Programozási ismeretek

Agócs Norbert és Nagy Dániel

2020. szeptember 8.

Tartalom

Tartalom

Mit tanultunk eddig?

Hardware:

- Arduino alaplap
- Breadboard
- LED
- Ellenállás
- Gomb
- Ultrahang szenzor
- Szervomotor
- Joystick

Software

- Változó létrehozása
- Különféle változótípusok + specifierek
- Alapműveletek (=, +, -, /, *)
- Alapvető parancsok
- Elágazás: `If()...else`
- Iteráció: `For()`
- Szervomotor irányítása
- Függvényírás ← Mai szakkör
- Tömbök ← Mai szakkör

Tartalom

Ismétlő teszt: Hány állítás igaz?

Az állítások:

- Előre nem lehet megmondani hogy a `for` ciklus hányszor fut le.
- Ha egy változó elé azt írjuk hogy `const` akkor annak a változónak az értéke bármikor megváltoztatható.
- A `pinMode()`, `digitalWrite()`, `analogRead()` kifejezések függvények
- Az `if` állítás eldönti egy állításról, hogy igaz-e és ez alapján végrehajt vagy kihagy egy kódrészletet.
- A `Serial.print("Hello world")` függvényben a `"Hello world"` a függvény argumentuma
- Egy függvénynek bármennyi argumentuma lehet

Ismétlő teszt: Megoldások

A javított állítások:

- X Előre meg lehet megmondani hogy a lstinlinefor ciklus hányszor fut le.
- X Ha egy változó elé azt írjuk hogy `const` akkor annak a változónak az értéke sosem változtatható meg.
- ✓ A `pinMode()`, `digitalWrite()`, `analogRead()` kifejezések függvények
- ✓ Az `if` állítás eldönti egy állításról, hogy igaz-e és ez alapján végrehajt vagy kihagy egy kódrészletet.
- ✓ A `Serial.print("Hello world")` függvényben a `"Hello world"` a függvény argumentuma.
- ✓ Egy függvénynek bármennyi argumentuma lehet, amennyiben ezt a függvényt mi írtuk.

Mit ír ki a kód?

```
const int Pi = 3.1415926;
int r = 0;

void setup() {
    Serial.begin(9600);
    r = 10;
    Pi = 3; //Mérnökök leszünk, szeretünk közelíteni
    float T = r*r*Pi;
    Serial.print("A kör területe:");
    Serial.println(T);
}

void loop() {
}
```

Mit ír ki a kód?

```
const int Pi = 3.1415926;
int r = 0;

void setup() {
    Serial.begin(9600);
    r = 10;
    Pi = 3; //Mérnökök leszünk, szeretünk közelíteni
    float T = r*r*Pi;
    Serial.print("A kör területe:");
    Serial.println(T);
}

void loop() {
}
```

Megoldás: A kód nem fut le, hibát dob, mivel egy konstans változót szeretnék átírni, amit nem tehetek meg!

Tartalom

Függvények

Amit már tanultunk

- `void pinMode(int pin, int allapot);`
- `void digitalWrite(int pin, int allapot);`
- `int analogRead(int pin);`
- `int digitalRead(int pin);`
- `void delay(int varakozas);`
- `unsigned long millis();` (még nem tanultuk de ilyen is van)

Amit érdemes megfigyelni

1. Tartalmaznak zárójelet
2. Áll előttük egy típus
3. Az Arduino IDE a nevüket narancssárgával írja
4. Az argumentumok száma különböző (0,1,2 de lehetne több is)

Függvények

Hogy használtuk őket

```
void pinMode(int pin, int allapot);
```

```
const int LED_PIN = 2;
pinMode(LED_PIN,OUTPUT);
```

```
int digitalRead(int pin);
```

```
int olvasottErtek;
olvasottErtek = digitalRead(3);
```

```
void delay(int varakozas);
```

```
delay(5000);
```

1. A `void` típusú függvényeknél nincs értékadás
2. Az `int` típusú függvénynél, a függvény ad értéket egy változónak
3. Az argumentum lehet egy szám, `const int` vagy `int` típus is.
4. Annyi argumentumot adtunk meg amennyi a definícióban szerepel

Függvények

Mire jók?

