Recent Innovations in Mechatronics (RIiM) Vol. 5. (2018). No. SI.
DOI: 10.17667/riim.2018.si/8.

# *Android Linux alapú alkalmazás fejlesztése & RFID beléptető rendszer monitorozása*

Uszkai Sándor
Debreceni Egyetem,
InformatikaiKar
Debrecen, Magyarország
uszkaisanyi@gmail.com

Papp Beatrix
School of law and social sciences
Criminology with psychology
United Kingdom, London
pappb@lsbu.ac.uk

Erdei Timotei István
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
timoteierdei@eng.unideb.hu

***Absztrakt*— Napjainkban az IoT eszközök egyre szélesebb körben vannak jelen az iparban és ez a trend kezd elterjedni a kis- és középvállalkozások körében is. Az efféle új technológiai ágazatok az Industry 4.0 megjelenése óta robbanásszerű növekedésnek indultak. Az informatikusok számára új kihívást jelent ez az új irányzat, mivel a korábbinál alaposabb ismeretekre van szükségük a számítógépes hálózatok területén, valamint a hálózatra kötött eszközöket nemcsak programozniuk kell megtanulni, de ismerniük kell azok egyes elemeit és működésüket. Projekt során egy olyan beléptető rendszert került létrehozásra, ami nemcsak a nagyvállalatok, de a kisebb vállalkozások, valamint magánszemélyek számára is megfizethető, és könnyen kezelhető.**

**Kulcsszavak—*Android; RFID; adatbázis; Raspberry Pi; Arduino; IoT; Industry 4.0***

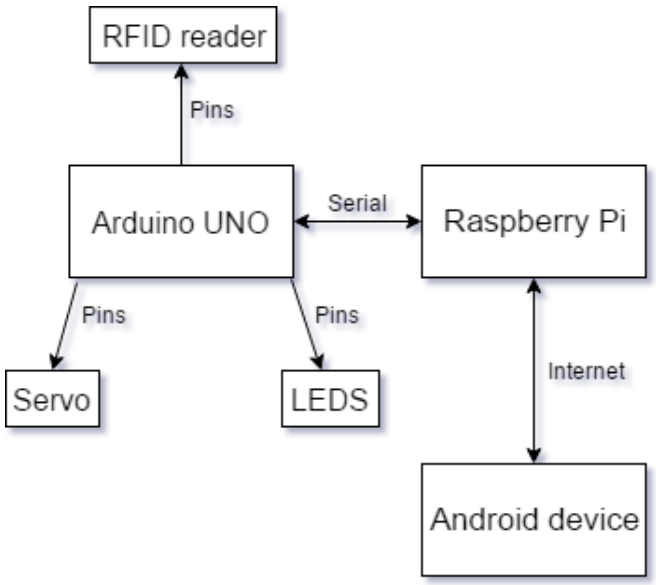
## I. BEVEZETŐ

A legújabb technológiai vívmányok és irányzatok szembe mennek a korábbi látásmóddal, miszerint a gépeket óvni kell az internet okozta veszélytől. Ez az új tendencia lehetővé teszi a gépek és berendezések hálózatra való csatlakoztatását, ami megkönnyíti azok kezelését, működésük megfigyelését, viszont alaposabb kiberbiztonsági intézkedések végrehajtását követelik meg [1].

A munka során a már kipróbált és bevált technológiák segítségével internet hozzáféréssel rendelkező IoT szemléletű beléptető rendszer lett létrehozva a Debreceni Egyetem Villamosmérnöki és Mechatronikai Tanszéke számára. A rendszer minimális átalakítással alkalmazható vállalatok számára, illetve magánszemélyek által való felhasználásra is.

