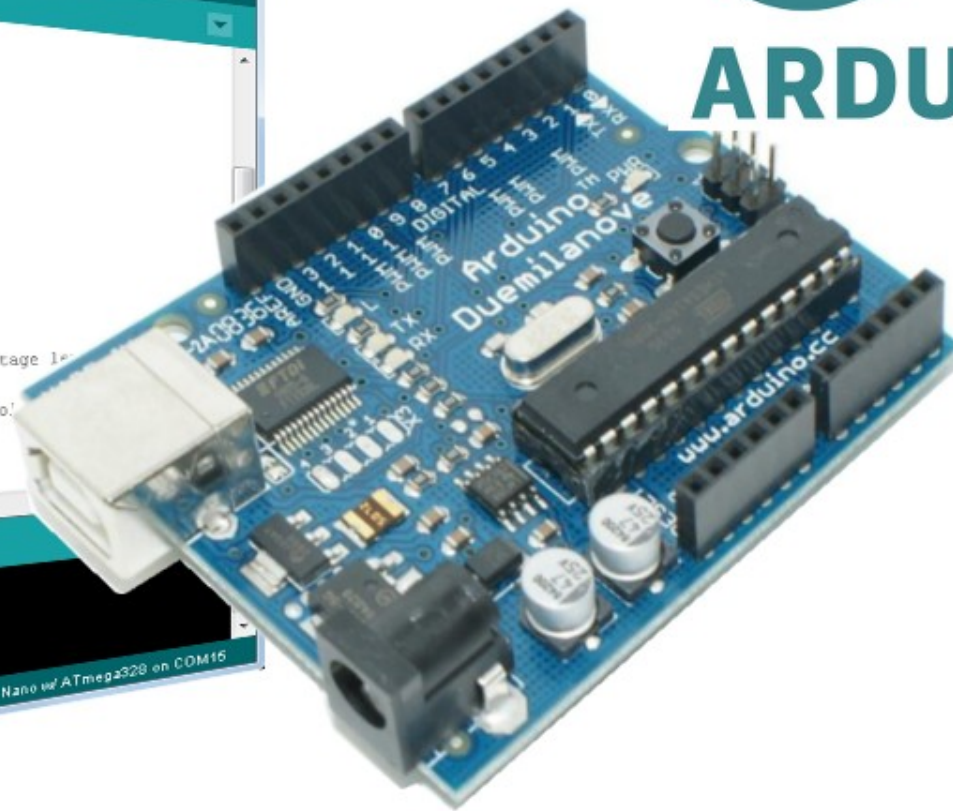
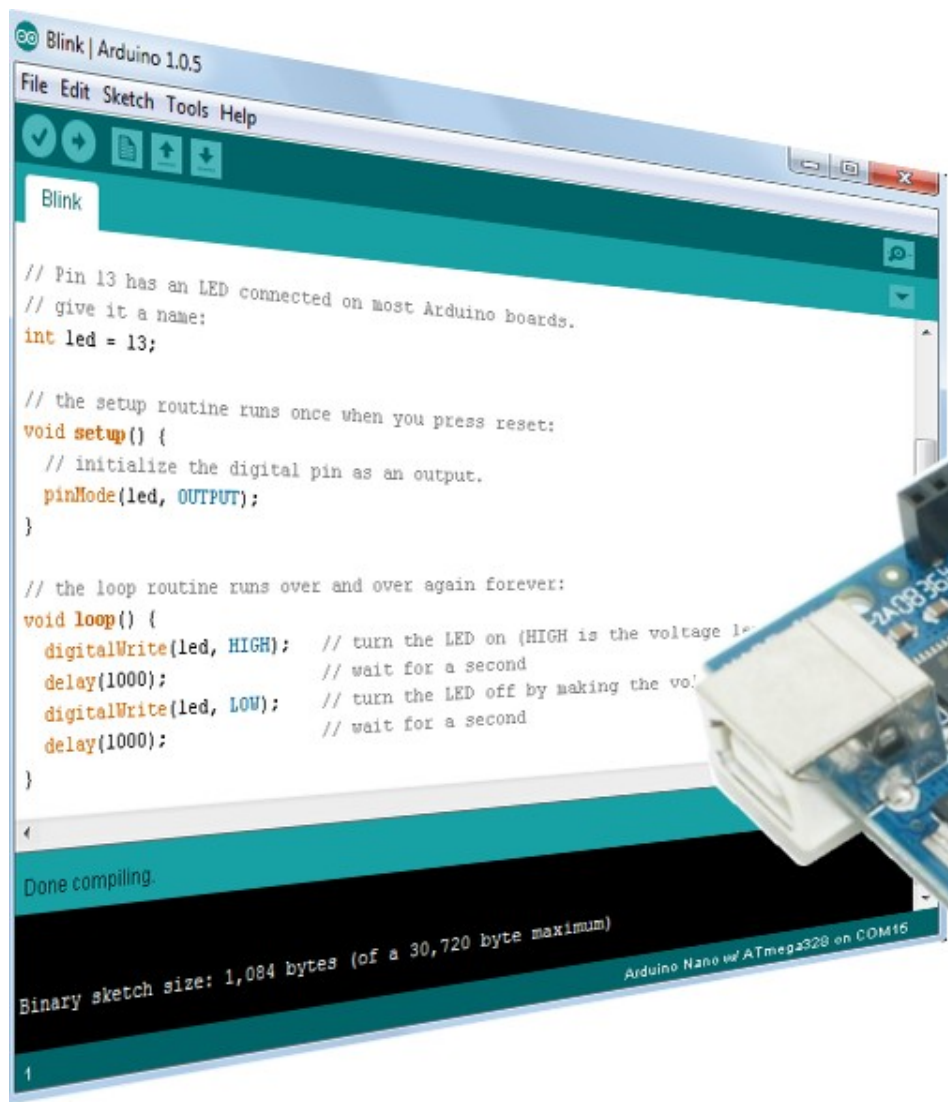


Arduino tanfolyam kezdőknek és haladóknak



1. Ismerkedés az Arduino kártyával, digitális I/O

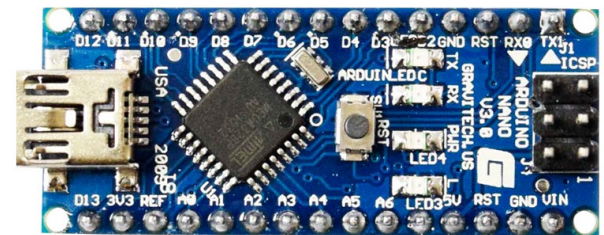
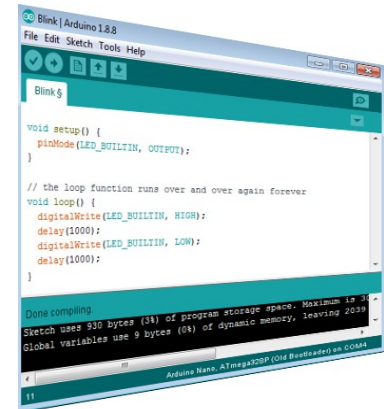
Mi az Arduino?

- Az **Arduino** egy szabad szoftveres, nyílt forráskódú elektronikai fejlesztőplatform, vagy ökoszisztéma az elektronikus eszközök könnyen megtanulható kezeléséhez

- ❖ **Arduino IDE** (integrált fejlesztői környezet): Java alapú, keresztplatformos fejlesztői környezet (szerkesztő, fordító, programletöltő stb.)

- ❖ **Arduino kártya: ATmega328P** vagy más mikrovezérlőn alapuló hardver, amely önállóan vagy a számítógéppel összekapcsolva is működhet

- ❖ **Arduino programnyelv és programkönyvtár-gyűjtemény:** amely lehetővé teszi, hogy a mikrovezérlő részleteinek pontos ismerete nélkül, egyszerűen írassunk programot



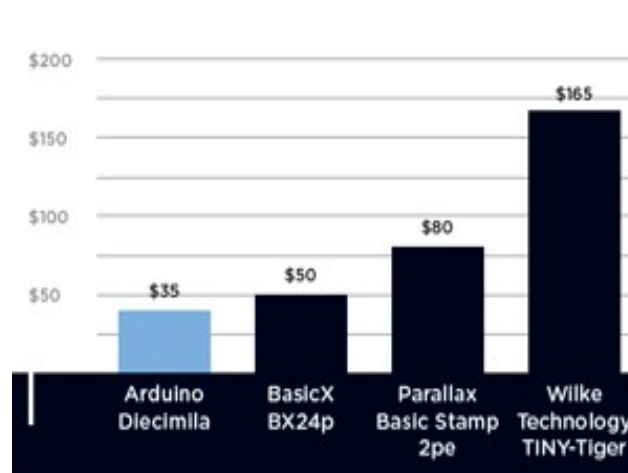
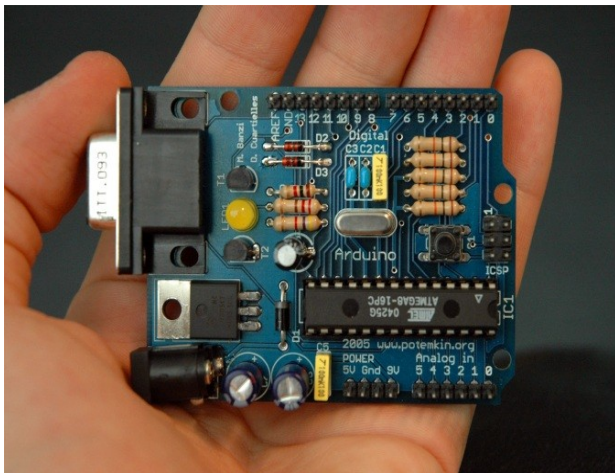
```
void setup() {  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(1000);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(1000);  
}
```

