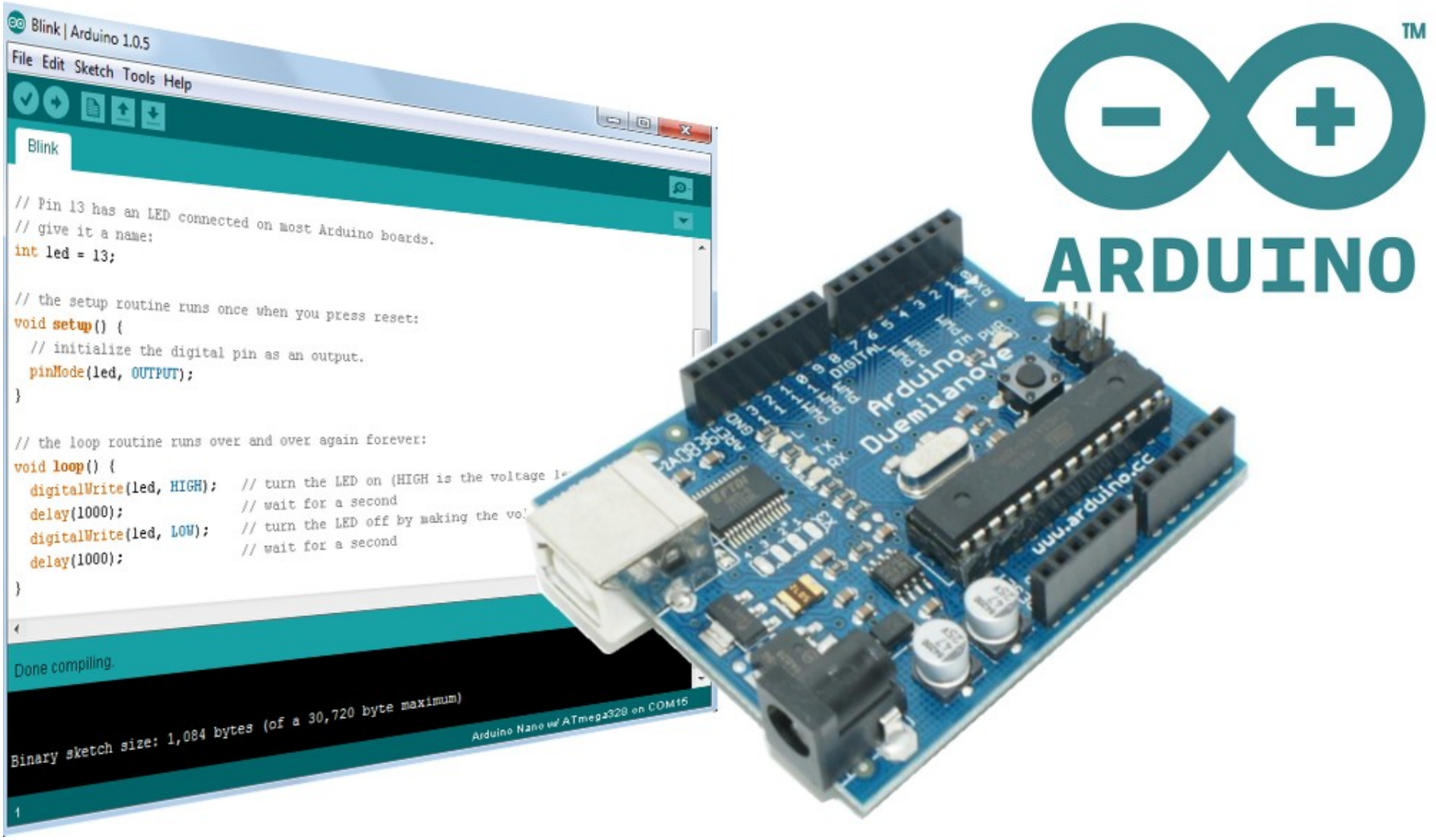
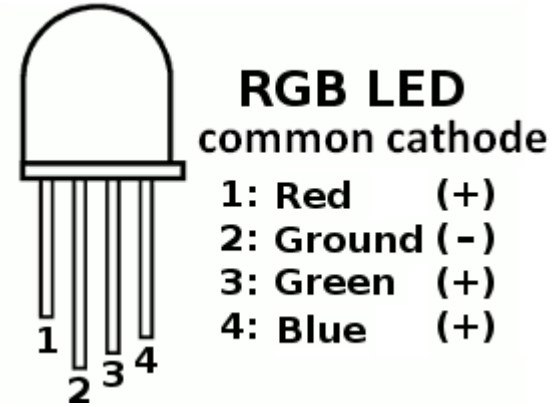
Arduino tanfolyam kezdőknek és haladóknak

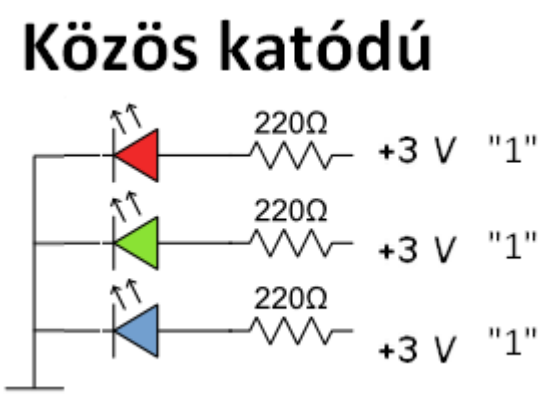


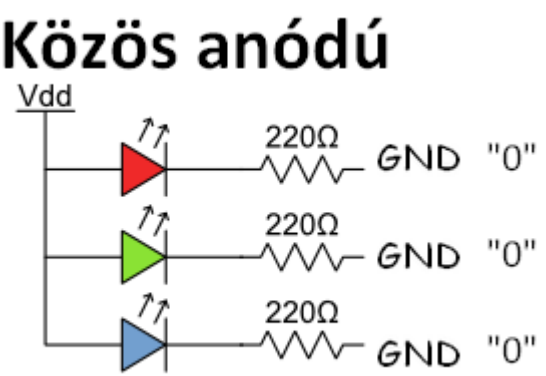
6. Színkeverés RGB LED-del, jelalak vizsgálat

Az RGB LED

- Az RGB LED három, különböző színű LED egy közös tokban. A három szín a három alapszín, amelyből minden más szín kikeverhető (additív színkeveréssel). Az RGB elnevezés az alapszínek angol neveinek kezdőbetűiből áll össze.
  - **R**ed = piros
  - **G**reen = zöld
  - **B**lue = kék

