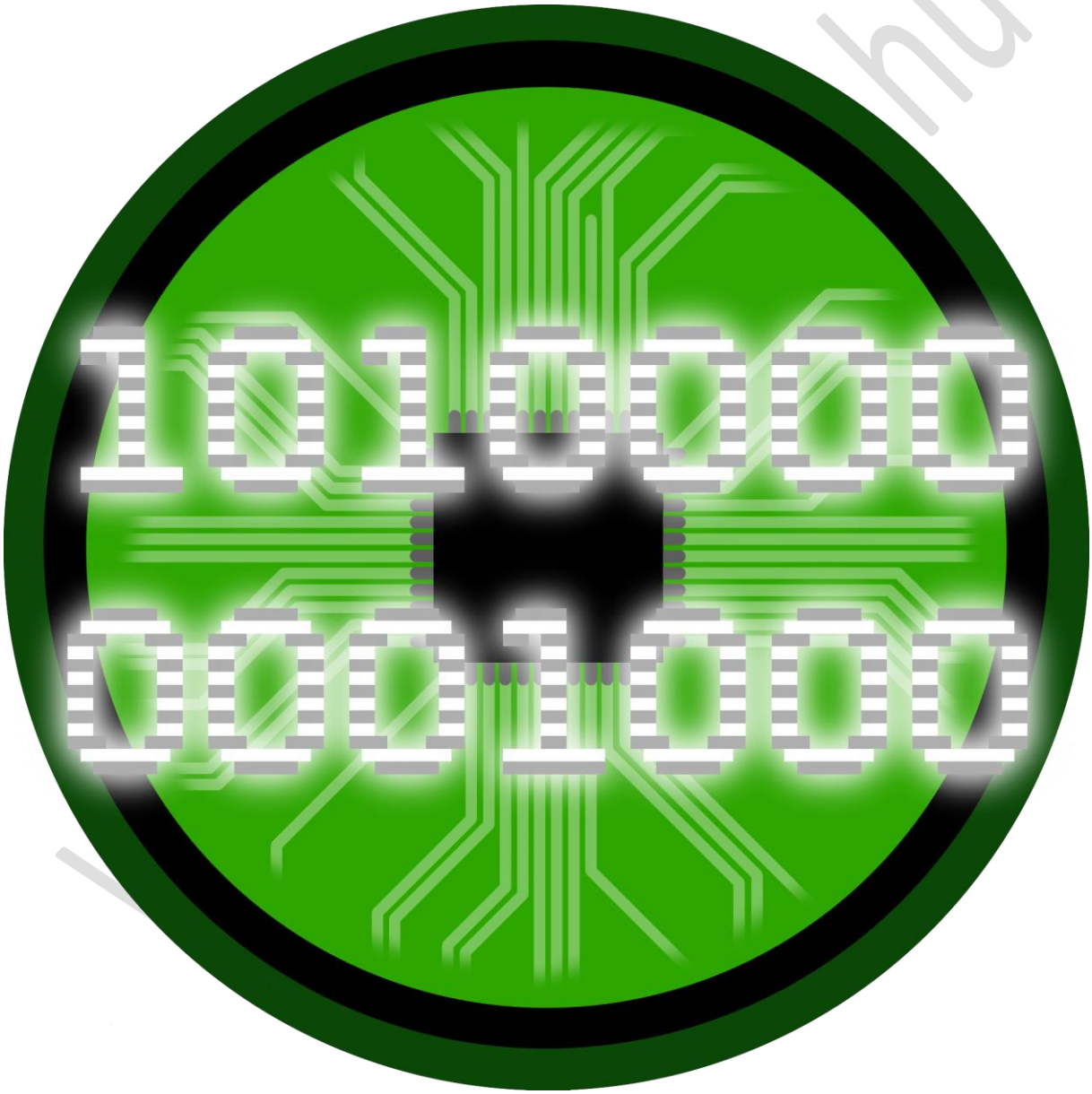
ARDUINO ALAPOK

Segédlet kezdőknek



Tartalom

Arduino alapok

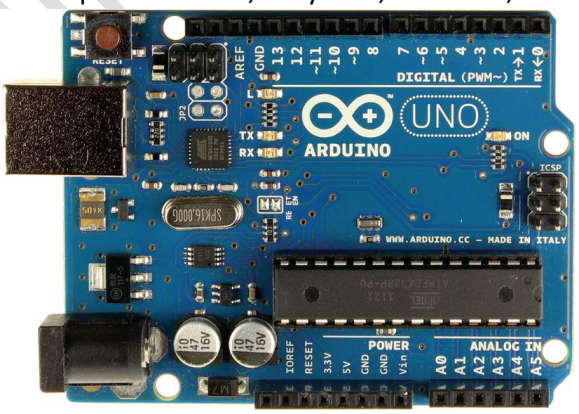
Jelen iromány célja az Arduino fejlesztői modul, ezen belül főként az Arduino Uno használatának bemutatása oktató jelleggel nagyon kezdők, számára.

A modul használatához, elektronikához is érteni kell valamilyen szinten, így igyekszem az elektronikai alkatrészek bemutatását elvégezni Arduino programozási példákon keresztül. Nem árt az előképzettség. A példaprogramok főként azok lesznek, amelyek a modul programozásához szükséges szerkesztőprogramban vannak, néhány egyéni szkripttel kiegészítve. A használt egyéni példaprogramokat megtaláljátok a mellékletben.

Arduino Uno használathoz szükséges kellékek

A modul használatához a következő kellékekre lesz szükség, a példaprogramok során:

1. USB-porttal rendelkező számítógép Windows operációsrendszerrel (más operációsrendszereken futó programmal is lehet a modult programozni, de itt most csak ezt tárgyalom)
2. Arduino skript szerkesztő (https://www.arduino.cc/en/donate/ -oldalról letöltető ingyenesen)
3. 4,5V elem
4. Maga az Arduino Uno modul (beszerezhető különböző oldalakon vagy elektronikai boltokban pl.: arduino.cc, ebay.com, vatera.hu, hestore.hu stb.)



5. USB kábel (függ a típusa az Arduino Uno modul bemeneti csatlakozójától)

6. 270 ohm ellenállás 5db

7. 1K ellenállás 5db

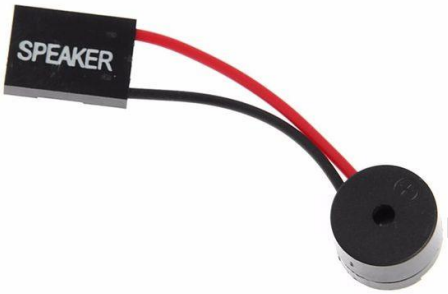
8. 10K ellenállás 3db

9. 100K ellenállás 1db

10. 10K potenciométer 1db

11. Nyomógomb 5db

12. Hangszóró (PC-speekernek használt) 1db

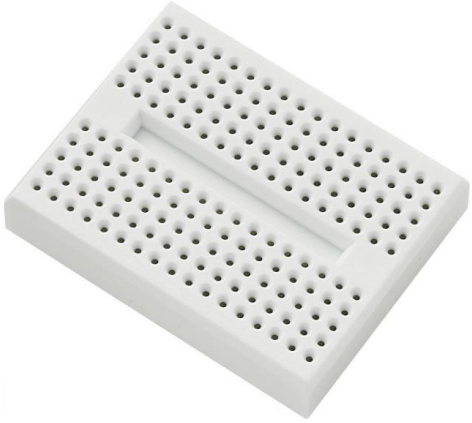


13. Szervo motor SG90 1db

14. LCD KC-1602-GB 1db vagy DEM 16102

    https://www.hestore.hu/prod_10036401.html

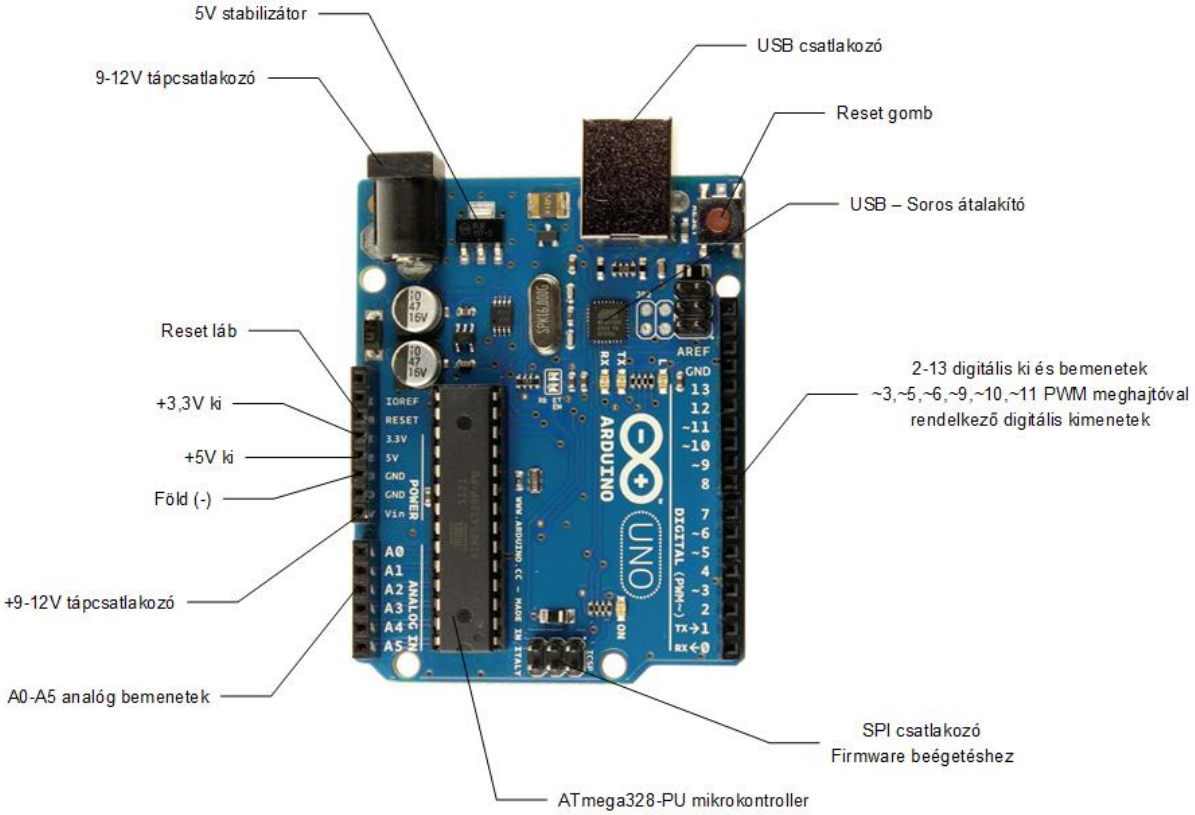
15. Valamilyen próbapanel (pl.: BB-170W)
    https://www.hestore.hu/prod_10035535.html



16. Vezeték csatlakozókkal a végén (pl.: https://www.hestore.hu/prod_10035721.html)

17. LED piros 3mm 5db
https://www.hestore.hu/prod_10020048.html

Arduino Uno felépítése



Jelöletlen lábakat nem tárgyalom, mivel a kapható klón, áramkörökön nem szerepelnek, és következőekben bemutatott példákban sem fognak szerepet játszani. A klón megnevezést a nem az arduino.cc kapható modulokra értem. Legkönnyebben úgy tudjuk megkülönböztetni, hogy amíg az eredetin az USB – soros átalakító egy ATmega8 mikrokontroller, a klónokon valamilyen cél IC (FT232RL vagy olcsóbb) látja el ezt a szerepet. Hangsúlyoznám, mivel az Arduino egy nyílt (opensurce) projekt olyan, hogy klón nem létezik, mivel mind a hardver, mint a szoftver szabadon másolható.

Az Arduino Uno fő mikrokontrollere, amelyen a programunk fut, egy ATmega328-PU (két betűs végződés változik klónok esetében). Aki részleteiben kíváncsi a mikrokontroller adataira, belső felépítésére a Microchip oldalán megtalálja. (https://www.microchip.com/en-us/product/ATMEGA328P)

Vázlat szerkesztő-beégető (Arduino IDE) használata

Első körben ellátogatunk az arduino.cc weboldalra és ott a software (szoftver) menüt választva szembe találjuk, magunkat egy menüvel ahol kiválaszthatjuk az operációsrendszerünknek megfelelő telepítőt. A segédlet írásának pillanatában a szoftver a (Arduino IDE) 1.8.16 –os verziónál tart.

https://www.arduino.cc/en/software

Miután kiválasztottuk a telepítőt a következő oldalon eldönthetjük, hogy csak simán letöltjük a programot (Just download) vagy adakozunk tetszés szerinti összeget a program fejlesztőinek.

