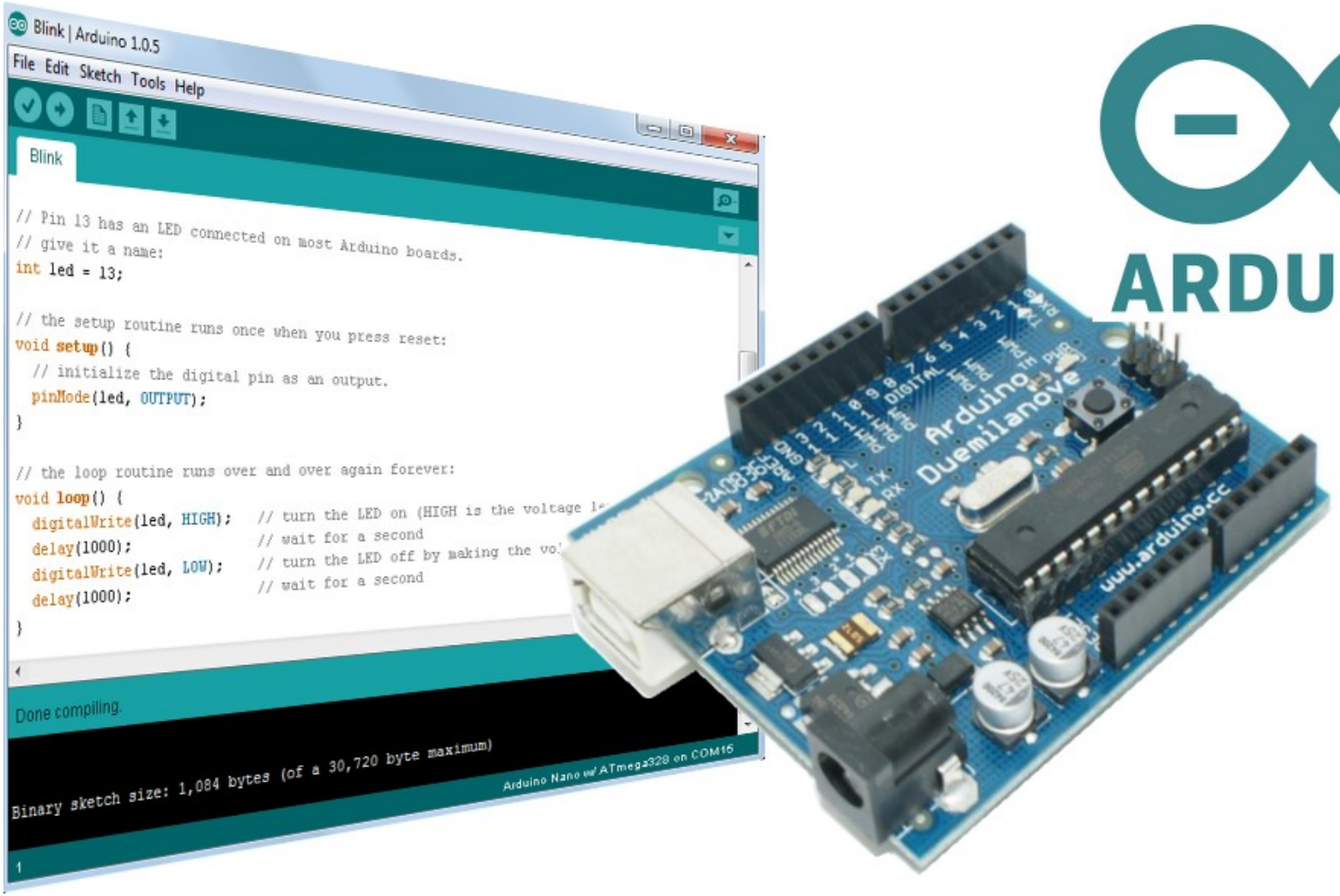
Bevezetés az elektronikába



10. Arduino programozás – I/O portok kezelése, változók típusai, feltételvizsgálat, programelágazás.

Egyszerű I/O vezérlés

Digitális I/O

- **pinMode**(pin, mode) – kivezetés üzemmódjának beállítása
- **digitalWrite**(pin, state) - kimenetvezérlés
- **digitalRead**(pin) – bemenet állapotának lekérdezése