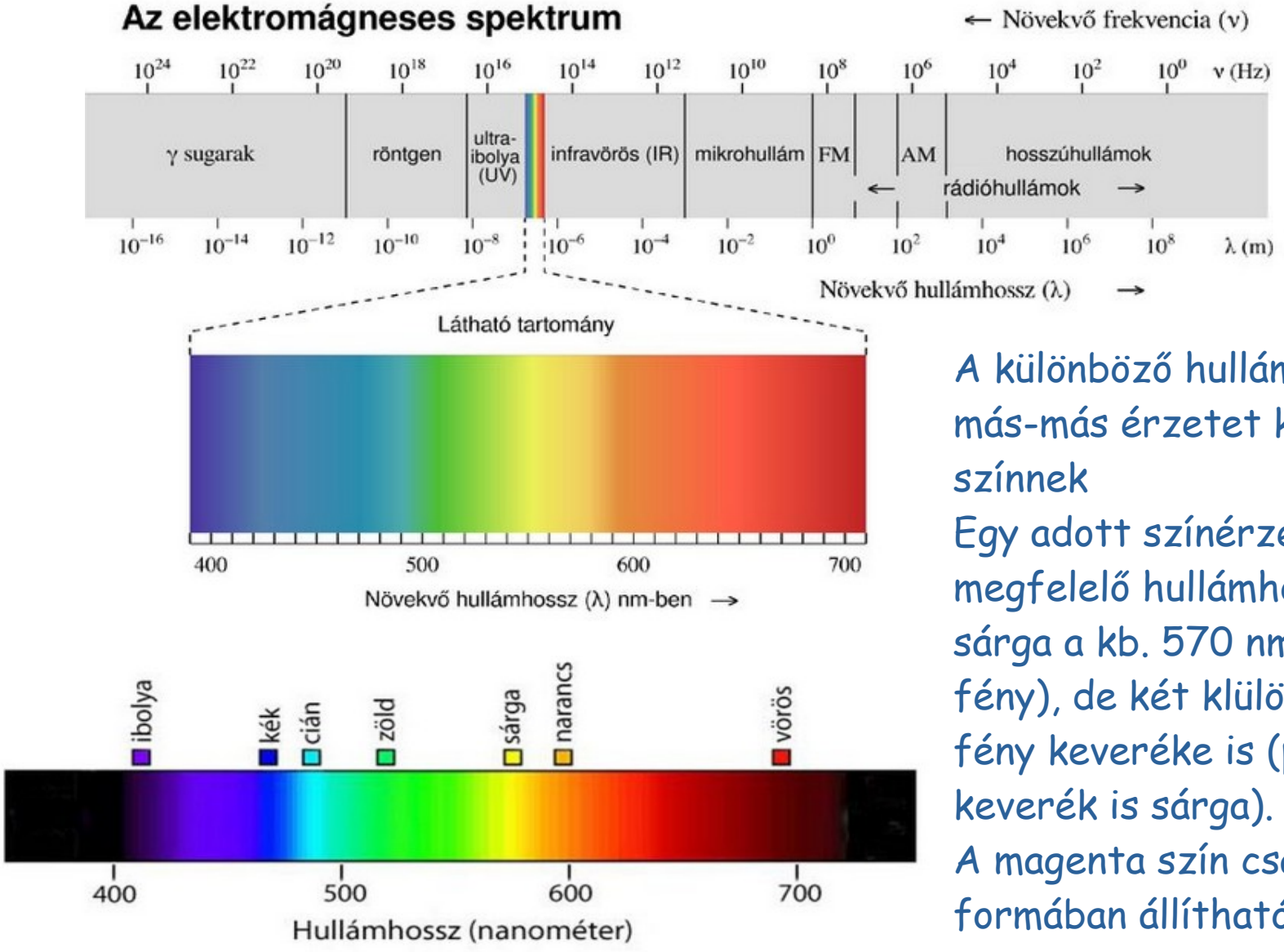






Fény, színek

- Fény: az elektromágneses spektrum szemmel érzékelhető része



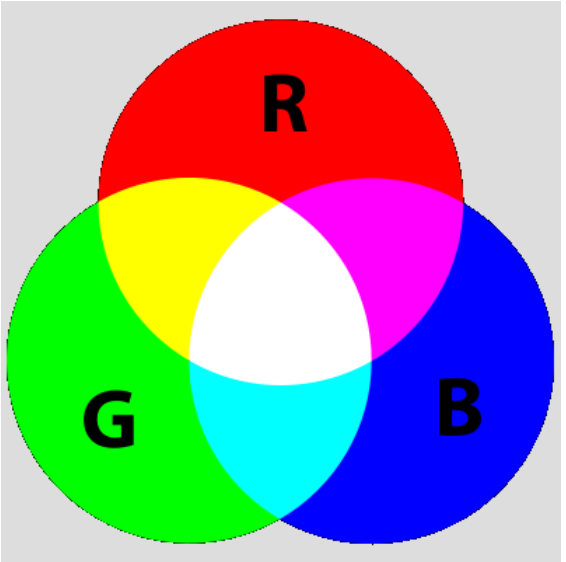
A különböző hullámhosszúságú fény más-más érzetet kelt, ezt nevezzük színnek

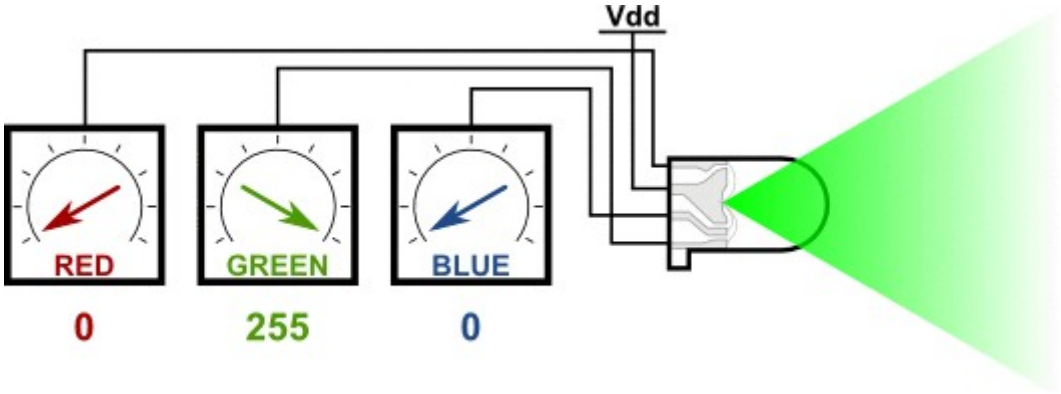
Egy adott színérzetet adhat a megfelelő hullámhosszú sugárzás (pl sárga a kb. 570 nm hullámhosszú fény), de két klülönböző hosszóság fény keveréke is (pl. zöld + piros keverék is sárga).

A magenta szín csak keverék formában állítható elő (piros+kék)

Additív színkeverés elve

- Az additív színkeverés elve:
  a színkomponensek „összeadódnak”
- Ha különböző arányban keverjük az RGB alapszíneket, akkor elvileg bármely színárnyalat kikeverhető
- Fehér: mindegyik alapszín egyenlő arányban szerepel benne
- Fekete: ilyen szín nincs, ez a fény hiánya





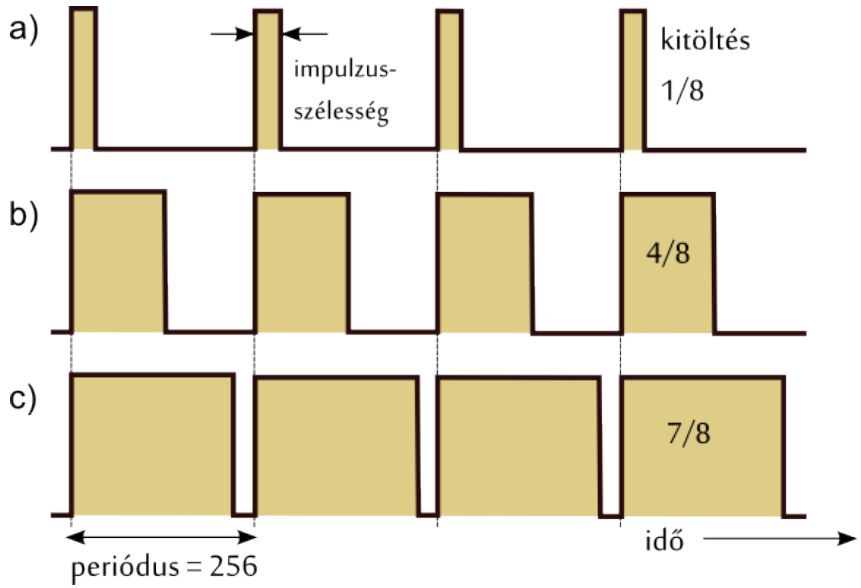
Forrás: http://www.mbeckler.org/microcontrollers/rgb_led/

Emlékeztető: Analóg I/O függvények

- **analogReference(*típus*)** – az analóg bemenetek viszonyítási (referencia) feszültségét konfigurálhatjuk vele
  A választható referencia típusok:
  `DEFAULT` – a tápfeszültség a referencia (5V helyett inkább 4,75 V)
  `INTERNAL` – a belső 1,1 V-os referencia
  `EXTERNAL` – külső forrásból a $\mathbf{V_{ref}}$ lábra adhatunk feszültséget (0-5V)
- **analogRead(*pin*)** – elindít egy mérést a megadott analóg bemeneten (`A0-A7`) és a visszatérési érték a konverzió eredménye lesz (0 – 1023 közötti egész szám)
- **analogWrite(*pin,adat*)** – kiír egy analóg értéket (490 vagy 980 Hz PWM hullám) a megnevezett lábra
  Korlátozások:
  - ATmega328 esetén ***pin*** csak 3, 5, 6, 9, 10, 11 lehet
  - ***adat*** 0 – 255 közötti egész érték lehet