Előnyök

- Ha egy kódrészlet többször előkerül a kódban akkor érdemes belőle függvényt készíteni
- Manapság ha egy kódrészlet 2x előkerül már függvényt írnak belőle, Arduinon ez nem feltétlen szükséges
- Átláthatóvá és rövidebbé teszi a kódot

Példák

- Egy kód során, többször is szeretnénk kiválasztani két szám közül a nagyobbat, ezért elkészíthetünk egy maximum függvényt, ami ezt megcsinálja nekünk.
- Egy kód méréseket végéz, és többször is szükséges a mérési eredmények átlagának a kiszámítása, erre írhatunk egy átlag függvényt.

Hogy írjunk függvényt - Fejléc

```
függvénytípus név(típus1 arg1, típus2 arg2...){
    kód;
    kód;
    return érték;
}
```

függvénytípus

Lehet `int`, `float`, `void` stb... ez a visszatérési érték típusa.

név

Ez a függvényünk neve, betűvel kell kezdődnie és nem lehet benne space. Pl: `delay`, `osszeadas`, `atlag`

(típus1 arg1, típus2 arg2...)

`típus1`, `típus2` a belső változók típusai és `arg1`, `arg2` a belső változók nevei, pl `(int pin, float ido)` vagy `(int pin1, int pin2, int pin3)`

Hogy írjunk függvényt - Kifejtés

```
függvénytípus név(típus1 arg1, típus2 arg2...){
    kód;
    kód;
    return érték;
}
```

Kapcsos zárójelek

Egy { jelenti a kifejtés kezdetét, és egy } a függvény végét

kód

Bármi, `for`, `if`, `else`, értékadások, matek műveletek, függvényhívások

return érték;

A függvény visszatérése, ha a kód a return szócskához ér, akkor a függvénynek vége, és visszaadja a mögötte álló értéket.

Példa I

2 szám összeadása függvénnyel

```
int osszead(int a, int b){
    return a + b;
}
```

Összeadás függvény használata:

```
int a = -5; int b = 8; int c;
c = osszead(a,b); // c = 3
int d = osszead(2,-3); // d = -1
```

Teljesen egyenértékű kód:

```
int a = -5; int b = 8; int c;
c = a + b; // c = 3
int d = 2 + (-3); // d = -1
```

Példa II

Másodfokú függvény/Parabola

A matekból jól ismert $f(x) = x^2$ függvényt is megírhatjuk!

```
float f(float x)
{
    return x*x;
}
```

Néhány érték kiszámítása:

```
float y1 = f(-2); //y1 = 4
float y2 = f(-1); //y2 = 1
float y3 = f(-0.5); //y3 = 0.25
float y4 = f(0); //y4 = 0
float y5 = f(10); //y5 = 100
```

Példa III

Előjel függvény

Ha egy szám kisebb mint 0, akkor -1 legyen a visszadott érték, ha egy szám nagyobb mint 0 akkor a visszadott érték legyen 1

```
int elojel(float x)
{
    if(x < 0)
    {
        return -1;
    }
    else
    {
        return 1;
    }
}
```

Néhány példa kiszámítása:

```
float e1 = elojel(-2); //e1 = -1
float e2 = elojel(1000.1); //e2 = 1
float e3 = elojel(-0.00001); //e3 = 1
```

A `void` típus

- Gyakorlatilag azt jelenti hogy üres
- `void` típusból változókat nincs értelme létrehozni, hiszen nem tud értéket tárolni.
- Függvényeknél hasznos, hiszen minden függvénynek kötelező típust adni. Amennyiben a függvényünknek nincs visszatérési értéke, akkor használjuk a `void` típust.
- Ha a függvény típusa `void` akkor a függvénybe nem kell `return`-t írni. A `return;` továbbra is megadható de nem állhat mögötte semmi

