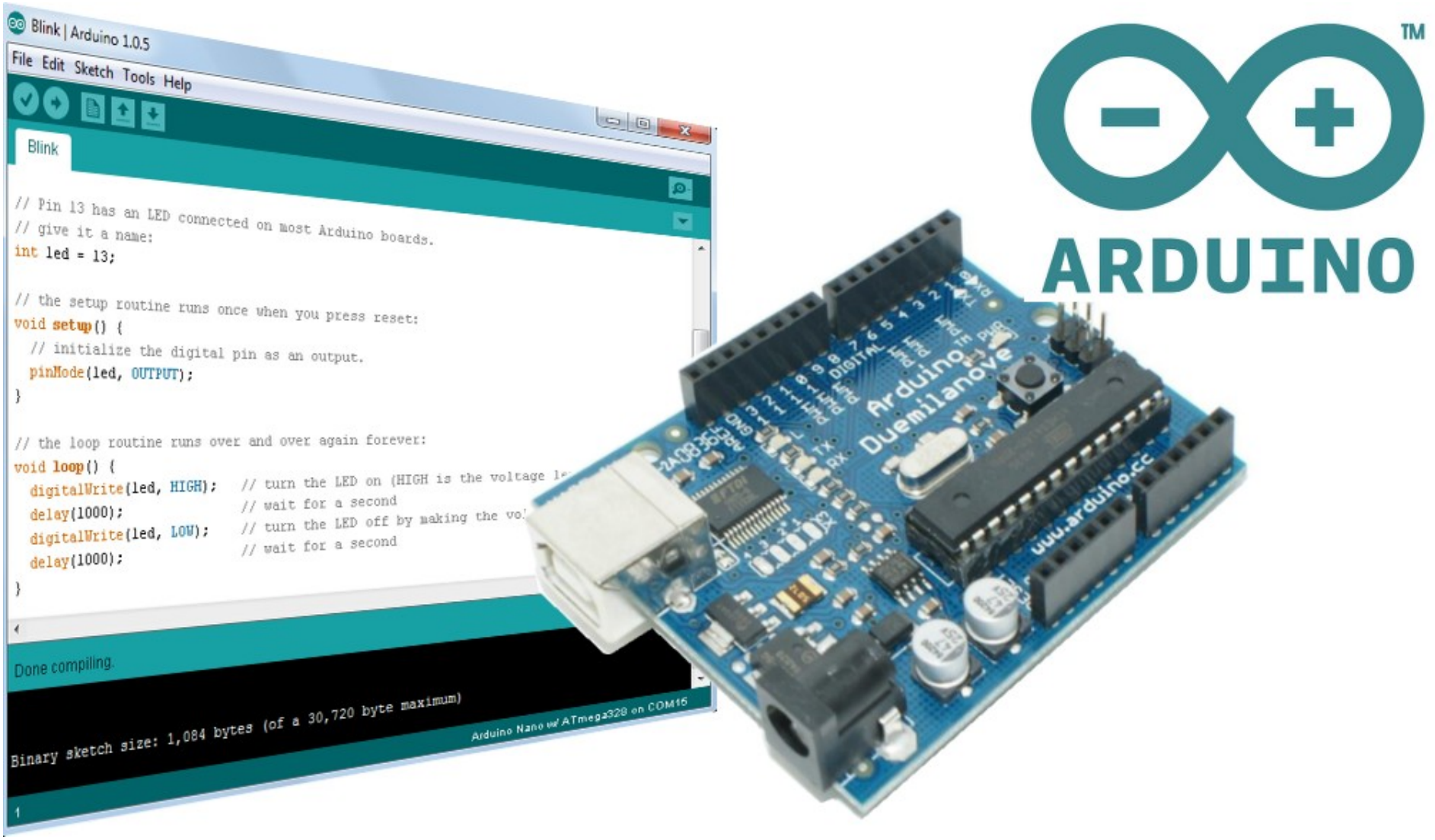
Arduino tanfolyam kezdőknek és haladóknak