Letöltés után telepítjük az Arduino IDE-t a letöltött exe futtatásával melynek varázslóját végig követve egy Arduino logót ábrázoló ikonnal gazdagodik az asztalunk ikonrengetege.

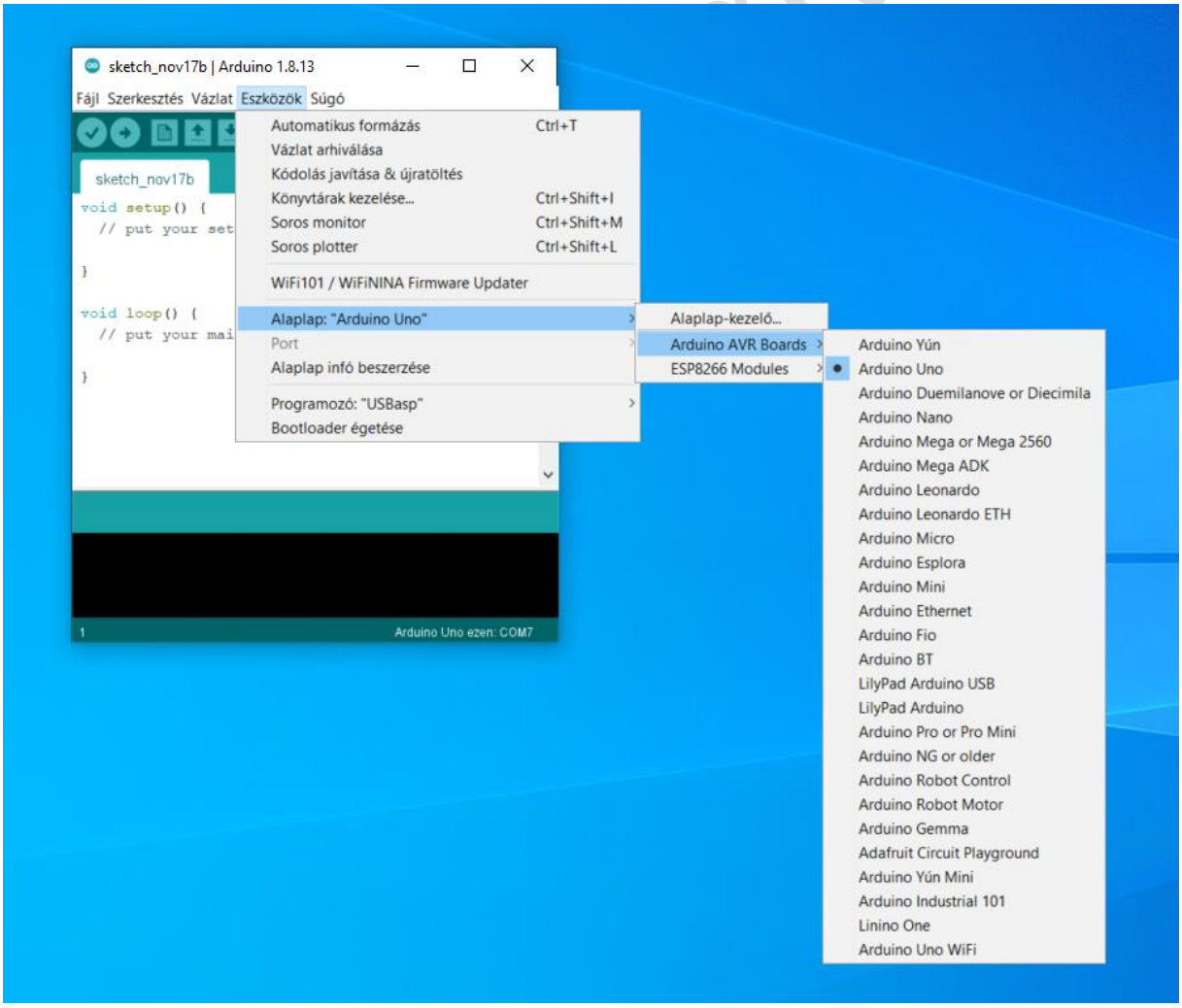
USB kábelünkkel csatlakoztatjuk a számítógéphez az Arduino Uno modult, ami jó esetben villogni kezd a rajta található LED-ekkel. 13-lábon található LED folyamatosan villogni fog ugyanis alapból a Blink nevű vázlat van a modulba feltöltve. Csatlakoztatás után a számítógépünk egy új hardvert érzékel és soros portot telepít. A soros port számát nézzük meg az eszközkezelőben.

Vezérlőpult -> Eszközkezelő ->LPT- Comportok

Az ikonra kattintva, a program elindul és egy szerkesztő felülettel találkozunk. Megnyitjuk az Eszközök menüt, legörgetünk az Alaplap, majd az abból megnyíló AVR Boards menüre és kiválasztjuk az Arduino Uno modult.

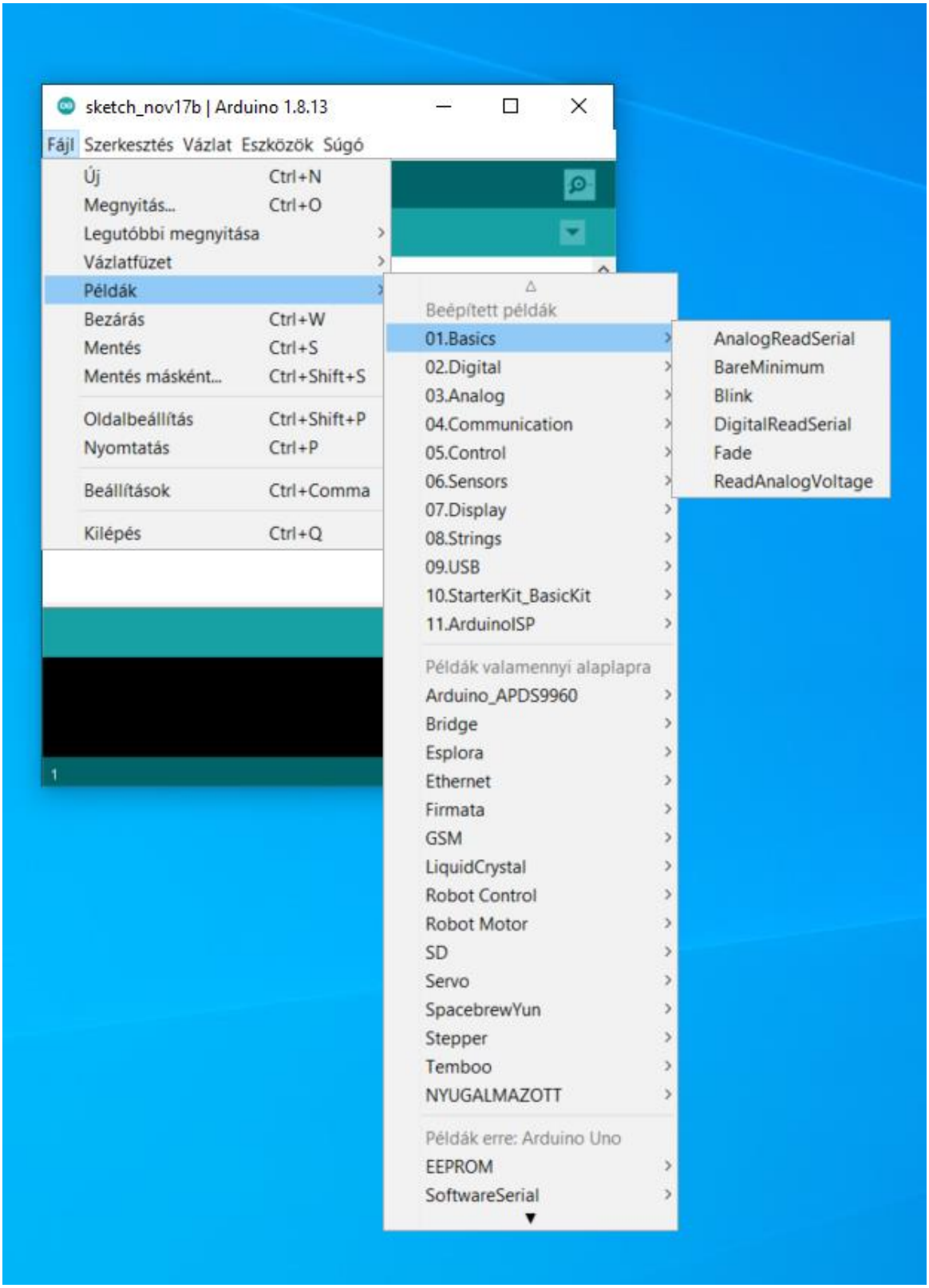
Rögtön az Alaplapok menüpont alatt Port menüpont található ahol kiválasztjuk az eszközkezelőből kilesett com-portot.

Készen állunk az első vázlatunk (szkript) feltöltésére.



Az én esetemben COM7 volt a kommunikációs port, de ez nálatok másképp alakulhat.

Példa vázlatokat a Fájl -> Példák menüpont alatt találjuk. Jelen esetben a Blink példát fogjuk kiválasztani majd az Arduino Uno modulra feltölteni. Ahogy a lenti ábrán látjuk:



Blink (Villog) vázlat annyit tud, hogy villogtatja 1 másodpercenként az Arduino Uno modulon található (13 digitális kimenet) beépített LED-et. (Vázlat megnyitáskor a program új ablakot nyit.)

Arduino IDE C nyelvet használ, így ennek a nyelvnek az alapjait fogjuk megkarcolgatni ☺

Programunk egy hosszú komment résszel indul, amelyben szöveg található le van írva a vázlat célja, ki írta miért, és hogy milyen licencezési jogok vonatkoznak a vázlatra.

Kommentelés: /* szöveg */ ami a /**/ karakterpáros közé kerül, a fordító nem értelmezi programkódként, akár ki is törölhetnénk, a komment lehet több soros, célja hogy feljegyzéseket készítsük magunknak vagy másoknak, hogy mit miért írtunk a vázlatba

Egy soros kommentelés: // ami // karakterpáros után van az csak megjegyzés – csak egy soros lehet

Vázlat 2 fő része:

1. Beállítás (void setup()):

void setup() {

// initialize digital pin LED_BUILTIN as an output.

pinMode(LED_BUILTIN, OUTPUT);

}

Ebben a részben adjuk meg, hogy melyik lába az Arduino Uno modulnak lesz kimenet vagy bemenet a program futása során, valamint a változókat és ezek kezdő értékét.

Jelen esetben csak egy kimenetet adunk meg, ami speciális és csak a beépített LED-re vonatkozik. Csak erre használható.

pinMode - láb beállítás

LED_BUILTIN – beépített LED

OUTPUT – kimenet

pinMode(LED_BUILTIN, OUTPUT);

Minden, ami a void setup() utáni zárójelek között van {} a programunk elején és csak egyszer fog végrehajtódni

2. A program maga void loop():

// the loop function runs over and over again forever

void loop() {

digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

}

A program e része folyamatosan fut, amíg a modulunk tápellátása meg nem szakad, vagy reset gombot meg nem nyomjuk.

További sorok magyarázata:

digitalWrite(LED_BUILTIN, HIGH); - 13-láb magasra kapcsolása, azaz 5V-ra (LED bekapcsol)

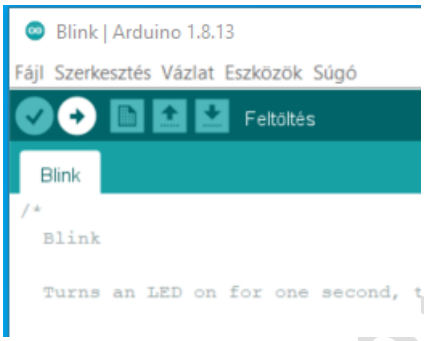
delay(1000); - várakozás 1000 mS (ezred másodperc)

digitalWrite(LED_BUILTIN, LOW); -13-láb alacsonyra kapcsolása 0V-ra (LED kikapcsol)

delay(1000); - várakozás 1000 mS (ezred másodperc)

Első program feltöltésre készen állunk, esetleg módosíthatjuk a 2 delay (késleltetés) hosszúságát pl.: az egyiket írjuk át 1500 mS-ra.

Ha ezzel megvagyunk, akkor nyomjuk mega feltöltés gombot. A vázlatunk befordítódik, ellenőrzés alá kerül szintaktikailag, majd feltöltődik a modulba.



Ha mindent jó csináltunk a modulon a soros kommunikációt jelző LED-ek őrült villogásba kezdenek, majd a vázlatunk feltöltődik, a modul újraindul és a modulon lévő beépített LED villogni kezd a késleltetéseknek megfelelően. Ezzel el lehet játszani, hogy begyakoroljuk a használatot.

