

DEBRECENI EGYETEM • INFORMATIKAI KAR

# Házbiztonsági óvintézkedések megvalósítása Arduino Mega-val

SZAKDOLGOZAT

TÉMAVEZETŐ:

**Dr. Kocsis Gergely**

adjunktus

KÉSZÍTETTE:

**Dán Balázs**

Gazdaságinformatika (BSc) szakos hallgató

Debrecen

2021

# Tartalomjegyzék

|  |    |
|--|----|
| 1. Bevezető.....                                 | 1  |
| 2. Hardver.....                                  | 3  |
| 2.1. Arduino Mega 2560 R3 .....                  | 3  |
| 2.2. ESP32-CAM AI-Thinker .....                  | 5  |
| 2.3. Arduino Ethernet Shield R3 .....            | 7  |
| 2.4. HC-SR501 PIR .....                          | 9  |
| 3. Előkészületek .....                           | 12 |
| 3.1. Arduino IDE Software.....                   | 12 |
| 3.2. PHP.....                                    | 13 |
| 3.3. XAMPP .....                                 | 15 |
| 3.4. Fritzing .....                              | 17 |
| 3.5. React Native .....                          | 18 |
| 3.5.1. A React Native beüzemeltése .....         | 19 |
| 4. Hardverek összeállítása.....                  | 21 |
| 4.1. RFID-s ajtózárszerkezet összeállítása ..... | 22 |
| 4.2. A kamerarendszer összeállítása .....        | 24 |
| 5. Összeállított hardverek programozása .....    | 25 |
| 5.1. Az adatbázisrendszer megalkotása .....      | 25 |
| 5.2. A kártyaolvasó-rendszer programozása .....  | 29 |
| 5.3. A kamerarendszer programozása .....         | 32 |
| 6. Az applikáció fejlesztése.....                | 38 |
| 7. Utolsó módosítások.....                       | 47 |
| 7.1. Domain és ngrok .....                       | 47 |
| 7.2. A kész projekt elérhetővé tétele .....      | 48 |
| 8. Összefoglalás.....                            | 49 |
| 9. Irodalomjegyzék.....                          | 50 |

# 1. Bevezető

Mindannyian igyekszünk a lakhelyünket minél biztonságosabbá tenni különböző házvédelmi eszközök használatával. Előfordulhat, hogy amikor valamilyen okból kifolyólag el kell hagynunk a lakásunkat, legyen az munka, vagy iskola, nem feltétlenül bizonyosodtunk meg arról, hogy megfelelően zártuk be lakásunk ajtaját. Ez bármelyikünkkel megtörténhet, és általában erre csak akkor eszmélünk rá, amikor már elhagytuk lakásunkat. Ha esetleg már a munkahelyünkön tartózkodunk, akkor nem feltétlen tudunk egy bizonyos ideig hazamenni azért, hogy meggyőződjünk arról, hogy ténylegesen bezártuk-e lakásunk ajtaját. Olyan eset is könnyedén előfordulhat, hogy egy családtagunk elfelejt házkulcsot vinni magával és be szeretne menni a lakásba. Ebben az esetben két lehetősége van: vagy várakozik, vagy elmegy valaki olyanhoz, akinek van a lakáshoz kulcsa.

Általában mindig én hagytam el a lakást utoljára, és ha nem tudtam biztosra, hogy jól bezártam-e a lakást, mindaddig stresszes voltam, míg a családból valamelyikünk haza nem ment és meg nem győződött az ajtó állapotáról. Ilyenkor mindig eltöprengtem azon, hogy bár lenne egy könnyű, kényelmes és stresszmentes módja ennek ellenőrzésére.

Az utóbbi időben, szabadidőmben, mikor különböző platformokon videókat nézegettem, alkalmanként olyan reklámokat dobott fel például a Youtube, amely házvédelemmel volt kapcsolatos. Ilyen volt a SimpliSafe is. Rengeteg csatornát szponzorálnak, akik bemutatják a termékeiket. Felkeltette az érdeklődésemet, és elgondolkodtam a beruházás lehetőségén. Sok mindennel szolgál a SimpliSafe, legyen az mozgásérzékelő, vezeték nélküli kültéri kamera, beltéri kamera, nyitásérzékelő, üvegtörésérzékelő, és még sok más, amik könnyen üzembe helyezhetőek. Lehetősége van a vásárlóknak egy személyre szabott csomag összeállítására, attól függően, hogy mekkora méretű az illető lakása és hogy milyen védelemmel szeretné ellátni. Egy hátulütője van, ha egy komolyabb rendszert szeretnénk összeállítani, az kissé költséges.

Mindenképp szerettem volna valamilyen biztonsági rendszerrel ellátni a lakásunkat, és ekkor határoztam el, hogy létre szeretnék hozni egy rendszert, amely képes ellátni az alapvető házbiztonsági óvintézkedéseket. Ehhez először is ki kellett találnom, hogy miként szeretném ezt megvalósítani. Némi kutatás eredményeként jutottam arra, hogy Arduino-val fogom ezt megvalósítani, és azon belül is kártyabeléptető rendszerrel.

Minden egyes lakó rendelkezni fog egy RFID kártyával, amely igazolja majd a személy validitását, és annak megfelelően nyitja ki a személy számára az ajtót. Önmagában a hagyományos kulccsal nyitható zárszerkezet lecserélése egy RFID kártyabeléptető rendszerrel praktikus és kényelmes, de nem küszöböljük ki azt a lehetőséget, hogy a lakó abban az esetben is be tudjon lépni a lakásba, ha netán elhagyja, avagy otthon felejtí az RFID kártyáját. Erre való tekintettel döntöttem el, hogy egy Android alkalmazást szeretnék létrehozni, amely tartózkodási helytől függetlenül képes lenne az ajtózárszerkezet kinyitására a felhasználó validitásától függően. Az alkalmazással az összes lakó rendelkezni fog, amelyre beregisztrálva hozzáadhat magának egy RFID azonosítót, majd a későbbiekben az admin (a rendszer kezelője) módosíthatja a felhasználók RFID-ját, amelyekkel nyithatja a szerkezetet.

Mindezek mellett fontosnak tartottam, hogy egy kamerát is integráljak a rendszerbe. Kamerának az ESP32-CAM kamerát választottam, mert igazán pénztárcakímélő, és programozható.

Összességében egy olyan Android alkalmazást szerettem volna megvalósítani, ami egy vagy több Arduino eszközt használva eleget tesz a következő feltételeknek valid felhasználótól függően:

- van felhasználói felülete
- alkalmas az ajtózárszerkezet távoli vezérlésére
- tárolja a felhasználókat és a hozzájuk tartozó RFID-kat, illetve hogy melyik RFID melyik Arduino-t képes nyitni
- képes állókép készítésére, amely később megtekinthető
- betekintheünk a kamera által készített aktuális képsorozatba
- log-ot készít az eseményekről (ajtónyitás, zárás), amely az admin számára megtekinthető
- a felhasználói felület okostelefonról elérhető és megtekinthető
- a felhasznált hardver elemek költsége legyen elfogadható

## 2. Hardver

A célok eléréséhez beszerzetem egy Arduino Mega 2560 vezérlőpanelt, egy Ethernet Shield R3-at, egy ESP32-CAM kamerát, egy FT232RL FTDI programmert és egy RFID kártyát, illetve egy HC-SR501 PIR mozgásérzékelő modult, amelyeket felhasználok projektem megalkotásához.

### 2.1. Arduino Mega 2560

Az Arduino Mega 2560 egy mikrovezérlő kártya, amely ATmega2560 alapú (1. ábra). 54 digitális bemeneti/kimeneti tűvel (ebből 15 használható PWM kimenetként), 16 analóg bemenettel, 4 UART-tal (hardver soros port), 16 MHz-es kristályoszillátorral, USB-csatlakozóval, tápcsatlakozóval, ICSP fejléccel, és egy reset gombbal rendelkezik. A mikrokontroller támogatásához minden szükséges dolgot tartalmaz a Board, egyszerűen csak csatlakoztatni kell egy számítógéphez egy USB kábellel, vagy árammal kell ellátni egy AC-DC adapterrel vagy akkumulátorral. A Mega 2560 kártya kompatibilis a legtöbb pajzzsal, amelyet az Uno és a korábbi Duemilanove vagy Diecimila táblákhoz terveztek.

Az Arduino Mega 2560 nyílt forráskódú hardver. A Mega 2560 az Arduino Mega frissítése, amelyet lecserél [1].



1.ábra: Arduino Mega 2560 R3 mikrovezérlő kártya [1]

Az Arduino Mega 2560 mikrovezérlő kártya általános jellemzői [1]:

- Üzemi feszültség: 5V
- Input feszültség (ajánlott): 7-12V
- Input feszültség (limit): 6-20V
- Digitális I/O tűk száma: 54 (ebből 15 használható PWM kimenetként)
- Analóg input pin-ek száma: 16
- SRAM: 8KB
- EEPROM: 4KB
- Beépített LED: 13-as pin
- Flash Memória: 256KB amelyből 8KB-ot a bootloader használ

A Mega 2560 board az Arduino szoftverrel (IDE) programozható. A Mega 2560 ATmega2560-ja előre programozott bootloaderrel rendelkezik, amely lehetővé teszi új kód feltöltését egy külső hardverprogramozó használata nélkül. Az eredeti STK500 protokoll (kommunikáció, C fejléc fájlok) használatával kommunikál. A bootloader is megkerülhető, és a mikrokontrollert az ICSP (In-Circuit Serial Programming) fejléccen keresztül programozható Arduino ISP vagy hasonló használatával.

A Mega 54 digitális pinjének mindegyike használható bemenetként vagy kimenetként a `pinMode()`, a `DigitalWrite()` és a `DigitalRead()` funkciók használatával. 5 voltól működnek. Minden tű 20 mA-t képes biztosítani vagy fogadni az ajánlott működési feltételek mellett, és belső felhúzó ellenállása (alapértelmezés szerint lekapcsolva) 20-50 kiloohm. Legfeljebb 40 mA az az érték, amelyet nem szabad túllépni a mikrokontroller maradandó károsodásának elkerülése érdekében.

Ezenkívül néhány csap speciális funkciókkal rendelkezik:

- RX és TX: TTL soros adatok fogadására (RX) és továbbítására (TX) használható. A 0-ás és 1-es érintkezők szintén az ATmega16U2 USB-TTL soros chip megfelelő csapjaihoz vannak csatlakoztatva. Ilyen Pin-ek például: 0 (RX) és 1 (TX); 1. Sorozat: 19 (RX) és 18 (TX); 2. sorozat: 17 (RX) és 16 (TX); 3. Sorozat: 15 (RX) és 14 (TX).
- PWM: 2-13 és 44-46. 8 bites PWM kimenetet az `analogWrite()` funkcióval.

- LED: 13. Beépített LED van csatlakoztatva a 13-as digitális Pin-hez. Ha a pin HIGH értékű, a LED világít, ha a pin LOW, akkor nem világít.
- TWI: 20 (SDA) és 21 (SCL). Támogatja a TWI kommunikációt a Wire könyvtár használatával. Ezek a csapok nem ugyanazon a helyen vannak, mint a TWI csapok a régi Duemilanove vagy Diecimila Arduino táblákon.

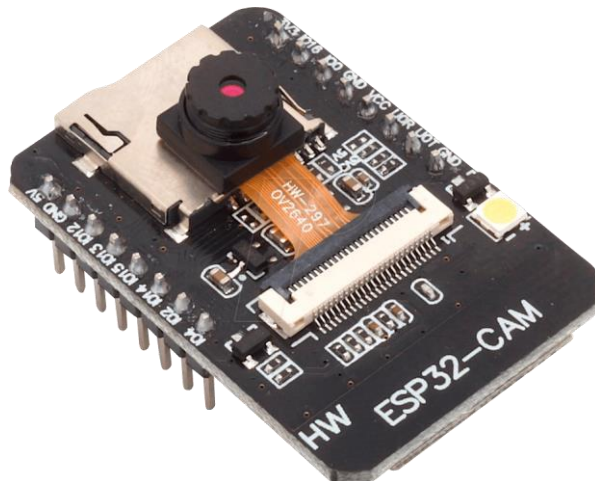
A soros perifériás interfész (SPI) egy szinkron soros adatprotokoll, amelyet a mikrokontrollerek használnak egy vagy több perifériás eszközzel rövid távú kommunikációhoz [16]. Két mikrokontroller közötti kommunikációra is használható. SPI kapcsolat esetén mindig van egy fő eszköz (általában mikrokontroller), amely a perifériás eszközöket vezérli. Általában három sor van, ami közös az összes eszközön:

- MISO (Master In Slave Out) - A Slave vonal az adatok elküldéséhez a Masterhez,
- MOSI (Master Out Slave In) - A Master vonal az adatok perifériákra küldésére,
- SCK (Serial Clock) - Az óra impulzusok, amelyek szinkronizálják a Master által generált adatátvitelt
- SS (Slave Select) - minden eszköz azon csapja, amelyet a Master használhat bizonyos eszközök engedélyezéséhez és letiltásához.

Az SS egy sor, amely minden különböző típusú eszközön eltérő. Ha egy eszköz Slave Select csapja alacsony, kommunikál a masterrel. Ha magas, figyelmen kívül hagyja a mestert. Ez lehetővé teszi több SPI-eszköz megosztását ugyanazon MISO, MOSI és CLK vonalakon.

## **2.2. ESP32-CAM AI-Thinker**

Az ESP32-CAM egy funkciók teljes skálájával rendelkező mikrokontroller, amely integrált videokamerával és microSD-kártya-aljzattal is rendelkezik (2. ábra). Olcsó és könnyen használható, és tökéletes olyan IoT-eszközökhöz, amelyek fejlett funkciókkal, például képkövetéssel és -felismeréssel rendelkező kamerát igényelnek [2].



2. ábra: ESP32-CAM AI-Thinker [3]

Az alaplapot az Espressif ESP32-S SoC, egy nagy teljesítményű, programozható MCU hajtja, WIFI-vel és Bluetooth-szal. Ez a legolcsóbb (körülbelül 7 dolláros) ESP32 fejlesztői kártya, amely egyszerre kínál beépített kameramodult, MicroSD kártya támogatást és 4 MB PSRAM-ot. A jelerősítéshez külső wifi antenna hozzáadása extra forrasztási munkát igényel. Az alaplapon nincs hagyományos USB port, vagy FTDI programozót, vagy kiegészítő HAT-ot, vagy Arduino UNO-t és Arduino IDE/ESP-IDF DEV eszközöket kell használni a kódok feltöltéséhez [4].

Az ESP-32 CAM két bemeneti tápcsatlakozóval rendelkezik, és alapértelmezetten a Board 3,3 V-ot vesz fel. Előfordulhat, hogy ha az OV2640-et 3,3 V alatt használják, a Board instabillá válhat (hibák vagy színes vonalak). 5V bemenet javasolt.

Mivel alacsony költségű tábla, elég kis méretű, így rendkívül népszerűvé vált számos IoT és gépi látási alkalmazásban. Bár az ESP32-CAM nagy lehetőségeket rejt magában az IoT-vel kapcsolatos projektekben, nem tökéletes tábla, különösen a kamera részen, a hozzá tartozó 2 MP-es OV2640 kamerának vannak bizonyos korlátai.