  mode: OUTPUT, INPUT, INPUT_PULLUP *state: LOW, HIGH*

Analóg I/O

- **analogReference**(ref) – ADC referenciájának megadása
- **analogRead**(chan) – analóg-digitális konverzió eredménye
- **analogWrite**(pin) - PWM teljesítményvezérlés

  ref: DEFAULT (VCC), INTERNAL (1V1) vagy EXTERNAL

Digitális ki- és bemenetek konfigurálása

```
pinMode(pin, mode)
```

A kivezetés azonosítója

0 – 13, A0 – A5

Az adatáramlás iránya

OUTPUT: kimenetként viselkedik

INPUT: bemenetként viselkedik

INPUT_PULLUP: bemenet, felhúzással

- Az adatáramlás irányának beállításán kívül a fenti függvény feladata a digitális mód engedélyezése, s szükség esetén az adott lábra kapcsolódó megosztott funkciók (oszcillátor, timer, PWM, soros kommunikációs periféria, stb.) letiltása.
- A belső felhúzás hatása olyan, mintha egy ellenállással a tápfeszültségre kötnénk a bemenetet – szabadon hagyva magas szintet „érzékel”.

Digitális ki/bemenetek írása/olvasása

```
digitalWrite(pin, state)
```

A kivezetés azonosítója

0 – 13, A0 – A5

A kimenet állapota

LOW: alacsony szint („0”)

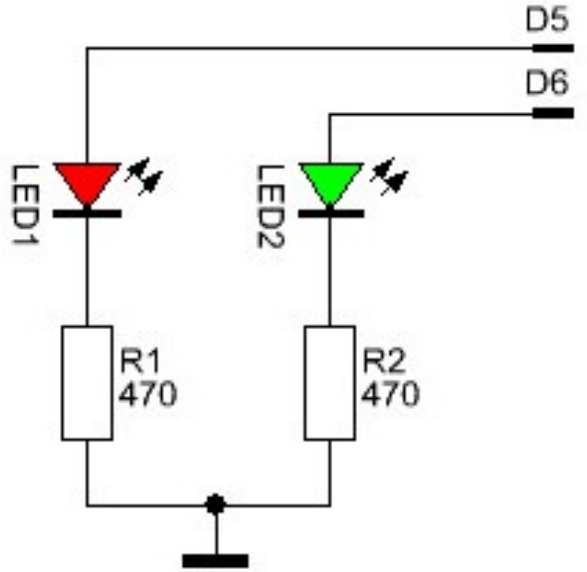
HIGH: magas szint („1”)