PWM: impulzus-szélesség moduláció

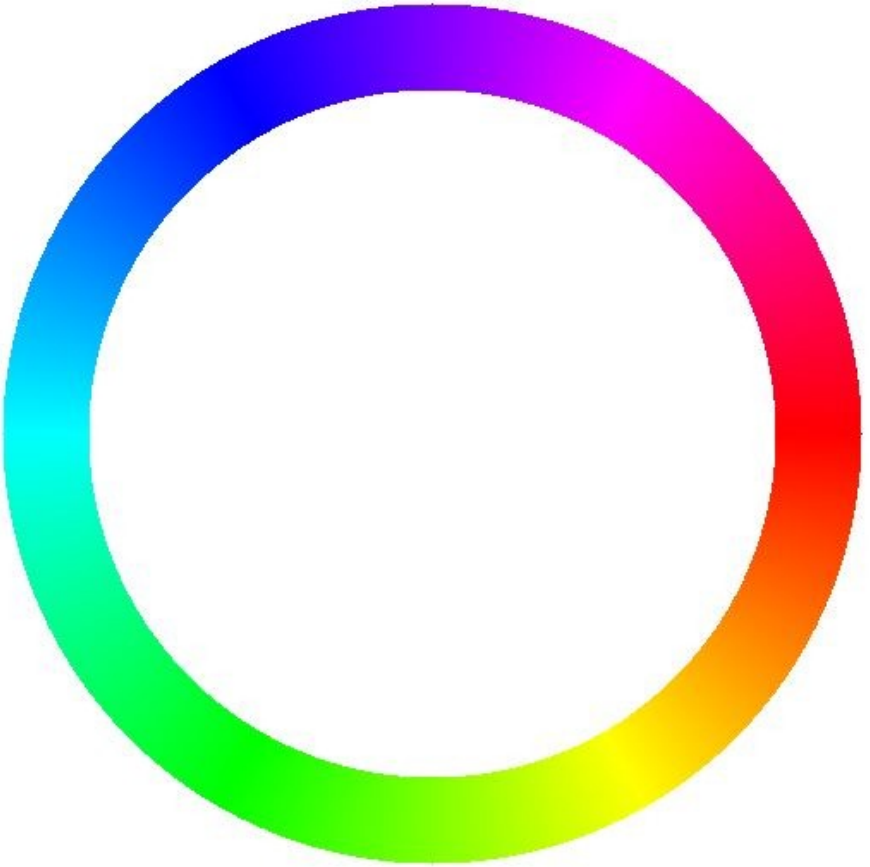
- PWM = pulse width modulation (impulzus-szélesség moduláció)
- A frekvencia állandó
- A kitöltés változtatható 0- 255 között
- Az **`analogWrite()`** függvény csak bizonyos lábakra vonatkozóan használható
- Arduino Uno/Nano (**Atmega 328P**):



Időzítő	Csatorna	Kivezetés	Frekvencia
Timer0	OC0A, OC0B	6, 5	980 Hz
Timer1	OC1A, OC1B	9, 10	490 Hz
Timer2	OC2A, OC2B	11, 3	490 Hz

Automatikus színátmenet

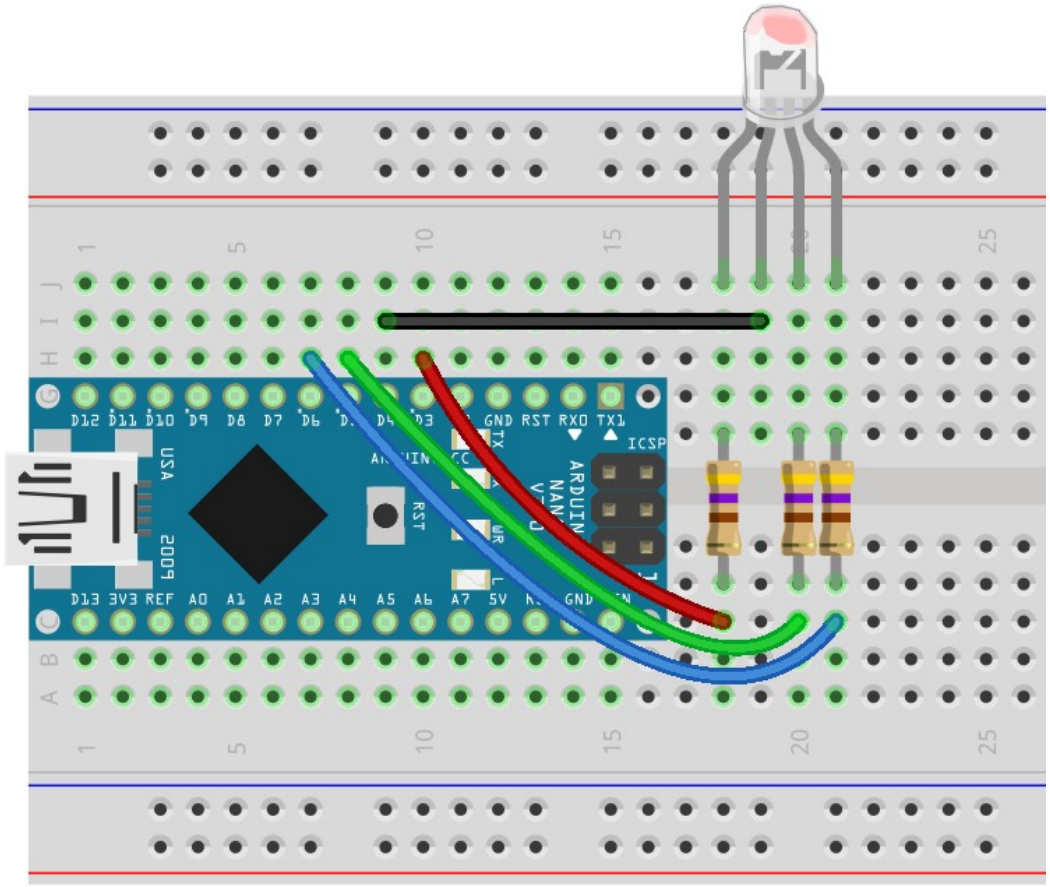
- Állítsuk elő a színkerék folytonos színátmeneteit!
- Három színátmenettel oldhatjuk meg, például:
  - Piros ⟶ kék átmenet
  - Kék ⟶ zöld átmenet
  - Zöld ⟶ piros átmenet
- A leírás és az eredeti program itt található:
  RGB LED Smooth Color Transition



Kapcsolási elrendezés

- **D3** – Red (piros)
- **D4** – közös elektróda
- **D5** – Green (zöld)
- **D6** – Blue (kék)
- Áramkorlátozás: 3 x 470 Ω

- A program testreszabása előtt vizsgáljuk meg a LED-ek polaritását (hogy a katód, vagy az anód a közös)!



RGB_transition.ino

```
#define red    3           // Piros LED = D3
#define common 4           // Közös elektróda = D4
#define green  5           // Zöld LED = D5
#define blue   6           // Kék LED = D6
#define common_level HIGH // Közös elektróda(LOW: katód, HIGH: anód)
byte r = 255, g = 0, b=0;
int t = 25;

void setup() {
  pinMode(common,OUTPUT);
  setRGB();
}

void loop() {
  for (; r >= 0, b < 255; b++, r--) { // Piros → kék átmenet
    setRGB();
  }

  for (; b >= 0, g < 255; g++, b--) { // Kék → zöld átmenet
    setRGB();
  }

  for (; g >= 0, r < 255; r++, g--) { // Zöld → kék átmenet
    setRGB();
  }
}
```

Csonka **for** ciklus, ami inkább **while** ciklusra hasonlít
- Nincs bennük inicializálás, mert globális változókat használunk benne
- Egynél több kilépési feltételt vizsgálunk

RGB_transition.ino

```
void setRGB(void) {
  if (common_level == HIGH) {
    digitalWrite(common, HIGH);
    analogWrite(red,255-r);
    analogWrite(green, 255-g);
    analogWrite(blue, 255-b);
  }
  else {
    digitalWrite(common, LOW);
    analogWrite(red,r);
    analogWrite(green, g);
    analogWrite(blue, b);
  }
  delay(t);      // Várunk egy kicsit ...
}
```