## II. TERVEZÉSI SZEMPONTOK

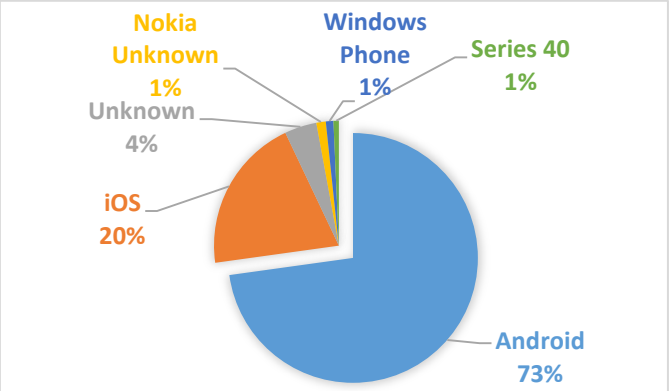
A Debreceni Egyetem Műszaki Karának B épületében számos kutatólaboratórium található, amelyek nagy értékű gépekkel és robotokkal vannak felszerelve. Ebből kifolyólag világossá válik, hogy a termekbe való belépést szigorú szabályozásnak kell alávetni. Az internet korában szinte már elképzelhetetlenné vált, hogy biztonsági megoldásként a hagyományos kulcsos megoldást alkalmazzuk, ezért esett a választás az RFID technológiára. Mivel a rendszer többféle hardver- és szoftverelemet tartalmaz, a tervezés legfontosabb kérdése az volt, hogy lehet-e ezeket az eszközöket és programozási nyelveket együttesen alkalmazni, valamint az, hogy ez hogyan kerüljön megvalósításra. A kompatibilitási kérdések leküzdése érdekében a fizikai eszközök kiválasztásánál azok kerültek előtérbe, amelyek paraméterei publikusak. A projekt szoftver komponenseinél elsősorban a platformfüggetlen programozási nyelvek élveztek előnyt.




1. ábra: A rendszer elemei és azok kapcsolódási interfészei

A rendszer biztonságos működése érdekében a belépések naplózására szolgáló adatbázis Linux alapú operációs rendszeren működik, mivel ezek a rendszerek fokozott biztonsági komponensekkel vannak ellátva, valamint sokkal ritkábbak a sikeres támadások ellenük.

A mobilalkalmazás fejlesztése során az Androidra esett a választás, mivel ez a legelterjedtebb operációs rendszer napjainkban, ahogy ezt a StatCounter [9] 2017 áprilisi statisztikája is mutatja (2. ábra).




2. ábra: A mobil operációs rendszerek elterjedtsége 2017 áprilisában

## III. A RENDSZER FIZIKAI FELÉPÍTÉSE

A beléptető rendszer fizikai eszközeinek fontos eleme a szervomotor. Ez valósítja meg az ajtók ki-be zárását. Ehhez egy Adafruit TowerPro SG-5010 [2] típusú szervo került felhasználásra. Mivel a 39g tömegű eszköz 5.5kg-cm forgatónyomatéka viszonylag magas, ezért alkalmas a zárak eltolására. A kiválasztásnál további szempont volt, hogy viszonylag kis feszültségről működjön az eszköz. Mivel az optimális feszültségként a gyártó 5V-ot határozott meg, ezért kompatibilis az Arduino UNO fejlesztőpanellel.

Az UNO az Arduino gyártó egyik belépő szintű terméke, ami ideális kezdő fejlesztők számára, de bonyolultabb rendszerek vezérlésére is alkalmas. Az eszközt egy ATmega328P típusú, 16 MHz órajelű mikrokontroller vezéreli. 14 digitális pinnel rendelkezik, ezek közül 6 képes az impulzusszélesség-modulációra (PWM). További 6 analóg pin áll rendelkezésre az analóg eszközökkel való kommunikációra, a tesztelési fázisban ezen pinek egyikébe volt bekötve a potenciométer, amivel a szervomotor szükséges forgási szöge került meghatározásra. A panel további be-és kimenetei a tápellátásért, illetve a földelésért felelősek.

Az Arduino UNO előnye, hogy nyilvános a specifikációja, így az egyes elemeiről könnyedén megtalálható minden információ a gyártó weboldalán [3]. Ezen felül az Arduino IDE saját fejlesztésű integrált fejlesztői környezetet biztosít, amivel a forráskódokat szerkeszteni, fordítani, illetve feltölteni is lehet.

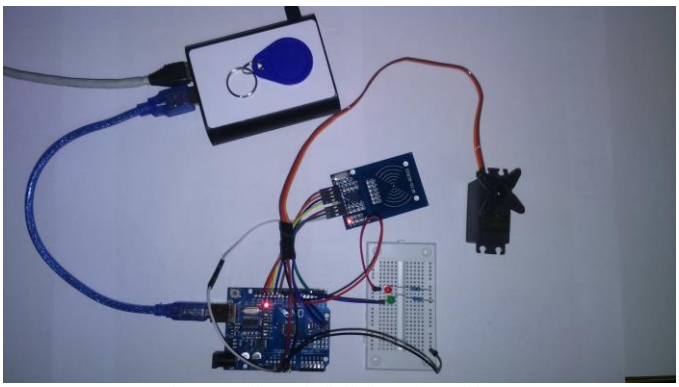
A megépített rendszer a felhasználók azonosítására RFID olvasót tartalmaz. Az RFID-RC522 eszköz az UNO alábbi táblázatban megadott pinjeire csatlakozik:

