



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI INTÉZET INTÉZET



SZAKDOLGOZAT



OE-KVK
2020.

Deák Gergely
T007186/FI12904/K



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
MŰSZERTECHNIKAI ÉS AUTOMATIZÁLÁSI INTÉZET



SZAKDOLGOZAT FELADATLAP

Hallgató neve: Deák Gergely

Törzskönyvi száma: T007186/FI12904/K

Szak, specializáció: villamosmérnök, műszer-automatika

A dolgozat címe: Távolról is elérhető, jogosultság alapján nyitható értékmegőrző szekrény modellezése.

A dolgozat címe angolul: Modelling remotely accessible, authorization-based safety deposit boxes.

Feladat részletezése: Értékmegőrző szekrények elektronikus nyitása, kezelése. A szekrények távoli elérése jogosultság alapján. Eseménynapló készítése a használt funkciók alapján.

Intézményi konzulens neve: Sándor Tamás mestertanár

Külső konzulens neve

Munkahelye:

A kiadott téma elévülési határideje: a kiadás évétől számított 3 év

Beadási határidő: 2020. december 15.

A záróvizsga tárgyai:

Automatika I.

Beágyazott rendszerek

Automatizált gyártórendszerek

A szakdolgozat titkos/nem titkos

Kiadva: Budapest, 2020. október 31.

A dolgozatot beadásra alkalmasnak találom:



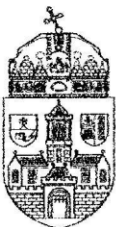
intézet igazgató

2020. 11. hó 29. nap

20..... hó nap

belső konzulens

külső konzulens



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR
MŰSZERTÉCHNIKAI ÉS AUTOMATIZÁLÁSI INTÉZET



HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat/diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatban/diplomamunkában található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Budapest, 20.12.01.....


.....
hallgató aláírása

1. Tartalomjegyzék

1.	Tartalomjegyzék.....	1
2.	Bevezetés	3
2.1.	A feladat pontos leírása.....	3
2.2.	A feladat célja	3
2.3.	Működési követelmények.....	3
2.3.1.	Adminisztrátori és felhasználói szerepkörök	4
3.	Hasonló megoldások másoktól	6
4.	Feladat megvalósításának lehetőségei.....	7
4.1.	Ajtózár lehetőségek	7
4.2.	Közei vezérlés lehetőségei	8
4.3.	Távoli elérés lehetőségei.....	11
5.	Megoldás kiválasztása modellezéshez.....	13
6.	A modellezés megvalósításhoz használt eszközök/technológiák és kapcsolódásuk	14
6.1.	Mikrokontroller	14
6.2.	Ajtózár	14
6.3.	Mátrixbillentyűzet	15
6.4.	LCD kijelző	15
6.5.	Számítógép	16
6.6.	Programok és kapcsolódásuk	17
7.	Mikrokontrolleres program	20
7.1.	Használt perifériák és driverek.....	20
7.2.	Fontos változók, függvények és a program felépülése.....	21
8.	Számítógépes program.....	25
8.1.	Adatbázis	25
8.2.	Adatbázis kapcsolat.....	29
8.3.	RS485 osztály	29
8.4.	MVC osztályai és kapcsolódásuk	29
8.4.1.	Controller osztályok és metódusaik.....	30
8.4.2.	Model osztályok	36
8.4.3.	View	38

9.	Kapcsolat.....	42
10.	Felhasználói működés, útmutató.....	46
10.1.	Helyi használat.....	46
10.1.1.	Felhasználói útmutató.....	46
10.1.2.	Adminisztrátori útmutató.....	49
10.2.	Távoli használat.....	50
10.2.1.	Felhasználói útmutató.....	50
10.2.2.	Adminisztrátori útmutató.....	53
11.	Tesztelés és hibakeresés	57
12.	Fejlesztési lehetőségek.....	58
13.	Összefoglalás	59
14.	Summary	60
15.	Irodalomjegyzék	61

2. Bevezetés

2.1. A feladat pontos leírása

A szakdolgozatom célja egy olyan értékmegőrző szekrényrendszer, szekrénycsoport modelljének az elkészítése, melynek adott szekrényei közvetlen közelről, valamint távolról is nyithatóak és zárhatóak. A szekrénycsoporton belüli szekrények különálló egységeket alkotnak. Ezen szekrények zárjai egy közös vezérlő egységhez vannak hozzárendelve. Az értékmegőrző szekrénycsoport szekrényei jogosultsághoz kötötten az imént említett vezérlő egységen keresztül, valamint valamilyen, a későbbiekben tárgyalt eszközön keresztül nyithatóak és zárhatóak közeli, illetve távoli használat esetén. A záruk állapotainak változásai eseményszerűen, időhöz és dátumhoz rendelve rögzítésre kerülnek.

2.2. A feladat célja

A feladat célja egy olyan értékmegőrző rendszer felépítése, melyet a felhasználók megbízható, biztonságos, kényelmes, valamint felhasználóbarát programokon keresztül vehetnek igénybe. Cél, hogy a felhasználóknak kizárólagos hozzáférési jogaik legyenek a szekrényeikhez és így a szekrényekben őrzött értékeikhez csak általuk lehessen hozzáférni. A távoli nyitás és zárás azt a célt szolgálja, hogy az adott szekrények felett rendelkező felhasználó úgy tehesse lehetővé mások számára a saját szekrényeinek a tartalmának az elérését, hogy ne kelljen megosztania az illetővel a szekrényhez tartozó kezelési információkat.

2.3. Működési követelmények

A rendszerrel szemben állított működési követelmények felsorolása:

- Elvárás a megvalósítással szemben az, hogy a zár rendszerek szekrényenként különböző kóddal legyenek nyithatóak a felhasználók által.
- Lényeges, hogy a felhasználóknak lehetőségük legyen a hozzájuk rendelt szekrények kódjainak a megváltoztatására.
- Szükség van olyan, több jogosultsággal ellátott felhasználóra (adminisztrátorra), akinek jogában áll az, hogy szekrényeket rendeljen hozzá az új felhasználókhoz, valamint az, hogy a szolgáltatást már nem igénybe vevő felhasználókat megfosszon a szekrények használati jogaitól.

- A rendszernek könnyen kezelhetőnek kell lennie a felhasználók, valamint az imént említett adminisztrátorok számára is közeli, illetve távoli elérés esetén is.
- Törekedni kell arra, hogy a távoli elérés a lehető legnagyobb hatósugarú legyen.
- Az eseménynaplónak tartalmaznia kell minden, a szekrényekhez kapcsolódó előzményt időhöz hozzárendelve.
- Az eseménynaplóhoz kizárólag az adminisztrátorok férhetnek hozzá.
- Minden szekrényhez egyetlen felhasználó lehet csak hozzárendelve, viszont egy felhasználóhoz több szekrény is tartozhat.
- Áramszünet esetén a rendszernek meg kell őriznie a szekrények kinyitásához szükséges adatokat, valamint nyitott állapotra kell váltaniuk a zárnak az őrzött értékek elérhetőségeinek érdekében.
- Áramkimaradás után a rendszernek a lehető leggyorsabban vissza kell állnia az áramszünet előtti állapotba.
- A távoli eléréshez szükséges program egy személyi számítógépen fut, ennek tápellátása a hálózaton keresztül történik az ismert módon. Külön 12V-os és 5V-os tápellátásra van szüksége a zárnak, és a vezérlő egységnek, ezeket biztosítani kell a rendszerhez.
- Törekedni kell arra, hogy a rendszer minél gyorsabb működést biztosítson a felhasználók számára.
- Lehetőleg bármilyen hiba fellépését jelezze a rendszer az adminisztrátornak vagy a felhasználónak.

2.3.1. Adminisztrátori és felhasználói szerepkörök

A rendszerben a felhasználói szerepkörbe tartoznak a szerkény nyitás, zárás, első használat és kód módosítás funkciók.

Távoli használat esetén a felhasználó egy bejelentkezési felületen keresztül tud belépni egy platformra, ahol kezelheti a hozzá tartozó szekrényeket. Bejelentkezéshez szükséges felhasználónevet és jelszót csak az adminisztrátor rendelhet hozzá egy felhasználóhoz. A felhasználó az első sikeres bejelentkezés után megváltoztathatja a bejelentkezéshez szükséges jelszavát.

Szerkénykezelés esetén az első használat akkor fordul elő, ha az adminisztrátor felszabadít egy adott szekrényt. Ilyenkor a felhasználó egy tetszőlegesen beírt kóddal teszi

használhatóvá az adott szekrényt. Ez a szekrény a későbbiekben adminisztrátori beavatkozásig vagy felhasználói kód módosításáig az először megadott kóddal lesz elérhető.

A rendszerben az adminisztrátori szerepkörbe tartoznak a szekrény zárolás, zárolás feloldás és felszabadítás funkciók, valamint a felhasználók és az előzmények kezelése.

Távoli használat esetén az adminisztrátor egy bejelentkezési felületen keresztül tud belépni egy platformra, ahol kezelheti a szekrényeket, hozzáférhet az előzményekhez, létrehozhat új felhasználói fiókokat és a meglévő fiókokhoz szekrényeket rendelhet vagy vonhat meg.

Közeli használat esetén az adminisztrátor csak a szekrénykezelési funkciókat éri el.

Egy szekrény zárolása elérhetetlenné teszi az adott szekrényt, addig amíg az adminisztrátor fel nem oldja a zárolást.

A felszabadítás funkció „első használat” állapotba teszi a szekrényt. Ekkor az adott szekrényt a feljebb leírt módon helyezheti üzembe egy felhasználó.

3. Hasonló megoldások másoktól

Végeztem egy előzetes kutatást az internet segítségével olyan eszközök, rendszerek után, amik értékmegőrzéssel foglalkoznak és távolról is vezérelhetők. A kutatás során több olyan zárrendszerrel is találkoztam, ami lakások, illetve házak ajtajaihoz terveztek és távolról is nyithatóak és zárhatóak voltak okostelefon segítségével Wifi, Bluetooth, illetve NFC-s vezeték nélküli kommunikáció segítségével. Ezeket a termékeket önálló helyiségek zárolására alkalmazzák, nem alkalmasak több, kisebb egység távoli vezérlésére.

A keresés egyik eredményeként találtam rá a KeySafe kulcsszekrény családra, valamint ezen belül az AutoSafe és a KeyDrawer termékcsaládokra: „A KeySafe® egy intelligens kulcsszekrény család, amely szigorúan ellenőrzött kulcskezelést tesz lehetővé személyre szóló proximity kártyás, PIN kódos, mobiltelefonos (NFC) - opcionálisan ujjlenyomatos - nyitással. A KeySafe Lock egyesével reteszeli (lezárja) a kulcsokat és nem engedi elvinni, csak annak, akinek az adott kulcsra jogosultsága van. Felügyelheti, naplózhatja, hogy ki, mikor, melyik kulcsokat viszi el és hozza vissza. A kulcsszekrények számítógépre kapcsolhatóak, távolról lekérdezhetőek, menedzselhetőek, épületfelügyeleti rendszerbe integrálhatóak.” (KeySafe)

Értelmezésem szerint a KeySafe termékei lehetővé teszik a távoli felügyeletet, illetve az előzménykezelést, viszont nem foglalkoznak a széfek távoli nyitásával és zárásával.

A keresés egy másik eredményeként a ProxerLock 3-11 termékeket találtam: „A ProxerLock 3-11 kártyával / RFID proximity karkötővel nyitható intelligens ajtózár öltözőszekrények, csomagmegőrzők, értékmegőrző szekrények biztonságos zárására szolgál, teljes körű számítógépes monitorozással, és menedzseléssel. A zár RFID proximity olvasóval, mágneses zárral, nyitásérzékelővel, állapotjelzővel, és számítógép-hálózati interfésszel egybeépített eszköz.” (procontrol)

Az imént említett termékek wellness, fitness, fürdő és múzeumi értékmegőrző szekrényként vannak legtöbbször alkalmazva. Értelmezésem szerint a ProxerLock szintén lehetővé teszi a távoli felügyeletet, monitorozást, valamint a szekrények menedzselését, de nem foglalkoznak azzal, hogy lehetővé tegyék a felhasználók számára a távoli nyitást és zárást. Ezeket a rendszereket főként rövid idejű értékmegőrzésre alkalmazzák, olyan esetekben, mikor a felhasználó a szekrények közelében tartózkodik.

4. Feladat megvalósításának lehetőségei

A dolgozatomban ezen részben bemutatom, hogy milyen megvalósítási lehetőségeket dolgoztam ki. A feladatnak több olyan pontja is van, ahol több megvalósítási lehetőség vehető figyelembe, így ezeket a lehetőségeket részenként mutatom be és hasonlítom össze.

4.1. Ajtózár lehetőségek

A feladat megvalósításához mindenféleképpen valamilyen elektromos zárat kellett választanom, azért, hogy a vezérlő egység segítségével, vezérlő jelen keresztül nyitott, illetve zárt állapotba lehessen állítani a zárat. A zárok közül csak olyan jöhetett szóba, ami korlátlan ideig tudja tartani mindkét állapotát. Ezen feltételek alapján a következő lehetőségek adódtak:

➤ Feszültségre záró zár:

Ennek a megoldásnak előnye az, hogy áramszünet esetén a szekrények ajtajai automatikusan kinyílnak, így nem áll fent az a lehetőség, hogy a felhasználó nem tud hozzáférni az értékeihez.

➤ Feszültségre nyitó zár:

A felhasználók által használt szekrények a használati idő nagy részében zárva vannak. A nem használt szekrények állapota a felhasználók számára lényegtelen, így lehetnek zárt állapotban. Ez a megoldás költséghatékonyabbnak bizonyul fogyasztási szempontból, mert az idő nagy részében a zárat nem kell feszültség alatt tartani.

Összehasonlítás:

1. táblázat

	Előny	Hátrány
Feszültségre záró zár	Egyszerű megoldás áramszünet esetére	Fogyasztási szempontból költségesebb
Feszültségre nyitó zár	Fogyasztási szempontból költséghatékony	Áramszünet esetén biztosítani kell számára valamilyen külső erőforrást

4.2. Közeli vezérlés lehetőségei

Az ajtózárak közvetlen közeli nyitásához és zárásához szükség van valamilyen vezérlő egységre. Ehhez az egységhez csatlakoznak az ajtózárak, valamint olyan perifériák, amelyek segítségével a felhasználók jelezni tudják nyitási és zárási szándékukat és ennek sikerességéről visszajelzést kapnak. Ezen perifériákról egy későbbi fejezetben esik majd szó.