Fontos tudni, hogy minden feltöltéskor a program jelenlegi állapota mentődik el automatikusan, így érdemes készíteni egy másolatot, mielőtt feltöltenénk. (Fájl menü -> Mentés másként…)

Rontsuk el szintaktikailag a programot és nézzük meg milyen hibákat kapunk, hogy bonyolultabb eseteknél tisztában legyünk azzal melyik hiba mit jelent.

Egy hibajelzés így néz ki:

A program jelzi, hogy hol talált problémát, és javaslatot tesz a javításra.

Ellenállás, LED használata, futófény programozása

A megvásárolandó alkatrészek között a segédlet elején található Arduino használatához szükséges kellékek fejezett alatt, található néhány ellenállás. Ebben az esetben a 270 ohm -ra lesz szükség.

Aki kíváncsi az ellenállás fogalmára Wikipédián megtalálja.

https://hu.wikipedia.org/wiki/Elektromos_ellen%C3%A1ll%C3%A1s

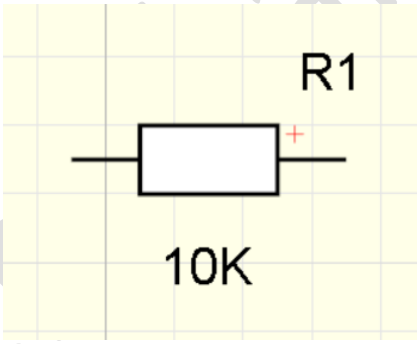
Röviden:

Az ellenálláson átfolyó áram hatására az ellenállás két vége közt az áram nagyságával egyenesen arányos feszültség mérhető.

Az ellenálláson eső feszültség kiszámítása (Ohm-törvény): $U=R*I$, ahol

- *U* az ellenálláson eső feszültség,
- *R* az ellenállás,
- *I* pedig az áram nagysága.

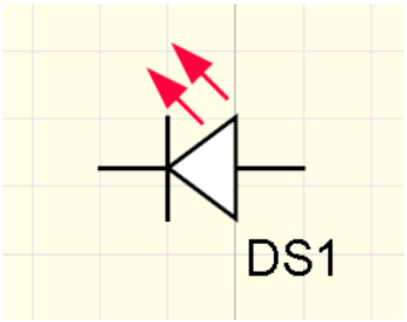
Ellenállás áramköri jele:



Szükségünk lesz 5db 3mm piros LED-re is.

A LED egy félvezető dióda, amelydióda lévén csak egy irányba vezet, valamint fényt bocsát ki.

Led áramköri jele:



A piros LED üzemi feszültsége 1.8-2.4 V ebből látható, hogy bajban vagyunk, ugyanis az Arduino minden kimenete magas állapotban **5V** feszültségen van. Ha rákötjük a LED-et, nagy áramot

fog felvenni és a mikrokontrollerünk tönkremegy. Áramkorlátozásra van szükség, így lép a képbe az ellenállás.

Arduino **40mA tud leadni egy kimeneten**. Megvizsgáljuk a LED-ünk üzemi áramát, az adatlapján, amely a webshop oldalán általában megtalálható, ami 20mA.

Ohm törvényével kiszámoljuk a szükséges ellenállás értéket:

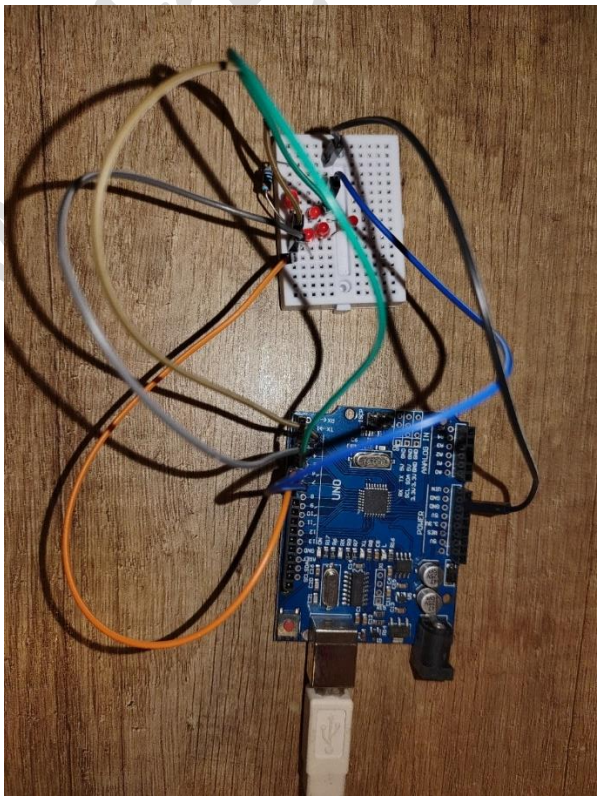
R=U/I

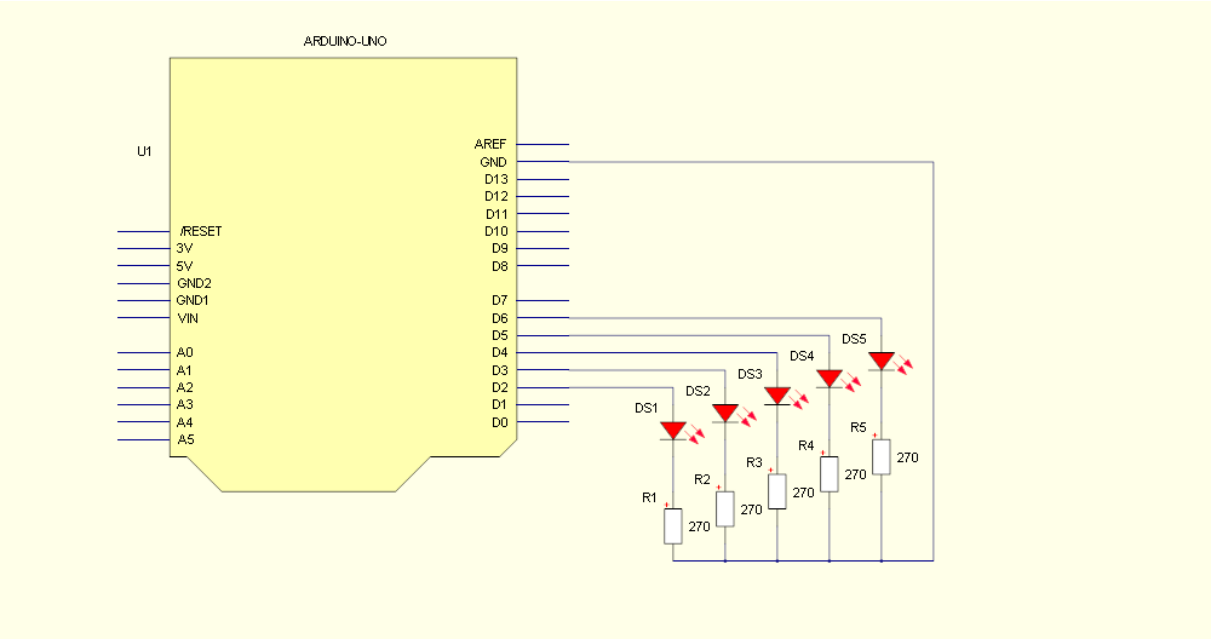
R=5V/0,02A *(mindig alapmértékegységben számolunk)*

R=250 Ω - ohm

Tehát a készletünkben lévő 270 ohm-s ellenállás megfelelő ugyanis, ha visszaszámoljuk az előző képlettel a 270 ohm-on 5V feszültség esetében áthaladó áramot az (I=U/R) 0,0185A, ami 18,5mA.

Az ellenállást sorba kötjük a LED-el majd csatlakoztatjuk az Arduino modulunkhoz, úgy hogy a hosszabbik lába legyen a 2 digitális kimenet felé (LED polaritás függő) a másik lába GND-re csatlakozzon. Ismételjük meg a 3,4,5,6 kimeneten. Használjuk a próbapanelünket, a lenti ábra szerint.





Nyissuk meg az Arduino IDE-t, kattintsunk a Fájl -> Új vázlat menüpontra és Mentés másként menüvel egy futofeny nevezetű könyvtárban mentsük el futofeny.ino néven. (Az ékezeteket szándékosan mellőztem, ugyanis a magyar karaktereket a szoftver nem mindig kezeli jól.)

A következő ablakkal találkozunk:

Itt kell bevezetnünk a **változó** fogalmát.

A változók több típusúak lehetnek, és a választásunk attól függ, hogy milyen számot vagy szöveget esetleg igaz-hamis állapotot tárolnánk benne. Ebből tudja fordító program mennyi memóriát foglaljon le az adott változónak.

Jelen esetben egy **int,** változóval dolgozunk, ami az angol integer, azaz egész szóból származik. Tehát az egész számok halmazába tartozó számot lehet megadni ebben a változóban -32 768 -tól 32 767 –ig. (Arduino Uno esetében. Mindegyik változó típust annál a program példánál tárgyalok ahol először használva lesz.)

Megadjuk, milyen kivezetések (pinek) lesznek használva:

```
int led1 = 2;  // LED 2 kimentre kötve
int led2 = 3;  // LED 3 kimentre kötve
int led3 = 4;  // LED 4 kimentre kötve
int led4 = 5;  // LED 5 kimentre kötve
int led5 = 6;  // LED 6 kimentre kötve
```

Ha a teljes programfutási idő alatt használjuk a változókat, ajánlott a program elején rögtön felvenni őket.

Megadjuk a kivezetések szerepét:

```
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
pinMode(led5, OUTPUT);
```

```
pinMode(Arduino kivezetés, Mód);
```

Mód lehet:

1. **INPUT** – bemenet (ha valami szenzort, kapcsolót stb. be akarunk olvasni)
2. **INPUT_PULLUP** – bemenet, amely a mikrokontrollerben található 10K ellenállással fel van húzva 5V-ra. Főleg nyomógombok kapcsolók esetén jön jól. Későbbi példában bemutatásra kerül.
3. **OUTPUT** - kimenet

void loop –on belül megírjuk az utasításainkat, amelyek folyamatosan zárt hurokkén hajtódnak végre a tápfeszültség megszakításáig.

```
digitalWrite(led1, HIGH);  // LED1 bekapcsolasa
delay(500);                // varakozas 500 mS (fél másodperc)
digitalWrite(led1, LOW); // LED1 kikapcsolasa
```

```
delay(500);
digitalWrite(led2, HIGH);  // LED2 bekapcsolasa
delay(500);          // varakozas 500 mS (fél másodperc)
digitalWrite(led2, LOW); // LED2 kikapcsolasa
delay(500);
digitalWrite(led3, HIGH);  // LED3 bekapcsolasa
delay(500);          // varakozas 500 mS (fél másodperc)
digitalWrite(led3, LOW); // LED3 kikapcsolasa
delay(500);
digitalWrite(led4, HIGH);  // LED4 bekapcsolasa
delay(500);          // varakozas 500 mS (fél másodperc)
digitalWrite(led4, LOW); // LED4 kikapcsolasa
delay(500);
digitalWrite(led5, HIGH);  // LED5 bekapcsolasa
delay(500);          // varakozas 500 mS (fél másodperc)
digitalWrite(led5, LOW); // LED5 kikapcsolasa
delay(500);
```

digitalWrite(Arduino kivezetés, Mód);

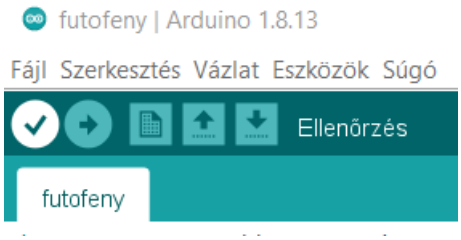
A fenti paranccsal megadjuk, hogy melyik láb kapcsoljon be vagy ki ahol:

Mód lehet:

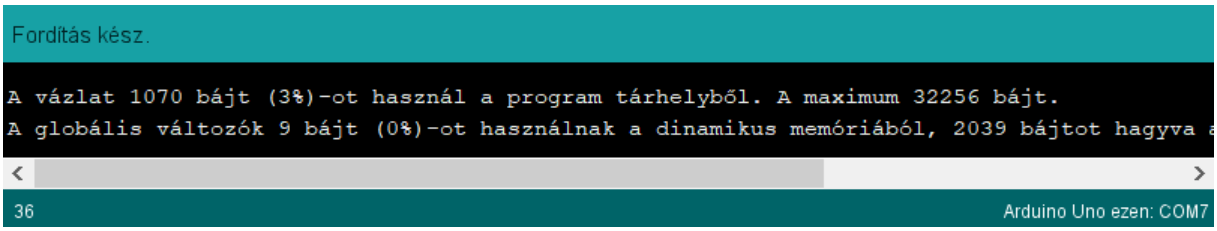
1. **HIGH** – magas azaz bekapcsolt állapot 5V
2. **LOW** – alacsony azaz kikapcsolt állapot 0V

Mit észleltétek, minden sor végére „;” kerül.

