizikai eszközök jelentik, amelyek az automatizált célterület helyszíni megfigyelését és vezérlését végzik. A másik fő rész a kliens alkalmazások összessége, amelyek segítségével egy központosított megjelenítő- és kezelőfelülethez jutnak a felhasználók. Ezek a kliens alkalmazások funkcionalitásban hasonlóak, viszont a futtató operációs rendszer által nyújtott lehetőségekhez alakítottak. A Windows és Linux operációs rendszereken futó kliens alkalmazások előnye a rendelkezésre álló nagy megjelenítő felület, amelyet felhasználva látványos grafikus felülettel látható el az alkalmazás, viszont a futtató számítógép helyhez kötött, így nem biztosít állandó kényelmes hozzáférést. Ezekkel szemben az Android operációs rendszerrel ellátott mobiltelefonok és tabletek előnye a mobilitás és ezáltal az állandó rendelkezésre állás.

A megjelenítő és fizikai modulok között egy szerverrendszer létesít kapcsolatot, amely 3 típusú szerverből és 2 típusú kiegészítő szoftverből áll, amelyekből típusonként akár több fizikailag különálló egység is létezhet egy rendszerben. A szerver típusok a következők: *Konfigurációs (XML) szerver*, *ESS (Environmental Settings Software) szerver*, illetve *Multimédia szerver*, míg a kiegészítő szoftverek *Adatrögzítő alkalmazás* és *SIP átjátszó* kategóriákba sorolhatóak. Ezek a komponensek TCP/IP kapcsolatok segítségével kommunikálnak egymással és a kliens alkalmazásokkal.

A rendszer által nyújtott lehetőségek végtelen sokasága állandó kényelmet és biztonságot nyújt a felhasználók számára otthonukban, munkahelyükön, vállalkozásukban, hétévesi házukban, vagy akár ezek bármilyen kombinációjában.

3. Funkcionalitások



2. Ábra Szerver kapcsolatok időzítése

3.1 Konfigurációs szerver

A Konfigurációs szerver a felhasználónév, jelszó és a kliens alkalmazást futtató operációs rendszer függvényében, a csatolt adatbázisban tárolt adatok alapján, szöveges módban, XML leíró nyelvet használva továbbítja a kliens alkalmazásnak a felhasználó által elérhető fizikai eszközök listáját, ezek tulajdonságait, funkcióit, egymáshoz való viszonyukat és a kívánt csoportosításukat a grafikus felületen. Ezek határozzák meg a grafikus felületről elérhető beavatkozási lehetőségeket és az alkalmazás aktuális kinézetét. Az alkalmazás az XML leírás alapján hozza létre a megfelelő Multimédia szerver kapcsolatokat is.

A kívánt konfiguráció egy *adminisztrációs interfészen* keresztül készíthető el, szabható személyre. Ezek segítségével többféle logikai csoportosítás is megvalósítható: épületek, szobák vagy funkcionalitás szerint. Ezenkívül hasznos a gyakran használt eszközök csoportosítása a gyors elérés céljából. A konfiguráló interfész az eszközök átrendezését, új

eszközök hozzáadását, meglévő eszközök paramétereinek módosítását és vészjelzők, szkriptek létrehozását is lehetővé teszi. Továbbá lehetőség nyílik a konfiguráció módosítására direkt módon is egy adatbázis adminisztráló szoftver segítségével.

A szerverrel való kommunikáció első lépése egy egyedi azonosító lekérdezése, amely nélkülözhetetlen a további lekérdezések végrehajtásához. Ezt követi a felhasználó és platform specifikus XML leíró szöveg lekérdezése. A Konfigurációs szerverrel való kapcsolat ideiglenes jellegű, így az adatok vételezése után fel is bomlik.

Egy eszköz leírása XML nyelvben:

```
<device>
  <name>Camera4</name>
  <index>0</index>
  <guiname>Receptie</guiname>
  <globalparameters>
    <conninfo host="188.26.16.27" lport="10065"
      rport="10060"/>
    <conninfo host="192.168.3.1" lport="10065"
      rport="10060"/>
  </globalparameters>
  <id>56</id>
  <type>CAMERA</type>
  <xcoord>1</xcoord>
  <ycoord>1</ycoord>
  <parameters>
    <display_iconified value="true"/>
    <master_dev name="Entrance"/>
    <dev_hidden value="false"/>
  </parameters>
  <demon>IBS</demon>
  <monitor>Can</monitor>
  <resx></resx>
  <resy></resy>
  <available_functions>
    <function>View</function>
  </available_functions>
</device>
```

3.2 ESS szerver

Hardver szinten a fizikai eszközök szokványos CAN-Bus kommunikációval csatlakoznak az ESS szerverhez, de a szerver emellett egységes elérést biztosít a más ipari protokollokon kommunikáló eszközökhöz is.

Az ESS szerver feladata az automatizált fizikai eszközök folytonos monitorizálása, állapotainak folyamatos, esemény alapú frissítése, valamint a kliens beavatkozó parancsainak továbbítása a fizikai eszközökhöz.

Minden eszköz egy külön funkcióhalmazzal rendelkezik, ezek határozzák meg viselkedésüket és a grafikus felületről elérhető beavatkozási lehetőségeket a kliens oldalon. Az ESS kapcsolat esetében ezek a funkciók egy vagy több állapot változóra vonatkoznak.

Az eszközökkel való kommunikáció nem az egyetlen feladata a szervernek, emellett személyre szabott szkriptek futtatására is alkalmas. Ily módon egyetlen gombnyomással komplex feladatok elvégzése válik lehetővé, fokozván a felhasználó kényelmét és az alkalmazás funkcionalitását. Az ESS szerver lehetőséget nyújt ugyanakkor a különböző változások, események figyelésére és elmentésére.