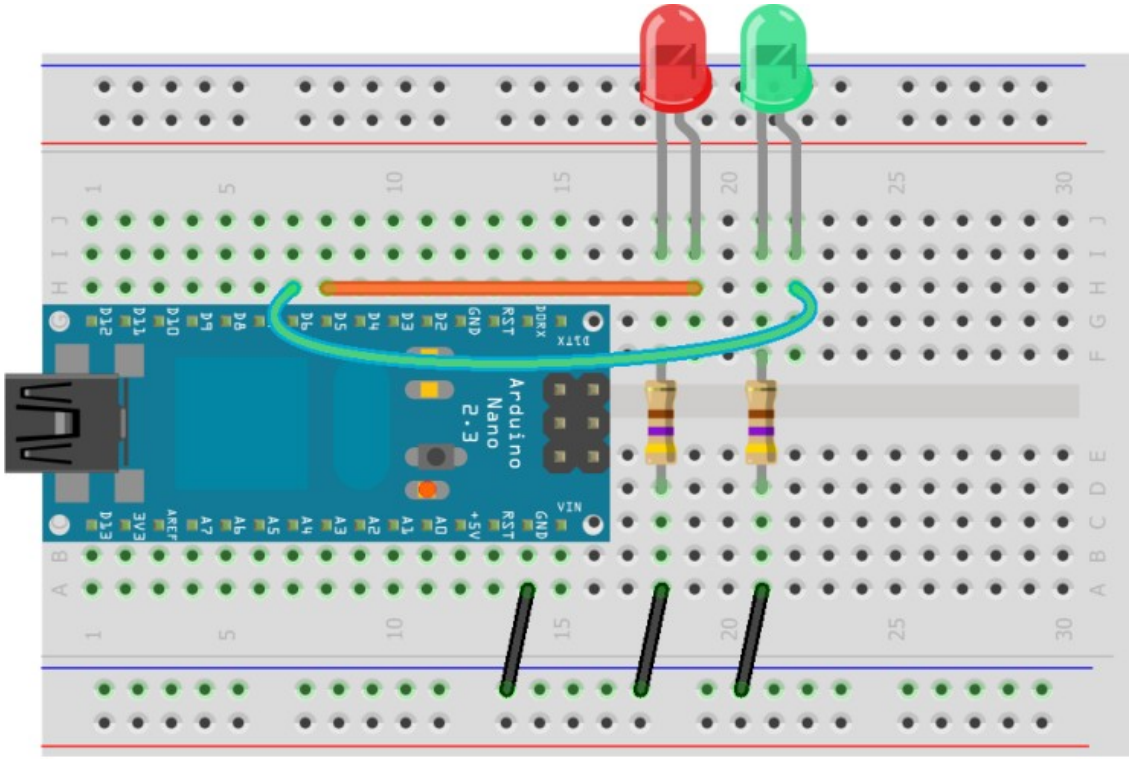
```
digitalRead(pin)
```

- A **digitalRead()** függvénynek csak egy bemenő paramétere van: az olvasni kívánt láb száma.
- A függvénynek van visszatérési értéke is, amely „1” vagy „0” lehet, a bemenet állapotától függően.

Két LED-es villogó: twoled.ino

- Villogtassunk két LED-et felváltva!
- **RED_LED** legyen a **D5, GREEN_LED** pedig a **D6** kimenetre kötve!





Made with Fritzing.org

Két LED-es villogó: twoled.ino

```
//Hardverfüggő rész: csak Arduino kártyához kell...
const int RED_LED = 5;
const int GREEN_LED = 6;

//Hardverfüggetlen rész (MSP430 Launchpad kártyán is futtatható!)
void setup() {
  pinMode(RED_LED,OUTPUT);        //legyen kimenet
  pinMode(GREEN_LED,OUTPUT);      //legyen kimenet
}

void loop() {
  digitalWrite(RED_LED,HIGH;      //RED_LED világít
  digitalWrite(GREEN_LED,LOW);    //GREEN_LED nem világít
  delay(1000);                    //1 s várakozás
  digitalWrite(RED_LED,LOW);      //RED_LED nem világít
  digitalWrite(GREEN_LED,HIGH);   //GREEN_LED világít
  delay(1000);                    //1 s várakozás
}
```

Változók és adattípusok

- **Arduino** környezetben az alábbi adattípusokat használhatjuk:

Típus	Méret byte-ban	Tárolható értékek	
		Minimum	Maximum
boolean	1	false (0)	true (1)
char	1	– 128	+128
byte	1	0	255
int	2	– 32 768	+32 767
unsigned int / word	2	0	65 536
long	4	– 2 147 483 648	2 147 483 647
unsigned long	4	0	4 294 967 295
float	4	– 3,4028235E+38	3,4028235E+38
double	4	Azonos a float típussal számítási erő hiányában	

STDINT megfelelők (ANSI C90)

