9. Kapu nyitása és zárása pin kóddal 4x4 karakteres Keypad segítségével

A következő projekt már a bonyolultabbak közé tartozik. Itt egy 4x4-es Keypad billentyűzet segítségével fogunk jelet kiadni az Arduino Uno valamely kimenetére. Valamint erre a kimenetre csatlakoztatjuk egy relé tekercsét mely egy elektromos kapuzárat fog kapcsolni. Természetesen a felhasználó számára szükséges információkat, mint például a pin kód és annak helyessége üzenetek formájában megjelenítésre kerül egy I2C buszrendszert használó 2x16 karakteres LCD kijelzőre. Végül az áramkörben elhelyezünk még egy led diódát is a kapu nyitott állapotának szimulálására.

Szükséges eszközök: Számítógép, Arduino IDE, Arduino Uno, egy db 2x16 karakteres LCD kijelző I2C adapterrel, egy db 4x4 karakteres Keypad billentyűzet, egy db relé, egy db led dióda, egy db elektromos zár, USB kábel és vezetékek.

A billentyűzet használatához már szükségünk lesz egy új header fájlra ez nem más, mint a *Keypad.h*. Ezen kívül beillesztjük még az I2C buszrendszer kezelésére használt *Wire.h* és az lcd kijelző vezérlésére használt *LiquidCrystat_I2C.h* fájlokat.

```
#include <Keypad.h> //A 4x4-es billentyűzethez használt könyvtár beillesztése
#include<Wire.h> //Wire könyvtár beillesztése az I2C busz használatához
#include <LiquidCrystal_I2C.h> //Az I2C Folyékony kristályos LCD kijelző kezelő könyvtára
LiquidCrystal_I2C lcd(0x27, 16, 2); //Az általunk használt kijelző karakterkészlete 16 karakter és 2 sor
#define Password_Lenght 7 //A jelszó hosszának defiálása 6 karakter + NULL karakter
char Data[Password_Lenght]; // 6 számot tartalmazhat + NULL karakter így 7 karakter hosszú
char Master[Password_Lenght] = "123456"; //A Master jelszó
byte data_count = 0, master_count = 0;
bool Pass_is_good; //Bináris típusu változó mely a jelszó helyességét igazolja
char customKey; //A felhasználó által lenyomott billentyűnek megfelelő ASCII kodot tartalmazó változó
const byte ROWS = 4; //Konstans byte típusú változó mely a billentyűzet sorait számozza
const byte COLS = 4; //Konstans byte típusú változó mely a billentyűzet oszlopait számozza
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'}, //Két dimenziós a karaktereket tartalmazó tömb
  {'4', '5', '6', 'B'}, //Két dimenziós a karaktereket tartalmazó tömb
  {'7', '8', '9', 'C'}, //Két dimenziós a karaktereket tartalmazó tömb
  {'*', '0', '#', 'D'}  //Két dimenziós a karaktereket tartalmazó tömb
};
bool kapu = true; //A kapu nyitását igazoló bináris típusú változó
int zar = 12; //Az elektromos zár pinjét tartalmazó változó
byte rowPins[ROWS] = {2, 3, 4, 5}; //A pinek melyek a sorokra csatlakoznak
byte colPins[COLS] = {6, 7, 8, 9}; //A pinek melyek az oszlopokra csatlakoznak
Keypad customKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS); //initialize an instance of class NewKeypad
void setup()
{
  pinMode(zar, OUTPUT); //A zárra csatlakoztatott pin kimenetté alakítása
  lcd.init(); //Az LCD kijelző inicializálása
  lcd.backlight(); //Az LCD kijelző háttérvilágításának bekapcsolása
  lcd.setCursor(2, 0); //Kurzor pozicionálás ez esetben 0. karakter a 0. sorban
  lcd.print("UDVOZOLJUK!"); //Megadott karakterlánc kiíratása
  delay(3000); //Várakozás 3 másodpercig
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
}
```

1. ábra kapu nyitása és zárása pin kóddal programkód
(forrás: saját szerkesztés)