Az ESP32-CAM általános tulajdonságai [5]:

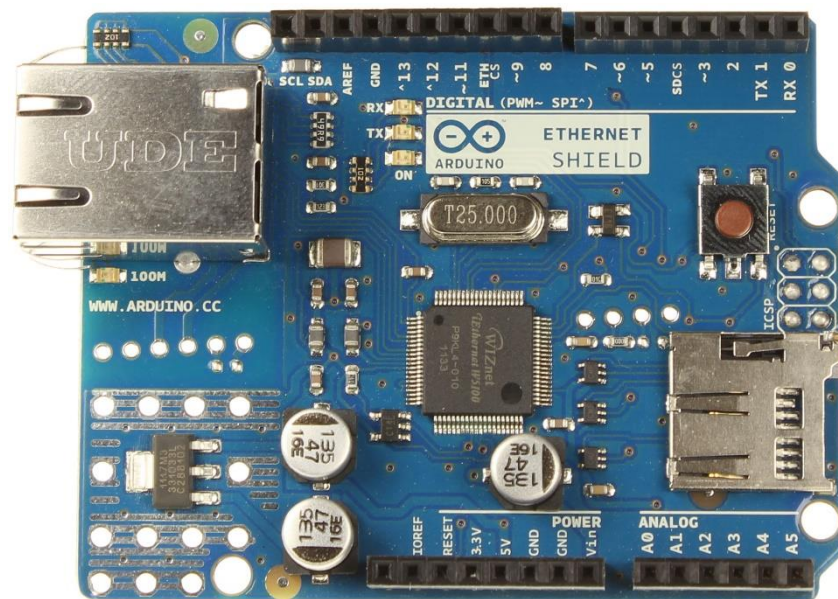
- 10 GPIO látható
- A legkisebb 802.11b/g/n Wi-Fi BT SoC modul (bővített Wi-Fi lefedettséget)
- Kis teljesítményű 32 bites CPU, az alkalmazásprocesszort is kiszolgálja
- Akár 160 MHz -es órajel, összefoglaló számítási teljesítmény akár 600 DMIPS
- Beépített 520 KB SRAM, külső 4MPSRAM
- Támogatja az UART/SPI/I2C/PWM/ADC/DAC -t



- Támogatja az OV2640 és OV7670 kamerákat
- Beépített zseblámpa
- Támogatja a kép WiFi feltöltését
- TF kártya támogatása
- Több alvási módot támogat
- Beágyazott Lwip és FreeRTOS
- Támogatja az STA/AP/STA+AP üzemmódot
- Támogatja a Smart Config/AirKiss technológiát
- Támogatja a soros port helyi és távoli firmware frissítéseit (FOTA)

## 2.3. Arduino Ethernet Shield R3

Az Arduino Ethernet Shield R3 lehetővé teszi, hogy az Arduino eszköz csatlakozzon az internethez [6][6]. A Wiznet W5100 ethernet chipen (adatlap) alapul. A Wiznet W5100 TCP és UDP protokollokra egyaránt alkalmas hálózati (IP) köteget biztosít. Legfeljebb négy egyidejű aljzatcsatlakozást támogat. Az Ethernet pajzs tartalmaz egy micro SD kártya csatlakozót és egy fedélzeti visszaállítási vezérlőt (Reset) (3. ábra). Kompatibilis az összes Arduino/Genuino táblával. A fedélzeti micro SD kártyaolvasó az Arduino IDE SD könyvtárán keresztül érhető el. Ekkor az SS a 4-es érintkezőn van. A jelenlegi pajzs Power Over Ethernet (PoE) modullal rendelkezik.



3.ábra: Arduino Ethernet Shield R3 [6]

Az Ethernet pajzs az Arduino board-hoz csatlakozik a huzalon átívelő hosszú huzalcsomagoló fejek segítségével. Ez megőrzi a csapok elrendezését, és lehetővé teszi egy másik pajzs felhalmozását a tetejére. Az Arduino az SPI busz használatával (az ICSP fejlécen keresztül) kommunikál mind a W5100, mind az SD kártyával. Ez az Uno 10-es, 11-es, 12-es és 13-as digitális csapjain, a Mega 50-es, 51-es és 52-es csapjain található. Mindkét táblán a 10. tűskével választható ki a W5100, és a 4. tűvel az SD -kártyához. Ezek a csapok nem használhatók általános I/O-hoz.

A Mega-n az 53-as hardveres SS-tű nem használható sem a W5100, sem az SD kártya kiválasztására, de azt kimenetként kell tartani, különben az SPI interfész nem fog működni. A pajzs szabványos RJ45 ethernet aljzatot biztosít.

Mivel a W5100 és az SD kártya megosztja az SPI buszt, egyszerre csak egy lehet aktív. Ha mindkét perifériát használja a program, erről a megfelelő könyvtáraknak kell gondoskodniuk. Ha azonban nem a program egyik perifériáját sem használja, akkor kifejezetten törölni kell a kijelölést. Az SD-kártya esetén a 4-es tűt be kell állítani kimenetként, magas értékkel. A W5100 esetében állítsa a 10-es digitális tűt magas értékű kimenetre kell állítani.

Az Arduino Ethernet Shield R3 általános jellemzői:

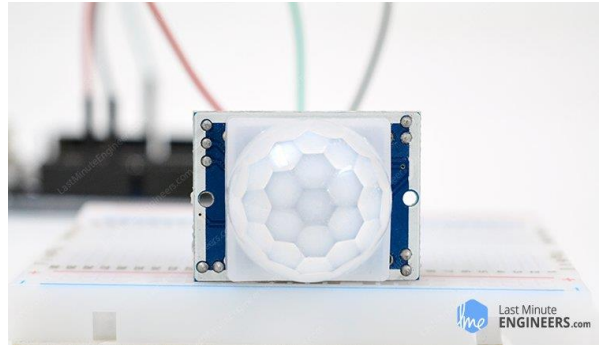
- Arduino táblát igényel (nem tartozék)
- Üzemi feszültség 5V (az Arduino Boardról)
- Ethernet vezérlő: W5100 belső 16K pufferrel
- Csatlakozási sebesség: 10/100Mb
- Kapcsolat az Arduino-val az SPI porton

A pajzs számos tájékoztató LED -et tartalmaz:

- PWR: azt jelzi, hogy a tábla és a pajzs áram alatt van
- LINK: jelzi a hálózati kapcsolat jelenlétét, és villog, amikor a pajzs adatokat továbbít vagy fogad
- FULLD: azt jelzi, hogy a hálózati kapcsolat full duplex
- 100 M: 100 Mb/s hálózati kapcsolat jelenlétét jelzi (szemben a 10 Mb/s -mal)
- RX: villog, ha a pajzs adatokat fogad
- TX: villog, ha a pajzs adatokat küld
- COLL: villog, ha hálózati ütközést észlel

## 2.4. HC-SR501 PIR

A HC-SR501 PIR mozgásérzékelő modul lehetővé teszi a mozgás érzékelését (4. ábra). Szinte mindig arra használják, hogy érzékeljék az emberi test mozgását az érzékelő hatótávolságán belül.



4.ábra: HC-SR501 PIR mozgásérzékelő [7]

A PIR érzékelőt kifejezetten az infravörös sugárzás érzékelésére tervezték. Alapvetően két fő részből áll: egy piroelektromos érzékelőből és egy speciális, Fresnel lencséből, amely az infravörös jeleket a piroelektromos érzékelőre fókuszálja [7].

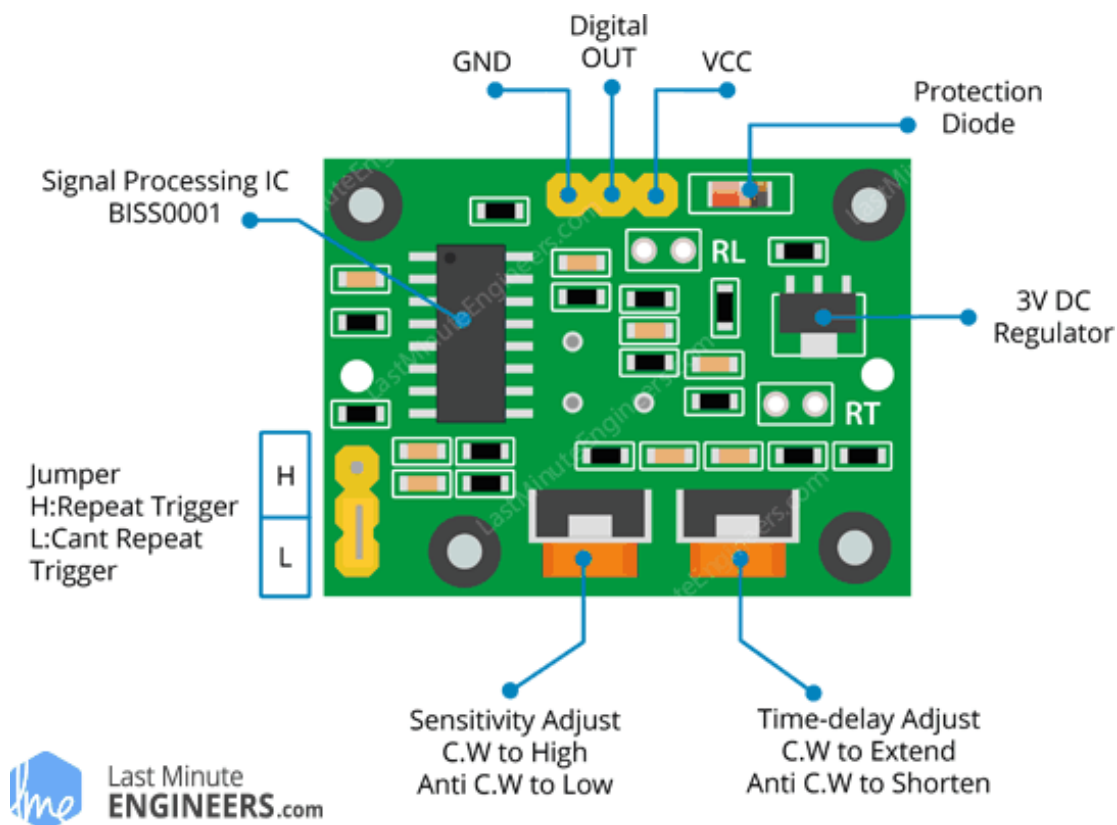
Ezek a mozgásérzékelők alacsony fogyasztásúak és olcsók, meglehetősen strapabíróak, széles lencse-választékkal rendelkeznek, könnyen kezelhetők és örülten népszerűek a hobbisták körében. A HC-SR501 PIR érzékelőnek három kimeneti csapja van: VCC, kimenet és földelés.

- A VCC a HC-SR501 PIR érzékelő tápegysége, amelyhez csatlakoztatjuk az 5 V-os csatlakozót az Arduino-n.
- A kimeneti tű egy 3.3V -os TTL logikai kimenet. A LOW azt jelzi, hogy nincs mozgás, a HIGH azt jelenti, hogy bizonyos mozgást észlelt.
- A GND -t csatlakoztatni kell az Arduino földjéhez.

Beépített feszültségszabályzóval rendelkezik, így bármilyen egyenáramú, 4,5 és 12 volt közötti feszültségről táplálható, de általában 5 V-ot használ. A modul beépített piroelektromos érzékelővel, kondicionáló áramkörrel és kupola alakú Fresnel lencsével rendelkezik.

A board-on két potenciométer található néhány paraméter beállításához (5. ábra):

- **Érzékenység** - Ez határozza meg a mozgás maximális észlelhető távolságát. 3 métertől körülbelül 7 méterig terjed. A szoba topológiája befolyásolhatja az elért tényleges hatótávolságot.
- **Idő** - Ez határozza meg, hogy mennyi ideig maradjon a kimenet MAGAS az észlelés után. Minimum 3 másodperc, maximum 300 másodperc vagy 5 perc.



5.ábra: PIR mozgásérzékelő szenzor [7]

Végül a táblán van egy jumper (egyres modelleknel a jumper nincs beforrasztva). Két beállítása van:

- **H**– Ez a Hold/Repeat/Retriggering. Ebben a helyzetben a HC-SR501 továbbra is HIGH jelet bocsát ki mindaddig, amíg a mozgást érzékeli.
- **L**– Ez az Intermittent (szakaszos) vagy No-Repeat/Non-Retriggering. Ebben a helyzetben a kimenet MAGAS marad a TIME potenciométer beállításával beállított ideig.

A PIR alapú alkalmazások tervezése előtt megfontolandó dolgok:

- A legtöbb PIR érzékelőhöz hasonlóan a HC-SR501-nek is időre van szüksége ahhoz, hogy akklimatizálódjon a helyiség infravörös energiájához. Ez 30-60 másodpercig tart az érzékelő első bekapcsolásakor.
- Ezenkívül az érzékelő körülbelül 5 vagy 6 másodperces „visszaállítási” periódussal rendelkezik a leolvasás után. Ez idő alatt nem érzékel semmilyen mozgást.
- A HC-SR501-en alapuló rendszer tervezésekor figyelembe kell vennie ezeket a késleltetési időszakokat.

## 3. Előkészületek

A hardverek összeállítását a szükséges program(ok) telepítésével kezdtem. Az Arduino IDE-t a hardverek leprogramozásához, míg az applikáció fejlesztéséhez a React Native-t használtam. A backend-et a PHP szkriptnyelven írtam, amelyet az Atom nevű szoftverben fejlesztettem. Az XAMPP szoftverrel létrehoztam egy helyi kiszolgálót (localhost), amelyen keresztül a phpMyAdmin-t használva dolgoztam ki adatbázisrendszeremet. A szakdolgozatom egyes kapcsolási rajzait a Fritzing nevű alkalmazással készítettem el.

### 3.1. Arduino IDE Software

Az Arduino Integrált Fejlesztési Környezet - vagy Arduino Software (IDE) - tartalmaz egy szövegszerkesztőt a kód írásához, egy üzenetterületet, egy szöveges konzolt, egy eszköztárat gombokkal a közös funkciókhoz és menük sorozatát [8]. A nyílt forráskódú Arduino szoftver (IDE) megkönnyíti a kód írását és a táblára való feltöltését. Ez a szoftver bármilyen Arduino táblával használható. Csatlakozik az Arduino hardverhez, hogy programokat töltsön fel és kommunikáljon velük. Magát a szoftvert könnyedén, pár kattintással az Arduino honlapjáról letölthetjük ingyenesen. Szakdolgozatom írásának idején az Arduino IDE 1.8.13-as verziót használtam (6. ábra).



6.ábra: Arduino IDE kezelőfelület

Az Arduino Software (IDE) segítségével írt programokat sketch-eknek – vázlatoknak – nevezzük. Ezek a vázlatok a szövegszerkesztőben íródnak, és az .ino fájlkiterjesztéssel kerülnek mentésre. A szerkesztő funkciókkal rendelkezik a vágáshoz/beillesztéshez és a szöveg kereséséhez/cseréjéhez. Az üzenetterület visszajelzést ad mentés és exportálás közben, valamint hibákat is megjelenít. A konzol megjeleníti az Arduino Software (IDE) által kiadott szöveget, beleértve a teljes hibaüzeneteket és egyéb információkat. Az ablak jobb alsó sarkában megjelenik a konfigurált kártya és soros port. Az eszköztár gombjai lehetővé teszik a programok ellenőrzését és feltöltését, vázlatok létrehozását, megnyitását és mentését, valamint a soros monitor megnyitását.