int8_t
uint8_t
int16_t

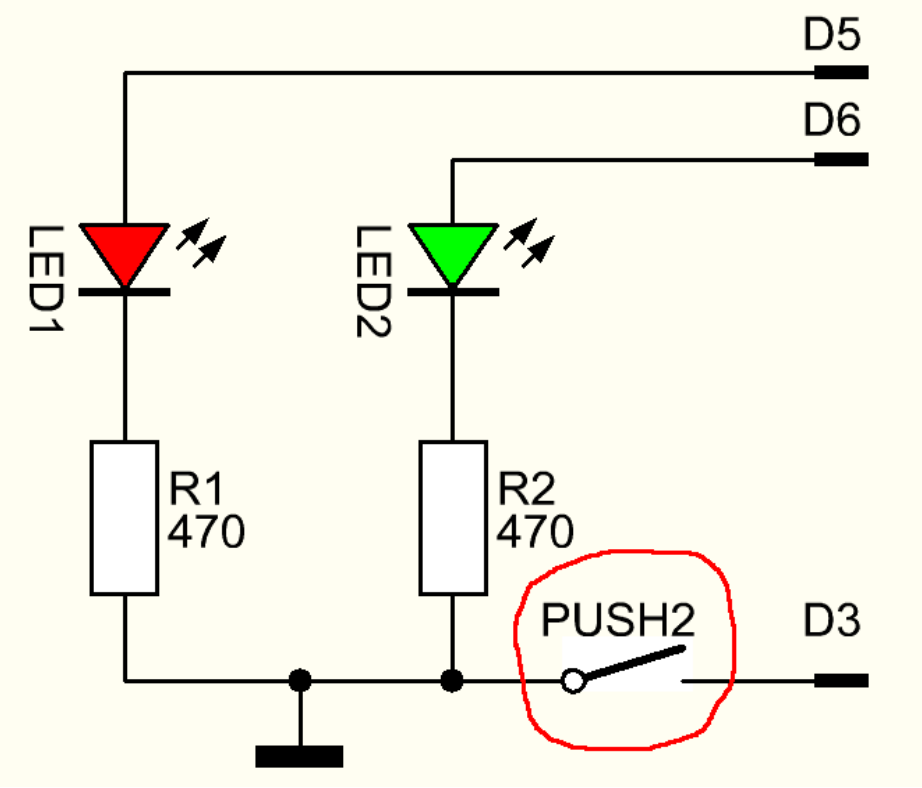
uint16_t
int32_t
uint32_t

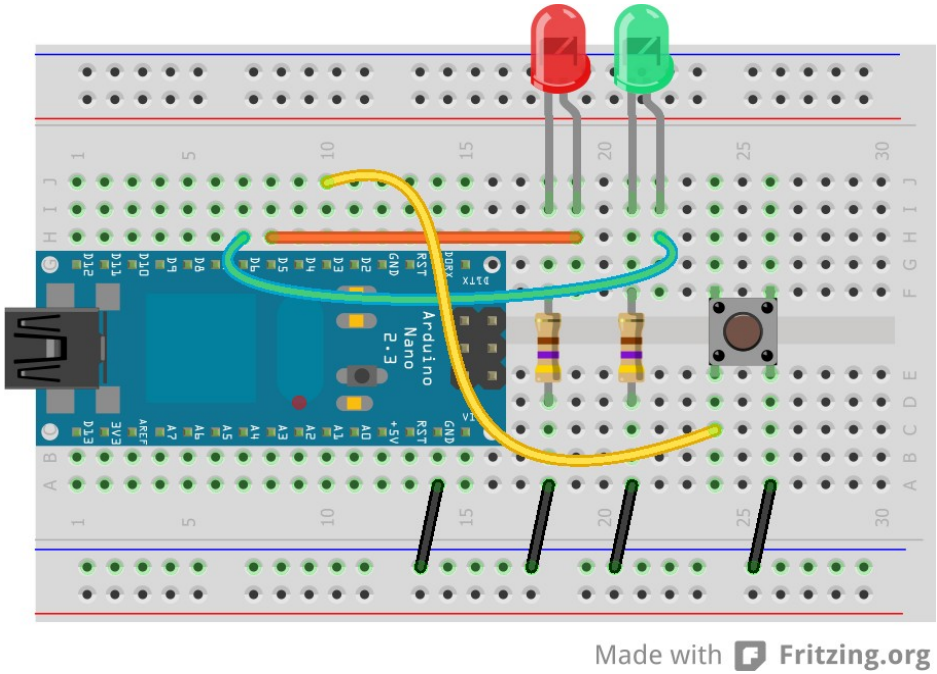
- Változók deklarálásakor meg kell adni az adattípust és a változó nevét. Opcionálisan kezdőértéket is adhatunk a változónak.

```
int eletkor;        char name[] = ”Pista vagyok”;  //tömböt definiál
unsigned int sum = 0;          //Előjel nélküli egész (2 byte)
boolean animal = false;        //logikai változó
const RED_LED = 13;  //A const kontanst (nem módosítható értéket) jelöl
```

Nyomógomb állapotának beolvasása

- **Feladat:** A két LED felváltva világítson, a kapcsoló állásától függően:
  - Ha a kapcsoló nyitva van, a piros LED világítson!
  - Ha a kapcsoló zárva van, a zöld LED világítson!
- A nyomógomb állapotát a **digitelRead()** függvénnyel vizsgáljuk!





