12. Automatizációs és biztonsági rendszer családi házak számára RFID azonosítással

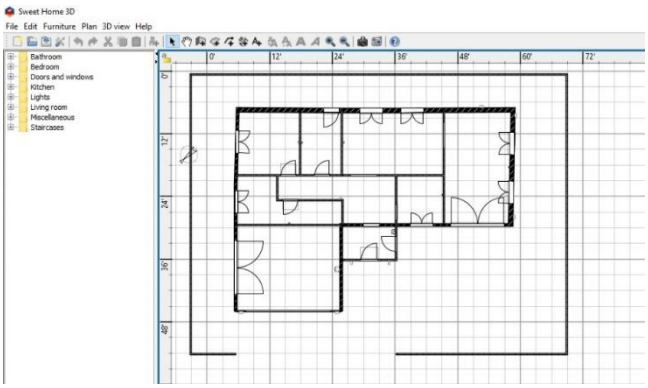
Ebben a feladatban az eddig tanultakat fogjuk hasznosítani és egy nagy projekt formájában megvalósítani. A projektet egy családi ház modell megépítésével fogjuk kezdeni. Ez a folyamat elég időigényes és precíz munkát igényel. A családi ház kartonlapból készül és méretének lehetőséget kell biztosítania a különféle szenzorok és egyéb eszközök elhelyezhetőségét. A működése a következő, egy MFRC522-es RFID olvasó segítségével tudjuk aktiválni a vezérlést, a mi esetünkben egy kártya lesz az melynek az azonosítója beolvasásával belépési jogot kapunk a rendszerünkbe. Miután a belépési szándék azonosítása és jóváhagyása megtörtént a valós élethez hasonlóan azt szeretnénk, hogy betudjunk jutni a házunk területére. Jelen esetben a rendszer úgy lesz megtervezve, hogy a belépni kívánó személy személygépjárművel érkezik a házhoz és a kapunál található MFRC522-es olvasóra helyett kártyája segítségével szeretne a ház garázsába jutni. Ezért, ha a kártya azonosítása megtörtént, akkor a bejárati kaput kell kinyitnunk, ezt egy szervomotor segítségével és a megfelelő mechanikai kapcsolattal a motor és a kapu közt tudjuk véghez vinni, miután a kapunyitásra került a garázsnyitásról kell gondoskodnunk, tehát egy újabb szervomotor segítségével ezt a mechanikai lépést is meg tudjuk oldani. Amíg a felhasználó egyén várakozik az előbb említett folyamatok végbe menetelére addig az azonosító terminálnál elhelyezett 3x20 karakteres kijelző különféle információk megjelenítését végzi, mint például a kint, illetve benti hőmérséklet és páratartalom, aktív vagy inaktív riasztó rendszer, aktív vagy inaktív szellőztető rendszer, aktív vagy inaktív hűtési rendszer, van e jelenleg, szénmonoxid, lpg gáz vagy esetleg, füst szivárgás. Amennyiben ezen állapotok közül valamelyik változik az imént említett LCD kijelzőn ez megfelelő figyelmeztetés formájában megjelenik. A kinti és benti hőmérséklet, illetve páratartalom mérésére egy-egy DHT11-es szenzort használunk, a szénmonoxid, lpg gáz vagy füst érzékelését egy MQ2-es szenzor figyeli. A riasztó rendszer PIR mozgásérzékelő szenzorok segítségével van megvalósítva. Amennyiben ezen szenzorok közül valamelyik mozgást észlel akkor a szobában, ahol a mozgás történt villogással és az LCD kijelzőre megjelenített figyelmeztetéssel jelez. A szellőztető rendszer néhány ventilátor segítségével lesz megoldva és ennek aktiválást az MQ2-es szenzorból kapott jel mérésénél meghatározott küszöbérték segítségével fogjuk aktiválni. Ehhez hasonlóan a hűtőrendszert pedig a DHT11-es szenzor egy általunk megadott hőmérsékleti küszöbértékkel tudjuk aktiválni, mely a mi esetünkben 25 Celsius fok. Végül azt szeretnénk, hogy a ház külső világítása, valamint a kerítésen

elhelyezett LED diódák a sötétség bekövetkeztével aktiválódjanak ezt egy egyszerű fotó ellenállás vagy potenciométer segítségével valósítjuk meg.

Szükséges eszközök az áramkör megépítéséhez: Számítógép, Arduino IDE, Arduino Mega 2560, USB kábel, négy db PIR mozgásérzékelő szenzor, egy db HC-SR04-es ultrahangos távolságmérő, kettő db DHT11-es hőmérséklet és páratartalom szenzor, egy db MQ2-es szénmonoxid, lpg gáz és füst érzékelő szenzor, kettő darab Tower Pro SG90 típusú szervomotor, egy db nyolccsatornás relé modul, egy db MFRC522-es RFID olvasó RFID kártyával és RFID tag-el, egy db 3x20 karakteres LCD kijelző, I2C adapterrel, harminc darab hidegfehér fényű LED dióda, négy darab villogó áramkörrel ellátott LED dióda, három db kisméretű 12 volt vagy 5 volt feszültségű ventilátor, 50 méter UTP kábel, tíz darab tizenkét csatlakozós sorkapocs, szükség szerint univerzális nyomtatott áramkör a LED világítás védőáramkörének a megépítésére.

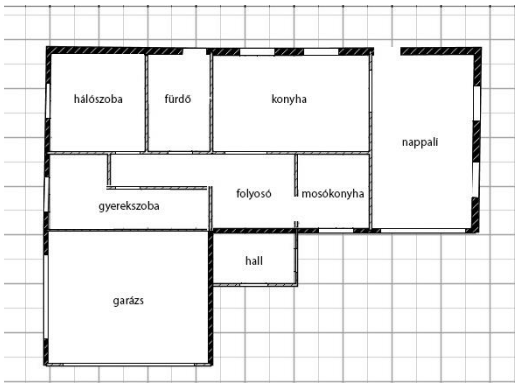
Szükséges eszközök a ház modell megépítéséhez: legalább egy centiméter vastagságú karton lap, kettő db styrofoam lap, zöld és kék színű szövet, ragasztópisztoly és ragasztórudak.

Ebben az esetben nem a programkóddal kezdünk, hanem a ház modellünk megtervezésével, ehhez segítségül a *Sweet Home 3D* ingyenesen letölthető tervezőprogramot használjuk. Itt egyszerűen a falak megrajzolásával kezdjük. Mely nagyon egyszerű a programban megjelenő felülnézeti ábra segítségével (1.ábra).



1. ábra Sweet Home 3D felülnézeti ábra
(forrás: saját szerkesztés)

Az elkészített modellünkön jelöljük meg az egyes helységeket, hogy a programozás során tudjunk rájuk hivatkozni. Ezt a következő módon végezzük el (2.ábra).

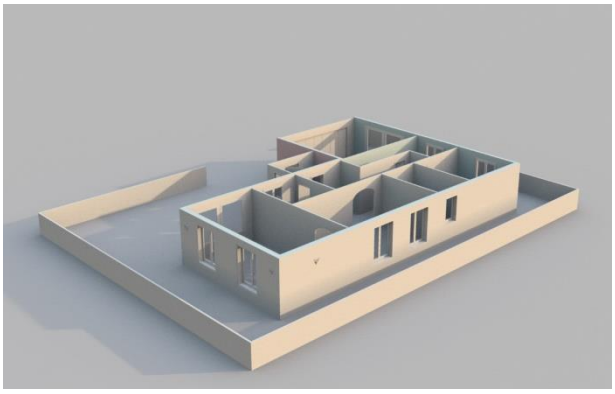


2. ábra A ház helységeinek jelölése
(forrás: saját szerkesztés)

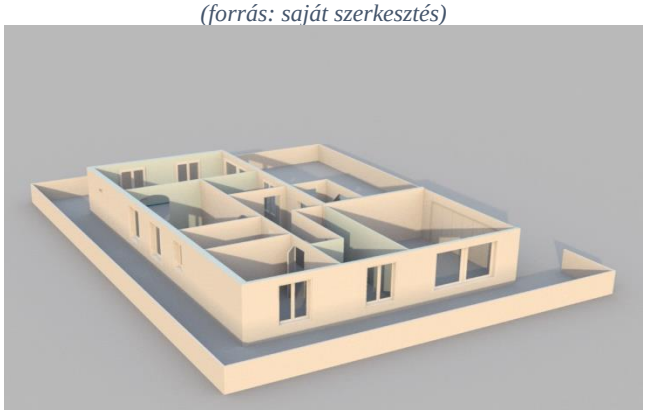
Amint láthatjuk a ház kilenc helységgel rendelkezik, garázs, gyerekszoba, hálószoba, fürdő, konyha, nappali, mosókonyha, folyosó és a hall. Az utóbbi két helyiséget vezérlés és egyéb szempontból egyként fogjuk kezelni. A modellünkön látható, az ajtók ablakok és egyéb nyílások elhelyezése is. Ez fontos lehet a szenzorjaink elhelyezésénél ezért gondoljuk át mit hova helyezünk. Ezt követően kiexportálhatjuk a modell háromdimenziós verzióját mely segítségünkre lehet a megépítésnél is.

3. ábra A ház előnézete 3d ben
(forrás: saját szerkesztés)

A fenti 3.ábrán a ház háromdimenziós modelljét láthatjuk előnézetből. Az élethűség érdekében a programban lehetőségünk van a fények ki és bekapcsolására is. Hasznunkra válik, ha a modell minden oldaláról készítünk egy kiexportált verziót melyet a későbbiekben kinyomtathatunk és mint afféle útmutatásként használhatunk a megépítés folyamatában. Ezért a ház hátuljáról és a bal oldaláról is készítünk egy-egy képet.



4. ábra A ház hátulnézeté 3d-ben
(forrás: saját szerkesztés)



5. ábra A ház bal oldala 3d-ben
(forrás: saját szerkesztés)

Mivel a tervezőprogramban van lehetőségünk az éjszakai módra is ezért egy ilyen képet is készíthetünk, ahol, az elhelyezett világító elemeket tudjuk megfigyelni (6.ábra).

