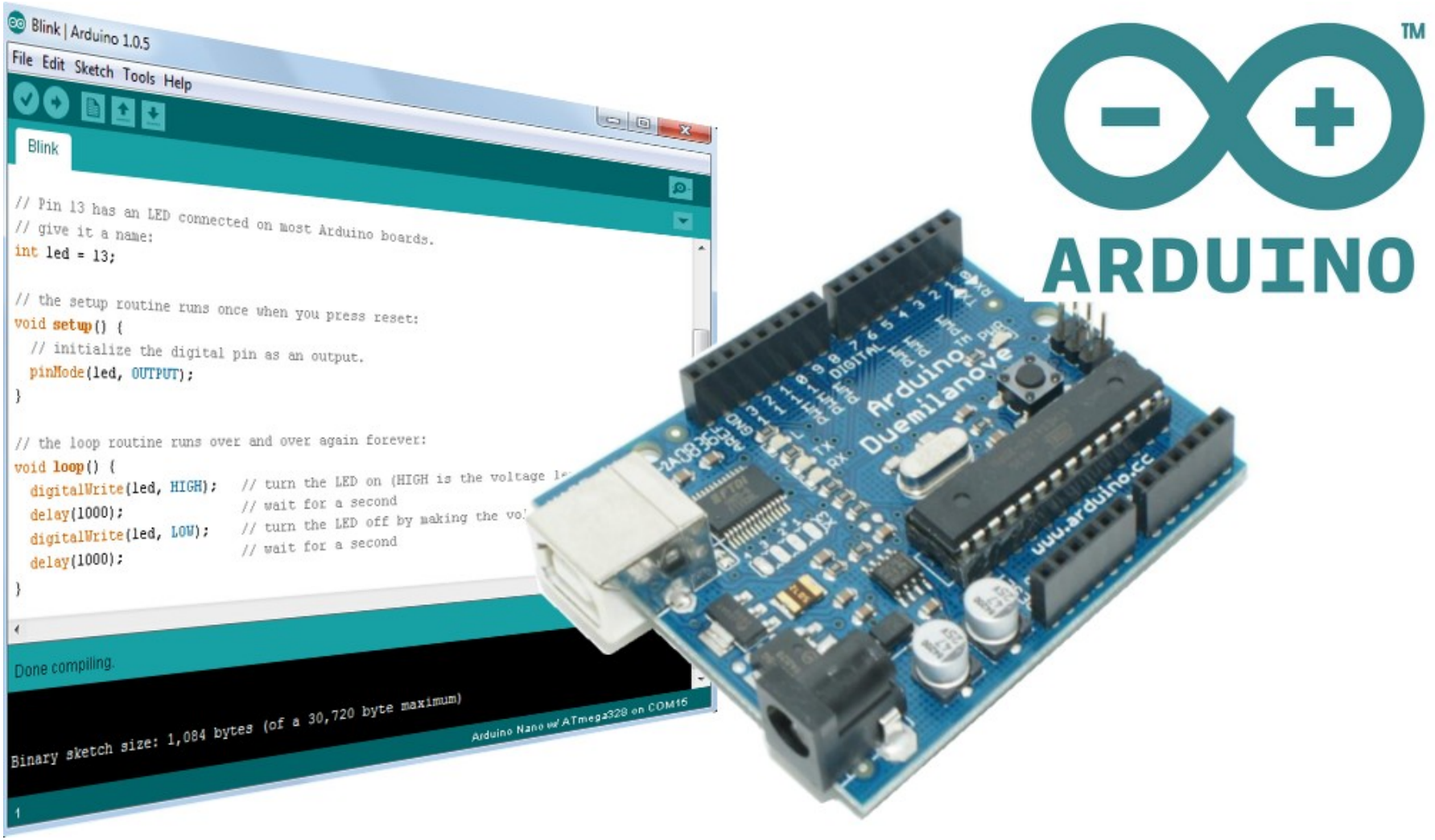
Bevezetés az elektronikába



11. Arduino programozás – vezérlési szerkezetek, programciklusok szervezése

Vezérlési szerkezetek

A műveletek végrehajtási sorrendjét határozzák meg.