Példa IV

LED-ek vezérlése függvénnyek

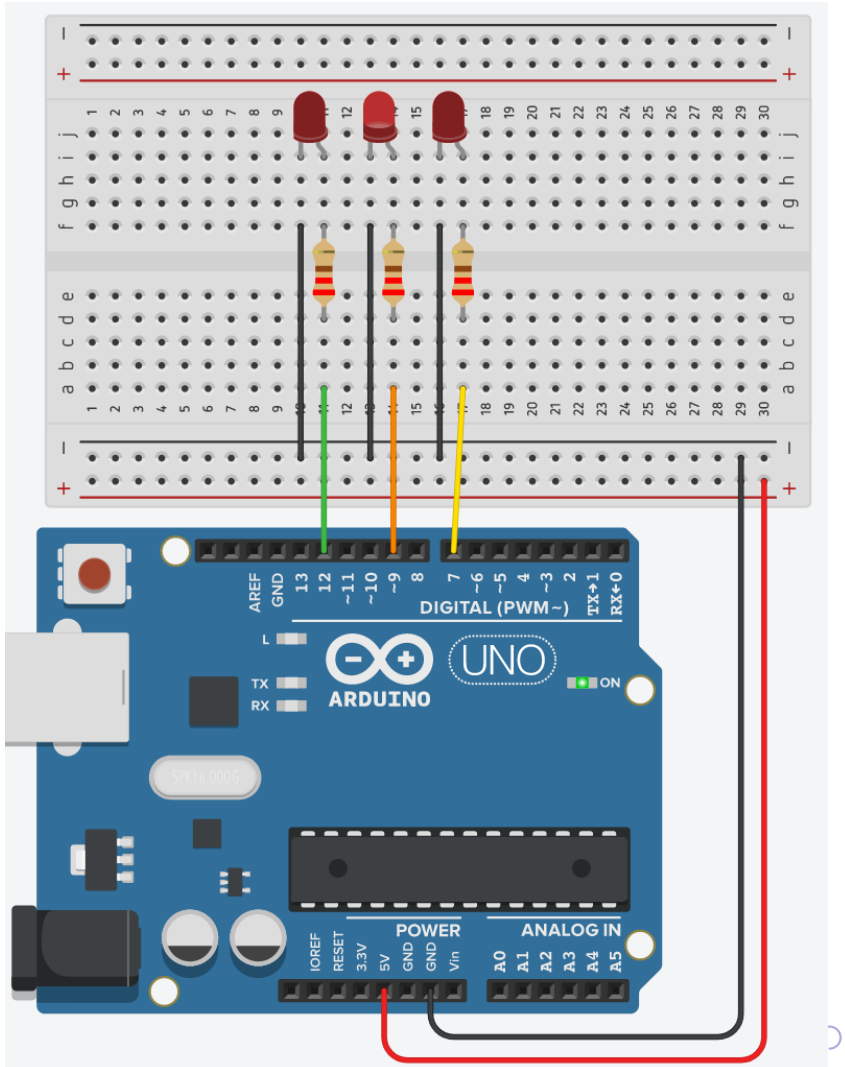
Írjunk egy függvényt ami kezeli a LED-eket, és lehetővé teszi bármely LED-ek ki és be kapcsolását

Mi kell hozzá?

- 3db LED
- 3db Ellenállás (220-500 Ω)

Hogyan fogjuk megírni a kódot?

- Megírjuk a `void ledAllapot()` függvényt
- A függvény paraméterei a LED-ek állapotai lesznek



Példa IV

```
const int LED1PIN = 12;
const int LED2PIN = 9;
const int LED3PIN = 7;

void ledAllapot(int allapot1, int allapot2, int allapot3){
   digitalWrite(LED1PIN,allapot1);
   digitalWrite(LED2PIN,allapot2);
   digitalWrite(LED3PIN,allapot3);
}
void setup(){
   pinMode(LED1PIN,OUTPUT); //ezeket továbbra is meg kell adni
   pinMode(LED2PIN,OUTPUT);
   pinMode(LED3PIN,OUTPUT);
}
void loop(){
   ledAllapot(HIGH,HIGH,HIGH); //mindhárom LED-et felkapcsolja
   //a HIGH érték bekerül allapot1, allapot2, allapot3 helyére
   delay(2000);
   ledAllapot(LOW,HIGH,LOW); //csak a középso világít
   delay(2000);
}
```

