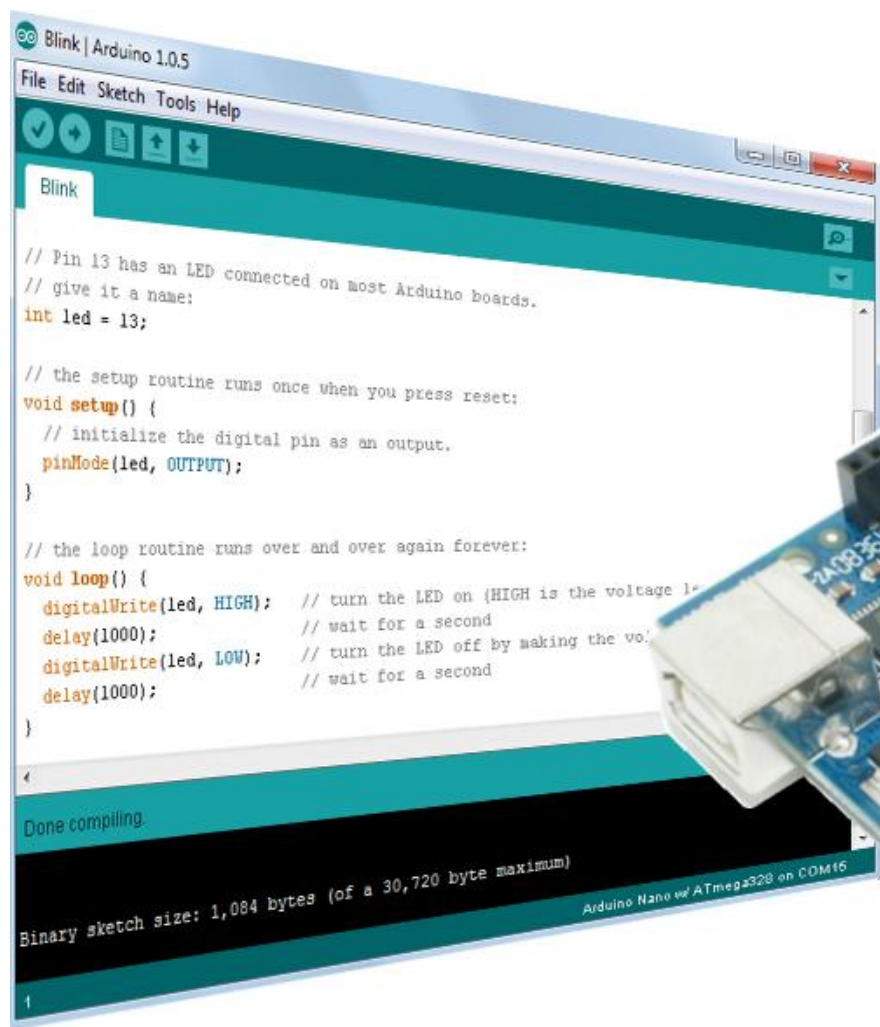




Bevezetés a mikrovezérlők programozásába:
I/O portok kezelése, számrendszerek, bitműveletek

Ajánlott irodalom

- ❑ Aduino LLC.: [Arduino Language Reference](#)
- ❑ ATMEL: [ATmega328p mikrovezérlő adatlapja](#)
- ❑ Brian W. Kernighan, Dennis Ritchie: [A C programozási nyelv](#)
- ❑ Cseh Róbert: Arduino programozási kézikönyv
- ❑ Ruzsinszki Gábor: Mikrovezérlős rendszerfejlesztés C/C++ nyelven I. – PIC mikrovezérlők
- ❑ Ruzsinszki Gábor: Mikrovezérlős rendszerfejlesztés C/C++ nyelven II. – Arduino

Lab 11 projektek



twoled – Két LED-et felváltva villogtatunk (digitális kimenetek kezelése)



twoled_fastio – Két LED-et felváltva villogtatunk, közvetlen portkezeléssel.



button2led – Két LED felváltva világít, a kapcsoló állásától függően (digitális bemenet kezelése)

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** (Arduino)

Energia esetén **INTERNAL** helyett **INTERNAL1V5** vagy **INTERNAL2V5** használható

Digitális be- és kimenetek konfigurálása

pinMode(pin, state)



A kivezetés azonosítója

Arduino: 1 – 13, A0 – A5

MSP430: P1_0 – P1_7, P2_0 – P2_7

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 (oszillátor, timer, PWM, soros kommunikációs periféria, stb.) letiltása.