button2led.ino

```
// Hardverfüggő rész Arduino kártyához
const int RED_LED = 5;
const int GREEN_LED = 6;
const int PUSH2 = 3;

// Hardverfüggetlen rész (MSP430 Launchpad kártyán is futtatható!)
void setup() {
  pinMode(RED_LED,OUTPUT);        // legyen kimenet
  pinMode(GREEN_LED,OUTPUT);      // legyen kimenet
  pinMode(PUSH2,INPUT_PULLUP);    // Bemenet belső felhúzással
}

void loop() {
  boolean sw;
  sw = digitalRead(PUSH2);        // beolvassuk a  nyomógomb állapotát
  digitalWrite(RED_LED,sw);       // RED_LED akkor világít, ha sw == HIGH
  digitalWrite(GREEN_LED,!sw);    // GREEN_LED akkor világít, ha sw == LOW
  delay(20);                      // pergésmentesítő késleltetés
}
```

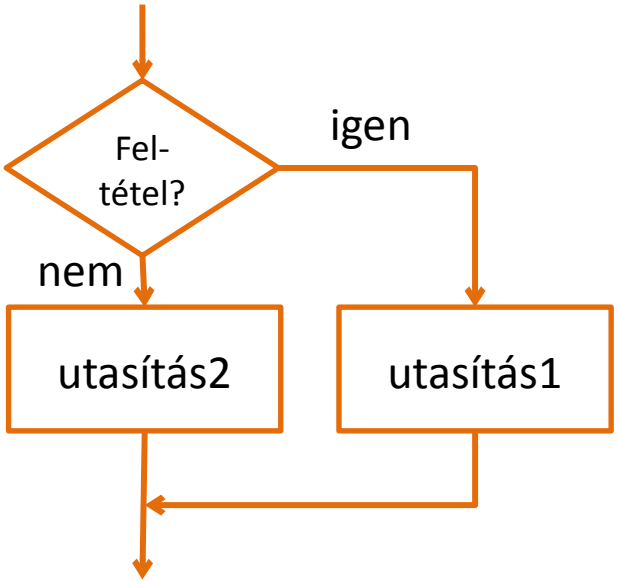
A !-jel a logikai tagadás (NEM) műveletének jele.

Feltételes elágazás: if – else utasítás

- Megvizsgál egy feltételt, s ezt követően a végrehajtás menete a feltételvizsgálat eredményétől függ

```
if(feltétel) utasítás1
else utasítás2
```

(az **else** ág elhagyható)



- A **feltétel** egy kifejezés, ami aritmetikai és relációs operátorokat is tartalmazhat. A kiértékelés eredménye **igaz,** ha a kifejezés értéke **true,** vagy 0-tól különböző, és **hamis** akkor, ha a kifejezés értéke **false,** vagy nulla.
- **Ha a feltétel teljesül,** akkor **utasítás1** kerül végrehajtásra, **egyébként** pedig **utasítás2** lesz végrehajtva. Példa:

```
if(digitalRead(PUSH2)) { digitalWrite(RED_LED,LOW); }
else  { digitalWrite(RED_LED,HIGH); } //Amíg a gomb lenyomva, a LED világít
```

Relációs és logikai operátorok

- Logikai kifejezésekben, illetve feltételvizsgálatoknál az alábbi operátorok (műveletek) állnak rendelkezésre:

Relációs operátorok

==	egyenlő	pl. x == y
!=	nem egyenlő	pl. x != y
<	kisebb mint	pl. x < y
>	nagyobb mint	pl. x > y
<=	kisebb, vagy egyenlő	
>=	nagyobb, vagy egyenlő	

Logikai operátorok

Logikai változókon vagy logikai kifejezéseken végezett műveletekhez használhatók

!	Tagadás (negáció)	pl. !a
&&	ÉS művelet	pl. a && b
||	VAGY művelet	pl. a || b

- **Megjegyzések:**
  - A relációs, illetve a logikai műveletek eredménye egy logikai (boolean) mennyiség, ami csak 0 (ha hamis) vagy 1 (ha igaz) értéket vehet fel.
  - A C fordítók kiterjesztett értelmezéssel bármilyen 0-tól különböző számot is elfogadnak igaz értékkent. Így például if(x != 0) helyett if(x) -et is írhatunk.
  - Ne keverjük össze a logikai operátorokat a bitenként logikai operátorokkal! Bitenkénti műveletek: ~, &, |, ^ Logikai műveletek: !, &&, ||