## 3.2. PHP

A PHP (rekurzív rövidítése a PHP-nek: Hypertext Preprocessor) egy széles körben használt, nyílt forráskódú, általános célú szkriptnyelv [9]. Különösen alkalmas webfejlesztésre, és beágyazható HTML-be (7. ábra).



```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>

    <?php
      echo "Hi, I'm a PHP script!";
    ?>

  </body>
</html>
```

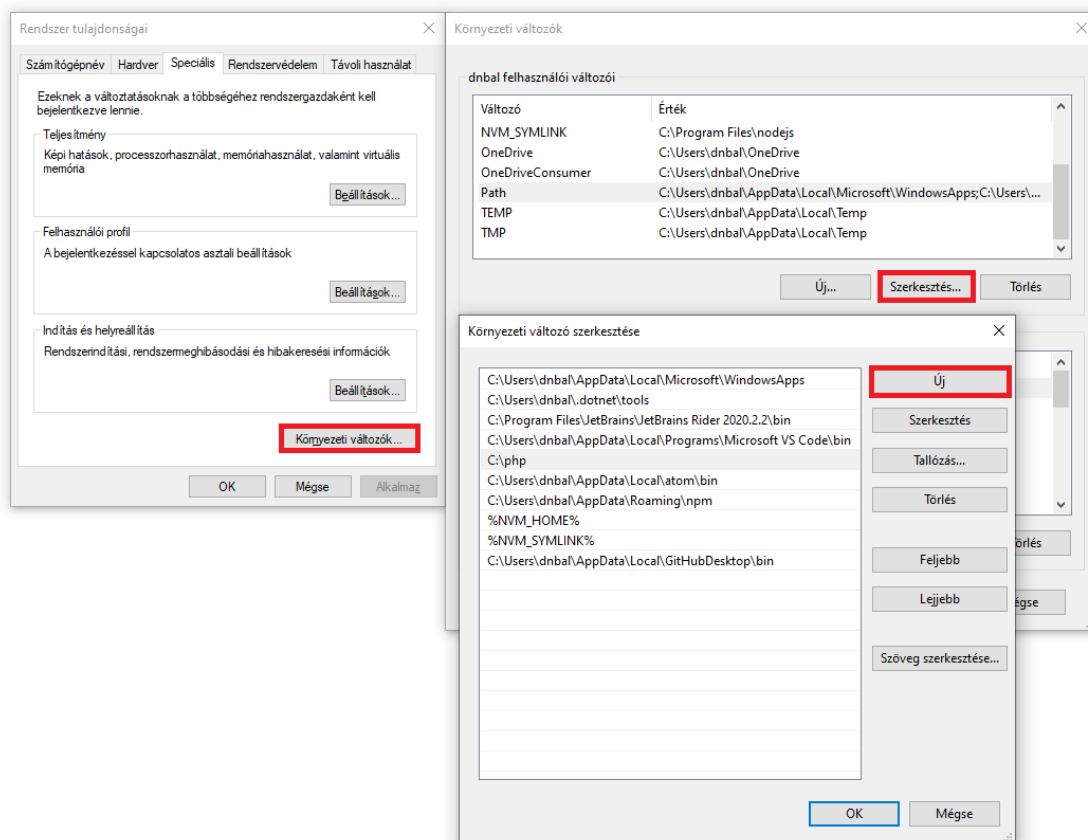
7.ábra: PHP példa [9]

A sok HTML parancs kiadására vonatkozó parancs helyett (amint az a C-ben vagy a Perl-ben is látható), a PHP oldalak olyan beágyazott kódú HTML-t tartalmaznak, amely „valamit” tesz (ebben az esetben a „Hi, I’m a PHP script!” Kimenet). A PHP kódot speciális kezdő és befejező feldolgozási utasítások <?PHP és?> tartalmazzák, amelyek lehetővé teszik a „PHP mód” ki- és bekapcsolását.

Az különbözteti meg a PHP-t a kliens oldali JavaScript-től, hogy a kódot a szerveren hajtják végre, HTML-t generálva, amelyet ezután elküld az ügyfélnek. Az ügyfél megkapja a szkript futtatásának eredményeit, de nem tudja, mi az alapkód. A webszervert akár úgy is konfigurálhatja, hogy az összes HTML-fájlt PHP-vel dolgozza fel, és akkor tényleg nincs mód arra, hogy a felhasználók megmondják, mi volt a kód.

A PHP egyik legerősebb és legjelentősebb tulajdonsága az adatbázisok széles körének támogatása. Az adatbázis-alapú weblap írása hihetetlenül egyszerű az adatbázis-specifikus kiterjesztések egyikével (pl. Mysql esetén), vagy egy absztrakciós réteg, például a PDO használatával, vagy az ODBC kiterjesztésen keresztül csatlakozhat bármely adatbázishoz, amely támogatja az Open Database Connection szabványt. Más adatbázisok cURL -t vagy socketeket használhatnak, mint például a CouchDB.

Telepítéshez először is le kell tölteni a PHP-t a PHP főoldaláról. Az én operációs rendszerem 64-bites, ezért én a legújabb verziójú .zip fájlt töltöttem le. Ezt az egyszerűség kedvéért kicsomagoltam a C:\php mappába. Ezután módosítanom kellett a rendszer környezeti változóit az alábbi módon (8. ábra) [9]:



8.ábra: Környezeti változók módosítása



Erre azért volt szükség, hogy definiáljam a `root directory`-t a Windows számára, hogy melyik mappában helyezkedik el a PHP-m. Ellenőrzésképp, hogy jól adtuk meg a rendszerváltozót, megnyitva a terminált, lefuttattam az alábbi parancsot:

```
echo %PATH%
```

Ezzel kiíratam a „PATH” rendszerváltozó összes útvonalát, meggyőződve arról, hogy megfelelően hozzáadtam a php-t a PATH-hez.

Annak érdekében, hogy megfelelően működik a php-m, még egy parancsot lefuttattam a terminálban:

```
php -v
```

Mivel a terminál kiíratott egy verziószámot és arra vonatkozó egyéb információkat, megbizonyosodtam róla, hogy megfelelően telepítettem fel a PHP-t.

Szakdolgozatom írása alatt az összes PHP backend fájlt – későbbiekben a frontend HTTP kéréseket is- a Postman alkalmazással teszteltem le. A Postman egy API tesztelésre használt alkalmazás [10]. Ez egy HTTP kliens, amely grafikus felhasználói felület segítségével teszteli a HTTP kéréseket (Request), amelyen keresztül különböző típusú válaszokat kapunk. A Postman számos végpont-interakciós módszert kínál. Ezek közül a leggyakoribbak:

- GET: Információ beszerzése
- POST: Információ hozzáadása
- PUT: Cserélje ki az információkat
- PATCH: Frissítsen bizonyos információkat
- TÖRLÉS: Információ törlése

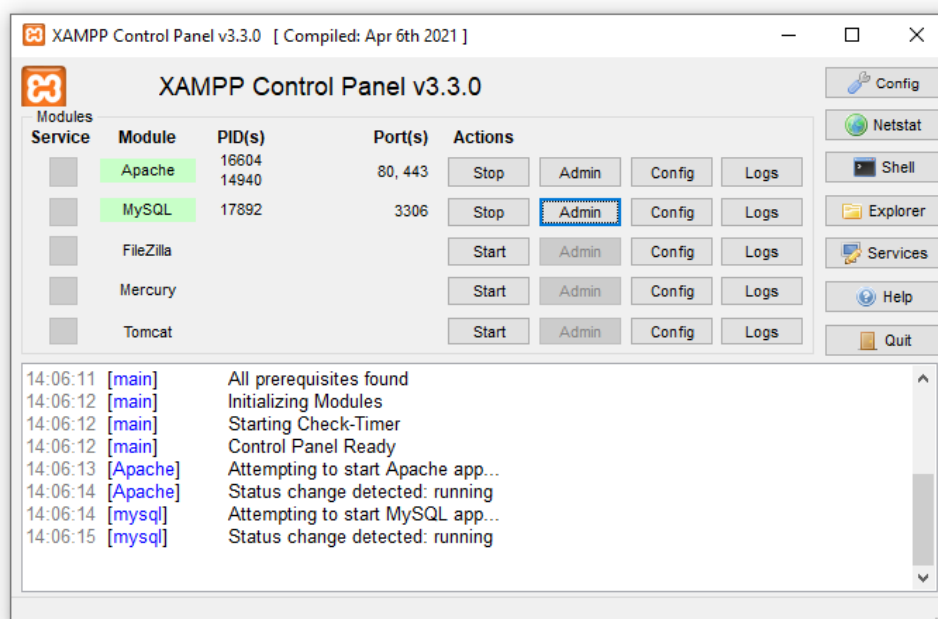
A Postman tesztelés előfeltétele volt számomra az XAMPP alkalmazás futtatása, hogy lokális hálózatot hozzak létre.

### 3.3. XAMPP

Az XAMPP egy széles körben használt platformok közötti webszerver, amely segít a fejlesztőknek programjaik helyi webszerveren történő létrehozásában és tesztelésében [11]. Az Apache Friends fejlesztette ki, és natív forráskódját bárki módosíthatja. Apache HTTP szerverből, MariaDB-ből és a különböző programozási nyelvek, például a PHP és

a Perl interpreter-ből áll. 11 nyelven érhető el, és különböző platformok támogatják, például a Windows IA-32 csomagja, valamint a macOS és Linux x64 csomagja.

Az XAMPP egy rövidítés, ahol az X jelentése Cross-Platform, A jelentése Apache, M jelentése MYSQL, a P-k pedig PHP és Perl. Ez egy nyílt forráskódú webes megoldáscsomag, amely tartalmazza az Apache disztribúciót számos kiszolgálóhoz és a parancssori futtatható fájlokat, valamint az Apache szerver, a MariaDB, a PHP és a Perl modulokat (9. ábra).



9.ábra: XAMPP vezérlőpanel

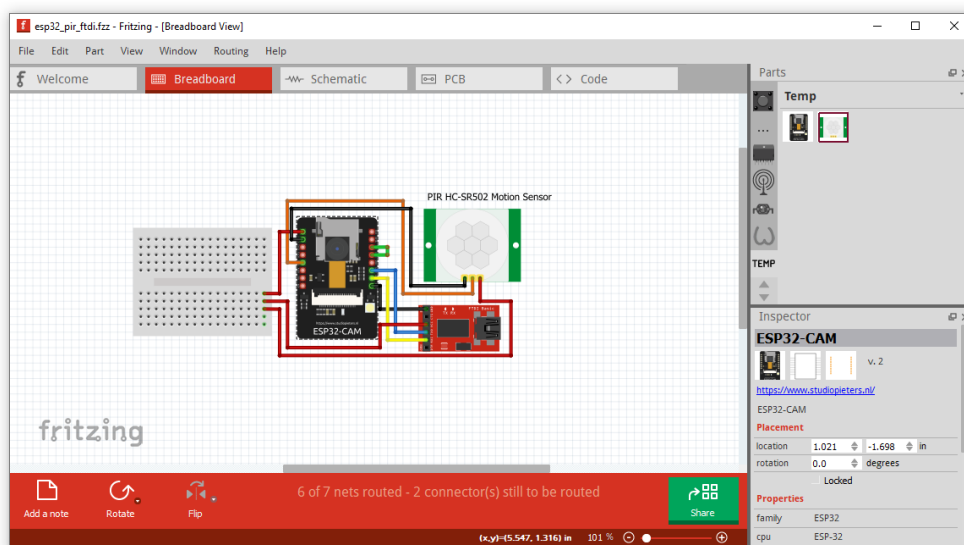
Az XAMPP segít egy helyi gazdagépnek vagy szervernek, hogy számítógépen és laptopon keresztül tesztelje webhelyét és ügyfeleit, mielőtt kiadná a fő szervernek. Ez egy olyan platform, amely megfelelő környezetet biztosít az Apache, a Perl, a MySQL adatbázis és a PHP alapú projektek működésének tesztelésére és ellenőrzésére a gazda rendszerén keresztül. Ezek közül a technológiák közül a Perl egy webfejlesztéshez használt programozási nyelv, a PHP egy backend szkriptnyelv, a MariaDB pedig a MySQL által kifejlesztett legélénkebben használt adatbázis. Az én rendszerem megvalósításához az Apache-t, illetve MySQL-t fogom használni az XAMPP vezérlőpaneléről.

Az Apache egy HTTP többplatformos webservert. Világszerte használják webtartalom szállítására. A szerveralkalmazás ingyenesen telepíthető, és az Apache

Software Foundation védelme alatt használható fejlesztői közösség számára. Az Apache távoli szervere szállítja a kért fájlokat, képeket és egyéb dokumentumokat a felhasználónak.

### 3.4. Fritzing

A Fritzing egy olyan szoftver, amely segít dokumentálni és megosztani az elektronikus prototípusprojekteket, tanítani az elektronikát, és professzionális nyomtatott áramköri lapokat (PCB) gyártani [12]. Valóságghűen ábrázolja az elektronikus alkatrészeket, és intuitív megközelítést alkalmaz, hogy a komplex technológiát nem technológiával foglalkozó szakemberek is felhasználhassák (alacsony belépési korlátot teremtve) Támogatja a készítőket (dizájnerek, művészek, hallgatók, barkácsolók) abban, hogy egy fizikai prototípus készítésétől egy tényleges termék felé haladjanak. Ideális eszköz a tanítási környezetben az elektronika megismeréséhez (10. ábra).



10. ábra: Fritzing felhasználói felület

Szakdolgozatom egyes kapcsolási rajzait ennek a programnak a segítségével készítettem el. Az olyan komponenseket, amelyek alapértelmezetten nem szerepeltek a Fritzing-ben, azt a Google keresőben a `site:fritzing.org „komponensnév”` keresési utasítással kerestem meg és használtam projektemben.

### 3.5. React Native

A React Native (RN) egy népszerű JavaScript-alapú mobilalkalmazás-keretrendszer, amely lehetővé teszi natív módon előállított mobilalkalmazások készítését iOS és Android rendszerre [13]. A keretrendszer lehetővé teszi, hogy ugyanazon kódbázis használatával alkalmazásokat hozzon létre különböző platformokhoz.

A React Native-t először a Facebook adta ki nyílt forráskódú projektként 2015-ben. Néhány év alatt a mobilfejlesztés egyik legnépszerűbb megoldásává vált. A React Native fejlesztést a világ vezető mobilalkalmazásainak, köztük az Instagramnak, a Facebooknak és a Skype-nak a működtetésére használják. A React Native globális sikerének több oka is van.

Először is, a React Native használatával a vállalatoknak elég csak egyszer létrehozniuk egy kódot, és azt felhasználhatják iOS- és Android-alkalmazásaik működtetésére. Ez óriási idő- és erőforrás-megtakarítást jelent.

Másodszor, a React Native a React-ra épült – egy JavaScript-könyvtárra, amely már a mobil keretrendszer kiadásakor is rendkívül népszerű volt.

Harmadszor, a keretrendszer felhatalmazta a frontend fejlesztőket, akik korábban csak webalapú technológiákkal dolgozhattak, hogy robusztus, gyártásra kész alkalmazásokat hozzanak létre mobil platformokra.

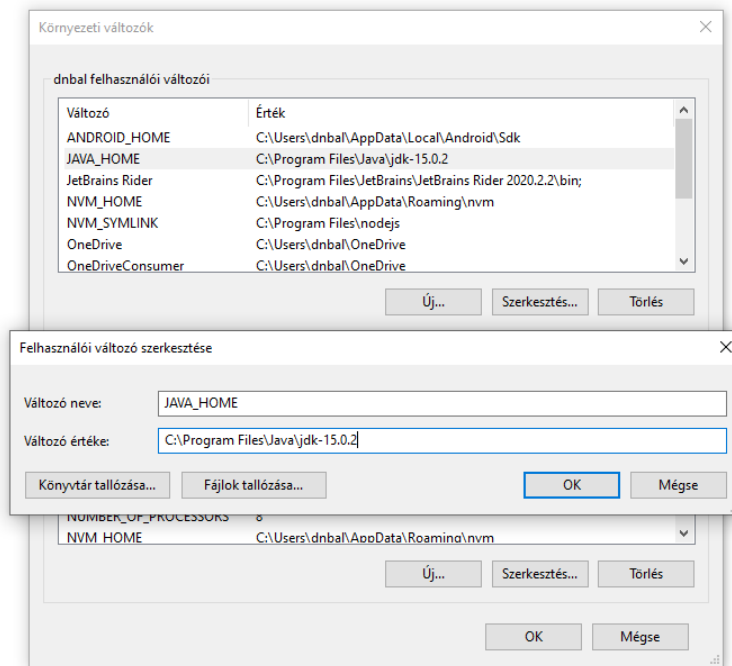
A React (más néven ReactJS) egy JavaScript-könyvtár, amelyet egy webhely frontendjének felépítésére használnak. A React Native-hez hasonlóan ezt is a Facebook mérnöki csapata fejlesztette ki. Eközben a React Native – amelyet a React hajt – lehetővé teszi a fejlesztők számára, hogy felhasználói felület-komponenseket használjanak iOS- és Android-alkalmazások gyors összeállításához és elindításához.