Az Arduino születése

- 2005-ben az olaszországi Ivreában az **Interaction Design Institute** tanárai és diákjai fejlesztették ki.
- Cél: olcsó és egyszerűen használható mikrovezérlős fejlesztőeszköz (hardver és szoftver) létrehozása, amellyel a diákok vagy hobbisták rövid idő alatt (~ 1 hó) interaktív eszközöket tudnak alkotni
- **Előzmények:**

Processing – nyíltforrású programnyelv és IDE (Casey Reas, Benjamin Fry)

Wiring – Nyíltforrású mikrovezérlős fejlesztőkártya és programnyelv (Hernando Barragán)



Gianluca
Martino

Massimo
Banzi

David
Cuertielas

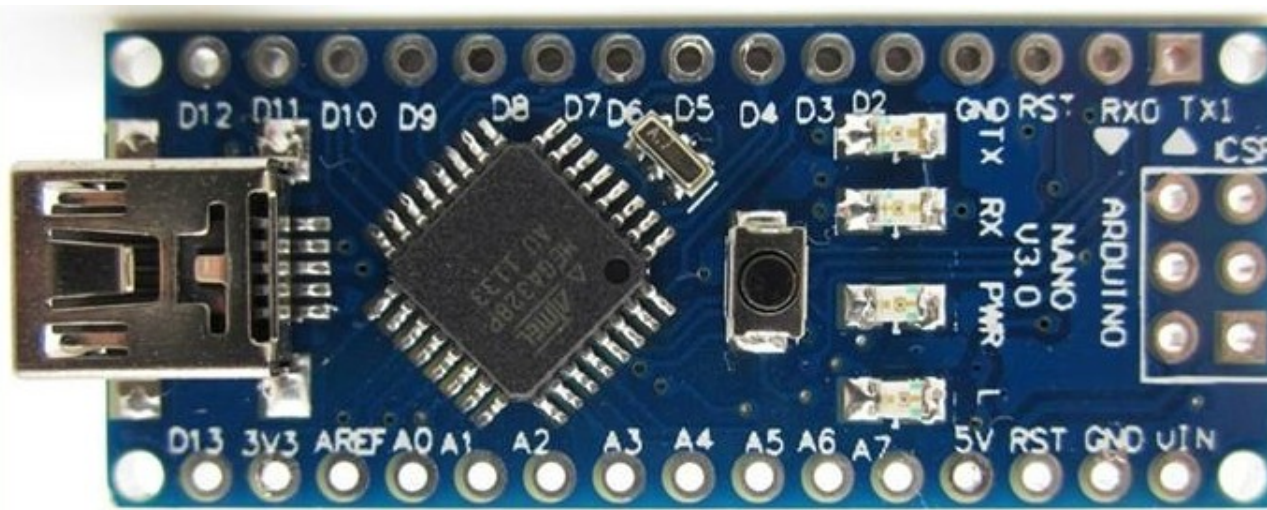
Miért az Arduino?

- Jelenleg ez az egyik legolcsóbban beszerezhető fejlesztőeszköz
- Könnyen használható, ingyenes programfejlesztői környezet
- Világszerte elterjedt, rengeteg mintapélda, programkönyvtár, leírás, tankönyv található hozzá
- Van hozzá többféle szimulátor, közöttük ingyenesek is
- Nagy választékban találunk hozzá olcsó kiegészítőket
 - ❖ Szenzorok
 - ❖ Kommunikációs modulok
 - ❖ Kijelzők
 - ❖ Motorvezérlők
 - ❖ Relé modulok
 - ❖ Robotépítő KIT-ek

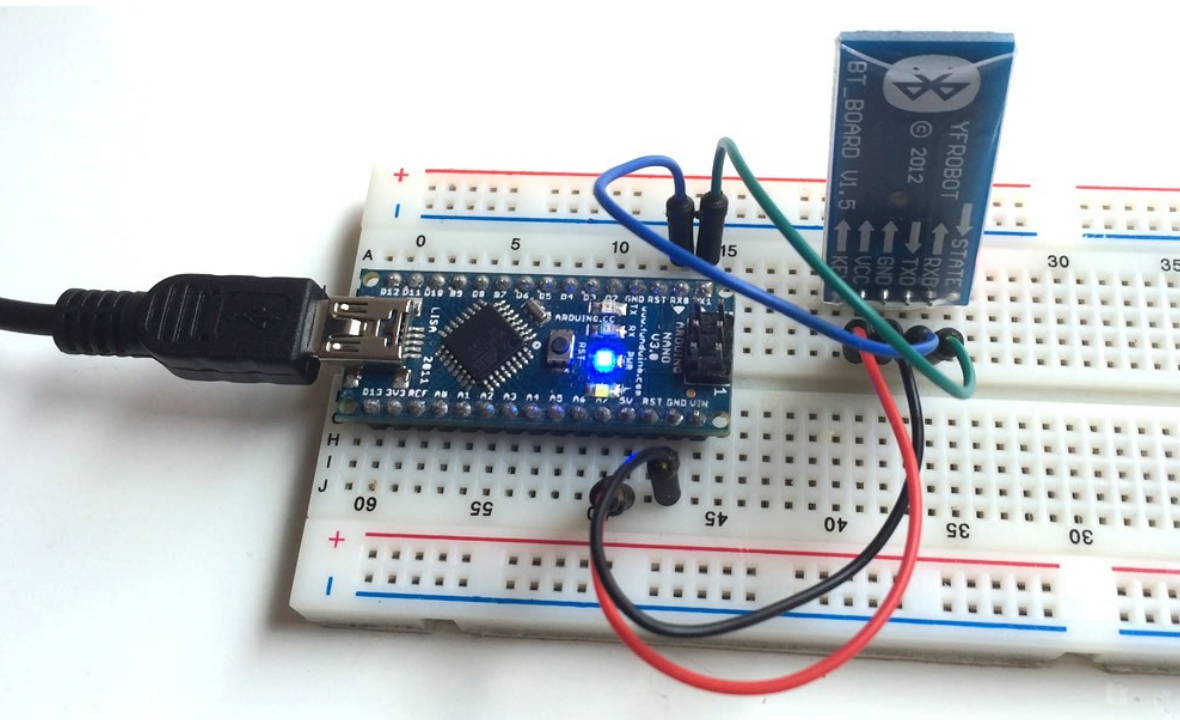


IR Control

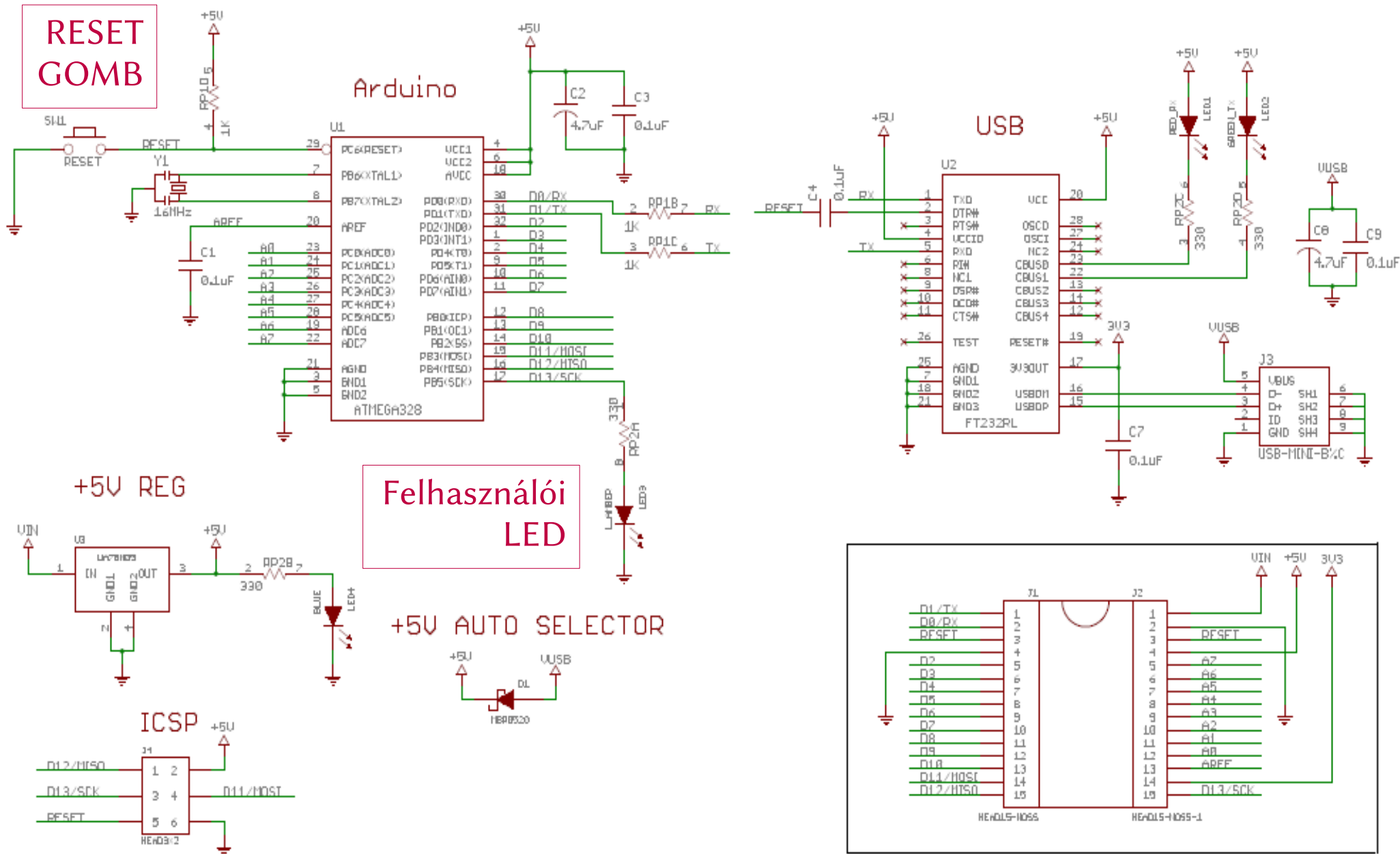
Arduino nano v3.0



- Dugaszolós próbapanelhoz optimális
- Ára kedvező
- A gyári fedlapokhoz csak kiegészítő kártyával használható!