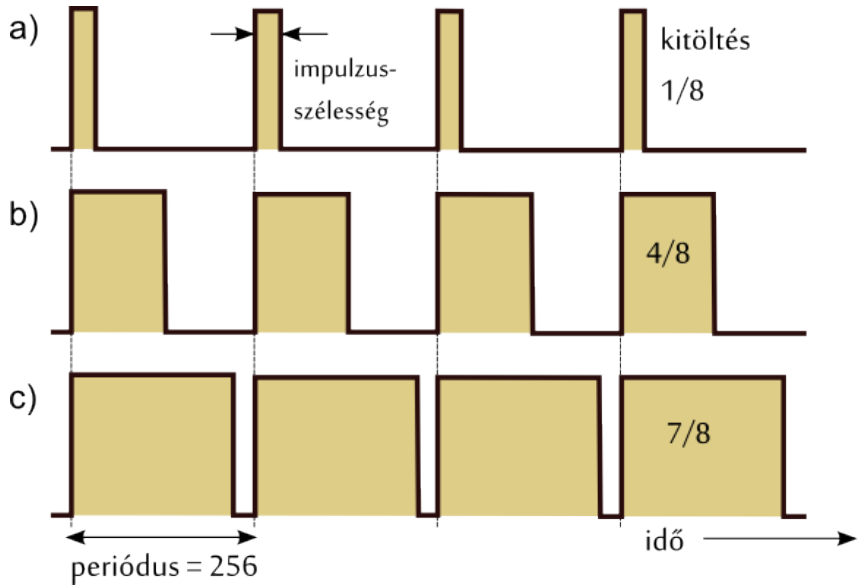
5. Analóg I/O

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

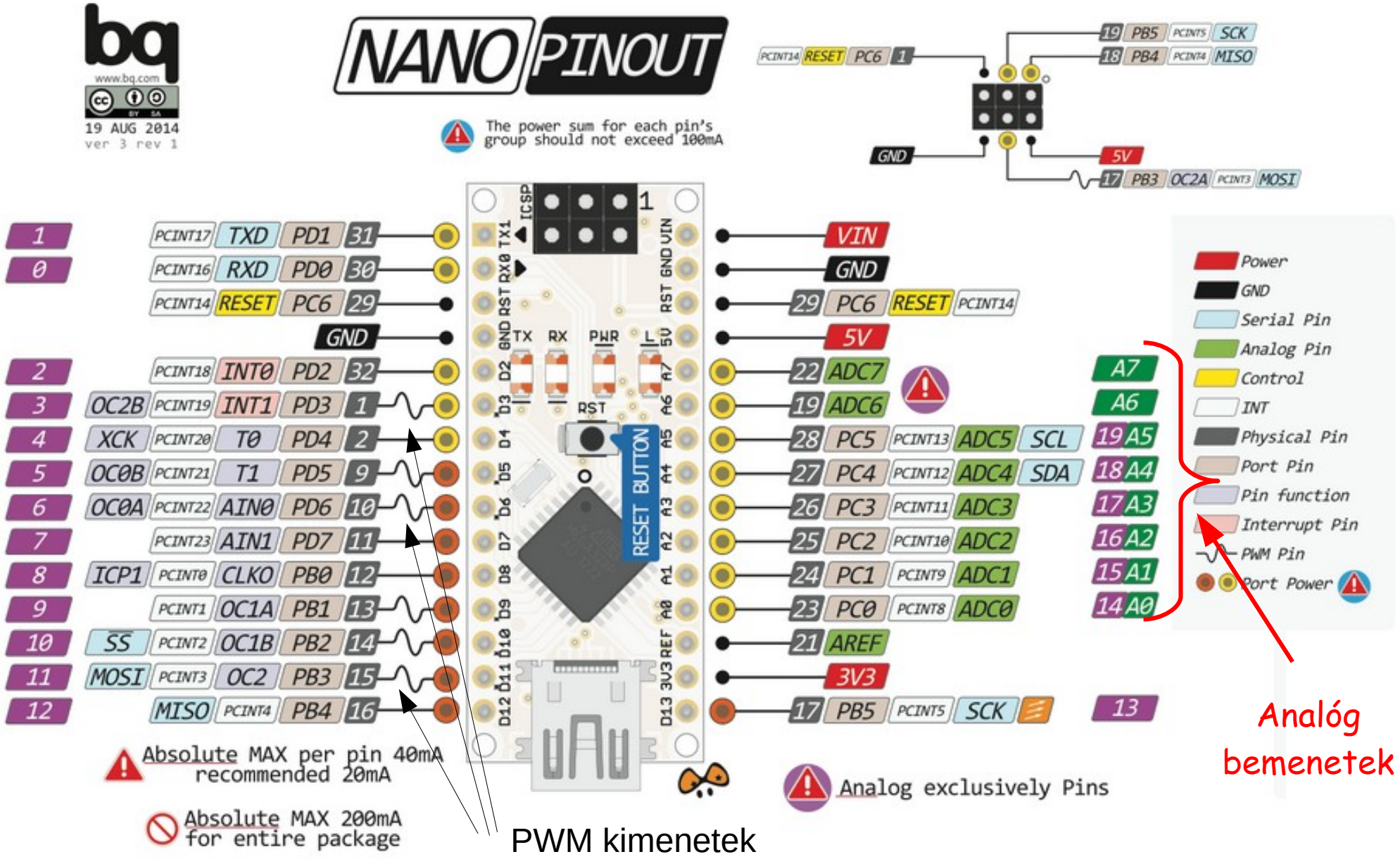
PWM: impulzus-szélesség moduláció

- 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**):



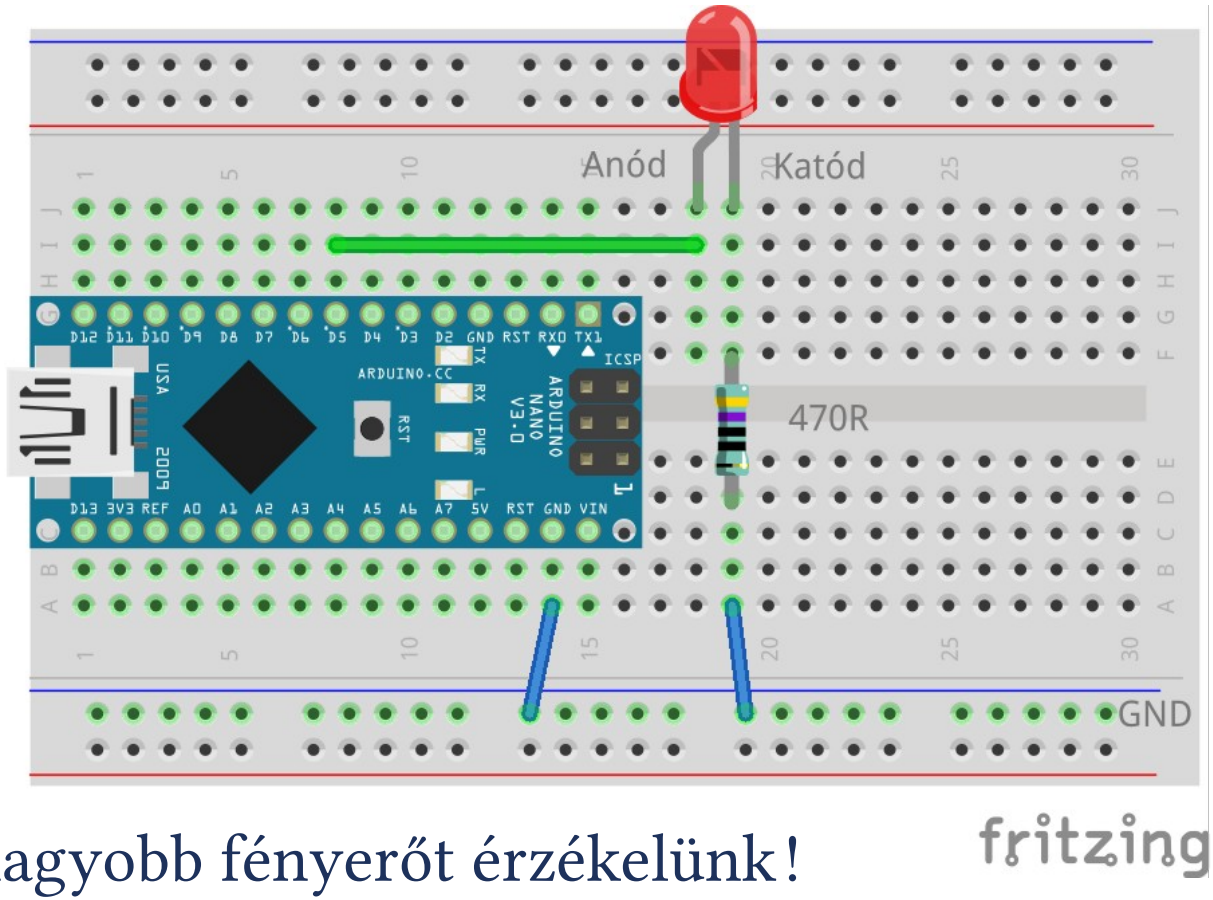
Időzítő	Csatorna	Kivezetés	Frekvencia
Timer0	OC0A, OC0B	6, 5	980 Hz
Timer1	OC1A, OC1B	9, 10	490 Hz
Timer2	OC2A, OC2B	11, 3	490 Hz