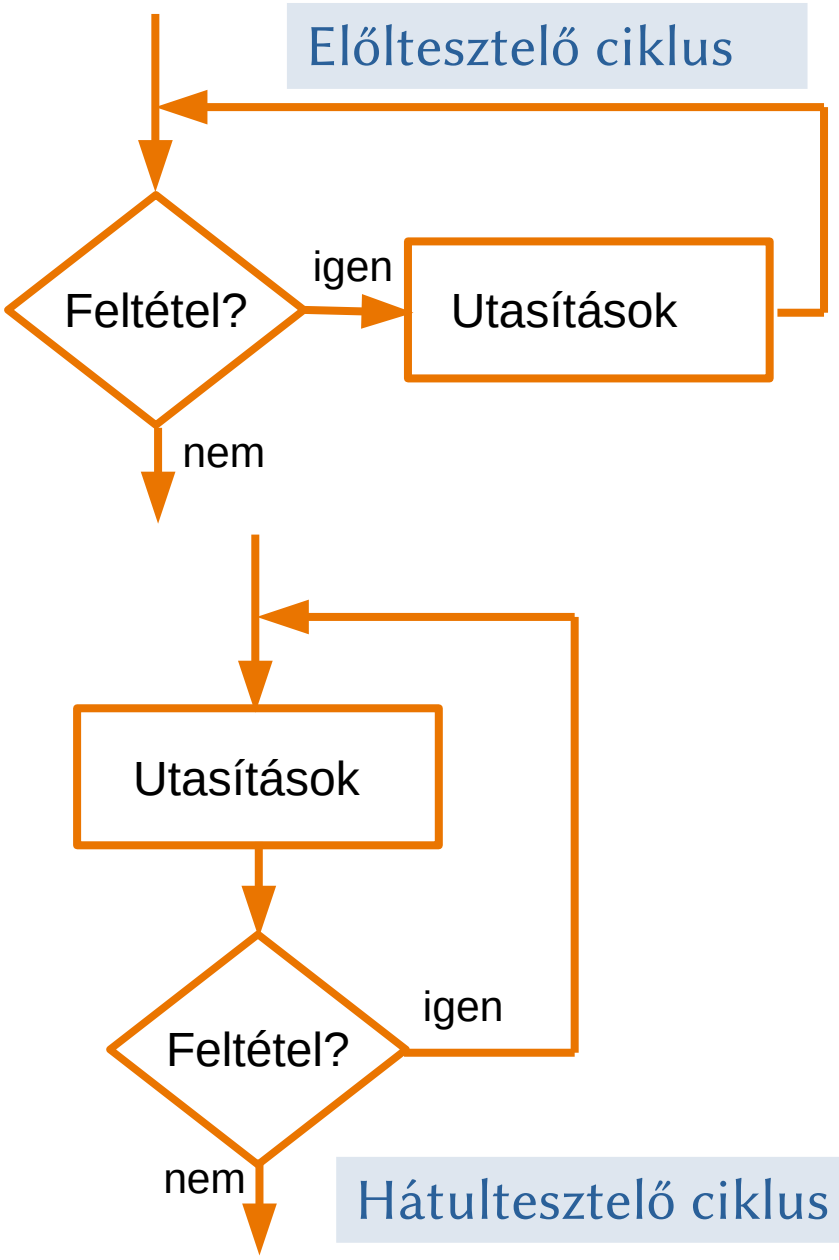
- Utasítások és blokkok
- Feltételes elágazás
  - **if – else** utasítás
  - **else – if** utasítás
  - **switch** utasítás
- Ciklusszervezés
  - **while** utasítás
  - **for** utasítás
  - **do – while** utasítás
- A **break** és **continue** utasítások
- A **goto** utasítás és **a címkék**

Az előző előadásban tárgyalt vezérlési szerkezetek

Forrás: BRIAN W. KERNIGHAN – DENNIS M. RITCHIE: A C programozási nyelv, 3. fejezet

Ciklusok szervezése

- Ismétlődő feladatok esetén az utasításainkat „újrahasznosíthatjuk” **ciklusok** szervezésével, amikor egy utasításblokkot többször lefuttatunk
- Nem (csak) végtelen ciklust akarunk, ezért kell bennmaradási (vagy kilépési) **feltétel**
- A feltételt az utasításblokk végrehajtása előtt vagy után is vizsgálhatjuk:
  - Előltesztelő ciklus: **while, for**
  - Hátultesztelő ciklus: **do ... while**