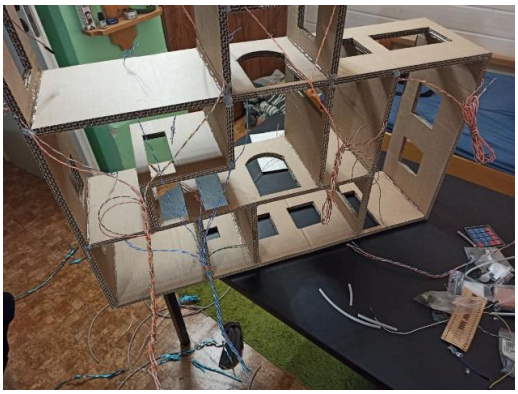
6. ábra Éjszakai nézet 3d-ben
(forrás: saját szerkesztés)

Miután, a tervezést befejeztük és a szükséges útmutatóként szolgáló ábrákat kinyomtattuk megkezdhetjük a modell megépítését. Ehhez 1.5 centiméter vastagságú kartonlapot fogunk

használni. Ebből fogjuk kialakítani a ház falait és a helységeket is egyaránt. A méreteket mi magunk határozhatjuk meg, ebben az esetben a falak magassága 20 centiméter. A kartonlap egy tulajdonság most nagyon a hasznunkra válik, mivel üreges kialakítású ezért a szenzorok, motorok ledek és egyéb eszközökre csatlakoztatott vezetékeket ezekben az üregekben szépen eltudjuk rejteni. A vezeték kiválasztásánál az UTP kábel mellet döntöttünk, mivel könnyen használható és mivel réz vezetékről beszelünk vezetőképessége is nagyon jó. A sorkapocsba kötésnél jó, ha a vezeték egy tömör maggal rendelkezik, érdemes ezt a tulajdonságot is fontolóra venni. Az építés jelenlegi állapota a 7.ábrán látható.

7. ábra

Következhet a kábelezes része, miután eldöntöttük, hova szeretnénk elhelyezni a szenzorokat és a led diódákat nincs más dolgunk, mint megfelelő hosszúságú vezetéket húznunk a kartonlap üregein keresztül, majd némi tartalék ráhagyásával méretre vágnunk azokat.



8. ábra a vezetékek elhelyezése
(forrás: saját szerkesztés)

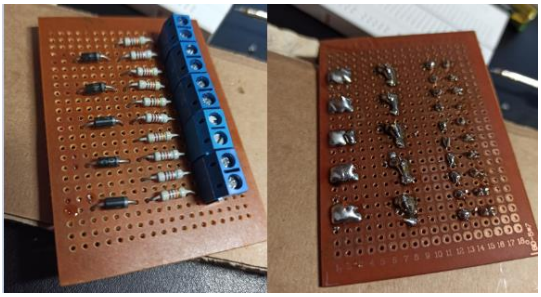
Jöhetnek a szenzorok, fontos észben tartani, hogy melyik szenzor mennyi pinnel rendelkezik és ezek szerint megadni nekik a kellő mennyiségű vezetéket. Például a PIR mozgásérzékelő három csatlakozóval rendelkezik, egy Vcc a tápfeszültség fogadására egy Gnd a földelésre és egy Data kimenet melyen a szenzorból kapott jel érkezik. Tehát ennek a szenzornak három szál vezetékre van szüksége. Érdemes továbbá a vezeték színeit használni, mint jelölést, például a sötét színek a tápellátást biztosítják a világosak pedig az adatok küldésére szolgálnak. A szenzorok elhelyezését követően a modell így néz ki (9.ábra).

9. ábra A szenzorok elhelyezése
(forrás: saját szerkesztés)

Szerencsére a modell alapját képező styrofoam alkalmas arra, hogy csavarokat csavarjunk bele, így tehát a sorkapcsokat biztosan tudjuk rögzíteni ez által a kisebb sérüléseknek kábel beakadásoknak is ellen tudnak állni. Igyekezzünk a kábeleket jól elrendezni, mert szükségünk lesz a helyre és fontos, hogy követhető legyen a vezetékek iránya is. A jelenlegi kábel elrendezést a következő 9.ábrán láthatjuk.

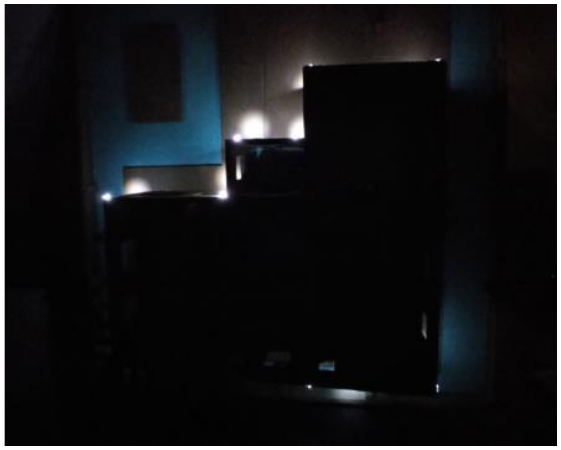
10. ábra a kábelezés jelenlegi állapota

A már említésre került védőáramköröket is megépítjük, ez nagyon egyszerű mivel csak egy ellenállást kell beiktatnunk a LED diódák pozitív csatlakozójára, vagyis anódjára. Ehhez szükségünk lesz egy univerzális nyomtatott áramkör lapra, melyen egyszerűen elhelyezzük egymás mellé a megfelelő méretű ellenállásainkat és forrasztópáka segítségével ráforassztjuk azokat az áramkör lapra, ez esetben elhelyezünk még néhány egyenirányító diódát is az áramkörön az esetleges polaritás cseré általi meghibásodás kiküszöbölése érdekében, végül soros kapcsokat is elhelyezünk az egyszerű csatlakozás érdekében.



11. ábra Védeárankör
(forrás: saját szerkesztés)

Hogy biztosak legyünk abban minden alkatrészhez eljut a feszültség kapcsoljunk, 5 volt egyenfeszültséget a védőáramkörünk bemenetére és csatlakoztassuk rá a világításként szolgáló LED diódákat. Amennyiben nem cseréltük fel az anódot a katóddal akkor a LED diódák világítanak (12.ábra).



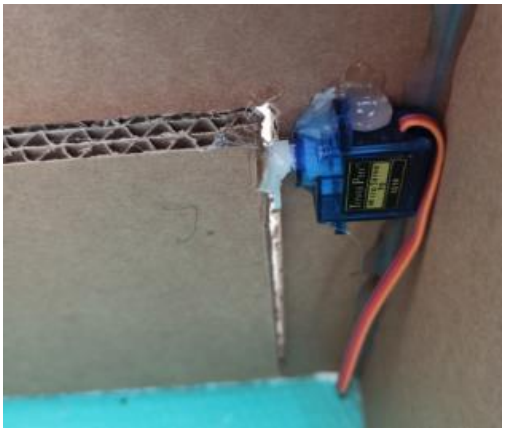
12. ábra LED diódák világítás közben
(forrás: saját szerkesztés)

Bizonyos szenzorok, a mi esetünkben az MQ2-es és a DHT11-es szenzor is rendelkezik egy led diódával mely az aktív állapotát jelzi, vagyis, ha a Vcc csatlakozójukon 5 volt feszültség jelenik meg akkor világítanak.

Következő lépésekben építsük meg a kerítést és helyezzünk minden sarkára egy-egy LED diódát. Később ezeket is szeretnénk ki vagy bekapcsolni bizonyos körülmények között. A kerítés építése közben alakítsuk ki a modell első részén a kaput, olyan módon, hogy az alkalmas legyen bizonyos mozgatásra mivel azt szeretnénk, hogy egy szervomotor ezt a nyitási mozgást elvégezze, ugyanígy járjunk el a garázsajtó esetében is mivel azt is vezérelni szeretnénk. Miután ezt megoldottunk elhelyezhetjük a két Tower Pro SG90 típusú szervomotorunkat is. A bejárati kapu nyitását megvalósító mechanikai kapcsolat és a szervomotor elhelyezkedés a következő megoldás szerint valósulhat meg (13.ábra).

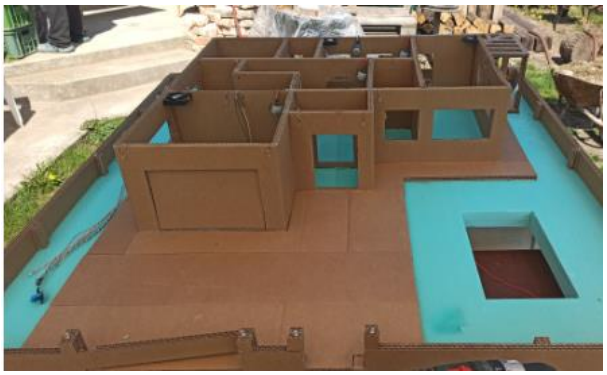
13. ábra kapunyitás mechanizmus
(forrás: saját szerkesztés)

Ugyanígy kell eljárnunk a garázsajtó esetében is, bár itt egyszerűbben megoldható a probléma. A szervomotorok melyeket használunk rendelkeznek egy műanyag elemmel, mely tökéletesen illeszkedik a motor tengelyén elhelyezett fogaskerékre, így tehát ennek a levétele és a garázsajtóra ragasztását követően csak egy fapálcikára van szükségünk a billenő ajtó kialakításához. Ennek a megoldása, valamint a motor elhelyezése így valósítható meg (14.ábra).



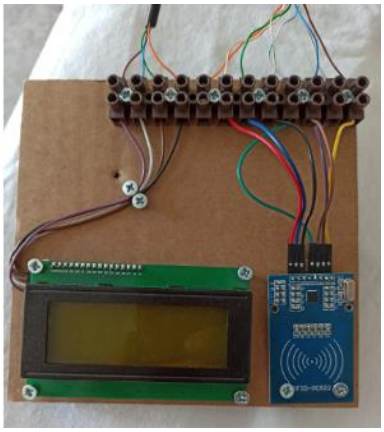
14. ábra garázsajtó mechanizmus

Miután ezeket a lépéseket is véghez vittük a modellünk udvarán kialakítunk egy medencét is (15.ábra). Ezt egyszerűen az alapként használt, styrofoam lap kivágásával oldhatjuk, fontos, hogy a modell alján található vezetékekre figyeljünk, mivel könnyen előfordulhat, hogy kihúzódnak a sorkapocsból vagy legrosszabb esetben el is törnek. A medence köré később világítást is elhelyezzük.