Példa IV

Az általunk készített függvény

```
void ledAllapot(int allapot1, int allapot2, int allapot3)
{
   digitalWrite(LED1PIN,allapot1);
   digitalWrite(LED2PIN,allapot2);
   digitalWrite(LED3PIN,allapot3);
}
```

- A saját függvényeket a `void setup()` elé írjuk mindig!
- 3 argumentum, ami a 3 LED állapotával azonosítható
- Az argumentumok típusa `int`, ahova írhatunk `HIGH` vagy `LOW` értéket is
- A függvény típusa `void`, nem tér vissza semmilyen értékkel, ezért `return` sincsen benne
- A függvény kifejtése 3 utasítás, amelyek a megfelelő LED-eket kapcsolják a `digitalWrite()` függvénnyel

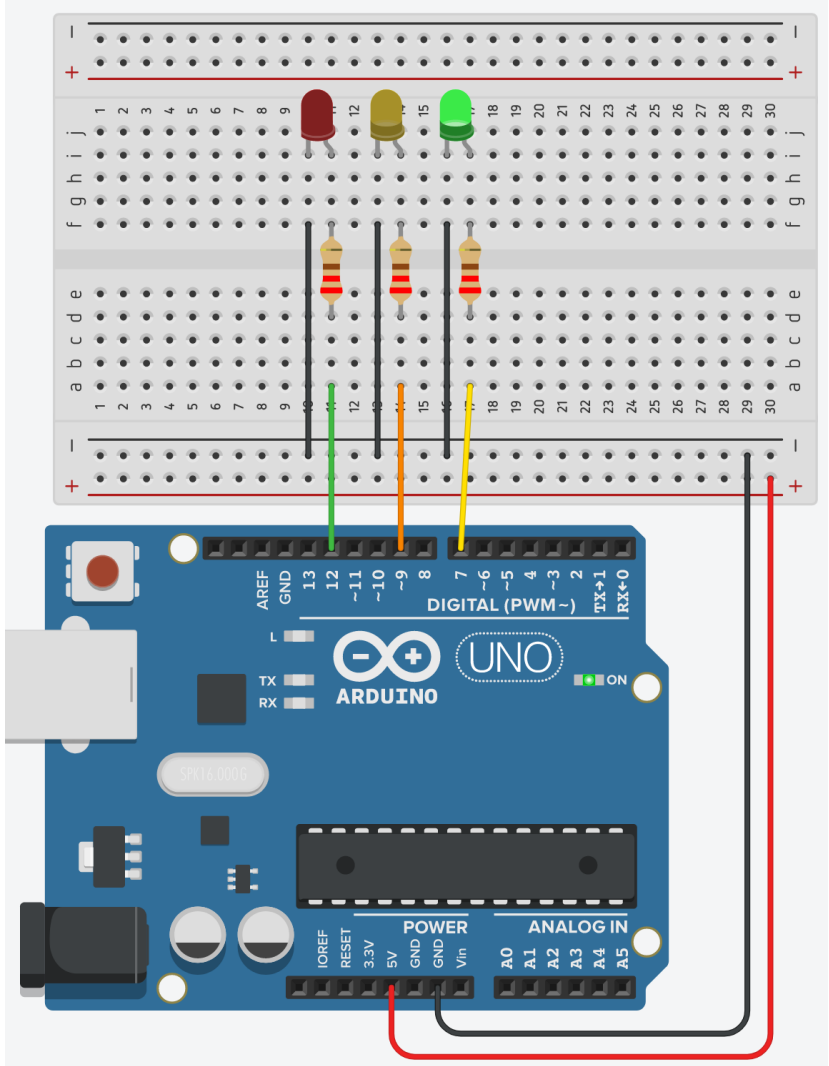
Jelzőlámpa függvénnyel

Írjunk egy függvényt ami kezeli a jelzőlámpa LED-eit. Működtessük vele a jelzőlámpát. Mi kell hozzá?