Arduino nano v3.0

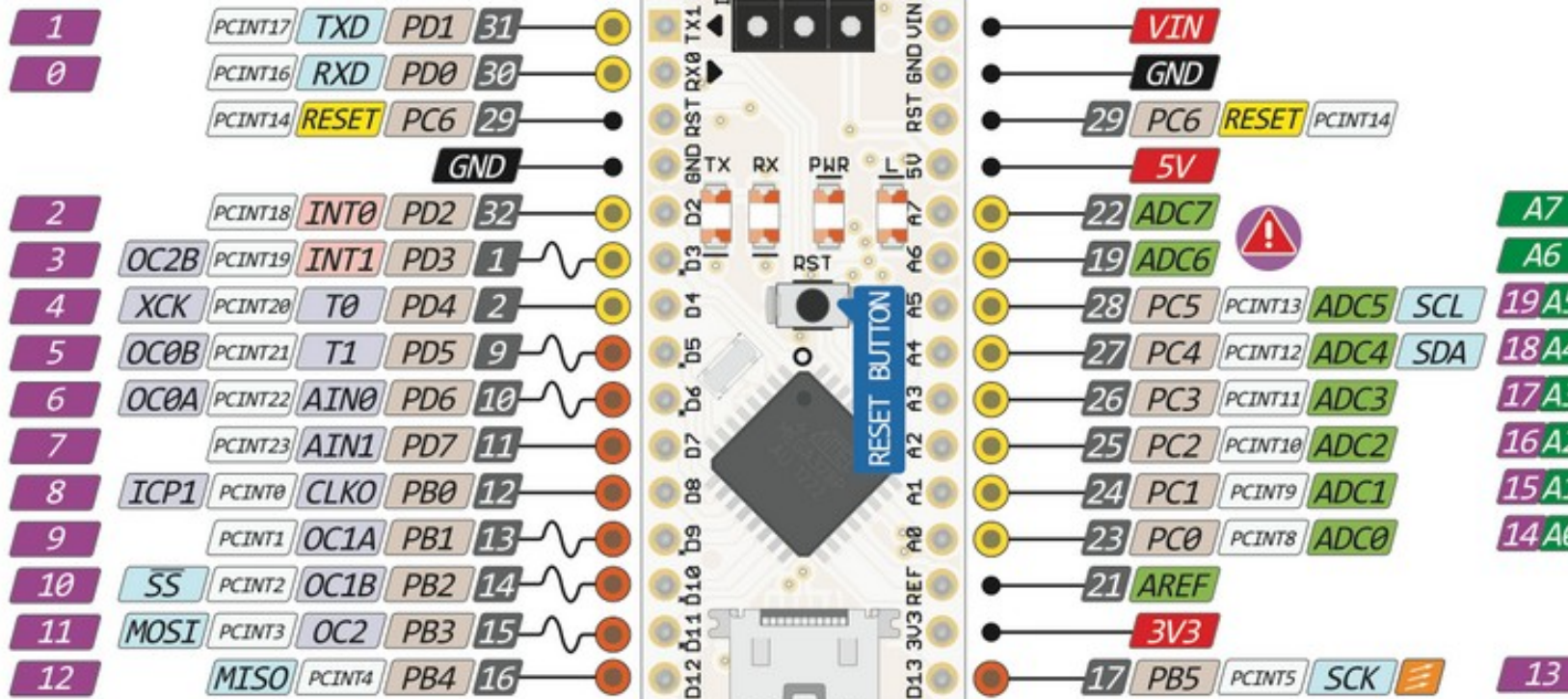
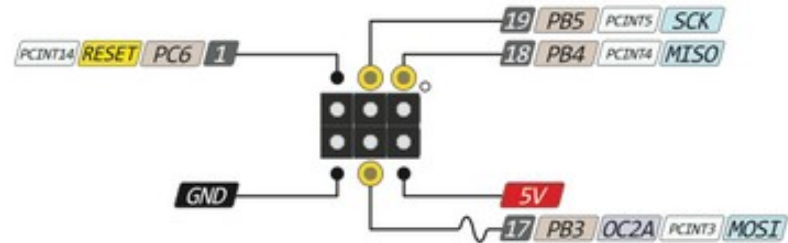


Az Arduino nano kártya kivezetései



NANO PINOUT

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



- Power
- GND
- Serial Pin
- Analog Pin
- Control
- INT
- Physical Pin
- Port Pin
- Pin function
- Interrupt Pin
- PWM Pin
- Port Power

Absolute MAX per pin 40mA recommended 20mA


Absolute MAX 200mA for entire package

Analog exclusively Pins

Az Arduino IDE telepítése

- Az **arduino.cc/en/Main/Software** oldalról töltük le a legfrissebb Arduino kiadást (én Windows 7-hez a ZIP változatot töltöttem le)
- A letöltés és telepítés után a kártyához való meghajtó programot is telepíteni kell (kínai Arduino klón esetén a **CH341SER.EXE** programot kell letölteni és futtatni)

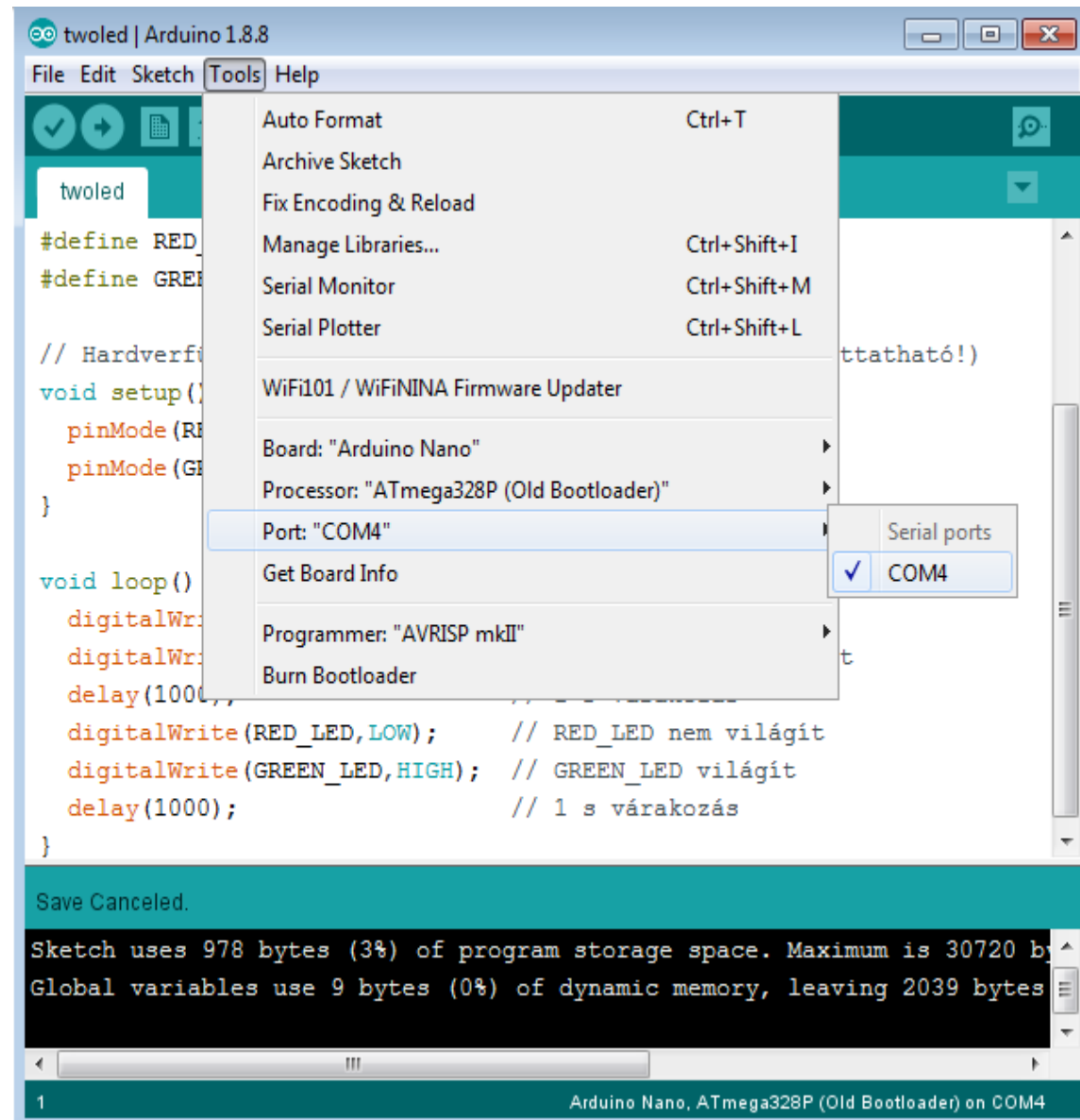
Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a circular logo with a minus sign and a plus sign. To the right of the logo, the text reads: **ARDUINO 1.8.9**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the right side of the page, there is a teal box with a red border containing the following options: "Windows Installer, for Windows XP and up", "Windows ZIP file for non admin install", "Windows app Requires Win 8.1 or 10" with a "Get" button, "Mac OS X 10.8 Mountain Lion or newer", "Linux 32 bits", "Linux 64 bits", "Linux ARM 32 bits", "Linux ARM 64 bits", "Release Notes", "Source Code", and "Checksums (sha512)".