Az Arduino nano kártya kivezetései



„Lélegző” LED – led_fade.ino

- A **D5** kivezetésre kötött LED fényerejét fokozatosan növeljük, majd a maximum elérése után fokozatosan csökkentjük
- Az **`analogWrite(pin,duty)`** függvény két paramétere a vezérelni kívánt digitális kivezetés sorszáma (csak 3, 5, 6, 9, 10, 11 lehet) és a PWM kitöltési tényező (0 – 255 közötti érték)
- Mivel a LED úgy van bekötve, hogy magas szintű jel esetén világít, így a kis kitöltés kisebb fényerőt, a nagy kitöltés nagyobb fényerőt jelent.
- Szemünk nem lineárisan érzékel, kétszer nagyobb kitöltésnél nem kétszer nagyobb fényerőt érzékelünk!



led_fade.ino

- A **D5** kivezetésre kötött LED fényerejét változtatjuk („lélegző” LED)

```
#define led      5          // ide van kötve a LED
int brightness = 0;         // fényerő értéke
int fadeAmount = 5;         // a változás léptéke

void setup() {
  pinMode(led, OUTPUT);     // D5 legyen kimenet (ez a sor elhagyható)
}
void loop() {
  // A LED kimeneten beállítjuk a kitöltést (a LED fényerejét)
  analogWrite(led, brightness);
  // megnöveljük a fényerő értékét az általunk megadott
  // léptékben, mindig, mikor lefut a loop, hozzáadódik
  brightness = brightness + fadeAmount;
  // a végén megfordítjuk a változás irányát:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // Várunk, hogy a szemünk is láthassa a változást:
  delay(50);
}
```

Harsányi Réka - Juhász Márton András: Fizikai számítástechnika

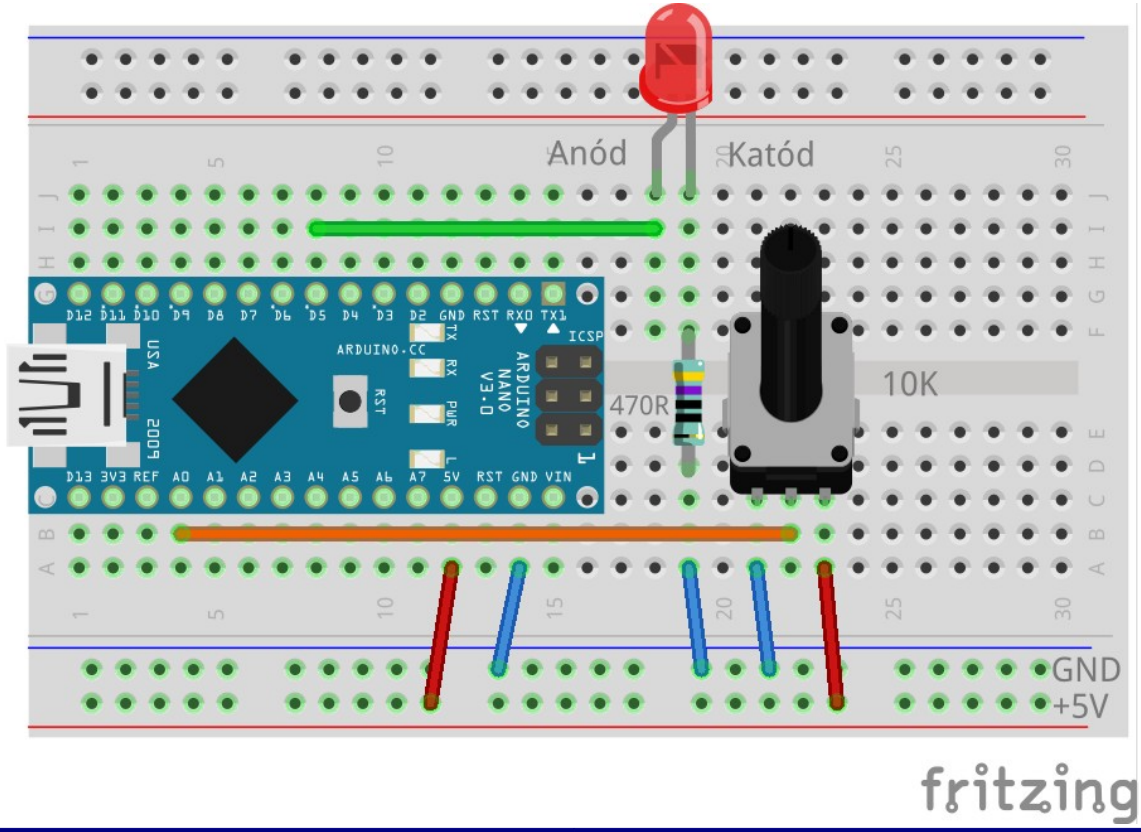
led_fade_exp.ino

- Nemlineáris teljesítmény-változtatással talán szebben „lélegzik”...

```
#define led      5              // D5-re van kötve a LED
uint16_t idx = 0;               // Index a kitöltési tényezők számításához
uint16_t next_sqr = 1;          // A következő négyzetszám
uint16_t sqr_step = 3;          // Új növekmény a négyzetszámok számításához

void setup() {
  pinMode(led, OUTPUT);         // D5 legyen kimenet
}

void loop() {
  analogWrite(led,next_sqr>>4);
  idx++;                        // A futó index növelése
  if (idx < 63) {               // Az első 63 lépésben felfelé lépünk
    next_sqr += sqr_step;
    sqr_step += 2;
  } else if (idx < 125) {       // A második 63 lépésben lefelé lépünk
    sqr_step -= 2;
    next_sqr -= sqr_step;
  } else idx = 0;               // Új periódus kezdődik
  delay(25);
}
```

