

Számítógép Architektúrák

Az ajtónyitó megvalósítása Arduino-val

Mészáros András

BME Hálózati Rendszerek és Szolgáltatások Tsz.
meszarosa@hit.bme.hu

2025. február 26.
Budapest

Tartalom

- A gyakorlat célja:
  - Megmutatni, hogy a manapság elterjedt és olcsó mikrokontrollerekkel milyen könnyű dolgozni
- Tartalomjegyzék:
  - Az Arduino bemutatása
    - Hardver
    - Programozás
    - Szenzorok
  - Az ajtónyitó megvalósítása
    - A kijelző
    - Az RFID olvasó
    - A billentyűzet
    - A teljes hardver
    - A teljes kód

Arduino

- 2005, Ivrea, Olaszország
- Célja: egyszerű prototípusfejlesztés
- Valójában egy teljes család (http://arduino.cc)
- Ami közös bennük:
  - Atmel AVR 8 bites processzor (**Harvard** architektúra!)
  - Beépített flash memória a programnak
  - Beépített EEPROM tartós adattárolásra
  - Beépített RAM
  - Digitális és analóg ki- és bemenetek
  - USB port-on át programozható (nem mindegyik)
  - C++-szerű nyelven programozható

Arduino

- Open source hardver → milliónyi variáns milliónyi hardverrel

- **Ami közös bennük: az interfész**
  - Az eredeti Arduino-val kompatibilis tüskesor
  - Az eredeti Arduino-val kompatibilis programozói felület
  - Kvázi szabvány – és nem konkrét gyártó/típus

  → Az Arduino (ma már) nem „játékszer”, komoly szereplő

Az Arduino család

- Eltérések:
  - Bemenetek/kimenetek száma
  - Memória mérete (flash/EEPROM/RAM)
  - Néhány jellegzetesség:
    - A Mega ADK-t hozzá lehet kötni az Android eszközökhöz
    - A LilyPad ruhára varrható

	CPU órajel	Flash	RAM	EEPROM	Digitális I/O	Analóg I/O
Uno	16 MHz	32 kB	2 kB	1 kB	14	6
Mega ADK	16 MHz	256 kB	8 kB	4 kB	54	16
Micro	16 MHz	32 kB	2,5 kB	1 kB	20	12
LilyPad	8 MHz	32 kB	2 kB	1 kB	9	4