Egy ESS állapotfrissítés a következőképpen néz ki:

(GET) (DEMON) (MONITOR) (PEER) (VARIABLE) (VALUE)

Ebből azonosítható a szóban forgó eszköz a *DEMON*, *MONITOR* és *PEER* mezők alapján, míg a *VARIABLE* és *VALUE* mezők a módosult állapotváltozót és ennek új értékét tartalmazzák. A változó értékének beállítása hasonló azzal a különbséggel, hogy a *GET* utasítást a *SET* helyettesíti.

3.3 Multimédia rendszer

A Multimédia szerver az audio és videó adatok valós idejű begyűjtéséért, elmentéséért, továbbításáért, visszajátszásáért és az adatfolyamok kezelését végző utasítások továbbításáért és feldolgozásáért felelős. Az adatfolyamok irányítása saját protokollon keresztül történik.

A videó adatokat egy *Adatrögzítő alkalmazás* továbbítja a Multimédia szerver felé TCP/IP kapcsolat segítségével. Az *Adatrögzítő alkalmazás* a különböző típusú kamerák által vételezett képeket egységes formátumba alakítja és így továbbítja a Multimédia szerver felé, ezáltal akár analóg kamera, WEB kamera, IP kamera vagy egy számítógép videó kártyája is csatlakoztatható a rendszerhez.

Az audio adatok két forrásból juthatnak el a Multimédia szerverhez. Két kliens alkalmazás közti audio kommunikáció esetén a hanganyagot az említett alkalmazások

továbbítják a szerverhez, míg kaputelefonos hívás esetén a hanganyag egy SIP átjátszó alkalmazástól jut el a Multimédia szerverhez. A *SIP átjátszó alkalmazás* feladata továbbítani a kaputelefonok által vételezett hanganyagot a Multimédia szerver felé, továbbá feladata a SIP hívás-fogadás üzeneteinek átjátszása a Multimédia szerver saját protokolljára. A kommunikáció ebben az esetben is TCP/IP csatornán keresztül valósul meg.

Az audió adat, videó adat és a vezérlő utasítások egyazon TCP/IP csatornán közlekednek a kliens alkalmazás és a Multimédia szerver között.

A valós idejű adatátvitel biztosítására a szerver rendelkezik sávszélesség szabályzó funkcióval, amelynek köszönhetően egy időben lehetséges valós idejű audió és videó adatfolyam átvitel is. A szerver a nagy méretű adatcsomagokat(videó képkocka) feldarabolhatja, hogy közbeiktatva nagy prioritású adatcsomagokat(hanganyag) továbbítson, ezáltal biztosítva a jó minőségű valós idejű hangátvitelt.

3.4 Végrehajtó eszközök

A rendszer feladata a megfelelő automatizált fizikai eszközökkel való kapcsolatlétesítés és azok vezérlése. Ezek az eszközök egyedi tervezésű univerzális felhasználásra alkalmas vezérlők, amelyekhez csatlakozhatnak az automatizálni kívánt eszközök.

Ilyen eszközök az univerzális kapcsolók, amelyekhez világítótestek, zárok illeszthetők, dimmerek, amelyek a világítótestek fényerejét szabályozhatják, több bemenetű eseményérzékelők, amelyekhez mozgásérzékelők, füstérzékelők csatlakozhatnak, analóg bemenetek, amelyekhez fényerő mérőt lehet csatlakoztatni, fűtésrendszer vezérlők, amelyek hőmérsékletet mérnek és a beállított referencia függvényében vezérlik a hőközpontot és ki-be kapcsolják a fűtőttesteket, redőnyvezérlők, kameravezérlők, amelyek lehetővé teszik a kamerák elfordítását és sok más a felhasználó kényelmét és biztonságát garantáló eszköz.