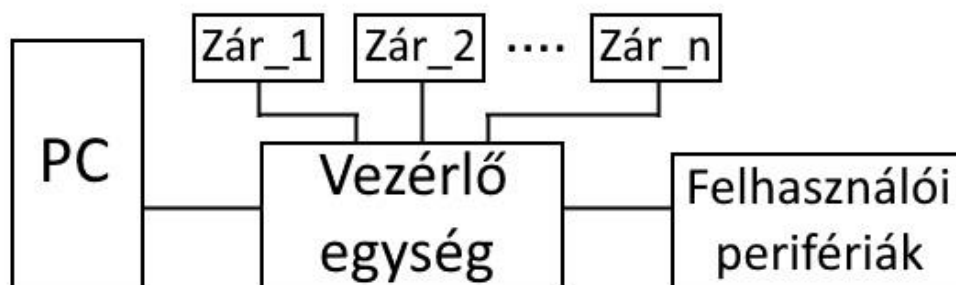
A vezérlő egységek és az ajtózárak eloszlására a következő lehetőségek merültek fel:

➤ Egy vezérlő kezeli az összes zárat:

Ebben az esetben minden szekrényajtót egy vezérlő egységen keresztül lehet közlelről elérni (1. ábra). Minden, a közeli nyitáshoz szükséges adat ezen a vezérlő egység memóriájában van tárolva. Ilyenkor a szekrényajtók számát korlátozza a vezérlő egység kimeneti pinjeinek a száma, valamint hátrány, hogy egyszerre csak egy felhasználó férhet hozzá közlelről a szekrényéhez. Egyedül ezen a vezérlőn fut a program, így a távoli elérésért felelős számítógéppel elég pont-pont kapcsolatot és kommunikációt kialakítani.

1. ábra

Illusztráció egy vezérlő esetén



Ezen megoldás alkalmazása olyan esetekben ajánlott, amikor viszonylag kevés szekrényt akarunk telepíteni és nem okoz problémát a felhasználók várakoztatása közlelérés esetén.

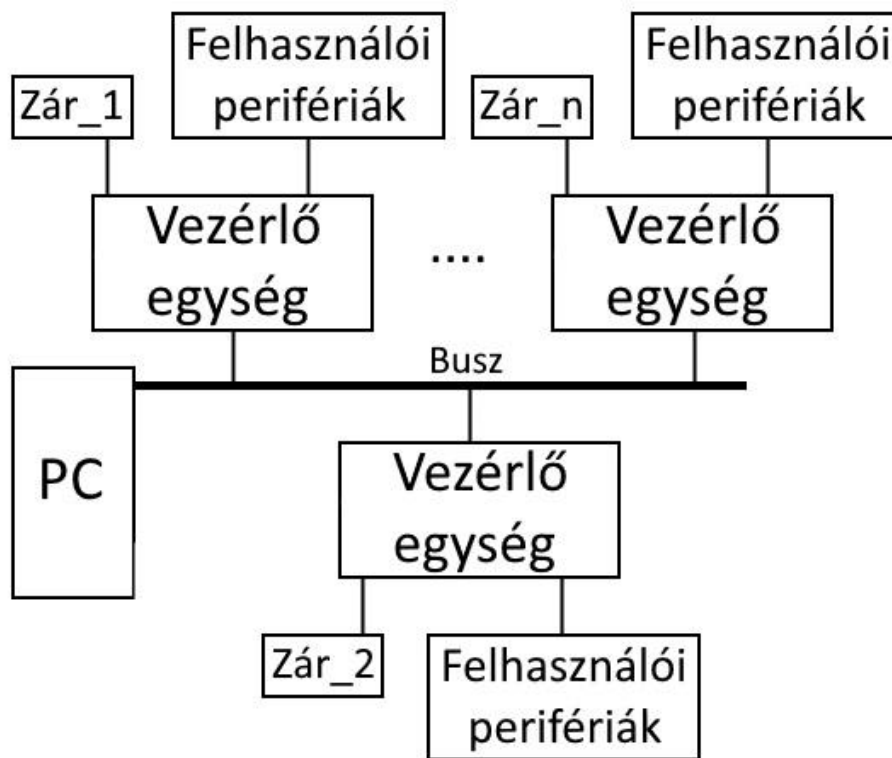
➤ Zárankén van egy vezérlő:

Ebben az esetben minden szekrényajtót külön vezérlő egységen keresztül lehet közlelről elérni. Ezen vezérlőknek csak a hozzájuk tartozó szekrény nyitásához és zárásához szükséges felhasználói adatokat kell tárolnia. Minden vezérlő rendelkezik felhasználói perifériákkal, így

lehetőség adódik arra, hogy a felhasználók egy időben akár az összes szekrényt egyszerre használhassák. Hátrány, hogy több vezérlő használata nagyobb fogyasztással jár, továbbá szükséges hozzá egy olyan buszrendszer, amin keresztül a szekrényeket vezérlő egységek kommunikálni tudnak a távoli elérésért felelős számítógéppel. Ilyenkor a szekrényajtók számát a buszrendszerre maximálisan fűzhető egységek száma határozza meg.

2. ábra

Illusztráció több vezérlő esetén



Ezen megoldás alkalmazása olyan esetekben ajánlott, amikor fontos, hogy a felhasználók várakozás nélkül, bármikor, azonnal igénybe vehessék a szekrényük közeli elérési használatát.

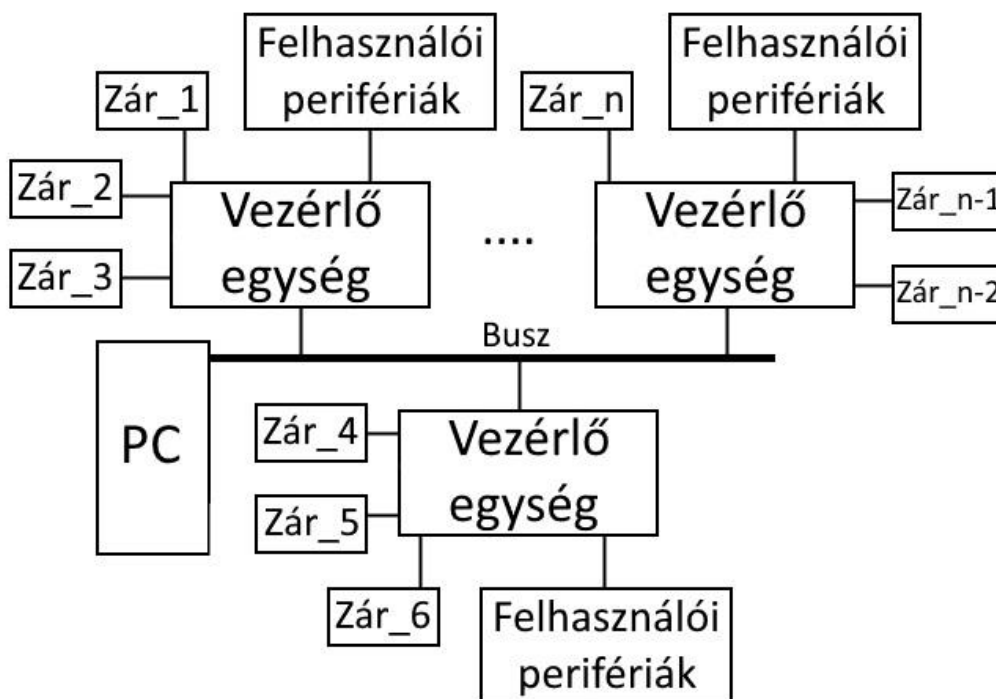
➤ Zár csoportonként van egy vezérlő:

Ez a megoldás az előzőekben leírt két megoldás alapján jött létre. Ebben az esetben adott számú szekrényajtónként van egy vezérlő egység (3. ábra). Így a vezérlő memóriájában az ezen vezérlőhöz tartozó szekrények információi kerülnek tárolásra. Mivel ennél a megoldásnál is több vezérlő egység van jelen a rendszerben, itt is valamilyen buszrendszeren keresztül kell

kapcsolatot létesíteni a vezérlő egységek és a távoli elérésért felelős számítógép között. Felhasználási szempontból egyszerre több felhasználó is tudja közelről használni a rendszert, viszont nem kezelhető egy időben az összes szekrény egyszerre. Ezen megoldás esetén a maximálisan telepíthető szekrények száma megegyezik a buszrendszerre fűzhető vezérlők és a vezérlőnkénti szabad kimeneti pinek szorzatával.

3. ábra

Illusztráció 3-mas szekrénycsoportok esetén



Ezen megoldás alkalmazása olyan esetekben ajánlott, amikor aránylag nagy mennyiségű szekrényt kívánunk telepíteni.

Összehasonlítás:

2. táblázat

	1) Egy vezérlő	2) Szekrényenkénti vezérlő	3)Szekrény-csoportonkénti vezérlő
Egy időben kezelhető szekrények száma	Egy	Korlátlan	Szekrénycsoportonként egy
Kommunikáció a számítógéppel	Pont-pont kapcsolat	Buszrendszer	Buszrendszer
Energiaigény	Alacsony	Magas	Közepes
Maximális szekrényszám	Megegyezik a vezérlő szabad kimeneti pinjeinek számával	Megegyezik a buszrendszerre fűzhető maximális egységek számával	A buszrendszerre fűzhető maximális egységek száma és egy vezérlő szabad kimeneti pinjeinek szorzata
Adatok tárolása	A vezérlő tárolja az összes szekrényhez tartozó adatot	Minden vezérlő egy szekrény adatait tárolja	A vezérlő a szekrénycsoporthoz tartozó adatokat tárolja
Költség	Alacsony	Magas	Közepes
Ajánlott használat	Kis forgalom és közepes szekrényszám esetén	Nagy forgalom és közepes szekrényszám esetén	Magas szekrényszám és közepes forgalom esetén

4.3. Távoli elérés lehetőségei

Az ajtózárok távoli nyitásához szükség van egy webalkalmazásra, amely képes kommunikálni a vezérlő egységekkel. Ehhez szükség van egy személyi számítógépre, amely csatlakozik a buszrendszerhez, vagy a vezérlő egységhez. A számítógép futtatja webalkalmazást és elérhetővé teszi a weblapot helyi hálózaton vagy a világhálón keresztül.

➤ Helyi hálózat megoldás:

Ezen megoldás esetén egy wifis router segítségével válik elérhetővé a weblap azon eszközök számára, amik csatlakoznak a helyi hálózatra, és alkalmasak valamilyen böngésző használatára. Az elérhetőség hatótávolsága a wifi jel hatótávolságától függ.

➤ Megosztás a világhálón:

Ebben az esetben, minden végberendezés, mely alkalmas a világhálózathoz való csatlakozásra, valamint valamilyen böngésző futtatására, elérheti a webalkalmazást korlátlan távolságról. A weblap megosztása a világhálón extra költséggel jár.

Összehasonlítás:

3. táblázat

	Megosztás helyi hálózaton	Megosztás a világhálón
Költség	Alacsony	Magas
Elérhetőség	Router wifi jel hatótávolsága	Bármilyen internet elérhetőségi helyről

5. Megoldás kiválasztása modellezéshez

A megvalósítási lehetőségek első pontjában tárgyalt feszültség hatására nyitó és feszültség hatására záró zárok közül a feszültség hatására záró zárat választottam, azért, mert fontosabbnak tartom az áramszünet esetén kialakuló biztonsági nyitás egyszerű lehetőségét, mint a rendszer fogyasztásának csökkentését.

A megvalósítási lehetőségek második pontjában tárgyalt lehetőségek közül azt a megoldást választottam a modellezéshez, amely szekrény-csoportonként tartalmaz egy vezérlő egységet. A választásomat az indokolja, hogy ez a megoldás magába foglalja az ezen pontban lévő másik két megoldás megvalósítását. Ezen lehetőség megvalósításával kevés hozzáadott munka segítségével át lehet alakítani a rendszert úgy, hogy valamelyik másik megoldás szerint működjön.

A megvalósítási lehetőségek harmadik pontjában tárgyalt lehetőségek közül a helyi hálózaton való megosztást választottam. Ennek indoka az, hogy az ezen pont alatt tárgyalt megoldások megvalósításai nem térnek el nagyon egymástól, viszont a modellezés szempontjából költséghatékonyabb a helyi hálózatos megoldás.

6. A modellezés megvalósításhoz használt eszközök/technológiák és kapcsolódásuk

Ebben a fejezetben részletesen bemutatom az elkészített modellhez használt eszközöket és termékeket, valamint ezeknek fizikai kapcsolatát. Továbbá megnevezésre kerülnek a rendszer belső működését leíró programok létrehozásához használt technológiák.

6.1. Mikrokontroller

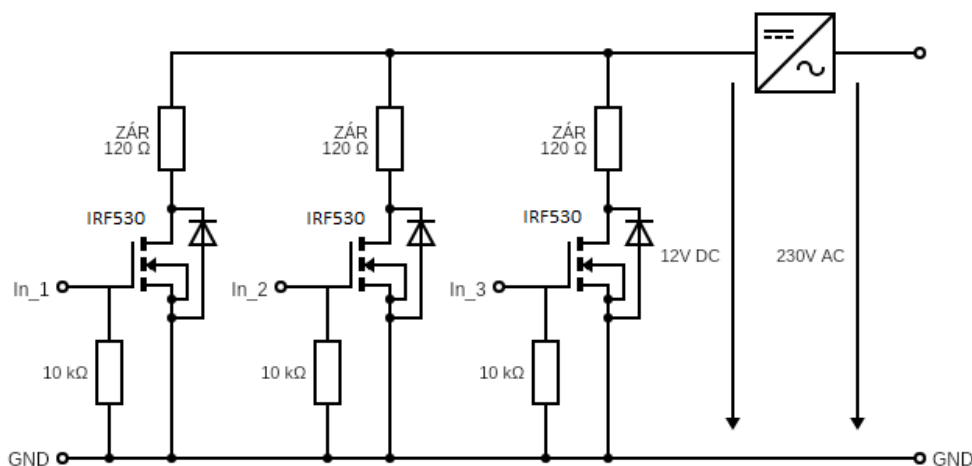
A feladat modellezéséhez Thunder-Bird 3 mikrokontrollereket választottam, mert sok szabad I/O porttal rendelkezik, létrehozható vele RS485 soros kommunikáció és könnyedén csatlakoztathatók rá felhasználói kezeléshez szükséges perifériák. Ezen mikrokontrollerek Atmega64-es processzorral rendelkeznek (Atmega64 datasheet). A mikrokontrollerekre töltött programokat az Atmel Studio 7 fejlesztői környezetet használva írtam meg C programozási nyelven. A mikrokontrollerek számára 5V-os tápellátást kell biztosítani.