Energia: előre definiált nevek az MSP430G2553 Launchpad kártyához

RED_LED = P1_0 a piros LED-et vezérlő kimenet

GREEN_LED = P1_6 a zöld LED-et vezérlő kimenet

PUSH2 = P1_3 az SW2 nyomógombhoz csatlakozó kivezetés

Digitális ki/bemenetek írása/olvasása

`digitalWrite(pin, mode)`

A kivezetés azonosítója

Arduino: 1 – 13, A0 – A5

MSP430: P1_0 – P1_7, P2_0 – P2_7

A kimenet állapota

LOW: alacsony szint („0”)

HIGH: magas szint („1”) (**Arduino:** 5V,
MSP430: 3.5V)

`digitalRead(pin)`

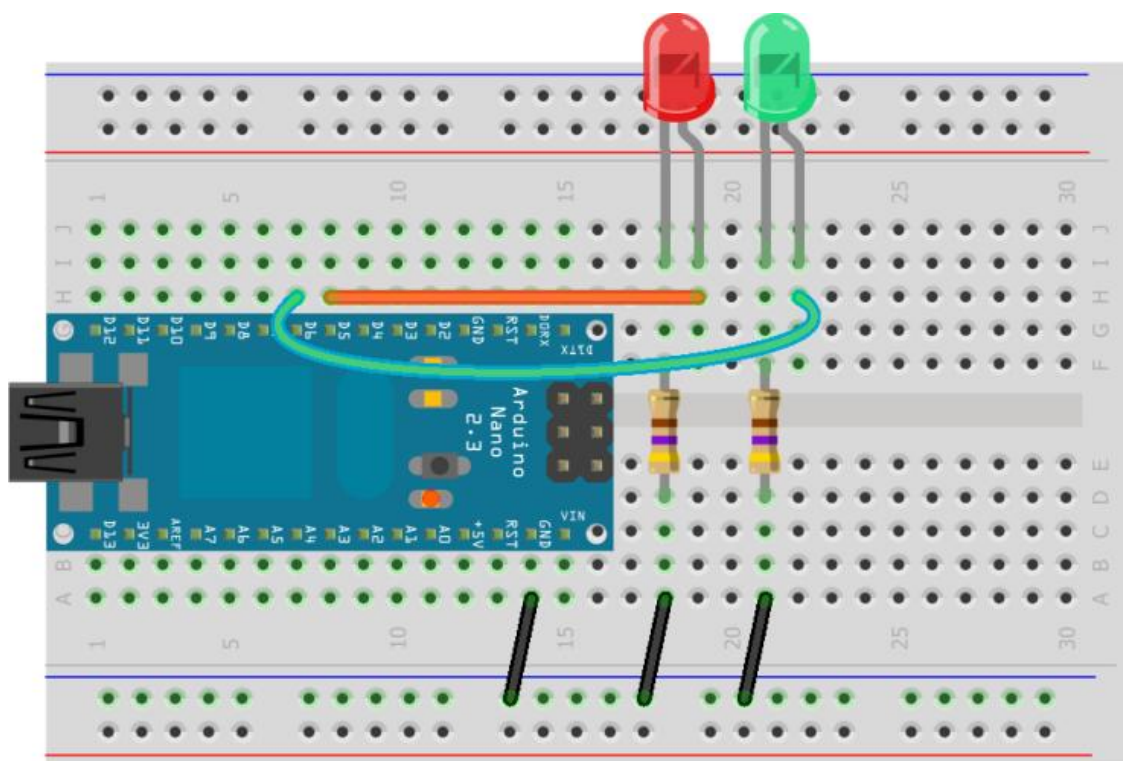
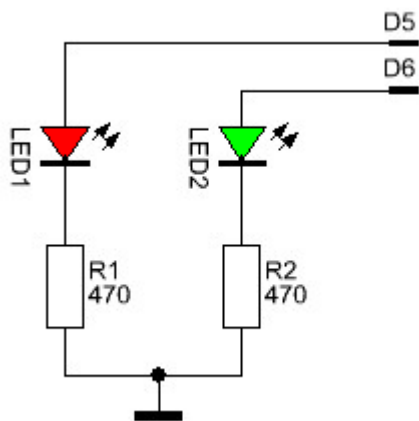
Ennek a függvénynek csak egy bemenő paramétere van: az olvasni kívánt láb száma. A függvénynek van visszatérési értéke is, amely „1” vagy „0” lehet, a bemenet állapotától függően.