Ciklusszervezés while utasítással

`while (feltétel) utasítás` vagy **`blokk`** Ha a feltétel igaz, a **while** utasításhoz kapcsolódó utasítás (vagy blokk) végrehajtásra kerül, majd a végrehajtás visszakerül a feltétel kiértékeléséhez. Példák:

```
while(true) {   //LED villogtatás végtelen ciklusban
  digitalWrite(LED,!digitalRead(LED)); //LED állapot átbillentése
  delay(500);                          //500 ms várakozás
}
```

```
while(false) digitalWrite(LED, HIGH);   //Sohasem hajtódik végre
```

```
int i = 0;
  while (i < 5) {              //ötszöri LED villantás
    digitalWrite(LED,HIGH);
    delay(50);
    digitalWrite(LED,LOW);
    delay(500);
    i++;                       //Ciklusváltozó léptetése (FONTOS!)
  }
```

Részletesebben lásd a következő oldalon!

five_ledflash.ino

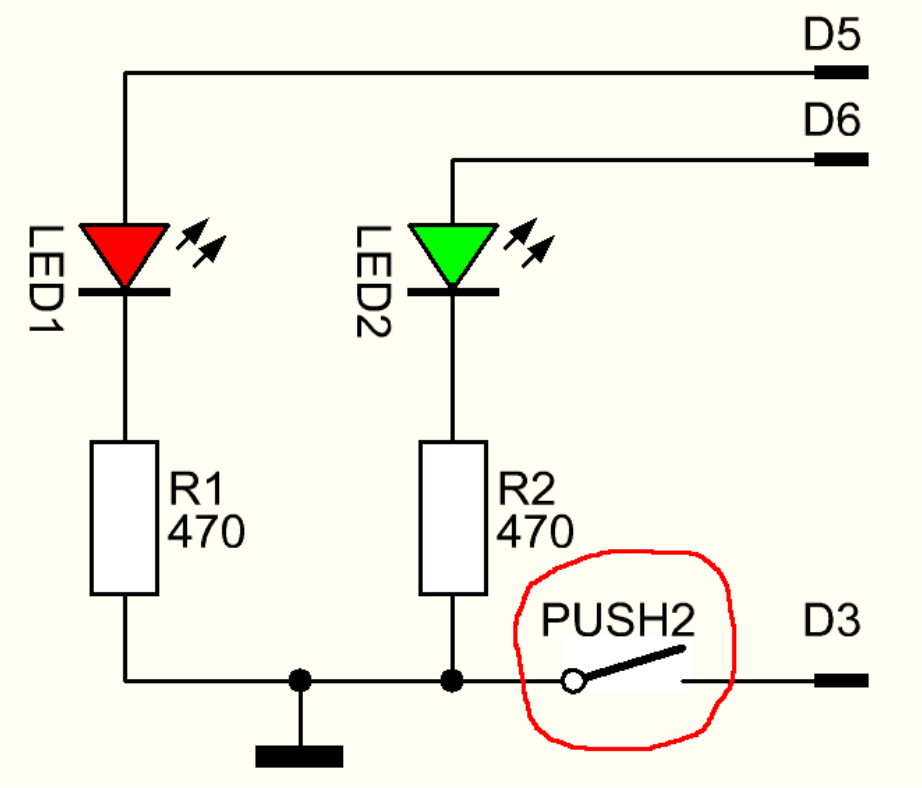
- Villantsunk ötöt a **D5** lábra kötött piros LED-del!
- A LED legyen bekapcsolva 50 ms, kikapcsolva 500 ms-ig!