Arduino vs Raspberry Pi

Arduino Mega

CPU: 16 MHz (8 bit)
RAM: 8 kB
Flash: 256 kB
I/O: 54 digitális, 16 analóg

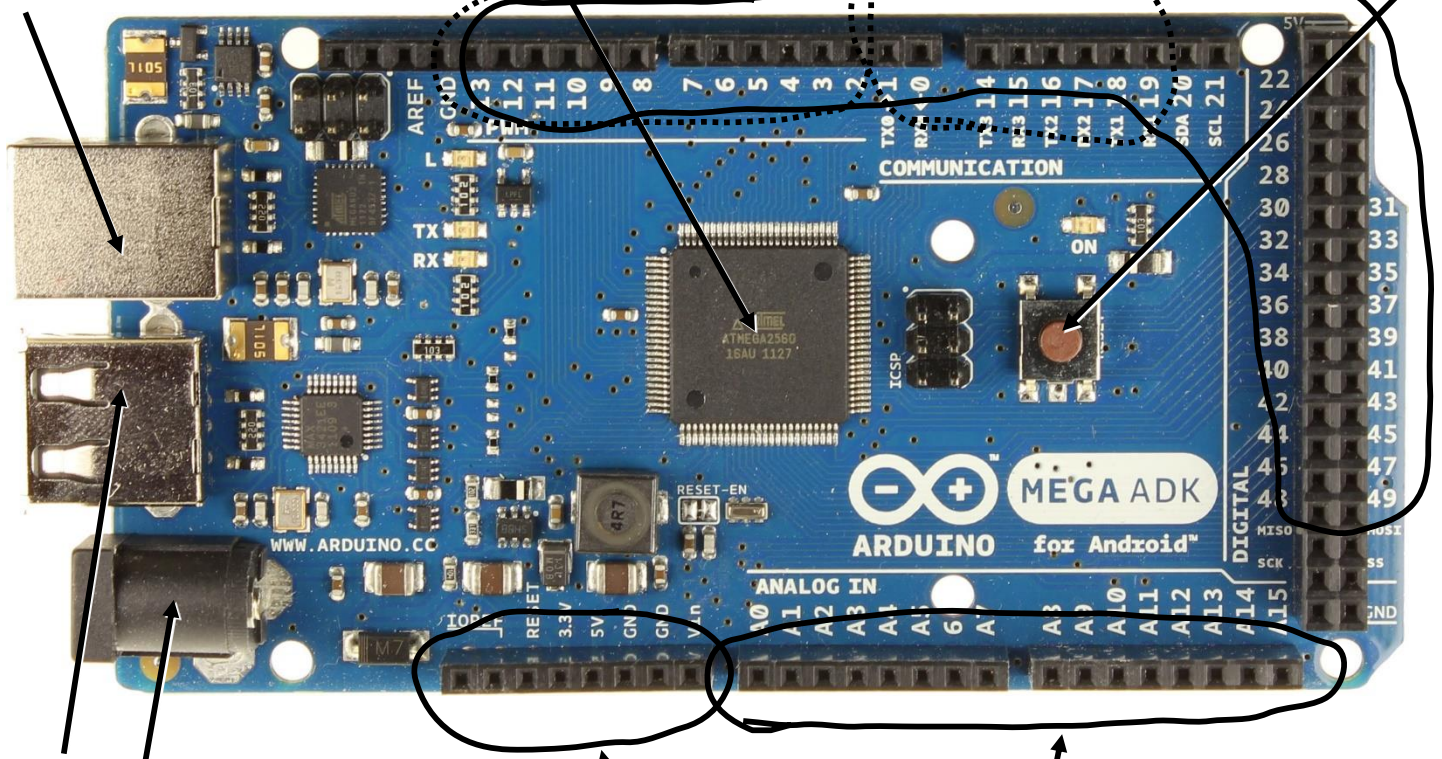
Ár: 42€

Raspberry Pi 4

CPU: 1.5 GHz, 4 mag (64 bit)
RAM: 1 GB DDR4
Flash: MicroSD (64 GB)
I/O: 40 GPIO
+ Ethernet, HDMI, USB
Ár: 43€

- **Más a cél:**
  - Arduino:
    - Hangsúly: I/O, minden mennyiségben, pofonegyszerűen
    - Bekapcsolom, és 1 másodperc múlva már megy is
    - Kis fogyasztás
    - Nagyon sok az Arduino kompatibilis eszköz
  - Raspberry Pi:
    - Hangsúly: általános célú (programozás oktatás)
    - Op. rendszer (linux) kell rá! Sokáig tart, mire elindul.
    - Nincs analóg I/O, soros perifériák illesztése problémás

Arduino Mega ADK



USB programozáshoz és tápellátáshoz

Mikrokontroller

PWM kimenetek

Soros portok

Reset kapcsoló

Digitális be- és kimenetek

USB Android kapcsolathoz

Alternatív tápellátás

Tápellátás a perifériáknak

Analóg bemenetek

Programozás

- Open-source cross-platform fejlesztői környezet (Arduino, VS Code)
- Programnyelv:
  - Arduino program: „sketch” ≈ C++ (.ino)
  - Elemi adattípusok: int, long, bool, **float**, char
- Két kötelező függvény:
  - **void setup () { … }** - az ide írt kód induláskor fut le, egyetlenegyszer
  - **void loop () { … }** - inicializálás után ez a függvény fut le újra és újra (ha befejeződik, újraindul a függvény elejétől)
- Fordítás:
  - A PC-s fejlesztőkörnyezet AVR kódot fordít → cross compiler
  - USB-n keresztül feltölti a mikrokontroller flash memóriájába
- Hibakeresés:
  - Amit az Arduino a default soros portjára ír, azt USB-n keresztül megmutatja a fejlesztőkörnyezet

Ki/Bemenetek

Digitális ki- és bemenetek

- A digitális lábak ki és bemenetként is szolgálhatnak
  - De egyszerre csak az egyik szerepben!
  - Beállítás:
    - **pinMode (4, INPUT);** – 4-es láb → bemenet üzemmód
    - **pinMode (5, OUTPUT);** – 5-ös láb → kimenet üzemmód
- A digitális lábak magasba, ill. alacsonyba húzása:
  - **digitalWrite (5, HIGH);** – az 5-ös lábra logikai 1-et tesz (5V)
  - **digitalWrite (5, LOW);** – az 5-ös lábra logikai 0-át tesz (0V)
- A digitális lábak olvasása:
  - **int val;**
  - **val=digitalRead (4);** – a 4-es láb logikai értékének olvasása
  - **if (val==HIGH) …**, vagy **if (val==LOW) …**

Analóg bemenetek

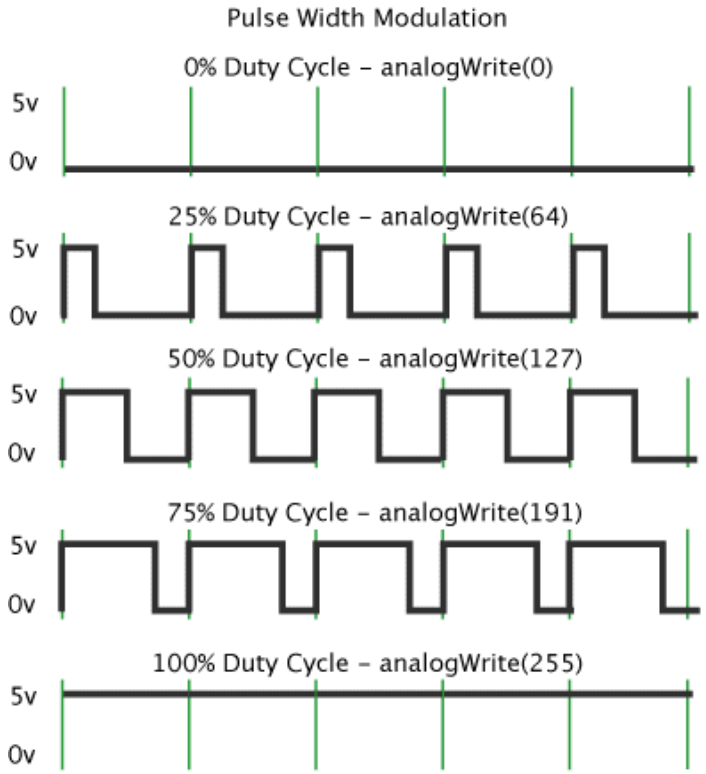
- Nemcsak HIGH és LOW értékeket lehet róla olvasni
- ADC (Analog-Digital Converter) 10 bites felbontással
  → 1024 különböző feszültségszintet tud megkülönböztetni
- 0V és 5V között 1024 szint → 4.88 mV különbséget tud tenni
- Használata:
  - **int val;**
  - **val = analogRead (3);** - a 3-as analóg bemenetről mintát vesz
  - **val** értéke: 0-tól (0V esetén) 1023-ig (5V)
- Az 5V maximum referencia változtatható az **analogReference()** függvénnyel, de 5V a maximum