Ezzel készen is vagyunk az első programunkkal, ami egy nagyon egyszerű futófény, leellenőrizhetjük a kódolásunk helyességét.



Ha a program mindent rendben talál, ezt látjuk az ablak alján:



Töltsük fel a programunkat az Arduino Uno modulra:

Ha mindent jól csináltunk a LED-ek egyenként világítanak. Eljátszogathatunk a késleltetések módosításával vagy több LED bekapcsolásával így különböző effekteket kapunk, mindenki fantáziájára bízom, mit alkot belőle.

For ciklus

A fenti programot villámgyorsan le tudjuk egyszerűsíteni for ciklussal.

Nyissunk egy új vázlatot, mentsük el futofeny_for (vagy kinek mi tetszik) néven. Az áramkörünkön nem kell változtatnunk. Az előző futófényt meg tudjuk alkotni sokkal rövidebben.

Az első for ciklusunk a void setup –ba kerül, beállítjuk a kivezetéseket kimenetnek, a második a void loop-ba kerül.

For ciklus:

```
for (beállítás; állapot; művelet) {
// további parancsok;
}
```

Lényege: a ***beállításban*** megadott változón addig hajts végre a megadott ***műveletet*** ameddig el nem éri a megadott ***állapotot.***

```
   for(int i=2; i<=6; i++){
pinMode(i, OUTPUT);
   }
```

Tehát megadunk egy i változót, ami egyenlő 2 –el. (i++ ugyanaz, mint i=i+1)Hozzáadunk 1-et amíg 6 nem lesz.

Vázlatunk:

```
void setup() {
  // put your setup code here, to run once:


  /*Beállítjuk a lábak szerepét ami lehet kimenet vagy bemenet,
   * a mi esetünkben kimenet
   */
   for(int i=2; i<=6; i++){
pinMode(i, OUTPUT);
   }

}

void loop() {
  // put your main code here, to run repeatedly:
  for(int i=2; i<=6; i++){
digitalWrite(i, HIGH);   // LED bekapcsolasa

delay(500);               // varakozas 500 mS (fél másodperc)

digitalWrite(i, LOW); // LED kikapcsolasa

delay(500);
  }


}
```

A két program ugyanazt az eredményt éri el, viszont amíg az első 3% használ el a program memóriából, addig a második csak 2%. Ez akkor lesz lényeges, amikor alig fér a programunk a korlátozott tárhelyünkre. Ami az Uno esetében 32Kb.(kilobájt) Mellékletben megtalálható a **futofeny_for_vissza.ino**, melyben látható hogyan fordítható vissza a fény futása for ciklussal.

While ciklus

Itt meg ragadnám az alkalmat, hogy beszéljünk egy kicsit a „while” ciklusról, ami az angol „amíg” szó. Magyarul a lényege, hogy hajtsd végre, ami a kapcsos zárójelben van, amíg a megadott feltétel nem teljesül.

Nézzünk, egy példát ahol felhasználjuk az előző programunkat:

```
int z=0; //lefutások száma

void setup() {

    // put your setup code here, to run once:

    /*Beállítjuk a lábak szerepét ami lehet kimenet vagy bemenet,

  * a mi esetünkben kimenet

  */
```

```
        for(int i=2; i<=6; i++){
                pinMode(i, OUTPUT);
        }
}
void loop() {
        // put your main code here, to run repeatedly:
        while(z<15){  //addig fusson amíg z kisebb 15-nél
                for(int i=2; i<=6; i++){
                        digitalWrite(i, HIGH);  // LED bekapcsolasa
                        delay(500);          // varakozas 500 mS (fél másodperc)
                        digitalWrite(i, LOW); // LED kikapcsolasa
                        delay(500);
                        z++; //növeld Z-t 1-el
                }
        }
}
```

Mint látható az a program, csak annyiban módosult, hogy kapott egy z változót, ami a tulajdonképpeni **void loop()** lefutások számát tárolja, valamint egy **while** ciklust, ami meghatározza, hogy az utána jövő parancsok tizenötször fussanak le. Ezért van az, hogy a futófény csak háromszor fut végig. Továbbá jó képet kapunk a ciklusok lefutásának miként jéről is.

If, else if, else feltételes utasítás

Futófény programunkat úgyis megírhatjuk, hogy magát a void loop –ot használjuk fel.

```
int  i=2; //kezdő kivezetés
int leptet=1; //kivezetés számát léptetjük ennyivel

void setup() {
  // put your setup code here, to run once:


  /*Beállítjuk a lábak szerepét, ami lehet kimenet vagy bemenet,
   * a mi esetünkben kimenet
   */
   for(int i=2; i<=6; i++){
pinMode(i, OUTPUT);
   }

}
```

```
        void loop() {
        // put your main code here, to run repeatedly:

        digitalWrite(i, HIGH);    // LED bekapcsolasa

        delay(100);               // varakozas 500 mS (fél másodperc)

        digitalWrite(i, LOW);     // LED kikapcsolasa

        delay(100);

        i=i+leptet;

        if(i==6 || i==2 ){ //ha a 6 kimenet vagy a 2 kimenet aktív
        leptet=-leptet;    //fordítsd a léptetést +1-ről -1-re és vissza
        }
}
```

Megvalósítottunk egy oda-vissza futófényt.

```
if (feltétel) {
  //utasítások
}
```

Behoztuk a képbe az **if** feltételes utasítást, ami magyarul a „ha” szónak felel meg. Tehát ha a zárójelben található feltételünk érvényesül, akkor végrehajtódnak a kapcsos zárójel között szerepelő utasítások.

Feltételek fajtája:

Fontos! Csak azonos típusú változókat hasonlíthatunk össze. pl.: int típusút int-el, string típusút stringgel stb.

Változó egyenlővé válik egy másikkal vagy egy értékkel.

i==6

i>=6

i<=6

i==x+y

Feltűnik még **||** logikai operátor használata ami a vagy szónak felel meg. Vagy a jobboldalán vagy a baloldalán található kifejezés igaz.

3 típusú logikai operátor létezik, az előző || vagy, && és, valamint a ! nem. && esetén mindkét kifejezésnek igaznak kell lenni, míg a ! negálja a mögé kerülő kifejezést, invertálja a mögé kerülő változót. (i!=x i nem egyenlő x –el)

Ezúttal alkossuk újra futófény programunkat, csupán if feltételes utasításokkal. Ezúttal a két szélső LED felől a középső felé futó effektet kapunk. Mentsük el programunkat futofeny_if.ino néven és alakítsuk át a következő módon:

```
int  i=1;

        void setup() {
```

```
 // put your setup code here, to run once:
 /*Beállítjuk a lábak szerepét ami lehet kimenet vagy bemenet,
  * a mi esetünkben kimenet
  */
        for(int i=2; i<=6; i++){
                pinMode(i, OUTPUT);
        }
}
        void loop() {
 // put your main code here, to run repeatedly:
if(i==1){
digitalWrite(2, HIGH);   // LED1 bekapcsolasa
digitalWrite(6, HIGH);   // LED6 bekapcsolasa
delay(200);              // varakozas 200 mS
digitalWrite(2, LOW); // LED1 kikapcsolasa
digitalWrite(6, LOW); // LED6 kikapcsolasa
delay(200);         // varakozas 200 mS
}
else if(i==2){
digitalWrite(3, HIGH);   // LED2 bekapcsolasa
digitalWrite(5, HIGH);   // LED4 bekapcsolasa
delay(200);          // varakozas 200 mS
digitalWrite(3, LOW); // LED2 kikapcsolasa
digitalWrite(5, LOW); // LED4 kikapcsolasa
delay(200);
}
else{
digitalWrite(4, HIGH);   // LED3 bekapcsolasa
delay(200);              // varakozas 200 mS
digitalWrite(4, LOW); // LED3 kikapcsolasa
delay(200);
```

```
}
i++; //1-et adjunk hozzá minden lefutásnál
if(i==4){ //ha i=4 akkor i=1 ugorjon a fény az elejére
  i=1;
}
}
```

A program lényege, hogy a main loop segítségével léptejük az **i** változót 1-től 4-ig amikor eléri a 4-et akkor vissza ugrunk a ciklus elejére mivel egyenlővé tesszük 1-el.

```
i++; //1-et adjunk hozzá minden lefutásnál
if(i==4){ //ha i=4 akkor i=1 ugorjon a fény az elejére
  i=1;
}
```

Ezzel a ciklussal vezéreljük a LED-eket.

3 ágú feltételes utasításra van szükségünk. Ha i=1 akkor az 1-5 LED világít (if ág), ha i=2 a 2-4 LED világít (else if ág), ha i=3 a 3 LED világít (else ág).

Ilyen típusú feltételes utasítás rendelkezhet több ággal, lényeg hogy mindig if-el kell, kezdődjön, else if ágból bármennyi lehet benne, valamint else ággal kell befejeződnie. A fenti programot 3 db if el is meg lehet valósítani. Fontos még megjegyezni, hogy az if valamint else if kiemelt esetekre, else minden fennmaradó estre vonatkozik. A programunk azért működik jól, mert az i változónk csak az 1,2,3 értékeket veheti fel.

Módosítsuk a programot és engedjük i változónkat elszámolni 10-ig:

```
if(i==10){ //ha i=10akkor i=1 ugorjon a fény az elejére
```

Rögtön látjuk, hogy az else ág lesz érvényes minden 2 felett felvett értékre és a 3-as LED villog többször majd ismétlődik a folyamat.

Gyakorlás céljából játsszál el a program módosítgatásával, hogy különböző effekteket kapjál.

Impulzusszélesség moduláció, analogWrite() használata

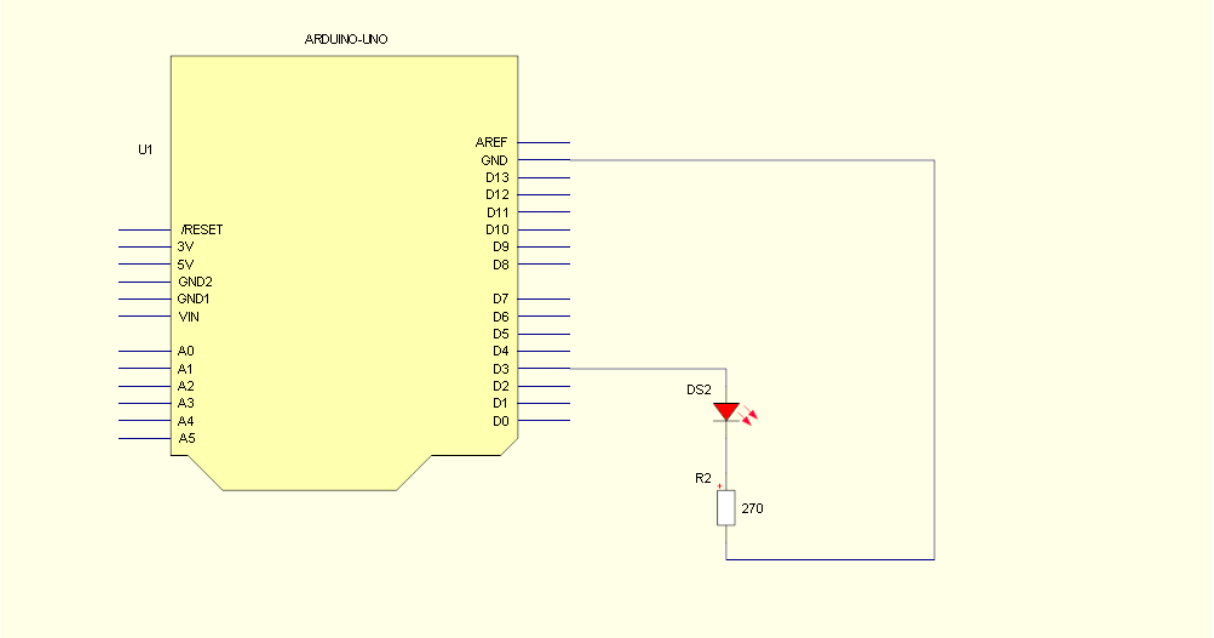
Ezúttal az egyik LED-et amely a 3- kivezetésre van csatlakoztatva, lassan elhalványítjuk majd újra lassan teljes fényerőre kapcsoljuk.

Ez úgy lehetséges, hogy az Arduino Uno rendelkezik néhány PWM kivezetéssel (Pulse Width Modulation) magyarul impulzus szélesség moduláció. Nevezetesen a 3,5,6,9,10,11 kivezetések képesek erre a varázslatra.