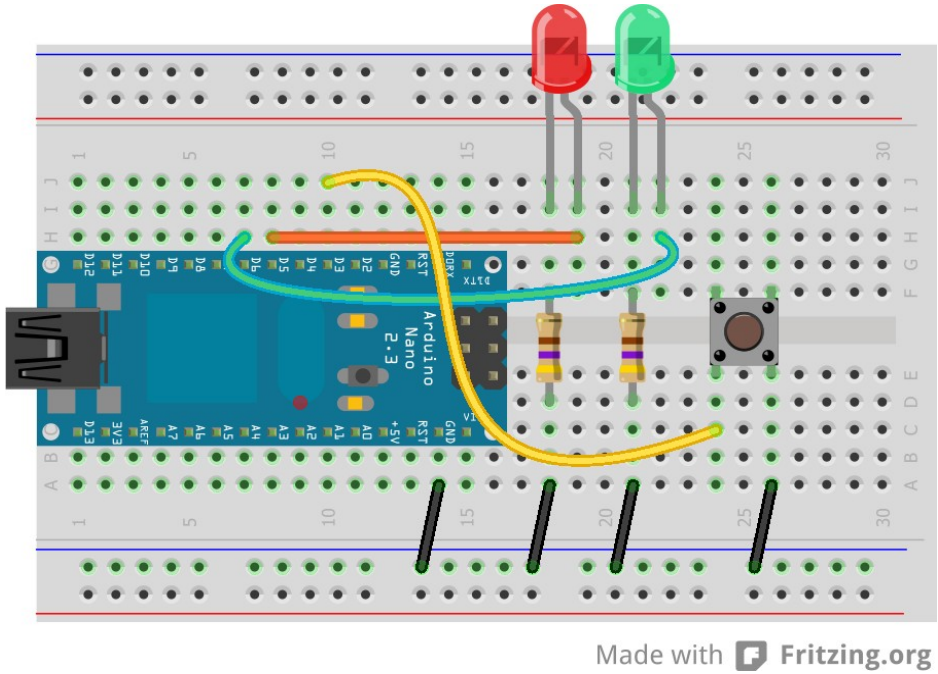
```
#define RED_LED 5                  // D5-re van kötve a LED
void setup() {
  pinMode(RED_LED,OUTPUT);         // Legyen kimenet
  int i = 0;
  while (i < 5) {                 // Ötszöri LED villantás
    digitalWrite(RED_LED,HIGH); // Felvillantjuk a LED-et
    delay(50);
    digitalWrite(RED_LED,LOW);   // Lekapcsoljuk a LED-et
    delay(500);
    i++;                           //Ciklusváltozó léptetése (FONTOS!)
  }
}
void loop() {
  // Nincs feladatunk
}
```

A programot a RESET gombbal indíthatjuk újra!

A piros LED kapcsolgatása nyomógombbal

- A múlt órai kapcsolásnál maradunk (a zöld LED-et nem használjuk)
  - Első (és minden páratlan) lenyomásra a piros LED világítson!
  - Második (és minden páros) lenyomásra a piros LED ne világítson!
- A nyomógomb állapotváltásaira **while** ciklusokban várakozzunk!





ledswitch2.ino

- A **while** utasítást itt arra használjuk, hogy egy esemény (pl. gomblenyomás, vagy felengedés) bekövetkeztére várakozzunk
- Várakozás közben nincs mit tenni, ezért a **while** ciklus törzse üres

```
const int RED_LED = 5;
const int PUSH2 = 3;
boolean led_state = false;          // Kezdetben a LED ki van kapcsolva
void setup() {
  pinMode(RED_LED,OUTPUT);          // Legyen kimenet
  pinMode(PUSH2,INPUT_PULLUP);      // Bemenet belső felhúzással
}

void loop() {
  while(digitalRead(PUSH2));        // Lenyomásra várunk
  led_state = !led_state;           // Állapot átbillentése
  digitalWrite(RED_LED,led_state); // LED állapot aktualizálás
  delay(20);                        // Pergésmentesítő késleltetés
  while(!digitalRead(PUSH2));       // Felengedésre várunk
  delay(20);                        // Pergésmentesítő késleltetés
}
```

Melyik a szimpatikusabb?

ledswitch.ino

```
#define RED_LED 5
#define PUSH2 3
boolean led_state = false;
boolean waitforpress = true;
void setup() {
  pinMode(RED_LED,OUTPUT);
  pinMode(PUSH2,INPUT_PULLUP);
}
void loop() {
  if(waitforpress) {
    if(!digitalRead(PUSH2)) {
      led_state = !led_state;
      digitalWrite(RED_LED, led_state);
      waitforpress = false;
    }
  }
  else {
    if(digitalRead(PUSH2)) {
      waitforpress = true;
    }
  }
  delay(20);
}
```