6.2. Ajtózár

A szekrény zárolásához az YE-304NO feszültségre záró zárat választottam (YE-304NO datasheet). Ez egy kis méretű és könnyen beszerelhető elektromos zár, ami elektromágneses alapon működik és több mint 150kg tartóképeséssel rendelkezik. A zár számára 12V-os tápellátást kell biztosítani és zárt állapot esetén 100mA áramot vesz fel.

A zárok és a mikrokontroller kapcsolatát egy MOSFET-es kapcsoló segédáramkör segítségével alakítottam ki (4. ábra).

4. ábra
Kapcsoló áramkör a zárakhoz



Az áramkör az In₁, 2, 3 vezérlő jelekkel léphet működésbe. Az In₁, 2, 3 jelek 5V-os vagy 0V-os, a mikrokontroller által kiadott vezérlőjelek, magas állapotban az adott vezérlő jelhez tartozó MOSFET kinyit, így a zár áram alá kerül és működésbe lép. A 10k Ohm-os ellenállások biztosítják, hogy a MOSFET semleges vezérlőjel esetén se nyisson. A Gate és a vezérlőjel közé érdemes egy kb. 100 Ohm-os ellenállást elhelyezni a mikrokontroller védelmének érdekében.

6.3. Mátrixbillentyűzet

A mátrix billentyűzet egy 3x4 billentyűből álló billentyűzet, amely segítségével a felhasználó adatokat tud továbbítani a mikrokontroller felé. A billentyűzet 7 pinen keresztül csatlakozik a mikrokontrollerhez, és nincs szüksége külső energiaforrásra.

6.4. LCD kijelző

Az LDC kijelző lehetővé teszi a felhasználók számára az egyszerű kezelést. Ezen kijelző segítségével a program használati utasításokat és visszajelzést tud közölni a felhasználóval. A kiválasztott kijelző neve: Hitachi HD44780U. A kijelző 4x16 karakter megjelenítésére alkalmas egy időben, 7 pinen keresztül csatlakozik a mikrokontrollerhez és nincs szüksége külső energiaforrásra.

6.5. Számítógép

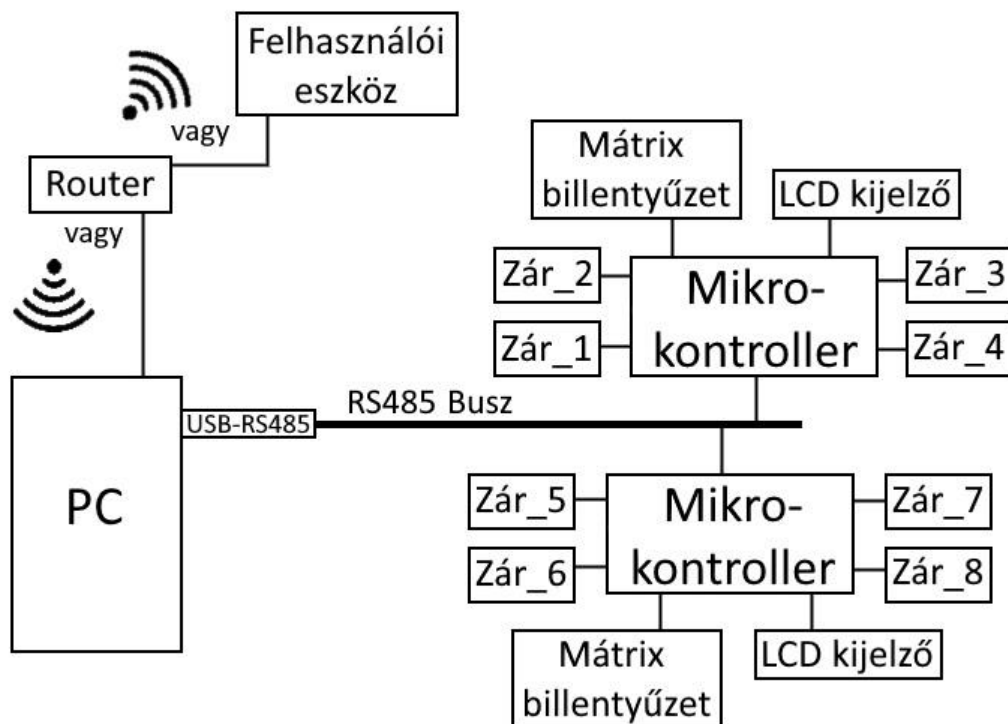
A távoli elérésért felelős webalkalmazás egy személyi számítógépen futtatható. A számítógép tápellátása a villamos hálózaton keresztül valósul meg. A számítógép és a mikrokontrollerek kapcsolata RS485 soros kommunikáción keresztül valósul meg egy USB átalakító segítségével. Azért választottam az RS485 kommunikációt, mert lehetővé teszi a számítógép és a szekrények (a mikrokontrollerek) közti nagy távolságot.

A számítógép és a helyi hálózatot kialakító router wifin vagy UTP kábelén keresztül csatlakozhatnak egymáshoz. A router és a távoli eléréshez használt eszközök ugyan ezen a módon csatlakozhatnak egymáshoz.

A webalkalmazást Visual Studio fejlesztői környezet használatával írtam meg C# programozási nyelven, valamint HTML hiperszöveges jelölőnyelven. A webalkalmazáshoz szükséges bejelentkezési adatok, valamint egyes szekrényekkel kapcsolatos adatok az MSSQL adatbáziskezelő szoftver által létrehozott adatbázisban vannak tárolva. Az adatbázis és a webalkalmazás összekapcsolása eredményeként a program kezelni tudja az adatbázisban lévő adatokat.

Az eszközök összesített kapcsolódása az 5. ábra ábrázolja.

5. ábra
Illusztráció a végső modellről



6.6. Programok és kapcsolódásuk

A fizikai felépítés megtervezése után a következő lépés a programok felépítésének, összekapcsolódásuknak a megtervezése. A programok működésének részletes leírását a következő fejezetben fejtem ki. Ennek az alfejezetnek a célja egy egyszerűsített logikai váz bemutatása a programrészek felépítéséről és összekapcsolódásáról.

A mikrokontrolleres program felépítéséhez első sorban az LDC kijelző, a mátrixbillentyűzet, valamint az RS485 kommunikációhoz szükséges drivereket írtam meg külön állományokban. A mikrokontrolleres fő programot két nagyobb részre osztottam. Az egyik rész a felhasználói programért felel, pontosabban kezeli a felhasználók által beírt adatokat, biztosítja a megjelenítést és tartalmazza a szekrényműveletekhez kellő logikát, valamint tárolja az ezekhez szükséges adatokat. A program másik nagyobb része a számítógéppel való kommunikációért felelős. Ez a programrész teszi lehetővé az adatok küldését és fogadását RS485 kommunikáción keresztül a számítógép felé, valamint az adatok kódolásáért és dekódolásáért is felel. A két programrész közötti kapcsolatot a mikrokontrolleren

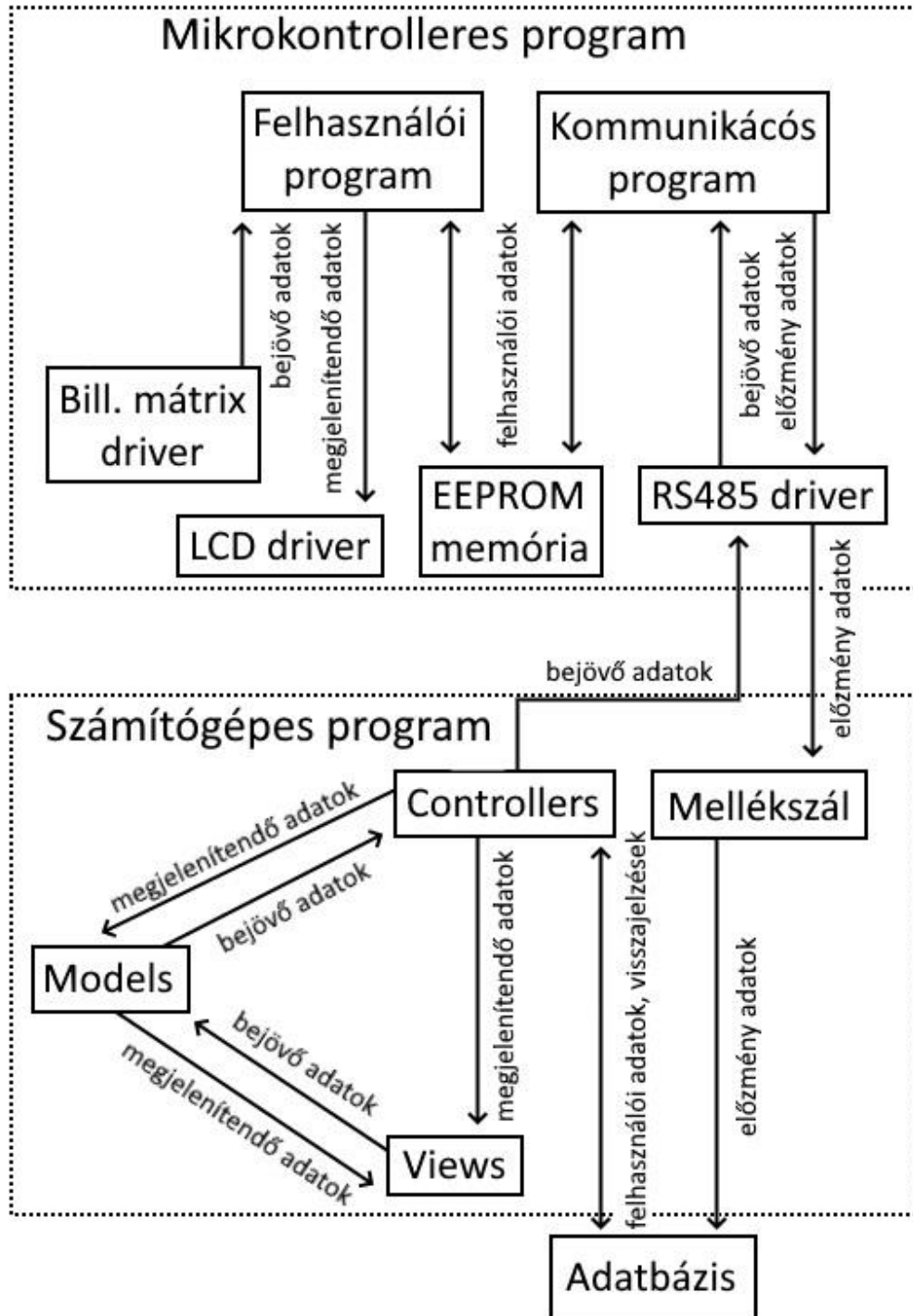
tárolt felhasználói adatok teremtik meg. Ezekről az adatokról biztonsági okok miatt a mikrokontroller EEPROM memóriájába folyamatos biztonsági mentés jön létre.

A webalkalmazás alapja egy relációs adatbázis, melyben a távoli felhasználáshoz szükséges felhasználói adatok, szekrényszámok, kapcsolatok, valamint előzmények vannak tárolva. Az adatbázis táblái a programból írhatóak és olvashatóak.

A számítógépes program egy két szálon futó program. A fő szál a webalkalmazás futtatásáért felel. A webalkalmazás az MVC (Model View Controller) programtervezési mintára épül. A felhasználók a View osztályok által megjelenített weboldalakon keresztül tudnak adatokat küldeni. Az elküldött adatok a Model osztályokon keresztül a Controller osztályokban kerülnek feldolgozásra, ahol a program az adatbázison keresztül ellenőrzi a felhasználók által beküldött adatokat, majd megjeleníti az eredményt szintén egy Model osztályon keresztül egy View osztály által. A mellékszál a mikrokontrolleres programmal való kommunikációért felel.

A programrészek kapcsolódásának a blokkvázlatát a 6. ábra mutatja.

6. ábra
 Illusztráció a programrészek összekapcsolódásáról



A következő fejezetekben bemutatom lépésről lépésre, hogy hogyan építettem fel a modellt. Részletezem a programok logikai felépítését, működését és egymással való kommunikációját.

7. Mikrokontrolleres program

A mikrokontrolleres programot C programozási nyelven írtam az Atmel Studio 7 fejlesztői környezetben. A fejlesztői környezetben Atmega64-es processzort kellett kiválasztani, hiszen a Thunder-Bird 3 mikrokontroller ezzel a processzor típussal van ellátva. Első lépésként a mikrokontrollerhez kapcsolódó perifériákhoz kellett meghajtó programrészeket írnom.