Az impulzus szélesség moduláció a következő képen valósul meg:

A fogyasztóba táplált feszültséget (mi esetünkben 5V) és áramerősséget a táp és a fogyasztó között lévő kapcsoló (Arduino PWM kivezetés) gyors ütemű be- és kikapcsolásával szabályozzák. Minél hosszabb ideig van a kapcsoló a bekapcsolt állapotban a kikapcsolt állapot időtartamához képest, annál nagyobb lesz a fogyasztóba táplált teljesítmény. Tehát a LED annál jobban világít.

Szedjük szét az előző áramkörünket, csak a 3-as kivezetésen hagyjuk csatlakoztatva a LED-et a 270 ohm áramkorlátozó ellenállással. A lenti ábra szerint:



A következő vázlatot töltsük a modulunkba, lényegében a példák között megtalálható Fade.ino magyarított változata, melyet a 3-as kivezetésre módosítottam:

```
int led = 3;          // a PWM kivezetés száma amire a LED van kötve

int fenyero = 0;    // a LED fényereje

int valtozz = 5;    // egy lefutás alatt mennyit sötétítsünk

        void setup() {

        // felvesszük a kimenetet:
```

```
 pinMode(led, OUTPUT);
}
void loop() {
        analogWrite(led, fenyero); //fényerő beállítása (a lényeg)
        // változtasd a fényerő értékét a következő lefutáshoz:
         fenyero = fenyero + valtozz;
        // fordítsd a változást:
         if (fenyero <= 0 || fenyero >= 255) {
        valtozz = -valtozz;
         }
 // 30 mS késleltetés
 delay(30);
}
```

A fenti programot az analogWrite() parancs miatt mutattam be:

analogWrite(kimenet száma, 0 tól 255-ig egész szám);

0 (mindig kikapcsolt) és 255 (mindig bekapcsolva) – ciklus beállítása.

Természetesen bármelyik digitális kimenetre leprogramozható az impulzus szélesség moduláció, de ez így sokkal egyszerűbb, és hardverből megy. Használható motorok sebességének a szabályozására, fűtés szabályozásra, szervomotorok vezérlésére, stb.

Soros kommunikáció

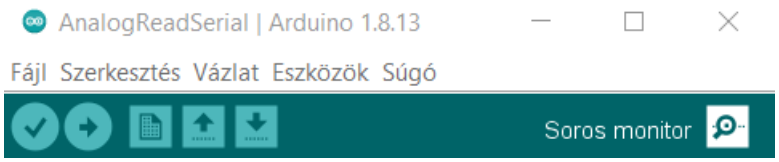
Az Arduino Uno képes soros kommunikációra a D0 – RX és a D1 – TX kivezetésen keresztül, ezekre a kivezetésekre csatlakozik a modulon található soros-USB átalakító. Mivel a modult ugyanezen a két kivezetésen programozzuk fel így a beégetett programunkkal is kommunikálhatunk ezen keresztül. Programunkban hibakeresésre is használhatjuk, amikor logikai hibáink vannak.

RX kivezetés jelölése az angol Receive azaz fogadás szóból ered.

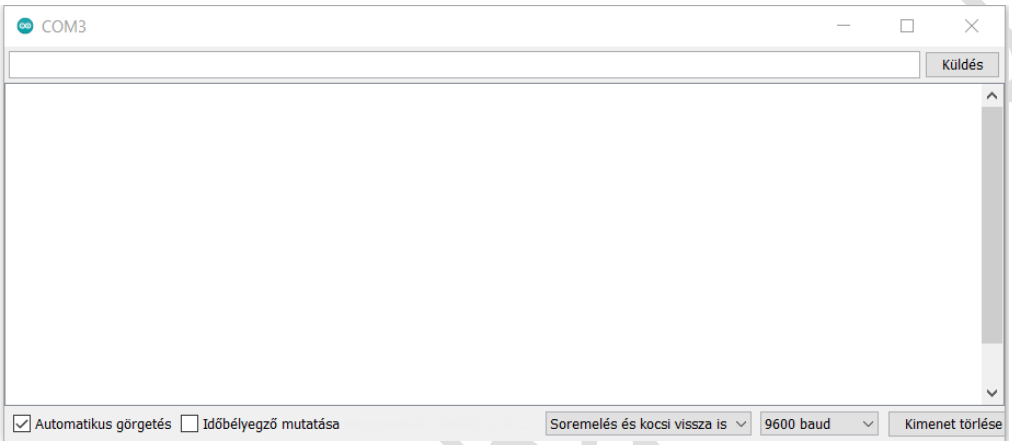
TX kivezetés jelölése az angol Transmit azaz küldés szóból ered.

Adatküldés modulból a számítógép felé:

A számítógépen egy soros terminál programra van szükségünk az ilyen kommunikációra, az Arduino IDE szoftverünk tartalmaz is egy ilyet:



Jobb oldat található nagyítós gombra kattintva tudjuk megjeleníteni a Soros monitor ablakot, amennyiben a modulunk csatlakoztatva van az USB kábellel a számítógéphez. Az ablak felső részében található sávba beírt adatokat „Küld” gomb segítségével tudjuk a modulnak továbbítani, az alatta található szövegmezőben jelennek meg az Arduino Uno –ból jövő adatok.



Az ablak alján tudjuk beállítani a kommunikáció sebességét, a kommunikáció megvalósulásához a modul és a számítógép is azonos sebességen kell, kommunikáljon, ami a vázlatainkban 9600 baud lesz. Soros kommunikáció esetén a választott sebesség főként a kábel hosszúságtól függ, minél hosszabb kábelt használunk, annál alacsonyabb sebességet kell választani. Ennek a részletezésébe nem megyek bele, ugyani csak alap dolgok bemutatásával foglakozok ebben az irományban.

Arduino Uno-ba töltsük fel a következő programot (mellékletben Soros_kommunikacio1.ino):

```
int i=0;
String sz = „Kezdjük, előröl”;
void setup() {
 Serial.begin(9600);//kommunikáció inicializálás és sebesség beállítás
 delay(10);
 Serial.println(„Soros kommunikáció Arduino Uno -> Számítógép”);
}
void loop() {
    Serial.print(i); //kírjuk i változót a terminálon
    Serial.print(„„”); //kieszünk egy vesszőt (string)
```

```
    delay(500);
    i++; // i vátozót növeljük 1-el
    if(i==11){
    Serial.println(sz); //kiírjuk sz szöveges változót
    i=0;
    }
}
```

Serial.begin(9600); - kommunikáció inicializálás és sebesség beállítás, sebesség 9600 baud

Serial.print(kiírandó adat); – adat küldése

Serial.println(kiírandó adat); – adat küldése + új sor karakter küldése (új sort kezd)

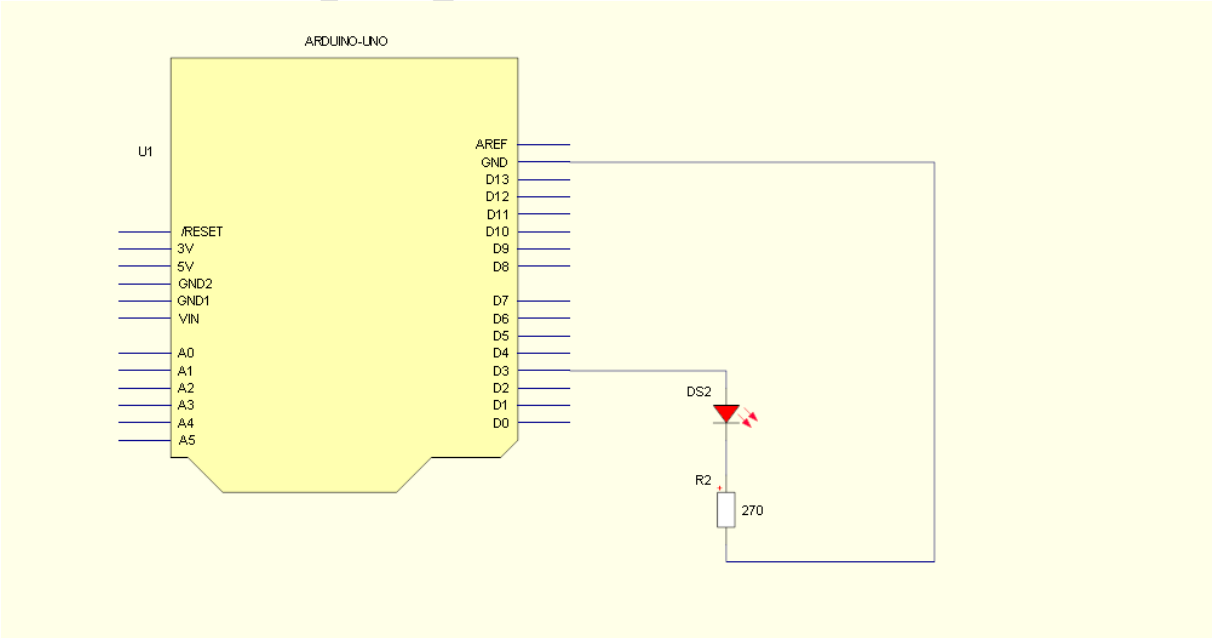
Bevezettünk egy új típusú változót, ami egy szöveges változó:

String sz = „Kezdjük, előröl!”; (szöveges változókat kerüljük, mert sok helyet emésztenek fel)

Feltöltés után, nyissuk meg a „Soros monitor ablakot” a számítógépen, és látni fogjuk, hogy a programunk elszámol 10 –ig kiírja, hogy kezdjük, előröl, és új sort kezd. Ezt csinálja a végtelenségig.

Adatküldés számítógépből a modul felé:

Egy nagyon egyszerű távirányítási példán keresztül fogjuk megnézni hogyan, tudjuk billentyűzetről kapcsolgatni a LED-et amit az Arduino D3-as kivezetésére kötöttünk.



Készítünk egy olyan programot, amely a számítógép billentyűzetéről küldött „1” re felkapcsolja a LED-et, „0”ra lekapcsolja azt. (Mellékletben vezerles.ino néven található.)

```
int led = 3;  // LED 3 kimentre kötve
void setup() {
        pinMode(led, OUTPUT); //kimenet inicializálás
         Serial.begin(9600);  //soros komunikáció beállítás
        }
void loop() {
         char karakter = Serial.read(); //soros bemenet olvasása karakterenként
         if (karakter=='1') { //ha a beolvasott karakter 1
                 Serial.println(karakter); //karakter kiíratása terminálra
                 digitalWrite(led, HIGH); //led bekapcsolása
                         }
         if (karakter=='0') { //ha a beolvasott karakter 0
                 Serial.println(karakter); //karakter kiíratása terminálra
                 digitalWrite(led, LOW); //led kikapcsolása
                         }
 delay(100);
}
```

Mint látjuk új parancs került bevezetésre nevezetesen a **Serial.read();.**

Ezt a parancsot használjuk a soros bemenet (RX) olvasására, karakterenként tesszük. Ha úgy tetszik byte-onként. Ezért bevezetésre került a **char** típusú változó, ami karaktereket tárol:

```
char valtozo = 'A';
char valtozo = 65; // mindkettő ugyanaz
```

Ha begépelünk több 1 és 0 a terminál küldő sorába majd utána kattintunk a küldés gombra, tapasztalni fogjuk, hogy a LED villog. A soros bement rendelkezik egy puffer memóriával, amelybe bekerülnek az adatok. Ennek a puffernek a mérete **64 byte** Arduino Uno esetében.

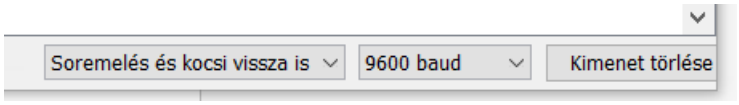
Ha valamilyen vezérlést szeretnék soros porton keresztül megvalósítani nem szeretnék minden lefutási ciklusban kiolvasni a küldött adatokat, csak ha van küldött adat a pufferben. Valamint általában egy adat több karakterből áll, ezért bevezetjük a **Serial.available** parancsot. A parancs segítségével megtudhatjuk, hogy van e beolvasásra váró karakter.

A következő program esetében a bekapcsol, és kikapcsol, szavakkal fogjuk a LED-et vezérelni. A program vezerles2.ino néven található meg a mellékletben.

```
int led = 3;  // LED 3 kimentre kötve
String szoveg; //szöveges változó
void setup() {
        pinMode(led, OUTPUT); //kimenet inicializálás
        Serial.begin(9600);  //soros komunikáció beállítás
}
void loop() {
        if (Serial.available()>0){ //soros puffer olvasása, ha nem üres
        szoveg = Serial.readStringUntil('\r'); //szöveg beolvasása sor emelés karakterig
         }
        if (szoveg=="bekapcsol") { //ha a beolvasott szöveg bekapcsol
        Serial.print(szoveg); //karakter kiíratása terminálra
         digitalWrite(led, HIGH); //led bekapcsolása
        }
         if (szoveg=="kikapcsol") { //ha a beolvasott szöveg kikapcsol
        Serial.println(szoveg); //karakter kiíratása terminálra
        digitalWrite(led, LOW); //led kikapcsolása
         }
        szoveg="";// változó ürítése
        delay(100);
}
```

Mint látjuk (**Serial.available()>0**) megvizsgáljuk, hogy jött adat a soros bemeneten, ha igen akkor **Serial.readStringUntil**() paranccsal beolvassuk egy szöveges változóba, ez a parancs addig olvassa a bementet, amíg nem talál olyan karaktert, ami a zárójelben meg van adva, esetünkbe a soremelés karakter (**\r**). Ez a karakter láthatatlan karakterek közé tartozik, és enter lenyomásakor küldi el a gép, függ a terminál beállításától. Másik sűrűn használt láthatatlan karakter (**\n**) új sor karakter.