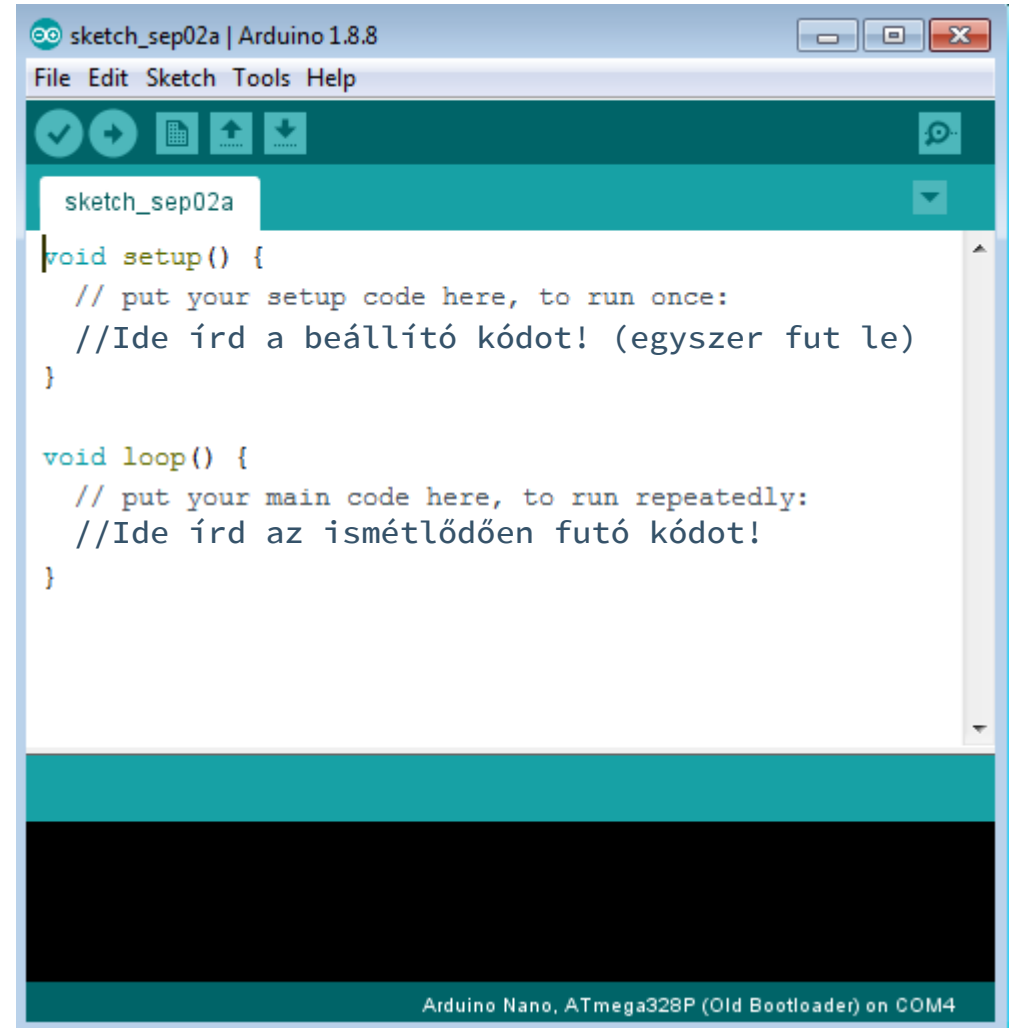
Az Arduino IDE beállítása

- *Tools* menüben *Arduino Nano* választása
- ATmega328(*Old Bootloader*) választása
- A kártyához tartozó virtuális soros portot válasszuk ki



Új program létrehozása

- Az Arduino IDE elindításakor, vagy a **File** → **New** menüpont kattintásával egy új programszerkesztő ablak nyílik
- A programot - Arduino körökben - vázlatnak (*sketch*) hívják
- A „Vázlatfüzet” alapértelmezetten a felhasználó **Dokumentumok/Arduino** mappájában van.
A **File** → **Preferences** menüponban átírható a helye
- Minden program kötelező elmei a **setup()** és a **loop()** függvények



```
sketch_sep02a | Arduino 1.8.8
File Edit Sketch Tools Help
sketch_sep02a
void setup() {
  // put your setup code here, to run once:
  //Ide írd a beállító kódot! (egyszer fut le)
}

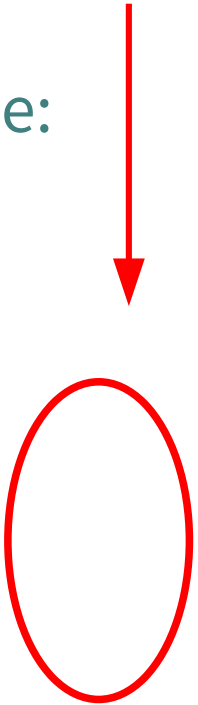
void loop() {
  // put your main code here, to run repeatedly:
  //Ide írd az ismétlődően futó kódot!
}

Arduino Nano, ATmega328P (Old Bootloader) on COM4
```

Az Arduino programok szerkezete

- Minden **Arduino** program kötelező elemei:
 - ❖ **setup()** függvény (előkészítés)
Csak egyszer fut le bekapcsoláskor, vagy újraindításkor
 - ❖ **loop()** függvény (programhurok)
Vég nélkül ismétlődik
- A { } kapcsos zárójelpár a C programnyelvhez hasonlóan blokkba foglalja az utasításokat
- Függvéynév előtt a **void** szó azt jelzi, hogy a függvény nem ad vissza eredményt (eljárás)
- A **setup** és **loop** függvényeknek nincs bemenő paramétere, ezért a () zárójelek közé nem írunk semmit

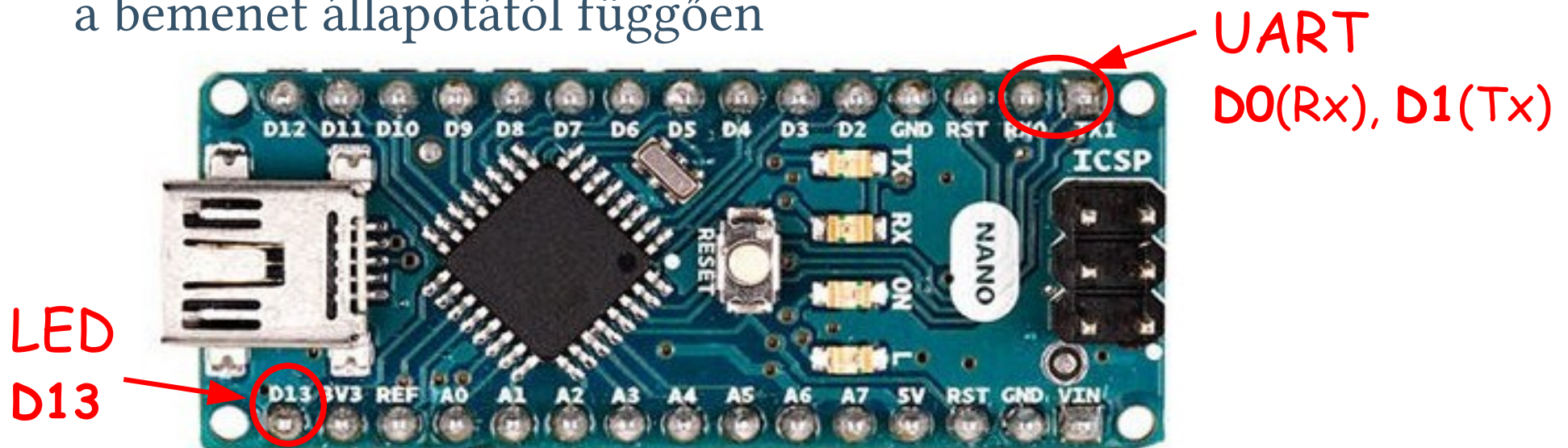
```
void setup() {  
  // csak egyszer futnak le:  
  utasítások;  
}  
  
void loop() {  
  // ismétlődő rész:  
  utasítások;  
}
```



Digitális ki- és bemenetek kezelése

Digitális I/O

- `pinMode(pin, mode)` – kivezetés üzemmódjának beállítása ahol *mode* lehet: *OUTPUT*, *INPUT*, vagy *INPUT_PULLUP*
- `digitalWrite(pin, state)` – kimenetvezérlés, ahol *state* lehet: *LOW*, vagy *HIGH*
- `digitalRead(pin)` – bemenet állapotának lekérdezése, a visszatérési érték **1** („magas”), vagy **0** („alacsony”) lesz, a bemenet állapotától függően



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, belső 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.
- 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)
```

Beállítja a kimenetet

A kivezetés azonosítója

0 – 13, A0 – A5

A kimenet állapota

LOW: alacsony szint („0”)

HIGH: magas szint („1”)

```
int sw = digitalRead(pin)
```

Mintavételezi a bemenet állapotát

- 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.

Példaprogram: LED villogtatás

- Írjuk meg első programunkat, amely a **D13** kivezetésre (*pin, túske*) kötött beépített felhasználói LED-et villogtatja!
- A túske sorszámához rendeljük hozzá a LED nevet!
- Beállításkor a **D13** kivezetés (azaz LED) legyen digitális kimenet!