7.1. Használt perifériák és driverek

➤ Mátrix billentyűzet:

A mátrix billentyűzethez tartozó driver a billmatrix.h és billmatrix.c nevű állományokban van definiálva. A mátrixbillentyűzet működése szerint egy időpillanatban egyszerre egy sornyi billentyűről tudjuk megállapítani, hogy leütötte-e a felhasználó. Mivel a billentyűzet 4 sorral és 3 oszloppal rendelkezik, így egy időpillanatban a következő jelzésű billentyűk lenyomását tudjuk érzékelni: első sor: 1, 2, 3 jelzésű billentyűk, második sor: 4, 5, 6 jelzésű billentyűk, harmadik sor: 7, 8, 9 jelzésű billentyűk, negyedik sor: *, 0, # jelzésű billentyűk. A billentyűzet 7 pinen keresztül csatlakozik a mikrokontrollerhez: a PC6-3 pinek a mikrokontroller kimenetre állított pinjei. Ezen pinek magas, vagy alacsony szintjei határozzák meg azt, hogy az adott időpillanatban melyik sorról kívánunk adatot beolvasni. A PC2-0 pinek a mikrokontroller bemenetre állított pinjei. Ezen pinek azt határozzák meg, hogy az éppen aktív sor mely billentyűjét ütötte le a felhasználó. A meghajtott sor és a leütött billentyű alapján számokat rendeltem az adott jelzésű billentyűkhöz. Ezek alapján hoztam létre a billmatrix nevű függvényt, melynek bemeneti értéke a meghajtott sor számánál egyel kisebb szám, valamint a visszatérési értéke az adott időpillanatban nyomva tartott billentyű jelzésével egyenlő. Kivételes esetek, hogy a * jelzésű billentyű esetén a visszatérési érték 10, a # jelzésű billentyű esetén a visszatérési érték 11, valamint, ha nincsen leütött billentyű az adott időpillanatban, akkor a függvény visszatérési értéke 12.

➤ LCD kijelző:

Az LCD kijelzőhöz tartozó driver az LCD_init.h és LCD_init.c nevű állományokban van definiálva. Ezt a drivert a Hitachi HD44780U adatlapjának segítségével írtam meg (Hitachi HD44780U datasheet). A kijelző a PE7-4 pineken, valamint a PF3-1 pineken keresztül van csatlakoztatva a mikrokontrollerhez. A PF3-1 pineken keresztül tudjuk beállítani azt, hogy

éppen adatot vagy parancsot küldünk vagy adatot fogadunk a kijelzőtől. A PE7-4 pineken keresztül egy időben 4bit adatot lehet küldeni a kijelzőnek. Ezeken a pineken keresztül történik a parancs küldés, az adat küldés, valamint az adat fogadás. Parancsot küldve a kijelzőnek különböző módokat lehet beállítani. Ezek alapján hoztam létre az LCD_init függvényt, mely bemeneti értékei alapján elvégzi a kijelző alapbeállításait. Az én beállításaim szerint a kijelző 4 sorban tud megjeleníteni soronként 16 karaktert, melyek egyenként 5x8 pixelből állnak. A karakterek aláhúzását és a kurzor villogását kikapcsolt állapotba állítottam. Létrehoztam az LCD_goto és az LCD_Puts függvényeket, melyek segítségével a kurzort tetszőleges pozícióba állíthatom, valamint egy karaktertömb segítségével tetszőleges karaktereket jeleníthetek meg a kijelzőn.

➤ RS485 kommunikáció:

Az RS485 kommunikáció a Thunder-Bird 3 mikrokontroller integrált tulajdonsága. Az RS485 kommunikációhoz tartozó driver az rs.h és rs.c, valamint az UART.h és UART.c nevű állományokban van definiálva. A kommunikáció baud rátája 9600. Ennek indoka az, hogy a küldött és fogadott adatok nagyobb eséllyel érnek célba hiba nélkül, mint nagyobb ráták esetén. Természetesen a kis baud ráta következménye a kisebb adatátviteli sebesség is, de ez a projekt esetében nem lényeges a kis méretű adatok miatt. Az adatok küldése egy struktúra segítségével történik. Az adatok fogadása az UART1 megszakítás függvényében hajtodik végre.

7.2.Fontos változók, függvények és a program felépülése

A driverek megírása után a következő lépés a fő program felépítése volt. A következő fejezetben bemutatom a fő program fontos változóit, függvényeit, valamint ezek működését.

A fő program elején az include kulcsszóval meghívtam a használt könyvtárakat és az előző fejezetben tárgyalt drivereket. Ezek után definiáltam a program során használt függvényeket, valamint globális változókat.

Főbb globális változók:

adatbázis változó:

Ez a mikrokontrolleres program legfontosabb változótömbje. Az adatbázis egy 4x4-es 2 dimenziós 32 bites egész számú változókat tartalmazó tömb. Ez a tömb szolgál a szekrényszámok, valamint a szekrényekhez tartozó kódok és állapotok tárolására a következő módon:

A tömb második index szerinti nulladik elemei (adatbázis[x][0]) tárolják a szekrényszámokat. Ezek állandó értékek. Jelen esetben én mikrokontrollerenként 4 szekrényt alkalmazok, ezért a tömb létrehozásánál a tömb első dimenziójának hossza 4. Így az alkalmazott mikrokontrollerek egyikébe 1-től 4-ig, a másikba pedig 5-től 8-ig kerültek be a szekrényszámok a tömbbe.

A tömb második index szerinti első elemei (adatbázis[x][1]) tárolják a szekrényekhez tartozó kódokat. Egy szekrényhez egy felhasználói kód tartozik, ami maximum 6 számjegyből áll. A megadás módjait egy későbbi részben részletezem.

A tömb második index szerinti második elemei (adatbázis[x][2]) tárolják a hozzájuk tartozó szekrény nyitott vagy zárt állapotát. Ezt az állapotot a felhasználó tudja változtatni sikeres belépés esetén. A tömb ezen részei 0: nyitott vagy 1: zárt értékeket tartalmazhatnak.

A tömb második index szerinti harmadik elemei (adatbázis[x][3]) tárolják a hozzájuk tartozó szekrény kezelési állapotát. A tömb ezen részei a következő értékeket tartalmazhatják:

0: Az adott szekrény hozzá van rendelve egy felhasználóhoz.

1: A felhasználó érvénytelen kóddal próbált meg belépni az adott szekrényhez.

2: A felhasználó egymás után kétszer próbált meg érvénytelen kóddal belépni az adott szekrényhez.

3: A felhasználó egymás után háromszor próbált meg érvénytelen kóddal belépni az adott szekrényhez.

4: A szekrény adminisztrátor által zárolva van.

5: A szekrény adminisztrátor által felszabadított állapotban van.

Az egyes állapotok részletesebb jelentését a későbbiekben fejtem ki.

admin_kod változó:

Ez egy állandó 32 bites szám, ami az adminisztrátori belépéshez szükséges kódot tartalmazza. A kód maximum 8 karakter hosszú lehet és megváltoztatása csak a program átírásával lehetséges.

admin_zar változó:

Ez a változó tárolja az egymás utáni sikertelen adminisztrátori belépések számát. A változó 0 és 3 közötti értéket vehet fel.

Főbb függvények:main függvény:

A könyvtárak és állományok meghívása és a függvények és globális változók definiálása után a program a main függvény kezdőcímére ugorva kezdi meg a működését. Első lépésben meghívásra kerül az init nevű függvény, amiben olyan alapbeállítási parancsok hajtodnak végre, melyek feltétlenül szükségesek a program későbbi, folyamatos működéséhez. Ezek a parancsok teszik lehetővé a későbbiekben a mátrix billentyűzet, LCD kijelző, RS485 kommunikáció, mikrokontroller LED-jei és gombjai és egyéb bemeneti és kimeneti portok használatát.

Második lépésben meghívásra kerül az adatbázis_init nevű függvény.

adatbázis_init függvény:

Ebben a függvényben hajtodik végre a korábban tárgyalt adatbázis 2 dimenziós tömb feltöltése. A folyamat során a mikrokontroller az EEPROM memóriából másolja át a korábban beírt felhasználói adatokat a tömbbe. A program működése során minden az adatbázis tömbbe írt érték átmásolódik az EEPROM-ba, annak érdekében, hogy egy esetleges áramkimaradás esetén ne vesszenek el a felhasználói adatok.

menu függvény és belőle hívódó függvények:

A perifériák beállítása és az adatbázis tömb feltöltése után a felhasználói programrész következik. A felhasználói programrész egy végtelen ciklus keretében egymást meghívó és egymásba visszatérő függvényekből áll.

A menu függvény végzi el a belépési menüpontok megjelenítését és figyeli a menüpontok kiválasztásához szükséges gombok állapotát. A gombok állapotváltozásának hatására meghívódik a felhasználó_beletes vagy az admin_beletes függvény.

A felhasználó_beletes függvény felelős a felhasználói belépéshez szükséges adatok megjelenítéséért, valamint a felhasználó által bevitt adatok ellenőrzéséért egy külön függvényen keresztül.

A felhasználó_fiók függvény sikeres felhasználói belépés esetén hívódik meg. Ezen függvény feladata a felhasználói műveletek menüjének a megjelenítése és az egyes menüpontokhoz tartozó gombok állapotának a figyelése. Az adott gombok lenyomásával a felhasználó további függvények meghívása által kinyithatja, bezárhatja a szekrényét és megváltoztathatja az ahhoz tartozó kódot.

Az admin_beletes függvény felelős az adminisztrátori belépéshez szükséges adatok megjelenítéséért, valamint az adminisztrátor által bevitt kód ellenőrzéséért.

Az admin_fiók függvény sikeres adminisztrátori belépés esetén hívódik meg. Ezen függvény feladata az adminisztrátori műveletek menüjének a megjelenítése és az egyes menüpontokhoz tartozó gombok állapotának a figyelése. Az adott gombok lenyomásával az adminisztrátor további függvények meghívása által zárolhat, feloldhat vagy felszabadíthat szekrényeket.

Timer0 megszakítás függvény:

Ez a függvény 0.025 másodpercenként hívódik meg a Timer0 számláló megszakítás generálása miatt. Feladata a sikertelen belépési próbálkozások miatt növekvő adatbázis[x][3] tömbben, valamint az admin_zar változóban lévő érték 3 percenkénti csökkentése. (3 sikertelen belépési kísérlet esetén az adott szekrény vagy adminisztrátori fiók zárolódik bizonyos ideig). Továbbá ez a függvény dolgozza fel az RS485 kommunikáción beérkező adatokat és kezdeményezhet válaszküldést.

8. Számítógépes program

A számítógépes program alapja egy adatbázis. Az adatbázis közvetlen kapcsolatban áll a működési logikáért felelős programrésszel. A működési logikáért felelős programrész pedig kapcsolatban áll a megjelenítésért felelős programrésszel. A következő fejezetekben ezen részek működését és leírását tárgyalom.

8.1. Adatbázis

Az adatbázis feladata a webalkalmazáshoz szükséges felhasználói (bejelentkezési) adatok tárolása. További feladata az összes szekrény sorszámanak a tárolása, valamint az előzményadatok tárolása. Az adatbázis ezeken kívül tartalmaz még egy különleges kapcsolási táblát, amely a felhasználók és a hozzájuk rendelt szekrények kapcsolatát tárolja.

Az adatbázist az MS Sql nevű program segítségével valósítottam meg. A táblákat a SzakdogaSql nevű adatbázisban hoztam létre. A táblák részletes leírása:

Felhasznalok tábla:

Ez a tábla tárolja a bejelentkezéshez szükséges adatokat. Változó tábla, a program működése során kerülhetnek bele új sorok. A Felhasznalok nevű tábla 4 oszlopból áll, melyek a következők:

FelhasznaloId:

Ez az oszlop az elsődleges kulcs a táblában. Érték szerint egész számokat tartalmaz, melyek feladata a felhasználók sorszámozása, ezért az értéke automatikusan folyamatosan növekvőre van állítva.

Felhasznalonev:

Ebben az oszlopban tárolódnak a felhasználónevek. Érték szerint maximum 50 karakterből álló szöveget tartalmazhat.

Jelszo:

Ebben az oszlopban tárolódnak a jelszavak. Érték szerint maximum 50 karakterből álló szöveget tartalmazhat.

Adminjog:

Az ebben az oszlopban szereplő értékek határoznak arról, hogy az adott sorban lévő felhasználó rendelkezik-e adminisztrátori joggal. Érték szerint True vagy False lehet.

Szekrenyek tábla:

Ez a tábla tartalmazza sorban a szekrényszámokat, ami a modellezés esetében az 1-8 számú szekrényeket jelenti. A tábla konstans, a program működése során nem változik. A Szekrenyek nevű tábla 2 oszlopból áll, melyek a következők:

SzekrenyId:

Ez az oszlop az elsődleges kulcs a táblában. Érték szerint egész számokat tartalmaz, melyek feladata a szekrények sorszámozása, ezért az értéke automatikusan folyamatosan növekvőre van állítva.

Szekrenyszam:

Ebben az oszlopban tárolódnak a szekrényszámok. Érték szerint egész számokat tartalmaz.

Kapcsolat tábla:

Ez a tábla kapcsolja össze a felhasználókat az általuk használt szekrényekkel a Felhasznalok tábla FelhasznaloId, valamint a Szekrenyek tábla SzekrenyId oszlopok tagjai által. A tábla működése biztosítja azt, hogy csak létező felhasználót és létező szekrényt lehessen összekapcsolni. Továbbá úgy van beállítva, hogy egy felhasználót több szekrénnel is össze lehet kapcsolni, viszont egy szekrény csak egy felhasználóhoz tartozhat. Ez egy változó tábla, a program működése folyamán kerülhetnek bele új sorok és törölődhetnek a meglévők. A Kapcsolat nevű tábla 2 oszlopból áll, melyek a következők:

FelhasználóId:

Ez az oszlop a Felhasznalok tábla FelhasználóId oszlopában lévő értékeket tartalmazhatja.

SzekrenyId:

Ez az oszlop a Szekrenyek tábla SzekrenyId oszlopában lévő értékeket tartalmazhatja.

Elozmenyek tábla:

Ez a tábla tárolja a szekrényekkel végzett műveletek előzményeit. Ez egy változó tábla, a program működése folyamán kerülhetnek bele új sorok. Az Elozmenyek nevű tábla 6 oszlopból áll, melyek a következők:

Sorszam:

Ez az oszlop az elsődleges kulcs a táblában. Érték szerint egész számokat tartalmaz, melyek feladata az előzmények sorszámozása, ezért az értéke automatikusan folyamatosan növekvőre van állítva.

Szekrenyszam:

Ez az oszlop a Szekrenyek tábla Szekrenyszam oszlopában lévő értékeket, valamint 0 értéket tartalmazhat. 0 érték közeli belépés vagy kilépés esetén fordulhat elő.

Felhasznalonev:

Ez az oszlop a Felhasznalok tábla Felhasznalonev oszlopában lévő értékeket tartalmazhatja, valamint ismeretlen felhasználót vagy ismeretlen adminisztrátort. Ezen utóbbi értékek közeli nyitás esetén fordulhatnak elő.