PWM kimenetek

- Folyamatos analóg kimenet (pl. 4.2V) nem tud előállítani
- Viszont a kimenet gyors ki-be kapcsolgatásával „átlagos értelemben” tetszőleges feszültséget elő tud állítani

  → **PWM: Impulzusszélesség-moduláció**

- Erre szolgál az **analogWrite ()** függvény
- Paraméterek: láb száma, kitöltöttség (Duty Cycle, 0…255 között)



Soros ki- és bemenetek

- A kommunikációhoz max. 2 ér kell:
  - RX: a vételhez (ha szükség van vételre)
  - TX: az adáshoz (ha szükség van adásra)
- Az Arduino ADK 4 soros portot tartalmaz
  - Az elsőt használja Debug célra a fejlesztőkörnyezet USB-n keresztül
- Használata: **class Serial**-on keresztül

Soros ki- és bemenetek

- *Inicializálás*: **Serial1.begin (9600);** - port megnyitása 9600 bps sebességre
- *Írás*: **Serial1.write (…);** - elküld 1 bájtot, egy NULL terminált string-et, vagy egy tömböt
- *Írás*: **Serial1.print (...);** - a paraméterét stringgé konvertálja (ha nem az), és elküldi. A **Serial1.println (...);** még egy soremelést is utánatesz.
- *Olvasás*: **int data = Serial1.read();** - egy megérkezett bájt kiolvasása (-1, ha nincs
- *Érkezett-e adat*: **int count = Serial1.available();** - megadja, hány bájt érkezett
- *Lezárás*: **Serial1.end ();** - a port lezárása után a megfelelő láb digitális ki- és bemenetként használható tovább

Soros ki- és bemenetek

- Működése:
  - Vételkor Interrupt keletkezik
  - Az Interruptot lekezeli az Arduino:
    - Beleteszi az érkezett bájtot egy tárolóba (buffer)
  - A **read()** a tárolóból olvas
  - Az **available()** a tárolóban lévő bájtok számát adja vissza

Memóriák

Memóriák

- Az Arduino az alábbi memóriákkal rendelkezik:
  - **Flash memória**: ez tárolja a programot
    - Tápfeszültség nélkül is megmarad a tartalma
  - **RAM**: ebben vannak a változók és a stack is
    - Tápfeszültség kell a tartalom megőrzéséhez
  - **EEPROM**: tartósnak szánt adatokhoz
    - Tápfeszültség nélkül is megmarad a tartalma
- Az AVR processzorok Harvard architektúrát követnek
  - 2 címtér:
    - Az utasításokat és konstansokat a flash memóriából veszi
    - A változókhoz/stack-hez a RAM-ot használja
  - És az EEPROM?
    - Ahhoz külön függvénykönyvtár van

EEPROM

- A rendszer része az EEPROM könyvtár
- #include <EEPROM.h>
- Írás:
  - **EEPROM.write (cím, adat);**
    - A cím int típusú
    - Az adat byte típusú
- Olvasás:
  - **byte a;**
    a = EEPROM.read (42);
    - Kiolvassa a 42-edik byte-ot az EEPROM-ból

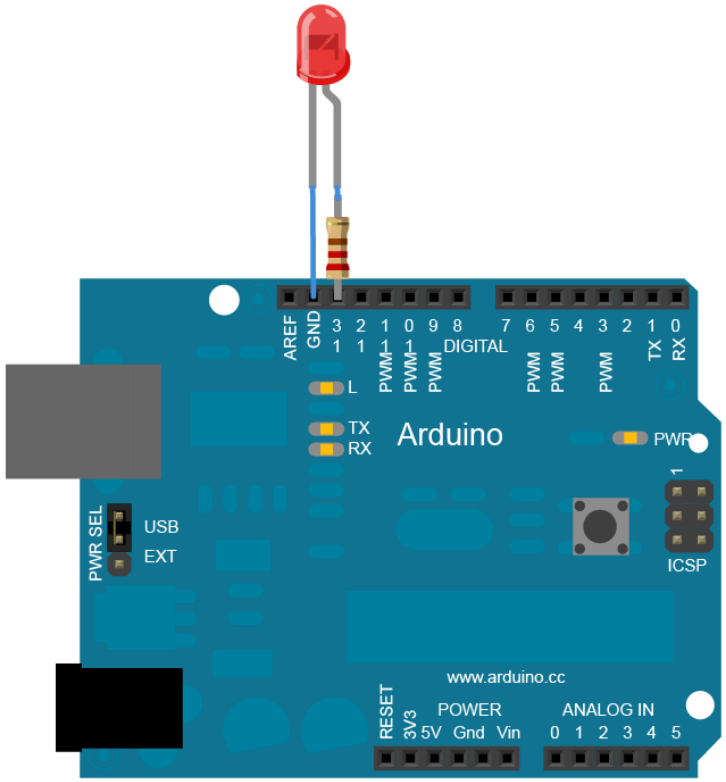
Perifériák illesztése

Digitális kimeneti perifériák

- **Példa**:
  LED villogtatás (Nem Arduino ADK, de ugyanaz az elv)

Alkatrész: LED, 220 Ohmos ellenállás

LED nélkül is megy!
A 13-as lábon van beépített LED.