A könyvtárak implementálását, valamint az lcd kijelző karakterkészletének a megadását követően definiálunk egy változót *Password_Length* mely az általunk használt jelszó hosszára mutat, ez a változó 6 darab számot tartalmaz, valamint a NULL karakter, vagyis hossza 7 karakter. Ez után létrehozunk kettő darab karaktereket tartalmazó tömb adattípus, a tömb hosszát megadó paramétert az imént deklarált *Password_Length* megadásával is megtehetjük. A kettő darab karaktereket tartalmazó tömbünk a *Data* és *Master* nevet viselik, mindkét tömb hossza azonos. Ami az értékadást illeti a *Data* nevezetű tömbünket üresen hagyjuk, viszont a *Master* nevezetűben elhelyezzük az általunk helyesnek ítélt jelszót, legyen ez az ”123456” számsorozat. A következő sorban létrehozunk szintén kettő darab immár byte típusú változót *data_count* és *master_count* ez két változó az általunk meghatározott jelszó vagyis a *Master* és a felhasználó által beütött jelszó hosszának a vizsgálatára és összehasonlítására fog szolgálni. Tovább haladva a szekvencián létrehozunk egy *Boolean* vagyis bináris típusú változót mely a jelszó helyességét fogja igazolni, valamint egy karakter típusú változót mely pedig a felhasználó által lenyomott billentyűre mutat. Szükséges még beállítanunk a billentyűzetünk karaktereinek az elhelyezkedését, ezt egy karaktereket tartalmazó kétdimenziós tömb létrehozásával oldjuk meg. A kapu nyitva, illetve zárt állapotát is fontos megjegyeznünk ezért itt szintén egy bináris típusú változót deklarálunk *kapu* néven melynek értékét igazra állítjuk. Ezt követi a relé tekercsére kapcsolt kimenet mely ebben az esetben a 12-es pin lesz. Végül a billentyűzet sorai és oszlopai által használt pinek meghatározása következik, ahol a sorok a 2-es pintől az 5-ös pinig helyezkednek el az oszlopok pedig a 6-os pintől egészen a 9-es pinig ez után a billentyűzethez rendeljük az imént meghatározott paramétereket. El is érkeztünk a programkód void setup () részéhez, itt a relére csatlakozó pin kimenetté alakításával kezdünk, majd a szokásos lcd inicializáló folyamat ám most egy üdvözlő üzenetet is megjelenítünk az általunk meghatározott ideig jelen esetben ez három másodperc. Ezután töröljük a kijelző tartalmát.

```
void loop() //ciklus
{
  if (kapu == 0) //Feltétel vizsgálat:
  {
    customKey = customKeypad.getKey(); //Karakterek beolvasása
    if (customKey == '#') //Feltétel vizsgálat:
    {
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      digitalWrite(zar, LOW); //A zár kikapcsolt állapotának megőrzése
      lcd.print("KAPU ZARVA"); //Megadott karakterlánc kiíratása
      delay(3000); //Várakozás 3 másodpercig
      lcd.clear(); //Az LCD kijelző tartalmának a törlése
      kapu = 1;    //A kapu változó értékének IGAZRA vagy 1-re változtatása
    }
  }
  else nyitas(); //Egyébként, nyitas funkció meghívása
}
void clearData() //Adattörlés funkció az adatokat tartalmazó tömb törlésére
{
  while (data_count != 0) //Előtesztelő ciklus amíg: data_count nem egyenlő 0-val addig:
  {
    Data[data_count--] = 0; //A tömb elemeinek törlése
  }
  return; //Kilépés a ciklus magjából
}
void nyitas() //Nyitas funkció
{
  lcd.setCursor(0, 0); //Kurzor pozicionálás ez esetben 0. karakter a 0. sorban
  lcd.print("A JELSZO:"); //Megadott karakterlánc kiíratása
  customKey = customKeypad.getKey(); //Karakterek beolvasása
  if (customKey) //A billentyű lenyomásának vizsgálata megfelel ennek (customKey != NO_KEY)
  {
    Data[data_count] = customKey; //A karakterek tömbben tárolása
    lcd.setCursor(data_count, 1); //Kurzor pozicionálás a tömbre az 1. sorban
    lcd.print(Data[data_count]); //Karakter kiíratása
    data_count++; //A jelszót tartalmazó tömb hosszának inkrementálása 1 elemmel
  }
```