```
// Example 01 : LED villogtatás
#define LED 13                // LED a D13 digitális kimenetre van kötve

void setup() {
  pinMode( , );              // LED (azaz D13) legyen digitális kimenet
}

void loop() {
  // felkapcsoljuk a LED-et (1, azaz HIGH)
  // várunk egy másodpercet
  // lekapcsoljuk a LED-et (0, azaz LOW)
  // várunk egy másodpercet
}
```

Mit írjunk ide?

Mit írunk ide?

Példaprogram: LED villogtatás

- Írjuk meg első programunkat, amely a D13 kivezetésre (*pin, túske*) kötött beépített felhasználói LED-et villogtatja!
- Állítsuk a LED kimenetet magas szintre, majd várjunk!
- Állítsuk a LED kimenetet alacsony szintre, majd várjunk!

```
// Example 01 : LED villogtatás
#define LED 13 // LED a D13 digitális kimenetre van kötve


void setup() {
  pinMode(LED, OUTPUT); // LED (azaz D13) legyen digitális kimenet
}

void loop() {
  digitalWrite( [ ], [ ] ); // felkapcsoljuk a LED-et (1, azaz HIGH)
  delay( [ ] ); // várunk egy másodpercet
  digitalWrite( [ ], [ ] ); // lekapcsoljuk a LED-et (0, azaz LOW)
  delay( [ ] ); // várunk egy másodpercet
}
```

Mit írjunk ide?

A delay() függvénynek a késleltetési időt ms egységekben kell megadni

Példaprogram: LED villogtatás

- Próbáljuk ki a kész programot (kattintsunk a  gombra)!
- Ha működik a programunk, akkor mentjük el (File → Save As) a vázlatfüzet mappába! Hozzunk létre egy **Lab19_01** nevű mappát (ha még nincs olyan nevű), majd abba mentjük el **blink** névvel!

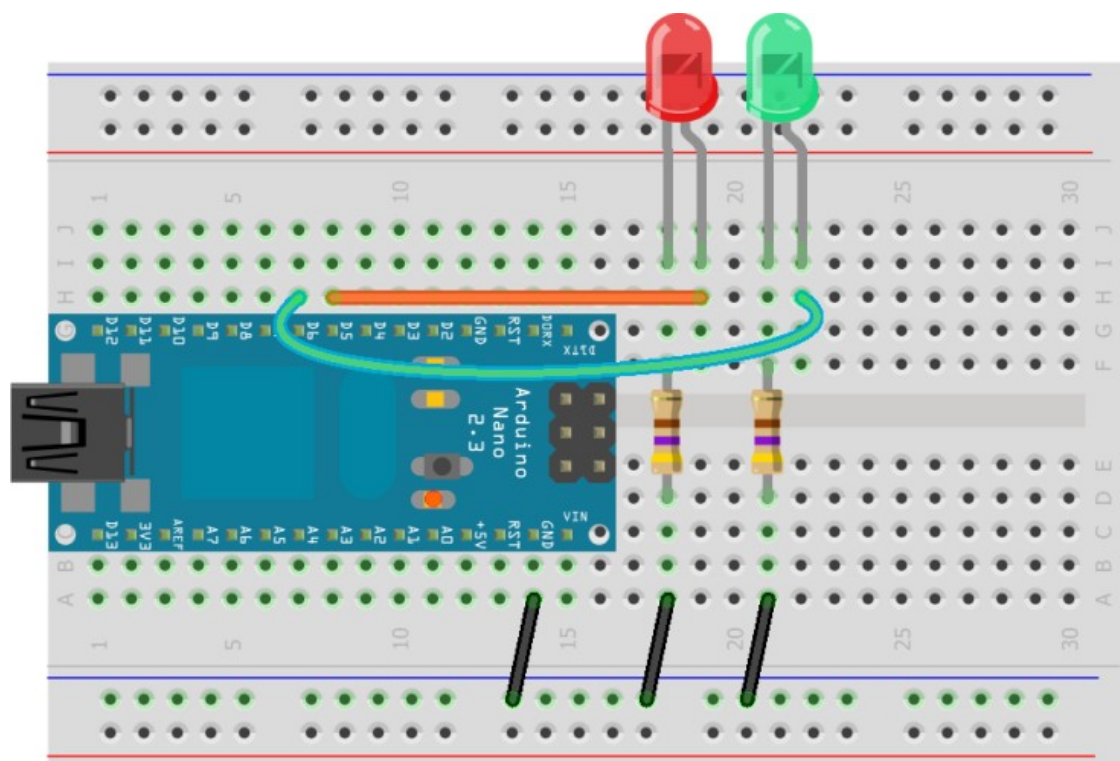
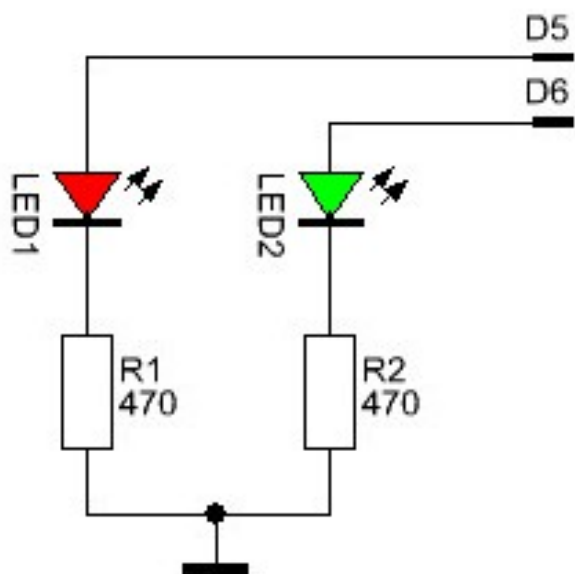
```
// Example 01 : LED villogtatás
#define LED 13                // LED a D13 digitális kimenetre van kötve

void setup() {
  pinMode(LED, OUTPUT);      // LED (azaz D13) legyen digitális kimenet
}

void loop() {
  digitalWrite(LED, HIGH);   // felkapcsoljuk a LED-et (1, azaz HIGH)
  delay(1000);               // várunk egy másodpercet
  digitalWrite(LED, LOW);    // lekapcsoljuk a LED-et (0, azaz LOW)
  delay(1000);               // várunk egy másodpercet
}
```


Két LED-es villogó: twoled.ino

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



Made with  Fritzing.org

Két LED-es villogó: twoled.ino

- Használjuk kreatív módon a Copy-Paste funkciót! (*Ctrl-C*, *Ctrl-V*)

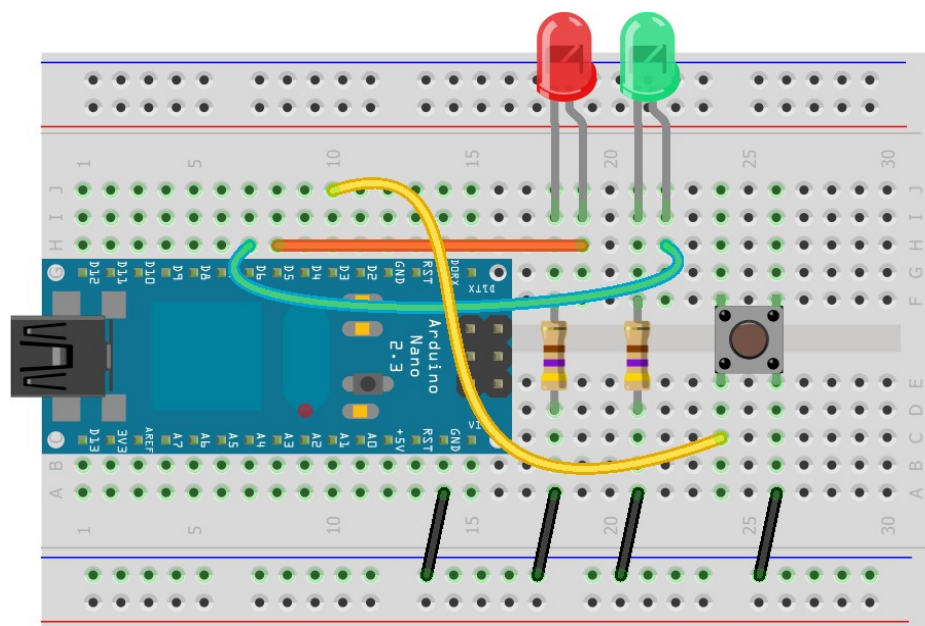
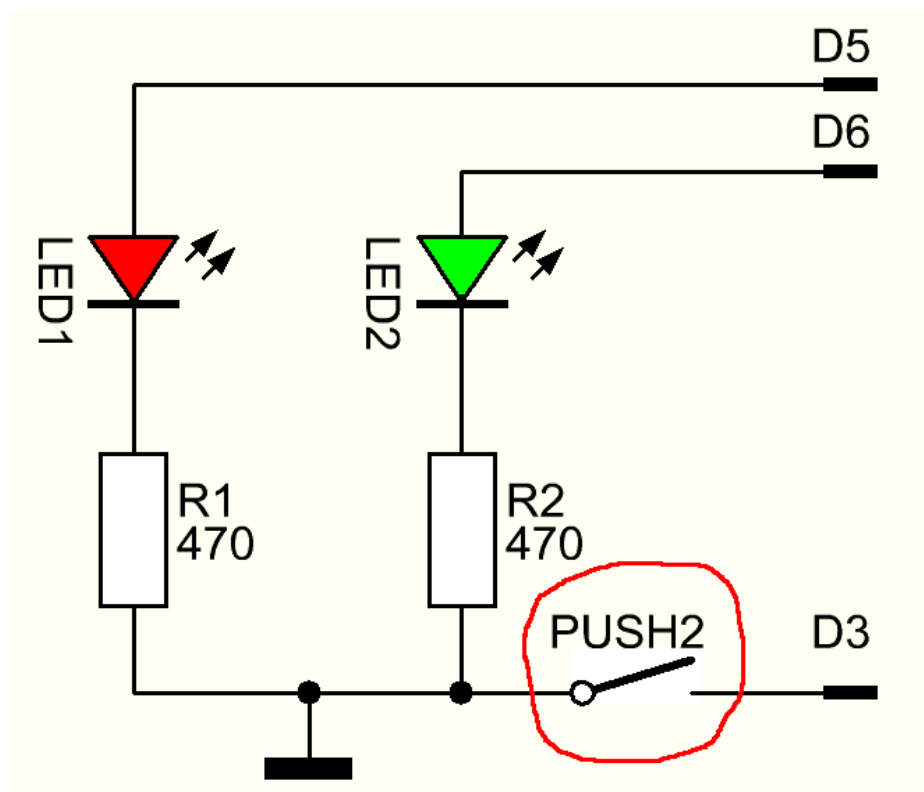
```
#define RED_LED    5
#define GREEN_LED  6

void setup() {
    pinMode(RED_LED,OUTPUT);    //D5 legyen kimenet
    pinMode(GREEN_LED,OUTPUT); //D6 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
}
```

Nyomógomb állapotának beolvasása

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



Made with Fritzing.org

button2led.ino