Mind a React, mind a React Native a JavaScript és egy speciális jelölőnyelv, a JSX keverékét használja. A JSX-összetevők elemeinek megjelenítéséhez használt szintaxis azonban eltér a React és a React Native esetében. Ezenkívül a React használ néhány HTML-t és CSS-t, míg a React Native lehetővé teszi a natív mobil felhasználói felület elemeinek használatát.

### 3.5.1. A React Native beüzemelése

A React Native-hez először is letöltöttem a Java JDK-t, a Node.js-t, Android Studio-t, és preferencia szerint a Visual Studio Code-ot, majd ezeket telepítettem. Mindenből az akkori legújabb verziót töltöttem és telepítettem fel, kivéve a JAVA JDK-ból, amiből a 15.0.2-es verziót telepítettem fel [14].

A JDK feltelepítése után módosítani kellett a környezeti változókat hasonlóképpen, mint ahogy a 3.2. PHP pontban a php mappa esetén tettem (11. ábra):



11. ábra: Környezeti változók szerkesztése

A Node.js feltelepítése után terminálban végrehajtottam a következő utasításokat:

`node -v`: hogy meggyőződjek a telepítés sikerességéről

`npm install -g react-native-cli`: globális változók csomagot telepít fel

Az Android Studio telepítését követően elindítottam az alkalmazást, hogy feltelepítsem az Android SDK-t. A `Configure > SDK Manager`, vagy a `Tools > SDK Manager` menüpontnál feltelepítettem az Android 10 (Q) SDK-t. Majd a `Tools > AVD Manager` menüpontnál létrehoztam egy új virtuális eszközt. Én a Pixel 5-öt választottam. Ezt követően hozzáadtam a környezeti változóimat. Hozzáadtam az ANDROID\_HOME környezeti változót a 11. ábrának megfelelően.

A szakdolgozati projektem létrehozásához a következő utasításokat hajtottam végre a Terminálban:

```
npx react-native init SzakdolgozatReactNative:
```

Mivel a Terminál megnyitását követően nem állítottam be új elérési útvonalat a „cd” paranccsal, így a „C:\Users\dnbal\” könyvtárban hozta létre a „SzakdolgozatReactNative” nevű React Native projektem főmappáját.

```
cd C:\Users\dnbal\AppData\Local\Android\Sdk\emulator:
```

Az emulátor definiálásához először is megadtam az emulátor főmappa elérési útvonalát.

```
emulator -list-avds:
```

 Ha van elérhető AVD, akkor kilistázza mindet.

```
emulator -avd Pixel_5_API_28:
```

 Megadtam, hogy melyik emulátor legyen az alapértelmezett.

Ezekkel a lépésekkel gyakorlatilag befejeződött a React Native beüzemelésének folyamata, a projekt a terminálból az `npx react-native run-android` paranccsal bármikor indítható. Én a `yarn android` paranccsal tettem ezt meg, amelyhez szükséges egy előzetes yarn telepítés az `npm install -global yarn` paranccsal. Ez csak preferencia kérdése.

## 4. Hardverek összeállítása

Mielőtt hozzákezdtem volna bármelyik rendszernek az összeállításához, először is meg kellett terveznem azok funkcionalitását.

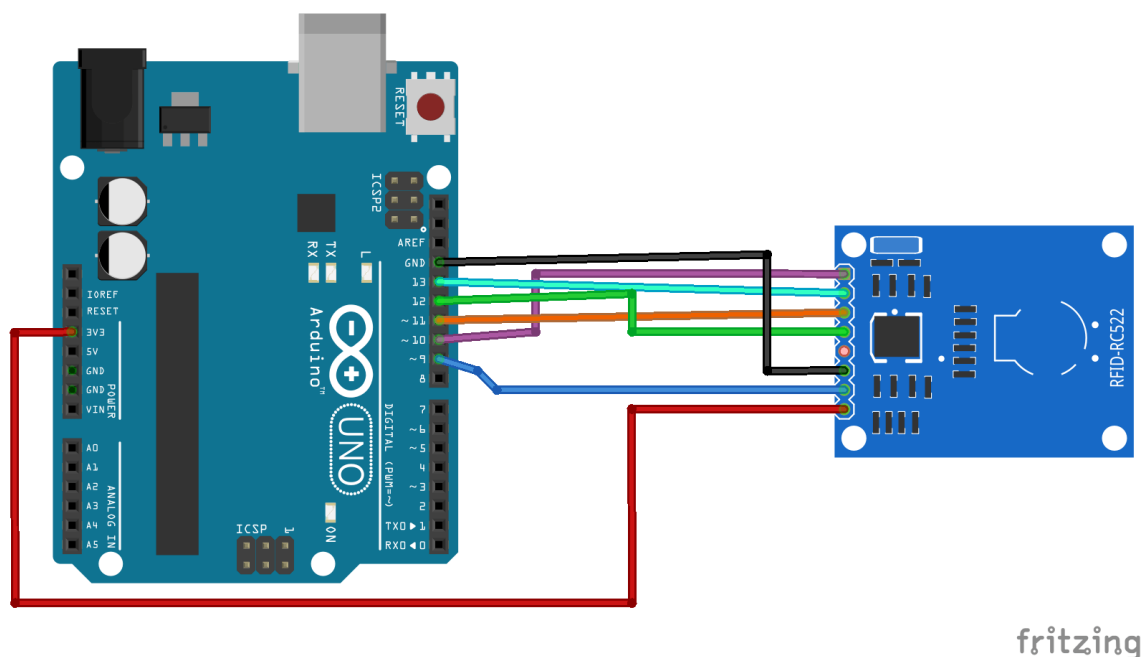
Az RFID-s kártyaolvasó-rendszer nyitásának feltétele, hogy kártyaérintés esetén megvizsgálja, hogy egy adott adatbázison belül szerepel-e a használt RFID ID-ja és a használt Arduino ID-ja. Ha az RFID-ID szerepel az adatbázis rendszerben, és annak az ajtónyitási státusza az engedélyezett az adott Arduino ID-ra, akkor kinyitja az ajtót. Más eltérő esetben megtagadja ezt, és nem nyílik ki az ajtó. Olyan RFID-ID ajtónyitási próbálkozása esetén, amely szerepel az adatbázisban, elmenti azon RFID ID-ját és annak használati idejét egy „logs” adattáblába, ajtónyitási státuszától függetlenül, valamint hogy melyik Arduino-t próbálta használni. A használati eszközt – Arduino vagy Applikáció – is feltölti az adattáblába. Adatbázison kívüli RFID azonosítót nem fog elmenteni a logs-ba. Figyelembe véve, hogy egy háztartásban feltehetően minden lakónak egy-egy RFID-ja lesz – de több bejáratú ajtóval rendelkezik a lakás –, biztosítani kell az egy RFID-val való, több Arduino-hoz való hozzáférés lehetőségét.

A kamerarendszernek képesnek kell lennie állókép készítésére és annak egy web-es felületre való feltöltésére, illetve videó közvetítésre. Az állókép készítése egy PIR mozgásérzékelő használatával történne. A videó úgyszintén egy web-es felületen elérhetőnek kell lennie.

A hardverek összeállítását követően, preferenciától függően dönthetünk arról, hogy a kamerarendszert maga az Arduino lássa majd el árammal és ahhoz legyen kötve, vagy egy teljesen külön áramforráshoz legyen kötve. Figyelembe véve a kamera egyik helyről a másik helyre való áthelyezését, nem találtam praktikusabb megoldásnak a két hardver összeillesztését, így a két hardvert két különböző áramforráshoz kötöttem. Célszerű szünetmentes áramforráshoz kötni mindkét hardvert.

## 4.1. RFID-s ajtózárszerkezet összeállítása

Elsősorban összeillesztettem az Arduino Mega 2560 Board-ot az Ethernet Shield R3-mal, amelyet az RFID kártyaolvasó-rendszer létrehozásához fogok felhasználni. A rendszer megalkotásához Jumper kábeleket használtam, amelyeket az alábbi, az Arduino Uno-hoz tartozó, az Arduino hivatalos oldalán közzétett kötési rajz alapján kötöttem be (12. ábra) [15]:



12.ábra: Arduino Uno RFID sematikus rajza [15]

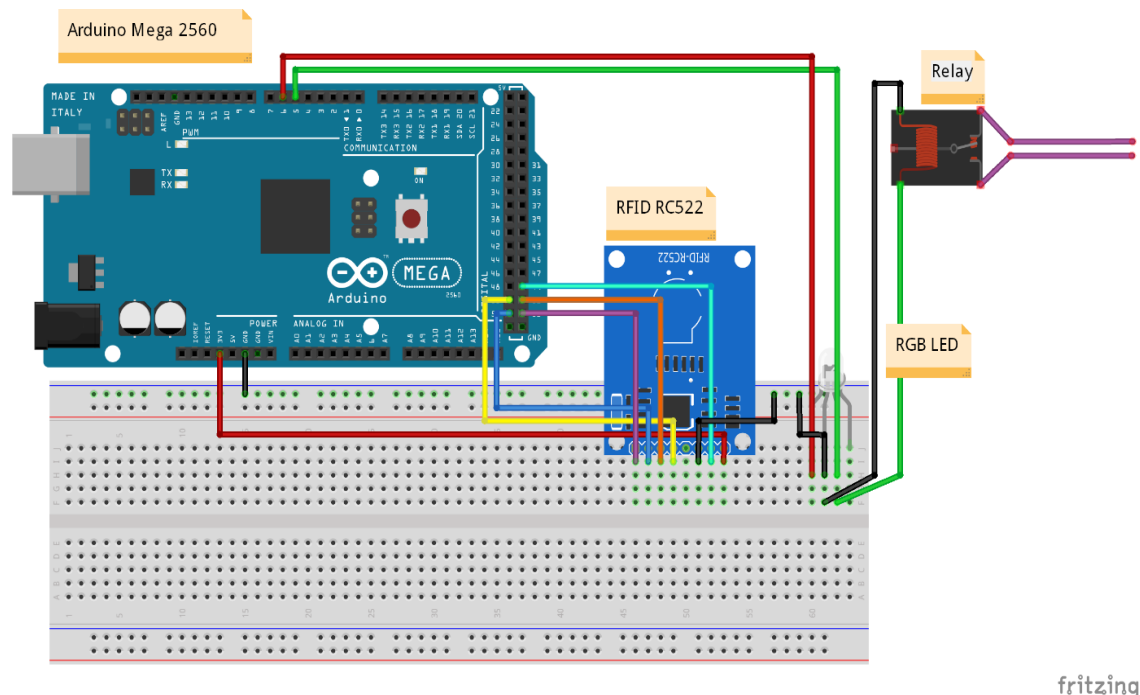
Mivel én Arduino Mega-val rendelkezem, és az Arduino Uno, illetve Arduino Mega 2560 SPI pin-jeinek az elhelyezkedései különbözőek, ezért némi módosításra volt szükségem. A két Arduino Board SPI pin-jeinek az elhelyezkedése az alábbi táblázat szerint [16]:

| Arduino/Genuino Board | MOSI | MISO | SCK | SS |
|-----------------------|------|------|-----|----|
| Uno                   | 11   | 12   | 13  | 10 |
| Mega2560              | 51   | 50   | 52  | 53 |

1. táblázat: Arduino Uno és Arduino Mega SPI Pin-ek elhelyezkedései



Az Arduino hivatalos oldalán feltüntetett sematikus rajz alapján elvégeztem a szükséges módosításokat. Ezután bekötöttem még egy RGB LED-et, mint indikátort (debugging-hoz), valamint egy relét, ami az ajtónyitást fogja elvégezni. Mindezen műveletek után a sematikus rajz az alábbi (13. ábra):



13. ábra: Kötési rajz az ajtózárszerkezetről Arduino Mega 2560-nal

A kötések táblázatos formában:

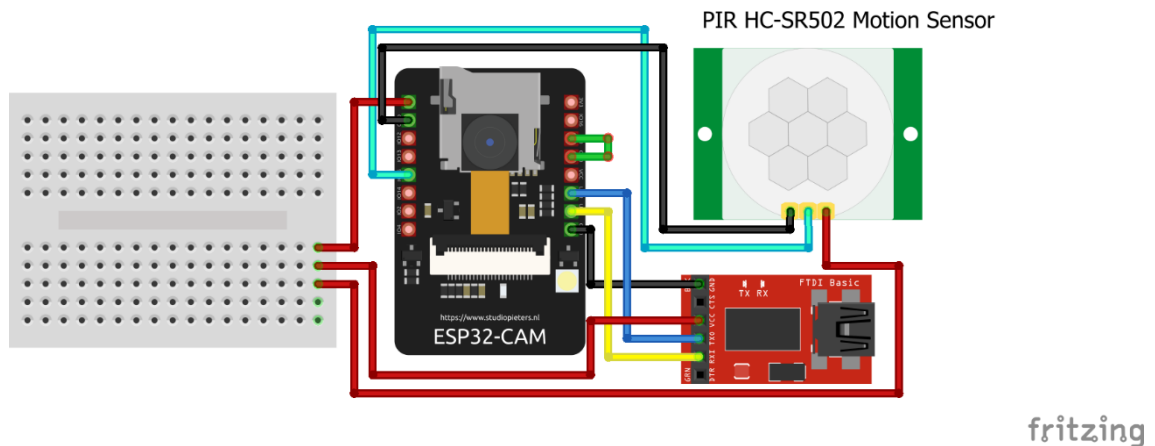
|           |      |     |     |  |      |      |     |     |
|-----------|------|-----|-----|--|------|------|-----|-----|
| Arduino   | 3.3V | Pin | GND |  | Pin  | Pin  | Pin | Pin |
| Mega 2560 | Pin  | 49  | Pin |  | 50   | 51   | 52  | 53  |
| RFID      | 3.3V | RST | GND |  | MISO | MOSI | SCK | SDA |
| RC522     | Pin  | Pin | Pin |  | Pin  | Pin  | Pin | Pin |

2. táblázat: Arduino Mega 2560 és RFID RC522 kötése

A relét az RFID bekötését követően hozzákötöttem az 5-ös digitális pin-hez. Mivel én használtam RGB LED-et indikátorként, ezért bekötöttem az RGB LED „green” katódját az 5-ös digitális pin-hez, és a „red” katódját a 6-os digitális pin-hez a Breadboard-on keresztül. Az RGB LED használata nem szükséges, csak az ajtónyitás folyamatát fogja vizualizálni zöld, illetve piros világítás formájában. Mindössze már csak a relét kellett összekötni az ajtózárszerkezettel, és készen is állt az RFID-s ajtózárszerkezet fizikailag.

## 4.2. A kamerarendszer összeállítása

A kártyaolvasó-rendszer megalkotása után hozzáálltam a kamerarendszer megalkotásához. Az ESP32-Cam AI-Thinker leprogramozásához egy FT232RL FTDI programert használok az alábbi kötési rajz alapján (14. ábra):



14.ábra: A kamerarendszer kötési rajza

A kötési rajz táblázatos formában:

|                 |     |     |     |     |     |     |        |
|-----------------|-----|-----|-----|-----|-----|-----|--------|
| FT232RL<br>FTDI | DTR | RX  | TX  | VCC | CTS | GND |        |
| ESP32-<br>CAM   |     | UOT | UOR | 5V  |     | GND | GPIO15 |
| PIR<br>HC-SR502 |     |     |     | 5V  |     | GND | OUT    |

3. táblázat: A kamerarendszer huzalozása

A kamerarendszerem huzalozásához és annak kapcsolási rajzának elkészítéséhez az „ESP32-CAM Post Images to Local or Cloud Server using PHP (Photo Manager)” [17] és a „Discord Security Camera with an ESP32” [22] projektek kapcsolási rajzait vettem alapul.