Közös anód esetén

Közös katód esetén

Kézivezérelt színkeverés

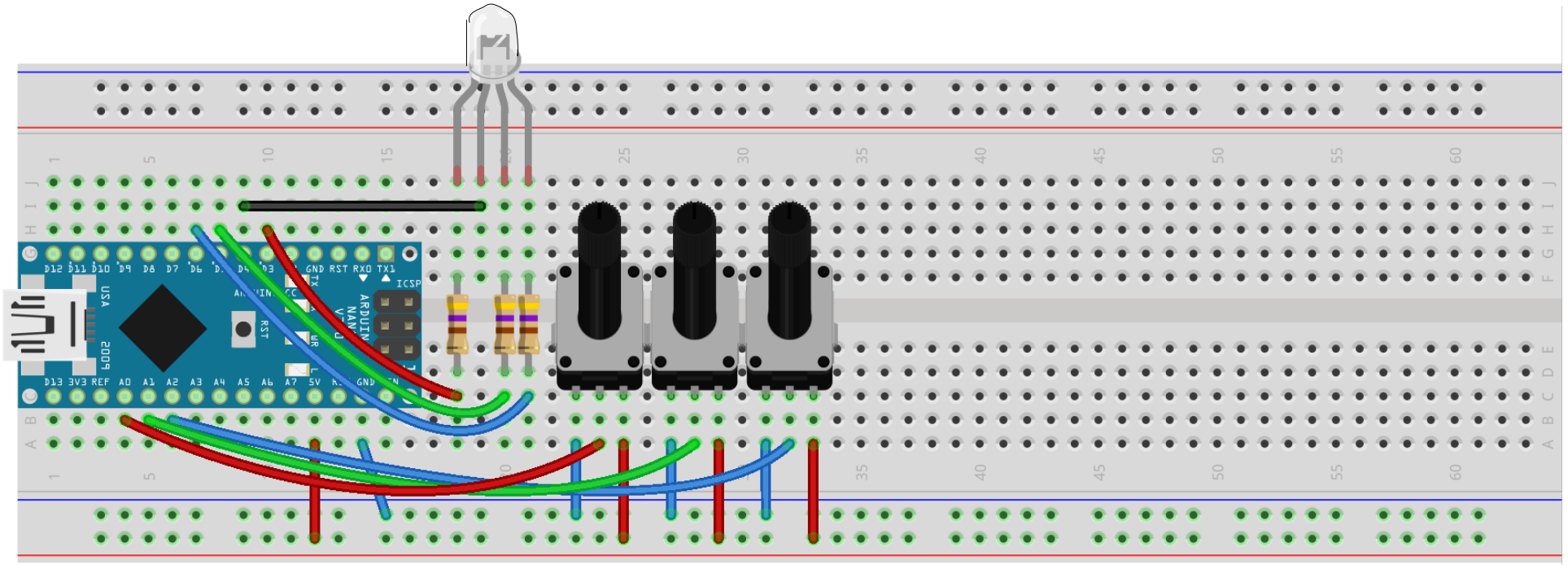
- **Kapcsolási elrendezés:**
- **D3** – Red (piros)
- **D4** – közös elektróda
- **D5** – Green (zöld)
- **D6** – Blue (kék)

A0 – 1. potméter csúszka (piros)

A1 – 2. potméter csúszka(zöld)

A2 – 3. potméter csúszka (kék)

Az áramkorlátozás 3 x 470 Ω



RGBled_pwm.ino

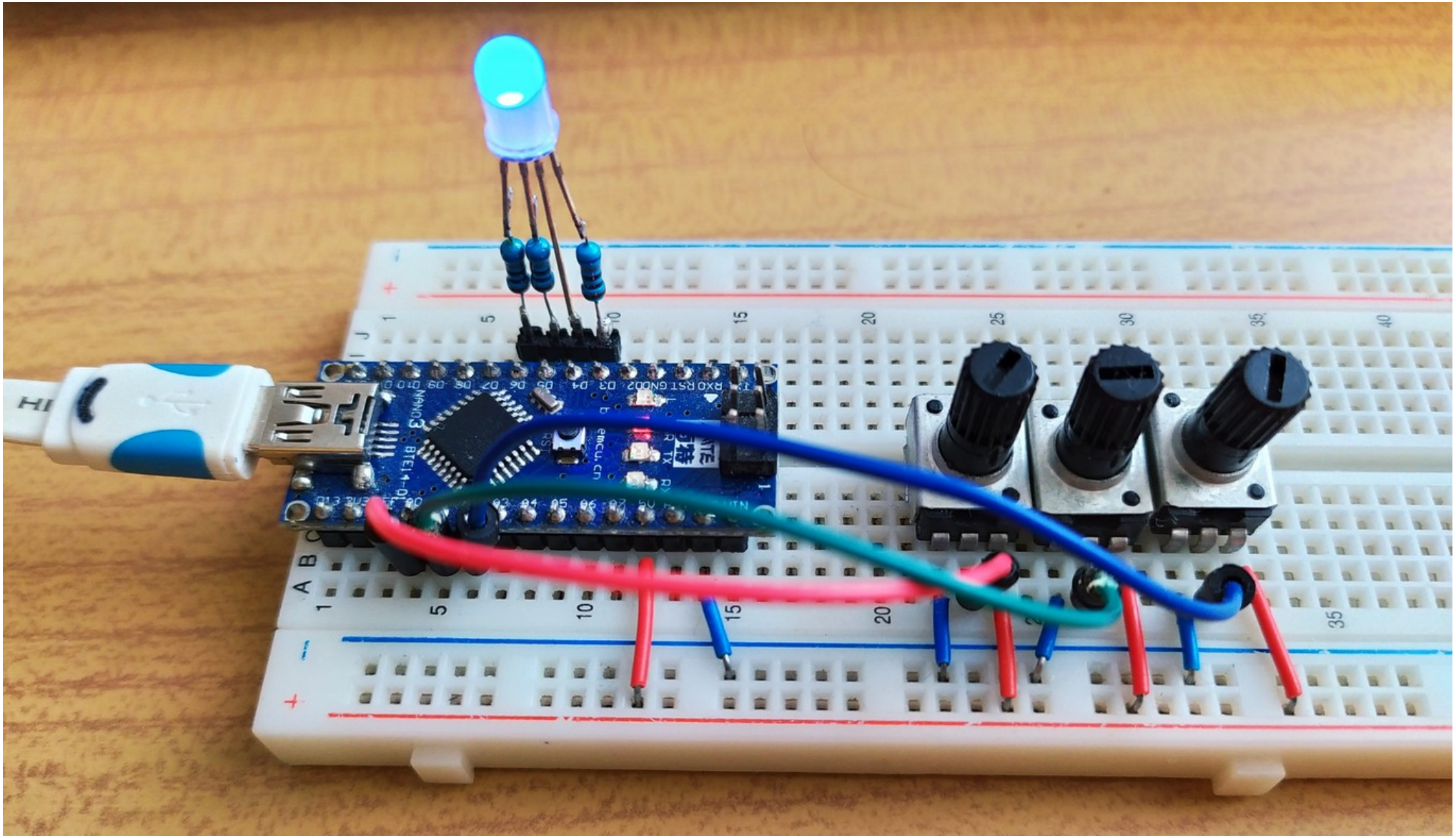
- Az **A0, A1, A2** bemenetekre kötött potméterekkel vezéreljük a **D3, D4, D5, D6** kivezetésekre kötött RGB LED-et *(D4 a közös láb)*

```
#define red    3           // Piros LED = D3
#define common 4           // Közös elektróda = D4
#define green  5           // Zöld LED = D5
#define blue   6           // Kék LED = D6
#define common_level HIGH // Közös elektróda (LOW: katód, HIGH: anód)
```