```
// Hardverfüggő rész Arduino kártyához
#define RED_LED 5
#define GREEN_LED 6
#define PUSH2 3

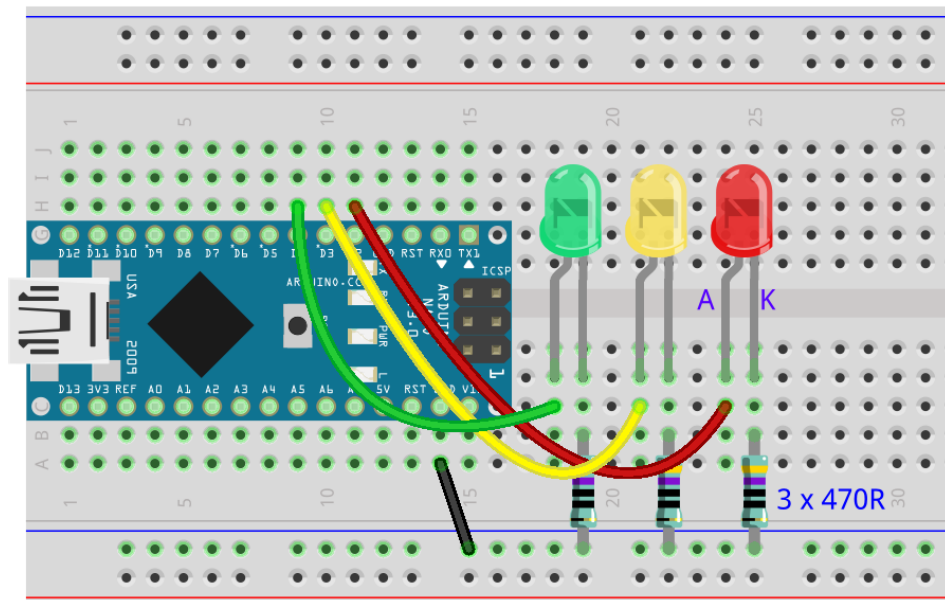
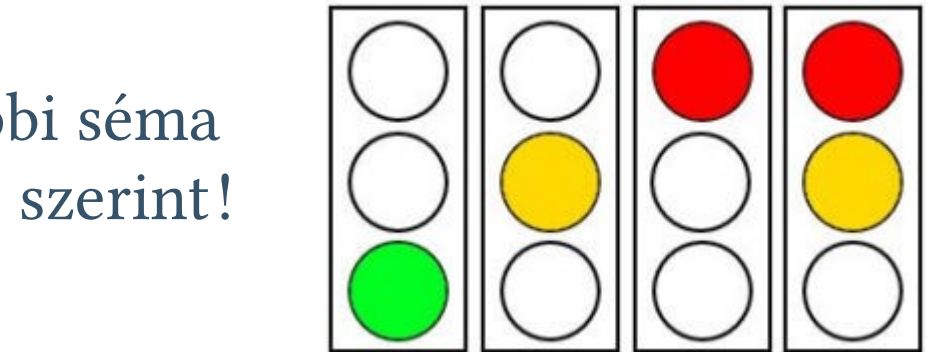
// Hardverfüggetlen rész (MSP430 Launchpad kártyán is futtatható!)
void setup() {
  pinMode(RED_LED,OUTPUT); // legyen kimenet
  pinMode(GREEN_LED,OUTPUT); // legyen kimenet
  pinMode(PUSH2,INPUT_PULLUP); // Bemenet belső felhúzással
}

void loop() {
  digitalWrite(RED_LED,digitalRead(PUSH2)); // világít, ha PUSH2 == HIGH
  digitalWrite(GREEN_LED,!digitalRead(PUSH2)); // világít, ha PUSH2 == LOW
  delay(20); // pergésmentesítő késleltetés
}
```

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

Egyszerű közlekedési lámpa

- Kössük három LED anódját a **D2, D3, D4** kimenetekre, katódjaikat pedig egy-egy áramkorlátozó ellenálláson keresztül a **GND** kivezetésre!
- Kapcsolgassuk a LED-eket az alábbi séma szerint!



traffic_lamp.ino

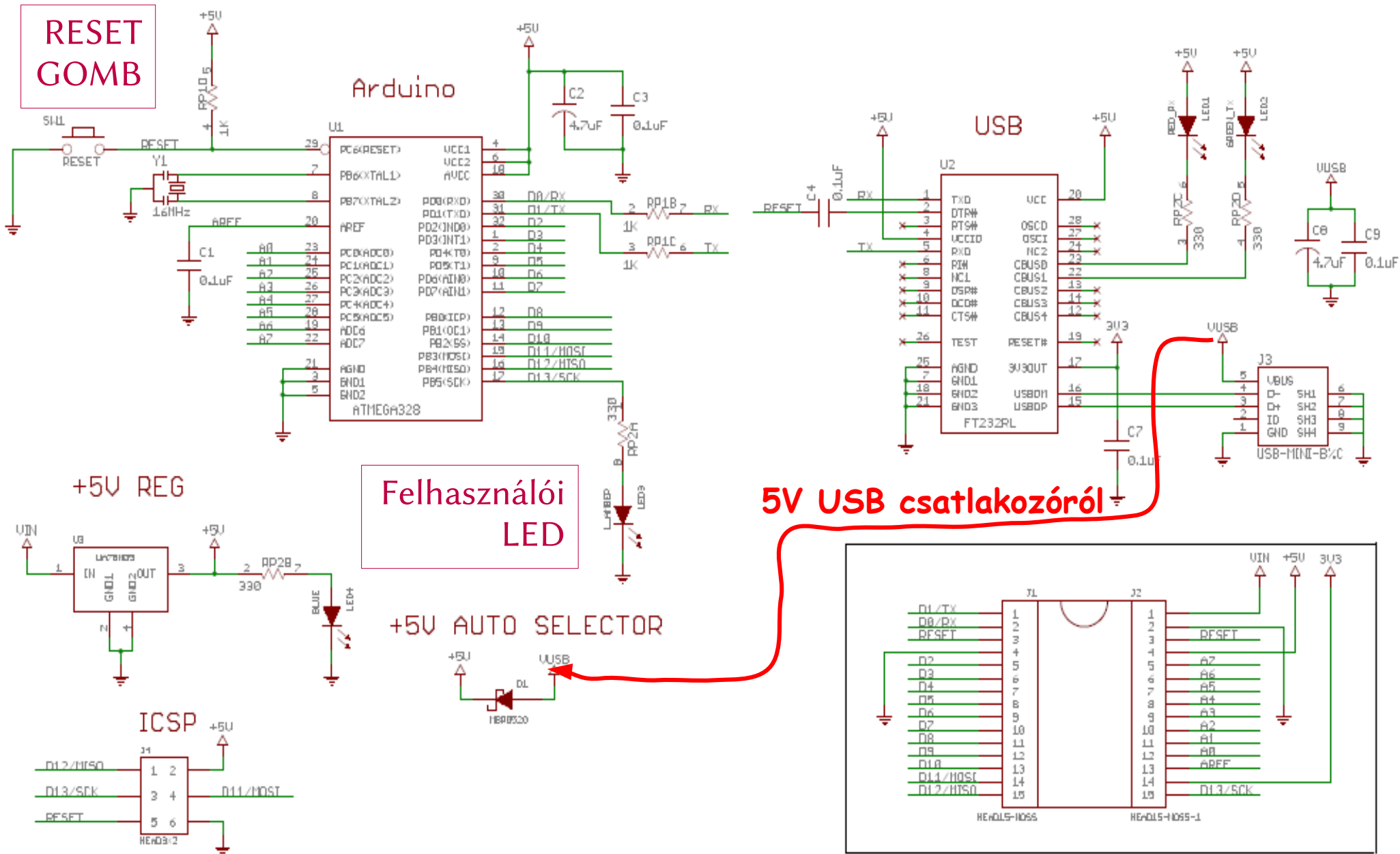
```
#define RED_LED      2           // Piros LED D2-re
#define YELLOW_LED  3           // Sárga LED D3-ra
#define GREEN_LED   4           // Zöld LED D4-re

void setup() {
  pinMode(RED_LED,OUTPUT);      // D2, D3, D4 legyenek digitális kimenetek!
  pinMode(YELLOW_LED,OUTPUT);
  pinMode(GREEN_LED,OUTPUT);
}