Muvelet:

Ez az oszlop tartalmazza az adott szekrényen végrehajtott művelet nevét. Érték szerint lehet: Nyitás, Zárás, Első használat, Kód módosítás, Zárolás, Zárolás feloldás, Felszabadítás, Belépés, Kilépés.

Eredmeny:

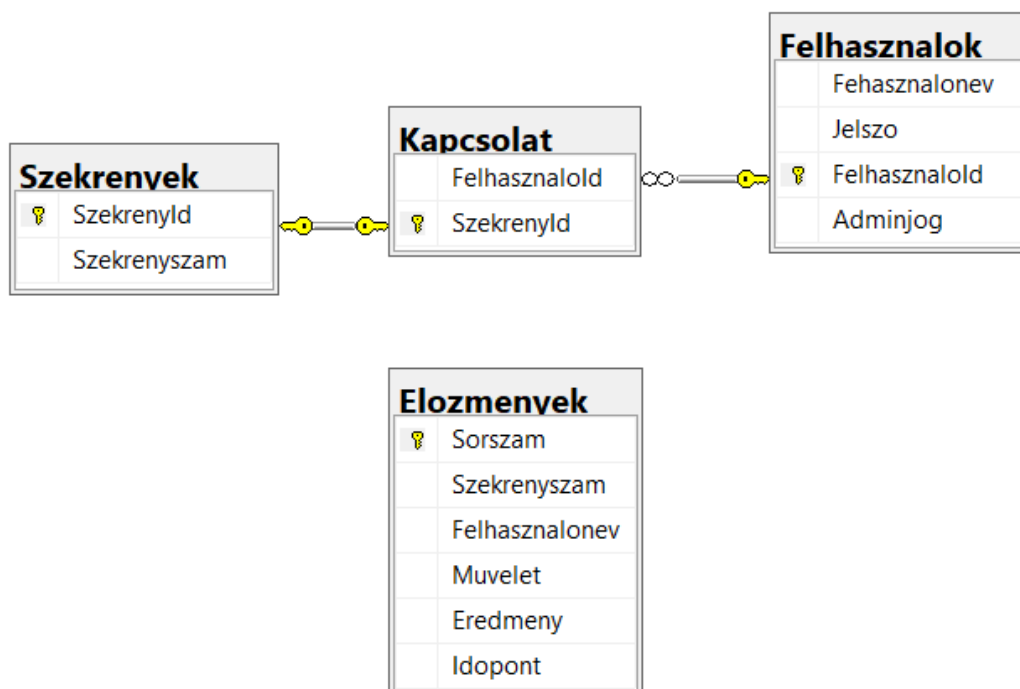
Ez az oszlop tartalmazza az elvégzett művelet eredményét. Érték szerint lehet: Sikeres vagy Sikertelen.

Idopont:

Ez az oszlop tartalmazza az elvégzett művelet pontos időpontját. Érték szerint időpontot tartalmaz évszám, hónap, nap, óra, perc és másodperc pontossággal.

A táblák közötti kapcsolatot a 7. ábra mutatja.

7. ábra
Adatbázis táblái és kapcsolataik



A számítógépes programot C# és HTML programozási nyelveken írtam meg Visual Studio fejlesztő környezetben. Első lépésben létrehoztam az adatbázis és a program közötti kapcsolatot.

8.2. Adatbázis kapcsolat

A program kapcsolatát az adatbázissal a SzakdogaSqlContext osztályban hoztam létre az adatbázis connectionstringjén keresztül. Minden, az adatbázisban létező táblához létrehoztam egy külön osztályt. Az osztályokon belül pedig az adott táblák oszlopai szerint létrehoztam változókat az oszlopok nevei és típusai szerint. A létrehozott osztályok lehetővé tették azt, hogy az általuk létrehozott objektumok által lekérdezhessem és módosíthassam az adatbázis tetszőleges tábláinak a sorait.

Második lépésben létrehoztam a számítógépes program és az RS485 buszrendszer közötti kapcsolatot. A felhasználói számítógép egy USB-RS485 átalakítón keresztül csatlakozik a buszrendszerre, így a programon belül a számítógép az egyik USB porton keresztül kommunikál.

8.3. RS485 osztály

Az RS485 osztályban létrehoztam a megfelelő USB porthoz tartozó soros kommunikációs kapcsolatot. A soros kommunikáció beállítását és alkalmazását a System.IO.Ports könyvtár tette lehetővé. A kapcsolat létrehozásához és helyes működéséhez a soros port beállításainak ugyan annak kellett lennie, mint a mikrokontrollereken beállított alaptulajdonságoknak. Ezek után az adatok küldése és fogadása egy objektumon keresztül valósult meg. A soros porton érkező adatokat lehetőség van többféleképpen figyelni. Közvetlen lekérdezés esetén a program addig várakozik, amíg nem érkezik adat a soros porton keresztül. Esemény figyelése esetén a program a fő programrész futásától függetlenül figyeli az érkező adatokat és adat érkezése esetén végrehajt tetszőleges utasításokat. A modellezéshez írt programban a közvetlen lekérdezést alkalmazom, amiért egy külön szál felel. A számítógépes program és a mikrokontrollerek közötti kommunikációt részletesebben a későbbiekben fejtem ki.

8.4. MVC osztályai és kapcsolódásuk

A számítógépes program további részeit az MVC programtervezési minta alapján hoztam létre. Jelen esetben a Model különböző osztályokat jelent, melyekben különböző, az adott feladathoz szükséges változók vannak definiálva. A Model azon osztályok sokasága, melyeknek a példányai megteremti a kinézet (View) és a működési logika, másnéven vezérlő (Controller) közötti kapcsolatot. A View a megjelenésért felel. A programon belül a Views

mappában találhatóak a webalkalmazás különböző oldalainak a kinézetét leíró programrészei. A felhasználóknak a program működése során a böngésző ezeket a programrészeket jeleníti meg egy másik, felhasználóbarát nézetben. A böngésző által megjelenített esetleges beviteli mezőkbe írt adatokat a kinézetért felelős program Model osztályokon keresztül juttatja el a Controller-hez. A Controller felelős a Model osztályokon keresztül beérkező adatok feldolgozásáért. Az adatok feldolgozása után a Controller Model osztályokon keresztül visszatér a View-hoz, ami megjeleníti a felhasználók számára az eredményeket.

Ebben a fejezetben gyakran elő fog fordulni a Session és TempData kifejezések. Ezek jelentése, rövid leírása a következő:

Session:

Ez egy speciális azonosító, amely kliens és szerver oldalon is elérhető. Jelen esetben Sessionben kerül tárolásra a felhasználó azonosítója és ezen keresztül történik a bejelentkezés ellenőrzése.

TempData:

Ez egy speciális változó, mely ideiglenesen tárolja az adatokat, és egy érték lekérése után automatikusan eltávolítja azokat. Jelen esetben a TempData egyszeri üzenetek megjelenítésére szolgál. A TempData funkciója, hogy Controller által létrehozott adatot csak egyszer továbbítja a View felé.

8.4.1. Controller osztályok és metódusaik

A Controller osztályok a Controller mappa elemei. A Controller osztályokban különböző metódusok találhatóak. Ezekben a metódusokban van megírva a működési logika. Működéstől függően ezek a metódusok visszatérhetnek másik metódusba vagy hozzá tartozó View osztályba. Ebben a fejezetben kifejtem részletesen az egyes Controllerekben lévő metódusok funkcióit. Előfordul, hogy egy metódusnak, több azonos nevű megfelelője is van. Ezen megfelelőknek külön funkciójuk van, így ezeket külön részletezem a leírásban. Az alkalmazásban 3 Controller osztály van jelen. Az IndexController a bejelentkezésért felelős. A FelhasznaloController a felhasználói tevékenységekért felelős. Az AdminController pedig az adminisztrátori tevékenységekért felelős.

IndexController metódusai:Login (get):

A bejelentkezéskor értéket kapó Sessionök értékét nullra állítja, majd visszatér a Login View-ba. Minden, a továbbiakban tárgyalt metódus ebbe a metódusba tér vissza hibás Session adat tárolása esetén.

Login (post):

Bejelentkezésért felelős metódus. Ellenőrzi a bemeneti LogininputModel által átadott bejelentkezési adatokat az adatbázis kapcsolaton keresztül. Helyes adatok esetén értéket ad az adott Sessionnek és visszatér a Felhasznalo vagy AdminControllerbe. Helytelen adatok esetén valamilyen hibaüzenethez tartozó értéket ad az adott TempData változónak és visszatér a Login get metódusába. (A TempData értékét később a View dolgozza fel)

4. táblázat

Név:	Request:	Bemeneti Model:	Kimeneti Model:	Visszatérés:
Login	Get	-	-	Login (V)
Login	Post	LoginInput(M)	-	Felhasznalo(C): Home, Admin(C): Home

(M): Model, (V): View, (C): Controller

FelhasznaloController metódusai:Home (get):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Helyes érték esetén visszatér a Home View-ba.

Szekrenykezeles (get):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. A Szekrenykezeles metódus feladata az, hogy eljuttassa a Viewhoz (a felhasználóhoz) az adott felhasználóhoz hozzárendelt szekrények számát. A metódus az adatbáziskapcsolaton keresztül kilistázza az adott felhasználóhoz tartozó szekrények számát és az ElerhetoszekrenyekModel-en keresztül visszatér a Szekrenykezeles View-ba.

Szekrenykezeles (post):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Ez a metódus felel a távoli szekrenykezelésért felhasználó esetében. Feladata, hogy ellenőrizze a felhasználó által bevitt értékeket és a bevitt adatoknak megfelelően kapcsolatba lépjen az egyik mikrokontrollerrel, majd a tőle kapott választ valamilyen üzenet formájában továbbítsa a felhasználó felé a Viewen keresztül. A metódus ellenőrzi, hogy a bemeneti SzekrenykezelesModel által átadott értékek lehetnek e helyesek. Helyes adatok esetén kapcsolatba lép az adott mikrokontrollerrel az RS485 osztályon keresztül. A mikrokontrollertől kapott választól függően előzmény adatokat ír az adatbázisba az adatbáziskapcsolaton keresztül. Valamilyen üzenethez tartozó értéket ad az adott TempData változónak, majd visszatér a Szekrenykezeles get metódusába.

Jelszocsere (get):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Visszatér a Jelszocsere View-ba.

Jelszocsere (post):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Ez a metódus felel azért, hogy a felhasználó meg tudja változtatni a bejelentkezéshez szükséges jelszavát. Ellenőrzi a bemeneti JelszocsereModel által átadott értékeket az adatbáziskapcsolaton keresztül. Helyes adatok esetén adatokat ír az adatbázisba az adatbáziskapcsolaton keresztül. Valamilyen üzenethez tartozó értéket ad az adott TempData változónak, majd visszatér a Szekrenykezeles View-ba.

Kijelentkezés (get):

Az adott Session értékét null-ra állítja, majd visszatér az IndexControllerbe.

5. táblázat

Név:	Request:	Bemeneti Model:	Kimeneti Model:	Visszatérés:
Home	Get	-	-	Home (V), Index(C): Login
Szekrenykezeles	Get	-	ElerhetőSzekrenyek (M)	Szekrenykezeles (V), Index(C): Login
Szekrenykezeles	Post	Szekrenykezeles (M)	-	Szekrenykezeles (V), Index(C): Login
Jelszocseres	Get	-	-	Jelszocseres (V), Index(C): Login
Jelszocseres	Post	Jelszocseres (M)	-	Jelszocseres (V), Index(C): Login
Kijelentkezés	Get	-	-	Index(C): Login

(M): Model, (V): View, (C): Controller

AdminController metódusai:Home (get):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Helyes érték esetén visszatér a Home View-ba.

Szekrenykezeles (get):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. A Szekrenykezeles metódus feladata az, hogy eljuttassa a Viewhoz (az adminisztrátorhoz) az összes szekrény számát. A metódus az adatbáziskapcsolaton keresztül kilistázza az összes szekrény számát és az ElerhetőszekrenyekModelen keresztül visszatér a Szekrenykezeles View-ba.

Szekrenykezeles (post):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Ez a metódus felel a távoli szekrénykezelésért adminisztrátor esetében. Feladata, hogy ellenőrizze az adminisztrátor által bevitt értékeket és a bevitt adatoknak megfelelően kapcsolatba lépjen az egyik mikrokontrollerrel, majd a tőle kapott választ valamilyen üzenet formájában továbbítsa az adminisztrátor felé a Viewen keresztül. A metódus ellenőrzi, hogy a bemeneti SzekrenykezelesModel által átadott értékek lehetnek e helyesek. Helyes adatok esetén kapcsolatba lép az adott mikrokontrollerrel az RS485 osztályon keresztül. A mikrokontrollertől kapott választól függően előzmény adatokat ír az adatbázisba az adatbáziskapcsolaton keresztül. Valamilyen üzenethez tartozó értéket ad az adott TempData változónak, majd visszatér a Szekrenykezeles get metódusába.

Felhasznalokezeles (get):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Feladata az, hogy eljuttassa a Viewhoz (az adminisztrátorhoz) az összes felhasználónevet és a felhasználókhoz tartozó szekrényszámokat, valamint a többi szekrényszámot. A metódus az adatbáziskapcsolaton keresztül kilistázza a létező felhasználók nevét, sorszámát és a hozzájuk tartozó szekrényszámokat, valamint kilistázza a nem használt szekrények számait. A kilistázott adatokkal visszatér a Felhasznalokezeles View-ba a SzekrenykezelesModelen keresztül.

Ujfelhasznalo (post):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Ez a metódus felel az új felhasználók létrehozásáért. Ellenőrzi a bemeneti LogininputModel által átadott értékeket. Helyes adatok esetén létrehoz egy új sort a Felhasznalok táblába az adatbáziskapcsolaton keresztül. Valamilyen üzenethez tartozó értéket ad az adott TempData változónak, majd visszatér a Felhasznalokezeles metódusba.

Kapcsolatkezeles (post):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Funkciója, hogy az adminisztrátor a weblapon keresztül tudjon felhasználó és szekrény közötti kapcsolatot módosítani. Ellenőrzi a bemeneti KapcsolatkezelesModel által átadott értékeket. Helyes adatok esetén adatokat ír az adatbázisba az adatbáziskapcsolaton keresztül. Hibás

adatok esetén valamilyen hibaüzenet értéket ad át az adott TempData változónak. Visszatér a Felhasznalokezeles metódusba.