TwoLED: két LED-es villogó

Villogtassunk két LED-et felváltva!

Arduino: RED_LED legyen a D5, GREEN_LED pedig a D6 kimenetre kötve!

Energia: MSP430 Launchpad kártyánál használjuk a beépített LED-eket!



Made with  Fritzing.org

twoled.ino

```
//Hardverfügő 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
```

```
}
```

Közvetlen I/O port vezérlés

Portvezérlő regiszterek

ATmega168/328 (x = B, C, D)

DDRx adatáramlási irány
(1: kimenet, 0: bemenet)

PORTx kimeneti adatregiszter

PINx olvasáskor a bemenetek
pillanatnyi állapotát adja át

MSP430G2553 (n = 1,2)

PnDIR adatáramlási irány
(1: kimenet, 0: bemenet)

PnOUT kimeneti adatregiszter

PnIN olvasáskor a bemenetek
pillanatnyi állapotát adja át

A közvetlen portkezelés hátrányai

- ❖ Nehezebb a program áttekintése, hibakeresése
- ❖ Nem hordozható a program
- ❖ Könnyebben okozhatunk galibát, hibás működést, mellékhatást

A közvetlen portkezelés előnyei

- ❖ Csak így oldható meg a gyors ki-bekapcsolás
- ❖ Több bit egyidejűleg ki-bekapcsolható
- ❖ Memóriatakarékosabb programot eredményez