- 3db LED (zöld, sárga, piros)
- 3db Ellenállás (220-500 Ω)

Hogyan fogjuk megírni a kódot?

- Megírjuk a
  `void jelzolampaAllapot()`
  függvényt
- A függvény paraméterei a LED-ek állapotai lesznek, és a várakozási idő a következő kapcsolásig



Jelzőlámpa függvénnyel

```
const int PIROS_LED = 12;
const int SARGA_LED = 9;
const int ZOLD_LED = 7;
void jelzolampaAllapot(int zold, int sarga, int piros,int var){
  digitalWrite(ZOLD_LED,zold);
  digitalWrite(SARGA_LED,sarga);
  digitalWrite(PIROS_LED,piros);
  delay(var);
}
void setup() {
   pinMode(PIROS_LED,OUTPUT);
   pinMode(SARGA_LED,OUTPUT);
   pinMode(ZOLD_LED,OUTPUT);
}
void loop() {
  jelzolampaAllapot(1,0,0,1000); //zöldre vált és 1mp-et vár
  jelzolampaAllapot(0,1,0,500); //sárgára vált és 0.5mp-et
  jelzolampaAllapot(0,0,1,2000); //pirosra vált és 2mp-et vár
  jelzolampaAllapot(0,1,1,500); //piros-sárgára vált
}
```

Tömbök

Mik azok a tömbök?

Definíció

Olyan adatszerkezet, amely azonos típusú változókból áll, amikre sorszámukkal (index) lehet hivatkozni.

Értelmezzük!

- Valamilyen cucc, amiben tudunk dolgokat tárolni.
- Ebben a cuccban csak azonos típusú dolgokat tudunk benne tárolni, például csak `int` vagy csak `float`.
- A cuccban lévő dolgokat valamilyen számozással el tudjuk érni.
  - Legyenek ebben a cuccban most pl. számok: {0,2,4,6,8,10}
  - Ebben az esetben tudom, hogy a 3. eleme a cuccnak 4.

Most nézzük meg programozási szempontból a tömböket!

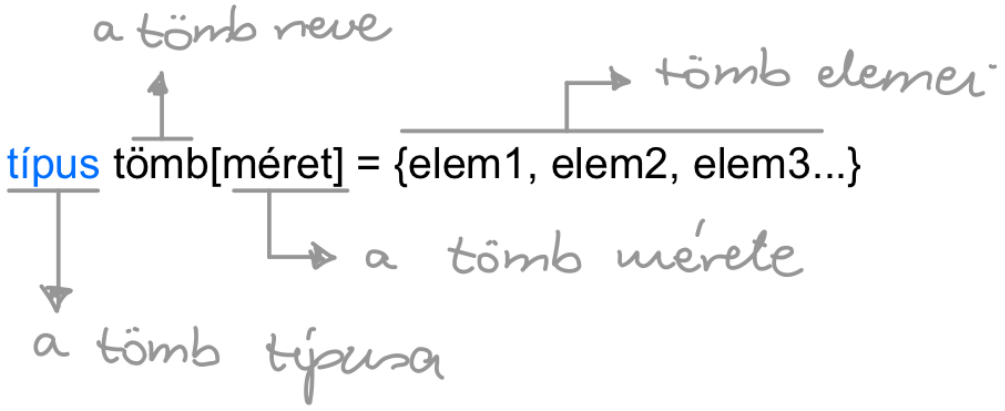
A tömbök tulajdonságai I.