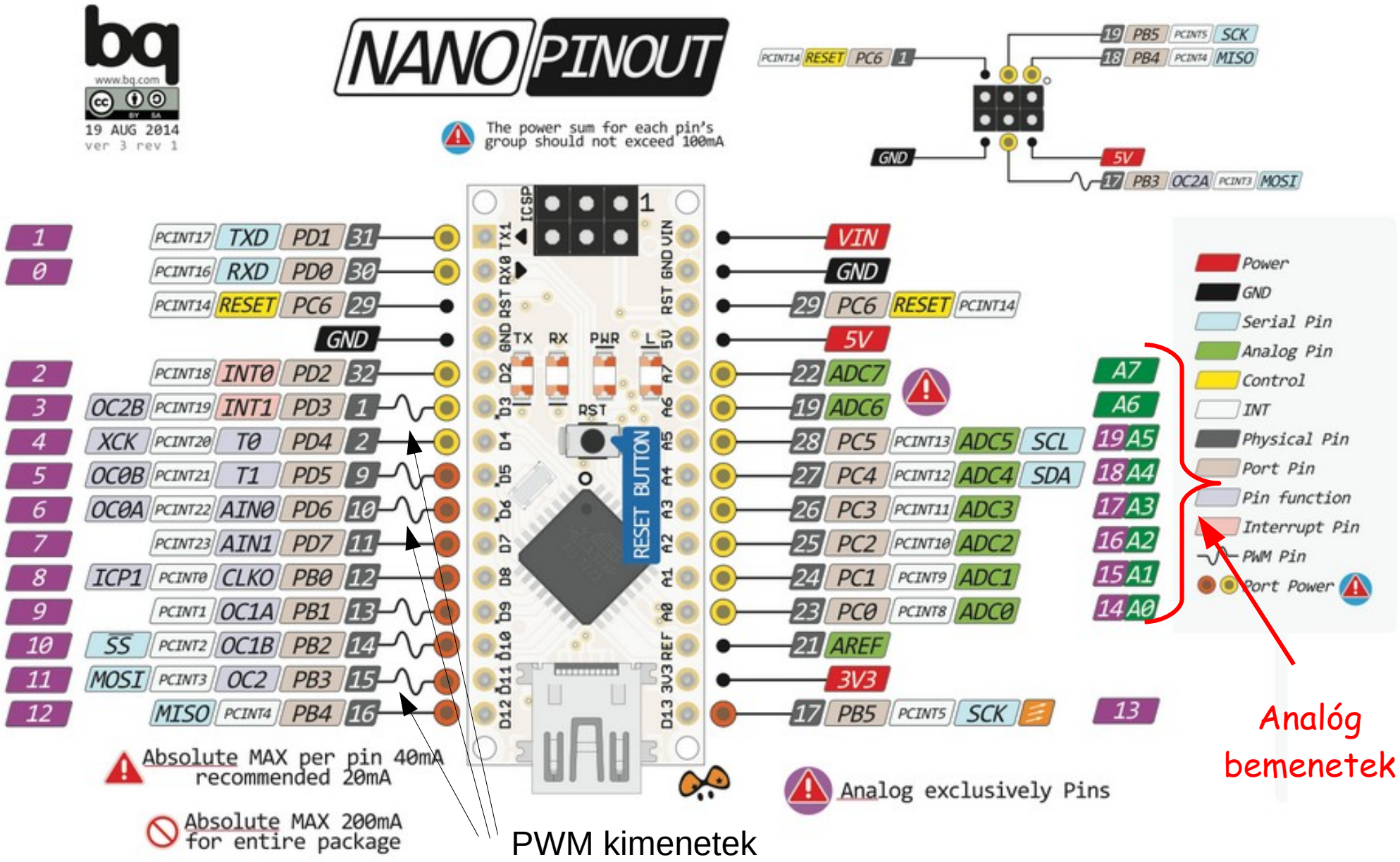
```
byte r = 0, g = 0, b=0;    // Szinkomponensek
int t = 25;                // 25ms késleltetés

void setup() {
  pinMode(common,OUTPUT);
  setRGB();
}

void loop() {
  r = analogRead(A0)>>2;   // 0-255 közés
  g = analogRead(A1)>>2;   // transzformáljuk
  b = analogRead(A2)>>2;   // az értékeket
  setRGB();
}
```

```
void setRGB(void) {
  if (common_level == HIGH) {
    digitalWrite(common, HIGH);
    analogWrite(red,255-r);
    analogWrite(green, 255-g);
    analogWrite(blue, 255-b);
  } else {
    digitalWrite(common, LOW);
    analogWrite(red,r);
    analogWrite(green, g);
    analogWrite(blue, b);
  }
  delay(t);
}
```

A megépített kapcsolás



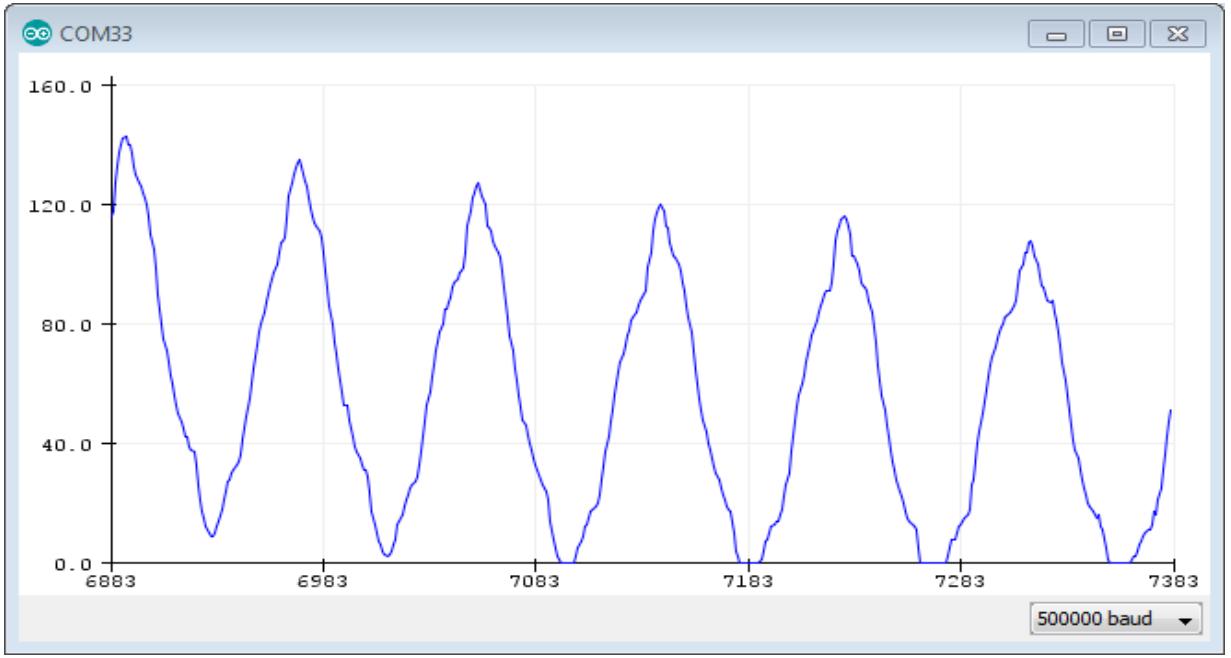
Az Arduino nano kártya kivezetései



Jelalak vizsgálat: hullamforma.ino

- **Lusta módszer:**
  Periodikusan **ADC**-vel megmérjük a bejövő jelet, majd soros porton kiíratjuk az eredményt és a **Serial Plotter** segítségével megjelenítjük
- A soros kapcsolat sebességét állítsuk nagyra (250 000, vagy 500 000 bps), hogy ne korlátozza a sebességet!
- Az alábbi példán az 50 Hz-es zaj látható
- Konklúzió: túl lassú!

```
void setup() {
  analogReference(DEFAULT);
  Serial.begin(500000);
}

void loop() {
  int a = analogRead(A0);
  Serial.println(a);
}
```



Gyorsabb mintavételezés

- Gyorsabb, és egyenlő időközönként végzett mintavételezéshez használjuk a **Timer1** időzítőt és azzal indítsunk **ADC** konverziót!
- Az **ADC** kiolvasását és az eredmény eltárolását az **ADC EOC** (End of Conversion – konverzió vége) megszakításban végezhetjük
- Sajnos, valós idejű megjelenítéshez túl lassú a soros átvitel, ezért tárolós oszcilloszkóp módjára adott számú (256 vagy 512) mintát rögzítünk, majd ebből állóképet jelenítünk meg s Serial Plotteren
- Az **ADC** és **Timer1** regiszter szintű kezeléshez az előző két előadásban található leírásokra lesz szükség, de sok hasznos információ és mintapélda található itt is:
  https://www.gammon.com.au/adc