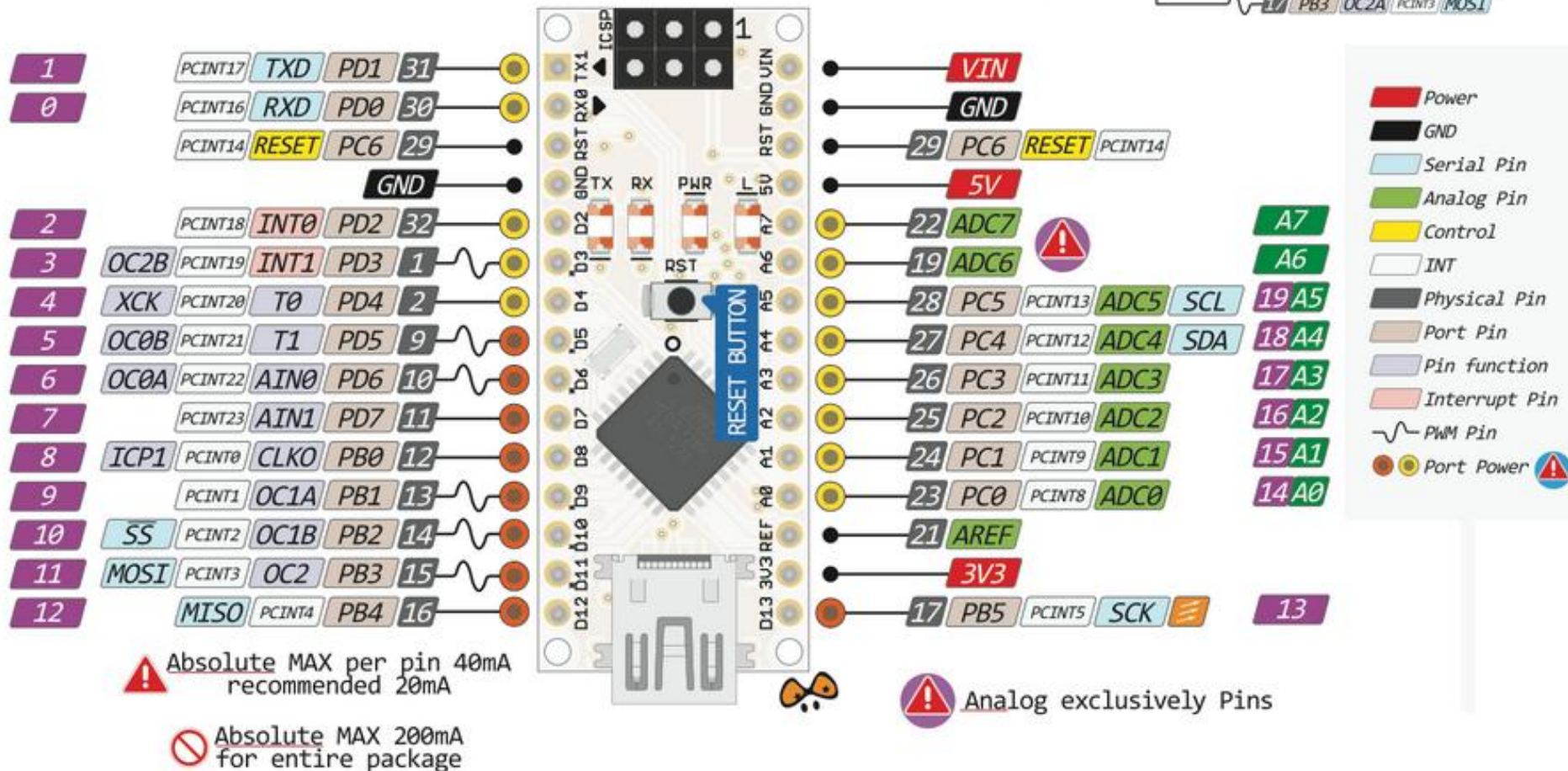
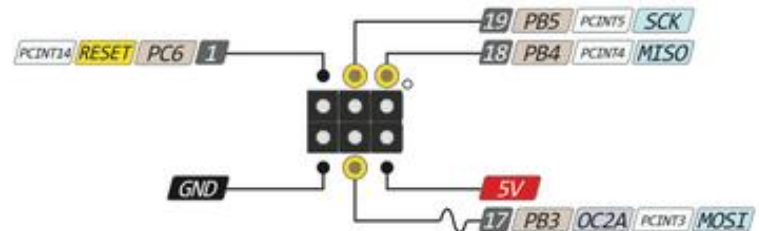
Forrás: www.arduino.cc/en/Reference/PortManipulation

Arduino nano v3.0



NANO PINOUT

⚠ The power sum for each pin's group should not exceed 100mA



twoled_fastio (Arduino)

Két LED-et villogtatunk ellenütemben, közvetlen portkezeléssel

Ez a program csak ATmega186, vagy ATmega328 kártyán fut (Arduino UNO, nano, mini...), MSP430 Launchpad kártyán nem, mivel a portkezelés itt hardverfügő módon történik!

RED_LED itt is a D5 lábra, GREEN_LED pedig a D6 lábra van kötve...

ARDUINO változat

```
void setup() {  
    //Kezdeti beállítások  
    DDRD |= 0b01100000;    //PORTD 5 és 6 bitje legyen kimenet!  
    PORTD |= 0b00100000;    //Kezdetben PORTD_5 legyen HIGH!  
    PORTD &= 0b10111111;    //Kezdetben PORTD_6 legyen LOW!  
}  
  
// a loop függvény újra és újra ismétlődik a végtelenségig  
void loop() {  
    PORTD ^= 0b01100000;    // a LED-ek állapotát átbillentjük (XOR művelettel)  
    delay(1000);            // várunk egy másodpercig  
}
```