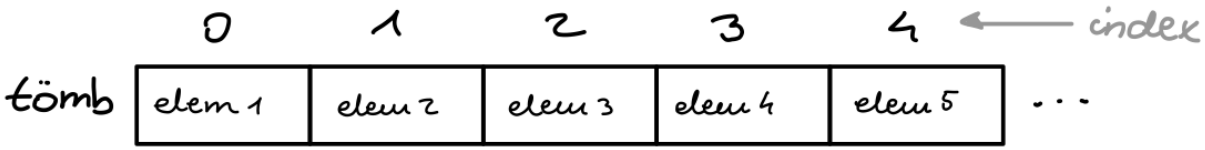
A tömb: A tömb egy adattároló, amiben azonos típusú adatokat tudunk tárolni és indexeléssel érni. Fontos tulajdonságai:

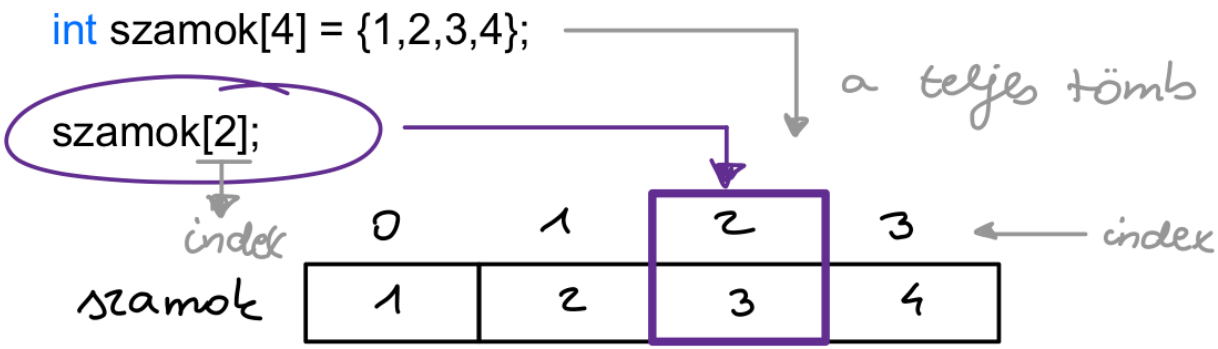
- A tömb mérete
  - A tömb méretét a tömb elemeinek a száma adja meg.
  - A tömbök kezeléséhez muszáj ismernünk ezt, ezért érdemes a tömb méretét egy külön változóba elmenteni.
- A tömb típusa
  - A tömb típusa adja meg, hogy milyen típusú elemeket tárolhatunk.
  - Példa: Egy `int` típusú tömbben csak `int`-ek lehetnek.
- A tömb elemeinek indexei
  - Egy tömbben minden elemnek van egy indexe.
  - Az indexet szögletes zárojelek közé kell tenni.
  - Fontos: Az indexelés 0-tól kezdődik, tehát az 1. elem indexe 0, a 2. elem indexe 1, a 3. elem indexe 2 és így tovább...

A tömbök tulajdonságai II.

Szemléletes ábra a megértéshez:







A tömbök programozása I.

Létrehozás

```
/* A legjobb létrehozási mód: */
// 1. A tömbök elemszámát mentsük el külön.
// 2. Ezzel hozzuk létre a tömböt.

const int meret = 5;
int szamok[meret] = {1,2,3,4,5};

/*Egyéb létrehozási módok: */

//Ha nem adok meg méretet, akkor a program kitalálja
//Ez így nem túl jó, mert később szükség lenne a méretre.

float tortszamok[] = {1.5,2.5,3.5};

//Ha nem használom ki a megadott méretet, akkor a maradék
    helyet 0-val tölti fel a program.

String szavak[meret] = {"egy","ketto","harom"};
```

A tömbök programozása II.

Elemek elérése egyesével

```
//A tömböm:
const int meret = 4;
String szavak[meret] = {"alma", "korte", "szilva", "banan"};

//Értékadás egyesével:
szavak[2] = "tehén";

void setup() {
   Serial.begin(9600);
   Serial.println(szavak[0]); //Amit kiír: "alma"
   Serial.println(szavak[1]); //Amit kiír: "korte"
   Serial.println(szavak[2]); //Amit kiír: "tehén"
   Serial.println(szavak[3]); //Amit kiír: "banan"

   //Serial.println(szavak[4]); -> Hiba!
   //Az indexelés 0-tól kezddik, tehát csak 0,1,2,3
       indexeink léteznek!
}
```

A tömbök programozása III.