Teljesítményvezérlés potméterrel

- A potméterrel leosztott feszültséget megmérjük az `analogRead(A0)` függvényhívással (0 – 1023 közötti értéket kapunk)
- A kapott számot átskálázzuk a 0 – 255 tartományba, majd az `analogWrite()` függvényhívással erre az értékre állítjuk be a **D5** PWM csatorna kitöltését
- **LED Arduino**
  anód – D5
  katód – GND
  (470 Ω-on keresztül)
- **Potméter Arduino**
  teteje +5 V
  csúszka A0
  alja GND
  (a potméter 10 kΩ-os)



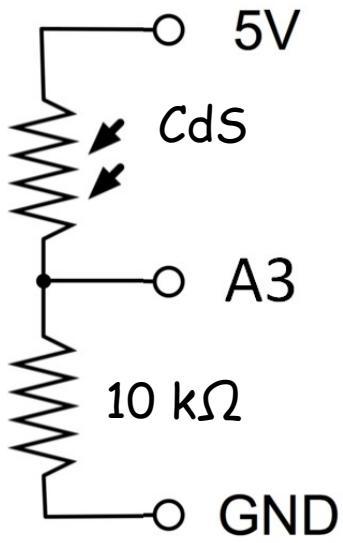
led_pwm.ino

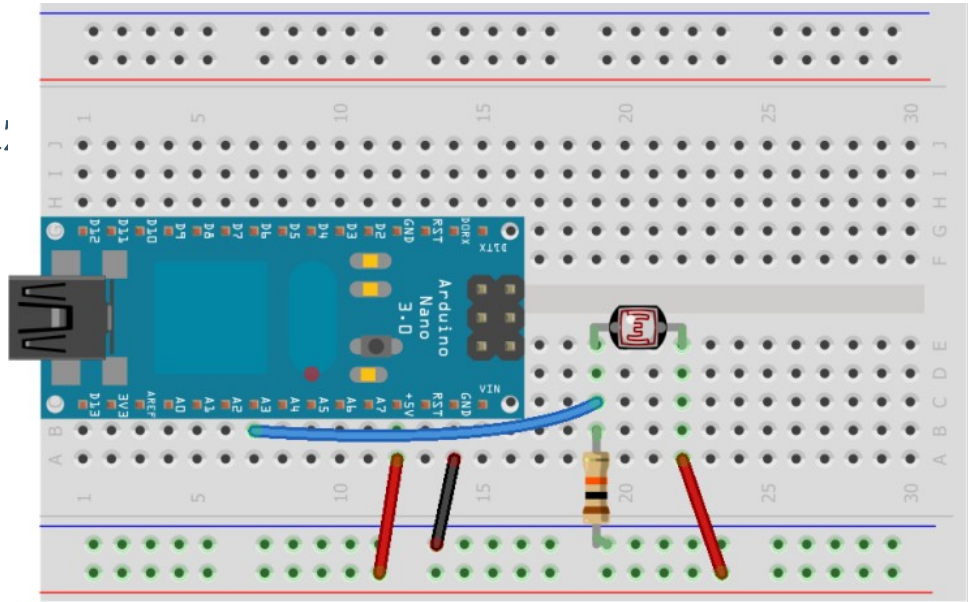
- Az ADC 0 – 1023 közötti számot ad vissza, ezt legalább néggyel kell osztani, hogy 0 – 255 közötti számot kapjunk a PWM vezérléséhez
- Itt most osztás helyet jobbraléptetést végzünk

```
#define 5                                   // D5-re van kötve a LED

void setup() {
                                            // Nincs tennivaló
}

void loop() {
  int reading = analogRead(A0);
  analogWrite(led, reading>>2 );  // Osztás 4-gyel
  // Várunk, hogy a szemünk is láthassa a változást:
  delay(50);
}
```

Fénymérés fényérzékeny ellenállással

- A kapcsolás feszültségosztóként működik, amelyikben a felső tag egy CdS fényérzékeny ellenállás, melynek ellenállása a megvilágítástól függően széles határok között változik. A megvilágítás hatására az ellenállása csökken…
- Az ellenállásosztó közös pontját a **A3** analóg bemenetre kötöttük
- Az eredményt soros porton kiküldjük a számítógépre, s a terminálablakban jelenik meg.





Photoresistor.ino

```
void setup () {
   Serial.begin(9600);
   analogReference(DEFAULT);   //VCC a referencia
   Serial.print("Photoresistor");
}

void loop () {
   long reading = 0;
   for(int i=0; i<4750; i++) {
      reading += analogRead(A3);
   }
   // Trükkös osztás 1024-gyel
   float voltage = reading>>10;
   Serial.print(voltage,0);
   Serial.print(" mV, ");
   // Átszámítás kOhm-ra
   // Rx = VCC*10k/voltage - 10k
   float rx = 47500/voltage - 10;
   Serial.print(rx,3);
   Serial.println(" kOhm");
   delay (5000);
}
```

4750 db mérés eredményét összeadjuk és nem szorzunk Vref-fel!

Photoresistor
3145 mV, 5.103 kOhm
3149 mV, 5.084 kOhm
3036 mV, 5.646 kOhm
1393 mV, 24.099 kOhm
1322 mV, 25.930 kOhm
4406 mV, 0.781 kOhm
4449 mV, 0.677 kOhm
4134 mV, 1.490 kOhm
2856 mV, 6.632 kOhm
2836 mV, 6.749 kOhm

Az érzékelési tartomány sávokra osztása

- Módosítsuk az előző programot úgy, hogy az ADC-vel mért értékeket sávokra osztjuk fel, s az eredmény szöveges értékelését kiíratjuk a soros porton!
- Legyenek a kiírandó szövegek:
  - sötét
  - derengő fény
  - közepes fény
  - erős fény
- Az eredmény szöveges értékeléséhez használjuk a **switch** utasítást!