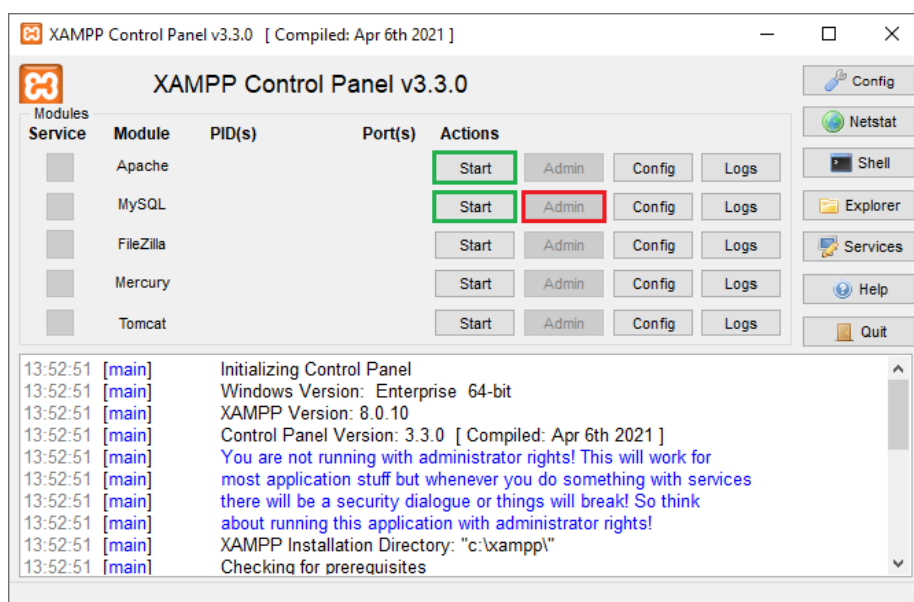
A GPIO0-nak és a GND-nak össze kell lennie kötve ahhoz, hogy programozható legyen a kamera [17]. Miután feltöltöttük programunkat az ESP32-CAM-re, eltávolíthatjuk a GND és GPIO0 huzalát, valamint magát az FT232RL FTDI programmer-t is teljesen kivehetjük a rendszerünkől, egyedül az áramellátást kell majd csak biztosítani a későbbiekben.

## 5. Összeállított hardverek programozása

Mindkét hardver – kártyaolvasó-rendszer és kamerarendszer – programozását az Arduino IDE-ben fogom megtenni. Mindezek előtt viszont szükségem volt egy backend-re, amely aszerint hajtja végre az ajtónyitás műveletét, hogy az RFID-ID és az Arduino-ID az szerepel-e az adatbázisban, vagy sem. A backend-hez PHP-t használtam, amelyet az Atom nevű szoftveren belül fejlesztettem.

### 5.1. Az adatbázisrendszer megalkotása

Az adatbázisrendszerem elkészítéséhez XAMPP-t, azon belül phpMyAdmin-t használtam. Ezt a későbbiekben Domain-re fogom lecserélni, de elsősorban szükségem volt egy lokális hálózatra, amely segítségével könnyen megalkothatom, illetve tesztelhetem a rendszeremet. Erre igencsak kényelmes opció az XAMPP használata. Amikor szükségem van a backend-re, vagy az adatbázisra, akkor előzetesen mindig elindítom az XAMPP-t az alábbi módon (15. ábra):



15. ábra: XAMPP vezérlőpanel

Ezekkel a lépésekkel létrehoztam saját lokális hálózatomat, amely adatbázissal is rendelkezik. A fejlesztés teljes időtartama alatt az adatbázisrendszerem folyamatos módosításokon esett át, míg a végleges adatbázisrendszeremet az alábbiakban leírt módon

készítettem el. Először is létrehoztam egy adatbázist „login\_db” néven, amelyen belül 5 táblázatot készítettem el:

1. users:

```
CREATE TABLE `users` (  
  `id` bigint(20) NOT NULL AUTO_INCREMENT,  
  `user_id` varchar(40) NOT NULL,  
  `email` varchar(40) NOT NULL,  
  `user_name` varchar(40) NOT NULL,  
  `password` varchar(100) NOT NULL,  
  `rfid_id` varchar(40) NOT NULL DEFAULT 'Undefined',  
  `date` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE  
  current_timestamp(),  
  `admin_user` tinyint(1) NOT NULL DEFAULT 0,  
  `token` varchar(40) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `user_id` (`user_id`),  
  KEY `date` (`date`),  
  ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4
```

A „users” adattáblában helyezkednek el a felhasználók regisztrációs adatai. Minden user-hez fog tartozni egy „user\_id”, amellyel a többi adattáblában tudjuk majd beazonosítani, hogy melyik Arduino-hoz fér hozzá attól függően, hogy rendelkezik-e RFID ID-val, stb. Természetesen van felhasználói neve („user\_name”), illetve jelszava („password”), amelyekkel sikeres regisztráció után majd be tud jelentkezni. Sikeres regisztrációt követően alapértelmezettként nem fog rendelkezni RFID ID-val, ez majd az applikáción belül vagy adatbázison keresztül módosítható. A jelszava backend-en keresztül md5() hash-sztringként van elmentve az adattáblában a nagyobb biztonság megteremtésének érdekében. Minden egyes bejelentkezés során a felhasználó egy új „token”-nel fog rendelkezni, amellyel beazonosítjuk a bejelentkezett felhasználót a „user\_id”-je alapján.

## 2. arduino\_devices:

```
CREATE TABLE `arduino_devices` (  
  `id` bigint(20) NOT NULL AUTO_INCREMENT,  
  `arduino_id` varchar(40) NOT NULL,  
  `video_url` varchar(40) NOT NULL DEFAULT 'Undefined',  
  PRIMARY KEY (`id`),  
  KEY `arduino_id` (`arduino_id`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4
```

Az „arduino\_devices” adattábla tartalmazza az összes jelenleg elérhető Arduino eszköz „arduino\_id”-ját és az ahhoz tartozó Video Streaming URL-jét. Manuálisan bővíthető.

## 3. rfid\_manager:

```
CREATE TABLE `rfid_manager` (  
  `id` bigint(20) NOT NULL AUTO_INCREMENT,  
  `rfid_id` varchar(40) NOT NULL,  
  `access_state` tinyint(1) NOT NULL DEFAULT 1,  
  `arduino_id` varchar(40) NOT NULL DEFAULT 'Undefined',  
  `isRequested` tinyint(1) NOT NULL DEFAULT 0,  
  `user_request` tinyint(1) NOT NULL DEFAULT 0,  
  PRIMARY KEY (`id`),  
  KEY `rfid_id` (`rfid_id`),  
  KEY `access_state` (`access_state`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4
```

Az „rfid\_manager” adattábla tartalmazza, hogy egyes „rfid\_id”-k melyik „arduino\_id”-kkal rendelkeznek, és hogy ezeknek milyen az „access\_state”-je. Az „access\_state” határozza meg, hogy a felhasználó „rfid\_id”-jának az adott „arduino\_id”-ra van-e belépési engedélye, vagy sem. Azon rekord esetén, ahol az „isRequested” frissítve van 1 értékre, elvégzi az Arduino az ajtónyitást, majd 0-ra

visszaállítja annak értékét. Admin felhasználó applikáción keresztül Undefined-ra állíthatja azon rekord „rfid\_id”-ját, ahol a „user\_request” értéke 1.

#### 4. arduino\_logs:

```
CREATE TABLE `arduino_logs` (  
  `id` bigint(20) NOT NULL AUTO_INCREMENT,  
  `rfid_id` varchar(255) NOT NULL,  
  `arduino_id` varchar(255) NOT NULL,  
  `date` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE  
  current_timestamp(),  
  `state` tinyint(1) NOT NULL,  
  `openedWith` varchar(11) NOT NULL,  
  PRIMARY KEY (`id`),  
  KEY `rfid_id` (`rfid_id`),  
  KEY `arduino_id` (`arduino_id`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4
```

Az „arduino\_logs” adattáblába mentődnek el a már rendszerben levő ajtónyitási próbálkozások RFID ID, Arduino ID, dátum, eszköz és sikeresség szerint.

#### 5. pictures:

```
CREATE TABLE `pictures` (  
  `id` bigint(20) NOT NULL AUTO_INCREMENT,  
  `img_name` varchar(40) NOT NULL,  
  `date_added` timestamp NOT NULL DEFAULT current_timestamp() ON  
  UPDATE current_timestamp(),  
  `arduino_id` varchar(40) NOT NULL,  
  PRIMARY KEY(`id`),  
  KEY `img_name` (`img_name`),  
  KEY `arduino_id` (`arduino_id`)  
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4
```

A „pictures” adattáblában szerepelnek az ESP32-CAM által készített képek nevei („img\_name”), illetve azon Arduino ID-ja, amellyel kapcsolatban áll. A kép készítési idejét is logolja.

## 5.2. A kártyaolvasó-rendszer programozása

Az Arduino IDE elég nagy számú példakóddal rendelkezik. Ezek a kódok kifejezetten hasznosak ahhoz, hogy megértsük egyes library-k és kódok működését. Esetemben a `File > Examples > Ethernet > WebClient` útvonalon elérhető példakód kifejezetten jó kiindulópontnak bizonyosodott. Ez a kód elsősorban megnézi, hogy van-e Ethernet Shield és Ethernet kábel megfelelően hozzácsatlakoztatva az Arduinohoz, és az alapján létrehoz egy Ethernet lokális hálózatot. Ezt követően, ha van kapcsolat az elején definiált szerverrel, akkor GET Request segítségével adatokat kér le a szerverről. Néhány változó, illetve kódrészlet átírását követően gyakorlatilag készen is van a program kapcsolati része (16. ábra).

```
void getDats(String uid, String UniqueIDString) {
  if (client.connect(server, 80)) {
    Serial.print("connected to ");
    Serial.println(client.remoteIP());
    file_loc = "/PHPproject/arduino/arduino_postman/rfid_test.php";
    req = "GET " + file_loc + "?rfid_id=" + uid + "&arduino_id=" + UniqueIDString + " HTTP/1.1";
    client.println(req);
    client.println("Host: 192.168.0.161");
    client.println("Connection: close");
    client.println();
  } else {
    Serial.println("connection failed");
  }
}
```

16. ábra: Arduino IDE Get Request

Egy másik példakód, amit felhasználtam, az az RFID beolvasásáért felel. A példakód a `File > Examples > MFRC522 > ReadNUID` elérési útvonalon érhető el. Az RFID ID az „uid” változóban van tárolva, míg az Arduino ID a „UniqueIDString”-ben [18][18]. Az MFRC522 az nem egy alapértelmezetten létező library, azt előzetesen telepíteni kell. Ezt könnyen meg lehet tenni a `Tools > Manage Libraries` menüpontonál.

Kártyaérintés esetén egy GET Request segítségével kér le adatokat az adatbázisból az „rfid\_test.php”-n keresztül. Abban az esetben, ha a használt RFID ID és a használt Arduino ID szerepel az adatbázisban, és az ajtónyitási státusz az engedélyezett, akkor

kinyitja az ajtót relé segítségével (17. ábra). Ellenkező esetben nem nyitja ki az ajtót. A próbálkozásokat elmenti a „logs” adattáblában attól függően, hogy az „rfid\_id” az valid.Az openDoor() funkció nyitja ki az ajtót a relé segítségével, amely digitalWrite(6, HIGH), majd digitalWrite(6, LOW) utasítást hajt végre.

```
int len = client.available();
if (len > 0) {
    for (int i = 0; i < len; i++) {
        char c = client.read();
        http_response += c;
    }
    response_start = http_response.indexOf("<data>") + 6;
    response_end = http_response.indexOf("</data>");
    http_response = http_response.substring(response_start, response_end);
    Serial.print("Website response : ");
    Serial.println(http_response);
    Serial.println();

    if (http_response == "true") openDoor();
    if (http_response == "false") wrongCard();
}
```

17. ábra: Arduino IDE http response kódrészlet

Így megtörténik az Arduino és PHP közötti kommunikáció, de későbbiekben ha az applikációból szeretnénk kinyitni az ajtót, az így nem működne, ezért az applikációból Post Request segítségével frissítem a „users” adattábla „isRequested” értékét 1-re a használt „rfid\_id” és „arduino\_id”-nál.

Az Arduino 5 másodpercenként egy másik Get Request-tel adatokat kér le az „open\_door.php”-n keresztül az „rfid\_manager” adattáblából. Ha van olyan rekord, ahol az „arduino\_id” az Arduino ID-val megegyezik és annak az „isRequested” értéke 1, akkor kinyitja az ajtót, majd az összes „isRequested” értéket frissíti 0-ra a használt Arduino ID-nál és log-olja az ajtónyitást. Applikációból csak olyan rekorddal küldhetünk Post Request-et, amelynek ajtónyitási státusza 1 az adott Arduino ID esetében (18. ábra).



```

1 <?php
2 include("connection.php");
3 if($_SERVER['REQUEST_METHOD'] == "POST"){
4     $headers = apache_request_headers();
5     if(isset($headers['token'])){
6         $token = $headers['token'];
7         $query = "SELECT * FROM users WHERE token = '$token' limit 1";
8         $result = mysqli_query($con,$query);
9         if(!mysqli_num_rows($result)){
10             header("HTTP/1.1 402 Unauthorized");
11             return;
12         }
13         $arduino_id = $_POST['arduino_id'];
14         $rfid_id = $_POST['rfid_id'];
15         $query = "UPDATE rfid_manager SET isRequested = 1 WHERE rfid_id = '$rfid_id' AND arduino_id = '$arduino_id'";
16         $result = mysqli_query($con,$query);
17         $res = (object) ['status' => boolval(true), 'isOpen' => boolval(true)];
18         echo json_encode($res);
19         return;
20     }
21 }
22
23 if ($_SERVER['REQUEST_METHOD'] == "GET"){
24     $arduino_id = $_GET['arduino_id'];
25     $query = "SELECT * FROM rfid_manager WHERE arduino_id = '$arduino_id'";
26     $result = mysqli_query($con,$query);
27     $all = mysqli_fetch_all($result, MYSQLI_ASSOC);
28     foreach($all as $row){
29         if ($row['isRequested'] == 1) {
30             $rfid_id = $row['rfid_id'];
31             $query = "UPDATE rfid_manager SET isRequested = 0 WHERE arduino_id = '$arduino_id'";
32             $result = mysqli_query($con,$query);
33             echo("<data>true</data>");
34             $query = "INSERT INTO arduino_logs(rfid_id, arduino_id, state, openedWith)
35                 VALUES('$rfid_id', '$arduino_id', 1, 'Application')";
36             $result = mysqli_query($con, $query);
37             return;
38         }
39     }
40 }
41 ?>

```

18. ábra: open\_door.php kódrészlet

A kártyaérintés esetén történő GET Request hasonlóan működik a fenti képen látható GET Request-tel.

A „connection.php” minden olyan fájl esetén include-olva van, ahol az adatbázissal valamilyen műveletet hajtok végre (19. ábra).

```

1 <?php
2 $dbhost = "localhost";
3 $dbuser = "root";
4 $dbpass = "";
5 $dbname = "login_db";
6 if(!$con = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname)){
7     die("Failed to connect". mysqli_connect_error());
8 }
9 ?>

```

19. ábra: connection.php

### 5.3. A kamerarendszer programozása

Ahhoz, hogy le tudjuk majd programozni az ESP32-CAM kameránkat, fel kell telepíteni magát az ESP32 ADD-on-t [19]. Alapértelmezetten, frissen feltelepített Arduino IDE esetén nem fog rendelkezésünkre állni az ESP32 Arduino a Boards Manager-nél. Előzetesen fel kell telepíteni az FTDI Drivert is, hogy az Arduino IDE lássa az FTDI programmer Port-ját és hogy le tudjuk programozni a kamerát. Az ESP32 Arduino Board feltelepítésének lépései:

1. Az Arduino IDE-ben kattintsunk a **File > Preferences** opcióra
2. Az **Additional Boards Manager URLs:**-hez adjuk hozzá az alábbi két fájlt:
  - [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json),
  - [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)
3. Ezt követően nyissuk meg a **Tools > Board > Boards Manager...**-t
4. A Boards Manager keresőfelületébe írjuk be, hogy ESP32, és installáljuk.