ledswitch2.ino

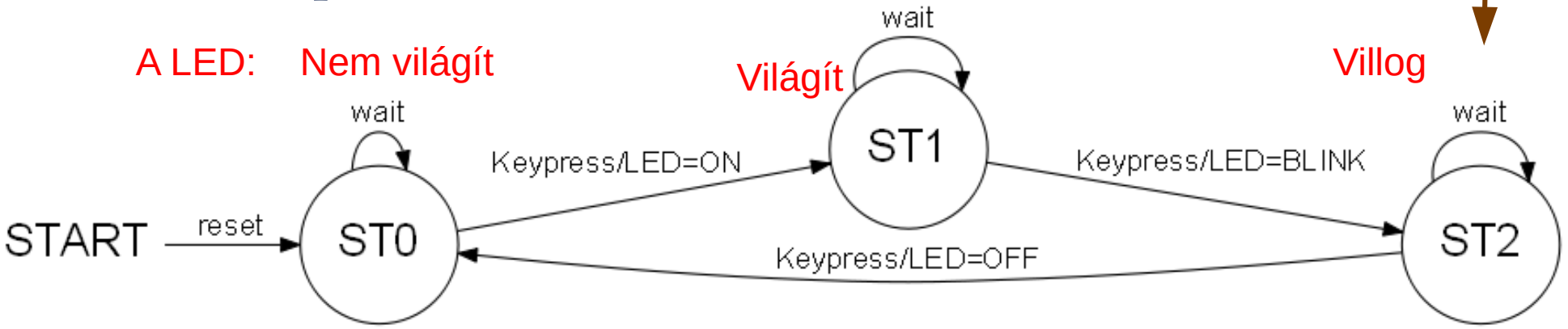
```
#define RED_LED 5
#define PUSH2 3
boolean led_state = false;
void setup() {
  pinMode(RED_LED,OUTPUT);
  pinMode(PUSH2,INPUT_PULLUP);
}
void loop() {
  while(digitalRead(PUSH2));
  led_state = !led_state;
  digitalWrite(RED_LED,led_state);
  delay(20);
  while(!digitalRead(PUSH2));
  delay(20);
}
```

A második program egyszerűbb. Az egyszerűség ára a blokkoló várakozások (**while** utasítások)

led_tristate.ino

Példaprogram, amelyben nem használhatunk blokkoló várakozást a LED villogtatása miatt!

- A piros LED vezérlése nyomógomb segítségével:
  - első lenyomásra a piros LED világít
  - második lenyomásra a piros LED villog (kb. 1 Hz)
  - harmadik lenyomásra a piros LED kialszik
- Minden gomblenyomás egy újabb állapotba viszi a LED-et:
  - 0: A LED nem világít (a kimenet LOW)
  - 1: A LED folyamatosan világít (a kimenet HIGH)
  - 2: A LED villog (a kimenet állapotát 500 ms-onként átbillentjük)
- A LED állapotát a **led_state** változóban tároljuk



led_tristate.ino

```
const int RED_LED = 5;              // D5-re van kötve a LED
const int PUSH2 = 3;                // D3-ra van kötve a nyomógomb

// Hardverfüggetlen rész (MSP430 Launchpad kártyán is futtatható!)
boolean waitforpress = true;        // Kezdetben lenyomásra várunk
int led_state = 0;                  // Kezdetben a LED ki van kapcsolva
int timecount = 25;                 // Villogás félperiódusának ideje
                                    // 20 ms egységekben (500 ms = 25*20 ms)
void setup() {
  pinMode(RED_LED,OUTPUT);          // Legyen kimenet
  pinMode(PUSH2,INPUT_PULLUP);      // Bemenet belső felhúzással
}
```

Folytatás a következő oldalon ...

A változók szerepe:

waitforpress: a nyomógomb *lenyomás – felengedés* ciklusának nyilvántartására használjuk (true = lenyomásra várunk, false = felengedésre várunk)

led_state: ebben tartjuk nyilván a LED állapotát (0 = nem világít, 1 = világít, 2 = villog)

timecount: a 20 ms-os időszeletek (vissza)számlálója villogtatáskor

led_tristate.ino

```
void loop() {
  if(waitforpress) {                  //Ha lenyomásra várunk és
    if(!digitalRead(PUSH2)) {         //Ha lenyomás történt...
      if(++led_state > 2) led_state = 0;     //Állapot léptetés
      digitalWrite(RED_LED, led_state > 0); //LED állapot beállítása
      waitforpress = false;           //Következő stáció: felengedésre várunk
    }
  }
  else {                              //Ha felengedésre vártunk és
    if(digitalRead(PUSH2)) {          //Ha felengedést észlelünk...
      waitforpress = true;            //Következő stáció: lenyomásra várunk
    }
  }
  delay(20);                          //pergésmentesítő késleltetés
  if(led_state == 2)                  //Ha villogtatás állapotban vagyunk
    if(--timecount == 0) {            //Ha az eltelt idő 25*20 ms, akkor
      digitalWrite(RED_LED,!digitalRead(RED_LED)); //LED átbillentése
      timecount = 25;                 //Következő billentés 25*20 ms múlva...
    }
}
```

KEYPRESS

BLINK

Adom a magyarázatot...