1. TÁBLÁZAT CSATLAKOZÁSI PONTOK

| RFID–RC522 | Arduino UNO |
|---|---|
| SDA | Digital Pin 10 |
| SCK | Digital Pin 13 |
| MOSI | Digital Pin 11 |
| MISO | Digital Pin 12 |
| RST | Digital Pin 3 |
| 3.3V | 3.3V |
| GND | GND |

Az RFID technológia előnye, hogy kényelmesen, egy érintéssel azonosíthatják magukat a belépésre jogosultak, ráadásul működik kártyával, illetve ún. tag-ekkel is. Mivel minden kártya vagy tag egyedi azonosítóval rendelkezik, ezért a belépés csak az adatbázisban szereplő, és arra jogosult ID-vel rendelkező kártyatulajdonosok számára biztosított.

A rendszer állapotának jelzését két darab LED biztosítja. Ezeket egy próbapanelen keresztül csatlakoztattam az Arduinohoz: a piros színű LED-et a 7-es, míg a zöldet a 6-os számú pinre. Ezek digitális pinek, amik kétállapotúak: magas- és alacsony feszültséget tudnak biztosítani. Magas feszültségen a LED világít, míg alacsony feszültség esetén elalszik. A magas feszültség esetén fellépő károsodások elkerülése esetében az áramkörben 1-1 100Ω-os ellenállást is elhelyeztem. Ezeket az Arduino közös GDN pinjére kötöttem a próbapanelen keresztül. A kártyaérintés után az Arduino soros porton keresztül elküldi a kártya azonosítóját a Raspberry Pi-nak, ami az adatbázisban való lekérdezés után igaz/hamis értéknek megfelelő egész számot küld vissza a soros porton keresztül attól függően, hogy az azonosító szerepel-e az adatbázisban vagy sem. Abban az esetben, ha jogtalan a hozzáférési kísérlet, felvillan a piros LED, majd némi késleltetés után elalszik. Ellenkező esetben a zöld LED villan fel, majd 5 másodpercig ebben az állapotban marad. Eközben a szervomotor elfordul, ami az ajtó nyitását biztosítja. A várakozási idő eltelte után a szervo visszaáll az eredeti állapotba, az ajtó bezárul, kialszik a zöld LED, és a rendszer újból várakozik a következő interakcióra.



3. ábra: Az elkészült fizikai rendszer felépítése

## IV. ARDUINO SKETCH

A sketch megalkotása során elsősorban az átlátható kód létrehozása volt a kitűzött cél, valamint a gyorsaság maximalizálása. A kódolás gyorsítása érdekében előre megírt RFID.h header került a forráskódba, ami az RFID modulhoz szükséges függvényeket tartalmazza. Ezen kívül a szervo vezérléséhez elengedhetetlen Servo.h header fájl is importálva lett a forráskódba.

A következő egység a kimenetek definiálását tartalmazza. Itt történik meg a különböző eszközök (például LED-ek) párosítása az Arduino UNO megfelelő kimenetével. Továbbá a Servo osztály példányosítása szintén ebben a kódrészletben valósul meg, mivel minden szervohoz egy objektumot kell

deklarálni, majd az objektum attach() függényét a megfelelő kimenet számával megegyező egész értékű paraméterrel meghívva történik meg a kimenet-szervo párosítás.

A kártyák azonosítója 5 számjegycsoportból áll, ezért ennek tárolására egy ötelemű tömb létrehozása szükséges. Az átláthatóság érdekében a speciális célt szolgáló függvények és eljárások külön kerültek deklarálásra a forráskód ezen szakaszában, az implementációjuk a forráskód végén van megvalósítva.