Elemek elérése ciklussal

```
//A tömböm:
const int meret = 5;
int szamok[meret] = {1,2,3,4,5};

void setup() {
   Serial.begin(9600);

   for(int i = 0; i<meret; i++) {
      Serial.println(szamok[i]);
   }

   /*Magyarázat:*/
   // 1. Az i változó 0-tól egészen a meret értékéig megy.
   // (Ezért kell tudnunk a méretet!!!)
   // 2. Minden lépésben, amikor lefut a ciklus, a szamok[i]
        segítségével megkapjuk az adott i. helyen lév elemet
   // 3. A Serial.println függvény mindig kiírja nekünk az
       aktuális értéket.
}
```

Feladat: LEDek vezérlése tömbbel

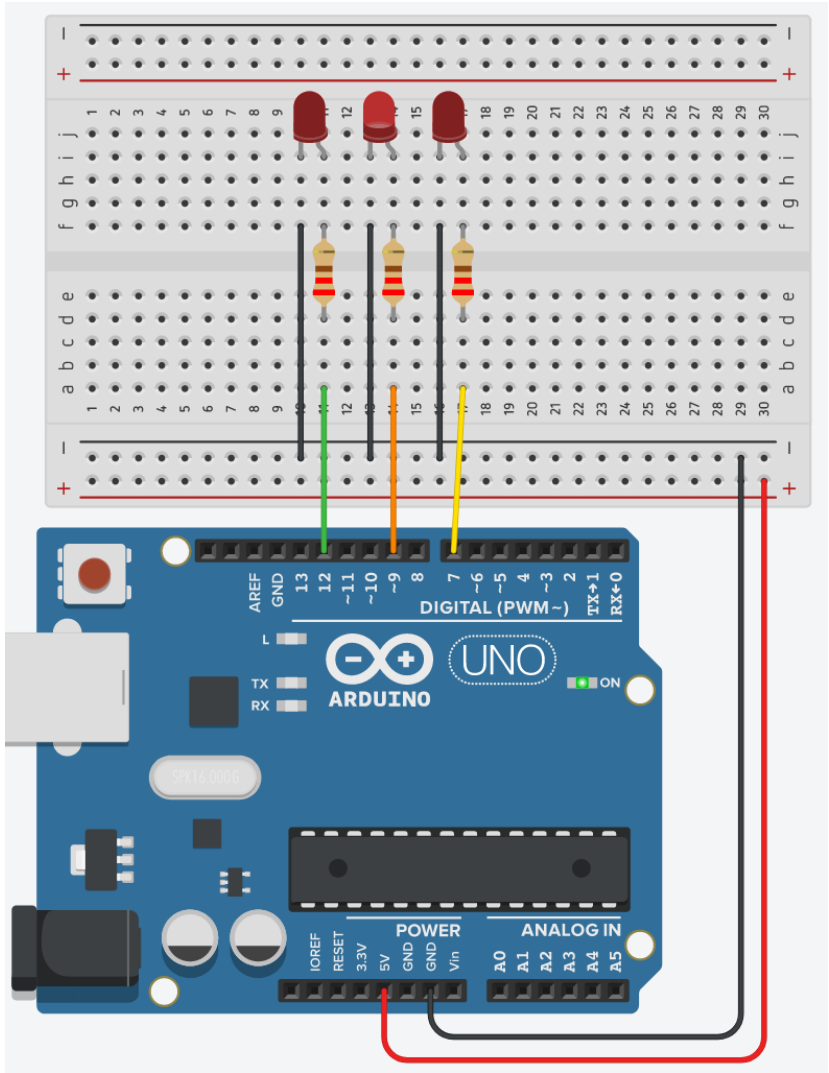
Írjunk olyan programot, ami egymás után elhelyezett LED-eket felvillant, majd lekapcsol.

Mi kell hozzá?

- 3db LED
- 3db Ellenállás (220-500 Ω)

Hogyan fogjuk megírni a kódot?

- Eltároljuk egy tömbben a LEDek pinjeit.
- Egy `for` ciklus segítségével villogatjuk a ledeket.



