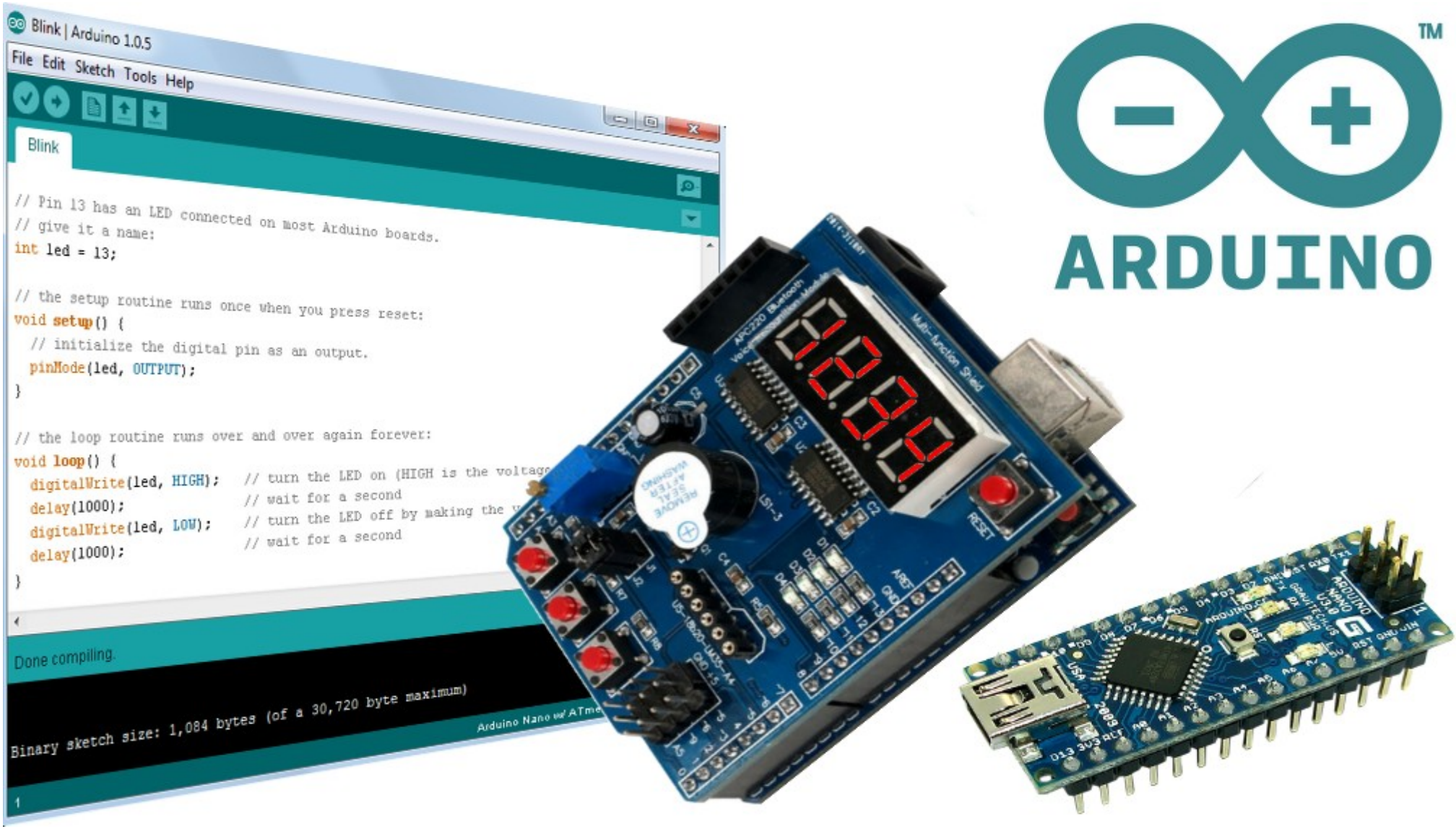
# Arduino tanfolyam kezdőknek és haladóknak