MSP430 Launchpad : Energia Pinout

<http://github.com/energia/Energia/wiki/Hardware>



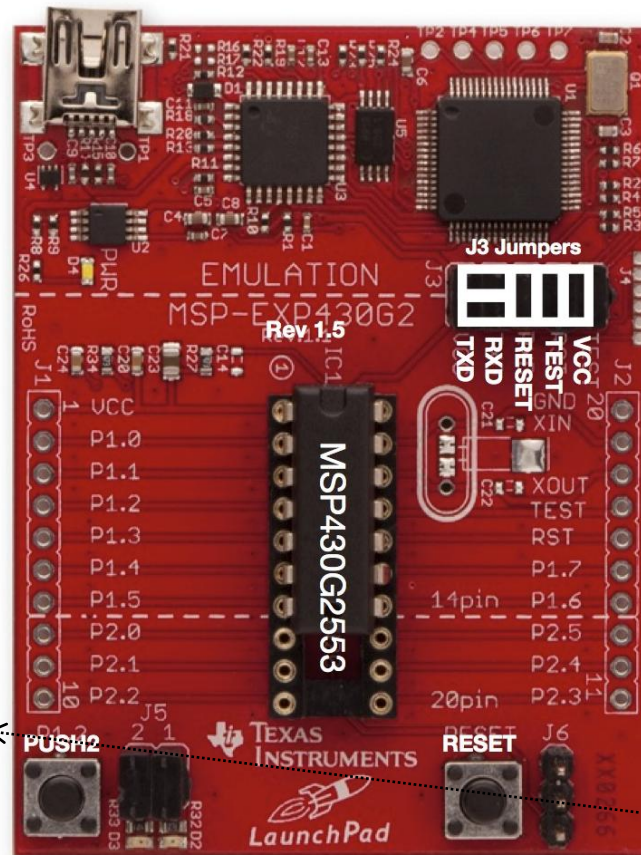
Energia

LaunchPad with MSP430G2553

Revision 1.5

Flash 16 KB
Serial Hardware

+3.3V				1
RED_LED		A0	P1_0	2
	RXD	A1	P1_1	3
	TXD	A2	P1_2	4
PUSH2		A3	P1_3	5
		A4	P1_4	6
	SCK (B0)	A5	P1_5	7
	CS (B0)		P2_0	8
			P2_1	9
			P2_2	10



Hardware
Pin number

I²C
Serial UART
SPI

analogRead()
digitalRead() and digitalWrite()
digitalRead(), digitalWrite()
and analogWrite()

20				GROUND
19	P2_6			XIN
18	P2_7			XOUT
17				TEST
16				RESET
15	P1_7	A7	SDA	MOSI (B0)
14	P1_6	A6	SCL	MISO (B0)
13	P2_5			GREEN_LED
12	P2_4			
11	P2_3			

Rei Vilo, 2012

embeddedcomputing.weebly.com

version 1.3 2102-09-09

Arduino/Energia logical pin #'s

twoled_fastio (Energia)

Két LED-et villogtatunk ellenütemben, közvetlen portkezeléssel

Ez a program csak **MSP430G2553** vagy **MSP430G2452 Launchpad** kártyán fut más kártyán nem, mivel a portkezelés itt hardverfügő módon történik!

RED_LED a P1_0 lábra, GREEN_LED pedig a P1_6 lábra van kötve gyárilag...

MSP430 Launchpad változat

```
void setup() {  
    //Kezdeti beállítások  
    P1DIR |= 0b01000001;    //PORT1 0 és 6 bitje legyen kimenet!  
    P1OUT |= 0b00000001;    //Kezdetben P1_0 legyen HIGH!  
    P1OUT &= 0b10111111;    //Kezdetben P1_6 legyen LOW!  
}  
  
// a loop függvény újra és újra ismétlődik a végtelenségig  
void loop() {  
    P1OUT ^= 0b01000001;    // a LED-ek állapotát átbillentjük (XOR művelettel)  
    delay(1000);            // várunk egy másodpercig  
}
```

Változók és adattípusok

Arduino/Energia környezetben az alábbi adattípusokat használhatjuk:

Típus	Méret byte-ban	Tárolható értékek	
		Minimum	Maximum
<i>boolean</i>	1	<i>false (0)</i>	<i>true (1)</i>
<i>char</i>	1	- 128	+128
<i>byte</i>	1	0	255
<i>int</i>	2	- 32 768	+32 767
<i>unsigned int / word</i>	2	0	65 536
<i>long</i>	4	- 2 147 483 648	2 147 483 647
<i>unsigned long</i>	4	0	4294967 295
<i>float</i>	4	- 3,4028235E+38	3,4028235E+38
<i>double</i>	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` életkor; `char` name[] = "Pista vagyok"; //tömböt definiál

`unsigned int` sum = 0; `boolean` animal = false;

`const` RED_LED = 13; //A `const` módosító konstans (nem módosítható érték) jelöl

Számrendszerek, átváltások

A bináris számrendszerben felírt számok bitjeit 4-es csoportokban könnyen átválthatjuk hexadecimális számjegyekké.

**Arduino (C/C++) programokban
így írhatjuk:**

```
x = 123;           //Decimális alakot  
x = 0x7B;          //Hexadecimális alak  
x = 0b01111011;    //Bináris alak
```

**Negatív számok ábrázolása
2-es komplementessel történik**

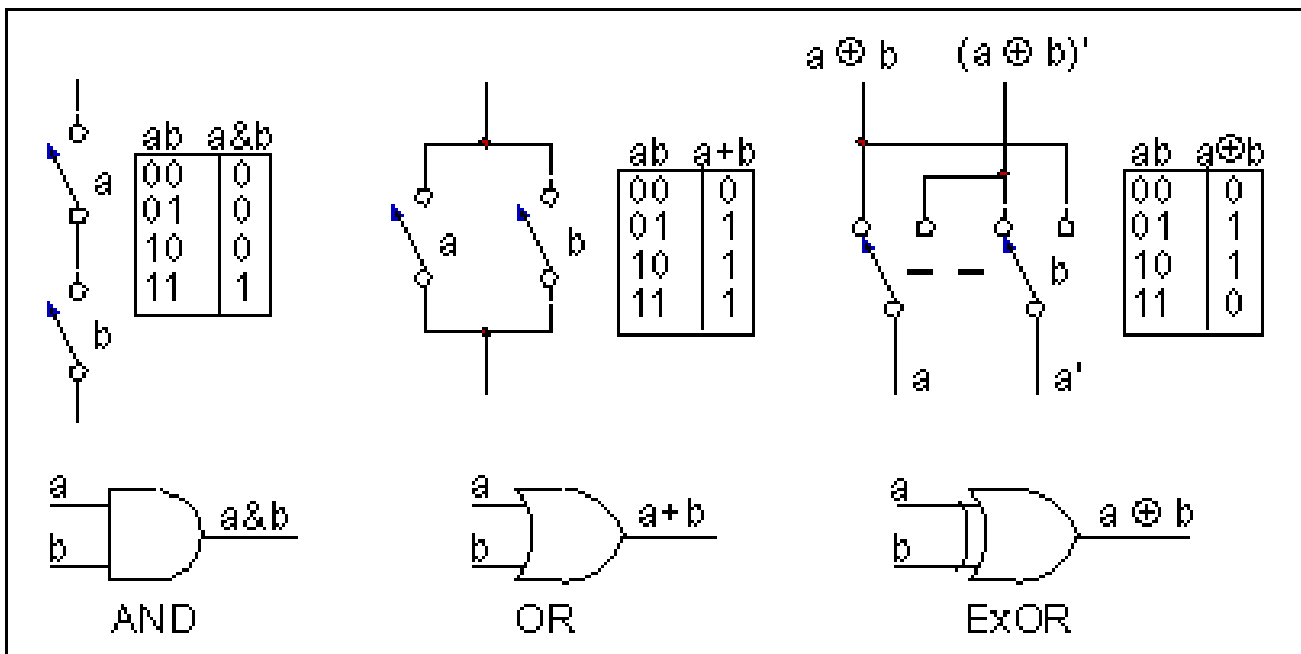
Szám	int8	int16
-1	0xFF	0xFFFF
-2	0xFE	0xFFFE
-3	0xFD	0xFFFD
-4	0xFC	0xFFFC
...	...	...