15. ábra a medence létrehozása
(forrás: saját szerkesztés)

Következhet a belépésre és információ megjelenítésre szolgáló panel megépítése. Itt szükségünk lesz az MFRC522-es RFID olvasóra és a 3x20 karakteres LCD kijelzőre I2C adapterrel. Ezt a panelt nem rögzítjük a modellünkhöz, azt szeretnénk, hogy áthelyezhető és mozgatható legyen. Ezért hosszabb vezetékkel is látjuk el. A terminál alapját egy karton lap képezi melyre egyszerűen felcsavarozható az RFID olvasó és az LCD kijelző is egyaránt. Az elkészült terminál a következő 16. ábrán látható.

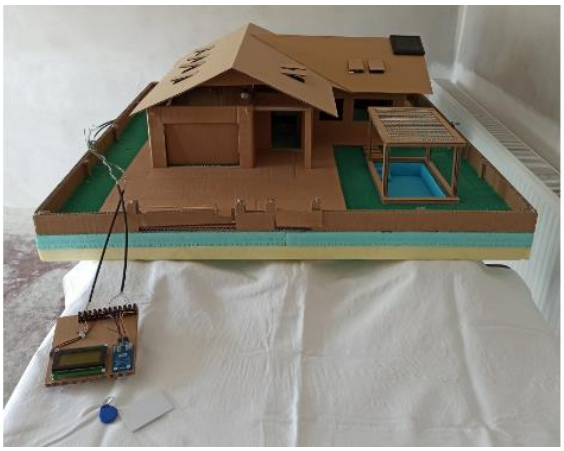


16. ábra Az azonosító terminál
(forrás: saját szerkesztés)

Amennyiben minden lépést precízen elvégeztünk, a modellünk majdnem kész. Csupán néhány apró módosítást és szépítést kell elvégeznünk. Készítsünk fedelet is házra, ahol helyezzünk el egy napkollektort, melyet akár a későbbiekben hasznosíthatunk a modellünkön.

17. ábra A modell fedél nélkül
(forrás: saját szerkesztés)

A feljebb elhelyezett ábrán látható, hogy némi zöld és kék szövet segítségével egész ügyesen utánozható a szép gyep és a medencében lévő víz is. A fedél elkészültével a modellünket késznek tekinthetjük (18.ábra).



18.ábra A kész modell
(forrás: saját szerkesztés)

Kezdődhet a modell életre keltése, vagyis az egészet működtető vezérlőegység elhelyezése és a programozás megkezdése. Elsőként a könyvtárak implementálásával indítunk. Mivel tudjuk, hogy az MFRC522-es RFID olvasó modul az SPI buszrendszert használja kommunikációra az Arduinoval ezért elsőként helyezzük el az SPI.h header fájlt. Ezután az LCD kijelző könyvtárát illesszük be mely a LiquidCrystal_I2C.h fájlt igényli, jöhet az MFRC522-es olvasó modul könyvtára az MFRC522.h. Mint tudjuk az LCD kijelzőnk az I2C buszrendszer kommunikál ezért nem szabad elfelejtenünk a Wire.h könyvtárat sem. A projektben használt szenzorok számára pedig importáljuk be az MQ2.h és a dht.h fájlokat is. Végül a szervomotorok által használt Servo.h-t sem hagyjuk ki. Miután a könyvtárakat elhelyeztük, végezzük el a szükséges típus, illetve pin definiálásokat is. Ez esetben az SPI busz SS pinje az Arduino Mega 2560 9- az RST pedig a 8-as digitális ki- és bementre csatlakozik. A DHT11-es szenzor által használt bemenet pedig a 47-es sorszámú lesz.

```
#include<SPI.h> //Az SPI kommunikációs protokoll használatához szükséges könyvtár
#include<LiquidCrystal_I2C.h> //Az I2C Folyékony kristályos LCD kijelző kezelő könyvtára
#include<MFRC522.h> //Az RFID olvasó használatához szükséges könyvtár
#define SS_PIN 9 //Az SS Soros bemeneti port definiálása
#define RST_PIN 8 //AZ RST vagyis nullázó port definiálása
#include<MQ2.h> //Az MQ2-es szenzor használatához szükséges könyvtár
#include<dht.h> //A DHT11-es szenzor használatához szükséges könyvtár
#include<Wire.h> //Wire könyvtár beillesztése az I2C busz használatához
#include<Servo.h> //A szervomotor vezérléséhez szükséges könyvtár beillesztése
#define DHT11_PIN 47 //A DHT11-es szenzor adat pinjének a definiálása
MFRC522 mfrc522 (SS_PIN, RST_PIN); //Típus definiálás az MFRC522-es RFID olvasónak
LiquidCrystal_I2C lcd(0x27, 20, 4); //Az általunk használt kijelző karakterkészlete 20 karakter és 4 sor
Servo kapumotor; //Servo típusú változó mely a kaput nyitó szervomotorra mutat
Servo garazsmotor; //Servo típusú változó mely a garázst nyitó szervomotorra mutat
dht DHT; //A dht könyvtárból használt egy objektum
int pir1 = 2; //Az 1-es PIR szenzor pinjét tartalmazó globális integer típusú változó
int pir2 = 3; //A 2-es PIR szenzor pinjét tartalmazó globális integer típusú változó
int pir3 = 18; //A 3-as PIR szenzor pinjét tartalmazó globális integer típusú változó
int pir4 = 19; //A 4-es PIR szenzor pinjét tartalmazó globális integer típusú változó
int trigger = 0; //ISR Flag változó a megszakítások ütemezésére
int nappaliLed = 30; //A nappali világítás
int konyhaLed = 28; //A konyhai világítás
int folyosoLed1 = 24; //A folyosó1 világítás
int folyosoLed2 = 25; //A folyosó2 világítás
int garazsLed = 23; //A garázs világítás
int medence1 = 32; //A medence1 világítás
int medence2 = 33; //A medence2 világítás
int kintiRele = 43; //A ház kinti világítását kapcsoló relé
int keritesRele = 40; //A kerítés világítását kapcsoló relé
int kiskapuRele = 41; //A kiskapu villogtatását kapcsoló relé
int garazskapuRele = 42; //A garázskapu villogtatását kapcsoló relé
int szelloztetesRele = 44; //A ház szellőztetését kapcsoló relé
int garazsRele = 45; //A garázs szellőztetését kapcsoló relé
int piezo = 48; //Piezo elem
int light = 0;//Fényérték meghatározása
int potmeter = A0; //fenyérték állítása, sötét/világos szimulálása
int lpg, co, smoke; //globális integer típusú változók az lpg, co, és a fust értékenk a tárolására
int gas = A2; //Az MQ2-es szenzor pinjének tárolása egy globális integer típusú változóban
```


Tovább haladhatunk és folytathatjuk a változók deklarálásával. Fontos mivel ez a projekt elég nagy számú változót tartalmaz, hogy ezek nevét úgy válasszuk ki, hogy egyértelmű legyen mire utalnak. Elsőként kezdjük a négy darab mozgásérzékelő PIR szenzor deklarálásával. Ezek az Arduino 2-es, 3-as, 18-as és 19-es pinjein kapnak helyet. Vajon miért nem négy egymás melletti pinre csatlakoztatjuk ezeket? Erre a kérdésre később adunk magyarázatot. Ezután létrehozunk egy ún. flag változót melyet trigger-nek nevezünk, ennek a változónak a szerepe a későbbiekben fog kiderülni. Következhet a különböző helységek és helyek világításának a megcímezése. Minden helységben egy darab led diódát helyeztünk el ezért az ide tartozó változók a következők, nappaliLed, konyhaLed, folyosoLed1, folyosóLed2, garazsLed. Majd a medence világításánál a meden1 és medence2 változók lesznek létrehozva. Ami a nyolccsatornás relé modult illeti a jelenlegi esetben hat darab relére lesz szükségünk. Az első relé a ház kinti világítását fogja kapcsolni erre a kintiRele változóval tudunk hivatkozni. A kerítés megvilágításáról gondoskodik a második relé melyet a keritesRele néven érhetünk el. A bejárati kapunál és a garázskapunál elhelyezett villogó áramkörrel rendelkező Led diódákat a kiskapuRele és a garazskapuRele