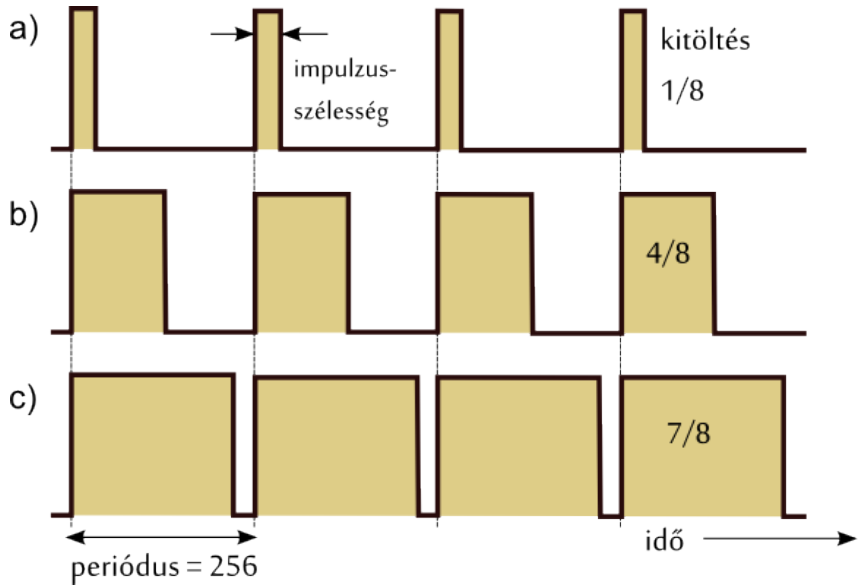

## 8. Impulzusszélesség moduláció (PWM)

# 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**):



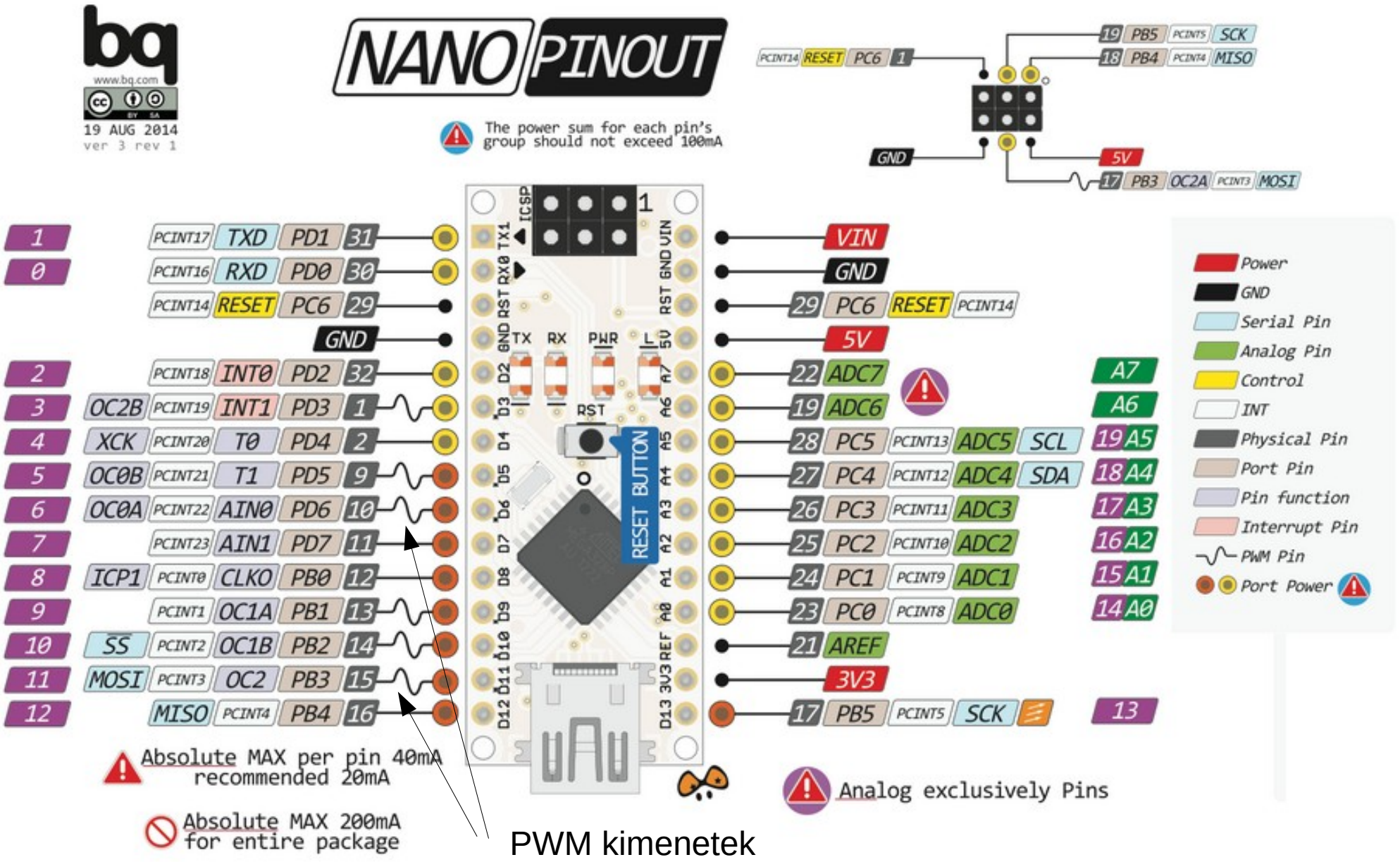

| Időzítő | Csatorna | Kivezetés | Frekvencia |
|---|---|---|---|
| Timer0 | OC0A, OC0B | 6, 5 | 980 Hz |
| Timer1 | OC1A, OC1B | 9, 10 | 490 Hz |
| Timer2 | OC2A, OC2B | 11, 3 | 490 Hz |