Decimal (Base 10)	Binary (Base 2)	Hexadecimal (Base 16)
0	00000000	0x0
1	00000001	0x1
2	00000010	0x2
3	00000011	0x3
4	00000100	0x4
5	00000101	0x5
6	00000110	0x6
7	00000111	0x7
8	00001000	0x8
9	00001001	0x9
10	00001010	0xA
11	00001011	0xB
12	00001100	0xC
13	00001101	0xD
14	00001110	0xE
15	00001111	0xF

Bitműveletek a C nyelvben

A C viszonylag hardverközelí nyelv, ezért fontos szerepe van a bitműveleteknek, amikor az azonos helyiértékű bitek között végzünk logikai műveleteket. Ezeket a műveleteket használhatjuk bitcsoportok törlésére, 1-be állítására, vagy logikai negálására.

A jobbra/balra léptetéssel pedig gyorsan és hatékonyan oszthatunk/szorozhatunk 2 hatványaival.



Bitwise Operators

$\&$ (bitwise and)

$|$ (bitwise or)

$\wedge$ (bitwise xor)

$\sim$ (bitwise not)

$\ll$ (bitshift left)

$\gg$ (bitshift right)

$Y = a \& b;$ $Y = a | b;$ $Y = a \wedge b;$

Összetett műveletek

A C nyelv tömörségére jellemző, hogy az **A = A + 1;** vagy **A = A * valami;** helyett így is írhatjuk: **A++;** vagy **A *= valami;**

Compound Operators

++ (increment)	a++ ➡	a = a + 1;
-- (decrement)	a-- ➡	a = a - 1;
+= (compound addition)	a += b; ➡	a = a + b;
-= (compound subtraction)	a -= b; ➡	a = a - b;
*= (compound multiplication)	a *= b; ➡	a = a * b;
/= (compound division)	a /= b; ➡	a = a / b;
&= (compound bitwise and)	a &= b; ➡	a = a & b;
= (compound bitwise or)	a = b; ➡	a = a b;

Megjegyzés: az **a++** és **a--** postfix műveletek, tehát ha értékadás jobboldalán vagy logikai relációban szerepelnek, azok kiértékelésénél az **a** változó régi értéke lesz felhasználva, s csak utána történik az inkrementálás/dekrementálás. Ha az új (megnövelt/csökkentett) értéket akarjuk felhasználni, akkor a művelet prefix megfelelőjét kell használni: **++a**, vagy **--a**.

Bit vagy bitcsoport 1-be állítása

(BIT6 + BIT0)

`P1DIR |= 0x41;` `// 0x41 = 0100_0001b`

Jelentése: $P1DIR = P1DIR \mid 0x41;$

Elv: $x \mid 1 = 1$

Példa:

0 x 1 0	1 1 0 x	kiindulási érték
0 1 0 0	0 0 0 1	bitmaszk
0 1 1 0	1 1 0 1	eredmény

Bit vagy bitcsoport 0-ba törlése

(BIT6 + BIT0)

`P1DIR &= ~0x41;` `// ~0x41 = 1011_1110b`

Jelentése: `P1DIR = P1DIR & 0xDE;`

Elv: $x \& 0 = 0$

Példa:

0	x	1	0	1	1	0	x	kiindulási érték
1	0	1	1	1	1	1	0	bitmaszk
0	0	1	0	1	1	0	0	eredmény

Bit vagy bitcsoport átbillentése

(BIT6 + BIT0)

`P1DIR ^= 0x41;` `// 0x41 = 0100_0001b`

Jelentése: $P1DIR = P1DIR \wedge 0x41;$

Elv: $x \wedge 1 = \sim x$ $x \wedge 0 = x$

Példa:

0	1	1	0	1	1	0	0	kiindulási érték
0	1	0	0	0	0	0	1	bitmaszk
0	0	1	0	1	1	0	1	eredmény