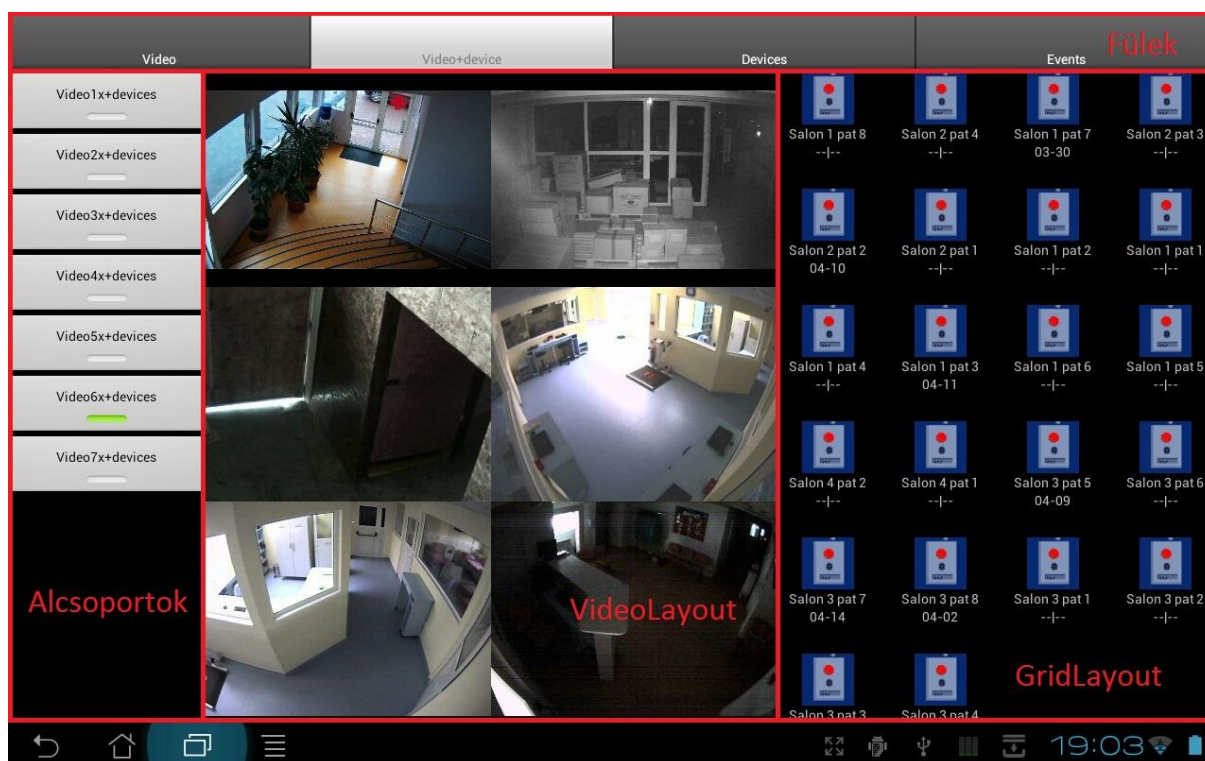
4. Android kliens alkalmazás

4.1 Grafikus felület

Az alkalmazás tervezésében elsődleges szempont egy alacsony teljesítményigényű és felhasználóbarát grafikus felület létrehozása. Annak érdekében, hogy az alkalmazás bármilyen Android operációs rendszeren és bármilyen eszközön optimálisan használja ki a

rendelkezésére álló erőforrásokat és a hasznos képernyőméretet, a generált felület méretezése teljes mértékben dinamikus. A dinamikus méretek kiszámolását befolyásolja az eszköz orientációja, a különböző komponensek száma, elhelyezése és mérete, valamint egyéb felhasználó beállítások.

A bejelentkezési ablakban lehet kiválasztani egy elmentett felhasználónevet és titkosított jelszavat, valamint a kívánt épületet. Ugyanitt lehetőség van az alkalmazás külső konfigurációs fájlt módosítani egy külön ablakban megjelenő segédszerkesztővel.



3. Ábra Grafikus felület felosztása

A felhasználói felület egy több füles megjelenítésen alapszik, amely a fő eszköz csoportok közötti navigációt teszi lehetővé. Minden fülön belül a tartalom 3 részre bontható: bal oldalt található egy lista, amely a fülön belül alcsoportokra osztja az eszközöket. A következő 2 rész tartalmazza a kiválasztott fő- és alcsoportban levő eszközöket. A második, ún. *VideoLayout* rész tartalmazza azokat a kamerákat, amelyek az XML konfiguráció szerint azonnali videóközvetítés kell nyújtsanak. A harmadik, ún. *GridLayout* rész táblázatos megjelenítésű és a hátralévő eszközöket foglalja magába.

A *VideoLayout* részben a kamerák elrendezése dinamikusan változik a rendelkezésre álló képernyő terület, az Android eszköz vízszintes vagy függőleges orientációja és a

megjelenítendő kamerák számának függvényében. A legmegfelelőbb elrendezés kiválasztása kritikus, hiszen az alkalmazás úgy a 9-10 inches képernyőkön, mint a kis képernyős okostelefonokon is átlátható és könnyen használható kell legyen.

A *GridLayout* rész az Android eszköz orientációja függvényében foglal helyet az előző két rész alatt vagy mellett. A táblázatban az eszközöket a típusuk és a jelenlegi állapotuknak megfelelő ikon jelképezi. A felhasználónak lehetősége van az alapértelmezett ikonokon kívül saját ikonokat is meghatározni. Az ikonok alatt ugyanakkor az eszköz nevén kívül megjelenik a legfontosabb állapot információ is, a minimális helyfoglalás érdekében rövidített formátumban.

A részletesebb nézet érdekében egy eszközt a *VideoLayout* vagy a *GridLayout* részekben a megfelelő képre vagy ikonra klikkelve lehet kiválasztani. Az előugró (*DeviceDialog*) ablakban megjelenik immár az eszköz összes változójának az állapota, valamint kamera esetében a valós idejű videó felvétel is. Emellett ábrázolásra kerülnek az XML konfigurációban meghatározott funkciók, valamint kompakt formában más, az aktuálshoz csatolt eszközök is. Csatolt eszközre jó példa egy kamerához csatolt, ezt mozgató elektronikai eszköz.

Kamerák esetében a képre klikkelve egy újabb, ún. *Fullscreen* ablak ugrik elő, mely a teljes képernyős megjelenítést valósítja meg. Ha a kiválasztott eszköz egy kamera, akkor a minőségi képmegjelenítés érdekében úgy a *DeviceDialog*, mint a *Fullscreen* ablakok esetében a háttérben megjelenő kamerák leállítódnak az ablak lezártáig.

4.2 Sajátos funkciók

Az alkalmazás sajátosságai közt megemlítenő az esemény rendszer, a kliensek közti audio kommunikáció és az automatikus bejelentkezés opció.

Az XML leírásban minden eszköz paraméterlistájához több esemény csatolható. Egy esemény az eszköz meghatározott változójával kapcsolatos állapotváltozásokat értékeli ki, és bizonyos feltételek teljesítése esetén aktiválódik, egy előre meghatározott cselekvést váltva ki. A cselekvés lehet egy eszköz-ablak megjelenítése, egy hang lejátszása, vagy egyszerűen csak a történet elmentése.

Egy esemény több mezővel rendelkezik: változó, reláció, referencia, üzenet, hang fájl, timeout, perszisztencia, illetve cél-eszköz. Az alkalmazás minden ESS és Multimédia

kapcsolatról érkező üzenetet megvizsgál, annak érdekében, hogy kiszűrje és kiértékelje azokat, amelyek esemény kiváltó állapotváltozások vagy vezérlő üzenetek.

Egy aktív esemény élettartamát 2 tényező befolyásolja: a perszisztencia, illetve a timeout. A perszisztencia azt jelenti, hogy egy aktív esemény akkor sem fog inaktív állapotba kerülni, hogyha az azt kiváltó feltétel már nem teljesül. Abban az esetben ha egy aktív esemény nem vált inaktívvá perszisztenciájából kifolyólag, akkor a timeout másodpercekben adja meg azt az időt aminek eltelte után egy aktív esemény megszűnik, avagy inaktív lesz. Ily módon ha az esemény perszisztens és a timeout értéke végtelen, akkor csakis a felhasználó beavatkozására szűnhet meg.