void loop() {
  digitalWrite(RED_LED,LOW);    // 1. fázis: zöld
  digitalWrite(YELLOW_LED,LOW);
  digitalWrite(GREEN_LED,HIGH);
  delay(1000);
  digitalWrite(GREEN_LED,LOW);  // 2. fázis: sárga
  digitalWrite(YELLOW_LED,HIGH);
  delay(1000);
  digitalWrite(YELLOW_LED,LOW); // 3. fázis: piros
  digitalWrite(RED_LED,HIGH);
  delay(1000);
  digitalWrite(YELLOW_LED,HIGH); // 4. fázis: piros-sárga
  digitalWrite(RED_LED,HIGH);
  delay(1000);
}
```

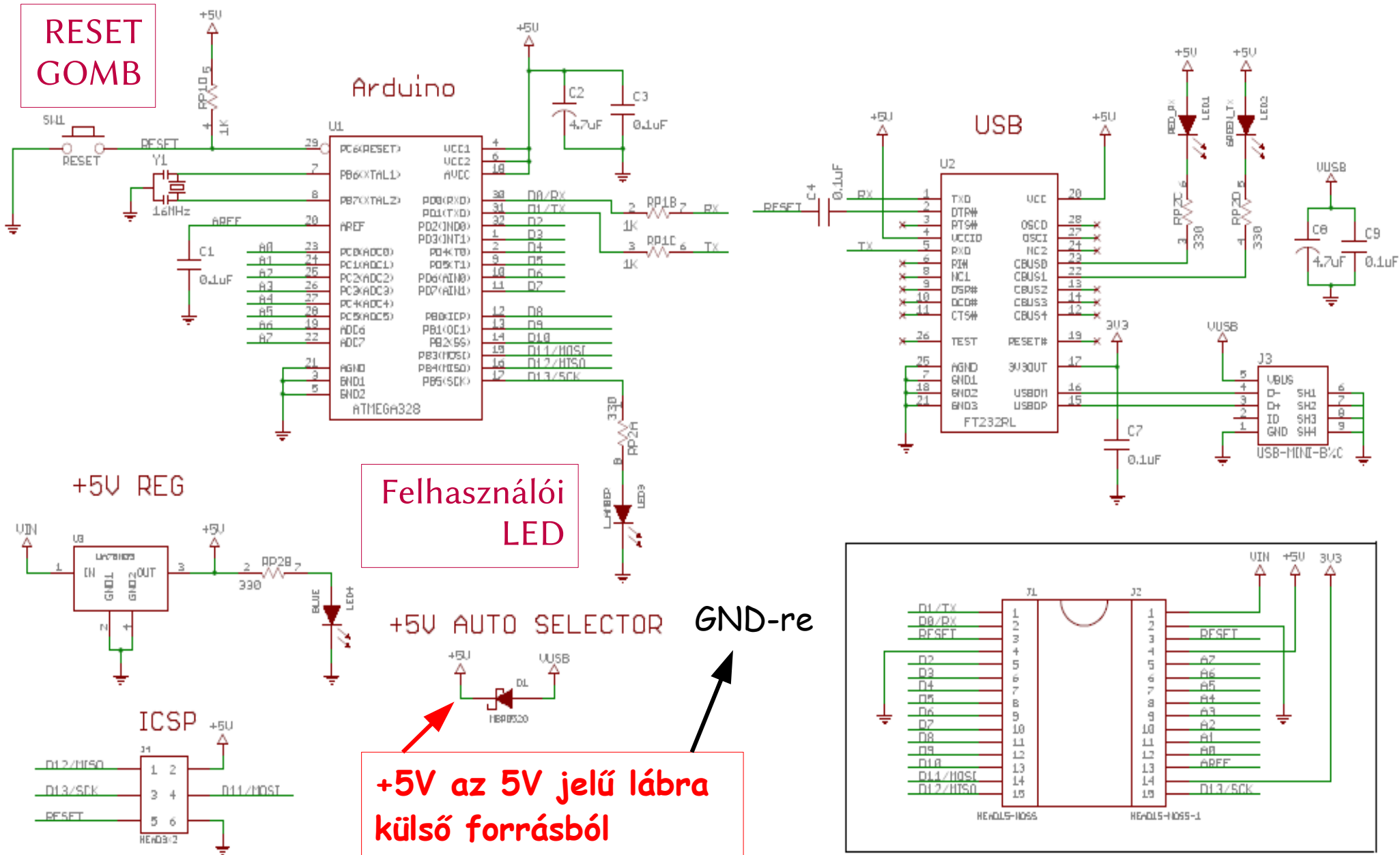
Az időzítéseket a késleltetési idők módosításával állíthatjuk be

Arduino nano tápellátása 1. lehetőség



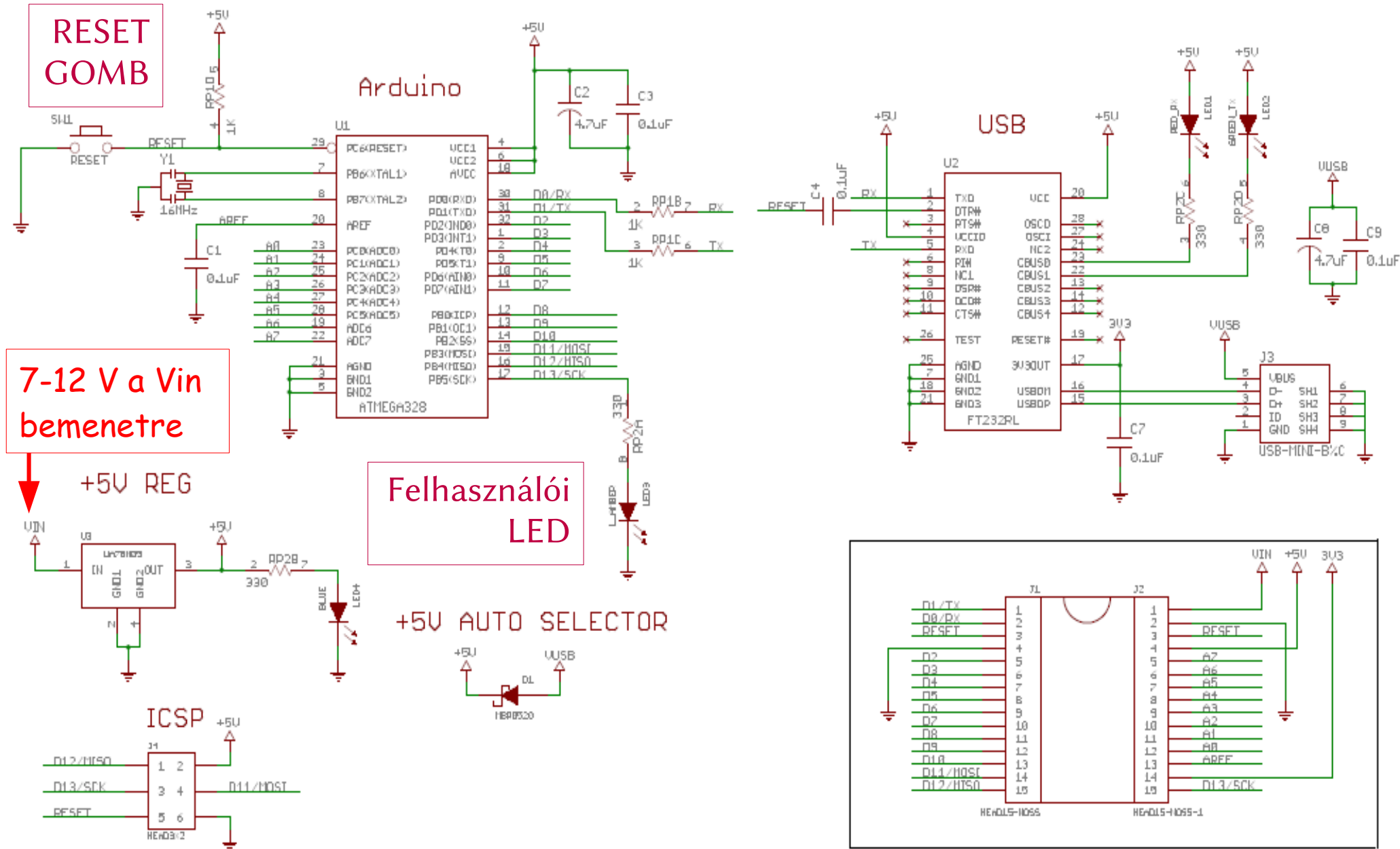
5V USB csatlakozóról

Arduino nano tápellátása 2. lehetőség



**+5V az 5V jelű lábra
külső forrásból**

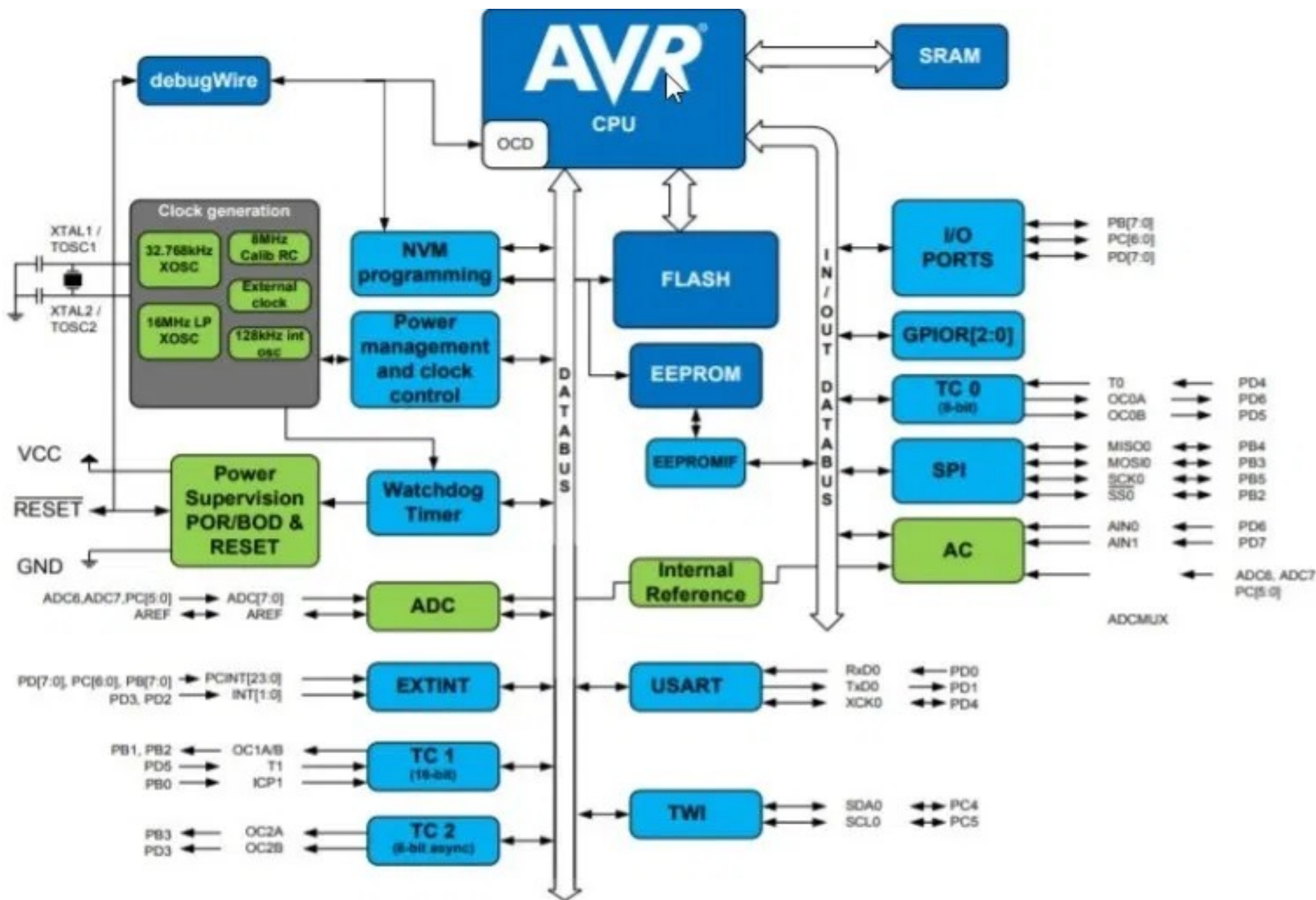
Arduino nano tápellátása 3. lehetőség



ATmega328p mikrovezérlő jellemzők

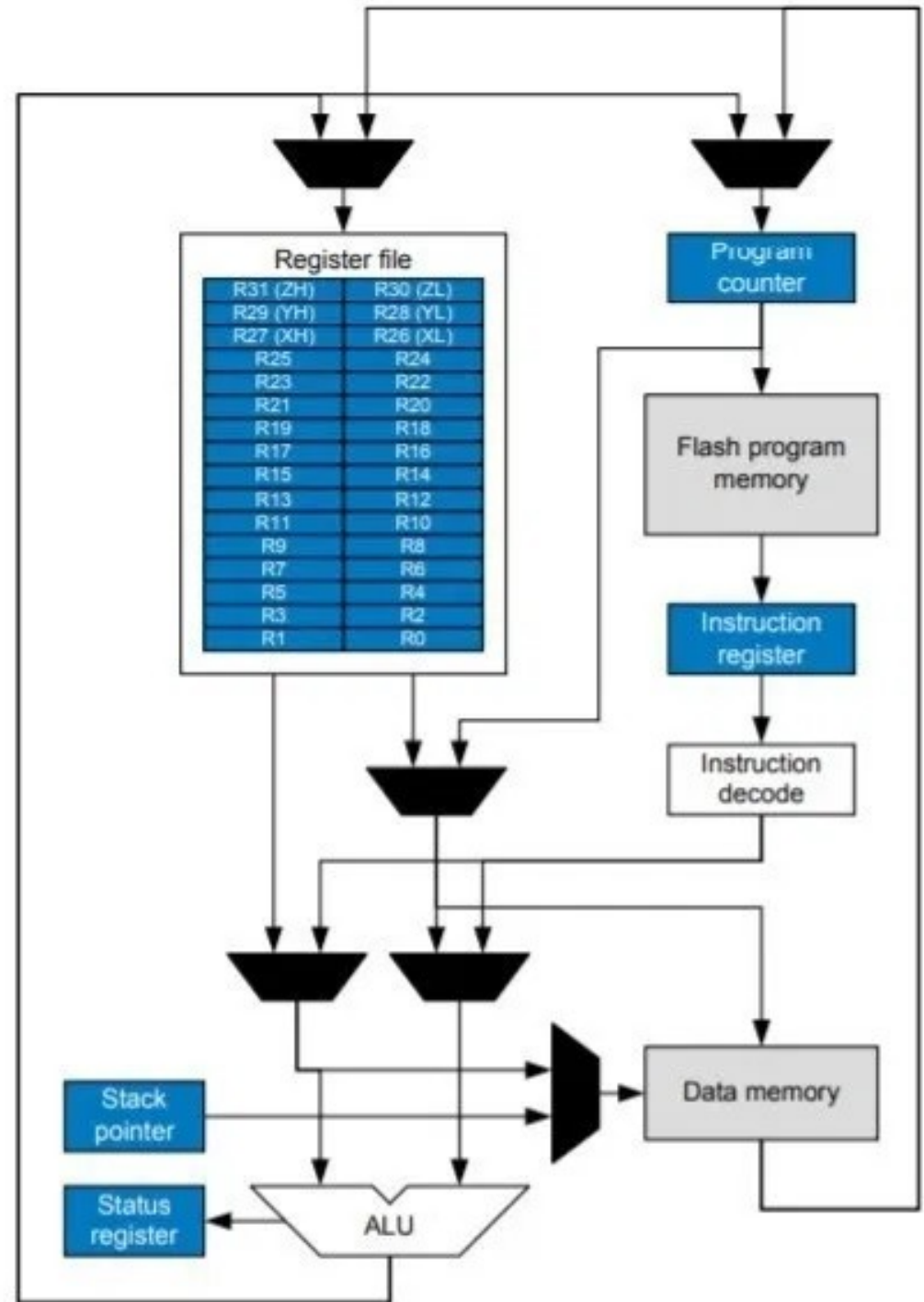
- A 8 bites ATmega328p egy RISC-alapú, Harvard felépítésű mikrovezérlő
- 131-féle utasítás (többségük egy ciklusú)
- 32 x 8 általános felhasználású regiszter
- Statikus működés (bármilyen kis frekvencián működőképes)
- Max. 20MHz órajel (tápfeszültség függő)
- 2-ciklusú szorzó
- Sokszor újraírható memória (flash 10 000, EEPROM 100 000)
- 32 KB Flash, 1 KB EEPROM, 2 KB RAM
- 3 Timer, 6 PWM csatorna, 10 bites ADC, USART, SPI, I2C, WDT, Analog Comparator, 2 interrupt és pin-változásra felébresztés
- Tápfeszültség: 1.8 V – 5.5 V

ATmega328p mikrovezérlő blokkvázlata



A CPU felépítése

- A veremtár az adatmemóriát használja
- A regiszterek közül 6 db páronként összefogható, 3 db 16 bites mutatóként, melyek indirekt címzésre használhatók
- Az ALU aritmetikai, logikai és bitműveletek végzésére szolgál
- A legutóbbi művelet eredményének jelzőbitjeit (I, T, H, S, V, N, Z, C) a Status Regiszter tárolja



Általános célú I/O vezérlése

- MCUCR regiszter PUD bit felhúzás tiltás (1) /engedélyezés (0)

MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS ⁽¹⁾	BODSE ⁽¹⁾	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- DDRB – B port adatirány-váltó regiszter (1: kimenet, 0: bemenet)

DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- PORTB – kimeneti adatregiszter, a beírt adat jelenik meg a kimeneten (bemenet esetén a belső felhúzást kapcsolja be)

PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Általános célú I/O vezérlése

■ PINB – bemeneti adatregiszter

PINB – The Port B Input Pins Address⁽¹⁾

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

- **A PINB regiszter olvasásakor** a B port bemeneteinek pillanatnyi állapotát kapuzza az adatbuszra (beolvasás)
- **A PINB regiszter írásakor** ha 1-et írunk valamelyik bitjébe, akkor a **PORTB** kimeneti adatregiszter megfelelő bitje átbillen (bit toggle), függetlenül az adatáramlási irány (**DDRB**) beállításától.

twoled_fastio.ino

- Két LED-et villogtatunk ellenütemben, közvetlen portkezeléssel
- Ez a program ATmega168, vagy ATmega328 kártyán fut (Arduino UNO, nano, mini...)
- Piros LED a **D5** lábra, zöld LED a **D6** lábra van kötve, ez a **D** port 5. és 6. bitje

```
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
}
```

bit_toggle.ino

- Most a **D13** kivezetésre (*PORTB 5. bit*) kötött beépített LED-et villogtatjuk, közvetlen port eléréssel és a bit toggle módszerrel
- **PORTB** 5. bitjének átbillentéséhez 1-et kell írunk a **PINB** regiszter 5. bitjébe

```
//Kezdeti beállítások
void setup() {
  DDRB |= 0b00100000;    // PORTB 5. bitje legyen kimenet!
  PORTB &= 0b11011111;  // Kezdetben PORTB_5 legyen LOW!
}

// a loop függvény újra és újra ismétlődik a végtelenségig
void loop() {
  PINB |= 0b00100000;    // a LED állapotát átbillentjük (bit toggle)
  delay(1000);           // várunk egy másodpercig
}
```


Ellenállás színkódok