A switch utasítás

- A **switch** utasítás egy kifejezés értékét több egész értékű állandó értékével hasonlítja össze, és azt az ágat hajtja végre, amelyiknél egyezést talál
- A switch utasítás általános felépítése:

```
switch (kifejezés) {
    case állandó kifejezés: utasítások; break
    case állandó kifejezés: utasítások; break
    . . .
    default: utasítások
}
```

A **break** szerepe az, hogy kiugorjon a **switch** utasítás törzséből

- Egy példa:

```
switch (led_state) {
  case 0: digitalWrite(RED_LED, LOW); break;
  case 1: digitalWrite(RED_LED, HIGH); break;
  default: digitalWrite(RED_LED, !digitalRead(RED_LED));
}
```

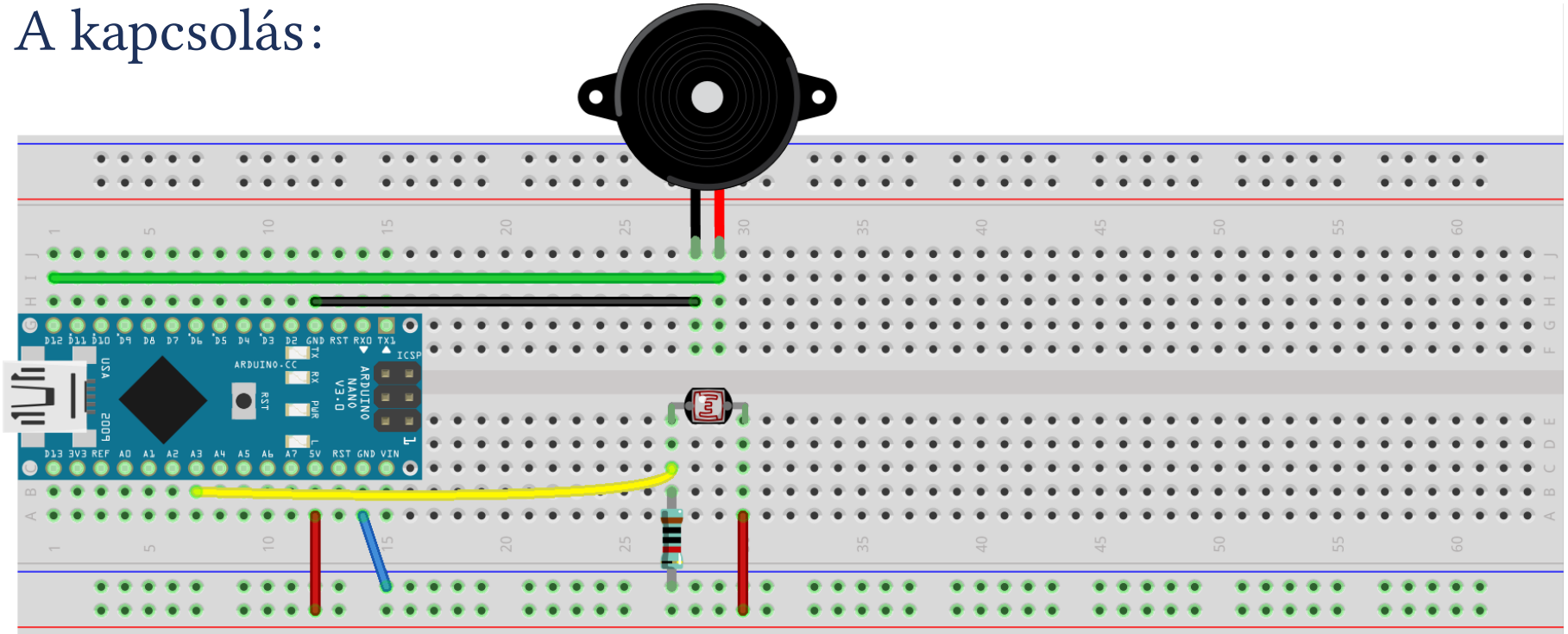
switch_case.ino

- A fénymérés eredményének szöveges kiértékelése

```
const int sensorMin = 0;          // szenzor minimum (tapasztalat alapján)
const int sensorMax = 800;        // szenzor maximum (tapasztalat alapján)

void setup() {
  Serial.begin(9600);             // Soros port inicializálása
}

void loop() {
  int reading = analogRead(A3); // Szenzor kiolvasás
  //--- A mérési tartomány leképezése négy esetre ---
  int range = map(reading,sensorMin,sensorMax, 0, 3);
  //--- Elágazás az eredmény szerint ----------------
  switch (range) {
    case 0:     // Eltakart szenzor esetén
      Serial.println("sötét"); break;
    case 1:     // kezed közel a szenzorhoz
      Serial.println("derengő fény"); break;
    case 2:     // kezed 10 cm-re a szenzortól
      Serial.println("közepes fény"); break;
    case 3:     // kezed nincs a fény útjában
      Serial.println("erős fény"); break;
  }
  delay(1000);  // Késleltetés a kiíratások között
}
```

light_theremin.ino

- A Theremin (Лев Сергеевич Термен fizikus nevéről) olyan elektronikus hangszer, amelyet gesztusokkal lehet vezérelni
- Az „ Arduino light theremin” egy fényérzékelő segítségével reagál a gesztusokra és az Arduino egy piezo csipogó segítségével szólaltatja meg a hangot (nem lesz túl zenei...)
- A kapcsolás:



light_theremin.ino

```
#define buzzerPin 12                    // Piezo buzzer D12 és a GND közé kötve
#define ledPin 13                       // A beépített LED
int ldr;                                // A beolvasott fényerősség (0-1023)
int ldrMin = 1023;                      // A fényerősség minimuma
int ldrMax = 0;                         // A fényerősség maximuma

void setup() {
  pinMode(ledPin, OUTPUT);              // ledPin legyen kimenet
  digitalWrite(ledPin, HIGH);           // LED be, kalibráció kezdete
  while (millis() < 5000) {             // 5 másodpercing kalibrálunk
    ldr = analogRead(A3);               // Fényszenzor leolvasása
    if (ldr > ldrMax) ldrMax = ldr;     // A maximális érték eltárolása
    if (ldr < ldrMin) ldrMin = ldr;     // A minimális érték eltárolása
  }
  digitalWrite(ledPin, LOW);            // LED ki, kalibráció vége
}

void loop() {
  ldr = analogRead(A3);                 // Fényszenzor leolvasása
  int freq = map(ldr,ldrMin,ldrMax,50,4000);  // A fényerősség leképezése
  tone(buzzerPin,freq,20);              // Megszólaltatjuk a hangot
  delay(10);                            // Egy kicsit várunk
}
```