2. ábra kapu nyitása és zárása pin kóddal programkód
(forrás: saját szerkesztés)

A projekt lényege a ciklus részben, vagyis a void loop() ban valósul meg. Itt rögtön egy feltétel vizsgálattal kezdünk, ahol a kapu nyitott vagy zárt állapotát vizsgáljuk és amint a kapu zárt állapotba kerül a feltétel igaz ága kerül lefuttatásra. Ahol egy lcd tartalom törlést követően egy kurzor pozicionálás és egy megadott karakterlánc ”KAPU ZARVA”megjelenítése megy végbe. Majd a kapu zárt állapotát igazoló bináris változó értékének igazra állítása is megtörténik. Az *else* ágban egy általunk a későbbiekben létrehozott funkciót fogunk meghívni. Ezt követően egy funkciót hozunk létre *clearData* néven mely a *Data* nevezetű tömb tartalmának a törlését fogja elvégezni. Ez a funkció tartalmaz egy *while* előtesztelő ciklust mely végigfut a tömb elemein és nullákkal tölti fel azt. Amint ez a folyamat végbe ment kilép a ciklusból a *return* funkció segítségével. Egy újabb funkció létrehozása következik mely a *nyitas* névre halgat. Itt egy lcd-re íratással kezdünk, méghozzá a 0-ik sorban és a 0-ik karakteren. Az lcd kijelzőn megjelenik az ”A

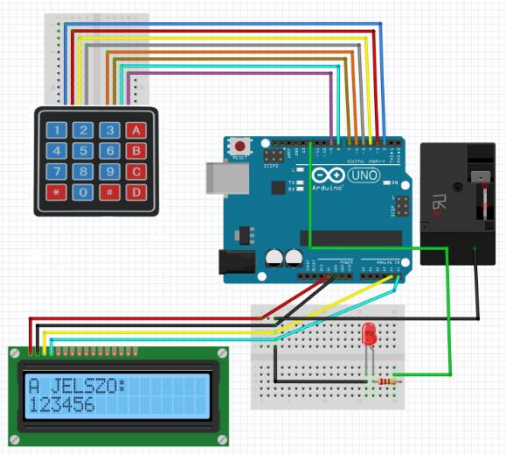
JELSZO:” üzenet. Majd megtörténik a felhasználó által lenyomott billentyű karaktereinek a tárolása és kiíratása is egyaránt. Ezt elsőként egy értékadási művelettel kezdjük, ahol a *customKey* változónk tartalmát a billentyűzetből kapott karakterrel töltjük fel, s ezt egy feltétel vizsgálat követi mely szerint, ha észlelhető lenyomott billentyű akkor annak a karakterét tárolja a *Data* tömbünkben. Szeretnénk visszajelzést is kapni a megnyomott karakterről ezért azt a kijelzőn is megjelenítjük, ez esetben a kurzor pozícióját nem a szokásos módon pozícionáljuk, a karakterre mutatás helyett a tömbünkre mutatunk és megadjuk a kiíratás sorát mely ez esetben az 1-es sor, végül a *data_count* hosszát inkrementáljuk a *data_count++* parancs segítségével.

```
if (data_count == Password Lenght - 1) //Feltétel vizsgálat miszerint,
{                                      ha a megadott jelszó hossza megfelelő akkor hasonlitás a Master-hez
  if (!strcmp(Data, Master)) //String Comparison
  {
    lcd.clear(); //Az LCD kijelző tartalmának a törlése
    delay(500); //Várakozás 500 milliszekundum ideig
    digitalWrite(zar, HIGH); //A zár nyitása
    lcd.print("KAPU NYITVA"); //Megadott karakterlánc kiíratása
    kapu = 0; //A kapu változó értékének HAMISA vagy 0-ra változtatása
  }
  else
  {
    lcd.clear(); //Az LCD kijelző tartalmának a törlése
    delay(500); //Várakozás 500 milliszekundum ideig
    digitalWrite(13, LOW); //A zár zárása
    lcd.print("HELYTELEN JELSZO"); //Megadott karakterlánc kiíratása
    delay(1000); //Várakozás 1 másodpercig
    kapu = 1; //A kapu változó értékének IGAZRA vagy 1-re változtatása
  }
  clearData(); //Adattörlés funkció hívása
}
}
```