Egy konkrét példa esemény a riasztó berendezéssel felszerelt házak esete, ahol ajtónyitás után a felhasználónak 10 másodperce van hatástalanítani a riasztót. Ilyenkor előnyös, hogyha az ajtó relléjéhez, vagy egy közeli mozgásérzékelőhöz egy olyan esemény van rendelve, amely szükség esetén megjeleníti a riasztó numerikus billentyűzetét, elmenti az eseményt és esetleg audió figyelmeztető jelet ad a felhasználónak.

Az Androidon kívül a más operációs rendszereken (Windows, Linux, iOS) futó kliensek, valamint a kliensként működő kapucsengő alkalmazások is külön eszközként jelennek meg a felhasználói felületen. Ezeknek az eszközöknek a funkciói hívás-kezdeményezést, megszüntetést illetve némítást tesznek lehetővé.

Alapértelmezetten hívás érkezésekor egy módosított eszköz ablak ugrik fel, mely a hívás elfogadását vagy visszautasítását teszi lehetővé. Ezt a viselkedést felül lehet írni az eseményrendszerrel, és hanghívásra egy olyan eseményt beállítani, amely egy meghatározott eszközt, például egy kapucsengő kameráját jelenít meg, melyhez a kapunyitó eszköz is csatolva van.

Fontos megemlíteni, hogy a minőségi hangátvitel érdekében párbeszéd közben egyszerre csak 1 kamera működhet.

A felhasználó kényelmének növelése érdekében az alkalmazás bejelentkezési ablakában beállítható az automatikus bejelentkezés egy elmentett konfiguráció alapján.

4.3 Több szál szerepe

Mivel az alkalmazás valós idejű és különböző forrásokból származó adatokat jelenít meg, szükséges az események párhuzamosítása a jobb teljesítmény elérése érdekében. Ennek

következtében az alkalmazás több szálon fut, amelyeknek száma nagyban függ az adatokat szolgáltató szerverek számától.

Minden kliens- szerver kapcsolat esetén a nagy forgalmú adatfolyam külön szálon fut, a valós idejű adatmegjelenítés érdekében. Továbbá külön programszálakat kapnak a nagy prioritású, sürgős eseményeket végző programrészek, vagy a rendszerdiagnosztikáért felelős komponensek.

Továbbá helyet kapott az alkalmazásban egy üzenetkezelő szál, amely várakozási sor segítségével kommunikál az alkalmazás többi részével és feladata főként a grafikus felület kezelése (üzenetek megjelenítése, grafikai állapotok megváltoztatása).

Úgy az XML szerverről, mint az ESS és Multimédia szerverről beérkező adatok is külön szálon kerülnek beolvasásra a TCP/IP kapcsolatokról és egy külön szálon kerülnek feldolgozásra.

Az ESS szerverről beérkező adatcsomagok a várakozási soron keresztül kerülnek a fent említett üzenetkezelő szálhoz, amely feldolgozza és megjeleníti az érkezett információt. Ez a megoldás nagyon gyors, ezáltal nem késlelteti az adatok valós idejű beolvasását és megjelenítését, viszont nem alkalmas nagy mennyiségű adat átvitelére, mint videó vagy hanganyag.

Mivel az audió és videó adatcsomagok egy és ugyanazon a TCP/IP kapcsolaton keresztül érkeznek be a Multimédia szervertől, kritikus az adatok feldolgozása és megjelenítése, mert az késést okozhat úgy a videó, mint az audió esetében is, ami nem elfogatható. Így egy gyors és jól szinkronizált mechanizmusra van szükség, amely az adatcsomagok prioritása szerint végzi az adatfeldolgozást a beolvasástól függetlenül.

Ez 3 programszál segítségével valósítható meg: az első szál végzi az adatlevételt az adatvonalról, állítja össze a teljes adatcsomagokat és küldi ki az adatokat feldolgozásra. A legnagyobb prioritású adat a hanganyag, amely dekódolást nem igényel, mert az Android operációs rendszer által támogatott PCM16 audió kódolásban érkezik a klienshez és egyenesen az operációs rendszer hanglejátszó objektumába kerül lejátszásra külön programszálon. A videó adatok dekódolásáért és megjelenítéséért külön szál felelős, amely szinkronizálása az olvasó szálhoz szemafor segítségével történik: míg az olvasó szál végzi az adatletöltést, addig a videóért felelős szál várakozik, amikor pedig elérhető a kész a képkocka, a feldolgozószál átmásolja az adatot egy csakis önmaga által látogatott memóriarészre,

felszabadítja a szemafor, dekódolja és megjeleníti a képkockát. Ezáltal az olvasó szál csupán az átmásolás idejéig van blokkolva, tehát a kritikus zóna a lehető legkisebb hosszúságú.

Az alkalmazás rendelkezik továbbá egy diagnosztikai szállal, amely periódikusan ellenőrzi a szerverkapcsolatokat, és ha megszakadt kapcsolatot diagnosztizál, újabb szálát indít, amely a visszacsatolódást végzi a szerverhez. Úgy a diagnosztikai, mint a visszacsatolásért felelős szál az operációs rendszer által biztosított Timer osztály segítségével van implementálva.

4.4 Öndiagnosztizáló funkció

Az alkalmazás állandó kapcsolatot kell biztosítson az automatizált eszközökkel a felhasználó számára és felmerülő kapcsolati hibák esetén a legrövidebb idő alatt kell biztosítsa a normál állapotba való visszakerülést. Mivel az Android eszközök mind vezeték nélküli kapcsolatok segítségével csatlódnak a világhálóra, a kapcsolatok stabilitása (állandósága, élettartama) korlátozott.