Mint minden Arduino forráskód, az általam írt szintén két fő részből áll: a setup() és loop() eljárásokból. A setup() eljárásban olyan kifejezések és eljáráshívások találhatóak, amik egyszer futnak le, és benne maradnak a memóriában. Az adott esetben a setup rész a szervo és Arduino kimenet egymáshoz rendelését szogáló „myservo.attach(9);" kifejezéssel kezdődik. A myservo a Servo osztály egy példánya, a paraméter értéke a digitális pin számát adja meg. A következő sorban az Arduino - Raspberry Pi kommunikációhoz elengedhetetlen soros port kerül definiálásra. A paraméterül adott 9600-as érték baud mértékegységben értendő, ami a másodpercenként küldött/fogadott jelek számát adja meg. Ezután az RFID eszköz konfigurációs eljárásai, valamint a zöld és piros LED-hez párosított pin kimenetként való beállítása szerepel. Végül a szervót $20^o$ -os szögelfordulásra állítjuk be, hogy az előző állapotától függetlenül mindig ugyanabban a pozícióban legyen a rendszer elindításakor. A $0^o$-$10^o$ és $170^o$-$180^o$ közötti tartomány kerülendő, károsíthatja az eszközt.

```
void indication(int led);
void allow();
void denied();
int pos = 0;
void setup()
{
  myservo.attach(9);
  Serial.begin(9600);
  SPI.begin();
  rfid.init();
  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  myservo.write(20);
}
```

4. ábra: A forráskód setup része

A setup részt a loop rész követi. Ez az eljárás tartalmazza az ismétlődő utasításokat, amelyeket az Arduino mindaddig ismétel, amíg új kódot nem töltünk fel rá, illetve amíg áram alatt van.

Ahhoz, hogy azonosítani tudja az olvasó a felhasználók által használt belépőkártyákat, ellenőrizni kell, hogy a kártyaazonosító megadott tartományon belül van-e. Hamis esetben újról kezdődik a ciklus, igaz esetben az RFID eszköz leolvassa a belépőkártya azonosítóját, és kiírja azt a soros portra, ahonnan majd a Raspberry Pi-n futó Python szkript ellenőrzi, hogy szerepel-e az adatbázisban az adott számsorozat. Annak érdekében, hogy a lekérdezés biztosan végrehajtódjon mire az Arduinonak szüksége lenne annak eredményére, 1 másodperces késleltetést állítottam be.

A szkript 1-es karaktert küld vissza (aminek ascii kódja a 49) a soros porton keresztül, ha a lekérdezés eredményes volt, és indulhat az ajtó nyitása – meghívódik az allow() eljárás. Ennek működése a következő: a zöld LED világítani kezd, eközben a szervo beáll $90^o$-os szögbe. Ezután a rendszer várakozik 5 másodpercig, majd a motor visszaáll eredeti állapotára, és újraindul a loop rész.

Ellenkező esetben a piros LED ég 1 másodpercig, jelezve a beléptetés sikertelenségét, aztán az eszköz elölről kezdi az ismétlődő utasításokat.

```
void loop(){
  String a;
  if (rfid.isCard()){
    if (rfid.readCardSerial()){
      for (i = 0; i < 5; i++){
        a += rfid.serNum[i];
        reading_card[i] = rfid.serNum[i];
      }
      Serial.println(a);
      delay(1000);
      if (Serial.available() > 0){
        if ( Serial.read() == 49)
          allow();
        else
          denied();
      }
    }
  }
  rfid.halt();
}
```

5. ábra: A forráskód loop része

## V. A RASPBERRY PI KONFIGURÁLÁSA

Mivel a beléptető rendszer egyszerre egy kártyát képes kezelni, ezért az adatbázis szerveréül szolgáló fizikai egység kiválasztásánál a teljesítmény nem élvezett előnyt a költséghatékonysággal szemben. Erre a célra a projektemhez egy Raspberry Pi 2 Model B, alacsony árú, bakkártya méretű számítógépet használtam fel. Előnye, hogy többféle Linux alapú operációs rendszer futtatására alkalmas. Tárhelyként egy 8 GB méretű micro SD kártya szolgál. A rendszer gördülékenysége érdekében class 10-es gyorsaságú kártyát helyeztem el, ami az egyik leggyorsabb olvasási és írási sebességű típus.

2. TÁBLÁZAT RASPBERRY PI PARAMÉTEREI

| | |
|---|---|
| Processzor | ARM Cortex-A7 |
| Processzor órajele | 900 MHz |
| Magok száma | 4 |
| Memória | 1 GB |
| GPIO pinek száma | 40 |
| Portok | 4 USB 2.0 |
| | Ethernet |
| | 3.5 mm jack |
| | Micro SD bemenet |

A Linux rendszerek előnye a Windows rendszerekhez képest a nyílt forráskód, valamint a legtöbb ingyen elérhető

bárki számára. Ez fontos szempont volt, mivel az iparban is a szerverek többsége Linux alapú.