$$f_{PWM} = \frac{16\,MHz}{64 \cdot 256} = 976.56\,Hz$$

$f_{CPU}$ — előosztás — periódus

$$f_{PWM} = \frac{16\,MHz}{64 \cdot 510} = 490.19\,Hz$$

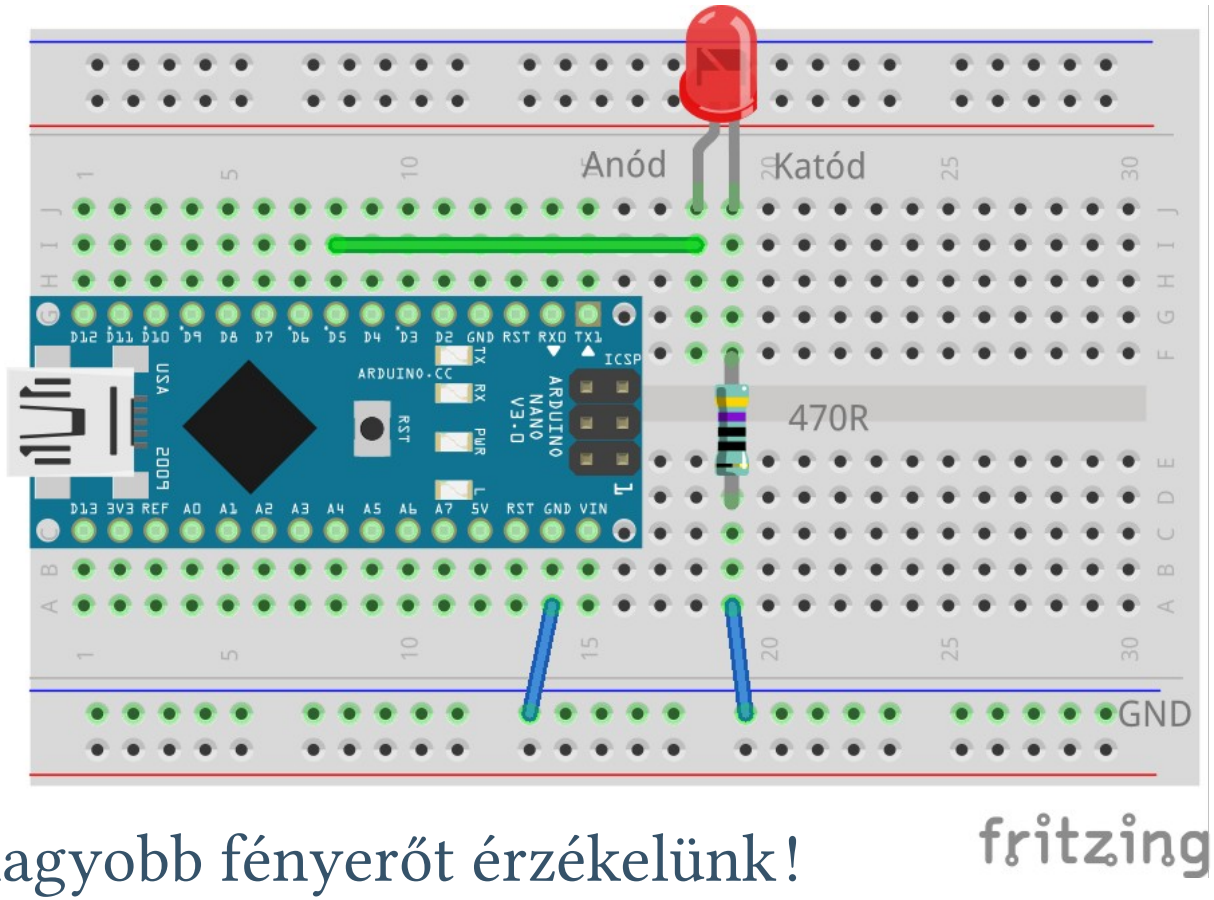
# Az Arduino nano kártya kivezetései




PWM kimenetek

# „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)

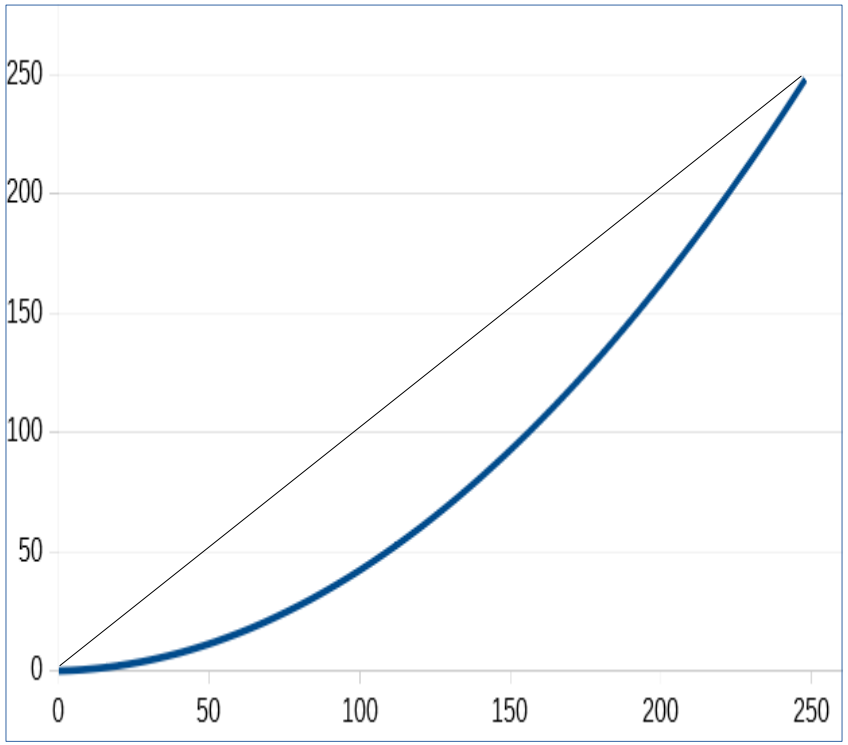
```
const int led   = 5;         // D5-re van kötve a LED
int brightness = 0;          // fényerő kezdőértéke
int fadeAmount = 5;          // a változás léptéke

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);
}
```

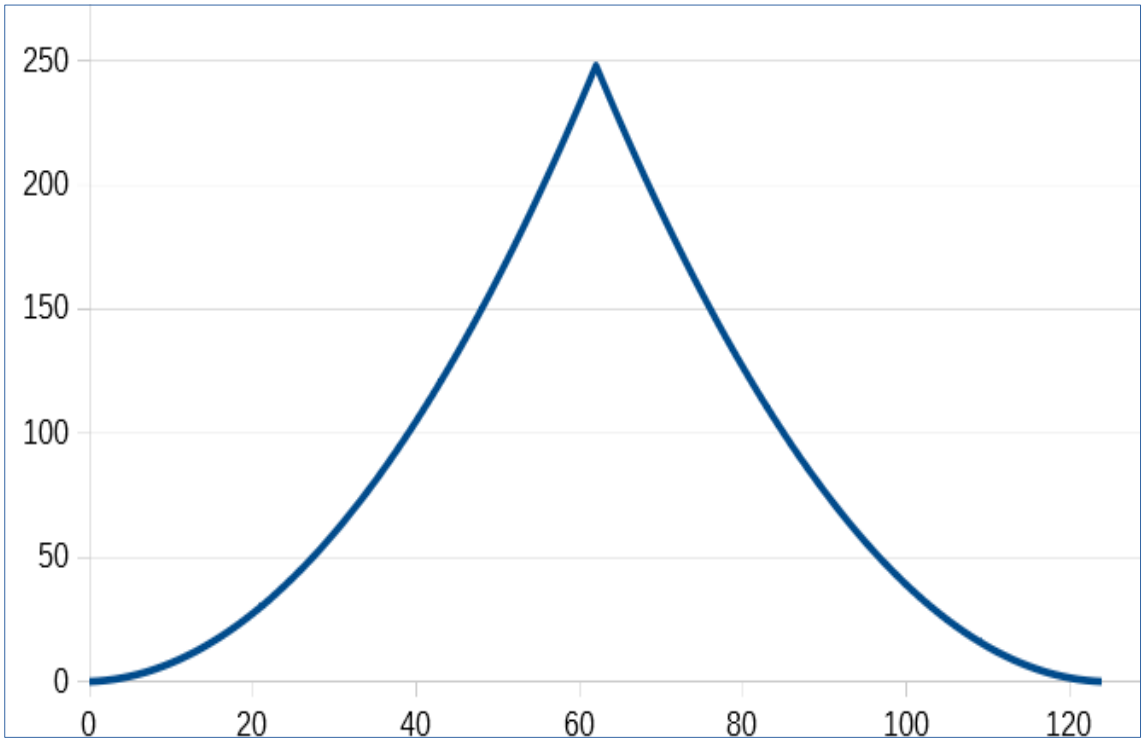
Harsányi Réka - Juhász Márton András: Fizikai számítástechnika

# „lélegző” LED – egy kicsit szebben

- Az $(x+1)^2 = x^2 + 2x + 1$ összefüggést felhasználva, minimális számítással képezhetünk nem lineárisan növekvő görbét

| x | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $x^2$ | 1 | 4 | 9 | 16 | 25 | 36 |
| 2x + 1 | 3 | 5 | 7 | 9 | 11 | 13 |