A lenti képen látható, hogy terminál ablakban (Soros monitor) az ablak alján állítható be, hogy milyen karaktereket küldjön, enter vagy a Küld gomb lenyomása estén.



Bemenetek használata

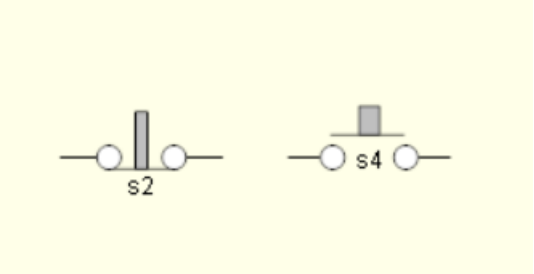
Az Arduino Uno kivezetései 0-13 valamint A0-A5 használhatóak bemenetnek, ezekre a bementekre különböző szenzorokat, kapcsolókat, gombokat köthetünk.

Digitális bemenetek, gomb

Digitális bemeneteknek használhatjuk a 0-13 kivezetéseket, ezekből a 0,1 kivezetést csak végszükség esetén mivel ez a modulunk soros portja. Ha a 0-1 kimenetre kötünk valamit, programfeltöltés ideje alatt érdemes lecsatlakoztatni, mivel megzavarhatja a folyamatot.

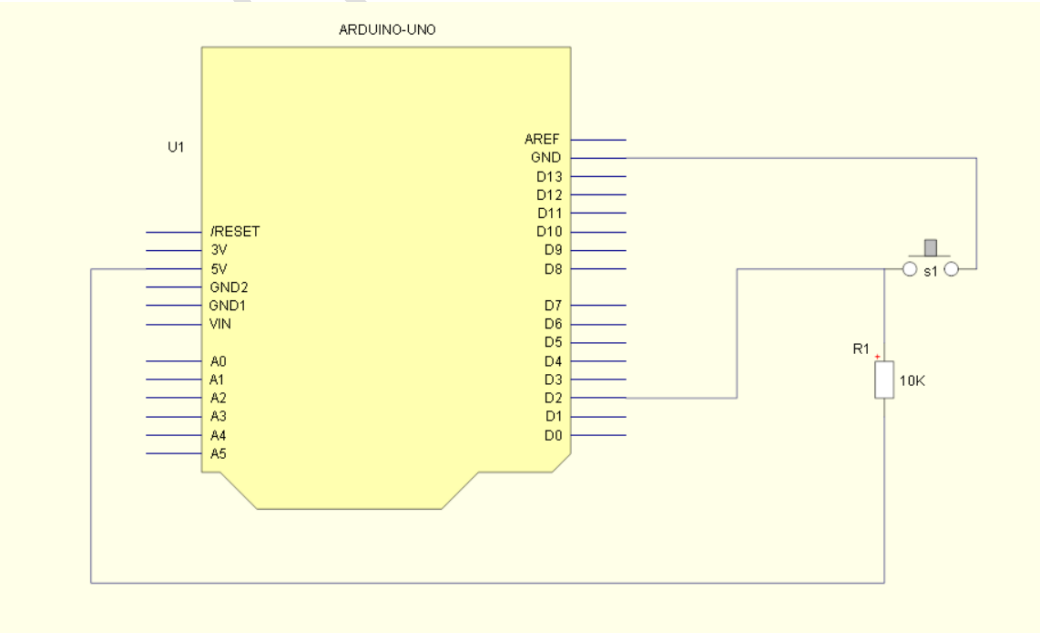
Készletünkben találgató nyomógomb segítségével fogjuk megtanulni használni a digitális bemeneteket. Szükségünk lesz még egy 10K ellenállásra is úgynevezett felhúzó ellenállásként fogjuk hasznosítani.

Gomb jele az elektronikai rajzokban:



Létezik belőle nyitó (nyugalmi állapotban zárt) s2 illetve záró (nyugalmi állapotban nyitott) s4.

Rakjuk össze a következő áramkört:



R1 felhúzó ellenállásra azért van szükség, hogy D2 kivezetés magas 5V értéken legyen alapból és S1 megnyomása esetén kerüljön alacsony 0V értékre.

Töltsük fel a követező programot az Arduinoba:

```
//konstansok olyan értékek, amelyek nem változnak
const int gomb = 2;     // nyomógomb bemenete
// változók
int gomb_allapot = 0;       // a gomb állapotának megállapítására szolgáló változó
void setup() {
          pinMode(gomb, INPUT); //bemenet inicializálása
         Serial.begin(9600); //soros kimenet inicializálása
}
void loop() {
          gomb_allapot = digitalRead(gomb); //gomb állapotának beolvadása
         // gomb állapotának összevetése, ha magas HIGH alacsony LOW
         if (gomb_allapot == HIGH) {
         // ha a gomb nincs, benyomva kiírjuk, KI
         Serial.println(„KI”);
         } else {
          // ha a gomb be van, nyomva kiírjuk, BE
         Serial.println(„BE”);
          }
  delay (200);
}
```

Feltöltés után kapcsoljuk be „Soros monitort” az Arduino IDE programban és rögtön látjuk, hogy folyamatosan KI szócskát írja ki. Ha benyomjuk a gombot, akkor a Be szócskára vált a szöveg a monitorban. Fenti programunkból látszik, hogy a **digitalRead(arduino kivezetés száma)** parancsot használtuk a bemenet kiolvasására ami lehet **HIGH** vagy **LOW** azaz 5V vagy 0V más értéket egy digitális bemenet nem vesz fel.

pinMode(gomb, INPUT); - segítségével incializáltuk a bementet

Helyettesítsük ezt az utasítást a következővel:

pinMode(gomb, INPUT_PULLUP); - ebben az esetben bekapcsoljuk a mikrokontrollerünkben található felhúzó ellenállást, tehát R1 ellenállásra nincs többé szükség az áramkörbe, így eltávolíthatjuk. Töltsük fel a programot, kapcsoljuk be a Soros monitort és próbáljuk ki a gomb kapcsolgatását.

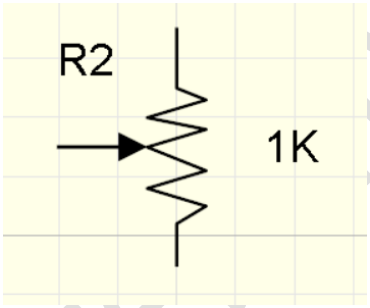
Analóg bemenetek, potenciométer

Ezúttal az A0-A5 felirattal ellátott bemenetekről lesz szó, ezek olyan speciális bemenetek melyekkel feszültség változást tudunk megfigyelni 0 és 5V között Arduino Uno esetében. Ezeknek a forrása, különböző szenzorok kimenete. De Volt vagy Amper mérésére is használhatjuk néhány ellenállással.

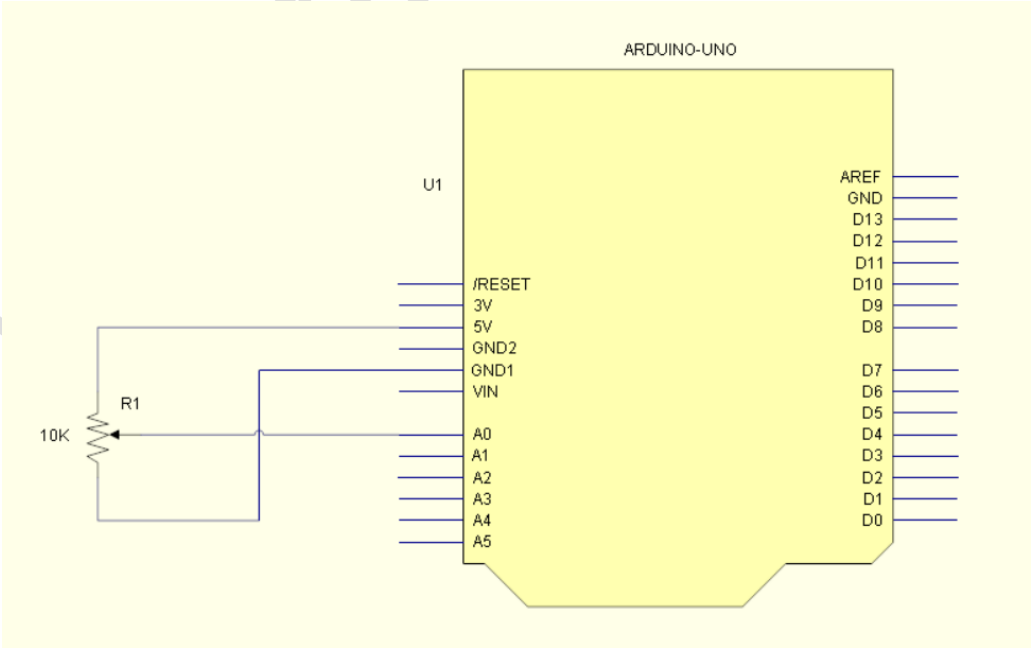
Vannak olyan Arduino modulok, amelyek 3,3V stabilizátorral rendelkeznek, ebben az esetben belső referencia esetén, 0 és 3,3V közötti feszültségeket tudunk megfigyelni az analóg bemenetekkel. Ebből adódik, hogy van olyan eset, hogy külső referencia feszültséget használunk, erre szolgál az AREF kivezetés a modulon, de mivel ez az iromány kezdőknek szól, ennek részletezésétől most eltekintünk.

Az analóg bemenet teszteléséhez szükségünk lesz egy 10K potenciométerre. A potenciométer nem más, mint egy változtatható értékű ellenállás.

Potenciométer elektronikai jele:



Ezúttal úgy kötjük az áramkörbe, hogy 0-5V között változzon a feszültség a potenciométer középső kimenetén, ha elfordítjuk a tengelyét. Rakjuk össze a lenti áramkört:



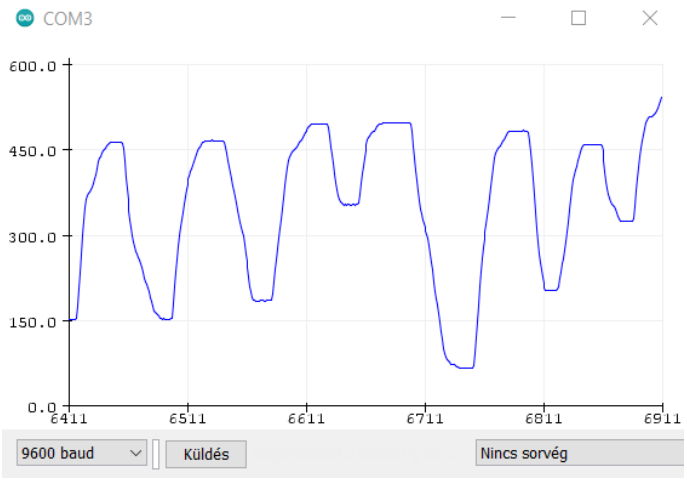
A következő egyszerű programot töltsük fel az Arduino Uno modulba, megegyezik a példák között található AnalogReadSerial.ino-val de ez magyarra van fordítva. Mellékletben megtalálható AnalogReadSerial_hu.ino néven.

```
void setup() {
        // soros kommunikáció inicializálása
        Serial.begin(9600);
}
void loop() {
        // beolvassuk az analog 0 értékét:
        int szenzorErtek = analogRead(A0);
        // kiíratjuk a soros kimenetre
        Serial.println(szenzorErtek);
        delay(1);       // késleltetés az olvasások között
}
```

Bekapcsoljuk a soros monitort és tapasztani fogjuk, hogy számok futnak felfelé az ablakban, ahogy tekergetjük, a potenciométert a számok változnak. Ha a potenciométer kimenete 0V –on van, akkor az ablak 0-át mutat, ha 5V a kimenet akkor 1023-at mutat.