A Raspberry Pi eszközön egy LAMP [4] szerver került felkonfigurálásra. A LAMP mozaikszó a Linux, Apache, MySQL, PHP szavak kezdőbetűiből áll össze, ami már a nevével is elárulja a felhasznált technológiákat.

Az eszközre operációs rendszerként a Raspbian legújabb verziója, a Jessie került feltelepítésre. Ennek oka az, hogy a Raspbian a Raspberry termékcsalád legjobban támogatott rendszere, továbbá a Jessie verzió a legújabb, és hosszú ideig nem szükséges lecserélni vagy frissíteni újabb verzióra. A telepítés után letiltásra került a grafikus környezet, ezzel is tehermentesítve a processzort. A távoli hozzáférést SSH vagy PuTTy segítségével történik meg, így a miniszámítógépre annak IP címét, valamint felhasználónevét és jelszavát ismerve be lehet lépni PC-ről is. Ez a megoldás kiváltja a monitor használatát a Raspberryhez.

Az Apache a legelterjedtebben használt webkiszolgáló a Linux rendszerek körében. Ez a webkiszolgáló képes a kliensek (jellemzően böngészők, de lehetnek más alkalmazások is) által kért weboldalak kiszolgálására. Támogatja a HTTP (HyperText Transfer Protocol), HTTPS (Hypertext Transfer Protocol Secure), valamint az FTP (File Transfer Protocol) protokollokat.

Az Apache 2 interfészt szolgáltat a mobilalkalmazás számára a szerverhez, ezen keresztül történnek a HTTP kérések, amelyek elengedhetetlenek az Android applikáció, valamint az Arduino működését befolyásoló Python szkript, és az adatbázis közötti kommunikáció biztosításához.

Az adatbázis kiépítésénél fontos szerepet játszott, hogy ingyenes, nyílt forráskódú megoldás kerüljön felhasználásra, így a választás a MySQL adatbázisra esett. Előnye, hogy az adatbázis-kezelő rendkívül sokféle programozási nyelv segítségével támogatását élvezi: PHP, C++, Java, Delphi, Lisp, Perl, Python, Ruby stb. Az elkészült beléptető rendszer adatbázisának lekérdezéseinek futtatásához a PHP 5-ös verzióját választottam, mivel ez a leggyakoribb párosítás a MySQL adatbázisokhoz. A táblák vizuális tervezéséhez és létrehozásához a phpMyAdmin programot használtam fel.

## VI. AZ ADATBÁZIS FELÉPÍTÉSE

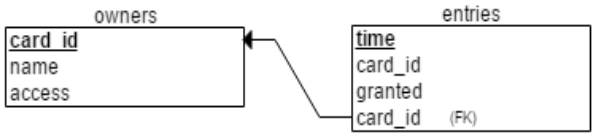
Az adatbázis séma az enterlog nevet kapta. Ez egy két táblából álló séma, amely a felhasználók adatait, valamint a belépések adatait tartalmazza. A táblák UTF-8 karakterkódolást használnak annak érdekében, hogy az ékezetes karakterekkel is működjenek a lekérdezések.

Az első, owners nevű, felhasználói adatokat tároló tábla az alábbi oszlopokat tartalmazza: card_id, name, access.

A card_id bigint típusú értékeket képes tárolni, ebben az oszlopban szerepelnek a felhasználók belépő kártyáinak azonosítói.

A card_id az owners tábla elsődleges kulcsa, tehát nem szerepelhet két sorban egyazon érték, valamint nem lehet az értéke NULL. A name nevű, text típusú mezőben a felhasználó neve szerepel. Az access mező tinyint típusú egész értékek tárolására alkalmas. Ennek a mezőnek 0 vagy 1 értéket adunk a felhasználók létrehozása vagy adatmódosítás során. 1-es érték esetén a Python szkript elindítja a beléptetést a lekérdezés után, 0-s érték esetén megtagadja az engedélyt.