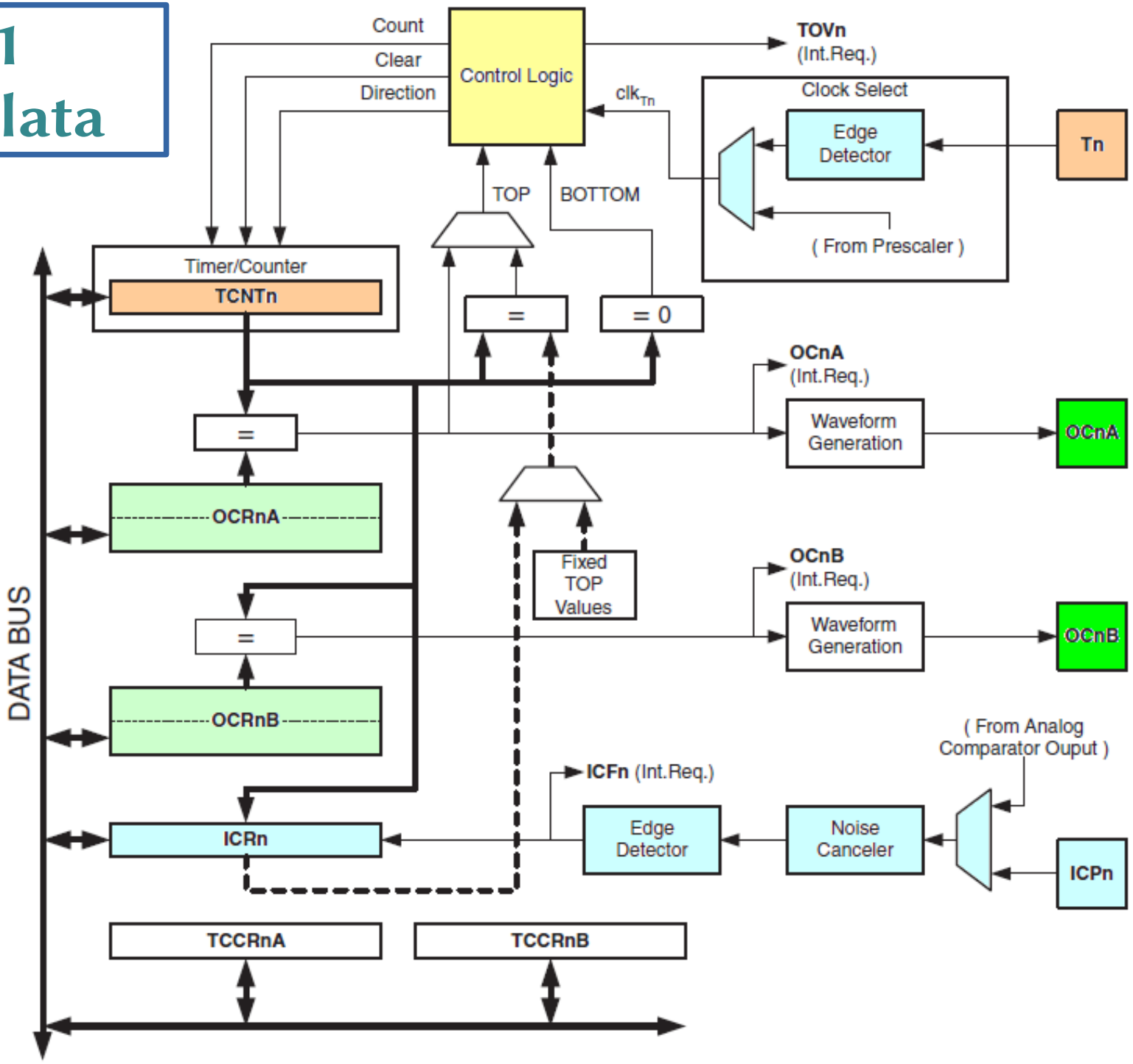
Arduino időzítők/számlálók

- Az ATmega328 mikrovezérlő három időzítő/számlálóval rendelkezik
- Egyik számláló sem szabad, ha átírjuk a regisztereket, akkor az adott időzítőhöz kapcsolódó függvényeket nem használhatjuk (például **Timer2** beállításainak módosítása után nem használhatjuk a **tone()** függvényt)
- Az időzítők órajele lehet belső (a rendszer órajel, vagy annak leosztott jele (/8, /64, /246, /1024 leosztás választható), vagy külső jel (csak Timer0 és Timer1 esetén)

Időzítő	Bitek	Csatorna	Kivezetés	Frekvencia	Funkció
Timer0	8-bit	OC0A, OC0B	6, 5	980 Hz	millis()
Timer1	16-bit	OC1A, OC1B	9, 10	490 Hz	Servo library
Timer2	8-bit	OC2A, OC2B	11, 3	490 Hz	tone()

Timer1 blokkvázlata

Timer0 és Timer2 felépítése is hasonló, de van néhány eltérés:
- 8 bitesek
- nincs bemeneti jelfogás (ICRn)
- Timer2 esetén nincs független Tn bemenet



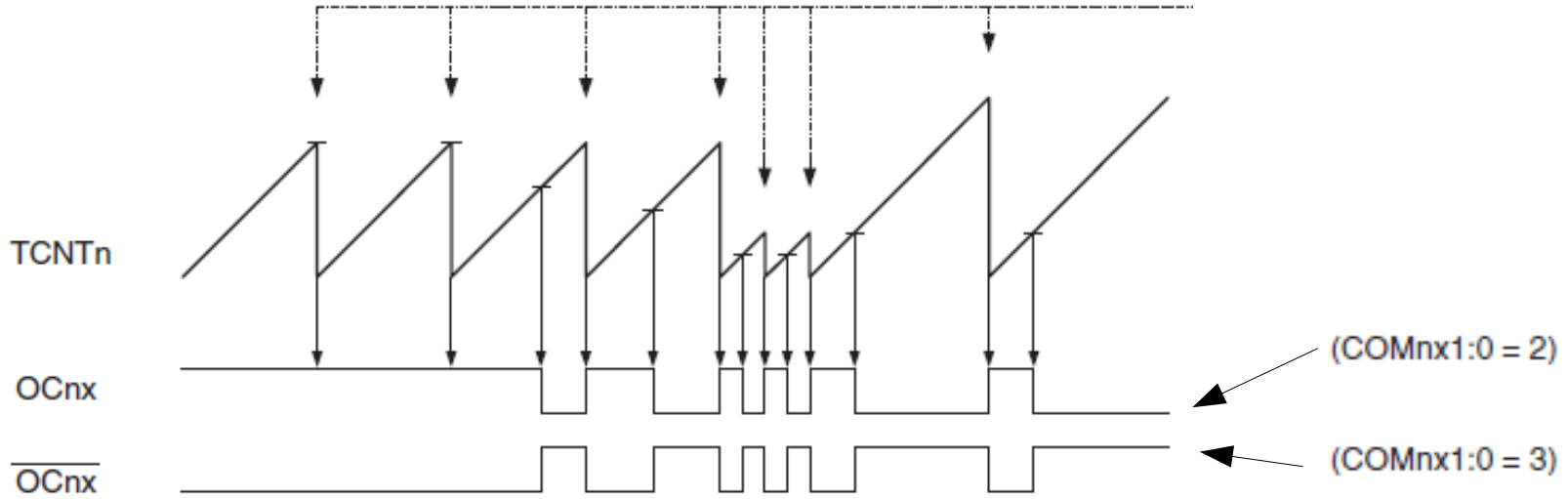
Timer1 regiszterek

- **TCCR1A** – Timer/Counter1 vezérlő regiszter *A*

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
	0	0	0	0	0	0	0	0	

Table 16-1. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on Compare Match.
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level).
1	1	Set OC1A/OC1B on Compare Match (Set output to high level).



Timer1 regiszterek

- **TCCR1B** – Timer/Counter1 vezérlő regiszter ***B***

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
	0	0	0	0	1	0	1	0	

- **ICNC1** – Input capture zajelnyomás engedélyezés (**0**: ki, **1**: be)
- **CES1** – Input capture aktív él kiválasztás (**0**: lefutó, **1**: felfutó)

CS12	CS11	CS10	Description	
0	0	0	No clock source (Timer/Counter stopped).	
0	0	1	$clk_{I/O}$/1 (No prescaling)	16 MHz
0	1	0	$clk_{I/O}$/8 (From prescaler)	2 MHz
0	1	1	$clk_{I/O}$/64 (From prescaler)	250 kHz
1	0	0	$clk_{I/O}$/256 (From prescaler)	62.5 kHz
1	0	1	$clk_{I/O}$/1024 (From prescaler)	15 625 Hz
1	1	0	External clock source on T1 pin. Clock on falling edge.	
1	1	1	External clock source on T1 pin. Clock on rising edge.	