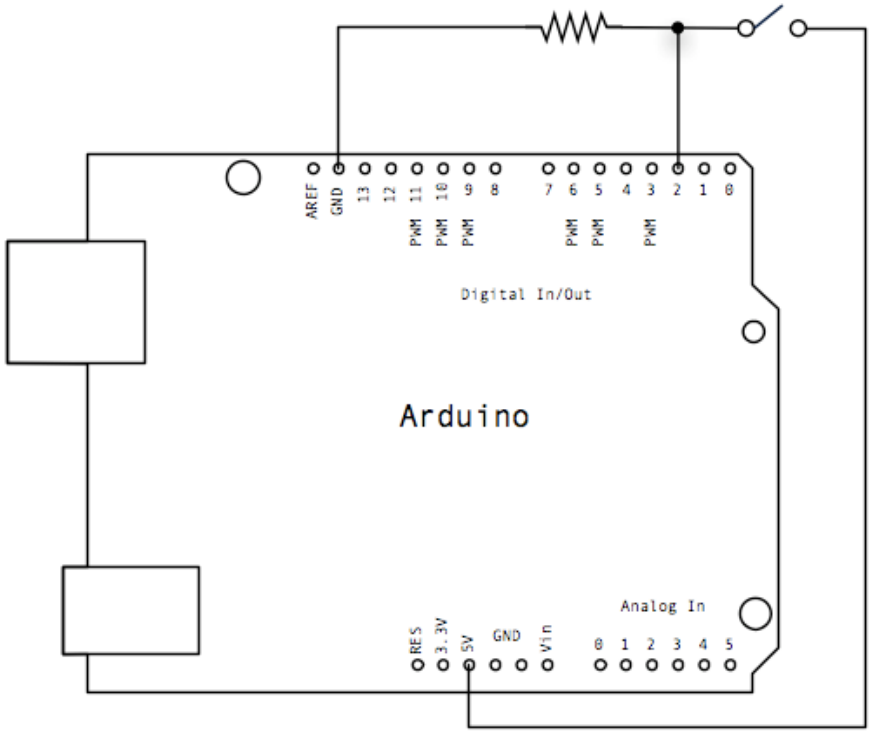


- **Kód**:

```
const int ledPin = 13;
void setup () {
    pinMode (ledPin, OUTPUT);
}
void loop () {
    digitalWrite (ledPin, HIGH);
    delay (1000);
    digitalWrite (ledPin, LOW);
    delay (1000);
}
```

Digitális bemeneti perifériák

- **Példa**:
  Nyomógomb állapotának lekérdezése, a beépített LED kigyújtása

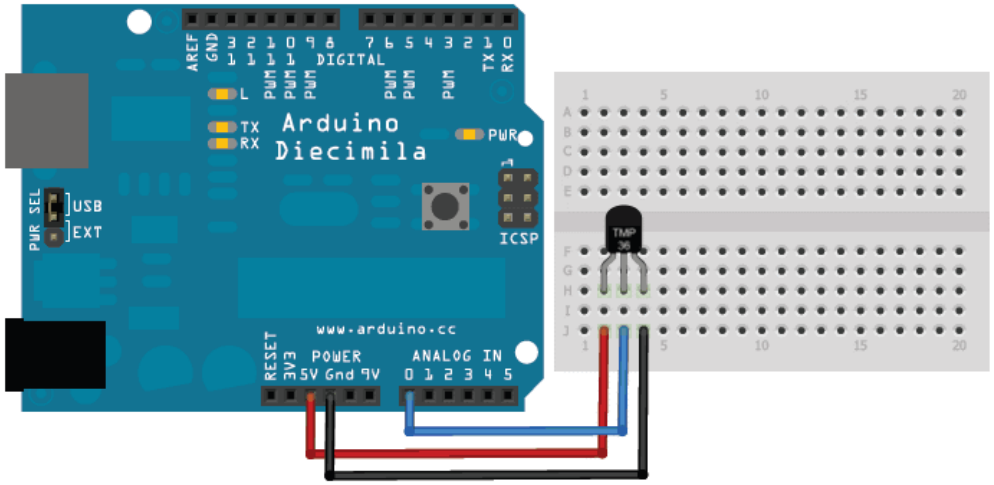


- **Kód**:

```
const int buttonPin = 2;
const int ledPin =  13;
int buttonState = 0;
void setup () {
    pinMode (ledPin, OUTPUT);
    pinMode (buttonPin, INPUT);
}
void loop () {
    buttonState = digitalRead (buttonPin);
    digitalWrite (ledPin, buttonState);
}
```

Analóg bemeneti perifériák

- **Példa**:
  TMP36 Hőmérséklet szenzor illesztése



- **Kód**:

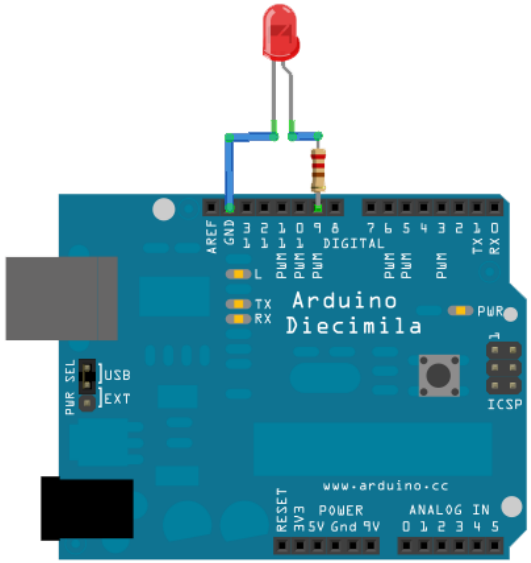
```
const int sensorPin = 0;
void setup () {
    Serial.begin (9600);
}
void loop () {
    int reading = analogRead (sensorPin);
float voltage = reading * 5.0 / 1024.0;
    float temperature = (voltage - 0.5) * 100;
    Serial.println (temperature);
    delay(1000);
}
```