A stabil kapcsolatok érdekében az alkalmazás rendelkezik egy stabilitást felügyelő mechanizmussal. Ez meghatározott periódusonként ellenőrzi a kapcsolat állapotát, adatforgalom hiánya esetén *Ping* üzeneteket küld a szervernek, ellenőrzi a visszaérkezett *Pong* üzeneteket és leszakadt kapcsolat esetén elindítja az újracsatolódáshoz szükséges programrészt.

A kapcsolaton bármilyen olvasás/írás hiba következtében is elindul az újracsatolódo mechanizmus, mert a TCP/IP Socket bármilyen olvasás/írási kivétel (IOException) esetén inkonzisztens állapotba kerül.

Az újracsatolódást végző rész állandó periódussal próbálkozik új kapcsolatot létesíteni a szerverrel, sikeres csatlakozás esetén frissítést kér a szervertől, ezáltal aktualizálja a megjelenített állapotokat és kiküszöböli az adatvesztést és az inkonzisztens állapotok által kiváltott nem kívánt eseményeket.

4.5 Különös megoldások a megbízhatóság növelésére

Mivel az Android operációs rendszer működési elve alapján fenntartja a jogot a nemhasznált, vagy kiritikus esetben a használt alkalmazások lezárására is memóriahiány esetében, az alkalmazás állandó futása nem garantált. Ez teljesen ellentmond a program filozófiájának.

Ennek elkerülése érdekében készült el az ESS Reconnecter nevű alkalmazás, amely egy független *foreground service*, amely állandóan figyeli az alkalmazás futását és leállás esetén újraindítja azt.

Jelenleg ismeretlen okok miatt az Android operációs rendszer hosszú üzemidő esetén elveszítheti a WiFi kapcsolatot, amely az alkalmazás használhatatlanságát eredményezi. Ennek elkerülése érdekében a az ESS Reconnecter alkalmazás a WiFi állapotát is figyeli és ha szükséges, visszacsatolódást kezdeményez.

A fent említett alkalmazás nagyon egyszerű, kis memóriaigényű és a *foreground service* jellegéből adódóan állandó futása normál működési körülmények között garantált.

Az alkalmazás rendelkezik hibajelentési funkcióval is, ha bármilyen okból a program futása közben hiba történik, erről automatikus jelentést küld az alkalmazás a fejlesztő csapatnak. Ez bármikor elérhető egy webes felület segítségével, ezáltal minden hiba a fejlesztők tudomására kerül, kijavítható és ezáltal növeli az alkalmazás stabilitását.

4.6 Vezeték nélküli kapcsolatok

Mivel az alkalmazás minden körülmény között vezeték nélküli kapcsolat segítségével csatlakozik a szerverekhez említésre méltók ezen technológia előnyei és hátrányai. Ez a kapcsolat elsősorban WiFi kapcsolat lehet, amely nagy sebességű de korlátozott hatótávú vezeték nélküli kapcsolatot biztosít. Emellett használható az alkalmazás mobil internet segítségével is, amely esetén nagyobb a hatótáv de nagy valószínűséggel korlátozott a sávszélesség.

Tehát bármely esetben a vezeték nélküli kapcsolatok esetén előnyként hozható fel a térbeli függetlenség, viszont fontos megjegyezni, hogy ezen kapcsolatok stabilitása kritikus pont és rengeteg befolyásoló tényező sorolható fel: jelerősség, az eszköz minősége (WiFi router), az aktuális térségben elhelyezkedő vezeték nélküli eszközök darabszáma.

Az ismert protokollok alapján ha két vezeték nélküli protokollon kommunikáló eszköz egy időben próbál egyazon csatornán kommunikálni, az adatütközés érdekében mindkét csomag újraküldést igényel, ezáltal az egy adott környezetben jelenlevő eszközök darabszámának növekedésével arányosan romlik a vezeték nélküli kommunikáció minősége.

A hétköznapi életben elterjedt wireless routerek nem ajánlanak fel beállítási lehetőséget a WiFi csatornákat illetően, ez a beállítás csak az összetettebb routereknél

állítható, ezáltal a wireless kommunikáció minősége nő a használt routerek teljesítményével, amely persze szoros összefüggésben áll azok árával is.

Ezen alkalmazás-rendszer esetén minden esetben olyan routerek kerülnek felszerelésre, amelyek OpenWRT operációs rendszer futtatására alkalmasak, ezáltal nagyban megnő a beállítási lehetőségek száma is, ami a biztonságot és a stabilitást is növeli.

A valós szerelések esetén fontos szerepet játszik a wireless hálózat kiépítése az automatizált ház területén, mert elsősorban itt kerülnek használatra az alkalmazást futtató telefonok, tabletek. A megfelelő jelfedettség érdekében érdemes a ház különböző részein több wireless routert is elhelyezni. Az egyszerű kezelhetőség érdekében mindenik router ugyanazt az SSID-t és csatlakozási jelszó beállítást kapja, így a hálózat egy nagy egységes WiFi rendszerként működik, a mobil eszköz bármikor átcsatlakozhat egyik router-ről a másikra ugyanazzal a kapcsolati konfigurációval a jelerősség függvényében.

4.7 Teljesítmény paraméterek

A letöltési elfoglalt sávszélesség nagy részét a kamerák képanyaga képezi, ez az eszközök számától függetlenül kb. 130 KB/s. A feltöltési sávszélesség nagyon csekély emellett, mindössze 2-3 KB/s.