Timer1 regiszterek

- Hullámforma generátor üzemmód bitek: megszabják a számlálási sorrendet, a maximum értéket és a generált hullámformát

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Timer1 regiszterek

TCCR1C – Timer/Counter1 Control Register C

Bit	7	6	5	4	3	2	1	0	
(0x82)	FOC1A	FOC1B	–	–	–	–	–	–	TCCR1C
Read/Write	R/W	R/W	R	R	R	R	R	R	
	0	0	0	0	0	0	0	0	

- **FOC1A, FOC1B** – force output compare A, B (csak nem PWM mód esetén hatásos, s a bit 1-be állítása azonnali egyezési eseményt jelez a hullámforma generátornak

TCNT1H and TCNT1L – Timer/Counter1

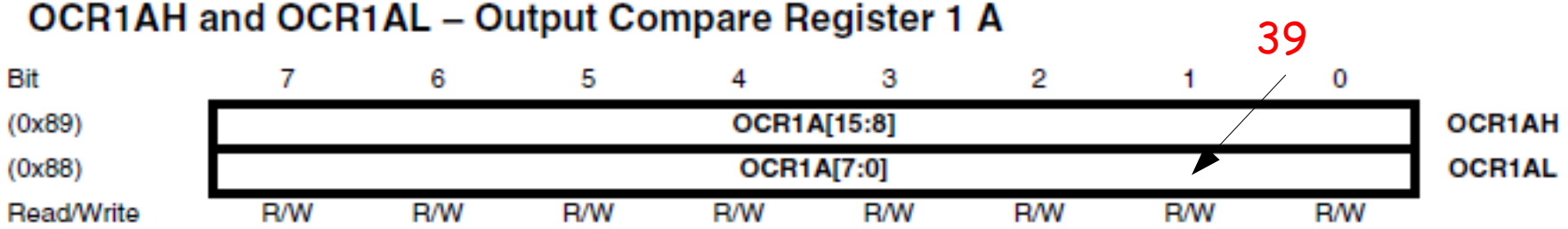
Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

ICR1H and ICR1L – Input Capture Register 1

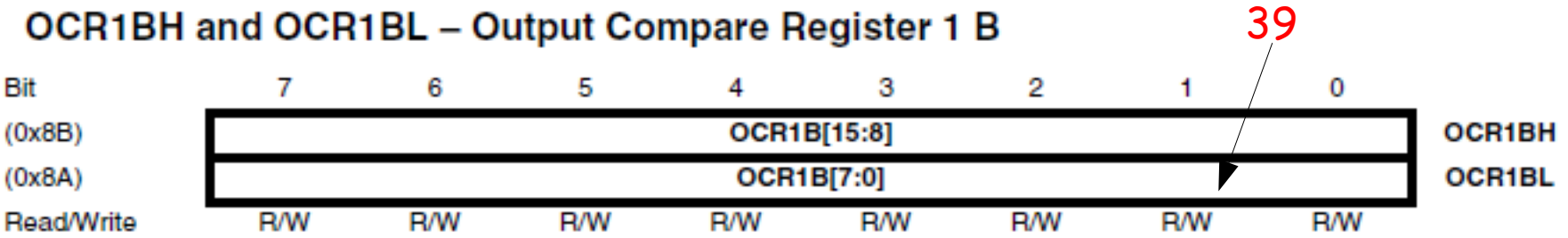
Bit	7	6	5	4	3	2	1	0	
(0x87)	ICR1[15:8]								ICR1H
(0x86)	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Timer1 regiszterek

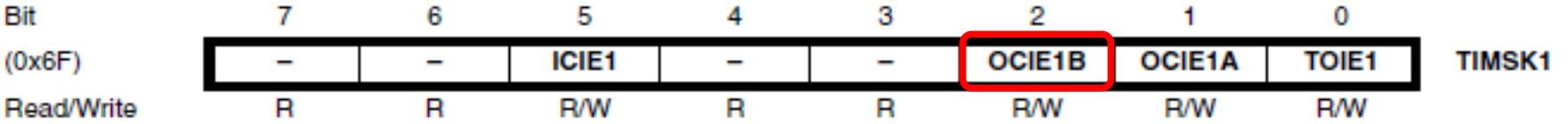
OCR1AH and OCR1AL – Output Compare Register 1 A



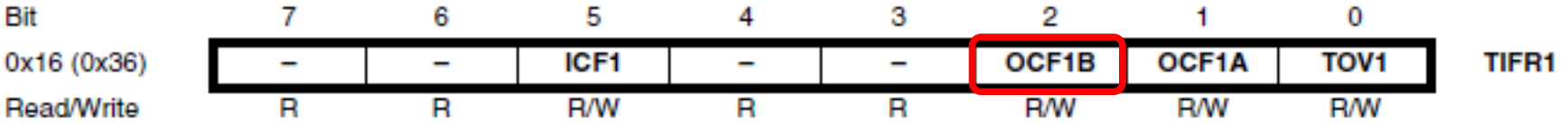
OCR1BH and OCR1BL – Output Compare Register 1 B



TIMSK1 – Timer/Counter1 Interrupt Mask Register



TIFR1 – Timer/Counter1 Interrupt Flag Register



adc_storage.iso

- Az adatokat az **ADC** konverzió végét jelző megszakításban tároljuk el a ***results[]*** nevű tömbbe
- **Timer1** megszakításai csak **ADC** konverziót indítanak, kiszolgálást nem igényelnek

```
const byte adcPin = 0;                    // A0
const int MAX_RESULTS = 512;
volatile byte results[MAX_RESULTS];
volatile int resultNumber = 0;

// ADC complete ISR
ISR (ADC_vect) {
  if (resultNumber >= MAX_RESULTS)
    ADCSRA = 0;  // turn off ADC
  else
    results [resultNumber++] = ADC >> 2;
}  // end of ADC_vect

EMPTY_INTERRUPT (TIMER1_COMPB_vect);
```

A program forrása:
www.gammon.com.au/adc
„Automatic mode" mintapéldája

Módosítások: A tömb méretét megdupláztuk, az ADC eredményeket pedig 0-255 közé transzformáljuk

adc_storage.iso

```
void setup () {
  Serial.begin (115200);
  Serial.println ();
  // reset Timer 1
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;
  TCCR1B = bit(CS11) | bit(WGM12);    // CTC, prescaler of 8, 2 MHz input frequency
  TIMSK1 = bit(OCIE1B);               // Enable Timer1 OC1B interrupt
  OCR1A = 39;                         // 20 uS - sampling frequency 50 kHz
  OCR1B = 39;
//--- ADC configuration -------------
  ADCSRA =  bit(ADEN) | bit(ADIE) | bit(ADIF);   // ADC on, interrupt on completion
  ADCSRA |= bit(ADPS2);               // Prescaler of 16  (1 MHz)
  ADMUX = bit(REFS0) | (adcPin & 7);
  ADCSRB = bit(ADTS0) | bit(ADTS2);   // Timer/Counter1 Compare Match B
  ADCSRA |= bit(ADATE);               // Turn on automatic triggering

//--- wait for buffer to fill
  while (resultNumber < MAX_RESULTS);
  for (int i = 0; i < MAX_RESULTS; i++)
    Serial.println (results [i]);
}  // end of setup

void loop () { }
```

Az ADC regiszterei

- **ADMUX** – referenciafeszültség és bemeneti csatorna választó

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

REFS – 00: `EXTERNAL`, 01: `DEFAULT` (Avcc), 11: `INTERNAL` (1,1V)
ADLAR – Az eredmény igazítása **0:** jobbra, **1:** balra
MUX – bemenetválasztás (**0000-0111:** A0-A7, **1000:** belső hőmérő, **1110:** 1,1V-os referencia, **1111:** GND)

- **ADCH** és **ADCL** adatregiszterek

ADLAR = 0

15	14	13	12	11	10	9	8	
–	–	–	–	–	–	ADC9	ADC8	ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
7	6	5	4	3	2	1	0	

ADLAR = 1

15	14	13	12	11	10	9	8	
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
ADC1	ADC0	–	–	–	–	–	–	ADCL
7	6	5	4	3	2	1	0	

Az ADC regiszterei

- **ADCSRA** – vezérlő és állapotregiszter I.

7	6	5	4	3	2	1	0	
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
1	0	1	1	1	1	0	0	

ADEN: ADC engedélyezése, **ADSC:** konverzió indítás, **ADIF:** konverzió vége jelzőbit,
ADIE: megszakításkérés engedélyezés, **ADPS[2:0]:** előszámláló választás (/2 ... /128)
ADATE: trigger enable

- **ADCSRB** – vezérlő és állapotregiszter II.