azonosítókkal láttuk el. Végül ház és a garázs szellőztetésére használt reléket a szelloztetesRele és a garazsRele azonosító alatt találjuk meg. Amennyiben hangjelzés kibocsátásara alkalmas piezo elemet is használunk azt a piezo változónév alatt érhetjük el. Következhetnek azok a változók melyek az Arduino Mega 2560 valamennyi bemenetére csatlakoztatott szenzorok bizonyos pinjeire hivatkoznak. Elsőként egy potenciométerrel kezdünk melyet az Arduino A0-sorszámú analóg bemenetére csatlakoztatunk ez a potenciométer szolgálhat, mint teszt eszköz bemeneti jel értékének a változtatására viszont később más funkcióval látjuk el. Az MQ2-es szenzor értékének kiolvasásához szükséges három változót hozzuk létre, melyek az lpg, co és smoke néven érhetőek el. E szenzor az Arduino A2-es sorszámú analóg bemenetére küldi majd a megfelelő jeleket.

```
MQ2 mq2(gas); //Az MQ2 szenzor inicializálása és az általa használt bemenet hozzárendelése
int trigPin = 10; //globális integer típusú változó mely az Ultrahangos szenzor trigger pin-jét tárolja
int echoPin = 11; //globális integer típusú változó mely az Ultrahangos szenzor trigger pin-jét tárolja
long duration; //long típusú változó mely az időtartam értékét tárolja
int distance; //integer típusú változó mely a távolság értékét tárolja
int safetyDistance; //integer típusú változó mely a biztonságos távolság értékét tárolja
int dht11 = 47; //globális integer típusú változó mely a DHT11 szenzor pin-jét tárolja
int p = 0; //globális integer típusú változó mely a szervomotor pozícióját tartalmazza
boolean card = false; //Boolean típusú változó mely a kártya jelenlétét jelzi
boolean sotet = true; //Boolean típusú változó mely a sötétséget igazolja vagy cáfolja
void setup()
{
  pinMode(pir1, INPUT); //Az 1-es PIR szenzor pinjének bemenetté alakítása(Nappali)
  pinMode(pir2, INPUT); //A 2-es PIR szenzor pinjének bemenetté alakítása(Konyha)
  pinMode(pir3, INPUT); //A 3-as PIR szenzor pinjének bemenetté alakítása(Folyosó)
  pinMode(pir4, INPUT); //A 4-es PIR szenzor pinjének bemenetté alakítása(Garázs)
  pinMode(nappaliLed, OUTPUT); //A nappali vilagitasának kimenetté alakítása
  pinMode(konyhaLed, OUTPUT); //konyha vilagitasának kimenetté alakítása
  pinMode(folyosoLed1, OUTPUT); //folyoso1 vilagitasának kimenetté alakítása
  pinMode(folyosoLed2, OUTPUT); //folyoso2 vilagitasának kimenetté alakítása
  pinMode(garazsLed, OUTPUT); //garazs vilagitasának kimenetté alakítása
  pinMode(medence1, OUTPUT); //medence1 vilagitasának kimenetté alakítása
  pinMode(medence2, OUTPUT);  //medence2 vilagitasának kimenetté alakítása
  pinMode(kintiRele, OUTPUT); //A kinti vilagitás reléjének kimenetté alakítása
  pinMode(keritesRele, OUTPUT); //A kerítés vilagitás reléjének kimenetté alakítása
  pinMode(kiskapuRele, OUTPUT); //A kiskapu vilagitás reléjének kimenetté alakítása
  pinMode(garazskapuRele, OUTPUT); //A garázskapuvilagitás reléjének kimenetté alakítása
  pinMode(szelloztetesRele, OUTPUT); //A szellőztetés reléjének kimenetté alakítása
  pinMode(garazsRele, OUTPUT); //A garázs szellőztetés reléjének kimenetté alakítása
  pinMode(piezo, OUTPUT);//A piezo elemet tartalmazó pin kimenetté alakítása
  pinMode(gas, INPUT); //Az MQ2-es szenzor által használt pin bemenetté alakítása
  pinMode(potmeter, INPUT); //A fényérték szimulálására használt potenciométer bemenetté alakítása
  pinMode(trigPin, OUTPUT); //A szenzor trigger pinje, mint digitális kimenet
  pinMode(echoPin, INPUT); //A szenzor echo pinje, mint digitális bemenet
  pinMode(dht11, INPUT); //A szenzor adat pinje, mint digitális bemenet
  digitalWrite(nappaliLed, LOW); //A nappali világítás kikapcsolt állapotban
  digitalWrite(konyhaLed, LOW); //A konyhai világítás kikapcsolt állapotban
  digitalWrite(folyosoLed1, LOW); //A folyosó1 világítás kikapcsolt állapotban
```


Majd a szenzor inicializálását sem szabad elfelejtenünk. Ha ezt megtettük hozzunk létre két szintén integer típusú változót a HC-SR04-es ultrahangos távolságmérő trigger és echo csatlakozói számára, valamint itt még szükségünk van egy időtartamot mérő long típusú változóra és a távolság értékét tartalmazó integer változóra is. Ami a hőmérséklet és páratartalom mérését illeti a DHT11-

es szenzorunk által használt digitális bemenet a 47-es sorszámú lesz. Ami a projekt mechanikai részét jelenti, tudjuk, hogy a szervomotorjainknak szüksége van egy változóra mely a motorpozíció értékét tárolja, ezért létrehozunk egy *p* integer típusú változót is erre a célra. Végül deklarálunk kettő darab bináris vagy Boolean típusú változót is melyek közül az egyik az RFID azonosításnál lesz hasznunkra a másik pedig fény vizsgálatára szolgál mi szerint, ha világos van akkor a változó hamis értéket vesz fel az ellenkező esetben pedig igaz értékkel bír. Ezzel a változók deklarálását be is fejeztük és az előzőleg megalkotott projektekből tudjuk, hogy mivel a programunk void setup szekciója előtt kerültek deklarálásra ezért az összes létrehozott változó globálisan elérhető. A következő lépésekben meghatározzuk, hogy az Arduino Mega 2560 egyes pinjei kimenetként vagy bemenetként üzemeljenek. Mint tudjuk az összes szenzorunk bemenetként működik ám akad közöttük egy melynek szüksége van egy kimenethez is a működéséhez, ez nem más, mint a HC-SR04-es ultrahangos távolságmérőnk echo csatlakozója, ezt nem árt az elején tisztázni még mielőtt megfeledkeznénk róla. Visszatérve az elejére kezdjük a mozgásérzékelő PIR szenzorokkal, tudjuk, hogy ezek egy darab adat küldésére szolgáló csatlakozóval rendelkeznek, ebből adódóan az általuk használt pineket bemenetté kell alakítanunk. Következnek a világítást, villogást, hangjelzést és reléket vezérlő pinek. Nyilvánvaló, hogy ezek pinjeit kimenetté kell alakítanunk. Végezetül a potenciométer, a DHT11-es szenzor, az MQ2-es szenzor és a HC-SR04-es távolságmérő echo pinjét alakítsuk bemenetté. Ezzel a ki- illetve bemenetek meghatározása meg is történt.

```
digitalWrite(folyosoLed2, LOW); //A folyosó2 világítás kikapcsolt állapotban
digitalWrite(garazsLed, LOW); //A garázs világítás kikapcsolt állapotban
digitalWrite(medence1, LOW); //A medence1 világítás kikapcsolt állapotban
digitalWrite(medence2, LOW); //A medence2 világítás kikapcsolt állapotban
digitalWrite(kintiRele, HIGH); //A kinti vilagitas kikapcsolt állapotban (Relé HIGH = OFF)
digitalWrite(keritesRele, HIGH); //A kerítés vilagitas kikapcsolt állapotban (Relé HIGH = OFF)
digitalWrite(kiskapuRele, HIGH); //A kiskapu villogas kikapcsolt állapotban (Relé HIGH = OFF)
digitalWrite(garazskapuRele, HIGH); //A garázskapu villogas kikapcsolt állapotban (Relé HIGH = OFF)
digitalWrite(szelloztetesRele, HIGH); //A szellőztetés kikapcsolt állapotban (Relé HIGH = OFF)
digitalWrite(garazsRele, HIGH); //A garázs szellőztetés kikapcsolt állapotban (Relé HIGH = OFF)
Serial.begin(9600); //A soros porton történő kommunikáció bitrátája
mq2.begin(); //Az MQ2-es szenzorral való kommunikáció indítása
SPI.begin(); //Az SPI buszon történő kommunikáció indítása
mfrc522.PCD_Init(); //Az MFRC522-es RFID olvasó modul inicializálása
lcd.init(); //Az LCD kijelző inicializálása
lcd.backlight(); //Az LCD kijelző háttérvilágításának bekapcsolása
lcd.setCursor(4, 0); //Kurzor pozicionálás ez esetben 4. karakter a 0. sorban
lcd.print("UDVOZOLJUK"); //Megadott karakterlánc kiíratása
lcd.setCursor(8, 1); //Kurzor pozicionálás ez esetben 8. karakter a 1. sorban
lcd.print("A"); //Megadott karakterlánc kiíratása
Serial.print("A rendszer aktív\n"); //Megadott karakterlánc kiíratása a soros portra
lcd.setCursor(2, 2); //Kurzor pozicionálás ez esetben 2. karakter a 2. sorban
lcd.print("RENDSZER AKTIV"); //Megadott karakterlánc kiíratása
lcd.setCursor(8, 3); //Kurzor pozicionálás ez esetben 8. karakter a 3. sorban
lcd.print(":)"); //Megadott karakterlánc kiíratása
delay(2000); //Várakozás 2 másodpercig
Serial.println("Érintse a kártyáját:"); //Megadott karakterlánc kiíratása
lcd.clear(); //Az LCD kijelző tartalmának a törlése
lcd.setCursor(8, 0); //Kurzor pozicionálás ez esetben 8. karakter a 0. sorban
lcd.print("A"); //Megadott karakterlánc kiíratása
lcd.setCursor(4, 1); //Kurzor pozicionálás ez esetben 4. karakter a 1. sorban
lcd.print("KARYTAJAT"); //Megadott karakterlánc kiíratása
lcd.setCursor(4, 2); //Kurzor pozicionálás ez esetben 4. karakter a 2. sorban
lcd.print("ERINTSE A"); //Megadott karakterlánc kiíratása
lcd.setCursor(3, 3); //Kurzor pozicionálás ez esetben 3. karakter a 3. sorban
lcd.print("TERMINALHOZ"); //Megadott karakterlánc kiíratása
delay(500); //Várakozás 500 milliszekundum ideig
lcd.setCursor(14, 3); //Kurzor pozicionálás ez esetben 14. karakter a 3. sorban
```


Tovább haladva a programunk void setup szekciójában a következő feladatunk az egyes kimenetek alacsony, illetve magas kezdeti értékének a megadása, ezt biztonsági okokból is el kell végeznünk, mivel nem szeretnénk, ha például valamely kikapcsolt állapot helyett aktív lenne. Ez a helyzet a világítással is. A program indulásánál nem szeretnénk, hogy valamelyik Led diódánk világítson mivel nem adtunk erre okot. Ezért érdemes az ezekre csatlakoztatott kimenetek kezdő értékét alacsonyra vagyis LOW értékre állítani, mint tudjuk ezt egy egyszerű digitalWrite paranccsal tudjuk megvalósítani. A világítás mellet a reléket is állítsuk inaktív üzemmódba, ám bár itt a ledekkel ellentéteséén a magas vagyis HIGH állapotba kell őket helyeznünk. Mivel a nyolccsatornás relé modulunk olyan üzemmódba van állítva, hogy a kiment alacsony vagyis LOW jelét érzékelve lép aktív módba magas jel esetén pedig inaktívvá válik. Ezt szintén biztonsági és egyéb okokból használjuk, mivel a programkódban egyszerűbb kiigazodást is biztosít egyben. Tehát egy példán bemutatva a világítás inaktív állapotát a digitalWrite (nappaliLed, LOW) parancs segítségével a relék inaktív állapotát pedig a digitalWrite (kintiRele, HIGH) parancs segítségével érhetjük el.