Próba felület Eszözök	ASUS Transformer (Android 4.0)		HTC Desire (Android 2.1)	
	CPU igény [%]	Memória [MB]	CPU igény [%]	Memória [MB]
1 kamera	10	30	17	25
2 kamera	11	34	21	31
3 kamera	11	39	25	37
7 kamera	14	80	38	54
5 eszköz	7	28	16	23
10 eszköz	9	31	18	26
3 kamera 5 eszköz	12	50	28	42
4 kamera 10 eszköz	13	70	35	50
7 kamera 20 eszköz	17	110	48	80

5. Következtetések

Az alkalmazás kialakításában fontos szerepet játszott a rendszerigény problémája, így a futtatáshoz szükséges operációs rendszer az Android 2.1 (vagy újabb verziók), továbbá erőforrás- és memória igényét a jelenleg piacon levő összes Android operációs rendszerrel ellátott eszköz kielégíti, ezáltal hardver inkompatibilitás nem állhat fenn.

A rendszer valós használat alatt is tesztelt, egy luxus lakóház teljes automatizációját biztosítja 2011. Decembere óta, megbízhatóan és hibamentesen. Ez bizonyítja, hogy a rendszer képes zavartalan és stabil működésre nagy időintervallumokban is.

Valós használat közben megfigyelhető, hogy olyan esetekben mikor az Android eszköz két WiFi hálózat hatótávolságán belül helyezkedik el, előfordulhat a két router közti oszcillációs jelenség, ezért kritikus a router-rendszer térbeli elhelyezése.

A grafikus felület egyszerű, de esztétikus, jól áttekinthető és könnyen használható, ezáltal könnyű kezelhetőséget biztosít akár technikai ismeretekkel nem rendelkező hétköznapi emberek számára is.

Az alkalmazás kialakítása révén internetes kapcsolat nélkül is képes működni, belső hálózaton, ezáltal az internet szolgáltatás szünetelése nem befolyásolhatja a helyszíni zavartalan működést.

6. Egyéni hozzájárulás

Ezen projekt fejlesztése 2011. júniusában kezdődött egy nyári szakmai gyakorlat keretén belül az S.C. IntegraSoft S.R.L. szoftverfejlesztő cégnél és jelenleg is folyamatban van.

A hardver eszközök, a kapcsolatteremtő szerveralkalmazások és a számítógépes kliensalkalmazások a projekt kezdetekor már rendelkezésre álltak, de ezek tesztelésében, továbbfejlesztésében, valamint a hibák kijavításában aktívan részt vállaltunk.

Részt vettünk továbbá a SIP protokollt használó kaputelefonok illesztését biztosító SIP átjátszó alkalmazás tervezésében, fejlesztésében és tesztelésében.

Az Android operációs rendszereken futó kliensalkalmazás tervezése, fejlesztése és tesztelése teljes mértékben a mi munkánk, amely úgy funkcionalitás, mint megbízhatóság terén túlmutat a más platformokon futó kliensalkalmazásokon.

7. Jövőbeli tervek

A Multimédia szerver saját hívás-fogadás protokolljának SIP protokollra való cserélése sürgőssé tenné a SIP átjátszó modul használatát, ezáltal megszűnne az átjátszás által bevitt késés. Következésképpen a SIP protokoll használatát az Android kliensalkalmazásnak is implementálnia kell.

A nagy képernyőméretű tabletek lehetővé teszik a készülékek alaprajzos ábrázolását is, ezáltal implementálható az Android kliensbe ezen képernyők felismerése és a készülékek az épületek alaprajzán való ábrázolása.

A kliensalkalmazások között létrehozható audió kapcsolat könnyedén kiegészíthető videó adatátvitellel, ami továbbfejleszthető és akár online konferencia lebonyolítására is alkalmassá tehető az Android kliens alkalmazás.

A piacradozás új tesztelési környezeteket és az új igényeknek megismerését eredményezheti, amely az alkalmazás továbbfejlesztésében játszik nagy szerepet, ezáltal cél a teljes rendszer népszerűsítése.

8. Könyvészet

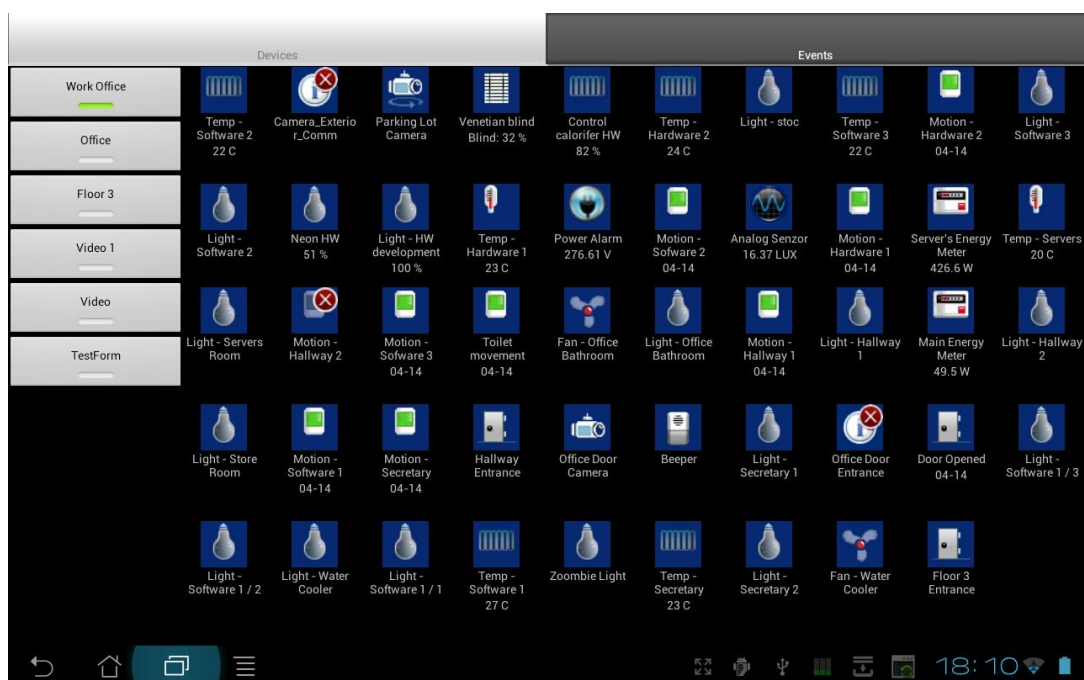
[1] Android Developers – developer.android.com