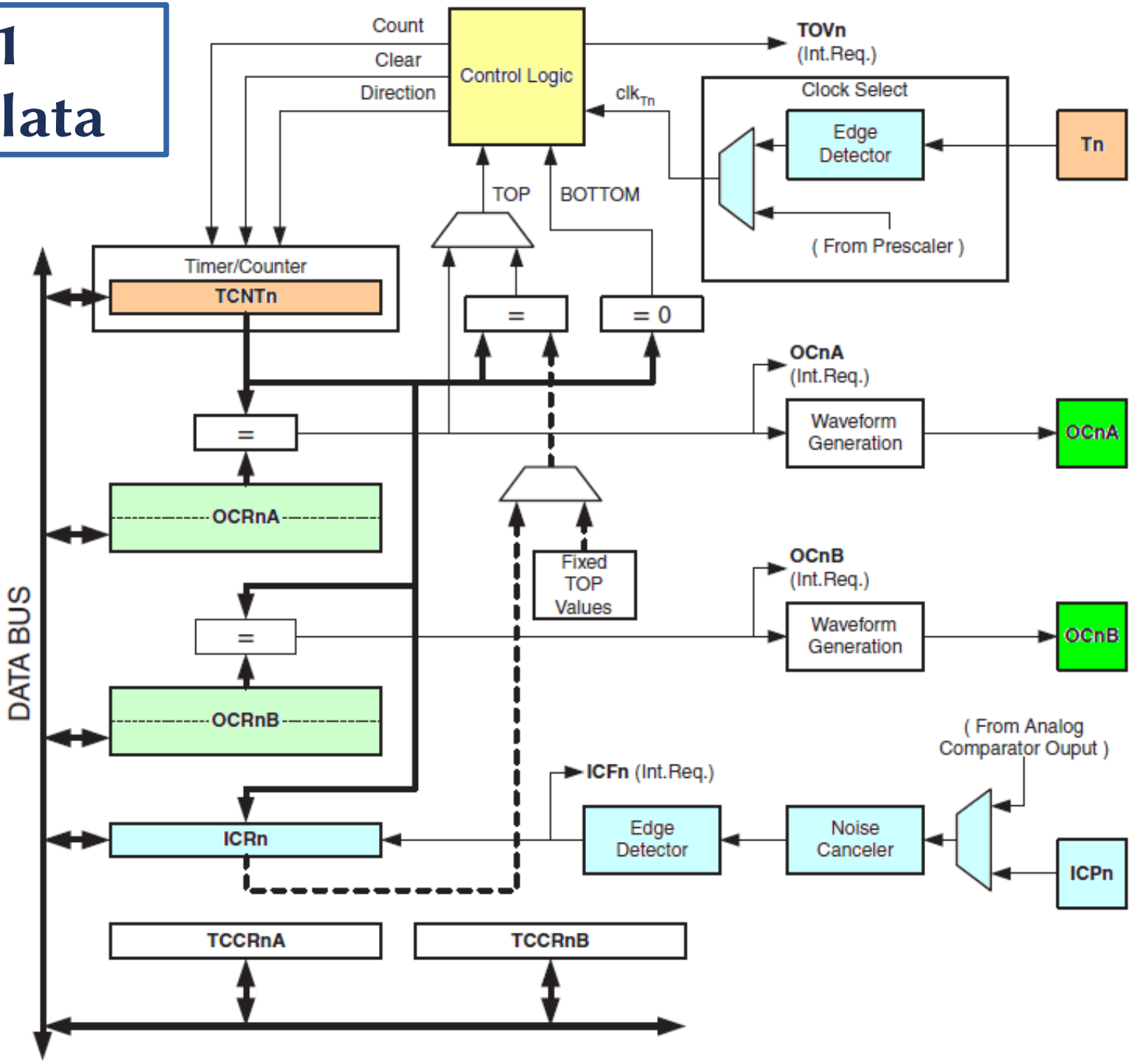
Arduino időzítők/számlálók

- 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



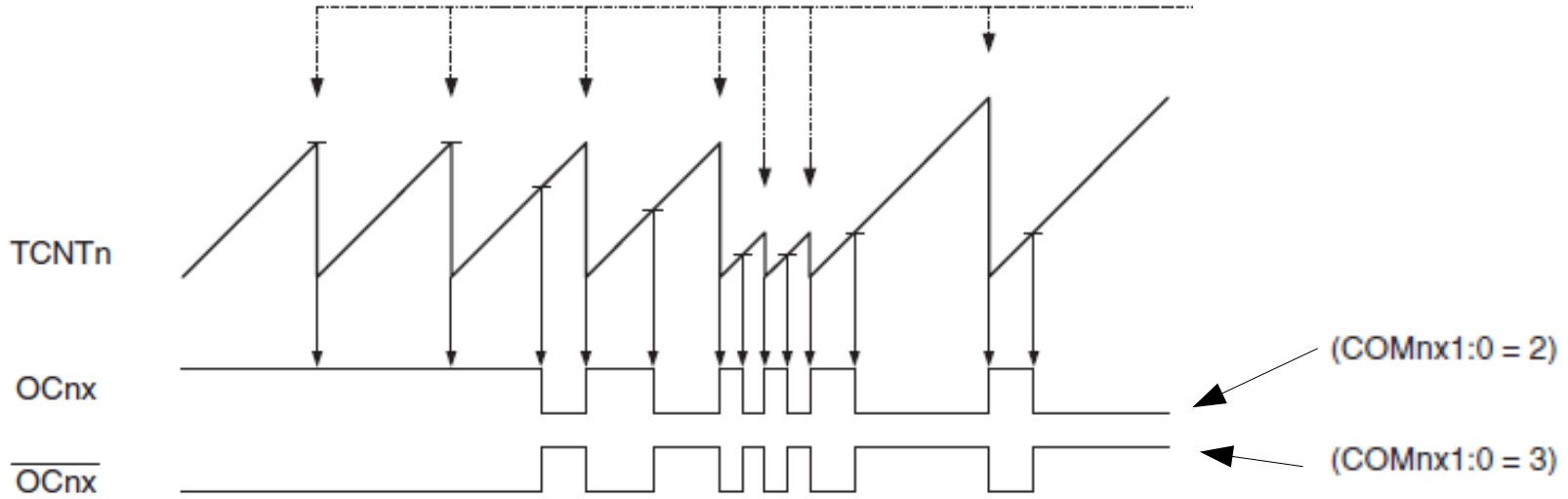
Timer1 regiszterek

- **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	