- A `loop()` függvény törzsében a nyomógombot kezelő állapotgép megegyezik a korábban bemutatott **`ledswitch.ino`** programmal, csak a gomblenyomáskot végzendő feladat lett más:

```
if(++led_state > 2) led_state = 0;     // Állapot léptetés
digitalWrite(RED_LED, led_state > 0); // LED állapot beállítása
```

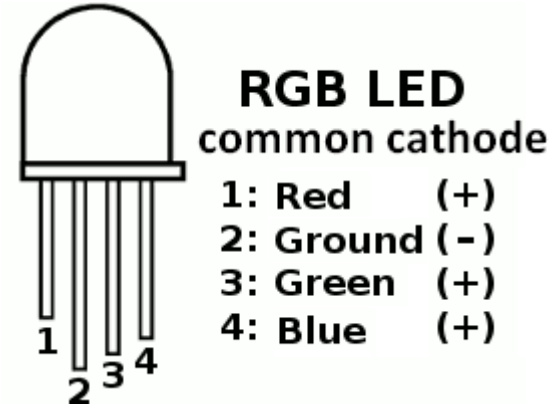
  Megnöveljük a számlálót és visszaállítjuk 0-ra, ha túl nagy lett
  led_state > 0 logikai kifejezés, **1:** ha led_state = 1 vagy 2, **0:** ha 0

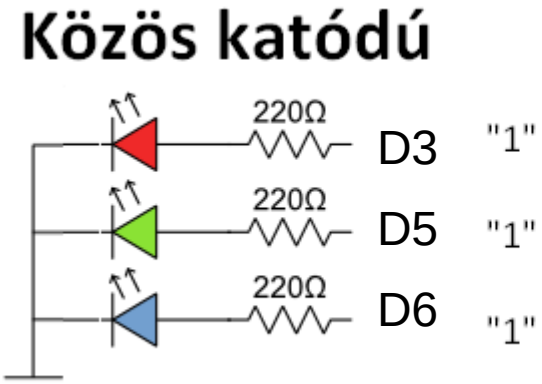
- Villogtatás csak akkor, ha led_state = 2
- LED állapot átbillentés, ha timecount 1-gyel csökkentve 0-ra fut (természetesen timecount értékét vissza kell állítani 25-re!

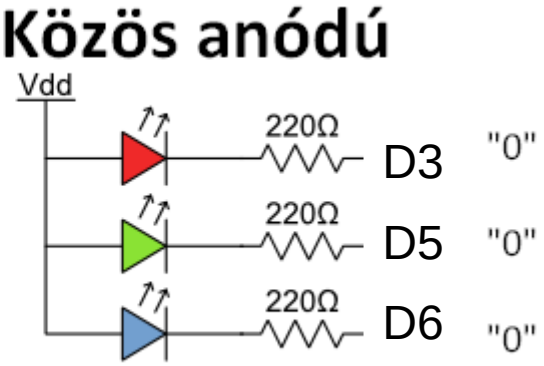
```
if(led_state == 2)                //Ha villogtatás állapotban vagyunk
    if(--timecount == 0) {        //Ha az eltelt idő 25*20 ms, akkor
      digitalWrite(RED_LED,!digitalRead(RED_LED)); //LED átbillentése
      timecount = 25;             //Következő billentés 25*20 ms múlva...
    }
```

RGB LED vezérlése

- Az RGB LED három, különböző színű LED, egy tokba zárva (**R:** piros, **G:** zöld, **B:** kék)
- Ügyeljünk rá, hogy közös katódú, vagy közös anódú LED-ünk van! A leghosszabb láb a közös kivezetés.
- A januári kísérleteknél már bevált közös anódú LED-et is használhatjuk, a közös lábnak ekkor a **D4** kimeneten biztosítunk magas szintet, a **D3, D5, D6** kimeneteket pedig invertáljuk a **!** (NEM) logikai művelettel
- Alkalmazzunk min. 220 Ω-os áramkorlátozó ellenállásokat!