A kamerarendszert a Random Nerd Tutorials oldalán publikált, valamint a CiferTech által az Arduino főoldalán publikált projektek felhasználásával készítettem el:

- „ESP32-CAM Post Images to Local or Cloud Server using PHP (Photo Manager)” [17]
- „How to Access ESP32-CAM Worldwide using ngrok” [20]

Egyes változók kivételével -amelyeket az alábbiakban részletezek- ugyanazt a kódot használtam, mint ami az általuk publikált, fentebb említett két projektben található.

Először is definiáltam a projekthez szükséges library-ket (20. ábra):

```
#include <Arduino.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <WiFi.h>
```

20. ábra: Szükséges library-k

Az ESP32 WiFi-re csatlakozik, ezért elmentem változókba az Arduino ID-ját, a WiFi-hálózat adatait és a server adatait, valamint definiáltam azt a Pin-t, amelyhez a mozgásérzékelő van kötve (21. ábra).

A server az még lokális hálózat, a domain-re való átírást a későbbiekben fogom megtenni, amikor a rendszer már teljesen működőképes. Az „upload.php” backend fájl

fogja elvégezni majd a képfeltöltést. Azt az Arduino ID-t adtam meg, amelyik a beléptető rendszert kezeli.

```
const char* ssid = "WiFiNev";
const char* password = "WiFiJelszo";
String serverName = "192.168.0.161";
String serverPath = "/PHPproject/arduino/arduino_postman/upload.php";
const int serverPort = 80;
int gpioPIR = 15;
String arduino_id = "102030405060708090";
unsigned long previousMillis = 0;
bool connected = false;
```

21. ábra: Szükséges változók definiálása

Ezt követően definiálom a WiFi klienseket és a kamera modellhez tartozó GPIO Pin-eket (esetemben a kamera az ESP32-CAM CAMERA\_MODEL\_AI\_THINKER) [21]. Ügyelni kell arra, hogy nem mindenkinek szükségszerűen ugyanezeket a GPIO Pin definiálásokat kell elvégeznie, ez minden modellnél kissé eltérő (22. ábra).

```
WiFiServer server(80);
WiFiClient live_client;
WiFiClient client;
// CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM    32
#define RESET_GPIO_NUM  -1
#define XCLK_GPIO_NUM    0
#define SIOD_GPIO_NUM    26
#define SIOC_GPIO_NUM    27
#define Y9_GPIO_NUM      35
#define Y8_GPIO_NUM      34
#define Y7_GPIO_NUM      39
#define Y6_GPIO_NUM      36
#define Y5_GPIO_NUM      21
#define Y4_GPIO_NUM      19
#define Y3_GPIO_NUM      18
#define Y2_GPIO_NUM       5
#define VSYNC_GPIO_NUM   25
#define HREF_GPIO_NUM    23
#define PCLK_GPIO_NUM    22
```

22. ábra: Kameramodel definiálása

A sendPhoto() funkció a mozgás esetén a képek készítését és azok elküldését biztosítja. Ez a kódrészlet szinte teljesen megegyezik a Random Nerd Tutorials által publikált „ESP32-CAM Post Images to Local or Cloud Server using PHP (Photo

Manager)” [17] projektben található kódrészlettel, annyi különbséggel, hogy a Post Requestnél az Arduino ID változót is tovább adom a „filename”-nél. A mozgásérzékelős Photo Capture nem készít webes felületet, hanem közvetlenül az „upload.php”-be felküldi az adatokat egy POST Request segítségével, amiben egy temp fájlt fog létrehozni és attól függően, hogy ez a fájl megfelelő kiterjesztésű és méretű, akkor ennek a temp fájlnak a tartalmát fogja áthelyezni egy jpg kiterjesztésű fájlba.

A Random Nerd Tutorials által publikált „uploads.php” fájlban szereplő kód felhasználásával készítettem el a saját „uploads.php” backend fájlt.

A fájlméretre, -típusra és -kiterjesztésre való ellenőrzések megegyeznek, de a fontosabb funkciók és a fájlnévezés kissé eltérnek (23. ábra).

```
6  $file_tmp = $_FILES['imageFile']['tmp_name'];
7  $file_name = $_FILES['imageFile']['name'];
8  $temp = explode(".", $file_name);
9  $tempname = explode("---", reset($temp));
10 $arduino_id = end($tempname);
11 $esp32 = reset($tempname) . "_";
12 //echo $file_name;
13
14 $directory = "uploads/";
15 $filecount = 0;
16 $files = glob($directory . "*"); /* = any filetype
17 if ($files) {$filecount = count($files);}
18 $filecount += 1;
19 $newfilename = $esp32 . $filecount . '.' . end($temp);
20 $file_destination = 'uploads/' . $newfilename;
```

23. ábra. fájlnévezés

Ha létezik „arduino\_id”, és az upload státusz az True, akkor megtörténik a fájl áthelyezése, valamint a fájl neve és az Arduino ID feltöltődik a „pictures” adattáblába (24. ábra).

```

60 if ($uploadOk == 0) {
61     header("HTTP/1.1 412 Precondition Failed"); //file was not uploaded.
62     $res = (object) ['isUploaded' => boolval(false), 'isValidType' => boolval(false)];
63     echo json_encode($res);
64     return;
65 }
66 else {
67     if (isset($arduino_id)) {
68         if(move_uploaded_file($file_tmp, $file_destination)){
69             include_once("connection.php");
70             $query = "INSERT INTO pictures (img_name, arduino_id)
71                 values ('$newfilename', '$arduino_id')";
72             $result = mysqli_query($con, $query);
73             if($result){
74                 $res = (object) ['isUploaded' => boolval(true)]; //was uploaded
75                 echo json_encode($res);
76             }
77         }
78         else{
79             header("HTTP/1.1 404 Not Found"); //file was not uploaded.
80             $res = (object) ['isUploaded' => boolval(false)];
81             echo json_encode($res);
82         }
83     }
84     else {
85         header("HTTP/1.1 428 Precondition Required"); //file was not uploaded, arduino missing.
86         $res = (object) ['isUploaded' => boolval(false), 'isMissing' => "arduino_id"];
87         echo json_encode($res);
88     }
89 }
90 ?>

```

24. ábra: uploads.php kódrészlet

A fentebb látható kódrészletet megelőzően csak fájlnevmódosítás, fájlformátumra és méretre való ellenőrzések történnek.

A PIR mozgásérzékelő programozása igencsak egyszerű, csak annyit kell megtennie, ha mozgást észlel (digitális olvasási értéke 1), akkor futtassa le a sendPhoto() funkciót [22]. Arra kell csak figyelni, hogy a loop()-ban helyezkedjen ez a kódrészlet (25. ábra).

```

void loop() {
  pinMode(gpioPIR, INPUT_PULLUP);
  int motionValue = digitalRead(gpioPIR);
  if (motionValue == 1)
  {
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= 2000) {
      sendPhoto();
      previousMillis = currentMillis;
    }
  }
  http_res();
  if (connected == true) {
    liveCam(live_client);
  }
}

```

25. ábra: Mozgásérzékelő kódrészlet

A Video Streaming esetén a kód teljesen megegyezik a CiferTech által publikált „How to Access ESP32-CAM Worldwide using ngrok” projekt kódjával, amelyet az Arduino főoldalán tettek közzé [20]. A 25. ábrán látható `http_res()` és a `liveCam()` funkciók felelnek a Video Streaming létrehozásáért.

Ha minden jól van definiálva, és az ESP32 Arduino Board is sikeresen telepítve lett, akkor máris programozható az ESP32-CAM az alábbi konfigurációkkal a **Tools** menüpontban:

|   |
|---|
| Board: „AI THINKER ESP32-CAM”<br>CPU Frequency: „240MHz (WiFi/BT)”<br>Flash Frequency: „80Mhz”<br>Flash Mode: „QIO”<br>Port: „COM5” |
|---|

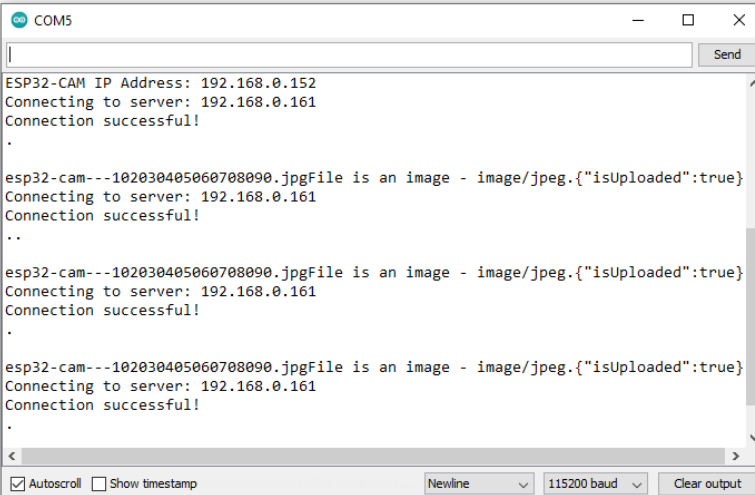
Elképzelhető, hogy más konfigurációs beállításokkal is megfelelően programozná le a kamerát, de én ezeket az opciókat használtam és tökéletesen működött. Kódfeltöltés során legyen összekötve az ESP32 egyik GND Pin-je a GPIO0 Pin-nel.

Ha a kódfeltöltés során megjelenne a **Connecting... ..** üzenet, akkor meg kell nyomni az ESP32-CAM-on a RESET gombot. Kódfeltöltés után meg kell szüntetni a GND és GPIO0 közötti kapcsolatot, majd ismételtén meg kell nyomni a RESET gombot.

A Serial-on megjelenő ESP32 IP-vel meg tudjuk tekinteni a Video Streaming-et és a képfeltöltésről egy állapotot ír ki (26. ábra). A Serial Monitor-on látható jpg név az az a fájlnev, amelyet az Arduino IDE-n belül adtam meg a GET Request-nél a „filename” változóba. Backend-en belül ezt alakítom át a 23. ábrán látható kódrészlet alapján.

## Index of /PHPproject/arduino/arduino\_postman/uploads

| Name                             | Last modified    | Size | Description |
|----------------------------------|------------------|------|-------------|
| Parent Directory                 |                  |      |             |
| <a href="#">esp32-cam_1.jpg</a>  | 2021-11-03 12:13 | 27K  |             |
| <a href="#">esp32-cam_2.jpg</a>  | 2021-11-03 12:13 | 18K  |             |
| <a href="#">esp32-cam_3.jpg</a>  | 2021-11-03 12:13 | 17K  |             |
| <a href="#">esp32-cam_4.jpg</a>  | 2021-11-03 12:14 | 22K  |             |
| <a href="#">esp32-cam_5.jpg</a>  | 2021-11-03 12:41 | 19K  |             |
| <a href="#">esp32-cam_6.jpg</a>  | 2021-11-03 13:05 | 23K  |             |
| <a href="#">esp32-cam_7.jpg</a>  | 2021-11-03 13:09 | 23K  |             |
| <a href="#">esp32-cam_8.jpg</a>  | 2021-11-03 16:38 | 21K  |             |
| <a href="#">esp32-cam_9.jpg</a>  | 2021-11-03 16:38 | 32K  |             |
| <a href="#">esp32-cam_10.jpg</a> | 2021-11-03 16:39 | 31K  |             |
| <a href="#">esp32-cam_11.jpg</a> | 2021-11-03 16:47 | 33K  |             |
| <a href="#">esp32-cam_12.jpg</a> | 2021-11-03 16:47 | 33K  |             |
| <a href="#">esp32-cam_13.jpg</a> | 2021-11-03 16:48 | 33K  |             |
| <a href="#">esp32-cam_14.jpg</a> | 2021-11-03 16:48 | 20K  |             |
| <a href="#">esp32-cam_15.jpg</a> | 2021-11-03 17:33 | 22K  |             |
| <a href="#">esp32-cam_16.jpg</a> | 2021-11-03 17:33 | 22K  |             |

```
ESP32-CAM IP Address: 192.168.0.152
Connecting to server: 192.168.0.161
Connection successful!
.
esp32-cam---102030405060708090.jpgFile is an image - image/jpeg.{\"isUploaded\":true}
Connecting to server: 192.168.0.161
Connection successful!
..
esp32-cam---102030405060708090.jpgFile is an image - image/jpeg.{\"isUploaded\":true}
Connecting to server: 192.168.0.161
Connection successful!
.
esp32-cam---102030405060708090.jpgFile is an image - image/jpeg.{\"isUploaded\":true}
Connecting to server: 192.168.0.161
Connection successful!
.
```

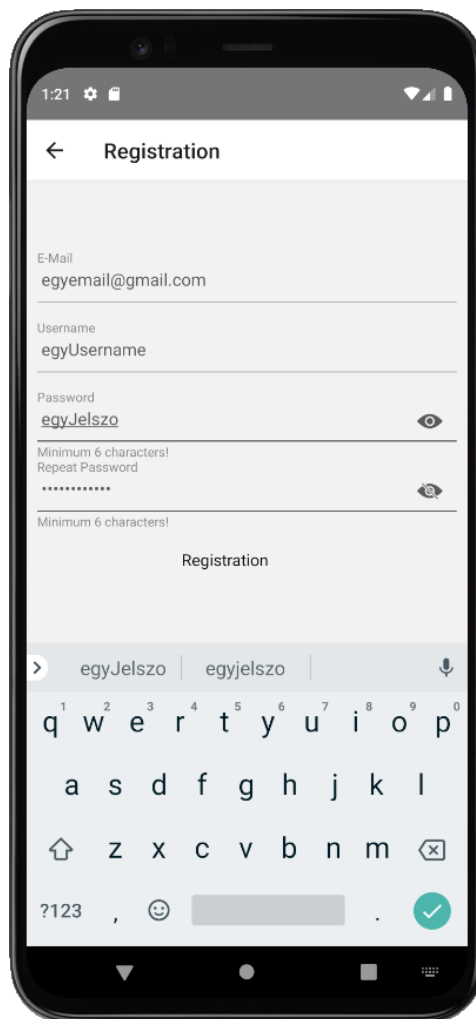
Apache/2.4.48 (Win64) OpenSSL/1.1.1f PHP/8.0.10 Server at 192.168.0.161 Port 80

26. ábra: Serial Monitor és a képek web-en tárolva

## 6. Az applikáció fejlesztése

Elsősorban a regisztráció, bejelentkezés és kijelentkezés opciókat valósítottam meg. Amikor megnyitjuk az applikációt mobil eszközünkről, de előzetesen már be voltunk jelentkezve, akkor az AsyncStorage-ba elmentődött felhasználói adatokkal azonnal belép.

A login és a regisztráció szerkezetileg szinte teljesen megegyezik. Az applikáció megnyitásakor a Login View lesz az első oldal, ahol bejelentkezhetünk (akár AsyncStorage-ba elmentett értékkel) vagy átléphetünk a Registration View-ra, ha nem rendelkezünk még regisztrált felhasználói fiókkal. A Registration View-nál az adatokat négy Text Field kezeli az input-okat, amelyek az „E-mail”, „Username”, „Password” és „Repeat Password” Text Field-ek. Egy TouchableOpacity-be van elhelyezve egy Text Field, ami a regisztrációs gombot fogja kezelni (27. ábra).