Tehát a fent használt ***analogRead(analóg kimenet száma)*** parancs eredménye egy a feszültséggel arányos egész szám 0 és 1023 között. Ez tulajdonképpen a mikrokontrollerben található analog/digitál átalakító felbontását határozza meg. Minél nagyobb ez a szám annál pontosabb feszültségmérőt tudunk alkotni belőle. Minden analóg bementre kötött érzékelő kimeneti feszültségét szoktuk mérni és ebből számoljuk vissza az érzékelt tulajdonságot, legyen az hő, fény, stb. Irományom végén bemutatok néhány gyakorlati példát.

Fontosnak tartom még megemlíteni, hogy az Arduino IDE program még tartalmaz az Eszközök menüpontja alatt, egy Soros Plotter nevezetű eszközt, amellyel a fenti program változásait meg tudjuk jeleníteni grafikonként, vagy egyéb általunk írt program által mért értékek ciklusonkénti változását.



Gyakorlati példák

A fentiekben végig vettük az Arduino Uno használatlanának alapjait, távolról sem néztünk át töviről-hegyire mindent, de néhány alprogrammal ennyiből már boldogulni lehet. A további tudás felhalmozásához az Arduino-nak nagyon jó online leírása van, példákkal a lenti oldalon:

https://www.arduino.cc/reference/en/

Igaz, nem magyarul, de a Google fordító segíteni fog. ☺ Mi magyarok nem igazán jeleskedünk a programok fordításában tapasztalataim szerint, sokszor találkozok azzal, hogy amíg más kelet európai országok nyelvére le van fordítva egy program, magyarul nem tud. Jó lenne ezen változtatni. Magam is igyekszem hozzájárulni a transifex.com oldalon keresztül. Ezért nem jár fizetség.

A következő gyakorlati példákból kiderül, hogy a fentieket mire tudjuk használni a teljesség igénye nélkül.

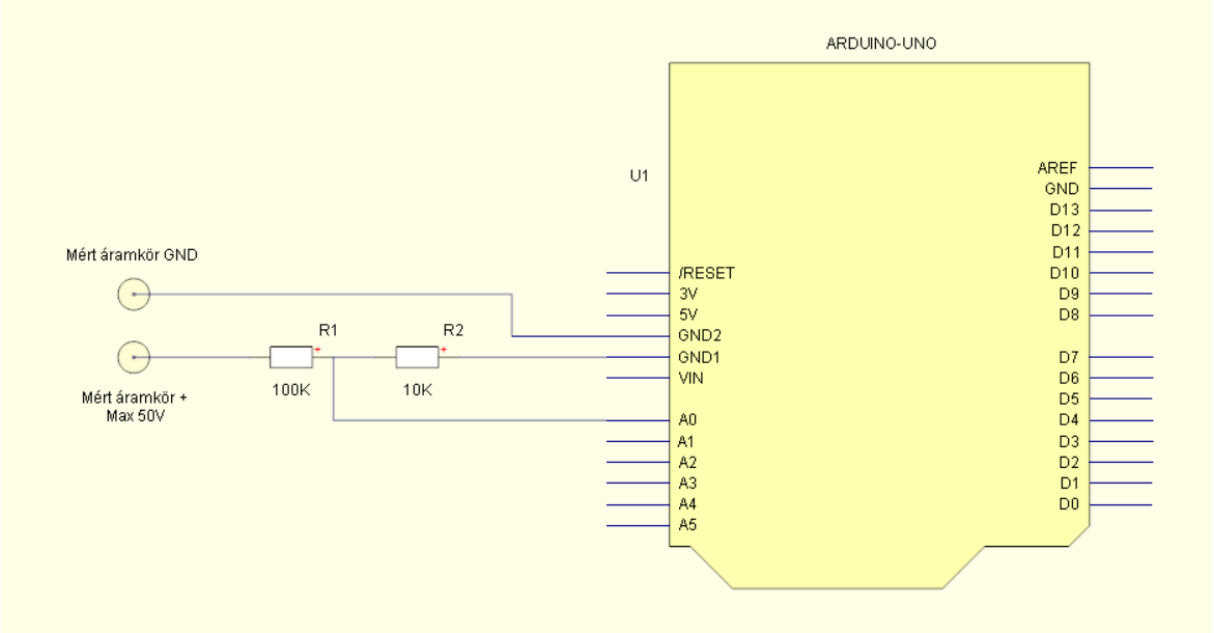
Feszültség mérő

Két ellenállás hozzáadásával készítünk egy feszültségmérőt, (voltmérőt). Jól jöhet saját készítésű barkácstápegységek készítése esetén. Mivel az Arduino Uno tartalmaz 6 db analóg – digitál átalakító bemenetet, ezért akár 6 bemenetünk is lehetne. Most csak egyet mutatok meg, de apró módosítás kell csak a többi használatához.

Le kell szegeznem, hogy ez egyenfeszültség mérésére alkalmas. Nem jó váltóáram mérésére. Méréshatár főleg R1 ellenállástól függ, neked kell kiszámolnod. Jelenlegi R1=100K, R2=10K osztóval maximum 50V-ig mérhetsz biztonságosan.

Konnektorba dugni tilos, életveszélyes az váltóáram és 230-240V.

Áramkör:



R1 és R2 ellenállások kiszámításához a terheletlen feszültségosztó képletét használjuk. A pontos mérőműszer készítéséhez ellenállások értékét mérjük meg, ne a rá jelölt értékekkel számoljunk.

Uki = 5V – a referencia feszültségünk, ami megegyezik a mikrokontroller tápfeszültségével mivel belső referenciát használunk.

Ube = mérni kívánt feszültség maximuma

R2 = 10Kohm

R1 = kiszámolható a lenti képlettel

$$U_{ki} = U_{be} \cdot \frac{R_2}{R_1 + R_2}$$

Program:

```
int be = A0; //0 analog bemenet
int dc = 0;
float voltbex;  // float lebegőpontos szám, egyszerűen tört szám végtelenbe tart
float voltki = 0.00;
float voltbe = 0.00;
float R1 = 97500.0; //  R1 ellenállás értéke ohm-ban
float R2 = 9700.0; // R2 ellenállás értéke ohm-ban
void setup() {
 // put your setup code here, to run once
Serial.begin(9600);
pinMode(be, INPUT);
}
void loop() {
 // put your main code here, to run repeatedly:
 for(int m=0;m<=40;m++){       //40 mintavétel
dc = analogRead(be); //ad átalakító leolvasása
   voltki = (dc * 5.0) / 1024.0; // 5.0 referencia feszültség
   voltbe = voltki / (R2/(R1+R2)); //kiszámolás (terheletlen feszültségosztó képlete átrendezve)
   voltbex=voltbex+voltbe;
   delay(5);
 }
 voltbex=voltbex/40;// átlag számítás
```

Serial.println(voltbex);//mért érték kiíratása soros portra

delay(200);

}

A mért feszültséget soros porton íratjuk ki.

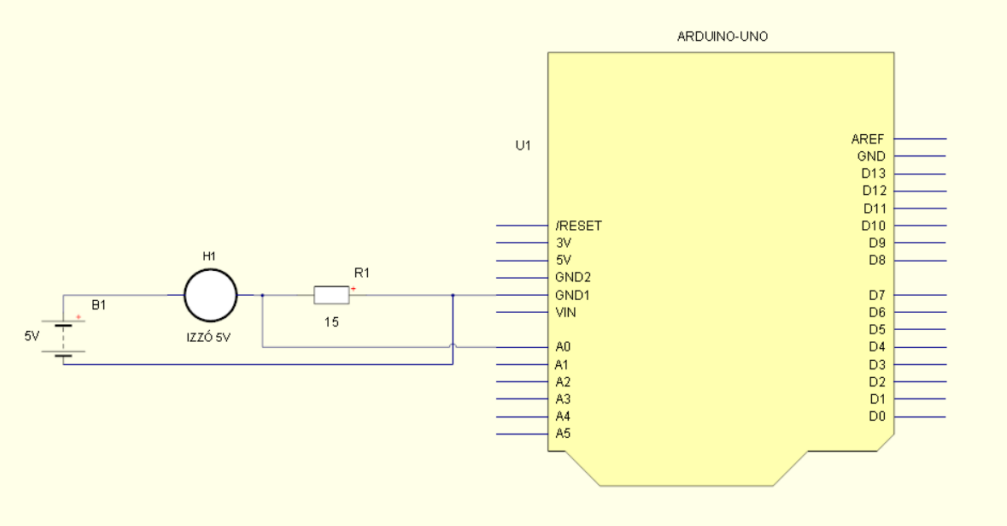
Mint látható bevezettünk egy float változót, amiben lebegőpontos számokat tárolhatunk (lehetnek pl.: -3.4028235E+38 -tól 3.4028235E+38 -ig).

Töltsük fel a programunkat az Arduino Uno modulba, kapcsoljuk be a soros monitort az Arduino IDE programban. Ragadjunk magunkhoz néhány elemet, nem baj, ha le vannak merülve és kezdjünk méregetni.

Ampermérő

Mivel az Arduino analóg-digitál átalakítója (A0-A5 láb) feszültséget tud mérni, ezért Ohm törvénye lesz a barátunk. Tehát a fogyasztó által felvett áramot ezzel tudjuk kiszámolni. I=U/R ahol I- a felvett áram, U – a mért feszültség az A/D átalakító által (rajzon U2), R – a sönt ellenállásunk (rajzon R1=15 ohm)

Áramkör:



Az ellenállás esetében minél nagyobb teljesítményűt és kis értékűt érdemes választani, főleg ha nagy áramokat szeretnénk mérni. (melegedés) Ajánlott kiszámolni a sönt értékét. Ha magasabb feszültségű áramkörön átfolyó áramot mérünk akkor, ahogyan a voltmérő esetében egy feszültségosztót kell az A/D átalakító elé helyezzünk úgy számolva, hogy a legmagasabb osztott feszültség 5V legyen. (ennyit bír ki az Arduino bemenete) Tehát U2 max 5V lehet, különben füst lesz. 😀

Ebben az esetben milliamperekről lesz szó, a terminálban is milliamperben fog megjelenni az eredmény.

Program:

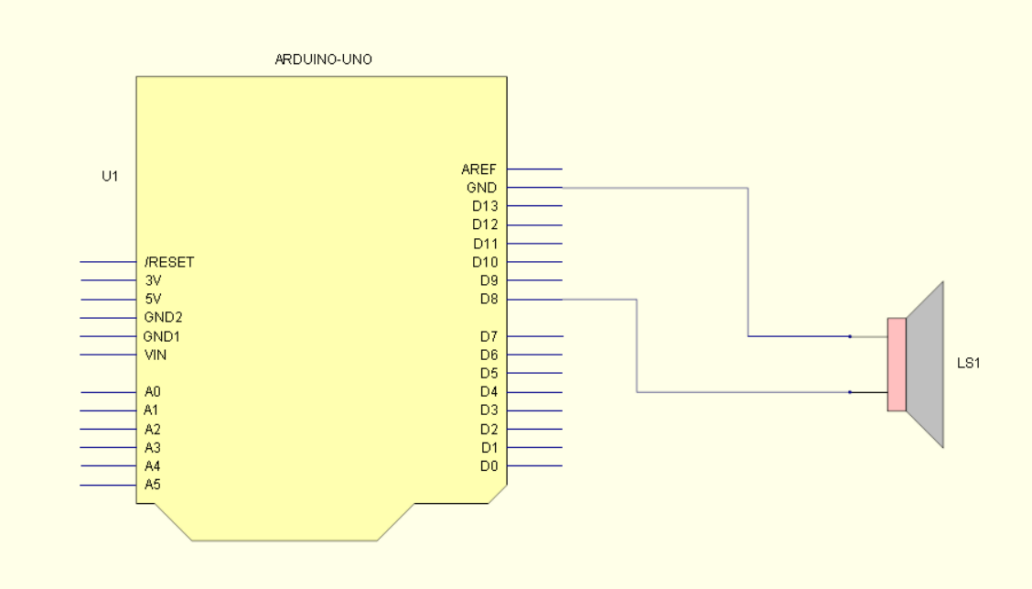
```
int be = A0; //0 analog bemenet
int dc = 0;
float voltbex;
float voltki = 0.00;
float aram = 0.00;
float R1 = 15; //  R1 ellenállás értéke ohm-ban


void setup() {
 // put your setup code here, to run once
Serial.begin(9600);
pinMode(be, INPUT);
}
void loop() {
 // put your main code here, to run repeatedly:
        for(int m=0;m<=40;m++){      //40 mintavétel
        dc = analogRead(be); //ad átalakító leolvasása
        voltki = (dc * 5.0) / 1024.0; // 5.0 referencia feszültség
        voltbex=voltbex+voltki;
        delay(5);
         }
 voltbex=voltbex/40;// átlag számítás
 aram=voltbex/R1; //Ohm törvénye
  aram = aram * 1000;//mA-é alakítás
  Serial.println(aram);//mért érték kiíratása soros portra
  voltbex=0;
  delay(200);
}
```

Feltöltjük a programunkat a modulba, majd elindítjuk a soros monitort (terminal) és le tudjuk olvasni a mért értéket. Ennél a példánál főleg a hogyanon van a hangsúly, a gyakorlati használathoz néhány védelmet be kell iktatni.