Nyomógomb vizsgálata: button.ino

- Vizsgáljuk a nyomógomb állapotát **if()** utasítással!
- Jelezzük ki a gomb állapotát a D5-re kötött LED-del!

```
#define buttonPin   3       // a nyomógomb bemenet sorszáma
#define ledPin      5       // a LED kimenet sorszáma

int buttonState = 0;        // változó a nyomógomb állapot kiolvasásához

void setup() {
  pinMode(ledPin, OUTPUT); // ledPin legyen kimenet
  pinMode(buttonPin, INPUT_PULLUP);     // Bemenet, belső felhúzással
}

void loop() {
  buttonState = digitalRead(buttonPin); //Nyomógomb állapot beolvasása
  //--- gomblenyomás ellenőrzése (lenyomott állapotban LOW az eredmény)
  if (buttonState == LOW) {             // Ha a gomb le van nyomva
    digitalWrite(ledPin, HIGH);         // LED bekapcsolása
  } else {                              // különben (ha a gomb nincs lenyomva)
    digitalWrite(ledPin, LOW);          // LED kikapcsolása
  }
}
```

A kapcsolás a 8. oldalon látható

LED ki/bekapcsolása nyomógombbal

- Próbáljunk meg egy LED-et ki- és bekapcsolni egy nyomógombbal úgy, hogy az első (és minden páratlan) lenyomás bekapcsolja a LED-et, amely világít akkor is, ha elengedjük a gombot, a második (és minden páros) lenyomás pedig kikapcsolja a LED-et!
- A legrosszabb, amit megtehetünk, ha így fogunk hozzá:

```
if (digitalRead(PUSH2) == LOW) {
    digitalWrite(RED_LED, !digitalRead(RED_LED));
}
```

- Mi a probléma a fenti elgondolással?

A kapcsolás a 8. oldalon látható

LED ki/bekapcsolása nyomógombbal

- Próbáljunk meg egy LED-et ki- és bekapcsolni egy nyomógombbal úgy, hogy az első (és minden páratlan) lenyomás bekapcsolja a LED-et, amely világít akkor is, ha elengedjük a gombot, a második (és minden páros) lenyomás pedig kikapcsolja a LED-et!
- A legrosszabb, amit megtehetünk, ha így fogunk hozzá:

```
if (digitalRead(PUSH2) == LOW) {
    digitalWrite(RED_LED, !digitalRead(RED_LED));
}
```

- Mi a probléma a fenti elgondolással?

A probléma az, hogy a program nagyon gyorsan fut a nyomógomb lenyomásához képest, így a LED akár többszázszor is állapotot válthat, mire elengedjük a gombot és a **PUSH2** bemenet magas szintre vált. Ezért gondoskodnunk kell róla, hogy lenyomásonként csak egy állapotváltás történjen!

Állapotjelzők használata

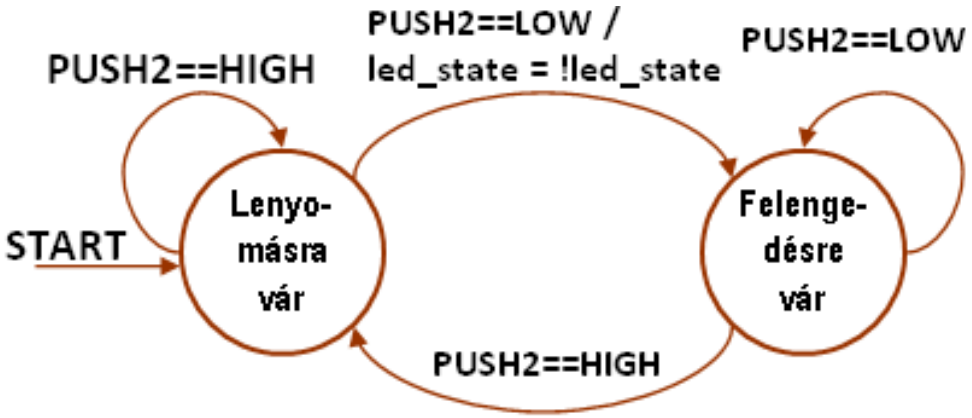
- **A nyomógombnak két állapota van:**
  - Felengedett állapot (azaz lenyomásra várunk)
  - Lenyomott állapot (azaz felengedésre várunk)
- Tároljuk az állapotot a **`waitforpress`** nevű logikai változóban Legyen **`waitforpress = true`** ha lenyomásra várunk, s legyen **`waitforpress = false`** , ha felengedésre várunk!
- Tároljuk el a LED állapotát is, hogy ne kelljen mindig beolvasni! **`led_state = true`** a bekapcsolt állapot, s **`false`** a kikapcsolt
- A programot tehát így kezdhetjük:

```
#define RED_LED 5                  // D5-re van kötve a LED
#define PUSH2 3                    // D3-ra kapcsolódik a nyomógomb
boolean led_state = false;         // Kezdetben a LED ki van kapcsolva
boolean waitforpress = true;       // Kezdetben lenyomásra várunk
void setup() {
  pinMode(RED_LED,OUTPUT);         // Legyen kimenet
  pinMode(PUSH2,INPUT_PULLUP);     // Bemenet belső felhúzással
}
```

A kapcsolás a 8. oldalon látható

Mi az állapotgép?