Analóg bemeneti perifériák

- A legérdekesebb perifériák!
  - Filléres szenzorok tömkelege:
    - 3 tengelyes gyorsulásmérő (3 analóg bemenetet használ)
    - Alkohol szenzor
    - Szénmonoxid szenzor
    - Szálló por koncentráció szenzor
    - Elhajlásmérő szenzor
    - Erő (súly) mérő szenzor
    - Vibráció szenzor
    - Giroszkóp (2 tengelyes → 2 analóg bemenet kell)
    - Távolságmérő (infra fénnyel vagy ultrahanggal)
    - Hőmérő
    - Páratartalommérő
    - Stb.
- A mért fizikai mennyiséget analóg jellé alakítják

PWM kimeneti perifériák

- **Példa**:
  LED halványítás/fényesítés



- **Kód**:

```
const int ledPin = 9;
void setup ()  {
}
void loop ()  {
    for (int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
        analogWrite (ledPin, fadeValue);
        delay (30);
    }
    for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
        analogWrite (ledPin, fadeValue);
        delay (30);
    }
}
```

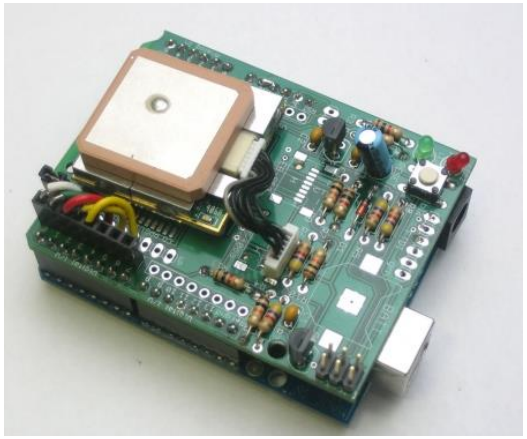
Soros perifériák

- Példák soros portra csatlakozó perifériákra:
  - RFID modul: soros porton küldi a leolvasott kártya kódját
  - GPS modul: soros porton küldi rendszeresen a koordinátákat
  - GSM/GPRS modul: soros porton kell vezérelni, illetve az átviendő adatok is a soros porton mennek
  - Stb.

Pajzsok (Shield)

- Az Arduino tetejére illeszthető komplett perifériák
  - Nagyon sok van:
    - GPS, LCD vezérlő, SD kártya kezelő, WIFI, Bluetooth, ZigBee, GSM, ...

GPS Shield
€18



MP3 Shield
€25

2.8” TFT és érintőképernyő
€40

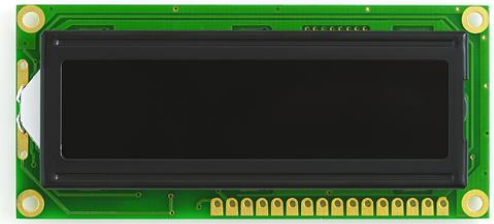
Az ajtónyitó megvalósítása

Ajtónyitó

- Milyen alkatrészek is kellenek?
  - Egy kis kijelző, amire üzeneteket írhatunk a felhasználónak
  - Egy numerikus billentyűzet
  - Egy kártyaleolvasó – RFID jó lesz
  - Egy relé, ami az ajtót nyitja
  - Egy megfelelően választott Arduino a népes családból

A kijelző

- Olcsó 2 soros, 16 oszlopos kijelző (2000 Ft)