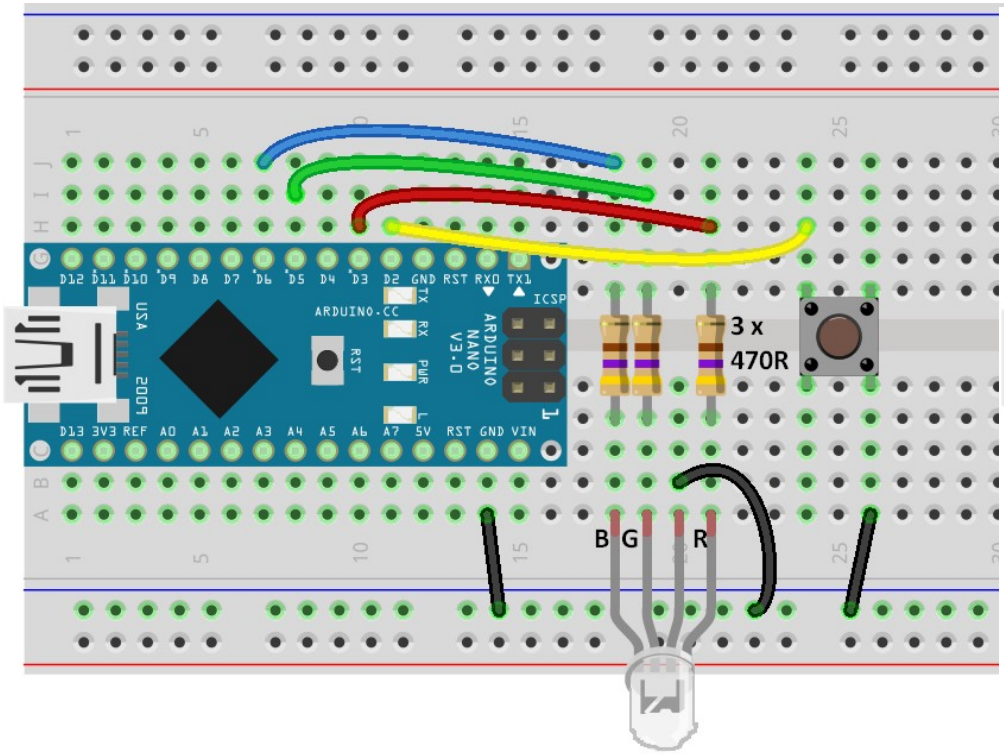




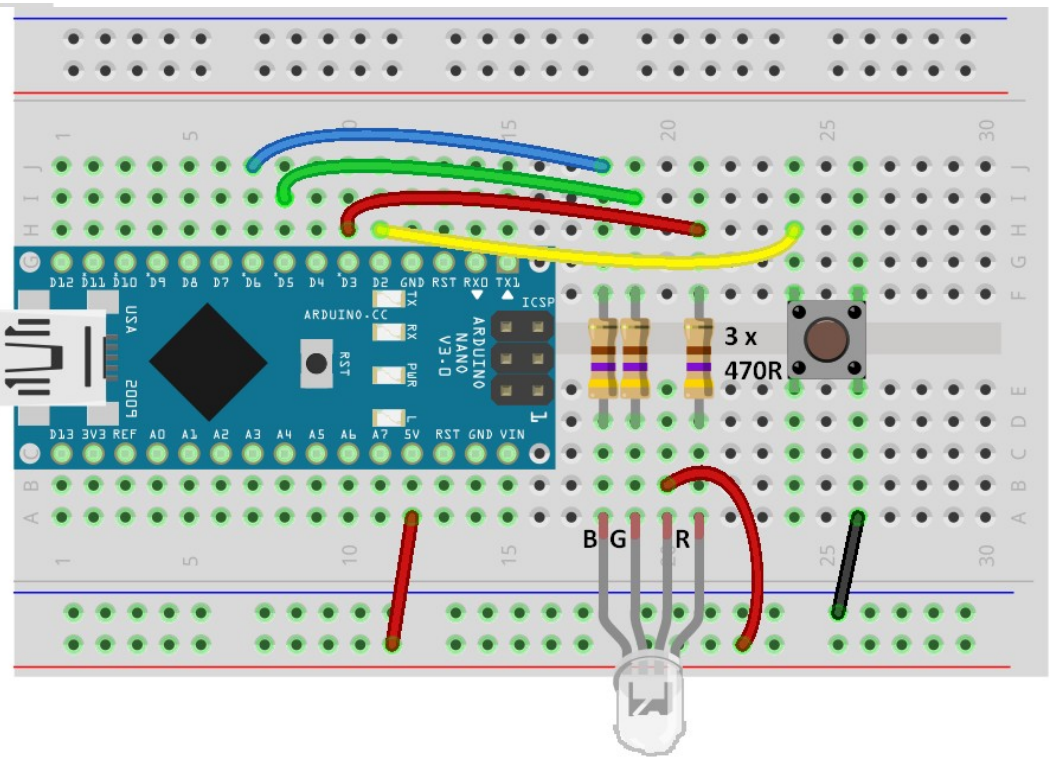
Kapcsolási elrendezés

- A nyomógombra, ami a **D2** lábra csatlakozik, az első példában nem, csak a másodikban lesz szükségünk
- Az alábbi kapcsolásokban 470 Ω-os ellenállásokat használtunk