A program következő szekciójában néhány inicializációs folyamatot fogunk elvégezni, kezdve a Soros port az MQ2-es szenzor az SPI busz az MFRC522-es RFID olvasó és az I2C buszra csatlakozó LCD kijelző inicializálásával és kapcsolatának a létesítésével. Szokás szerint a soros port bitrátájának az értéke még mindig változatlan 9600-as baud érték. Ha ezt megtettük aktiváljuk az LCD kijelző hattérvilágítását mivel információt szeretnénk rajta megjeleníteni. Az üzenet pedig a következő, *ÜDVÖZÖLJÜK A RENDSZER AKTÍV*. Az üzenetet úgy szeretnénk megjeleníteni, hogy a 3x20 karakteres LCD kijelzőnk mindhárom sorában szerepeljen valami. Ezért a kijelző kurzorát ennek megfelelően kell pozícionálnunk először a 0-ik sor 4-ik karakterére helyezzük és megjelenítjük az *ÜDVÖZÖLJÜK* szót, ezután a kijelző 1-es sorában egy *A* karaktert és így tovább egészen a 3-ik sorban elhelyezett hangulatjelig. Az üzenet két másodpercig válik láthatóvá azután az azonosításra használt RFID kártya odahelyezésére utasító üzenet jelenik meg.

```
  lcd.print("."); //Megadott karakterlánc kiíratása
  delay(700); //Várakozás 700 milliszekundum ideig
  lcd.setCursor(15, 3); //Kurzor pozicionálás ez esetben 15. karakter a 3. sorban
  lcd.print("."); //Megadott karakterlánc kiíratása
  delay(700); //Várakozás 700 milliszekundum ideig
  lcd.setCursor(16, 3); //Kurzor pozicionálás ez esetben 16. karakter a 3. sorban
  lcd.print("."); //Megadott karakterlánc kiíratása
  noTone(piezo);
  attachInterrupt(digitalPinToInterrupt(pir1), motion1, RISING);//Nappali PIR megszakítás funkció
  attachInterrupt(digitalPinToInterrupt(pir2), motion2, RISING);//Konyha PIR megszakítás funkció
  attachInterrupt(digitalPinToInterrupt(pir3), motion3, RISING);//Folyosó PIR megszakítás funkció
  attachInterrupt(digitalPinToInterrupt(pir4), motion4, RISING);//Garázs PIR megszakítás funkció

}
void motion1() //ISR funkció mozgás észlelésére a nappaliban
{
  trigger = 1; //flag változó értékének változtatása
}
void motion2() //ISR funkció mozgás észlelésére a konyhában
{
  trigger = 2; //flag változó értékének változtatása
}
void motion3() //ISR funkció mozgás észlelésére a folyosón
{
  trigger = 3; //flag változó értékének változtatása
}
void motion4() //ISR funkció mozgás észlelésére a garázsban
{
  trigger = 4; //flag változó értékének változtatása
}
void temperature() //Funkció a hőmérséklet értékének kiíratására
{
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(1, 1); //Kurzor pozicionálás ez esetben 1. karakter a 1. sorban
  lcd.print("A HOMERSEKLET:"); //Megadott karakterlánc kiíratása
  lcd.setCursor(3, 2); //Kurzor pozicionálás ez esetben 3. karakter a 2. sorban
  lcd.print(DHT.temperature); //A hőmérséklet értékének kiíratása
  delay(5000); //Várakozás 5 másodpercig
}
```


Az *ÉRINTSE A KÁRTYÁJÁT* üzenet után három . karakter egymás után 700 milliszekundum időzítést követően való kiíratásával szimuláljuk a kártya érintésére váró folyamatot.

Ami a mozgásérzékelő PIR szenzorok által használt pineket illeti most kerül megmagyarázatra, az, hogy miért nem tetszés szerint választottuk ki, hogy hova csatlakoztatjuk őket. Belegondolva és az előző projektekben tapasztalva rájöhettünk, hogy az Arduino programának ciklusa futása közben történő változások csak a ciklus következő ujraindulásában mutatkoznak, ezért az Arduino egy számunkra új eddig még nem használt funkcióját fogjuk igénybe venni, ami nem mást, mint az interrupt vagyis a megszakítás. Az eszközünk rendelkezik bizonyos pinekkel melyek képesek a program futása közben megjelenő ún. megszakító jelek fogadására. Ezek a megszakító jelek pedig képesek beavatkozni a programkódunkba. A megszakítást követő eljárást ISR-nek vagyis *Interrupt Service Routine*-nek nevezik. Ennek a definiálási formája a következő, *attachInterrupt(digitalPinToInterrupt(pin), funkció, és a jel impulzus érzékelésének a módja).* Ami a jel érzékelésénének a módját illeti létezik, *LOW*, *HIGH*, *FALLING*, és *RISING*. Tesztelést követően a nekünk megfelelő a *RISING* vagyis, ha a megszakítás képes bemenetre érkező jel az aktuálishoz képest növekszik akkor megy végbe a megszakításhoz létrehozott funkció. Minden egyes megszakítást egy funkció hívása követ. A mi esetünkben ezek a funkciók a *motion1(), motion2(), motion3(),* és a *motion4().* Mivel négy darab PIR szenzorunk van ezért négy funkciót hozunk létre, ahol a már említett *trigger* nevű flag változónk értékét változtatjuk és a programunk ennek a változónak megfelelően jár el a riasztás megvalósításánál.

A megszakító funkciók létrehozását egy újabb funkció megalkotása követi, mégpedig a *temperature ()* funkcióé. Ez a funkció arra szolgál, hogy a DHT11-es szenzortól kapott jel azon részét mely a hőmérséklet értékét hordozza megjelenjen az LCD kijelzőnkön öt másodpercnyi időre.

```
void humidity() //Funkció a páratartalom értékének a kiíratására
{
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(1, 1); //Kurzor pozicionálás ez esetben 1. karakter a 1. sorban
  lcd.print("A PARATARTALOM"); //Megadott karakterlánc kiíratása
  lcd.setCursor(3, 2); //Kurzor pozicionálás ez esetben 3. karakter a 2. sorban
  lcd.print(DHT.humidity); //A páratartalom értékének kiíratása
  delay(5000); //Várakozás 5 másodpercig
}
void checktemp() //Funkció hőmérséklet értékének a kiolvasására a DHT11-ből
{
  int chk = DHT.read11(DHT11_PIN); //A DHT11 értékének a kiolvasása és segédváltozóban tárolása
  if (DHT.temperature >= 25) //Feltétel vizsgálat:
  {
    lcd.clear(); //Az LCD kijelző tartalmának a törlése
    lcd.setCursor(3, 1); //Kurzor pozicionálás ez esetben 3. karakter a 1. sorban
    lcd.print("HUTES INDITASA"); //Megadott karakterlánc kiíratása
    digitalWrite(szelloztetesRele, LOW); //A hűtés bekapcsolása
    delay(10000); //Váralkozás 10 másodpercig
  }
  else //Egyébként
  {
    digitalWrite(szelloztetesRele, HIGH); //hűtés kikapcsolvva
  }
}
void fume() //Funkció a gazszivárgás megjelenítésére
{
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(1, 1); //Kurzor pozicionálás ez esetben 1. karakter a 1. sorban
  lcd.print("NINCS GAZSZIVARGAS"); //Megadott karakterlánc kiíratása
  delay(5000); //Várakozás 5 másodpercig
  trigger = 0; //Flag változó értékének változtatása
}
void leakage() //Funkció gázszivárgás észlelésére
{
  lpg = mq2.readLPG(); //Az lpg értékének kiolvasása és változóban tárolása
  co = mq2.readCO(); //A co értékének kiolvasása és változóban tárolása
  smoke = mq2.readSmoke(); //A füst értékének kiolvasása és változóban tárolása
```


Ezután a páratartalom értékének a kiíratására és létrehozunk egy funkciót melyet *humidity ()* azonosító név alatt helyezünk el a programunkban. Ez a funkció az előző *temperature ()* -hez hasonlóan működik azzal a különbséggel, hogy ez a DHT11-es szenzortól a páratartalom értékét jeleníti meg az LCD kijelzőn. Szintén öt másodperc erejéig. Következő lépésben hozzunk létre egy funkciót mely a ház hűtésének a működtetését valósítsa meg. Nevezzük ezt *checktemp ()* -nek. Ebben a funkcióban a DHT11-es szenzor által visszaadott aktuális hőmérséklet értékét fogjuk egy feltétel vizsgálattal ellenőrizni, és ha a mért hőmérséklet eléri a 25 Celsius fokot akkor a ház hűtési rendszere aktívvá válik mondjuk tíz másodpercnyi időre. Majd az idő letelése után inaktív állapotba kerül. Tehát a relé mely a szellőztetést és a hűtést végző ventilátort kapcsolja aktívvá válik, vagyis az értéke *LOW* kell, hogy legyen. A feltétel vizsgálat *else* ágában a relé értéke HIGH kell, hogy legyen.

A következő funkció *fume ()* mely létrehozásra kerül nem mást, mint az MQ2-es szenzor által kiíratott figyelmeztetést megvalósító funkció. Ennek a funkciónak a lényege, hogy amíg a szenzorunk nem érzékel szénmonoxid, lpg gáz vagy füst szivárgást akkor addig a *NINCS*

GÁZSZIVÁRGÁS üzenetet jeleníti meg az LCD kijelzőn. Amennyiben a szenzor által mért értékek elérnek egy bizonyos küszöbértéket akkor a következő funkció lép érvénybe.