A második tábla az entries nevet viseli. Első oszlopa a card_id, amely az owners tábla egyező nevű értékével azonos, a belépést kezdeményező kártya azonosítója alapján. Második mezője a time, ami datetime típusú, ez a belépés dátumát és időpontját tárolja. A time oszlop a tábla elsődleges kulcsa, mivel egyszerre nem lehetséges két belépés, ezért biztosan egyedi lesz, és minden esetben lesz értéke. Az utolsó mező a granted, ami azt mutatja, hogy sikeres volt-e a belépési kérelem.




6. ábra: Az adatbázis táblái és azok kapcsolatai

Az adatbázisból való lekérdezésért az Arduino oldaláról egy Python 3 [5] nyelvű program gondoskodik, az Android alkalmazás által kért adatokat pedig PHP szkript biztosítja.

A Python nyelvű program célja, hogy az Arduino eszköz soros portjáról beolvassa a kártya azonosítóját, majd ezzel az értékkel megegyező sort keres az owners tábla card_id oszlopában. Ezt követően a szkrip ellenőrzi, hogy a granted oszlopban milyen érték szerepel.

A granted mező a felhasználók adatbázisból való tiltását gyorsítja meg, a teljes rekord törlése nélkül. Ha részlegesen fel szeretnénk függeszteni valakinek a hozzáférési jogát egy helyiséghez (pl. a szabadságát tölti éppen), nincs más teendő, mint ennek a mezőnek az értékét nullára állítani.

Amennyiben sikeres volt a lekérdezés, a beleegyezésnek megfelelő 1-es ascii karaktert küldi vissza az Arduinonak a soros porton keresztül.

A PHP szkript célja az Android eszköz által igényelt adatbázis információk szolgáltatása. A forráskódban előre definiált SQL lekérdezések futtatását is ez a program végzi. Amikor a lekérdezés lefut, az eredmény egy tömbbe kerül. Ezt az Android rendszer nem tudja közvetlenül értelmezni, ezért szükséges a kimenet átalakítása.

Mivel az Apache web szerverek HTTP-n keresztül kommunikálnak, ezért csak szöveges fájlokat tudunk vele küldeni. Ennek érdekében az SQL lekérdezés eredményét a szkript JSON formátumra konvertálja. A JSON szöveg alapú, kis méretű állomány, emberi olvasásra alkalmas. További előnye, hogy bár a JavaScript nyelvből alakult ki, mégis platform független. Az elkészült JSON állomány továbbítódik ezután a hálózaton keresztül a mobilalkalmazáshoz. A 6. ábra mutatja a PHP szkriptet, a jelszó cenzúrázásával.

```php
<?php

$host = "localhost";
$user = "root";
$password = "password";
$db = "enterlog";

$sql = "SELECT b.time, a.name, a.card_id
        FROM owners a
        join entries b
        on (a.card_id = b.card_id)
        ORDER BY b.time DESC;";
$con = mysqli_connect($host,$user,$password,$db);
$result = mysqli_query($con,$sql);
$response = array();
while($row = mysqli_fetch_array($result))
{
array_push($response,
    array("time"=>$row[0], "name"=>$row[1], "card_id"=>$row[2]));
}
echo json_encode(array("server_response"=>$response));
mysqli_close($con);

?>
```

7. ábra: Szerver oldali PHP szkript

## VII. ANDROID ALKALMAZÁS

A mobileszközök robbanásszerű fejlődése és a tabletek megjelenése óta az Android napjaink egyik vezető operációs rendszere. A beléptető rendszer nyomon követése kényelmetlen lenne asztali számítógépről, SQL parancsok kézzel történő futtatásával. Innen eredt az az ötlet, hogy mobil készülékkel történhessen meg a rendszer ellenőrzése.

Mivel számos képernyőméretű, ár kategóriájú, kibocsátású, márkájú készülék létezik, valamint a tabletek is viszonylag elterjedtek, a fejlesztők számára fontos, hogy az általuk elkészített alkalmazás a lehető legtöbb eszközön fusson akadás mentesen. Ennek érdekében a projekt megkezdésekor minimum követelménynek az Android 4.0.3. verzióját választottam, célverziónak pedig a legújabb, 7.1.-es verziót, hogy kihasználhassa az újabb készülékek erőforrásait is. A fejlesztést az Android Studio [6] segítségével végeztem. Ennek adatai szerint a projekt elkezdésének idején (2017 március) a Google operációs rendszerét futtató eszközök több, mint 97%-a volt képes az alkalmazás futtatására, ebbe beleértve a mobileszközöket, illetve a tableteket is.