3. ábra kapu nyitása és zárása pin kóddal programkód
(forrás: saját szerkesztés)

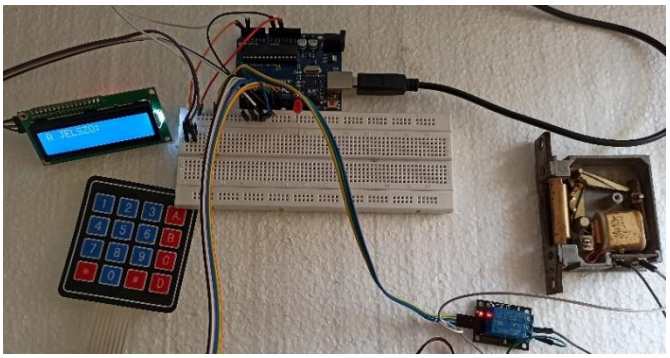
Miután a felhasználó által lenyomott billentyűk karaktereinek a tárolása megtörtént, meg kell vizsgálnunk, hogy annak hossza megegyezik-e a *Master* jelszó hosszával mínusz egy karakter. Amennyiben igen akkor egy újabb feltétel vizsgálatot végzünk, de most egy ún. *String comparison* operátort fogunk lefuttatni a *Data* és a *Master* tömbünk összehasonlítására. Ez az operátor nem mást csinál, mint végig pásztázza mindkét tömb elemeit és összehasonlítja azokat egymással. Ha a két tömb tartalma megegyezik akkor a feltétel igaznak mondható így a belépési szándék engedélyezhető, tehát nyithatjuk a zárat és visszajelezhetünk a felhasználónak az lcd kijelzőre kiíratott üzenettel. Ne felejtsük el a kapu zárt állapotát igazoló bináris változó értékét hamisra állítani. Amennyiben a két tömb tartalma nem azonos akkor a feltétel *else* ága kerül futtatásra, ahol a felhasználót a „”HELYTELEN JELSZO” üzenet figyelmezteti a nem megfelelő jelszó

megadására. A 13-as kimenet alacsony szinten marad tehát zárva van. A kapu zárt állapotát az erre a célra használt változónkkal is igazoljuk. Kilépve a feltétel vizsgálatból töröljük a *Data* tömb tartalmát a *clearData* funkció meghívásával.



4. ábra kapu nyitása és zárása pin kóddal bekötési rajz
(forrás: saját szerkesztés)

Projektünk megépítését a fentebb látható 4.ábra alapján visszük véghez. Szokásos módon az I2C busz csatlakozási pontjai nem változtak. A 4x4 karakteres billentyűzetünket a programkódban meghatározott pinekhez csatlakoztatjuk. A sorok az Arduino 2-es pinjétől az 5-ös pinjeig helyezkednek el, az oszlopok pedig a 6-os pintől a 9-es pin-ig. A kapu nyitását végző relé tekercsének egyik csatlakozója az Arduino Uno 12-es kimenetéhez a led dióda pedig a relé tekercsének mindkét csatlakozójára csatlakozik, vagyis, ha a relé bekapcsolt állapotba kerül a led dióda ezt világítással jelzi. A tekercs fennmaradt csatlakozója az áramkör közös földjéhez a GND-hez csatlakozik. A sikeres megépítés esetében az a kijelző tartalma a következő.



5. ábra kapu nyitása és zárása pin kóddal működés közben
(forrás: saját szerkesztés)