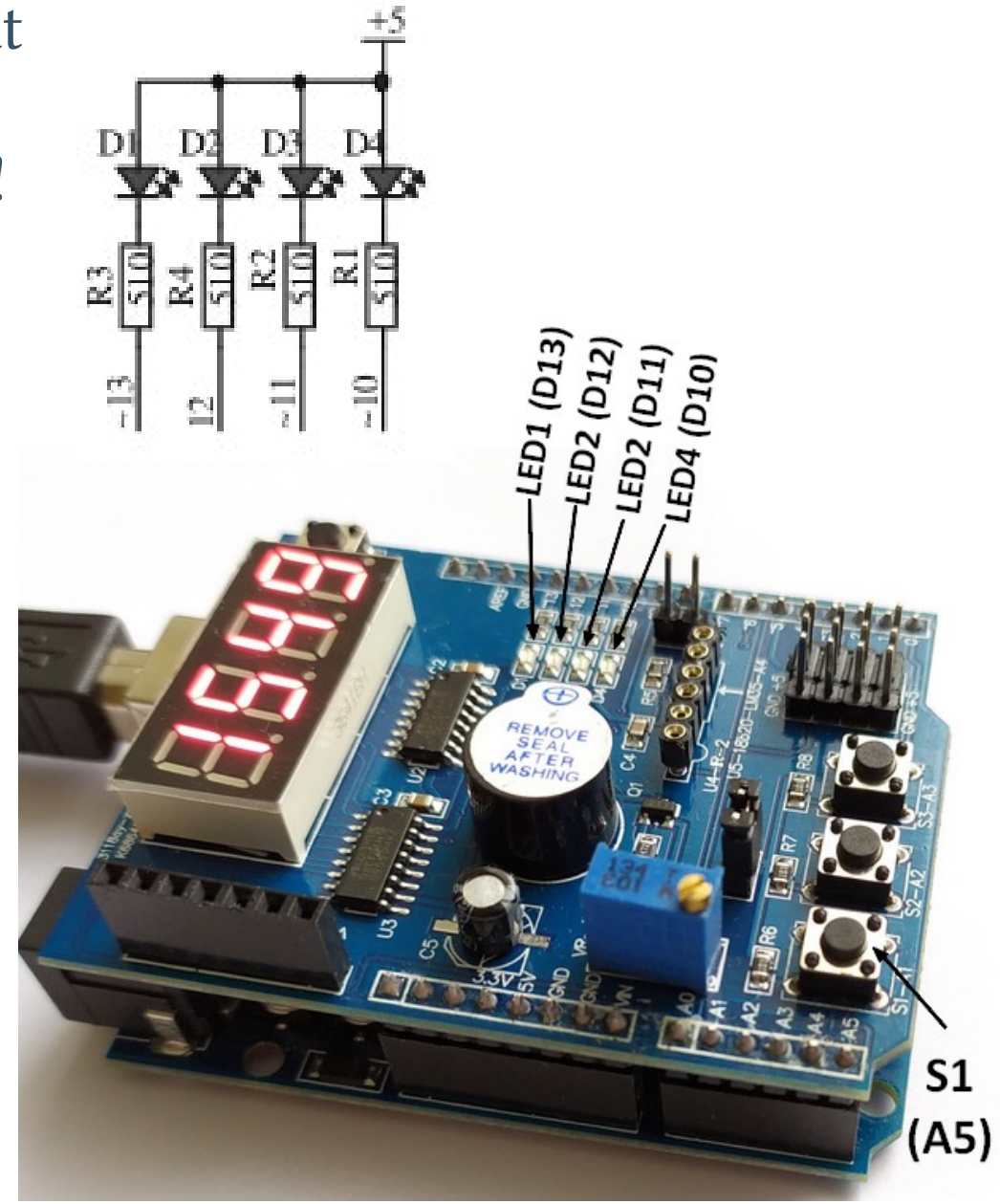
# led_fade_exp.ino

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

```
#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);
}
```

# Használjuk a multifunkciós kártyát!

- Módosítsuk az előző programokat úgy, hogy a **Multifunkciós kártya** egyik LED-jét vezéreljék!
- Vegyük figyelembe, hogy:
  - A Multifunkciós kártyán lehúzásra világítanak a LED-ek
  - Csak a *D4* és a *D3* LED csatlakozik **PWM** kimenetre (**D10** és **D11** kivezetések)
- Az alábbi programban mi most a *D3* LED-et (**D11** kivezetést) választottuk



# MF_led_fade.ino

- A **D11** kimenetre kötött LED katódját vezéreljük

```
#define led      11      // D11-re van kötve a LED
int brightness = 0;      // fényerő értéke
int fadeAmount = 5;      // a változás léptéke

void setup() {
  setup_7seg();          // Multifunkciós kártya inicializálás
  pinMode(led, OUTPUT); // D11 legyen kimenet
}

void loop() {
  // katódvezérlés miatt komplementáljuk brightness értékét
  analogWrite(led, 255 - 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 ;
  }
  delay(50);
}
```

Folytatás a következő oldalon

# MF_led_fade.ino

- Az alábbi kiegészítés (a **`setup_7seg()`** függvény) csupán azt a célt szolgálja, hogy inicializálja és elsötétítse a Multifunkciós kártya hétszegmenses kijelzőjét, hogy ezzel a zavaró fényeket kizárjuk

```
/*********************************************
   A Multifunkciós kártya 7-szegmens kijelzőjének
   inicializálása és elsötétítése
 *********************************************/
#define LATCH_DIO 4
#define CLK_DIO   7
#define DATA_DIO  8
void setup_7seg(void) {
  /* Vezérlő lábak konfigurálása */
  pinMode(LATCH_DIO, OUTPUT);
  pinMode(CLK_DIO, OUTPUT);
  pinMode(DATA_DIO, OUTPUT);
  digitalWrite(LATCH_DIO, LOW);
  shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, 0xFF);
  shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, 0x00 );
  digitalWrite(LATCH_DIO, HIGH);
}
```