7	6	5	4	3	2	1	0	
–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
R	R/W	R	R	R	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

ACME – az analóg komparátorhoz rendeli a bemeneti multiplexert (ADEN = 0 esetén)
ADTS[2:0] – konverziót indító (trigger) jelforrás választása:

000: Szabadonfutó mód (ADIF)	100: Timer0 Túlcsordulás
001: Analóg komparátor	101: Timer1 Compare esemény B
010: Külső megszakítás (INT0)	110: Timer1 Túlcsordulás
011: Timer0 Compare esemény A	111: Timer1 Capture esemény

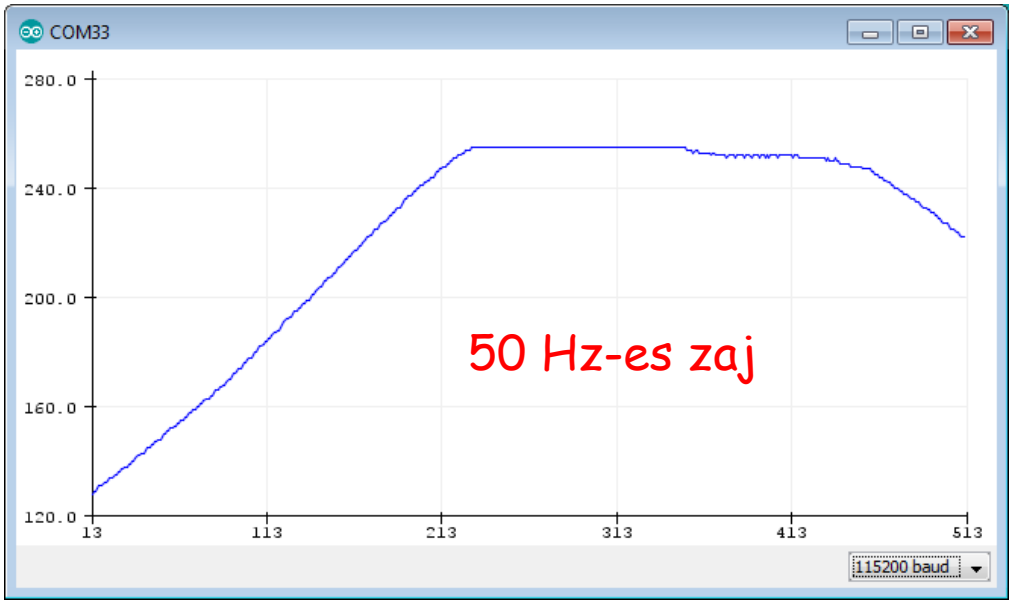
Az ADC regiszterei

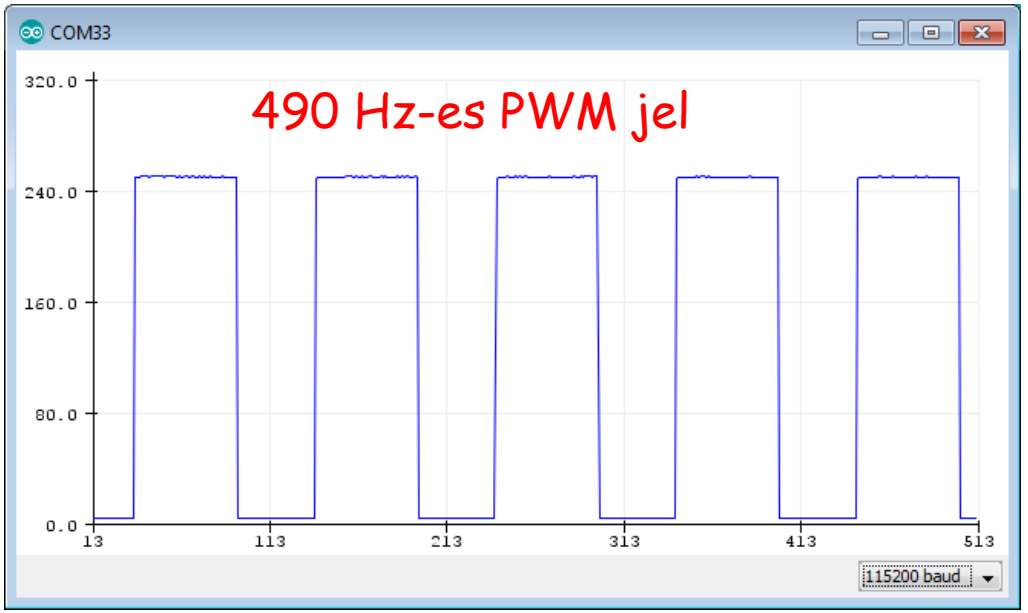
DIDR0 – Digital Input Disable Register 0

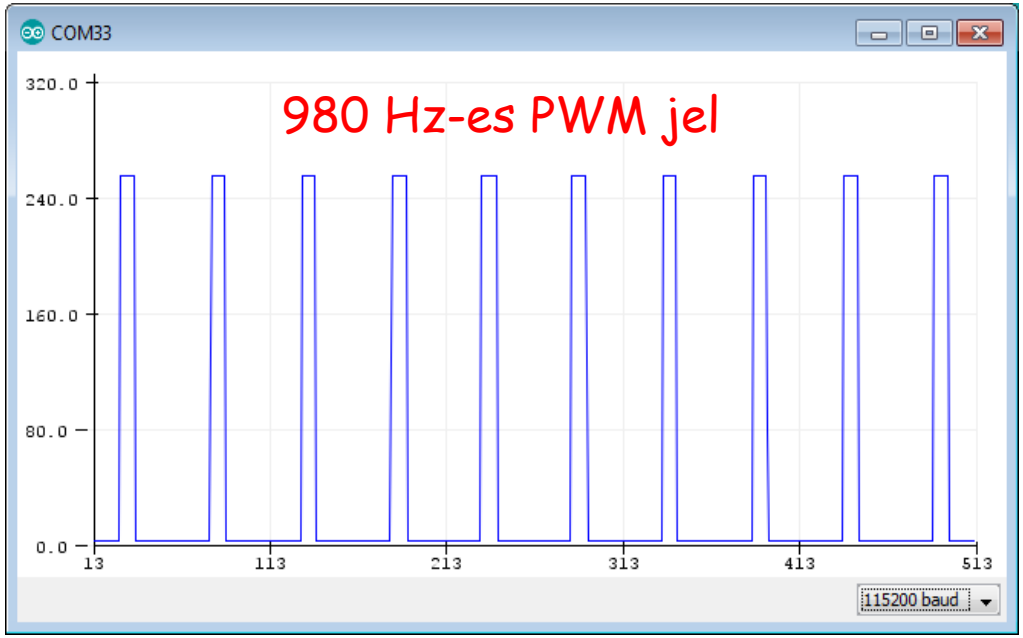
Bit	7	6	5	4	3	2	1	0	
(0x7E)	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	DIDR0
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

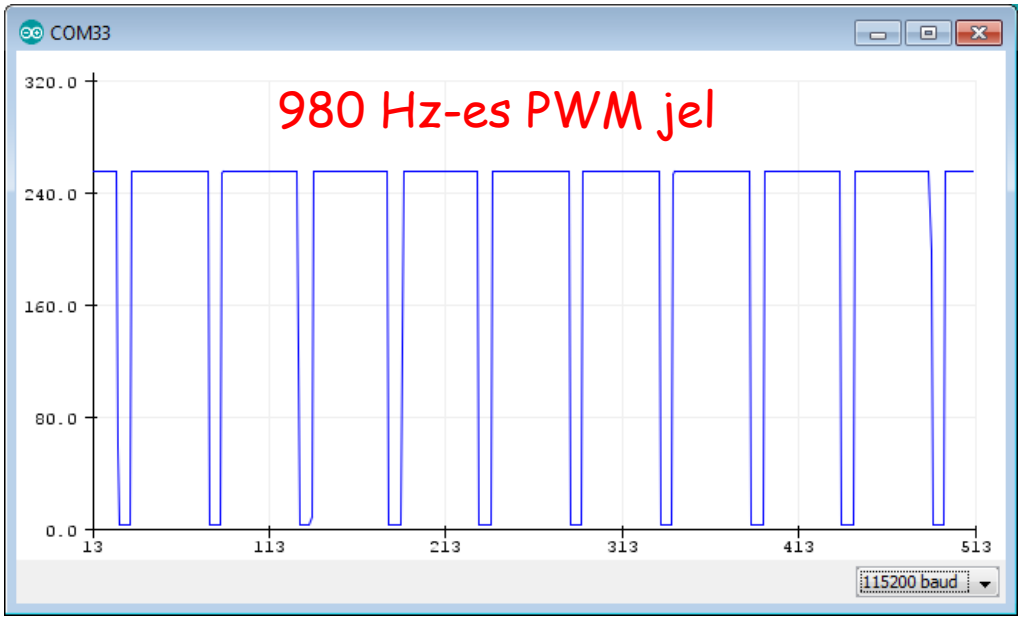
- Ha 1-et írunk valamelyik bitbe, az letiltja a hozzá tartozó **A0 – A5** kivezetés digitális bemeneti bufferét. **A6** és **A7** csak analóg bemenet, ezeknél nincs mit letiltani…

Futási eredmények









Ellenállás színkódok