27 . ábra: Registration View



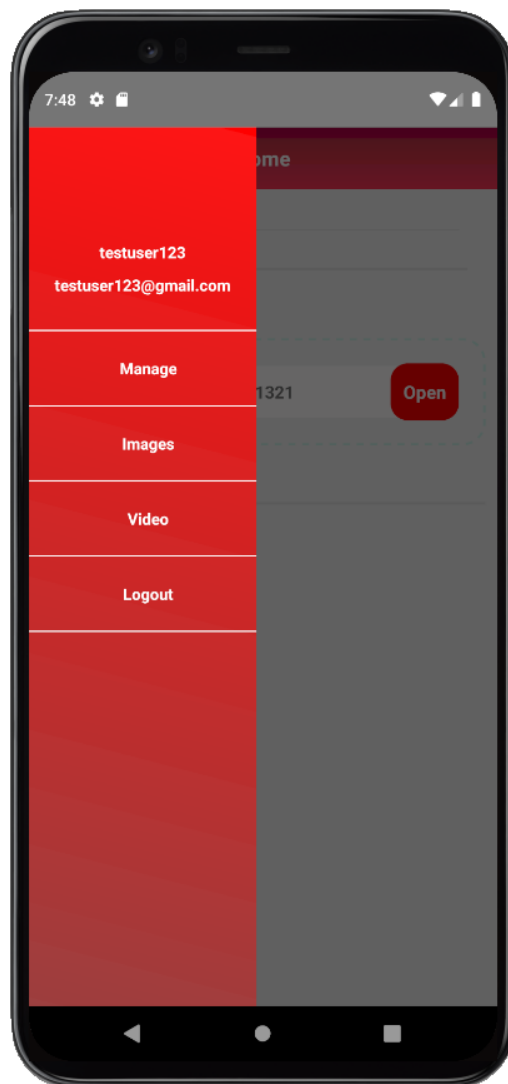
A Registration gomb megnyomását követően a begépelt text-eket egy POST Request segítségével elküldi a backend számára (28. ábra). Üres text mezők esetén nem lép tovább, hanem hibával tér vissza. Ha megfelelően voltak kitöltve a mezők, illetve a regisztrálandó „Username” az Unique (azaz nem szerepel az adatbázisban), valamint ha a „jelszó” és a „jelszó ismét” megegyezik, akkor feltölti a User adatait a „users” adattáblába. A jelszó md5()-ként mentődik [23]. Ekkor RFID ID-val és ahhoz rendelt Arduino ID-val még nem rendelkeznek az újonnan regisztrált felhasználók, mindkettő a regisztrációt követően még csak az alapérték „Undefined”-dal fog mentődni az adatbázisba.

```
Complexity is 6 It's time to do something...
73   register(email, username, password, password2) {
74     let formData = new FormData();
75     formData.append('email', email);
76     formData.append('user_name', username);
77     formData.append('password', password);
78     formData.append('password2', password2);
79
80     const url = configjson.base_url + 'login_postman/signup.php';
81
82     this.setState({loading: true});
83
84     fetch(url, {
85       method: 'POST',
86       body: formData,
87     }).then(response => {
88       this.setState({loading: false});
89
90       if (response.status >= 300) {
91         this.doSwitchCase(response.status);
92         return;
93       }
94       response.json().then(res => {
95         if (res.status) {
96           AsyncStorage.setItem('email', email);
97           AsyncStorage.setItem('username', username);
98           AsyncStorage.setItem('password', password);
99
100          this.props.navigation.navigate('Login');
101          Alert.alert('Signed up!', 'Welcome!');
102        }
103      });
104    });
105  }
```

28. ábra: Regisztráció Post Request

Sikeres regisztráció után átirányít a Login View-ra, valamint a regisztrációnál használt adatokat AsyncStorage-ba elmentve egy kattintással akár be is léphet az új felhasználó. A Login View működése szinte teljesen megegyezik a Registration View működésével. Itt már csak két Text Field van, ami a „Username” és „Password”. A begépelt értékeket elküldi a backend számára egy Post Request segítségével. Ha a „Username” szerepel az adatbázisban, és a jelszó md5()-ként vett értéke megegyezik az adatbázisban tárolt jelszóval, akkor belép a Home View-ra, és hozzáad egy „token”-t az adatbázisban az adott felhasználóhoz. Ezzel a token-nel azonosítom a bejelentkezett felhasználókat.

Home View-nál az AppBar-on lévő Menu gombra kattintva különböző menüpontok láthatóak. Legfelül helyezkedik el a felhasználó neve és e-mail címe, legalján a „Logout” menüpont, ezek között pedig a „Manage”, „Images” és „Video” menüpontok (29. ábra).



29. ábra: Home View Left Menu

Magán a Home oldalon az olyan Arduino ID-k jelennek meg, amelyeknek az ajtónyitási státusza az engedélyezve van a felhasználó „rfid\_id”-ja esetén. Az alábbi kódrészlet a Home oldalon az összes engedélyezett Arduino kilistázásának a megjelenítéséért felel (30. ábra).

```

358   renderInfos(myText, myText2) {
359     return (
360       <View style={HomeStyles.mainView}>
361         <Text style={HomeStyles.mainText}>{myText}</Text>
362         <TouchableOpacity
363           style={HomeStyles.touchable}
364           onPress={() => this.props.navigation.navigate('Manage')}>
365           <Text style={HomeStyles.blueText}>{myText2}</Text>
366         </TouchableOpacity>
367       </View>
368     );
369   }
370   renderDatas() {
371     if (this.state.infos.length == 0 && !this.state.loading)
372       return this.renderInfos(
373         'You Seem to have no Arduino IDs listed',
374         'Manage Arduino IDs here',
375       );
376     return (
377       <View>
378         {this.state.infos.length > 0 &&
379         !this.state.loading &&
380         this.renderInfos(
381           'My RFID ID: ' + this.state.infos[0].rfid_id,
382           'Manage User Arduino IDs here',
383         )}
384       <View style={HomeStyles.dashedBorder}>
385         {this.state.infos.map((item, index) => {
386           return (
387             <View
388               key={index}
389               style={HomeStyles.container}
390               onPress={() => {
391                 this.props.navigation.navigate('Home');
392               }}>
393               <View style={{marginLeft: 15}}>
394                 <Text style={{fontSize: 15, fontWeight: 'bold'}}>
395                   {index + 1}. ARDUINO ID: {item.arduino_id}
396                 </Text>
397               </View>
398               <TouchableOpacity
399                 onPress={() => {
400                   this.handleOpenButton(item.rfid_id, item.arduino_id);
401                 }}>
402                 <Text style={HomeStyles.openText}>Open</Text>
403               </TouchableOpacity>
404             </View>
405           );
406         })}
407       </View>
408       <View style={HomeStyles.topBorder}></View>
409     </View>
410   );
411 }

```

30. ábra: Home View kódrészlet

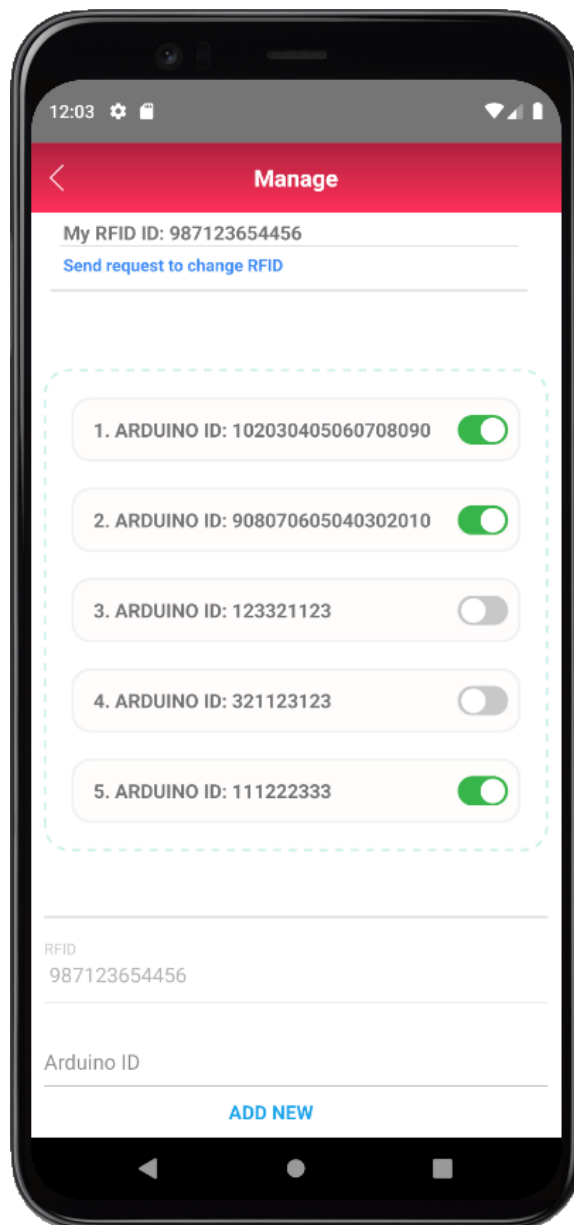
Ha még nincs RFID ID és/vagy Arduino ID, amivel rendelkezne a felhasználó, akkor azt a Manage oldalon megteheti. A „Manage Arduino IDs here” Touchable Text megnyomásával, vagy az AppBar Menu „Manage” menüpontjának megnyomásával léphet át a Manage oldalra.

A Home oldalon megjelenő Arduino ID-k a különböző ajtók nyitására felelnek. Ha az Open gombra kattint a felhasználó, akkor a 18. ábrán látható POST Request segítségével módosítja az „rfid\_manager” adattáblában az „isRequested” értékét 1-re ezen rekord esetén. Maga az Arduino 5 másodpercenként egy GET Request-tel adatokat kér le az adatbázisból, és ha az Arduino eszköz ID-je, amit szeretnénk kinyitni, és azon rekord „arduino\_id”-je, ahol a saját „rfid\_id”-nk szerepel az megegyezik, akkor kinyílik az ajtó. Ajtó kinyitását követően az „isRequested” érték 0-ra frissül.

A Logout-ra kattintva egy Delete Request-et küld a backend számára, hogy a felhasználó ki szeretne jelentkezni. A kijelentkezés a felhasználó token-je alapján működik. Ekkor megjelenik majd egy Modal, hogy a felhasználó ténylegesen ki szeretne-e jelentkezni. Igenre kattintva a Login View-ra navigálja és a User token-jét frissíti üres sztringre az adatbázisban.

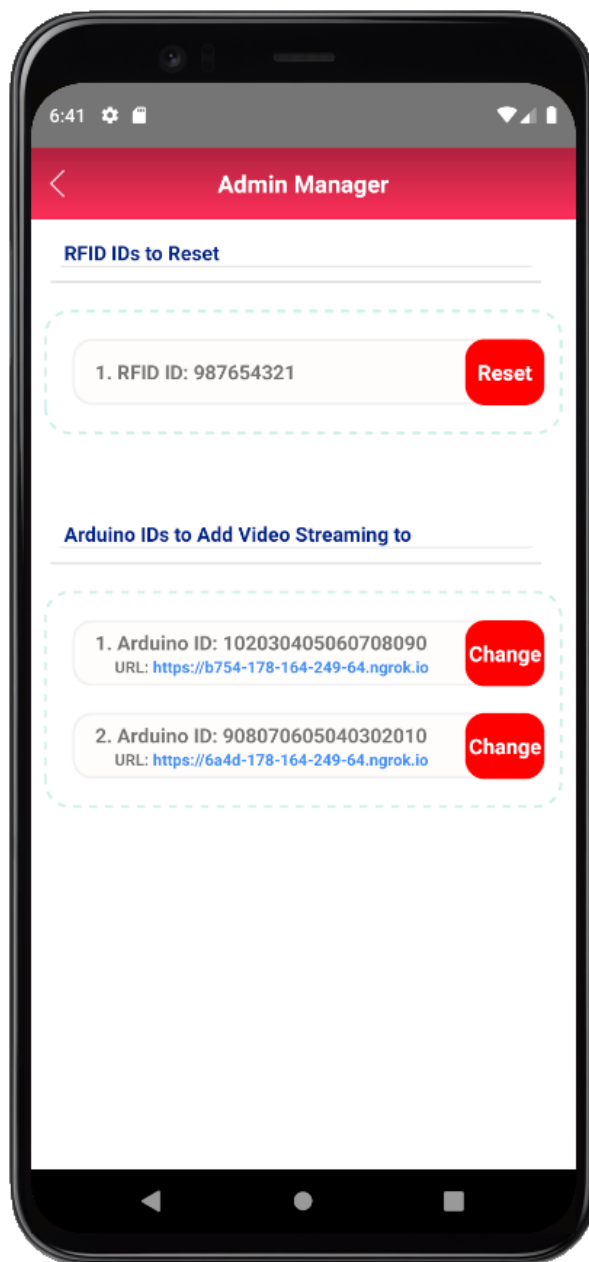
A Manage főoldal szinte ugyanúgy néz ki, mint a Home View, annyi különbséggel, hogy ezen az oldalon a felhasználó összes Arduino ID-ja szerepel, még azok is, amelyeknek az „access\_state”-je nincs engedélyezve. Itt az RFID ID-kat és Arduino ID-kat tudjuk kezelni. Új felhasználó esetén, mivel még nem rendelkezik RFID ID-vel, hozzáadhat magának egyet. A két Text Field-et kitöltve egy POST Request-et küld az „rfid\_manage.php” backend fájlra, és ha Unique az RFID ID, akkor beilleszti az adatokat az „rfid\_manager” adattáblába és a „user” táblában frissíti a felhasználó RFID ID-ját. Ügyelni kell arra, ha nincs aktív Arduino ID, akkor az „Images” és „Video” menüpontokban nem lesznek adatok, amik megjelennének. Mindkettő csak akkor jelenítődik meg, ha van olyan Arduino ID, amely alapján lehet azonosítani a megjeleníteni kívánt adatokat.

Ha a későbbiekben a felhasználó szeretne módosítani az RFID ID-ján, akkor küldhet szándékáról egy értesítést a rendszerkezelőnek a Send request to change RFID Text megnyomásával (31. ábra).



31. ábra: Manage View

Ha a felhasználó már rendelkezik RFID ID-val, akkor csak Arduino ID-t adhat még hozzá. Az Arduino eszközök „access\_state”-je állítható a Toggle gombok segítségével. Abban az esetben, ha a felhasználónak a „user” adattáblában szereplő „admin\_user” értéke 1, akkor lehetősége van kameraeszközök IP címének a hozzáadásához egyes Arduino eszközök esetén, valamint felhasználók „rfid\_id”-jeinek Reset-elésére. Ekkor a Send request to change RFID Text helyett a Manage all Users Text jelenik meg. Erre kattintva átirányítja az admin felhasználót az alábbi oldalra (32. ábra):



32. ábra: Admin Manager View

Az Admin Manager oldalon kezelheti az admin az összes Arduino eszközhöz tartozó Video Streaming URL linkjét, illetve lehetősége van Reset-elnie az RFID ID-ját azon felhasználók számára, akik ezt kérvényezték a Send request to change RFID Text megnyomásával. A Change megnyomását követően egy Modal ugrik fel, ahol egy Text mezőt kitöltve frissítheti az adatbázisban az adott Arduino ID-vel rendelkező rekord „video\_url” értékét.