# MF_led_fade_exp.ino

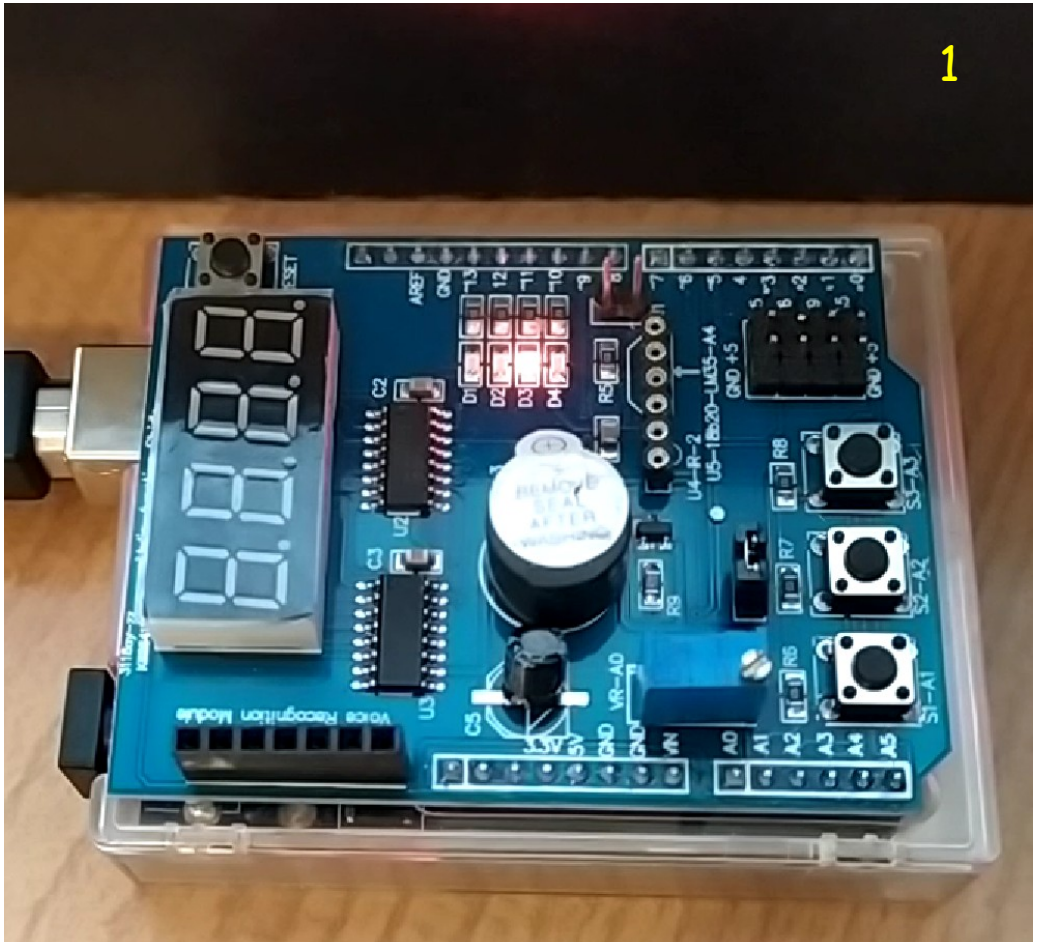
```
#define led      11            // D11-re van kötve a LED
uint16_t idx = 0;              // Index a kitöltési tényezők számításához
uint16_t next_sqr = 1;         // A következő négyzetszám
uint16_t sqr_step = 3;         // Növekmény a következő négyzetszám
kiszámításához

void setup() {
  pinMode(led, OUTPUT);        // D11 legyen kimenet
  setup_7seg();                // Multifunkciós kártya inicializálás
}
```

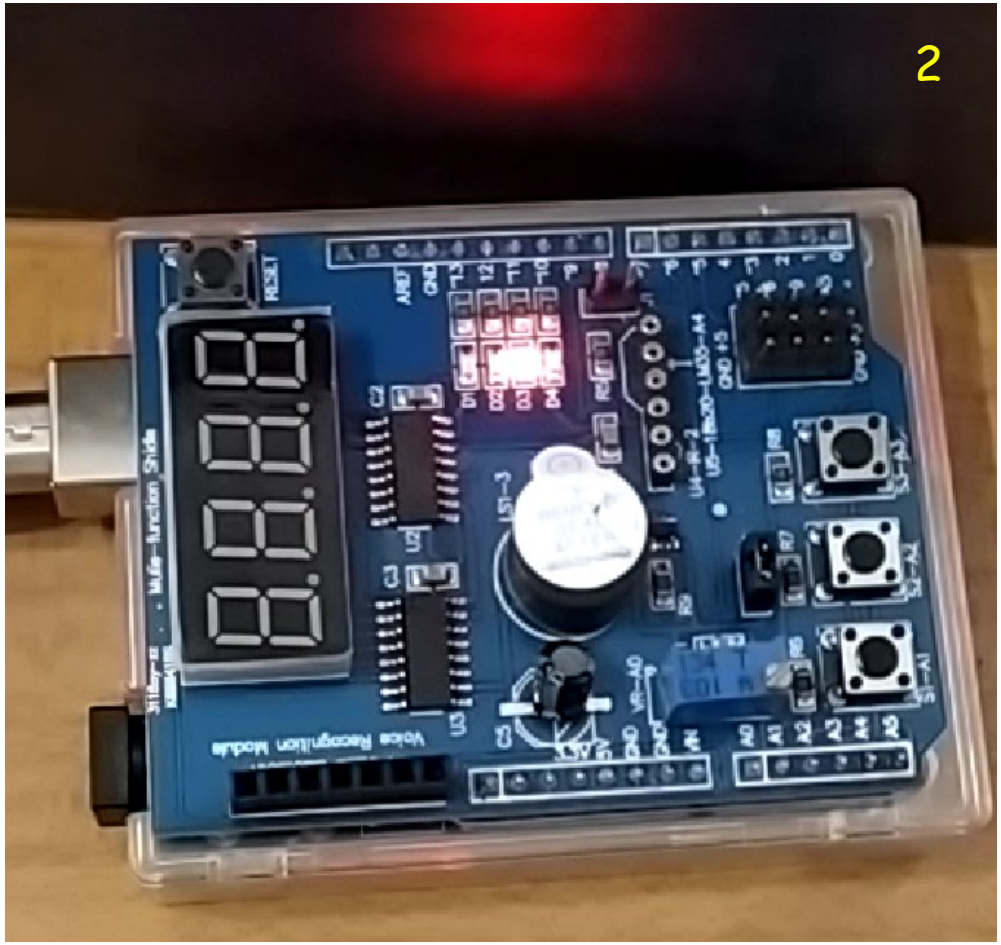
setup_7seg() ugyanaz, mint az előző programban, ezért itt nem részletezzük

```
void loop() {
  analogWrite(led, 255 - (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);
}
```