Feladat: A kód

Mit csináltunk volna eddig?

- Külön-külön definiáltuk volna a LED-ek pinjeit.

```
const int led1 = 12;
const int led2 = 9;
const int led3 = 7;
```

- Külön-külön megmondtuk volna, hogy ezek kimeneti eszközök

```
void setup {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
}
```

- És végül külön-külön irányítottuk volna őket.

```
void loop {
  Villogjatok!...Légyszi :(
}
```

Feladat: A kód

Mit csinálunk most?

```
const int ledek_szama = 3; //Ennyi elemet tárolok a tömbben
const int ledek[] = {12, 9, 7}; //A tömbben a LED-ek pinjei

//Mindegyik LED-et kimeneti eszközre állítom
void setup() {
  for (int i = 0; i<ledek_szama; i++) {
    pinMode(ledek[i], OUTPUT);
  }
}

//Mindegyik LED-et felvillantom, majd lekapcsolom
void loop() {
  for(int i = 0; i<ledek_szama; i++) {
    digitalWrite(ledek[i], HIGH);
    delay(500);
    digitalWrite(ledek[i], LOW);
    delay(100);
  }
}
```

Tartalom

Házi feladatok beküldése

- A házi feladatok elkészítése ajánlott ahhoz, hogy teljesen megértésétek a programozás ezen részét
- A feladatok egy része nem könnyű, annyi időt szánjatok rá, amennyit szeretnétek, nem kötelező az összeset megcsinálni
- Küldjétek el nekünk a feladatokhoz az Arduino kódokat amiket írtatok, ezek `.ino` kiterjesztésű fájlok. Azok aki vittek haza Arduinot, tőletek elvárjuk hogy legalább 1-2 feladatot megoldotok.
- Aki nem vitt haza eszközt de érdeklődik a téma iránt az TinkerCAD-ben tud Arduino kódot írni. `https://www.tinkercad.com/` regisztáció után keressétek a `Circuits` lehetőséget, itt egy projektet létrehozva, látni fogtok egy `Code` menüpontot.
- Email címünk: `n.nagyd@gmail.com` (Dani) és `nagocs@me.com` (Norbi)
- Ha elakadtatok elsősorban a csoportba írjatok

Feladatok a függvényekhez (Könnyebbek)

1. Írjatok egy függvényt, ami 3 számot összeszoroz, írjátok meg `float` és `int` típusra is majd próbáljátok ki. (Így nézzen ki: `int szoroz(int a, int b, int c)`)
2. Írjatok egy függvényt ami kiszámolja 4 szám átlagát. A használandó képlet: átlag $= \frac{a+b+c+d}{4}$ ahol a, b, c, d típusa `float`. Számoljátok ki -1.5, 2.5, 2.11, -11 átlagát
3. Írjatok egy függvényt, ami eldönti egy emberről az életkora alapján hogy fiatal, középkorú vagy öreg. Egy ember akkor fiatal ha az életkora 35 év alatt van, 60 év fölött öreg, a kettő között középkorú. Használjátok a `Serial.begin()` és `Serial.println()` függvényt a kiiratáshoz.

Feladatok a tömbökhöz (Nehezek)

1. Egy tömb segítségével számítástok ki az első 10 szám négyzetének összegét, majd a megoldást írássátok ki.
2. Számoljátok ki a {97, 87, 100, 99, 86, 91, 92, 85, 88, 100} tömb elemeinek az átlagát, majd a megoldást írrasátok ki.
3. Hozzatok létre egy 5 elemű tömböt, ami szavakat tartalmaz. Írassátok ki a szavakat fordított sorrendben.

Tartalom

Szavak

e Funktion /-en	függvény
s Feld (s Array)	tömb
r eckige/geschweifte Klammer /-	szögletes/kapcsos zárójel
e Wertzuweisung	értékadás
einen Wert zuweisen +D	értéket adni vminek
s Argument /-e	argumentum
e Ampel /-n	jelzőlámpa
r Index / e Indize	index
e Größe /-n	méret
e Anzahl /-e	mennyiség, szám