[2] Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, Doug Lea: *Java concurrency in practice*, Addison-Wesley Professional kiadó, 2006

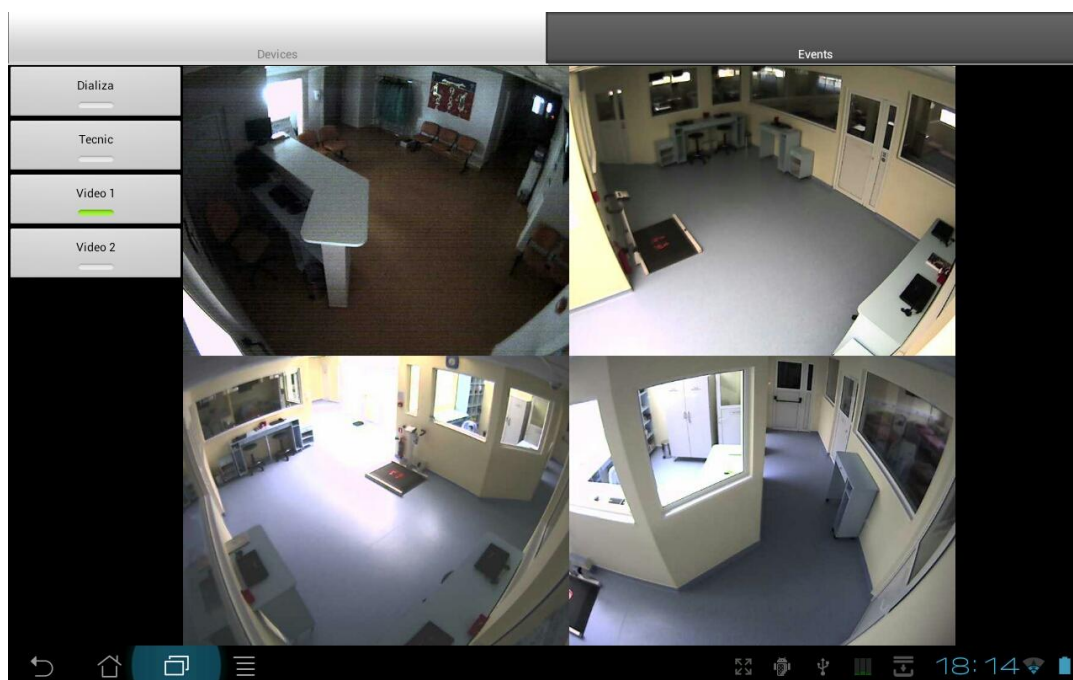
[3] SIP and Media Documentation – www.pjsip.org