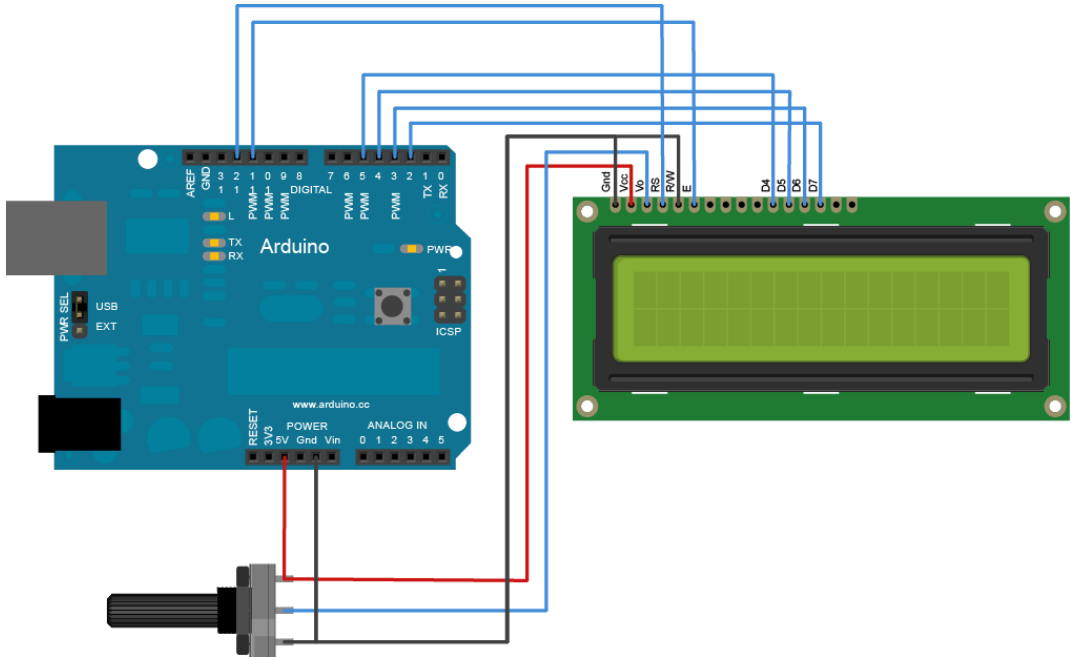


- Lábak:
  - *Adatbusz*: D0...D7, de nekünk 4 bites üzemmódban is jó (D4...D7)
  - *RS*: ezzel lehet jelezni a kijelzőnek, hogy egy ASCII karaktert küldünk neki, vagy vezérlő parancsot
  - *EN*: engedélyezés, a kijelző ekkor olvassa el a neki küldött adatot
  - *RW*: ha nem adatot küldünk a kijelzőnek, hanem tőle kérdezünk le információt – ezt nem kötjük be
  - *Tápfeszültség* (5V)
  - *Kijelző kontraszt* (potméter)
  - *Háttérvilágítás tápellátása*
- 6 adatvezetéket használunk: RS, EN, D7, D6, D5, D4

A kijelző használata

- **Bekötés**:
  Google „arduino display”

- **Használata**:
  LiquidCrystal class példányosítása
  (write(), setCursor(), blink(), clear(), stb.)



- **Példa**:

```
#include <LiquidCrystal.h>
const int numRows = 2;
const int numCols = 16;
// beállítjuk, melyik lábakra kötöttük
LiquidCrystal lcd (12, 11, 5, 4, 3, 2);
void setup () {
   lcd.begin (numCols,numRows);
}
void loop () {
   for (int thisLetter = 'a'; thisLetter <= 'z'; thisLetter++) {
      for (int thisRow = 0; thisRow < numRows; thisRow++) {
         for (int thisCol = 0; thisCol < numCols; thisCol++) {
            lcd.setCursor (thisCol, thisRow);
            lcd.write (thisLetter);
            delay (200);
         }
      }
   }
}
```

A numerikus billentyűzet

- Bárhol beszerezhető, 4x3-mas (900 Ft)

- Lábak:
  - 7 láb, a 4 sorhoz és a 3 oszlophoz
  - Ha megnyomunk egy gombot, rövidre zárja a sor és oszlopvezetékét

A numerikus billentyűzet használata

- Google „arduino keypad”
- **Használata**: **Keypad** class példányosítása (utána getKey(), waitForKey(), getState(), stb.)
- **Példa**:

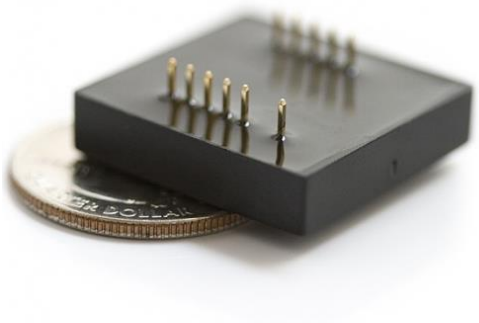
```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {  {'1','2','3'},   {'4','5','6'},   {'7','8','9'},   {'*','0','#'}};
byte rowPins[ROWS] = {32, 22, 24, 28}; // sorok lábait hova kötöttük az Arduino-n
byte colPins[COLS] = { 30, 34, 26 };  // oszlopok lábait hova kötöttük az Arduino-n

Keypad keyPad = Keypad ( makeKeymap (keys), rowPins, colPins, ROWS, COLS );

#define ledpin 13
void setup () {
   pinMode (ledpin, OUTPUT);
   digitalWrite (ledpin, HIGH);
}
void loop () {
   char key = keyPad.getKey();
   if(key)  {
      switch (key)  {
         case '*':
            digitalWrite(ledpin, LOW);
            break;
         case '#':
            digitalWrite(ledpin, HIGH);
            break;
      }
   }
}
```

Az RFID olvasó

- Rádiófrekvenciás azonosítás
- Minden kártyának garantáltan egyedi, 12 hexa karakteres kódja van
- Leolvasó: drágán vettük (7000 Ft), már olcsóbb is van
- Kártya: olcsó (250 Ft/db, kártya, korong, kulcstartó, stb.)