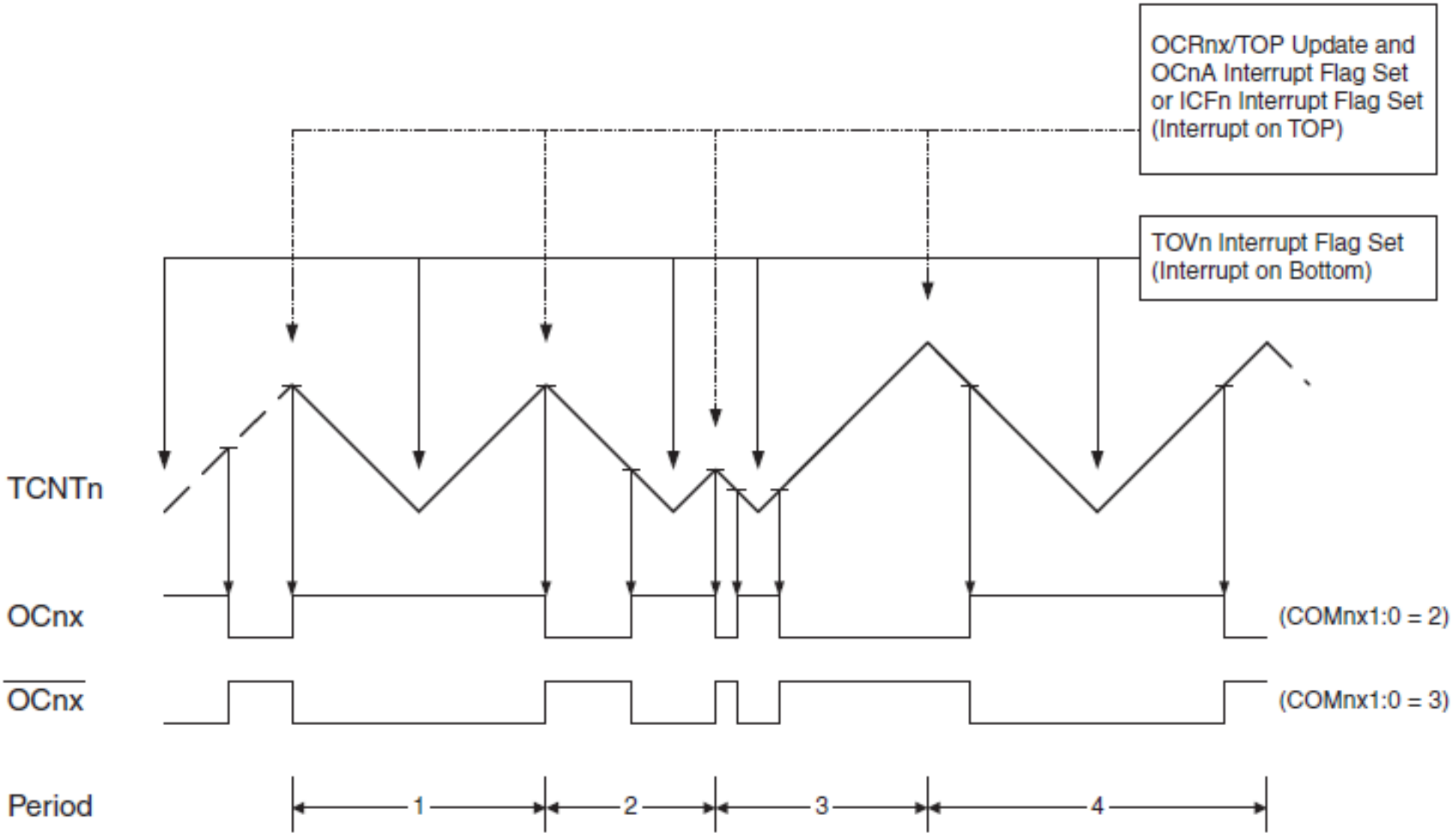
Table 16-1. Compare Output Mode, non-PWM

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).



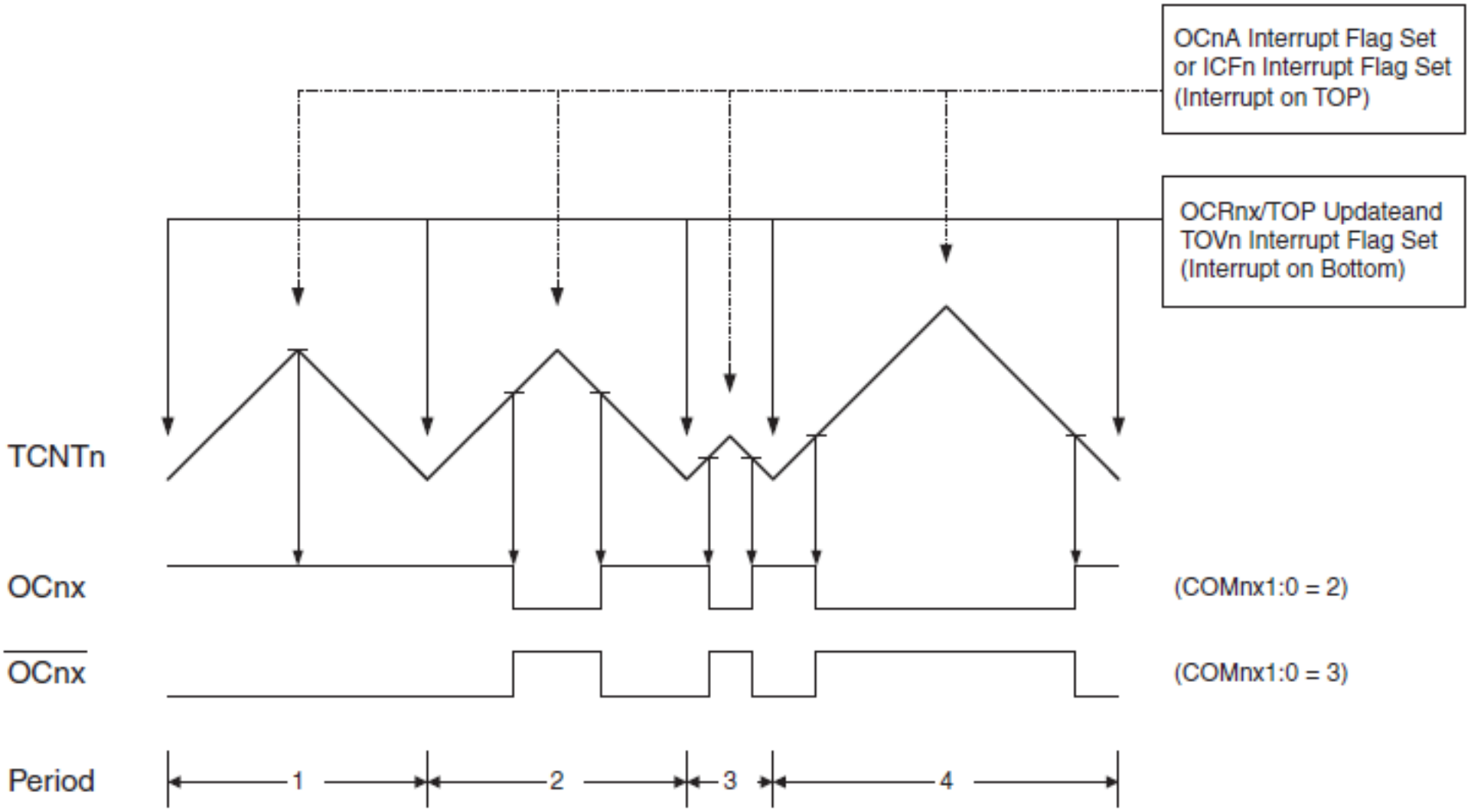
Fel-le számláló módú PWM 1.