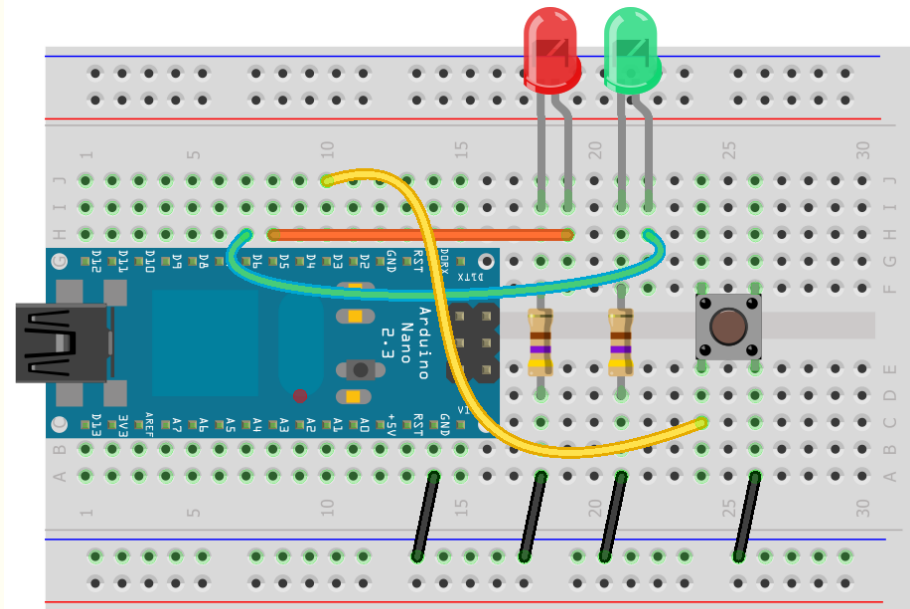
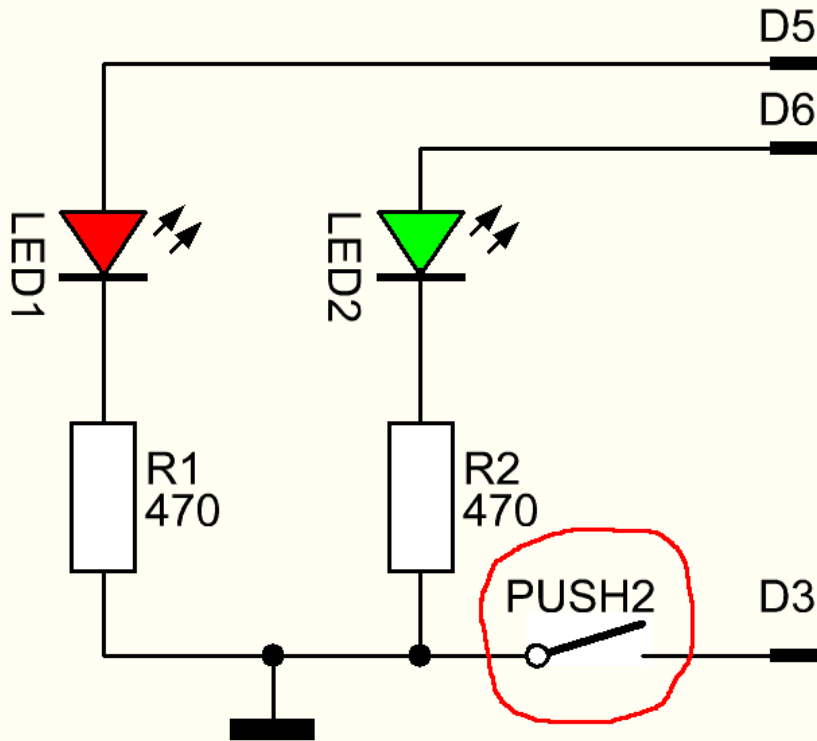
Nyomógomb állapotának beolvasása

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!

Megjegyzés: Az **MSP430 Launchpad** kártya esetében a gyárilag ráépített LED-eket és az SW2 nyomógombot használhatjuk.

Arduino kártya esetén az alábbi kapcsolást építsük meg!



Made with Fritzing.org

button2led.ino

```
const int RED_LED = 5;  
const int GREEN_LED = 6;  
const int PUSH2 = 3;
```

Hardverfüggő rész,
csak az Arduino kártyához kell!

```
//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 beslő 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  
}
```