# Futási eredmények

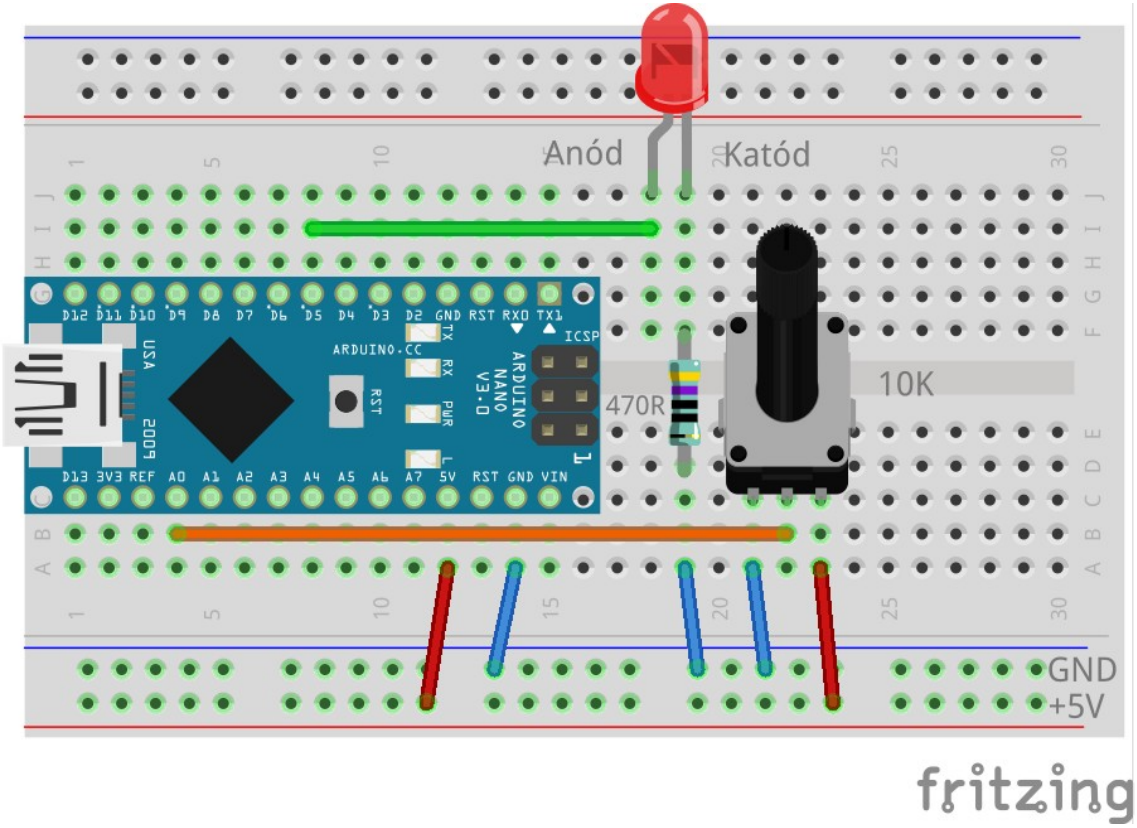
- A videókon talán nem látszik, de a nemlineáris vezérlés látványosabb



# 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 8-cal osztjuk az ADC-ből kapott értékeket, s osztás helyet jobbraléptetést végzünk

```
const int led = 5;       // ide van kötve a LED

void setup() {
  // Nincs tennivaló
}

void loop() {
  int reading = analogRead(A0);
  analogWrite(led, reading>>3 );  // Osztás 8-cal
  // Várunk, hogy a szemünk is láthassa a változást:
  delay(50);
}
```

# MF_led_pwm.ino

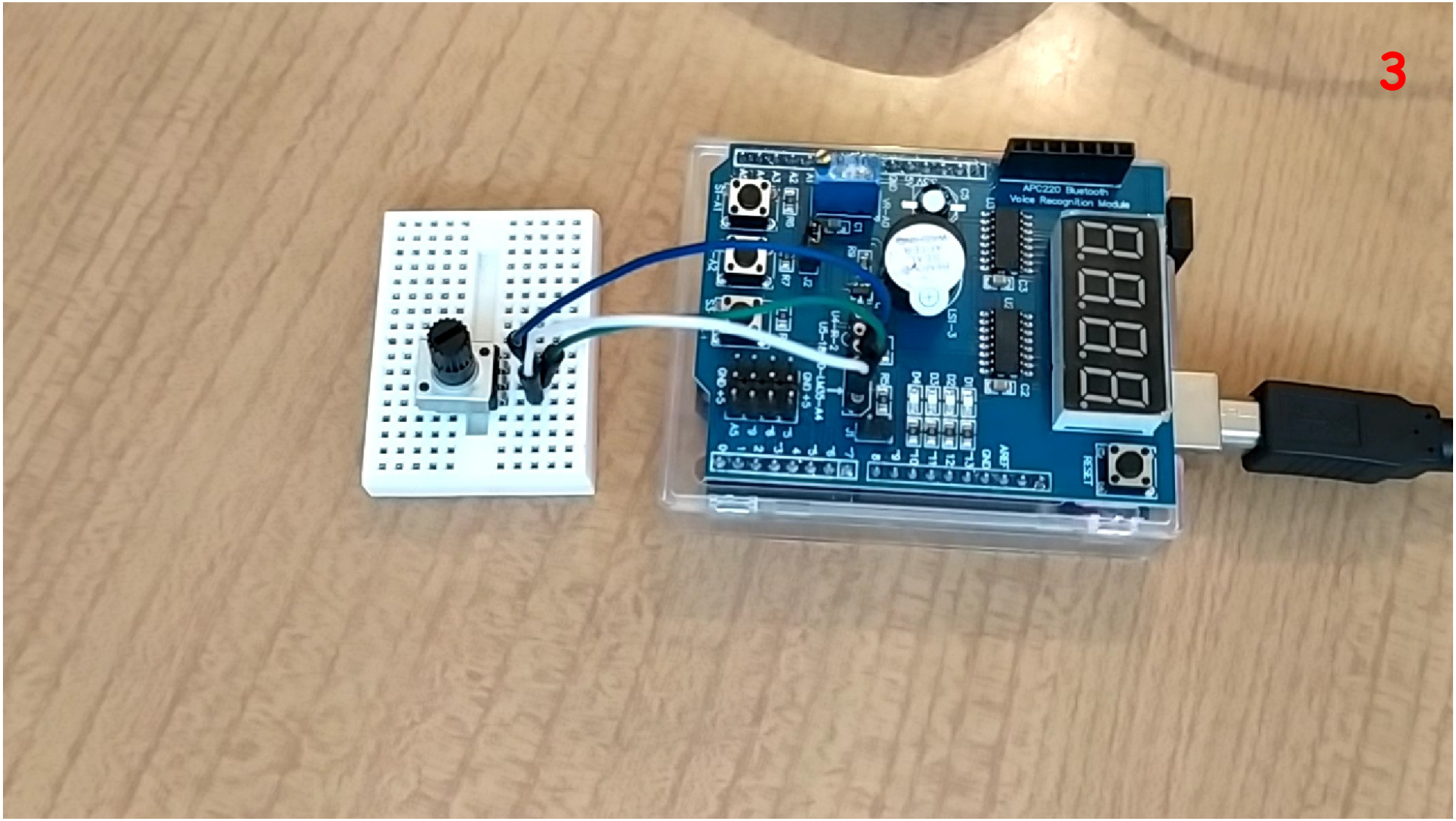
- A potméteres szabályozásnál is érdemes foglalkozni a nemlineáris vezérléssel. A hangerőszabályozáshoz hasonlóan próbálkozhatunk logaritmikus potenciométerrel, vagy valamilyen szoftveres korrekcióval
- Az **MF_led_pwm.ino** programban egy korrekciós táblázatot fogunk használni, emellett a Multifunkciós kártyára is adaptáljuk az előző programot
- Most a **D10** kivezetésre kötött *D4* LED-et vezéreljük (a katódja csatlakozik a **D10** kivezetésre)
- A Multifunkciós kártya ugyan tartalmaz egy beépített potmétert, ami az **A0** analóg bemenetre csatlakozik, de az sok fordulatú trimmer, ami nem alkalmas a tervezett célra. Ezért a hőmérő számára kialakított csatlakozó hüvelysorra kötöttünk egy 10 kΩ-os potmétert (ami az **A4** bemenetre csatlakozik)