Phase Correct PWM Mode, Timing Diagram



Fel-le számláló módú PWM 2.

Phase and Frequency Correct PWM Mode, Timing Diagram



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	

- **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	

- **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

Bit	7	6	5	4	3	2	1	0	
(0x89)	OCR1A[15:8]								OCR1AH
(0x88)	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

OCR1BH and OCR1BL – Output Compare Register 1 B

Bit	7	6	5	4	3	2	1	0	
(0x8B)	OCR1B[15:8]								OCR1BH
(0x8A)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	

TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	

Timer megszakítások

- Forrás: www.instructables.com/id/Arduino-Timer-Interrupts/
- A mintaprogram periodikus megszakításokat kelt mindhárom időzítő **OCR*n*A** csatornája segítségével (Timer0: 2 kHz, Timer1: 1 Hz, Timer2: 8 kHz)
- A megszakításokban egy-egy digitális kimenetet átbillentünk (Timer0: D5 pin, Timer1: D13 pin, Timer2: D12 pin)
- A mellékelt **timer_interrupts.ino** mintapéldából itt most csak Timer1-re vonatkozó részleteket mutatjuk be

timer_interrupts.ino (releváns részletek)

```
boolean toggle1 = 0;

void setup(){
  pinMode(13, OUTPUT);
  cli();                      // stop interrupts
  TCCR1A = 0;                 // set entire TCCR1A register to 0
  TCCR1B = 0;                 // same for TCCR1B
  TCNT1  = 0;                 // initialize counter value to 0
  OCR1A = 15624;              // set compare match register for 1Hz increments
  TCCR1B |= (1 << WGM12);   // turn on CTC mode
  TCCR1B |= (1 << CS12) | (1 << CS10);  // Set CS12 and CS10 bits for 1024 prescaler
  TIMSK1 |= (1 << OCIE1A); // enable timer compare interrupt
  sei();                      // allow interrupts
}

ISR(TIMER1_COMPA_vect){     // timer1 interrupt 1Hz toggles pin 13 (LED)
  if (toggle1){
    digitalWrite(13,HIGH);
    toggle1 = 0;
  }
  else{
    digitalWrite(13,LOW);
    toggle1 = 1;
  }
}                           // a loop függvény itt üres, ezért nem részleteztük itt ...
```

A TimerOne programkönyvtár

- A **TimerOne** könyvtár legújabb változatát *Paul Stoffregen* tette közzé
- Leírás: www.pjrc.com/teensy/td_libs_TimerOne.html
- Letöltés: github.com/PaulStoffregen/TimerOne
- **Konfigurálás:**
  - **Timer1.initialize(***microseconds***)** – inicializálás és periódusidő megadása
  - **Timer1.setPeriod(***microseconds***)** – periódusidő megváltoztatása
- **Az időzítő számlálásának vezérlése:**
  - **Timer1.start()** - elindítja az időzítőt, új periódust kezd
  - **Timer1.stop()** - leállítja az időzítőt
  - **Timer1.restart()** - újraindítja az időzítőt, új periódust kezdve
  - **Timer1.resume()** - újra elindítja az időzítőt, de nem kezd új periódust

A TimerOne programkönyvtár

- **PWM kimenő jel előállítása:**
  - **Timer1.pwm(*pin, duty*)** – az időzítő egyik PWM csatornájának konfigurálása (***pin:*** 9, vagy 10, ***duty,*** azaz a kitöltés: 0 –1023 lehet)
  - **Timer1.setPwmDuty(*pin, duty*)** – a PWM jel kitöltésének átdefiniálása
  - **Timer1.disablePwm(*pin*)** - PWM csatorna lezárása (*pin* újra GPIO láb lesz)
- **Timer1 megszakítások kezelése:**
  - **Timer1.attachInterrupt(*function*)** – visszahívási függvény hozzárendelése a **Timer1** megszakításhoz
  - **Timer1.detachInterrupt()** – a visszahívási függvény hozzárendelésének megszüntetése

timer_ledblink.ino

```
#include <TimerOne.h>
#define led LED_BUILTIN
int ledState = LOW;
volatile unsigned long blinkCount = 0;

void setup(void) {
  pinMode(led, OUTPUT);
  Timer1.initialize(150000);          // Periódusidő = 0.15 s
  Timer1.attachInterrupt(blinkLED);   // Periódus végén blinkLED meghívása
  Serial.begin(9600);
}

void blinkLED(void) {                 // Visszahívási függvény Timert1 megszakításhoz
  digitalWrite(led, ledState);        // LED állapot kijelzése
  ledState = !ledState;               // következő állapot ellentétes legyen (toggle)
  blinkCount++;                       // Megszakítások számlálása
}

void loop(void) {
  unsigned long blinkCopy;            // Ide másoljuk a számlálót
  noInterrupts();                     // Megszakítások ideiglenes tiltása
  blinkCopy = blinkCount;             // Másolat készítése  (kritikus szakasz)
  interrupts();                       // Megszakítások újraengedélyezése
  Serial.print("blinkCount = ");      // Megszakítások számának kiírása
  Serial.println(blinkCopy);
  delay(1000);                        // 1 másodpercenként írunk ki
}
```

Ellenállás színkódok