A soron következő funkció mely a *leakage ()* azonosítóval rendelkezik végzi az MQ2-es szenzor értékének az olvasását minden egyes meghíváskor, így biztosítva, hogy a szenzortól visszakapott érték mindig a legfrissebb érték legyen.

```
  if (smoke > 10) //Feltétel vizsgálat:
  {
    digitalWrite(szelloztetesRele, LOW); //A szellőztetés indítása
    lcd.clear(); //Az LCD kijelző tartalmának a törlése
    lcd.setCursor(3, 1); //Kurzor pozicionálás ez esetben 3. karakter a 1. sorban
    lcd.print("GAZSZIVARGAS"); //Megadott karakterlánc kiíratása
    delay(10000); //Várakozás 10 másodpercig
    trigger = 0; //Flag változó értékének változtatása
  }
  else //Egyébként
  {
    delay(1000); //Várakozás 1 másodpercig
    digitalWrite(szelloztetesRele, HIGH); //A szellőztetés kikapcsolva
  }
}
void beep () //Funkció hangjelzésre
{
  tone(piezo, 800); //A piezo elem megszólaltatása 800hz-en
  delay(100); //Szüneteltetés 100 milliszekundum ideig
  tone(piezo, 1000); //A piezo elem megszólaltatása 1000hz-en
  delay(100); //Szüneteltetés 100 milliszekundum ideig
  tone(piezo, 1200); //A piezo elem megszólaltatása 1200hz-en
  delay(100); //Szüneteltetés 100 milliszekundum ideig
  tone(piezo, 1400); //A piezo elem megszólaltatása 1400hz-en
  delay(100); //Szüneteltetés 100 milliszekundum ideig
  noTone(piezo); //A piezo elem elhallgattatása
}
void enter () //Funkció a belépés igazolására
{
  delay(1000); //Várakozás 1 másodpercig
  Serial.println("BELEPES ENGEDELYEZVE"); //Megadott karakterlánc kiíratása a soros portra
  Serial.println(); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("BELEPES ENGEDELYEZVE"); //Megadott karakterlánc kiíratása
  delay(2000); //Várakozás 2 másodpercig
  check(); //Check funkció meghívása
  card = true; //A változó értékének igaz-ra változtatása
  trigger = 0; //Flag változó értékének változtatása
}
```


Itt szintén egy feltétel vizsgálat segítségével döntjük, el, hogy szénmonoxid, lpg gáz vagy füst szivárgása esetén jelezzen a szenzorunk. A jelenlegi esetben mi a füst szivárgást szeretnénk figyelni, tehát ha az MQ2-es szenzor *smoke* vagyis a füst mért értékét tároló változó értéke eléri vagy nagyobb, mint az általunk meghatározott küszöbérték akkor fény és hangjelzés, valamint egy üzenet megjelenítése az LCD kijelző formájában figyelmeztetést kapunk arról, hogy a lakásunkban valószínűleg füst szivárgás észlelhető. S mivel a rendszerünk tartalmaz szellőztetést is ezért, ha gázszivárgás észlelés történik akkor aktiváljuk ezt, jelen esetben a hűtési rendszerhez hasonlóan

szintén tíz másodpercnyi időre. Amennyiben a mért értékünk nem éri el a feltétel vizsgálatban meghatározott küszöbértékek akkor az *else* ág kerül futtatásra, ahol a szellőztetés állapota inaktív marad.

Amennyiben a projektünkben használunk piezo hangsugárzó, akkor érdemes készíteni egy hangjelzést megvalósító funkciót is. Mi ezt egyszerűen *beep ()* -nek nevezzük el. Ebben a funkcióban nincs más dolgunk, mint egy tetszőleges hangjelzést létrehozni a *tone* paranccsal és a frekvencia tartomány beállításával.

Tovább haladva a funkciók között a következő *enter ()* funkció az RFID kártyánk olvasása után meghívásra kerülő két funkció közül az egyik. Mégpedig az, amely az RFID azonosítás helyességét igazolja. Ez a funkció nem mást tesz, mint megjeleníti az LCD kijelzőn a *BELPÉPES ENGEDÉLYEZVE* üzenetet, majd két másodpercnyi várakozás után a riasztó rendszer jelenlegi állapotát figyelő és visszajelző funkciót hívja meg. Amint a meghívott funkció lefut a bináris típusú *card* nevezetű változónk értékét igazra állítja.

```
void alarmcheck() //Funkció a riasztó állapotának ellenőrzésére
{
  if (sotet == true) //Feltétel vizsgálat:
  {
    lcd.clear(); //Az LCD kijelző tartalmának a törlése
    lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
    Serial.print("RIASZTO AKTIV\n"); //Megadott karakterlánc kiíratása a soros portra
    lcd.print("A RIASZTO AKTIV"); //Megadott karakterlánc kiíratása
    delay(2000); //Várakozás 2 másodpercig
  }
  else //Egyébként
  {
    lcd.clear(); //Az LCD kijelző tartalmának a törlése
    lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
    Serial.print("RIASZTO INAKTIV\n"); //Megadott karakterlánc kiíratása a soros portra
    lcd.print("A RIASZTO INAKTIV"); //Megadott karakterlánc kiíratása
    sotet = true; //A változó értékének igaz-ra változtatása
    delay(2000); //Várakozás 2 másodpercig
  }
}
void denied() //Funkció az elutasított belépés visszajelzésére
{
  delay(1000); //Várakozás 1 másodpercig
  Serial.println("BELEPES MEGTAGADVA"); //Megadott karakterlánc kiíratása a soros portra
  Serial.println(); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("BELEPES MEGTAGADVA"); //Megadott karakterlánc kiíratása
  card = false; //A változó értékének hamis-ra változtatása
}
```


Az *alarmcheck ()* funkció nevéből már gondolhatjuk, hogy ez a funkció a riasztórendszer ellenőrzésére szolgál. Pontosan ez a funkció jelez vissza nekünk az LCD kijelző segítségével arról, hogy a házunk biztonsági rendszer jelenleg aktív vagy inaktív állapotban van e. Amint már

említettük a programunk deklarációs részében, hogy rendelkezünk egy bináris típusú változóval mely a sötétség állapotát igazolja vagy cáfolja. S mivel úgy szeretnénk, hogy a mozgásérzékelés és riasztás csak sötétben legyen aktív ezért elhelyezünk ebbe a funkcióba egy feltétel vizsgálatot melynek igaz ága csak akkor kerül futtatásra, hogy ha a *sotet* nevezetű változónk igaz értékkel bír. Természetesen a riasztó állapotát megjelenítjük a kijelzőnkön is. Viszont, ha nincs sötét akkor a kijelzőn a *RIASZTÓ INAKTÍV* üzenet jelenik meg.

Ahogyan az *enter ()* funkció esetében a belépés igazolása úgy a jelenlegi, vagyis a *denied ()* funkcióban a belépés elutasításának a visszajelzése kerül megvalósításra egy üzenet megjelenítése, valamint a bináris típusú *card* változónk értékének a hamisra állítását követően.

```
void check() //Funkció a mozgás érzekelésére megszakítás portokon keresztül
{
  switch (trigger) //Több irányú elágazás a flag változó értékének megfelelően:
  {
    case 1: //Első eset ha a változó = 1
      Serial.print("Nappali mozgas\n"); //Megadott karakterlánc kiíratása a soros portra
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
      lcd.print("MOZGAS A NAPPALIBAN."); //Megadott karakterlánc kiíratása
      digitalWrite(medence1, HIGH); //A medence1 vilagít
      digitalWrite(medence2, HIGH); //A medence2 világít
      digitalWrite(kintiRele, LOW); //A kinti világitás bekapcsolva
      for (int i = 0; i <= 5; i++) //for ciklus a nappali világítás villogtatására:
      {
        digitalWrite(nappaliLed, HIGH); //A nappali világítás bekapcsolva
        beep(); //Hangjelzés funkció meghívása
        delay(250); //Várakozás 250 milliszekundum ideig
        digitalWrite(nappaliLed, LOW); //A nappali világítás kikapcsolva
        delay(250); //Várakozás 250 milliszekundum ideig
      }
      digitalWrite(medence1, LOW); //Medence1 kikapcsolva
      digitalWrite(medence2, LOW); //Medence2 kikapcsolva
      digitalWrite(kintiRele, HIGH); //Kinti világítás kikapcsolva
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      trigger = 0; //Flag változó értékének változtatása
      break; //Kilépés az elágazásból
    case 2: //Második eset ha a változó = 2
      Serial.print("Konyha mozgas\n"); //Megadott karakterlánc kiíratása a soros portra
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
      lcd.print("MOZGAS A KONYHABAN."); //Megadott karakterlánc kiíratása
      for (int i = 0; i <= 5; i++) //for ciklus a konyhai világítás villogtatására:
      {
        digitalWrite(konyhaLed, HIGH); //A konyhai világítás bekapcsolva
        beep(); //Hangjelzés funkció meghívása
        delay(250); //Várakozás 250 milliszekundum ideig
        digitalWrite(konyhaLed, LOW); //A konyhai világítás kikapcsolva
        delay(250); //Várakozás 250 milliszekundum ideig
      }
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      trigger = 0; //Flag változó értékének változtatása
      break; //Kilépés az elágazásból
    case 3: //Harmadik eset ha a változó = 3
      Serial.print("Folyoso mozgas\n"); //Megadott karakterlánc kiíratása a soros portra
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
      lcd.print("MOZGAS A FOLYOSON"); //Megadott karakterlánc kiíratása
      for (int i = 0; i <= 5; i++) //for ciklus a folyosó világítás villogtatására:
      {
        digitalWrite(folyosoLed1, HIGH); //A folyosó1 világítás bekapcsolva
        digitalWrite(folyosoLed2, HIGH); //A folyosó2 világítás bekapcsolva
        beep(); //Hangjelzés funkció meghívása
        delay(250); //Várakozás 250 milliszekundum ideig
        digitalWrite(folyosoLed1, LOW); //A folyosó1 világítás kikapcsolva
        digitalWrite(folyosoLed2, LOW); //A folyosó2 világítás kikapcsolva
        delay(250); //Várakozás 250 milliszekundum ideig
      }
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      trigger = 0; //Flag változó értékének változtatása
      break; //Kilépés az elágazásból
```

```
    case 4: //Negyedik eset ha a változó = 4
      Serial.print("Garazs mozgas\n"); //Megadott karakterlánc kiíratása a soros portra
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
      lcd.print("MOZGAS A GARAZSBAN."); //Megadott karakterlánc kiíratása
      for (int i = 0; i <= 5; i++) //for ciklus a garázs világítás villogtatására:
      {
        digitalWrite(garazsLed, HIGH); //A garázs világítás bekapcsolva
        beep(); //Hangjelzés funkció meghívása
        delay(250); //Várakozás 250 milliszekundum ideig
        digitalWrite(garazsLed, LOW); //A garázs világítás kikapcsolva
        delay(250); //Várakozás 250 milliszekundum ideig
      }
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      trigger = 0; //Flag változó értékének változtatása
      break; //Kilépés az elágazásból
    default: //Egyébként:
      Serial.print("Nincs mozgas\n"); //Megadott karakterlánc kiíratása a soros portra
      digitalWrite(kintiRele, HIGH); //A kinti világítás kikapcsolva
      //trigger = 0;
      break; //Kilépés az elágazásból
  }
}
```