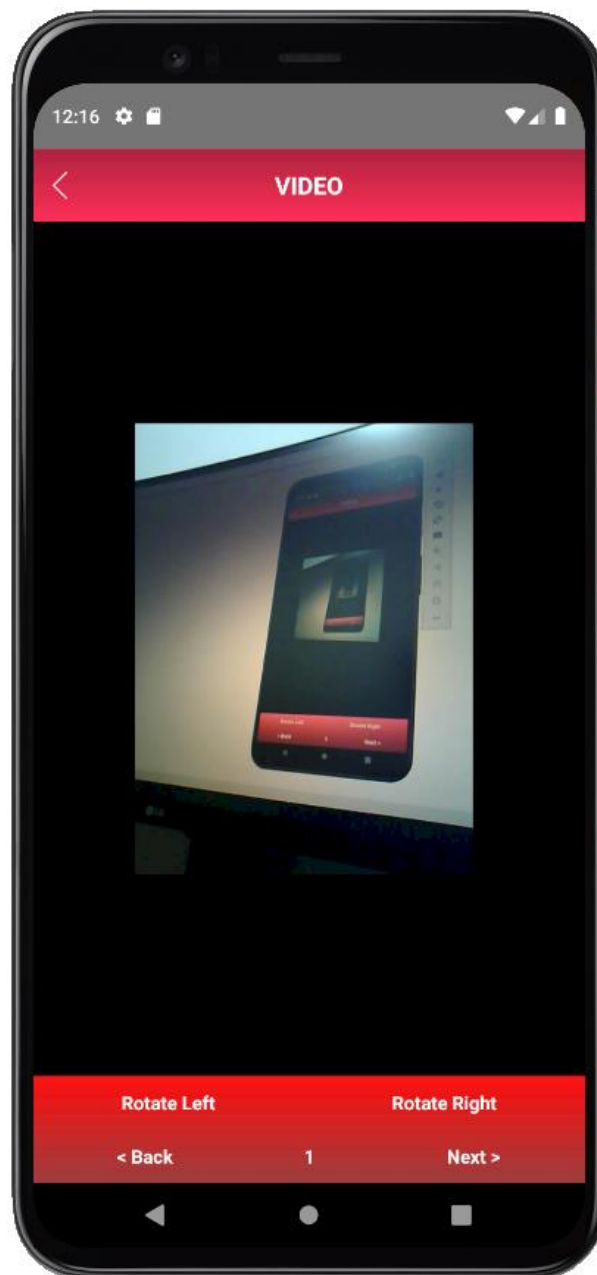
Az Images menüpontra kattintva az Images View-ra lép át. Ha rendelkezünk olyan Arduino ID-vel, amellyel készültek képek az ESP32 segítségével, és az „rfid\_manager” adattáblában engedélyezve van a hozzáférésünk, akkor megjelenítődnek a készülékekkel készült képek (33. ábra). Ellenkező esetben nem fognak képek megjelenni.



33. ábra: Images View

A könnyebb átláthatóság és a gyorsabb betöltés érdekében a képeket tízesével lehet megtekinteni az oldal megnyitásakor. Ezt a lépésközt a 33. ábrán látható alsó gombok segítségével lehet állítani. A Back és Next gombokkal lehet lépegetni az oldalak között.

A Video menüpontra kattintva az applikáció átirányít a Video oldalra. Ha a felhasználó rendelkezik olyan Arduino ID-val, amely ajtónyitási státusza engedélyezett és szerepel az „arduino\_devices” adattáblában, akkor az ugyanazon rekordban szereplő „video\_url” alapján megtekinthető a Video Streaming. A felhasználónak lehetősége van a Video Streaming forgatásához a Rotate Left és Rotate Right gombokkal. Több Arduino ID-val rendelkező felhasználónak lehetősége van a Video Streaming-ek közötti lépegetésre a Back és Next gombok segítségével (34. ábra).



34. ábra: Video View



## 7. Utolsó módosítások

Miután készen lett az applikáció, lokálisan már teljesen működőképpé vált így a projektem. A fő célom viszont az volt, hogy a létrehozott hardvereimet bárholnan elérhessem és kezelhessem. A lokális hálózatot Domain-re cseréltem, a Video Streaming-et, hogy bárholnan elérhessem, ahhoz ngrok-ot használtam.

Az ngrok egy többplatformos alkalmazás, amely lehetővé teszi a fejlesztők számára, hogy minimális erőfeszítéssel hozzáférjenek egy helyi fejlesztői szerverhez az Interneten keresztül. A szoftver az ngrok.com aldomainjére teszi a helyileg üzemeltetett webszervert, ami azt jelenti, hogy nincs szükség nyilvános IP-címre vagy tartománynévre a helyi gépen [24].

### 7.1. Domain és ngrok

A „tarhely.eu” oldalon előfizettem egy domain-re, ahová feltöltöttem a teljes, kész adatbázisomat és backend fájlaimat. A „connection.php”-ban levő négy változót kellett módosítanom ahhoz, hogy a Domain adatbázisrendszer és annak backend-je teljesen üzemképes legyen, míg az applikáció esetén a „config.json”-be elmentett URL-t módosítottam a Domain URL-re, hogy bárholnan elérhető legyen. Minden olyan backend fájlt, amely token-t kezel (például 18. ábra), módosítanom kellett a token header lekérési módján. Az `apache_request_headers()` funkció alapján létrehozott `$headers['token']` header változót egyszerűen csak át kellett írnom `$_SERVER['HTTP_TOKEN']`-re. A két Arduino programban is módosítanom kellett a GET és POST Request címeket. A kártyaolvasó-rendszer programját illetően módosítottam mindkét GET Request-en a `File > Examples > HttpClient > SimpleHttpClient` példakódban szereplő GET Request kódrészlet egészével.

A Video View viszont még így sem működne az applikációban, mert az még mindig csak lokálisan érhető el. Amikor leprogramoztuk és Serial Monitorban megnyitottuk a kamerarendszert, akkor ott mindig kiírt egy ESP32 IP címet. Ez esetben mindig egy statikus IP volt (192.168.0.152). Az ngrok főoldalára beeregisztrálva, majd bejelentkezve a „Your Authtoken” menüpontnál látható a saját Authtoken-ünk, amely szükséges az IP cím csatorna létrehozásához [20]. A főoldaltól az ngrok alkalmazás letöltését, majd futtatását követően a megjelenő terminálba írjuk az alábbi utasításokat:

```
ngrok authtoken „Saját-Auth-Token”;
```

```
ngrok http „IP-Cím”; esetenben 192.168.0.152
```

Ezen két parancs után megjelenik majd egy ablak, ahol a két web-címen elérhető a Video Streaming. A „https” linket kimásolva az applikációban vagy adatbázisban a rendszerkezelő hozzá tudja majd adni, hogy ez a link melyik Arduino-hoz tartozzon. Ezáltal csak olyan felhasználók esetén jelenik meg a Video Streaming, akik olyan „rfid\_id”-val rendelkeznek, amely ajtónyitási státusza az engedélyezett az adott Arduino ID-re.

Az ingyenes ngrok-nak annyi hátlütője van, hogy egy eszköznek futnia kell ahhoz, hogy egy IP cím elérhető legyen, és egy eszközről csak egy csatorna létesíthető. Az alkalmazás bezárása és újból való megnyitása esetén új IP címet fog hozzárendelni.

## 7.2. A kész projekt elérhetővé tétele

Szakedolgozatomat és a kész projektemet GitHub-on tettem elérhetővé. A Repository-kat az alábbi Terminál parancsokkal hoztam létre [25]:

1. `git init`
2. `git add .`
3. `git commit -m „message”`
4. `git remote add origin https://github.com/user/new_repository.git`
5. `git branch -M main`
6. `git push -u origin main`

Négy Repository-t készítettem, amelyekbe külön a React Native fájlokat, a backend fájlokat, a szakedolgozatban használt képeket és azok forrásait, valamint az Arduino programokat helyeztem el [26].

A Szakedolgozat\_Backend Repository-ba hozzáadtam még egy `.gitignore` fájlt, amelyben megadtam, hogy a „connection.php” fájlt soha ne push-olja, mert ebben szerepelnek az adatbázisom eléréséhez szükséges adatok. A git-en látható „connection.php”-t sablon szerűen feltöltöttem. A `.gitignore` létrehozásához először egy `gitignore.txt`-t hoztam létre, amelyet a Terminálban a `ren gitignore.txt .gitignore` paranccsal átneveztem. Ezt követően a `git add .gitignore` utasítással hozzáadtam, majd a `git commit -m „.gitignore” .gitignore` utasítással committáltam, amelyet követve pusholtam.

## 8. Összefoglalás

Szakedolgozatomban egy olyan Android applikáció fejlesztését mutattam be, amely egy RFID-s kártyaolvasó-rendszeren és egy ESP32-CAM Board-dal működő kamerarendszeren alapul. A második és harmadik fejezetben hardveres és szoftveres előkészületekről írok.

A szakedolgozatom második fejezetében a hardverek fő alkotóelemeit mutattam be. Itt kitérek az Arduino Mega 2560 R3, az ESP32-CAM AI-Thinker, az Ethernet Shield R3 eszközökre, illetve a HC-SR501 PIR mozgásérzékelő modulra. Általánosan bemutattam ezen eszközök fontosabb tulajdonságait. Az RFID-ről és az FT232RL FTDI programmer-ről csak említést tettem, de ezek is részét képezik a projekt megvalósításához.

A harmadik fejezetben bemutattam a fő szoftvereket, amelyeket szakedolgozatom megvalósításához használtam. Említést tettem az Arduino IDE, XAMPP, Fritzing és React Native szoftverekről, valamint a PHP-ről, amit a backend fejlesztéséhez használtam, mint szkriptnyelvet. Ha bonyolultabb feltelepítési módot igényeltek, akkor azok feltelepítését és beüzemelését is részleteztem. A PHP fejezetben belül megemlítettem még a Postman alkalmazást, amelyet a backend és frontend HTTP kérések tesztelésére használtam.

Ezt követően a kártyaolvasó-rendszer és kamerarendszer összeállításának fontosabb lépéseiről írok. Az itt látható végleges kapcsolási rajzokat a Fritzing alkalmazás segítségével készítettem el. Említést tettem az Arduino Board típusoknál eltérő SPI Pin-ekről, amelyek némi különbséget eredményezhetnek huzalozás terén. A könnyebb átláthatóság érdekében táblázatos formában kiírtam a Pin-ek huzalozását.

A hardveres rész összeállítása után írtam ezen két hardver programozásának a fontosabb lépéseiről, valamint az adatbázis létrehozásáról. Itt az adatbázis még lokális hálózaton van, amelyen csak akkor változtattam, amikor már minden megfelelően működött (hetedik fejezet).

A hatodik fejezetben részleteztem az Android applikáció fontosabb elemeit, oldalait és azok működését. A képeken a kész projekt felhasználói felülete látható.

A hetedik fejezetben a lokális hálózaton levő összes fájlt áthelyeztem lokális hálózatról egy Domain-re, szükség esetén módosítottam, majd a kész projektet elérhetővé tettem az interneten.

A projekt további fejlesztésére rengeteg lehetőség van, mind frontend és backend terén, valamint a hardverek működésén. Említésképp a kamerarendszert illetően igencsak hasznos lenne az ngrok lecserélése egy praktikusabb módszerre, az ajtózárszerkezet esetén egy websocket implementálása jobb kommunikáció megteremtése érdekében, stb.

## 9. Irodalomjegyzék

- [1] Arduino Mega 2560 Rev3 Overview:  
<https://store-usa.arduino.cc/products/arduino-mega-2560-rev3>  
Megtekintés ideje: 2021.09.25.
- [2] Getting started with the ESP32-CAM  
<https://dronebotworkshop.com/esp32-cam-intro/>  
Megtekintés ideje: 2021.09.27.
- [3] DEBO CAM ESP32 Developer boards – ESP32 camera, OV2640  
<https://www.reichelt.com/ro/en/developer-boards-esp32-camera-ov2640-debo-cam-esp32-p266036.html>  
Megtekintés ideje: 2021.09.27.
- [4] ESP32-CAM: Machine Vision Tips, Camera Guides and Projects:  
<https://www.arducam.com/esp32-machine-vision-learning-guide/>  
Megtekintés ideje: 2021.09.27.
- [5] ESP32-CAM Specifications and Features:  
<https://makeradvisor.com/esp32-cam-ov2640-camera/>  
Megtekintés ideje: 2021.09.27.
- [6] Arduino Ethernet Shield R3 (V1) Overview:  
<https://www.arduino.cc/en/Main/ArduinoEthernetShieldV1>  
Megtekintés ideje: 2021.09.26.
- [7] How HC-SR501 PIR Sensor Works & Interface It With Arduino  
<https://lastminuteengineers.com/pir-sensor-arduino-tutorial/>  
Megtekintés ideje: 2021.10.02.
- [8] Arduino Software (IDE):  
<https://www.arduino.cc/en/Guide/Environment>  
<https://www.arduino.cc/en/software>  
Megtekintés ideje: 2021.10.03.
- [9] PHP:  
<https://www.php.net/manual/en/intro-what-is.php>,  
<https://www.php.net/manual/en/faq.installation.php>  
Megtekintés ideje: 2021.10.03.

- [10] What is Postman API Test?:  
<https://www.encora.com/insights/what-is-postman-api-test>  
Megtekintés ideje: 2021.10.03.
- [11] XAMPP:  
<https://www.apachefriends.org/index.html>,  
<https://www.javatpoint.com/xampp>  
Megtekintés ideje: 2021.10.03.
- [12] Abstract: What is Fritzing?  
<https://fritzing.org/media/uploads/publications/FritzingInfoBrochure-highRes.pdf>  
Megtekintés ideje: 2021.09.30.
- [13] What is React Native?  
<https://www.netguru.com/glossary/react-native>  
Megtekintés ideje: 2021.10.03.
- [14] Setting up the development environment:  
<https://reactnative.dev/docs/environment-setup>  
Megtekintés ideje: 2021.09.20.
- [15] Security Access Using RFID Reader:  
<https://create.arduino.cc/projecthub/Aritro/security-access-using-rfid-reader-f7c746>  
Megtekintés ideje: 2021.10.05.
- [16] A Brief Introduction to the Serial Peripheral Interface (SPI):  
<https://www.arduino.cc/en/reference/SPI>  
Megtekintés ideje: 2021.10.05.
- [17] ESP32-CAM Post Images to Local or Cloud Server using PHP (Photo Manager)  
<https://RandomNerdTutorials.com/esp32-cam-post-image-photo-server/>  
Megtekintés ideje: 2021.10.05.
- [18] Arduino UniqueID:  
<https://forum.arduino.cc/t/uniqueid/671432/4>  
Megtekintés ideje: 2021.10.02.
- [19] Installing the ESP32 Board in Arduino IDE (Windows, MAC OS X, Linux):  
<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>  
Megtekintés ideje: 2021.10.05.

- [20] How to Access ESP32-CAM Worldwide using ngrok:  
<https://create.arduino.cc/projecthub/CiferTech/how-to-access-esp32-cam-worldwide-using-ngrok-210aa0>  
[https://github.com/cifertech/ESP32CAM\\_ngrok/blob/main/ESP-CAM\\_CLOUD.ino](https://github.com/cifertech/ESP32CAM_ngrok/blob/main/ESP-CAM_CLOUD.ino)  
Megtekintés ideje: 2021.10.05.
- [21] ESP32-CAM Camera Boards: Pin and GPIOs Assignment Guide:  
<https://randomnerdtutorials.com/esp32-cam-camera-pin-gpios/>  
Megtekintés ideje: 2021.10.05.
- [22] Discord Security Camera with an ESP32:  
<https://create.arduino.cc/projecthub/WillMakesTV/discord-security-camera-with-an-esp32-7c4621>  
<https://github.com/WillMakesTV/discord-spycam/blob/main/discord-spycam.ino>  
Megtekintés ideje: 2021.10.06.
- [23] How to Decrypt MD5 Password in PHP?:  
<https://www.edureka.co/blog/decrypt-md5-password-PHP/>  
Megtekintés ideje: 2021.10.09.
- [24] What is ngrok?:  
<https://www.pubnub.com/learn/glossary/what-is-ngrok/>  
Megtekintés ideje: 2021.11.01.
- [25] Start a new git repository:  
[https://kbroman.org/github\\_tutorial/pages/init.html](https://kbroman.org/github_tutorial/pages/init.html)  
Megtekintés ideje: 2021.11.06.
- [26] Szakdolgozat GitHub:  
[https://github.com/skazalien/Szakdolgozat\\_Files](https://github.com/skazalien/Szakdolgozat_Files),  
[https://github.com/skazalien/Szakdolgozat\\_Arduino](https://github.com/skazalien/Szakdolgozat_Arduino),  
[https://github.com/skazalien/Szakdolgozat\\_ReactNative](https://github.com/skazalien/Szakdolgozat_ReactNative),  
[https://github.com/skazalien/Szakdolgozat\\_Backend](https://github.com/skazalien/Szakdolgozat_Backend)  
Létrehozás ideje: 2021.11.12.

# Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Kocsis Gergelynek a szakdolgozatom írása alatt nyújtott folyamatos segítségéért, tanácsaiért és ötleteiért.