Elozmenyek (get):

Ellenőrzi az adott Session értékét. Hiba esetén visszatér az IndexControllerbe. Az adatbáziskapcsolat segítségével kilistázza az Elozmeny tábla minden sorát majd az ElozmenyModel-en keresztül visszatér az Elozmenyek View-ba.

Kijelentkezés (get):

Az adott Session értékét null-ra állítja, majd visszatér az IndexControllerbe.

6. táblázat

Név:	Request:	Bemeneti Model:	Kimeneti Model:	Visszatérés:
Home	Get	-	-	Home (V), Index(C): Login
Szekrenykezeles	Get	-	ElerhetőSzekrenyek (M)	Szekrenykezeles (V), Index(C): Login
Szekrenykezeles	Post	Szekrenykezeles (M)	-	Szekrenykezeles (V), Index(C): Login
Felhasznalo-kezeles	Get	-	Felhasznalokezeles (M)	Felhasznalokezeles (V), Index(C): Login
Ujfelhasznalo	Post	Logininput (M)	-	Admin(C): Felhasznalokezeles, Index(C): Login
Kapcsolat-kezeles	Post	Kapcsolatkezeles (M)	-	Admin(C): Felhasznalokezeles, Index(C): Login
Elozmenyek	Get	-	Elozmeny (M)	Elozmenyek (V), Index(C): Login
Kijelentkezés	Get	-	-	Index(C): Login

(M): Model, (V): View, (C): Controller

8.4.2. Model osztályok

A Model osztályok a Models mappa elemei. Ezeket az osztályokat jelen esetben a funkciójuk szerint két csoportra lehet osztani. A Modellek egyik része a Controller felől közvetít adatokat a View felé, a másik része pedig a View felől közvetít adatokat a Controller egy metódusának. Ebben a fejezetben az általam használt Model osztályokat részletezem.

Modellek, melyeket a Controller továbbít a View felé:

Ebben az esetben a Model osztályokat a Controller egy metódusa példányosítja és ad értékeket a megfelelő változóknak. A View állományok a @model kulcsszóval tudják meghívni a megfelelő Modelt.

ElerhetoszekrenyekModel:

Szekrényszámok továbbítására létrehozott Model, mely egész számok listáját tartalmazza. Felhasználói és adminisztrátori szekrénykezelésnél használt Model.

FelhasznalokezelesModel:

Felhasználó és szekrény, valamint a köztük lévő kapcsolati adatok átadására létrehozott Model, mely tartalmaz egy string listát, valamint négy egész számokat tároló listát. Adminisztrátori oldalon, felhasználó és kapcsolat kezelésénél használt Model.

ElozmenyModel:

Előzményadatok átadására létrehozott Model, melynek változói megegyeznek az adatbázis Elozmenyek táblában szereplő oszlopok listázott változóival: két egész számokat tároló lista, egy időpontokat tároló lista, valamint három string lista. Előzményt megjelenítő oldalon használt Model.

7. táblázat

Név:	Controller felől:	View felé:
Elerhetoszekrenyek(M)	Felhasznalo(C): Szekrenykezeles, Admin(C): Szekrenykezeles	Felhasznalo: Szekrenykezeles, Admin: Szekrenykezeles
Felhasznalokezeles(M)	Admin(C): Felhasznalokezeles	Admin: Felhasznalokezeles
Elozmeny(M)	Admin(C): Elozmenyek	Admin: Elozmenyek

(C): Controller, (M): Model

Modellek, melyeket a View továbbít a Controller felé:

A View állományban a bemeneti mezőknek a neve megegyezik a használt Model osztályban szereplő változóknak a nevével. A Controller metódusaiban bemeneti értéként a használt Model van megadva, így a Viewtől átadódnak az adatok a Controller metódusainak.

LogininputModel:

Kliens oldalon bevitt felhasználónév és jelszó átadására létrehozott Model. Két stringet tartalmaz. Bejelentkezésnél és adminisztrátor által létrehozott új felhasználó bevitelénél használt Model.

SzekrenykezelesModel:

Szekrényművelethez szükséges bevitt adatok átadására létrehozott Model. Négy egész számot tároló változót tartalmaz. Felhasználói vagy adminisztrátori szekrényművelet bevitelénél használt Model.

JelszocsereModel:

Felhasználói jelszó megváltoztatásához bevitt adatok átadására létrehozott Model. Három stringet tartalmaz.

KapcsolatkezelesModel:

Felhasználó és szekrény kapcsolatát módosító bevitt adatok átadására létrehozott Model. Három egész számot tároló változót tartalmaz. Adminisztrátor által kezelt kapcsolatkezelő oldalon használt Model.

8. táblázat

Név:	View felől:	Controller felé:
Logininput(M)	Index: Login, Admin: Felhasznalokezeles	Index(C): Login, Admin(C): Ujfelhasznalo
Szekrenykezeles(M)	Felhasznalo: Felhasznalokezeles, Admin: Felhasznalokezeles	Felhasznalo(C): Szekrenykezeles, Admin(C): Szekrenykezeles
Jelszocseres(M)	Felhasznalo: Jelszocseres	Felhasznalo(C): Jelszocseres
Kapcsolatkezeles(M)	Admin: Felhasznalokezeles	Admin(C): Kapcsolatkezeles

(C): Controller, (M): Model

8.4.3. View

A megjelenéshez tartozó elemek a Views mappában találhatóak. Minden Controller osztályhoz tartozik egy, az osztállyal azonos nevű mappa. Ezekben a mappákban találhatóak az egyes metódusokhoz tartozó, megjelenésért felelős állományok. Ezek az állományok cshtml kiterjesztésűek és vegyesen tartalmazznak HTML és C# kódokat. Ebben a fejezetben ezen állományok funkcióit részletezem.

IndexControllerhez tartozó View-ek:

Login:

A bejelentkezési felület megjelenéséért felelős. Két beviteli mezőt és egy gombot jelenít meg, melyek a bejelentkezéshez szükségesek. A gomb egy Post request-en keresztül adja át a bevitt adatokat a Login metódusnak a LogininputModelen keresztül. Hibás vagy hiányzó adatok esetén az adott TempData változón keresztül hibaüzenetet jelenít meg a View.

9. táblázat

Név:	Request:	Bemeneti Model:	Kimeneti Model:	Visszatérés:
Login	Post	-	Logininput (M)	Index(C): Login

(M): Model, (C): Controller

FelhasznaloControllerhez tartozó View-ek:Home:

A felhasználói belépés után a felhasználói lehetőségeket jeleníti meg. Három gombot jelenít meg: Szekrények kezelése, Jelszó változtatása, Kilépés. A gombokra kattintva a program külön Get requesteken keresztül jut el a Controller megfelelő metódusába.

Jelszocsere:

Két beviteli mezőt és két gombot jelenít meg, amik a jelszó megváltoztatásához szükségesek. Az első gombra kattintva a program egy Post request-en keresztül átadja a bevitt adatokat a Jelszocsere metódusnak a JelszocsereModelen keresztül. Hibás vagy hiányzó adatok esetén az adott TempData változón keresztül hibaüzenetet jelenít meg a View. A második gombra kattintva a felhasználó visszajuthat a Home Viewba a Home metóduson keresztül Get request segítségével.

Szekrenykezeles:

Két legördülő listát, két beviteli mezőt és két gombot jelenít meg, melyek a felhasználói szekrényműveletekhez szükséges adatok beviteléhez szükségesek. Az első gombra kattintva a program egy Post request-en keresztül átadja a bevitt adatokat a Szekrenykezeles metódusnak (FelhasznaloController) a SzekrenykezelesModelen keresztül. A View az adott TempData változón keresztül üzenetet jeleníthet meg. A második gombra kattintva a felhasználó visszajuthat a Home Viewba a Home metóduson keresztül Get request segítségével.

10. táblázat

Név:	Request:	Bemeneti Model:	Kimeneti Model:	Visszatérés:
Home	Get	-	-	Felhasználó(C): Szekrenykezeles, Jelszocseres, Kijelentkezés
Jelszocseres	Get, Post	-	-, Jelszocseres (M)	Felhasználó(C): Jelszocseres, Home
Szekrenykezeles	Get, Post	Elerhetoszekrenyek (M)	-, Szekrenykezeles (M)	Felhasználó(C): Szekrenykezeles, Home

(M): Model, (C): Controller

AdminControllerhez tartozó View-ek:Home:

Az adminisztrátori belépés után az adminisztrátori lehetőségeket jeleníti meg. Négy gombot jelenít meg: Felhasználók kezelése, Szekrények kezelése, Előzmények, Kilépés. A gombokra kattintva a program külön Get requesteken keresztül jut el a Controller megfelelő metódusába.

Felhasznalokezeles:

Megjeleníti az összes felhasználót, valamint a hozzájuk tartozó szekrényeket. Felhasználónként megjelenít két lenyíló listát, valamint egy gombot, melyek új szekrény-felhasználó kapcsolat létrehozásához, vagy meglévő kapcsolat törléséhez szükséges beviteli elemek. Ezen gombokra kattintva a program egy Post request-en keresztül átadja a bevitt adatokat a Kapcsolatkezeles metódusnak a KapcsolatkezelesModelen keresztül. Továbbá megjelenít még két beviteli mezőt és egy gombot, melyek új felhasználó létrehozásához szükségesek. Erre a gombra kattintva a program egy Post requesten keresztül adja át a bevitt adatokat az Ujfelhasznalo metódusnak a LogininputModelen keresztül. A View az adott TempData változón keresztül üzenetet jeleníthet meg. Az előbbieken említett beviteli mezőkön kívül a Viewon megtalálható még egy gomb, melyre kattintva az adminisztrátor visszajuthat a Home Viewba a Home metóduson keresztül Get request segítségével.

Szekrenykezeles:

Két legördülő listát, egy beviteli mezőt és két gombot jelenít meg, melyek az adminisztrátori szekrényműveletekhez szükséges adatok beviteléhez szükségesek. Az első gombra kattintva a program egy Post request-en keresztül átadja a bevitt adatokat a Szekrenykezeles metódusnak (AdminController) a SzekrenykezelesModelen keresztül. A View az adott TempData változón keresztül üzenetet jeleníthet meg. A második gombra kattintva az adminisztrátor visszajuthat a Home Viewba a Home metóduson keresztül Get request segítségével.

Elozmenyek:

Megjeleníti az összes előzményt és egy gombot. A gombra kattintva az adminisztrátor visszajuthat a Home Viewba a Home metóduson keresztül Get request segítségével.

11. táblázat

Név:	Request:	Bemeneti Model:	Kimeneti Model:	Visszatérés:
Home	Get	-	-	Admin(C): Szekrenykezeles, Felhasznalokezeles, Elozmenyek, Kijelentkezés
Felhasznalo- kezeles	Get, Post	Felhasznalokezeles (M)	-, Logininput (M), Kapcsolatkezeles (M)	Admin(C): Ujfelhasznalo, Kapcsolatkezeles, Home
Szekrenykezeles	Get, Post	Elerhetoszekrenyek (M)	Szekrenykezeles (M)	Admin(C): Szekrenykezeles, Home
Elozmenyek	Get	Elozmeny (M)	-	Admin(C): Home

(M): Model, (C): Controller

9. Kapcsolat

Ebben a fejezetben részletezem a kommunikációs buszrendszer működését. A számítógép és a mikrokontrollerek master-slave kapcsolatban vannak. Ez jelen esetben azt jelenti, hogy minden esetben a számítógépes program (master) kezdeményezi a kommunikációt a mikrokontrollerekkel (slave). Normál működés esetén a mikrokontrollerek csak adott időn belül, közvetlen a számítógéptől kapott, az adott mikrokontrollernek szóló üzenet után küldhet válaszüzenetet a buszrendszeren keresztül. A mikrokontrollerek nem kommunikálnak egymással.

A végponti egységek megkülönböztetése:

A végponti eszközök megkülönböztetésére több módszert is alkalmaztam. A kommunikáció csak adott hosszúságú üzeneteken keresztül történhet. A számítógép által küldött üzenetek 3, 8 vagy 9 bájt hosszú üzenetek lehetnek. A mikrokontrollerek által küldött üzenetek kizárólag 4 bájt hosszúak lehetnek. Ebből adódik, hogy a számítógép kizárólag 4 bájt, a mikrokontrollerek pedig 3, 8 vagy 9 bájt hosszú üzeneteket dolgoznak csak fel. Mind a 4 fajta üzenet első és utolsó bájtja egy adott értékű keretnek felel meg, ez üzenetenként eltérő. Az eszközök a keret által azonosítják a beérkező üzenet hosszát. Minden végponti slave egység rendelkezik egy 1 bájt hosszú címmel. A számítógép ezek alapján a címek alapján tud az adott mikrokontrollernek szóló üzenetet küldeni.

Az üzenetek ellenőrzése:

Számítógép:

A számítógép csak üzenet küldés után fogad beérkező adatokat egy adott időintervallumon belül. Ezen időintervallumon belül folyamatosan ellenőrzi a beérkező bájtok értékét. Ha egy beérkező bájtnak az értéke megegyezik a 4 bájtos üzenet első keretértékével, akkor a következő 3 beérkező bájt értékét elmenti egy változóba. Ha ezek közül az értékek közül a 3. beérkező adat megegyezik a 4 bájtos üzenet hátsó keretértékével, akkor feldolgozza a középső 2 adatot.

Mikrokontrollerek:

A mikrokontrollerek folyamatosan figyelik a buszrendszert egy megszakítás segítségével. Bármilyen adat érkezésekor megszakítás generálódik a mikrokontrolleres programban és a bejövő adatok feldolgozásra kerülnek. Ha egy beérkező bájtnak az értéke megegyezik a 3, 8 vagy 9 bájtos üzenet első keretértékével, akkor a következő 2, 7 vagy 8 beérkező bájtnak az értékét elmenti egy változóba. Ha ezek közül az értékek közül a 2., 7. vagy 8. beérkező adat megegyezik a 3, 8 vagy 9 bájtos üzenet hátsó keretértékével, akkor feldolgozza a középső 2, 7 vagy 8 adatot. Minden a számítógéptől érkező üzenet 2. bájta a megszólítandó mikrokontroller címe. Helyes keret beérkezése után a mikrokontroller ellenőrzi az üzenet 2. bájton lévő címet. Ha a cím nem egyezik meg a sajátjával akkor az üzenet többi részét figyelmen kívül hagyja, nem küld választ a számítógépnek.