Elérkeztünk a programunk leghosszabb funkciójához, amely a *check ()* azonosítót viseli. Nevéből eredően szintén egy valamilyen ellenőrzést végző funkcióról van szó. Ez a funkció végzi a mozgásérzékelést és riasztást mind fény és hangjelzéssel. Valamint a mozgás helyét is visszaadja és megjeleníti az LCD kijelzőnkön értesítés formájában. Ebben a funkcióban helyet kap egy több ágú elágazás, vagyis egy *switch* melynek négy *case* ága van és rendelkezik egy *default* ággal is. Ennek a működését nem kell bemutatnunk mivel már az előző projektek valamelyikében használtuk ezt a típust. A vizsgált változónk nem mást, mint a *flag* ként használt *trigger* nevű változó mely a megszakítás képes bemenetek által meghívott funkciókban van értékkel ellátva. Ehhez az értékhez mérten hoztuk létre a következő elágazást. Mely szerint, ha a változónk értéke egyenlő 1-gyel akkor az 1-es *case* tartalma kerül lefuttatásra, ha értéke 2-nek felel meg akkor a 2-es *case* fog lefutni és így tovább egészen addig még a változó értéké nem lesz nagyobb, mint 4. Amennyiben a változó értéke nagyobb 4-nél akkor az elágazás *default* ága kerül futtatásra. Nézzük az egyes *case*-ek tartalmát. Az első esetben, vagyis, ha *trigger = 1*-gyel akkor az 1-es akkor a *motion1 ()* funkciónk került lefutásra, ami egyenlő azzal, hogy az 1-es PIR szenzor érzékel mozgást, és ebből tudjuk, hogy az 1-es PIR szenzor a nappaliban helyezkedik el. Vagyis tudjuk, hogy hol volt észlelhető mozgás, ezért azt szeretnénk, hogy mindig az a szobavilágítás váljon aktívvá, ahol maga a mozgás történt, jelen esetben nem csupán aktiváljuk az aktuális szoba világítását, hanem villogtatjuk azt, hangjelzés és az LCD kijelzőre juttatott *MOZGÁS A NAPPALIBAN* üzenettel kisérve. Ami a villogtatást érinti egy for ciklus segítségével oldjuk meg azt, hogy az aktuális led dióda ötször ki és bekapcsolást végezzen. Jelen esetben a riasztást követően a kinti és a medence körüli világítást is aktiválni szeretnénk, tehát ennek a feltételnek eleget téve kell eljárnunk, ezt egyszerűen a megfelelő relé és a 32 és 33 as kimenetek aktív

állapotba helyezésével oldhatjuk meg. Fontos, hogy az elágazás ágaiban a *flag* változónk, vagyis a *trigger* értékére odafigyeljünk, mivel mind a négy mozgásérzékelő funkciónk ennek a változónak az értékét módosítja, ezért ezt minden egyes ágban nulláznunk kell azt, valamint ne felejtkezzünk meg a *brake* parancsról sem melynek a segítségével tudjuk elhagyni az aktuális *case*-t. A nappaliban történt mozgás érzékeléshez hasonlóan működik ez a többi három mozgásérzékelő által figyelt helyiségben is. Ezzel a projekt riasztást végző részét be is fejeztük.

```
void kapunyitas() //Funkció a kapu nyitására
{
  Serial.print("Kapunyitas\n"); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("KAPUNYITAS"); //Megadott karakterlánc kiíratása
  digitalWrite(garazskapuRele, LOW); //A garázskapu villogás bekapcsolva
  digitalWrite(kiskapuRele, LOW); //A kiskapu villogás bekapcsolva
  delay(3000); //Várakozás 3 másodpercig
  kapumotor.attach(12); //A kapumotor illesztése a 12-es pinre
  kapumotor.write(0); //A  kapumotor 0-dik pozícióba állítása
  for (int i = 0; i <= 100; i++) //For ciklus a szervomotor léptetésére:
  {
    kapumotor.write(i); //A kapumotor pozíciója megegyezik a ciklusváltozó(i) értékével
    delay(40); //A léptetés sűrűsége 40 milliszekundum
  }
  Serial.print("Kapu nyitva\n"); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("KAPU NYITVA"); //Megadott karakterlánc kiíratása
}
void kapuzaras() //Funkció a kapu zárására
{
  Serial.print("Kapuzaras\n"); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("KAPUZARAS"); //Megadott karakterlánc kiíratása
  delay(3000); //Várakozás 3 másodpercig
  for (int i = 95; i > 0; i--) //For ciklus a szervomotor léptetésére:
  {
    kapumotor.write(i); //A kapumotor pozíciója megegyezik a ciklusváltozó(i) értékével
    delay(40); //A léptetés sűrűsége 40 milliszekundum
  }
  delay(5000); //Várakozás 5 másodpercig
  kapumotor.write(0); //A  kapumotor 0-dik pozícióba állítása
  kapumotor.detach(); //A kapumotor leválasztása
  Serial.print("kapu zarva\n"); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("KAPU ZARVA"); //Megadott karakterlánc kiíratása
  digitalWrite(garazskapuRele, HIGH); //A garázskapu villogás kikapcsolása
  digitalWrite(kiskapuRele, HIGH); //A kikskapu villogás kikapcsolása
}
```


Tovább haladhatunk a mechanikai feladatok elvégzésére használt szervomotorok funkcióihoz. Elsőként a kapu nyitásáról kell gondoskodnunk. Az erre a feladatra készített funkciónk a *kapunyitas ()* azonosítóval lett ellátva. Ebben a funkcióban történik a házunk kapujának jelző fényeinek aktiválása és maga a kapu kinyitása. Elsőként egy üzenet formájában figyelmeztetjük a felhasználót a folyamat megkezdéséről, ezért a *KAPUNYITÁS* üzenet jelenik meg az LCD kijelzőn. Ezt követi a figyelmeztető villogás bekapcsolása és egy három másodperces várakozás. A várakozás után csatlakozásra kerül a kapunyitást végző szervomotor és megkezdődik

a pozíciójának változtatása egy for ciklus segítségével. Miután a motor elérte a kívánt pozíciót, vagyis a kapu nyitottnak tekinthető akkor a kijelzőn a *KAPU NYITVA* üzenet jelenik meg. Mivel a kaput kinyitottuk ezért be is kell tudnunk csukni azt. Erre a célra hozzuk létre a *kapuzaras ()* funkciót mely ugyanezt az elvet követi azzal a különbséggel, hogy az itt található for ciklus ezúttal nem a motor 0-ik pozíciójáról indul, hanem a motor jelenlegi pozícióját lépteti vissza a 0-ik pozícióba. A kapu csukását a villogó áramkörrel ellátott ledek bekapcsolásával kezdjük. A kapu állapotát is szintént jelezzük a felhasználó számára, mégpedig ezúttal a *KAPU ZÁRVA* üzenettel.

```
void garazsnyitas() //Funkció a garázs nyitására
{
  Serial.print("GARAZSNYITAS\n"); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("GARAZSNYITAS"); //Megadott karakterlánc kiíratása
  delay(3000); //Várakozás 3 másodpercig
  garazsmotor.attach(5); //A garázsmotor illesztése az 5-ös pinre
  garazsmotor.write(0); //A garázsmotor 0-ik pozícióba állítása
  for (p = 0; p < 100; p++) //For ciklus a szervomotor léptetésére:
  {
    garazsmotor.write(p); //A garázsmotor pozíciója megegyezik a ciklusváltozó(p) értékével
    delay(50); //A léptetés sűrűsége 50 milliszekundum
  }
  delay(5000); //Várakozás 5 másodpercig
  Serial.print("GARAZS NYITVA\n"); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0, 1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("GARAZS NYITVA"); //Megadott karakterlánc kiíratása
}
void garazszaras() //Funkció a garázs zárására
{
  Serial.print("GARAZSZARAS\n"); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0,1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("GARAZSZARAS"); //Megadott karakterlánc kiíratása
  delay(3000); //Várakozás 3 másodpercig
  for (p = 100; p >= 1; p--) //For ciklus a szervomotor léptetésére:
  {
    garazsmotor.write(p); //A garázsmotor pozíciója megegyezik a ciklusváltozó(p) értékével
    delay(50); //A léptetés sűrűsége 50 milliszekundum
  }
  garazsmotor.write(0); //A garázsmotor 0-ik pozícióba állítása
  delay(5000); //Várakozás 5 másodpercig
  garazsmotor.detach(); //A garázsmotor leválasztása
  Serial.print("GARAZS ZARVA\n"); //Megadott karakterlánc kiíratása a soros portra
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
  lcd.setCursor(0,1); //Kurzor pozicionálás ez esetben 0. karakter a 1. sorban
  lcd.print("GARAZS ZARVA"); //Megadott karakterlánc kiíratása
  delay(2000); //Várakozás 2 másodpercig
  digitalWrite(garazsRele, LOW); //Garázs szellőztetés bekapcsolása
  delay(5000); //Szellőztetés 5 másodpercig
  digitalWrite(garazsRele, HIGH); //Garázs szellőztetés kikapcsolása
}
```


Ezt a folyamatot meg kell ismételnünk a garázsajtó nyitásának automatizációjánál is. Szintén két funkcióval fogjuk az ajtó nyitását és a zárását is elvégezni. A nyitást végző funkciót *garazsnyitas ()* a zárást végzőt pedig a *garazszaras ()* azonosítókkal látjuk el. Természetesen ezen két funkció indulását vagy végbemenetelét is jeleznünk kell a felhasználó számára. Ezért a garázsajtó nyitásánál a *GARÁZSNYITÁS* üzenet jelenik meg, a bezárásánál pedig a *GARÁZSZÁRÁS* jelenik meg.