Kapcsolás közös katód esetén



Kapcsolás közös anód esetén



RGB_ledblink.ino

```
#define LED_RED    3              // Piros LED
#define LED_ANODE  4              // Közös anód esetén
#define LED_GREEN  5              // Zöld LED
#define LED_BLUE   6              // Kék LED

int color = 0;                    // Számláló

void setup() {
  // Kimenetként inicializáljuk a LED vezérlő lábakat
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_ANODE, OUTPUT);     // Közös anód esetén
  pinMode(LED_BLUE, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  digitalWrite(LED_ANODE,HIGH);   // Közös anód esetén
}

void loop() {
  digitalWrite(LED_RED, !bitRead(color,0));   // piros LED beállítása
  digitalWrite(LED_GREEN,!bitRead(color,1));  // zöld LED ki
  digitalWrite(LED_BLUE,!bitRead(color,2));   // kék LED ki
  delay(2000);                                // várunk 2 másodpercig
  color = ++color & 0x07;                     // Csak 0 – 7-ig számoljon!
}
```

Ebben a programban egyszerűen végiglépkedünk a sötét, piros, zöld, sárga, kék, magenta, cián és fehér színeken

A negáció csak közös anódú RGB LED esetén kell!

Hogyan működik a számláló?

- A változót `int` típusúnak deklaráltuk, de csak 3 bitjét használjuk
- Léptetéskor eggyel megnöveljük az értékét, majd 7-tel maszkoljuk

```
color = ++color & 0x07;     // & a bitenkénti ÉS művelet
```

A maszkolás miatt 8 helyett 0 lesz az érték:

```
  0000 1000
AND 0000 0111
-------------
  0000 0000
```

- A színek kijelzéséhez bitekre kell bontani a **color** változót. Ehhez az Arduino nyelv `bitRead(x,n)` függvényt használjuk, ahol **x** a felbontani kívánt szám, **n** pedig a bit sorszáma (0. a legkisebb helyiértékű bit)
  - Piros szín: `bitRead(color,0)`
  - Zöld szín: `bitRead(color,1)`
  - Kék szín: `bitRead(color,2)`

BIT2	BIT1	BIT0
BLUE	GREEN	RED

RGB_ledswitch.ino

- Bonyolítsuk az előző programot azzal, hogy a színváltások egy nyomógomb lenyomásakor történnek!
- Szerencsére a nyomógombot kezelő programmal már rendelkezünk, csak a gomblenyomáskor elvégzendő teendőket kell kicserélni!

```
#define LED_RED    3              // Piros LED
#define LED_ANODE  4              // Közös anód esetén
#define LED_GREEN  5              // Zöld LED
#define LED_BLUE   6              // Kék LED
#define PUSH2      2              // Nyomógomb
int color = 0;                    //színkód (0..7)
boolean waitforpress = true;

void setup() {
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_ANODE, OUTPUT);     // Közös anód esetén
  pinMode(LED_BLUE, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  digitalWrite(LED_ANODE, HIGH); // Közös anód esetén
  pinMode(PUSH2, INPUT_PULLUP);
}
```

Új sorok, amelyekkel bővült a program az előzőhöz képest

Folytatás a következő oldalon!

RGB_ledswitch.ino

- A nyomógomb kezelése már ismerős a **ledswitch.ino** programból, csak a bekeretezett tevékenységeket kell idemásolni az **RGB_ledbink.ino** programból

```
void loop() {
  if (waitforpress) {                 //Ha lenyomásra várunk és
    if (!digitalRead(PUSH2)) {        //Ha lenyomás történt...
      color = ++color & 0x07;         //Szín léptetése
      waitforpress = false;           //Következő stáció: felengedésre várunk
    }
  }
  else {                              //Ha felengedésre vártunk és
    if (digitalRead(PUSH2)) {         //Ha felengedést észlelünk...
      waitforpress = true;            //Következő stáció: lenyomásra várunk
    }
  }
  digitalWrite(LED_RED, !bitRead(color, 0));   // piros LED beállítása
  digitalWrite(LED_GREEN, !bitRead(color, 1)); // zöld LED beállítása
  digitalWrite(LED_BLUE, !bitRead(color, 2));  // kék LED beállítása
  delay(20);                          //pergésmentesítő késleltetés
}
```

Az Arduino nano kártya kivezetései