# MF_led_pwm.ino

```
const byte dim_curve[] = {
  0,   1,   1,   2,   2,   2,   2,   2,   2,   3,   3,   3,   3,   3,   3,   3,
  3,   3,   3,   3,   3,   3,   3,   4,   4,   4,   4,   4,   4,   4,   4,   4,
  4,   4,   4,   5,   5,   5,   5,   5,   5,   5,   5,   5,   5,   6,   6,   6,
  6,   6,   6,   6,   6,   7,   7,   7,   7,   7,   7,   7,   8,   8,   8,   8,
  8,   8,   9,   9,   9,   9,   9,   9,   10,  10,  10,  10,  10,  11,  11,  11,
  11,  11,  12,  12,  12,  12,  12,  13,  13,  13,  13,  14,  14,  14,  14,  15,
  15,  15,  16,  16,  16,  16,  17,  17,  17,  18,  18,  18,  19,  19,  19,  20,
  20,  20,  21,  21,  22,  22,  22,  23,  23,  24,  24,  25,  25,  25,  26,  26,
  27,  27,  28,  28,  29,  29,  30,  30,  31,  32,  32,  33,  33,  34,  35,  35,
  36,  36,  37,  38,  38,  39,  40,  40,  41,  42,  43,  43,  44,  45,  46,  47,
  48,  48,  49,  50,  51,  52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,
  63,  64,  65,  66,  68,  69,  70,  71,  73,  74,  75,  76,  78,  79,  81,  82,
  83,  85,  86,  88,  90,  91,  93,  94,  96,  98,  99,  101, 103, 105, 107, 109,
  110, 112, 114, 116, 118, 121, 123, 125, 127, 129, 132, 134, 136, 139, 141, 144,
  146, 149, 151, 154, 157, 159, 162, 165, 168, 171, 174, 177, 180, 183, 186, 190,
  193, 196, 200, 203, 207, 211, 214, 218, 222, 226, 230, 234, 238, 242, 248, 255,
};
#define led  10         // Ide van kötve a LED
#define potm A4         // Ide van kötve a potméter
void setup() {
  setup_7seg();         // Multifunkciós kártya inicializálás
}

void loop() {
  int reading = analogRead(potm); // a katódvezérlés miatt komplementálni kell
  analogWrite(led, 255 - dim_curve[reading>>2]);
  delay(50);
}
```