Ezzel az összes szükséges funkciót létrehoztuk ahhoz, hogy a projektünk a megfelelő módon működjön. Nincs más hátra, mint ezen funkciók megfelelő sorrendben történő meghívása és az MFRC522-es RFID olvasó által elvégzett ellenőrzés megvalósítása. Ezeket a lépéseket a programunk void loop ciklusában valósítjuk meg. Először a kártya azonosítójának az ellenőrzése történik egy feltétel vizsgálat mely segítségével azonosítjuk az RFID kártyánkat. Ezt az eljárást már egy korábbi projektben bemutattuk, ahol egy RFID azonosítós beléptető rendszert hoztunk létre.

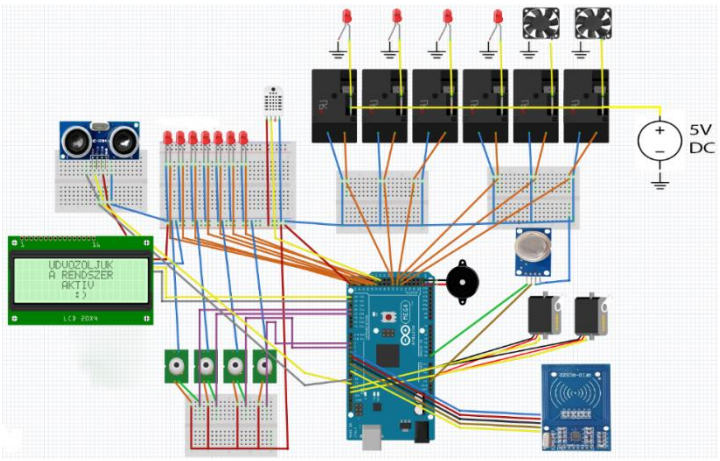
```
void loop() //ciklus
{
  int chk = DHT.read11(DHT11_PIN); //A DHT11 értékének a kiolvasása és segédváltozóban tárolása
  if ( !mfrc522.PICC_IsNewCardPresent()) //Feltétel vizsgálat, ha új kártya van jelen:
  {
    return; //kilépés a feltétel vizsgálatból
  }
  if (! mfrc522.PICC_ReadCardSerial()) //Feltétel vizsgálat kártya ID-jének kiolvasása
  {
    return; //kilépés a feltétel vizsgálatból
  }
  Serial.print("UID tag:"); //Megadott karakterlánc kiíratása a soros portra
  String content = ""; //String típusú üres karakterláncot tartalmazó változó
  byte letter; //Byte típusú változó
  for (byte i = 0; i < mfrc522.uid.size; i++) //for ciklus mely végighalad a kártya ID-ján:
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""); //Az ID ben lévő számok Hexadecimális stringre alakítása
    Serial.print(mfrc522.uid.uidByte[i], HEX); //Az átalakított ID kiíratása a soros portra
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ")); //konkatenálása az ID vel és 0x hozzarendelése
    content.concat(String(mfrc522.uid.uidByte[i], HEX)); //A konkatenáció kiíratása a soros portra
  }
  Serial.println(); //Üres karakterlánc kiíratása a soros portra
  content.toUpperCase(); //A változó tartalmának nagy karakterekre alakítása
  if (content.substring(1) == "F0 1E 78 7A") //Feltétel vizsgálat:
  {
    enter(); //Belépési funkció hívása
    kapunyitas(); //Kapunyitás funkció hívása
    garazsnyitas(); //Garázsnyitás funkció hívása
    kapuzaras(); //Kapuzárás funkció hívása
    garazszaras(); //Garázs zárás funkció hívása
    temperature(); //Hőmérséklet megjelenítése funkció hívása
    humidity(); //Páratartalom megjelenítése funkció hívása
    fume(); //Gázszivárgást megjelenítő funkció hívása
    alarmcheck(); //A riasztó állapotát ellenőrző funkció hívása
  }
  else //Egyébként
  {
    denied(); //Elutasítást igazoló funkció hívása
  }
  for ( ;;) //for ciklus a hőmérséklet, gázszivárgás és mozgás érzékelésére
  {
    checktemp(); //A hőmérséklet mérése funkció hívása
    leakage(); //Gázszivárgást ellenőrző funkció hívása
    check(); //Mozgás észlelését figyelő funkció hívása
  }
}
```


Tehát ha az RFID kártya azonosítója megegyezik az általunk belépési joggal ellátott azonosítóval akkor a *BELÉPÉS ENGEDÉLYEZVE* üzenetet megjelenítő *enter ()* funkció kerül meghívásra. Amennyiben viszont, ha az azonosító nem egyezik meg az általunk meghatározottal, akkor a *denied ()* funkció kerül meghívásra, mely a *BELÉPÉS ELUTASÍTVA* üzenetet jeleníti meg az LCD kijelzőn. Jelen esetben a kártyánkat használva belépési engedélyt kaptunk a rendszerünkbe, ezt követően azt szeretnénk, hogy a ház kapuja nyíljon ki, ezt a *kapunyitas ()* funkció hívásával

oldhatjuk meg. A kapunyitásától mérten némi késleltetéssel nyitjuk a garázsajtónkat is, tehát a *garazsnyitas ()* funkciót kell meghívnunk következőként. Miután a felhasználó személygépjárművel elfoglalta helyét a garázsban akkor, némi várakozási idő után bezárjuk a kaput és becsukjuk a garázsajtót. Mivel a modellünkön elhelyezésre került egy HC-SR04-es ultrahangos távolságmérő szenzor is, ezért a garázsajtó csukását egy feltételen keresztül is elvégezhetjük miszerint, ha a kocsi a garázsban van, vagyis bizonyos távolságra szenzortól helyezkedik el és ha ez a távolság egy általunk megadott értékkel egyenlővé válik akkor némi késleltetést alkalmazva csukhatjuk a garázsajtónkat. Miután a kapu is bezárásra került és a mechanikai feladatokat elvégző funkciók lefutottak. Akkor jöhet némi információ megjelenítése a kijelzőre. Mégpedig elsőként a *temperature ()* funkció hívásával kezdjük mely a hőmérséklet mindig aktuális értékét jeleníti meg a kijelzőn, ezután a *humidity ()* páratartalmat jegyző funkció, majd a gázszivárgást ellenőrző *fume ()* funkció és végül a riasztórendszer állapotát figyelő funkció mely az *alarmcheck ()* azonosító alatt található meg. Végül létrehozunk egy rekurziót is egy for ciklus segítségével, ahol, a DHT11-es, MQ2-es és a PIR szenzoroktól kapott jeleket állandóan olvassuk.

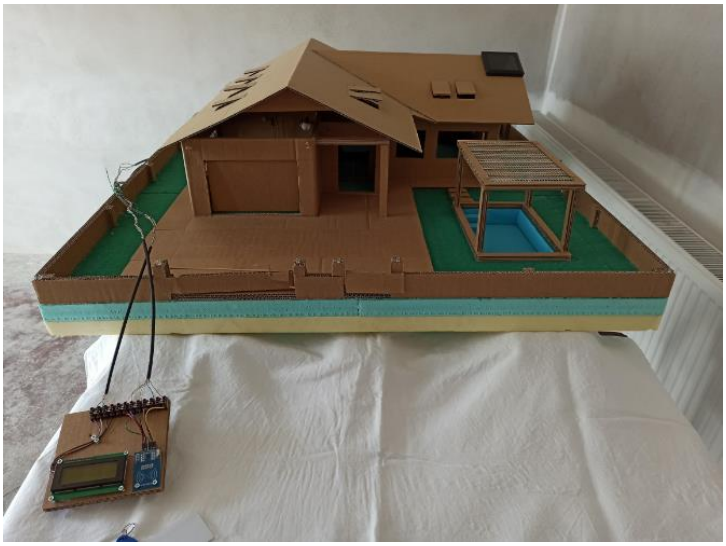
Belépést nyertünk tehát rendszerünkbe, tehát ha a riasztórendszerünk aktív állapotban van akkor annak tesztelését úgy tudjuk megoldani, hogy valamely PIR szenzor előtt mozgást generálunk, legyen most ez a szenzor az 1-es sorszámú a nappaliban elhelyezett szenzor, mivel a programunkban úgy határoztunk, hogy ha mozgást érzékel valamelyik szenzorunk akkor a helységben melyben a mozgási aktivitás történt ott villogjon a megvilágítás melyet hangjelzés is követ s végül a kijelzőn egy üzenetet eredményez. Tehát ha a nappaliban elhelyezett szenzor előtt mozgást generálunk az ennek megfelelő folyamatok zajlanak le, függetlenül attól, hogy éppen mi van jelen az LCD kijelzőnk, vagyis, hogy a programunk éppen melyik szakászában tart. Ezt a funkciót teszik lehetővé számunkra az Arduino Mega 2560 *interrupt* pinjei.

Amennyiben a futtatást követően az eddig említett funkciók, működnek akkor a projektünket késznek tekinthetjük.



31

A fentebbi 31-es ábrán a projektünk bekötési ábráját láthatjuk. Itt figyelhetjük meg mely szenzorok mely pinekre csatlakoznak, illetve a kimeneti egységek hol, és hogyan helyezkednek el. Fontos, hogy a szenzorok és a kimeneti egységek Gnd vagyis közös földeléséről és a megfelelő feszültség és amper értéket biztosító tápegységről gondoskodjunk. Valamint ügyeljünk a vezetékek megfelelő csatlakoztatására és kerüljük el a zárlatokat is. Amennyiben a bekötést helyesen véghez vittük a projektünk véglegesnek tekinthető (32,ábra).



32. ábra Az elkészült modell