- Soros porton kommunikál
- Lábak:
  - Tápfeszültség (5V)
  - Külső antenna (nekünk a belső bőven elég)
  - Formátum kiválasztó (mi az ASCII módot választjuk)
  - 2 vonal adatátvitelre (csak az egyiket használjuk, soros portként)
  - LED/berregő a kártyaleolvasáskor (nem kötünk rá ilyet)
  - Reset
- Arduinohoz 2 vezetéket kötünk: Reset (digitális lábra), D0 (soros RX lábra)

Az RFID olvasó használata

- Google „arduino id-12”
- **Használata**: soros eszközként

- **Példa**: (reset a 2-es, D0 az RX1 lábra kötve)

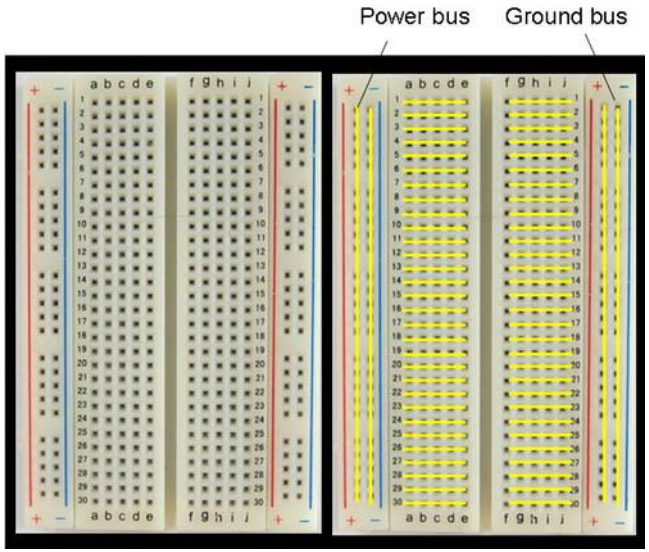
```
const int RFIDResetPin = 2;              // A 2-es digitális lábra kötöttük a leolvasó reset lábát
char userID[13];                         // Ebbe a tömbbe olvassuk majd be a kártya azonosítóját
void setup () {
   Serial.begin(9600);                   // 0-ás soros port beállítása 9600 bps-re (debug-hoz)
   Serial1.begin(9600);                  // 1-es soros port beállítása 9600 bps-re (RFID-t ide kötöttük)
   pinMode(RFIDResetPin, OUTPUT);        // RFID Reset láb beállítása „kimenet” módba
   digitalWrite(RFIDResetPin, LOW);  // RFID reset, felfutó élet csinálunk. Alacsonyba visszük...
   digitalWrite(RFIDResetPin, HIGH); // … majd magasba emeljük.
}
void loop () {
   if (Serial1.available()) {            // Ha jött byte az RFID soros portjától, feldolgozzuk.
      int readByte = Serial1.read();     // Kiolvassuk a következő byte-ot
      if (readByte == 2)                 // Az ASCII 0x2 az „üzenet eleje” karakter
         index = 0;                      // Visszapörgetjünk az indexet, a userID-t az elejétől töltögetjük.
      else if (readByte == 3) {          // Az ASCII 0x3 az „üzenet vége” karakter
         userID[index] = 0;              // Lezárjuk, NULL terminált karakterlánccá tesszük.
         Serial.println(userID);         // Elküldjük a debug rendszernek. Ki fogja írni a PC-n.
      }
      else
         userID[index++] = readByte;     // Ha az üzenet közepén vagyunk, eltároljuk a karaktert.
   }
}
```

A megfelelő Arduino kiválasztása

- Szempontok:
  - Hány digitális ki- és bemenet kell:
    - A kijelzőnek: 6
    - A billentyűzetnek: 7
    - Az RFID olvasónak: 2
    - Összesen: 15
    - A Micro 20-at kezel, de a beszerzéskor még nem létezett.
      Az elődje 14-et tudott, ezért az ADK-t választottuk (54-et tud)
  - További ki- és bemenetek:
    - Analóg be, PWM ki nem kell
    - Soros port kell az RFID-nek
    - Ha debug-olni is akarunk, ahhoz kell még egy soros port
      (USB-n a PC serial monitor-jához)
    - A Micro pont 2 soros portot tud (az elődje csak 1-et tudott)
- Ideális választás: **Micro**

A kapunyitó összeállítása

- Breadboard segítségével, forrasztás nélkül
  - Breadboard: lyukrendszer szabványos lyuktávolsággal és megadott összekötöttséggel:



A kapunyitó szoftvere

- 3 állapotú állapotgépet valósítunk meg:
  - ***start*** állapot: kiinduló állapot, várjuk a kártyalehúzást
  - ***rfid*** állapot: lehúzták a kártyát, az RFID olvasó épp küldi a kártya ID karaktereit
  - ***code*** állapot: számjegy leütésére vár
- A 12 karakteres kártya ID-t összefűzzük a beírt 4 betűs kóddal

  → Így egy 16 karakteres string-et kapunk
- Ha ez a 16 karakter megegyezik a letároltak valamelyikével, akkor kinyitjuk az ajtót

A kapunyitó szoftvere – globális változók

```
#include <LiquidCrystal.h>
#include <Key.h>
#include <Keypad.h>

const int RFIDResetPin = 2;
const int LEDPin = 13;
const int STX = 2;                          // „Start of Text”: a soros porton érkező üzenet ezzel kezdődik
const int ETX = 3;                          // „End of Text”: a soros porton érkező üzenet ezzel végződik
const int CR  = 13;
const int NONE = -1;

char userID[17];                            // ide kerül a kártya ID (12 karakter), utánaírva a lenyomott gombok
int index = 0;
enum estate { start, rfid, code };          // állapotgép aktuális állapota
estate state = start;

LiquidCrystal lcd(49, 47, 48, 46, 44, 42);  // ide kötöttük a kijelzőt

const byte ROWS = 4;
const byte COLS = 3;

char keys[ROWS][COLS] = {  {'1','2','3'},   {'4','5','6'},   {'7','8','9'},   {'*','0','#'} };

byte rowPins[ROWS] = {32, 22, 24, 28};      // ide kötöttük a billentyűzet sor-vezetékeit
byte colPins[COLS] = { 30, 34, 26 };        // ide kötöttük a billentyűzet oszlop-vezetékeit

Keypad keyPad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

// Az elfogadott kártyák és kódjaik. Első 12 karakter: kártya ID, utána írt 4 karakter: a számkód
const char* codes[] = {"010B4CF292261234", "010B4CED58F37899", "010B11C56FB19024", "010B1147E8B41290",
                       "010B11C5F12F7085", "010B112F3F0B0963", "010B11481C4F7412", "010B1148095A3254",
                       "010B1147F3AF6325", "010B114806551589", "010B1147FEA28563", "010B11C56DB33574",
                       "010B4CF0D5637412", "010B4CF26DD96521", "010B4CE9B9164589", NULL};
```