Az üzenetek:

Számítógép által küldött üzenetek:

3 bájtos üzenet:

Ennek az üzenetnek a célja a mikrokontrollerek megszólítása. Ezt az üzenetet a számítógép adott időközönként periodikusan kiküldi.

1.-3. bájtnak: keret

2. bájtnak: megszólítandó mikrokontroller címe

8 bájtos üzenet:

Ez az üzenet teremti meg a kapcsolatot a távoli nyitás, zárás, zárolás, zárolás feloldás és felszabadítás esetén a mikrokontroller és a számítógép között. A számítógép művelet, szekrény és kód adatokat küld a mikrokontrollernek.

1.-8. bájtnak: keret

2. bájtnak: az adott szekrényhez tartozó mikrokontroller címe

3. bájtnak: 0-2 bit: művelet kódja, 5-7 bit: adminisztrátori művelet esetén az adminisztrátori kód utolsó 3 bitje

4. bájtnak: szekrényszám

5.-7. bájt: az adott szekrényhez tartozó kód

9 bájtos üzenet:

Ez az üzenet teremti meg a kapcsolatot a kód módosítás esetén a mikrokontroller és a számítógép között. A számítógép szekrény, kód és új kód adatokat küld a mikrokontrollernek. Ebben az esetben nincs szükség művelet küldésére, mert az az üzenet hosszából megállapítható.

1.-9. bájt: keret

2. bájt: az adott szekrényhez tartozó mikrokontroller címe

3. bájt: szekrényszám

4.-6(alsó). bájt: az adott szekrényhez tartozó kód

6(felső).-8. bájt: új kód

Mikrokontroller által küldött üzenetek:

4 bájtos üzenet:

Ennek az üzenetnek 2 funkciója van. Ha a számítógép 3 bájtos üzenettel szólítja meg a mikrokontrollert, akkor a mikrokontrollernek lehetősége van előzmény választ adni. Ez abban az esetben fordul elő, mikor a mikrokontrolleren valamilyen közeli művelet hajtott végre két megszólítás között. Ebben az esetben a mikrokontroller a megszólítást követően előzmény adatokat küld a számítógépnek, hogy az rögzíteni tudja az új előzményt az adatbázisban. Ilyenkor a küldendő adatok a következőféleképpen oszlanak el:

1.-4. bájt: keret

2. bájt: 0.-2. bit: művelet kódja, 3. bit: művelet végrehajtásának eredménye (sikeres/sikertelen), 7. bit: előzményadat/válasz küldését jelző bit

3. bájt: szekrényszám (adminisztrátori belépés esetén 0)

Ha nincs új előzményadat az adott mikrokontrollerben, akkor az egy üres keretet küld vissza a számítógépnek.

Ha a számítógép 8 vagy 9 bájtos üzenettel szólítja meg a mikrokontrollert, akkor a mikrokontroller választ ad a távoli művelet elvégzésének sikerességéről. Ilyenkor a küldendő adatok a következőféleképpen oszlanak el:

1.-4. bájtt: keret

2. bájtt: 0.-2. bit: művelet kódja, 3.-4. bit: művelet végrehajtásának eredménye (sikeres/sikertelen/zárolt/felszabadított), 7. bit: előzményadat/válasz küldését jelző bit

3. bájtt: szekrényszám

Az adatok ellenőrzése:

Számítógép:

Minden beérkező adat esetében a számítógép ellenőrzi az adatok helyességét. Hibás előzményadat fogadása esetén az adatbázisba nem kerül új előzménysor. Távoli művelet végrehajtása esetén a válasz több ellenőrző műveleten megy keresztül és hiba esetén hibaüzenetet közöl a felhasználóval. A hibaüzeneteket a későbbiekben tárgyalom részletesebben.

Mikrokontroller:

A mikrokontroller a távoli műveleteket szigorú ellenőrzésnek veti alá. Sorban, külön ellenőrzi a szekrényszám helyességét, a szekrényszám elérhetőségét (zárolt), a művelet helyességét, valamint a kód helyességét. Bármely pontban való elakadás esetén válaszként különböző hibaüzenetet küld a számítógépnek.

Szinte minden esetben előfordul, hogy az üzenetek bájtaiban előfordulnak kihasználatlan bitek. Ezeket a biteket minden esetben arra használom, hogy ellenőrizsem az üzenet helyességét, tehát a kihasználatlan biteknek mindig adott állapotban kell lenniük, amik ellenőrzésre kerülnek az adatok feldolgozása folyamán.

Sikertelen belépés esetén a 13. ábra 3. üzenete jelenik meg. 3 sikertelen próbálkozás után a szekrény zárolttá válik körülbelül 3 percre.

Sikeres belépést követően megjelenik a felhasználói menü (12. ábra). A felhasználó választhat a nyitás, zárás, kód megváltoztatása és kilépés menüpontok közül. Az első 3 menüpont valamelyikét választva a műveletek elvégzése után a G5 (visszalépés) gombot megnyomva újra a felhasználói menü jelenik meg. Az első 3 menüpont valamelyikét választva a műveletek elvégzése után a 13. ábra 4., 5. vagy 6. üzenete jelenhet meg. A 4. menüpontot választva a program a kezdő menüt jeleníti meg. Ebben az esetben újbóli belépéssel lehet a szekrényműveletekhez jutni.

12. ábra
Képernyőkép közeli felhasználói menüről

10.1.2. Adminisztrátori útmutató

A közeli adminisztrátori szekrénykezelés lehetséges képernyőképei a következőek:

A kezdő menün a G2 gomb nyomása után megjelenik az adminisztrátori belépés képernyője (14. ábra). Az adminisztrátori kód bevitelekor a bevitt számokat „*” karakter takarja.

14. ábra
Képernyőkép közeli adminisztrátori belépésről

K	o	d	:													1. sor
---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--------

Helytelen kód bevitele után a 16. ábra 3. üzenete jelenik meg. Zárolt adminisztrátori belépés esetén a 16. ábra 2. üzenete jelenik meg.

Helyes adminisztrátori kód bevitele után megjelenik az adminisztrátori menü (15. ábra). Az adminisztrátor választhat a szekrényzárolás, szekrényzárolás feloldása és szekrény felszabadítása menüpontok közül. Zárolás esetén az adminisztrátor a használatban lévő szekrényeket elérhetetlenné teheti a felhasználó számára. Felszabadítás esetén a felszabadított szekrény első használat előtti állapotba kerül. Az első 3 menüpont valamelyikét választva a műveletek elvégzése után a G5 (visszalépés) gombot megnyomva újra az adminisztrátori menü jelenik meg. Az első 3 menüpont valamelyikét választva a műveletek elvégzése után a 16. ábra 4., 5. vagy 6. üzenete jelenhet meg. A 4. menüpontot választva a program a kezdő menüt jeleníti meg. Ebben az esetben újbóli belépéssel lehet a szekrényműveletekhez jutni.

15. ábra
Képernyőkép közeli adminisztrátori menüről

G1	:		Z	a	r	o	l	a	s							1. sor
G2	:		F	e	l	o	l	d	a	s						2. sor
G3	:		F	e	l	s	z	a	b	a	d	i	t			3. sor
G5	:		K	i	l	e	p	e	s							4. sor

Adminisztrátori használat esetén a következő üzenetek jelenhetnek meg (16. ábra):

16. ábra
Képernyőkép a közeli adminisztrátori üzenetekről

1.	H	i	b	a	s		s	z	a	m	!						4. sor
2.	A	z		a	d	m	i	n		z	a	r	o	l	t	!	4. sor
3.	H	i	b	a	s		k	o	d	!							4. sor
4.	S	i	k	e	r	e	s		z	a	r	o	l	a	s	!	4. sor
5.	Z	a	r	o	l	a	s		t	o	r	o	l	v	e	!	4. sor
6.	F	e	l	s	z	a	b	a	d	i	t	v	a	!			4. sor

10.2. Távoli használat

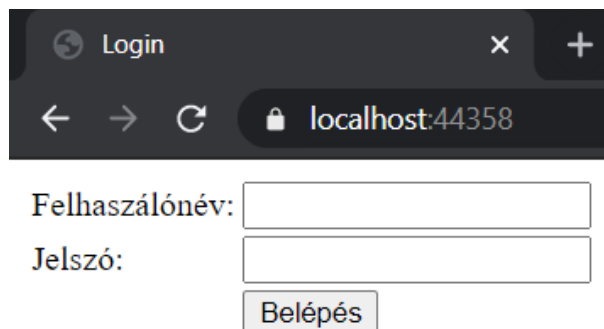
Távoli használat esetén a rendszer működésbe lépése után a számítógépes programot futtató számítógépen egy böngészőn keresztül lehet elérni a távoli használat kezelőfelületét a localhost címen.

10.2.1. Felhasználói útmutató

A távoli felhasználói szekrénykezelés lehetséges képernyőképei a következők:

Bejelentkezési képernyő (17. ábra):

17. ábra
Képernyőkép távoli bejelentkezésről



Login

← → ↻ 🔒 localhost:44358

Felhasználónév:

Jelszó:

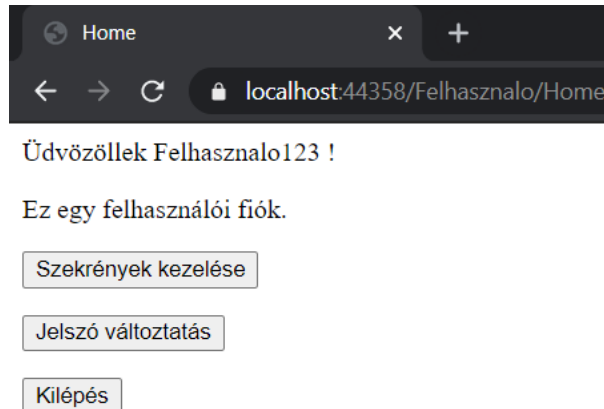
Belépés

Sikertelen bejelentkezés esetén a 21. ábra 7. vagy 8. üzenete jelenhet meg.

Sikeres bejelentkezés esetén megjelenik a felhasználói menü (18. ábra). Itt a felhasználó választhat a szekrények kezelése, jelszó megváltoztatása és kilépés menüpontok közül.

18. ábra

Képernyőkép távoli felhasználói menüről



A felhasználói menüben a szekrények kezelése gombra kattintva megjelenik a szekrénykezelés oldala (19. ábra). Az első legördülő listából lehet kiválasztani a kezelni kívánt szekrényt. A második legördülő listából lehet kiválasztani az elvégezni kívánt műveletet. Nyitás és zárás esetén a kód mezőben kell megadni a szekrényhez tartozó kódot. Kód módosítása esetén a kód mezőben kell megadni a régi kódot és az új kód mezőben az új kódot. Első használat esetén csak az új kód mezőt kell kitölteni és az ekkor megadott kód lesz érvényes a későbbi műveleteknél. A küldés gombra kattintva a rendszer eljuttatja a bevitt adatokat a szekrény felé. A vissza gombra kattintva megjelenik a felhasználói menü (18. ábra).

19. ábra
Képernyőkép távoli felhasználói szekrénykezelésről

Szekrenyek kezelése

Szekrenyszám: Válassz szekrényt! ▾

Művelet: Válassz művelet! ▾

Kód:

Új kód: Csak kód megváltoztatása és első használat esetén!

Küldés

Vissza

A küldés gombra kattintva a 21. ábra 2., 3., 4., 5., 6. vagy 7. üzenete jelenhet meg.

A felhasználói menüben a jelszó változtatás gombra kattintva megjelenik a jelszó változtatás oldala (20. ábra). Itt a jelszó mezőt a jelenlegi jelszóval és az új jelszó mezőt egy tetszőleges új jelszóval kitöltve a felhasználó megváltoztathatja a fiók jelszavát. A módosítás gombra kattintva a rendszer ellenőrzi a bevitt adatokat. A vissza gombra kattintva megjelenik a felhasználói menü (18. ábra).

20. ábra
Képernyőkép távoli felhasználói jelszó változtatásáról

Fiók jelszavának megváltoztatása

Jelszó:

Új jelszó:

Módosítás

Vissza

A módosítás gombra kattintva a 21. ábra 1., 7. vagy 8. üzenete jelenhet meg.

A felhasználói menüben a kilépés gombra kattintva megjelenik a bejelentkezési oldal (17. ábra).

Felhasználói használat esetén a következő üzenetek jelenhetnek meg (21. ábra):

21. ábra

Képernyőkép távoli felhasználói üzenetekről

1. Sikeres csere!
2. Zárolt szekrény!
3. Felszabadított szekrény!
4. Sikeres művelet!
5. Sikertelen művelet!
6. Kapcsolati hiba!
7. Hiányzó adatok!
8. Hibás adatok!

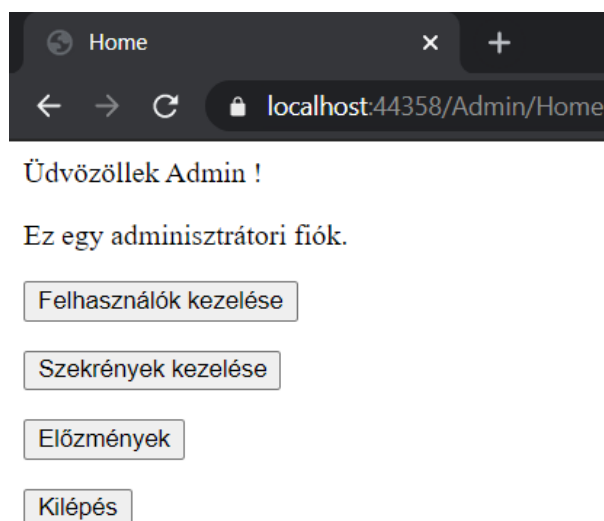
10.2.2. Adminisztrátori útmutató

A távoli adminisztrátori felület lehetséges képernyőképei a következők:

Sikeres bejelentkezés esetén megjelenik az adminisztrátori menü (22. ábra). Itt az adminisztrátor választhat a felhasználók kezelése, szekrények kezelése, előzmények és kilépés menüpontok közül.