[4] Intelligent Building Solutions – www.ibs-smarthouse.com

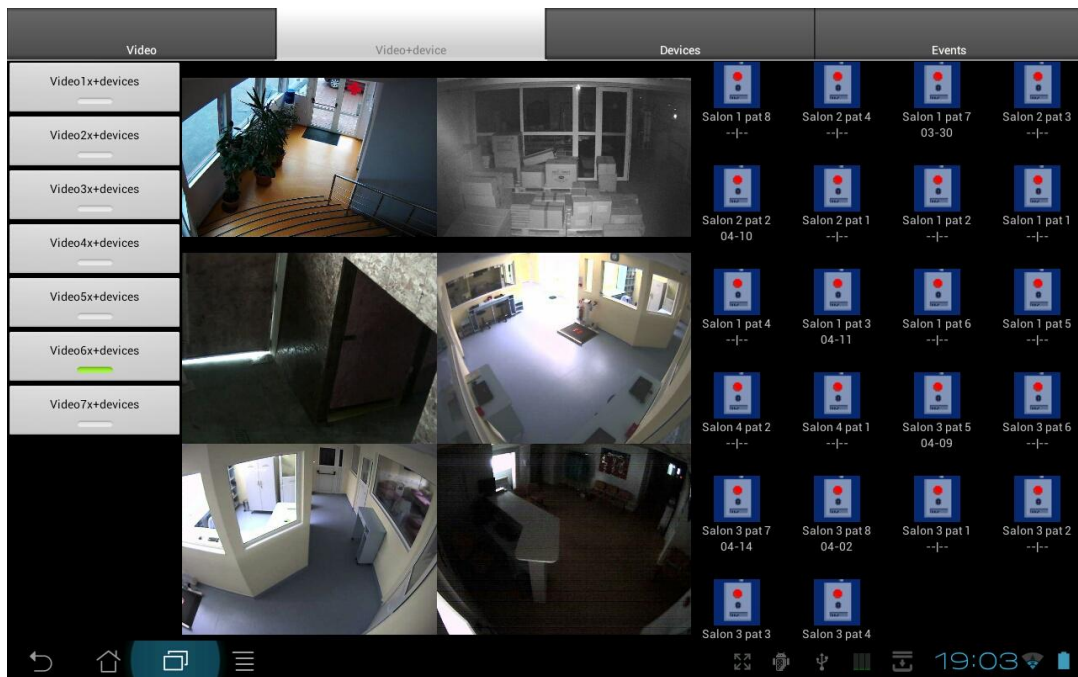
9. Függelék



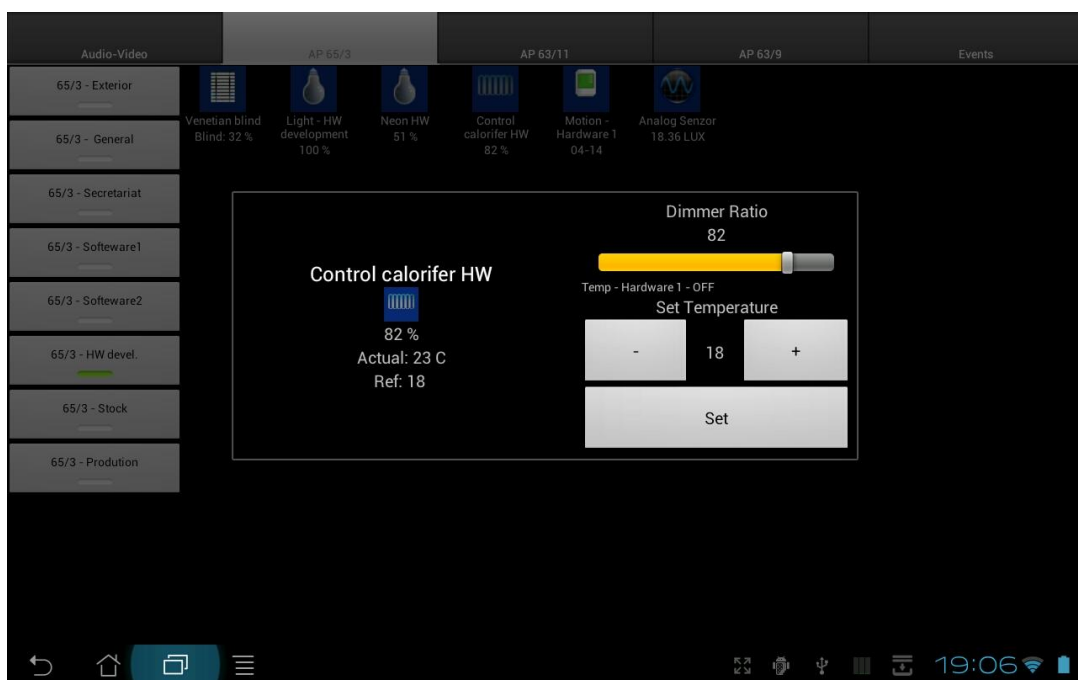
1. Függelék. Grafikus felület, eszközök táblázatos megjelenítésben



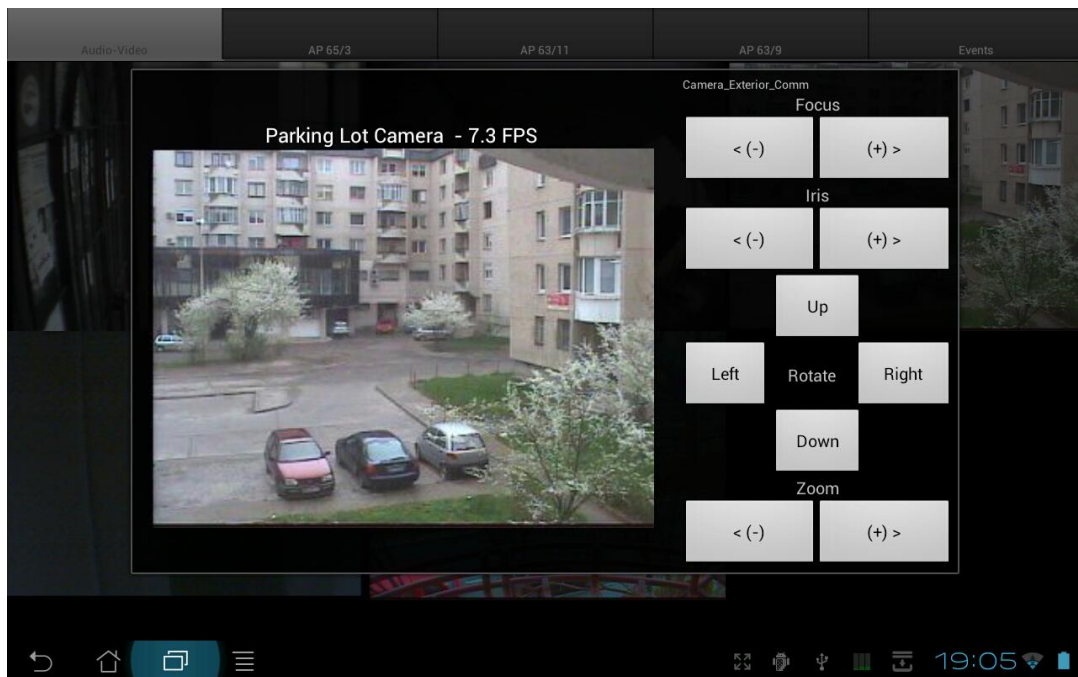
2. Függelék Grafikus felület, kamerák állandó közvetítésre megjelenítve



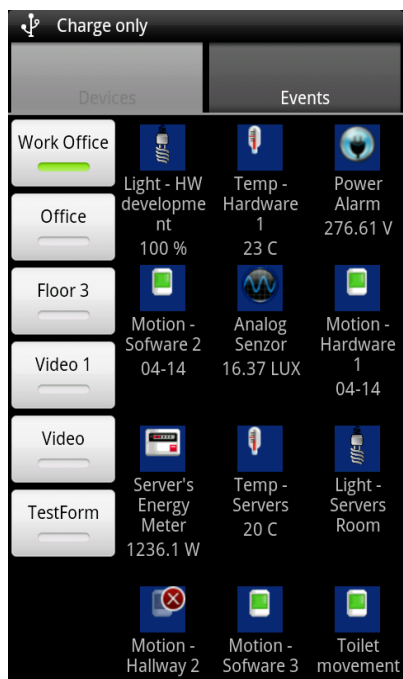
3. Függelék Grafikus felület, kamerák és eszközök



4. Függelék Grafikus felület, fűtésszabályzó eszköz előugró ablakban(Device Dialog)



5. Függelék Grafikus felület, kamera nézet előugró ablakban(Device Dialog)



6. Függelék Grafikus felület telefonon