Az applikáció az RFIDoor nevet viseli. 2 activity-t tartalmaz: az első a MainActivity, a második a DisplayListView activity. Az Android alkalmazásoknak 4 fő komponense van: activity, service, content provider és broadcast reciever. Az activity-k hivatottak a felhasználóval való interakcióra, továbbá itt hozhatunk létre objektumokat, osztályokat definiálhatunk stb.

A MainActvity az RFIDoor alkalmazásban a hálózati kommunikáció kialakításáért, valamint a JSON szétdarabolásáért, feldolgozásáért felelős. A felhasználói felületet az Android alkalmazásokban XML nyelven adjuk meg. Jelen esetben ez egy LinearLayout elrendezést foglal magába, ami tartalmaz két Button elemet. Ezek a képernyőn látható gombok megjelenését és szövegét írják le.

A felső gombra való kattintás során egy új szál indul el a program futása során. Ez azért szükséges, mert a felhasználói felületet biztosító száltól elkülönítve kell végrehajtani a hálózati kommunikációt használó metódusokat ezen az operációs rendszeren. A háttérben lefut a doInBackground() metódus, ami kiépíti a HTTP kapcsolatot a MySQL szerverrel, majd megkapja eredményül a JSON fájlt, amit egy sztringben tárol el. Ebből a sztringből fogja később felépíteni az alkalmazás a táblázatot, ami az adatbázis sorait jeleníti meg.

Az alsó gomb működése attól függ, hogy előtte sikerült-e letölteni a JSON formátumban tárolt rekordokat. Amennyiben a felső gombra előtte nem kattintott a felhasználó, vagy nem sikerült a letöltés (pl. nem volt internet kapcsolat), egy felugró toast ablakban kapunk utasítást arról, hogy először töltsük le az adatokat. Abban az esetben, ha már a gombnyomást megelőzően sikeresen letöltöttük az adatokat, létrejön egy intent objektum, ami a következő activity elindításáért felelős. Mivel az intenteknek lehet megadni attribútumokat is, az adatbázisból való lekérdezés adatait tartalmazó sztringet is átadjuk ennek az objektumnak, hogy később más activityből is hozzáférhető legyen. Amikor meghívjuk a startActivity() metódust az adott intent paraméterrel, az operációs rendszer elindítja a DisplayListView activityt.


8. ábra: A MainActivity futás közben

Az applikáció második részegysége a DisplayListView activity. Itt jelenik meg a letöltött adat, táblázat formában. Ehhez két új Java osztályt volt szükséges létrehozni. Az Entries osztályban a táblázat egy-egy sorának attribútumai vannak definiálva. Az EntryAdapter osztály az ArrayAdapter ősosztály leszármazottja, ami a sorok tárolásáért felelős. Külön XML erőforrás biztosítja az egységes megjelenését. A hívó activityben létrehozott intent objektumból egy metódus segítségével megkapató a táblázat tartalmát adó sztring. Ezután a szöveges állomány feldarabolásra kerül, majd JSON objektumok tömbjévé alakítja át az algoritmus. Az EntryAdapter osztály objektuma felépíti a táblázatot, majd megjelenítésre kerül a képernyőn, ezzel elérve az applikáció végleges célját.


9. ábra: Az adatbázis adatainak megjelenítése

Az alkalmazás az angol mellett támogatja a magyar nyelvet is. A készülék lokalizációjától függően a szövegek magyar területi beállítások mellett magyarul, minden egyéb mellett angolul jelennek meg.

## VIII. TESZTELÉS

Abból kiindulva, hogy nincs tökéletes rendszer, a projektet számos tesztelési folyamatnak vetettem alá. Mind a hardver mind a szoftver elemek különféle metódusok által lettek megvizsgálva, a rendszer biztonságos működéséről való meggyőződés érdekében. A Raspberry Pi statikus IP címet kapott. A jelenlegi architektúra csak saját hálózaton belül működik, viszont a routerek beállításánál lehetőség van az úgynevezett port forwarding opcióra, amivel bárhonnan rá tudunk csatlakozni a rendszerre, és ellenőrizni azt, hogy ki és mikor járt az adott helyiségben. Elsőként meg kellett győződnöm a felhasznált eszközök minőségéről és sértetlenségéről, mivel egy elektromos zárlat az áramkörben akár tűzveszélyessé is teheti a rendszert. Továbbá az Autodesk Circuits [7] nevű szimulációs környezetben ellenőriztem, hogy az általam kiválasztott ellenállások és LED-ek megfelelőek-e az együttes használatra. A tápfeszültség forrása is kérdéses volt az Arduino UNO esetében, ezt a problémát a Raspberry Pi egyik USB kimenete oldotta meg, mivel a soros porton való összeköttetéshez egyébként is szükséges volt ez a kapcsolat.