22. ábra

Képernyőkép távoli adminisztrátori menüről



Az adminisztrátori menüben a felhasználók kezelése gombra kattintva megjelenik a felhasználók kezelése oldal (23. ábra). Itt az új felhasználó létrehozása szöveg alatt a felhasználónév és jelszó mezőt kitöltve az adminisztrátor létrehozhat új felhasználót. A felhasználók és szekrények kezelése szöveg alatti táblázat megjeleníti az összes felhasználót és a hozzájuk tartozó szekrényeket. A törlés legördülő lista segítségével az adminisztrátor megfoszthatja az adott felhasználót a szekrényeitől. A hozzáadás legördülő lista segítségével pedig használati jogot adhat egy éppen nem használt szekrény használatára. A létrehozás vagy a módosítás gombokra kattintva a rendszer ellenőrzi az adatokat és elvégzi a műveleteket. A vissza gombra kattintva megjelenik az adminisztrátori menü (22. ábra).

23. ábra
Képernyőkép távoli felhasználókezelésről

Felhasználók kezelése

Új felhasználó létrehozása:

Felhasználónév: Jelszó:

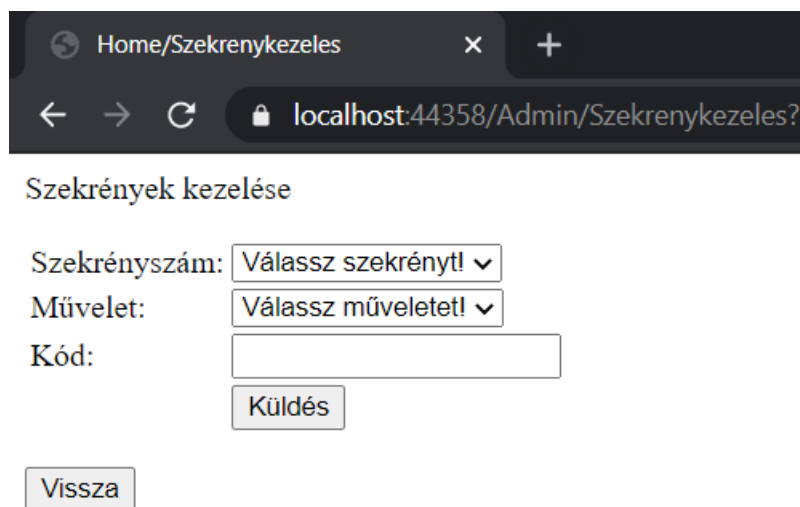
Felhasználók és szekrények kezelése:

Sorszám	Felhasználónév	Szekrények			
8	ujfelhasznalo	1, 4	Törlés ▼	Hozzáadás ▼	Módosítás
10	Ujfelhasznalo	3	Törlés ▼	Hozzáadás ▼	Módosítás
11	Felhasznalo123	2, 7	Törlés ▼	Hozzáadás ▼	Módosítás

A létrehozás vagy módosítás gombra kattintva a 26. ábra 1. üzenete jelenhet meg.

Az adminisztrátori menüben a szekrények kezelése gombra kattintva megjelenik a szekrények kezelése oldal (24. ábra). Az első legördülő listából lehet kiválasztani a kezelni kívánt szekrényt. A második legördülő listából lehet kiválasztani az elvégezni kívánt műveletet. A kód mezőbe minden esetben be kell írni az adminisztrátori kódot a műveletek sikeres elvégzésének érdekében. A küldés gombra kattintva a rendszer eljuttatja a bevitt adatokat a szekrény felé. A vissza gombra kattintva megjelenik az adminisztrátori menü (22. ábra).

24. ábra
Képernyőkép távoli szekrénykezelésről



Home/Szekrenykezeles

localhost:44358/Admin/Szekrenykezeles?

Szekrények kezelése

Szekrényszám: Válassz szekrényt! ▼

Művelet: Válassz műveletet! ▼

Kód:

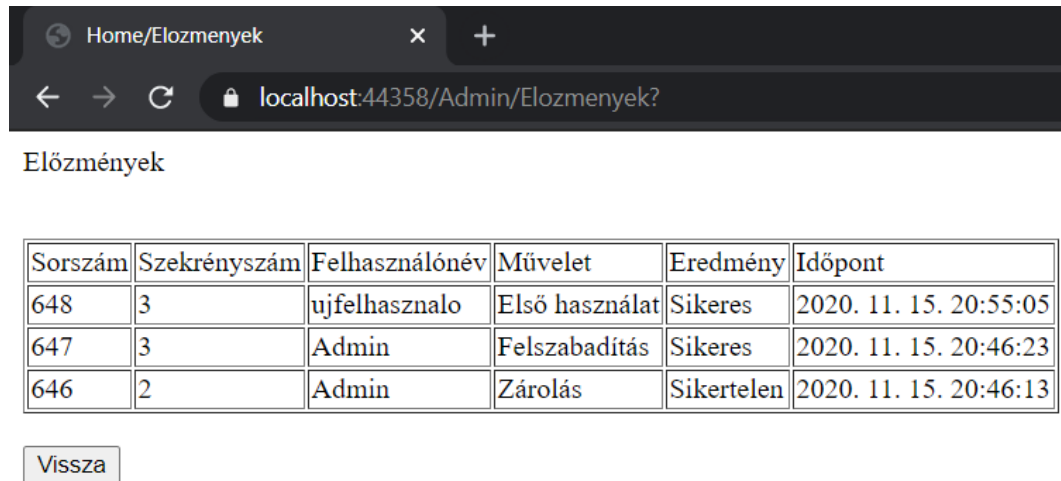
Küldés

Vissza

A küldés gombra kattintva a 26. ábra 1., 3., 4., 5. vagy 6. üzenete jelenhet meg.

Az adminisztrátori menüben az előzmények gombra kattintva megjelenik az előzmények oldal. Ezen az oldalon időrendi sorrendben megjelenik minden szekrényművelettel kapcsolatos tevékenység.

25. ábra
Képernyőkép távoli előzményekről



Sorszám	Szekrényszám	Felhasználónév	Művelet	Eredmény	Időpont
648	3	ujfelhasznalo	Első használat	Sikeres	2020. 11. 15. 20:55:05
647	3	Admin	Felszabadítás	Sikeres	2020. 11. 15. 20:46:23
646	2	Admin	Zárolás	Sikertelen	2020. 11. 15. 20:46:13

Vissza

Az adminisztrátori menüben a kilépés gombra kattintva megjelenik a bejelentkezési oldal (17. ábra).

Adminisztrátori használat esetén a következő üzenetek jelenhetnek meg (26. ábra):

26. ábra
Képernyőkép távoli adminisztrátori üzenetekről

1. **Hiányzó adatok!**
2. **Hibás adatok!**
3. **Sikeres művelet!**
4. **Sikertelen művelet!**
5. **Kapcsolati hiba!**
6. **Zárolt adminisztrátor!**

11. Tesztelés és hibakeresés

A mikrokontrolleres és számítógépes program fejlesztése során nagy szerepet játszott a tesztelés és hibakeresés. A mikrokontrolleres programot lépésről lépésre kis részenként fejlesztettem. A viszonylag hosszú és sok részből álló programot próbáltam tölem telhetően minél kisebb és érthetőbb részekre bontani. Így a mikrokontrolleres program végeredményében 9 állományból és kb. 30 függvényből építettem fel. A fejlesztés során a legtöbb függvény megírása után debug módban vagy a mikrokontrollerre rátöltve teszteltem a programrészeket és csak akkor haladtam tovább a program többi részének a fejlesztésével, ha az előző tesztek sikeresek voltak. A mikrokontrolleres tesztelésben a mátrix billentyűzet, gombok, LED-ek és LCD kijelző segített az egyes bemeneti értékek bevitelében és a kimeneti adatok megjelenítésében.

A számítógépes programnál adott volt a program egyes részeinek a tagolása az MVC programtervezési minta miatt. A programtervezési mintán belül és kívül szintén törekedtem a logikus és minél kisebb részekre tagolt fejlesztésre. A programba beépülő külső elemeket (adatbázis, RS485 kommunikáció) mindig külön programon keresztül teszteltem és csak akkor építettem be a fő programba, mikor már megfelelően működtek. A számítógépes programot debugolással és a programot futtatása közbeni működést figyelve teszteltem.

A rendszer megvalósítása során az RS485-ös kommunikáció létrehozása volt számomra a legnehezebb feladat. A mikrokontrolleres és a számítógépes programban is létre kellett hoznom olyan programrészeket, amik adatokat tudnak küldeni és fogadni a buszrendszeren keresztül. A kommunikáció során felmerülő hibák tesztelése eleinte hosszú folyamat volt, mert nem lehetett tudni, hogy a küldő vagy a fogadó oldalon van a hiba. A könnyebb hibakeresés érdekében egy LCD kijelzővel ellátott mikrokontrollert csatlakoztattam a buszrendszerre, ami a kijelzőn folyamatosan mutatta a kommunikáción áthaladó bájtok értékét hexadecimális formában. Ez a megoldás nagyban elősegítette a tesztelés és hibakeresés folyamatát a későbbi programrészek tesztelésénél is.

12. Fejlesztési lehetőségek

A rendszerrel kapcsolatos fejlesztési lehetőségek felsorolása:

- Mikrokontroller memóriájában tárolt, a szekrények nyitásához szükséges kódok titkosítása.
- Adatbázisban tárolt, felhasználói fiókokhoz tartozó jelszavak titkosítása.
- RS485-ös kommunikáción küldött szekrénykódok titkosítása.
- Adminisztrátori weboldalak kibővítése a felhasználók törlése, módosítása, adminisztrátori fiók létrehozása funkciókkal.
- Felhasználói weboldalak bővítése a felhasználóhoz tartozó szekrények előzményadataival.
- Weblap bővítése design elemekkel a kellemesebb megjelenítés érdekében.
- Mobil alkalmazás vagy weblap mobil kinézet fejlesztése a tágabb körű elérhetőség érdekében.
- Automatikus tesztelés a lehetséges hibák könnyebb detektálása érdekében.

A lista nem teljes, további fejlesztési lehetőségekkel bővíthető.

13. Összefoglalás

Szakedolgozatom célja egy értékmegőrző szekrényrendszer létrehozása volt, melynek szekrényajtajai közelről és távolról is nyithatóak, zárhatóak és kezelhetők. A rendszer felhasználhatóságát tekintve alkalmas lenne olyan értékmegőrző pont kialakítására, ahol a felhasználóknak nem kellene másokkal megosztaniuk a szekrényük nyitási adatait, hanem szekrényinformációk átadása nélkül tudnák megosztani a szekrényük tartalmát.

Azért választottam ezt a témát, mert izgalmas és szerteágazó tanulási lehetőséget nyújtott. A rendszer tervezése és megalkotása során több villamosmérnöki területtel meg kellett ismerkedjek. Szakedolgozatom elkészítése során valós példán gyakorolhattam az egyetemi éveim alatt elsajátított C nyelvű programozást mikrokontrolleres környezetben, megismerkedtem részletesen az RS485-ös kommunikációval és egy buszrendszer valós működésével, elsajátítottam alapfokú webalkalmazás létrehozásához szükséges C# és HTML ismereteket, továbbá megismerkedtem az alkatrészválasztás és a forrasztás folyamataival.

A dolgozatom eleje tartalmaz egy rendszer specifikációt. Az ezt követő fejezetekben tárgyaltam a specifikáció alapján a feladat megoldását. A dolgozatban igyekeztem lépésről lépésre, megfelelő sorrendben kifejtetni az adott részfeladatok megoldásait.

A feladat megoldásának részletes kifejtése után szerepel egy felhasználói útmutató, melyben részleteztem a rendszer helyes használatát felhasználói szemszögből.

A dolgozat elején meghatározott követelményeknek megfelelően sikeresen megvalósítottam a feladatot.

14. Summary

The aim of my dissertation was to create a value-preserving cabinet system, the cabinet doors of which can be opened, locked and operated both up close and remotely. In terms of usability of the system, it would be suitable to create a value storage point where users would not have to share their cabinet opening data with others, but would be able to share the contents of their cabinet without passing on cabinet information.

I chose this topic because it provided an exciting and diverse learning opportunity. During the design and creation of the system, I had to get acquainted with several areas of electrical engineering. During the preparation of my dissertation, I was able to practice my C programming language skills I acquired during my university years in a microcontroller environment. I got acquainted in detail with RS485 communication and the real operation of a bus system. I have mastered the C # and HTML skills needed to create a basic web application, and I have become familiar with the processes of component selection and soldering.

The beginning of my dissertation contains a system specification. In the following chapters, I discussed the solution of the problem based on the specification. In the dissertation I tried to explain the solutions of the given subtasks step by step, in the appropriate order.

After explaining the solution of the task in detail, there is a user guide in which I detailed the correct use of the system from the user's point of view.

I successfully completed the task according to the requirements set at the beginning of the dissertation.

15. Irodalomjegyzék

Atmega64 datasheet. (dátum nélk.). Letöltés dátuma: 2020. 10 11, forrás:

http://ww1.microchip.com/downloads/en/DeviceDoc/atmel-2490-8-bit-avr-microcontroller-atmega64-l_datasheet.pdf

Hitachi HD44780U datasheet. (dátum nélk.). Letöltés dátuma: 2020. 10 11, forrás:

<https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>

KeySafe. (dátum nélk.). Letöltés dátuma: 2020. 09 22, forrás: <http://www.keysafe.hu/termekek.html>

procontrol. (dátum nélk.). Letöltés dátuma: 2020. 09 22, forrás:

https://www.procontrol.hu/hu/category/details/id/181834-ProxerLock_3-11_On-line_szekrenyzar_beepitett_RFID_olvasoval,_PoE_tapellatas,_RS485_interfesz,jobbos

YE-304NO datasheet. (dátum nélk.). Letöltés dátuma: 2020. 10 14, forrás:

https://www.forschnit.hu/index.php?route=product/product/get_file&file=YE-304NO-Adatlap.pdf