Dallam lejátszása

Készletünkben található egy apró hangszóró, ennek segítségével zenét is tudunk lejátszani az Arduinoval. Ehhez az Arduino IDE-ben található egyik példaprogramot fogjuk használni.

Áramkör:



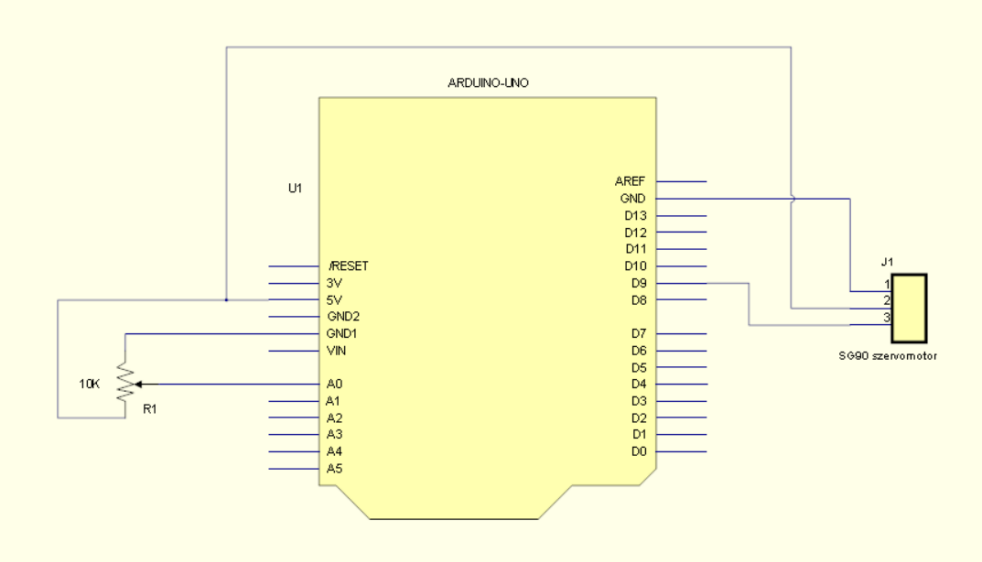
Töltsd be a Pédák -> Digital -> toneMelody programot a modulba, amely egy rövid dallamot játszik le. A Kód a VoidSetup() –ban található ezért nem ismétlődik.

Rögtön látható hogy a program 2 fájlból áll egy pitches.h fül jelent meg a szokásos fül mellett, valamint látható, hogy **#include** paranccsal van csatolva. Ezzel az utasítással tudunk bármilyen c –ben megírt fájlt csatolni a programunkhoz, különböző egyéb modulok vezérlése esetén szükséges, úgynevezett driverek (vezérlőprogramok) csatolása. Ezeket a fájlokat a Documentumok/Arduino/Libraries könyvtárba kel bemásolni könyvtárastól. Jelen esetünkben csak a hangokat tartalmazó fájlról van szó. Programot nem részletezem, ajánlom, játssz el a hangok sorrendjének módosításával.

Szervomotor vezérlése

Készletünkben található egy aprócska szervomotor, ennek a vezérlésével foglalkozunk ebben a részben. 10K - s potenciométerünkre is szükség lesz, ugyanis azt szeretném elérni, hogy amerre, fordítom a potenciométert arra és annyit forduljon a motor.

Áramkör:



Rakjuk össze a fenti áramkört, a szervomotorunk lábkiosztása következő:

1. GND - barna
2. 5V- piros
3. Jel – sárga

Program:

```
#include <Servo.h> //szervo vezérlő meghívása
Servo myservo;  // szervo objektum

void setup() {
    myservo.attach(9);  // csatlakoztatjuk a szervót a 9 lábhoz
}

void loop() {
    int pos = analogRead(A0); //Potenciométer pozíció kiolvasása
    pos = map(pos, 0, 1023, 0, 180); //érték átalakítása
    // szervónak fokokat kell megadni
    myservo.write(pos);          // szervo menjen az adott pozícióra
    delay(15);                   //késleltetés

}
```

A program elég egyszerű, 2 új dolog van benne:
map –parancs, ami egy érték átalakító parancs
map(változó, változó min. értékről, változó max. értékig, min. értékre, max. értékig)
myservo.write() –utasítás, amely egész számmal megadott fokra állítja a szervomotort

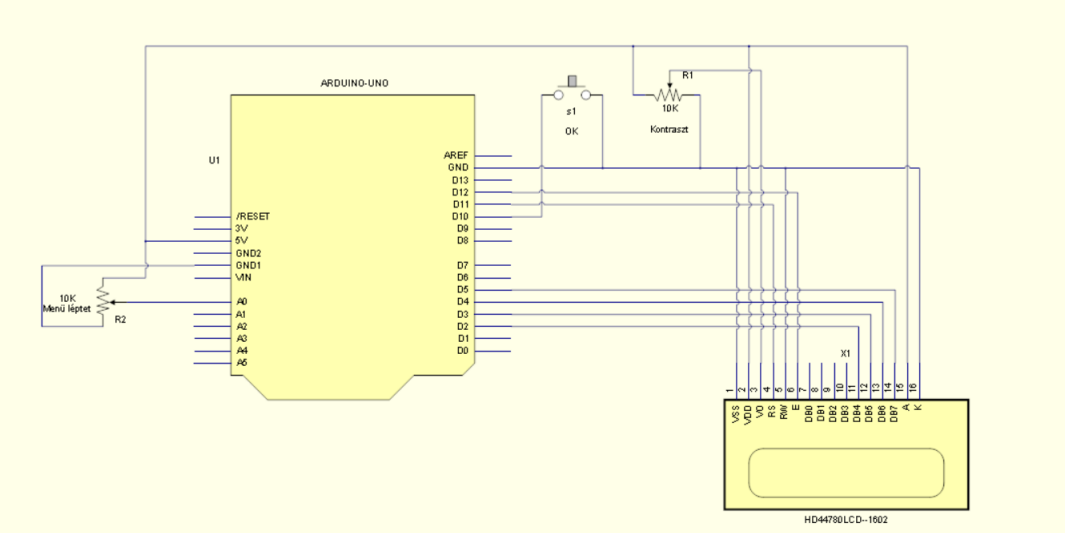
Láthatjuk, hogy ahogy csavargatjuk a potenciométerünk tengelyét, úgy követi a szervomotor állása.

LCD kijelző kezelése

A következő részben bemutatom az Arduino-ban eleve megtalálható LCD kijelző vezérlést. Nem egy bonyolult dolog, 3 számot fogunk beállítani egy LCD kijelző egy potenciométer és egy gomb segítségével. Három menüpontunk lesz, a menüket a potenciométerrel választjuk, ki a gombbal belépünk, potenciométerrel beállítjuk a kívánt étéket majd a gombbal kilépünk.

Rakjuk össze a következő áramkört, kicsit bonyolultnak tűnik, de ha követni kell csak a rajzot.

Áramkör:



Töltsük fel a következő programot az Arduino modulba. Lcd_menu.ino néven található a mellékletben.

Program:

```
#include <LiquidCrystal.h> //LCD kijelző vezérlő meghívása

LiquidCrystal lcd(11, 12, 2, 3, 4, 5); //LCD kijelző inicializálása (kivezetések beállítása melyik hova csatlakozik az arduinon)

int gombAllapot = 0;

int gombElozoAllapot = 0;

int menuRogzit = 0;

int gombKapcsol = 0;

int ertek1 = 0;

int ertek2 = 0;

int ertek3 = 0;

int pos;

void setup()

{
```

```
 pinMode(10, INPUT_PULLUP);
 lcd.begin(16,2);         // beállítás LCD 16x2
 //16x2 LCD esetén csak a következő sor kell
  // lcd.print("Meno,Mano,Menu");
 lcd.print("Meno,Man"); // üzenet kiírása LCD-re
 // Egy soros kijelző esetén kell DEM16102  16x1 LCD második 8 karakter kiírásához
 lcd.setCursor(40,0);
 lcd.print("o,Menu");
 delay(3000); // 3 másodperc késleltetés
}
void loop() {
 if(menuRogzit == 0){ //ha beléptünk a menübe pos ne változzon
 pos = analogRead(A0); //Potenciométer pozíció kiolvasása
 }
//potenciométert 3 menüre osztottuk mindegyik menüben be tudunk állítani egy változót
if(pos<=341 || menuRogzit == 1){  //Első szám beállítása
  gombAllapot = digitalRead(10); //gomb állapotának beolvasása
 if(gombAllapot != HIGH && gombElozoAllapot == HIGH){ //ha a gomb állapota magas és az előző
állapot magas volt (prellezés vagy pergés megakadályozása)
    if(gombKapcsol==0){
     gombKapcsol=1;
    }else{
     gombKapcsol=0;
    }
 }
 gombElozoAllapot = gombAllapot;
 if (gombKapcsol == 0) { //ha gomb nincs benyomva első szám értékének kiíratása a kijelzőre
    menuRogzit= 0; //menürögzítés feloldása
    lcd.clear(); //kijelző ürítése
    lcd.print("Nr1:"); //Nr1: kiíratása a kijelzőre
    lcd.setCursor(4,0); //foytatás a 4 karaktertől
```

```
    lcd.print(ertek1); //érték1 változó kírása
  } else {
    menuRogzit= 1; //menú rögzítése (poti elforgatására nem lép másik menüre)
     ertek1 = analogRead(A0); //potenciométer beolvasása
    lcd.clear();//kijelző törlése
    lcd.print(ertek1);//potenciométer állásának kírása 0-1023-ig
  }
}
 if(pos>341 && pos<=682 || menuRogzit == 2){ //Második szám beállítása
   gombAllapot = digitalRead(10);
 if(gombAllapot != HIGH && gombElozoAllapot == HIGH){
     if(gombKapcsol==0){
       gombKapcsol=1;
     }else{
       gombKapcsol=0;
     }
 }
 gombElozoAllapot = gombAllapot;
 if (gombKapcsol == 0) {
     menuRogzit= 0;
     lcd.clear();
     lcd.print("Nr2:");
     lcd.setCursor(4,0);
     lcd.print(ertek2);
 } else {
    menuRogzit= 2;
     ertek2 = analogRead(A0);
    lcd.clear();
    lcd.print(ertek2);
  }
}
```

```
if(pos>682 || menuRogzit == 3){ //Harmadik szám beállítása
  gombAllapot = digitalRead(10);
 if(gombAllapot != HIGH && gombElozoAllapot == HIGH){
    if(gombKapcsol==0){
     gombKapcsol=1;
    }else{
     gombKapcsol=0;
    }
 }
 gombElozoAllapot = gombAllapot;
 if (gombKapcsol == 0) {
    menuRogzit= 0;
    lcd.clear();
     lcd.print("Nr3:");
    lcd.setCursor(4,0);
    lcd.print(ertek3);
 } else {
   menuRogzit= 3;
    ertek3 = analogRead(A0);
   lcd.clear();
   lcd.print(ertek3);
 }
}
 delay(200); //késleltet 200 mS kijelző frissül minden lefutáskor
}
```

Csak az első menüt kommenteztem mivel lényegében megegyezik a másik kettővel.

Akinek 16x2 kijelzője van, az vegye ki a „//” jelet a következő sor előtt:

```
// lcd.print("Meno,Mano,Menu");
```

Valamint törölje ki a következő sorokat:

```
 lcd.print("Meno,Man"); // üzenet kiírása LCD-re
 // Egy soros kijelző esetén kell DEM16102  16x1 LCD második 8 karakter kiírásához
```

lcd.setCursor(40,0);

lcd.print("o,Menu");

LCD inicializálás, ami új a programban, és a gomb pergés gátlás, ez a második könnyen megérthető ha végig követjük a program logikáját.

LCD inicializálás:

LiquidCrystal lcd(rs, enable, d4, d5, d6, d7);

Követnünk kell a kijelző leírását (letölthető a gyártó weboldaláról) tartani kell a bekötési sorrendet.

LiquidCrystal lcd(11, 12, 2, 3, 4, 5);

Az Arduino digitális kivezetések számát adjuk meg, amelyeket használni szeretnénk.

lcd.begin(16,2);

16 karakteres, 2 soros kijelző megadása. (1 sorosat furcsán kezeli de az volt fiókban)

lcd.print("Meno,Mano,Menu");

Kiírja az LCD –re a megadott adatot.

lcd.setCursor(4,0);

Kurzor léptetése, első sor (azaz 0-dik sor) 4 karakterre.

Mindenkinek jó fejlesztést, szórakozást!