A kapunyitó szoftvere - setup()

- A kötelező **setup ()** függvény:

```
void setup() {

    Serial.begin(9600);            // 0-ás soros port beállítása 9600 bps-re (debug-hoz)
    Serial1.begin(9600);           // 1-es soros port beállítása 9600 bps-re (RFID-hoz)

    pinMode(RFIDResetPin, OUTPUT);   // RFID Reset láb beállítása „kimenet” módba
    digitalWrite (RFIDResetPin, HIGH);    // … majd a Reset jel kiadása
    pinMode(LEDPin, OUTPUT);           // Beépített LED lábának beállítása „kimenet”-re

    lcd.begin(16, 2);              // 16 oszlopos, 2 soros kijelzőnk van
    lcd.print("J\224het a k\240rtya!");     // Írjuk ki rögtön a szöveget
}
```

A kapunyitó szoftvere - loop()

```
void loop () {
        int readByte = NONE;
        char key = keyPad.getKey();                                   // Megnézzük, volt-e gombnyomás
        if (Serial1.available())                                      // Először feldolgozzuk az RFID olvasó által küldött kártya ID-t
           readByte = Serial1.read();                                 // Soron következő karakter
        switch (state) {
           case start:
              if (readByte == STX) {                                  // Ha a kiinduló állapotban megjött az „üzenet eleje” karakter (ASCII 0x2),
                    state = rfid;                                     // átmegyünk a kártya ID olvasó állapotba (rfid)
                    index = 0;
              }
              break;
           case rfid:
              if (readByte == ETX) {                                  // Ha megjött az „üzenet vége” karakter (ASCII 0x3),
                 state = code;                                        // átmegyünk a billentyűzetfigyelő állapotba (code),
                 lcd.clear();                                         // töröljük a kijelzőt,
                 lcd.print(userID);                                   // kiírjuk a kártya ID-t a kijelzőre,
                 lcd.setCursor (0,1);                                 // második sorba megyünk,
                 lcd.print("J\224het a k\242d!");                     // kiírjuk, hogy jöhet a kód
              }
              else if (readByte != STX && readByte != CR && readByte != ETX && readByte != NONE && index<12)    // ha még nincs vége az RFID kód elküldésének
                 userID[index++] = readByte;                          // eltároljuk a soros porton érkezett karaktert
              break;
           case code:
              if (index == 16) {                                      // Ha kódot olvastunk, és vége (mert megvan a 12 betűs kártya ID és a 4 betűs kód),
                 userID[index] = 0;                                   // lezáró 0-t teszünk a végére,
                 Serial.println(userID);                              // kiírjuk a soros terminálra debug célból,
                 checkCardAndCode();                                  // ellenőrizzük a helyességét, és végrehajtjuk, amit kell,
                 state = start;                                       // visszatérünk kiinduló állapotba.
                 lcd.clear ();                                        // Kijelző törlése
                 lcd.setCursor (0,0);                                 // Kurzor a sarokba
                 lcd.print("J\224het a k\240rtya!");                  // Kiírjuk, hogy jöhet a kártya
              }
              else if (state == code && key)                          // Ha jött egy számjegy, de még nem írta be mind a 4-et,
                    userID[index++] = key;                            // hozzáírjuk a többihez
              break;
        }
}
```

A kapunyitó szoftvere - 5

- A kártya és a kód ellenőrzése:

```
void checkCardAndCode () {
   lcd.clear ();                          // kijelző törlése
   lcd.setCursor (0, 0);                  // kurzor a sarokba
   int ix = 0;
   while (codes[ix]) {                    // végignézzük az összes tárolt kódot
      if (!strcmp(userID, codes[ix])) {   // ha egyezik, kész
         lcd.print ("Rendben!");          // szólunk, hogy nyílik a kapu
         digitalWrite (LEDPin, 1);        // kigyújtjuk a LED-et
         break;                           // nem keresünk tovább
      }
      ix++;
   }
   if (!codes[ix])                        // ha a lista végére értünk, és nincs meg,
      lcd.print ("Rossz k\242d!");        // közöljük a rossz hírt

   delay (1000);                          // várunk 1 másodpercet, hogy el lehessen olvasni

   digitalWrite (LEDPin, 0);              // LED lekapcsolása
   digitalWrite (RFIDResetPin, LOW);      // RFID reset, felfutó élet csinálunk. Alacsonyba visszük...
   digitalWrite (RFIDResetPin, HIGH);     // … majd magasba emeljük.
}
```

Összegzés

- Megismertük az Arduino-t
  - Milyen kimenetei/bemenetei vannak
  - Hogyan kell ezekre perifériát illeszteni
  - Hogyan kell a hozzátartozó szoftvert megírni
- Megvalósítottuk az ajtónyitót
  - Bolti, megvásárolható eszközökből
  - Megírtuk a szoftverét