- A determinisztikus véges állapotgép a pillanatnyi álapottól és a beérkező jeltől (vagy bekövetkező eseménytől) függően lép a következő állapotba
- Ilyen állapotgépet valósít meg a programunknak az a része is, ami a nyomógombot kezeli
- A nyomógomb két állapotából a lenyomás, illetve a felengedés esemény visz új állapotba
- A LED állapotát átkapcsoló tevékenységet a lenyomás hatására bekövetkező állapotváltáskor kell elvégezni



A nyomógomb kezelését végző programrész állapotdiagramja

Mire jó az állapotgépes megközelítés?

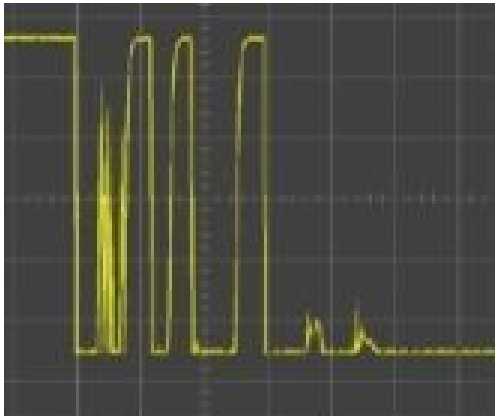
- Segít abban, hogy átlátható formában tervezzünk programot
- Tippek:
  `digitalRead(PUSH2)==LOW` helyett `!digitalRead(PUSH2)` is írható
  `digitalRead(PUSH2)==HIGH` helyett `digitalRead(PUSH2)` is írható

```
void loop() {
  if(waitforpress) {                  // Ha lenyomásra várunk és
    if(!digitalRead(PUSH2)) {         // Ha lenyomás történt...
      led_state = !led_state;         // Állapot átbillentése
      digitalWrite(RED_LED, led_state);  // LED állapot aktualizálás
      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
}
```

A kapcsolás a 8. oldalon látható

Mi az a pergésmentesítés?

- A nyomógombok, kapcsolók érintkezője „pereg”, azaz többször be- és kikapcsol, amikor átváltjuk. Ennek a bizonytalan állapotnak az ideje 10 – 15 ms.



A nyomógomb lenyomásakor többször megváltozik a jelszint, mire stabilizálódik az alacsony szint.
A gomb felengedésekor is hasonló jelenség figyelhető meg

- Szoftverese pergésmentesítés lényege: a gyors változásokat figyelmen kívül hagyjuk. Többféle megoldás lehetséges:
  - Csak bizonyos időközönként (pl. 20 ms) vizsgáljuk a nyomógomb állapotát (ritka mintavételezés). Erre szolgál programunk végén a 20 ms késleltetés.
  - Csak a stabilizálódott (bizonyos ideig változatlan) állapotokat vesszük figyelembe (sűrű mintavételezés és számlálás)

LED vezérlése: ledswitch.ino

```
#define RED_LED 5                   // D5-re van kötve a LED
#define PUSH2 3                     // D3-ra kapcsolódik a nyomógomb
boolean led_state = false;          // Kezdetben a LED ki van kapcsolva
boolean waitforpress = true;        // Kezdetben lenyomásra várunk
void setup() {
  pinMode(RED_LED,OUTPUT);          // Legyen kimenet
  pinMode(PUSH2,INPUT_PULLUP);      // Bemenet belső felhúzással
}
void loop() {
  if(waitforpress) {                // Ha lenyomásra várunk és
    if(!digitalRead(PUSH2)) {       // Ha lenyomás történt...
      led_state = !led_state;       // Állapot átbillentése
      digitalWrite(RED_LED, led_state);  // LED állapot aktualizálás
      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
}
```

A kapcsolás a 8. oldalon látható

Az Arduino nano kártya kivezetései