A rendszert teszteltem 80MBit-es hálózat használatával, valamint csökkentett bitrátával is, gyenge hálózati kapcsolat mellett is késedelem nélkül működik. Az adatbázisséma bővíthető ugyan, de projektemben csak a működéshez szükséges mezők kerültek bele a teljesítmény maximalizálásának érdekében [10].

Az Android applikáció fejlesztése közben, illetve a befejezése után szintén tesztelésre került emulátoron és fizikai eszközökön egyaránt. Emulátoron a legújabb, Nougat rendszert futtató Nexus 6-on végeztem tesztelést. Fizikai eszközök közül saját készülékemen, egy 5.1-es rendszerű Sony Xperia M2-n, illetve tableten is teszteltem. Mindegyik készüléken akadásmentes, és hiba nélküli volt a futás.

Összességében elmondható, hogy a rendszerben a végleges verzió kipróbálása során nem mutatkozott semmiféle nem várt viselkedés.

## IX. ÖSSZEGZÉS

Az RFID technológiát használó beléptető rendszer megépítésre került, a tesztelési folyamaton sikeresen átment. A nyílt forráskódú, Linux alapú eszközök használatának, valamint a hálózati megoldásoknak köszönhetően az IoT és az Industry 4.0 főbb alapelveinek eleget tesz [8]. A rendszer szükségtelenné válása esetén annak elemei újra programozhatók, illetve újrahasznosíthatóak, így költséghatékony megoldásnak bizonyult. Előnye a gyártók által létrehozott hasonló célt szolgáló architektúrákkal szemben az, hogy teljes körű irányítást birtokolhatunk a rendszer tervezőjeként, valamint a változtatásokat is könnyebb rajta végrehajtani, mint társainál.

## KÖSZÖNETNYILVÁNÍTÁS

A hardver eszközök rendelkezésemre bocsátásáért szeretnék köszönetet mondani Tallódi Lászlónak, a Debreceni Egyetem Informatikai Karának Mérnökinformatikus BSc. szakos hallgatójának, valamit Fejes Ferencnek, a Debreceni Egyetem Informatikai Karának Mérnökinformatikus MSc. szakos hallgatójának. Továbbá szeretném köszönetem kifejezni a tervezés folyamata során adott hasznos tanácsaiért és észrevételeiért Erdei Timotei Istvánnak, a Debreceni Egyetem oktatójának.

A publikáció elkészítését az EFOP-3.6.1-16-2016-00022 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

## HIVATKOZÁSOK

[1] Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim A.T. Hahsem, Faiz Alotaibi, „Internet of Things Security: A Survey,” Journal of Network and Computer Applications, June 2017; 88:10–28

[2] Adafruit, (2017, May 14). [Online]. Available: https://www.adafruit.com/product/155

[3] Arduino, (2017, May 14). [Online]. Available: https://www.arduino.cc/en/main/arduinoBoardUno

[4] Sander van Vugt, „Setting Up a LAMP Server,” The Definitive Guide to SUSE Linux Enterprise Server 12, pp.309-329

[5] Sanjib Sinha, „Python Environment,” Beginning Ethical Hacking with Python, pp.39-41

[6] Murat Yener, Onur Dundar, „Android Application Development With Android Studio,” Expert Android® Studio, pp.45-79

[7] Autodesk Circuits, (2017, May 14). [Online]. Available: https://circuits.io/

[8] Amy J. C. Trappey, Charles Trappey, Usharani Hareesh Govindarajan, John J. Sun, „A review of essential standards and patent landscapes for the Internet of Things: A key enabler for Industry 4.0,” Advanced Engineering Informatics, December 2016

[9] Satcounter (2017, May 14). [Online]. Available: https://starcounter.com/

[10] T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi, „A Novel Design of an Augmented Reality Based Navigation System & its Industrial Applications," 15th IMEKO TC10 – Technical Diagnostics in Cyber-Physical Era Budapest, Hungary, 6 – 7 June, 2017 - Organised by: MTA SZTAKI – Hungarian Academy of Sciences - Institute for Computer Science and Control.