setup_7seg() ugyanaz, mint az korábbi programokban, itt nem részletezzük

# MF_led_pwm: futási eredmény



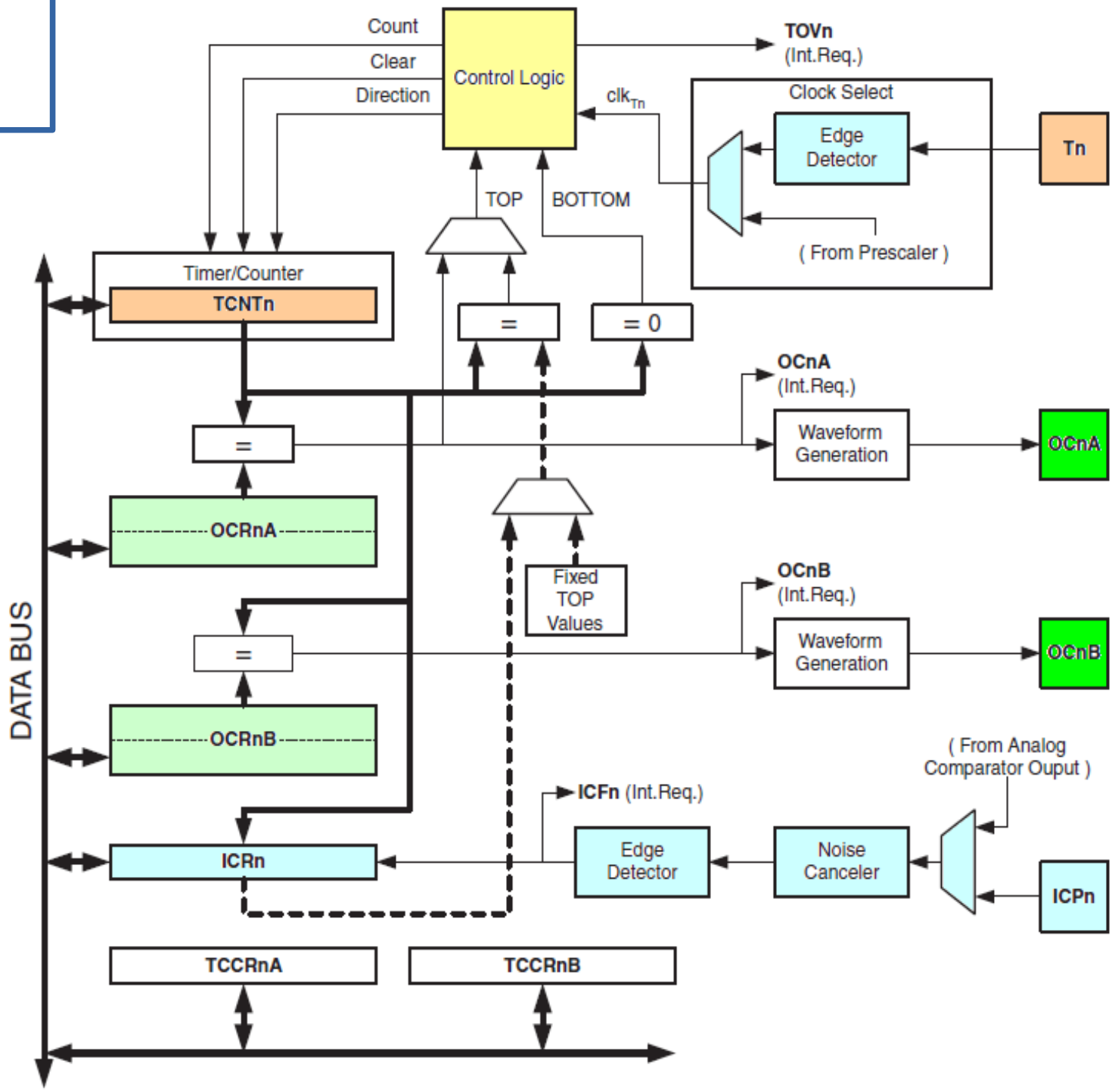
# 16-bites PWM Timer1 használatával

- Az **Arduino** könyvtári függvények implicit módon mindegyik **PWM** csatornát 8 bites módba állítják, bár ez a felbontás nem minden esetben kielégítő. Szerencsére nem ez a maximális felbontás, mert **Timer1** akár 16 bites felbontással is használható
- Az **Atmega328** mikrovezérlő esetében csak **Timer1** az egyetlen 16 bites időzítő melynek két **PWM** csatornája van, melyek kimenetei a **D9** és **D10** kivezetéseken érhetők el
- A továbbiakban azt mutatjuk be, hogy hogyan valósíthatjuk meg és használhatjuk a 16 bites üzemmódot
- Az Arduino kártyákon rendelkezésre álló 16 bites **PWM** csatornák száma:
  - **ATmega328** – Timer1: 2 csatorna *( Arduino Uno, nano)*
  - **ATmega32U4** – Timer1: 2 csatorna, Timer3: 1 csatorna *(Arduino Pro Micro)*
  - **ATmega2560** – Timer1: 2 csatorna, Timer3, 4 és 5: 3-3 csatorna *(Arduino Mega)*

# Timer1 blokkvázlata

## Jellemzők:

- 16-bites számláló
- Belső órajel, vagy külső jel fogadása
- 2 PWM csatorna
- Kettős bufferelés
- CTC mód *(Clear on Terminal Count)*
- Él igazított és fázishelyes PWM
- Bemeneti jelfogás
- Megszakítási források: (TOV1, ICF1, OCF1A, és OCF1B)



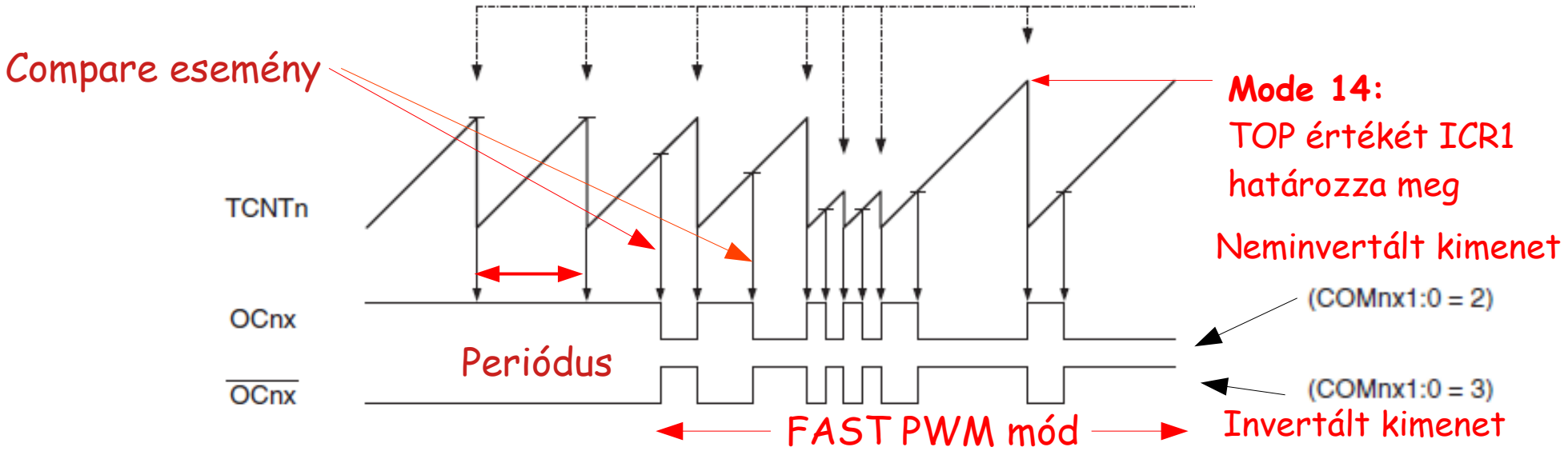
# Timer1 regiszterek

- **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). |



# 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)
- **ICES1** – 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

- A 14-es módban (16-bit fast PWM) az **ICR1** regiszter értéke szabja meg a számlálás felső határát (**TOP**)
- **Megjegyzés:** a 16 bites regiszterek egyetlen, `uint16_t` típusú egységnek is tekinthetők, ekkor **TCNT1,** illetve **ICR1** írandó!
- 16 bites felbontásnál a maximális PWM frekvencia (N = 1):

$$f_{PWM} = \frac{f_{CPU}}{N \cdot 65536} = \frac{16\,000\,000\ Hz}{65536} = 244.14\ Hz$$

**TCNT1H and TCNT1L – Timer/Counter1**

Ez a számláló regiszter

| 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**

Ez szabja majd meg a periódust

| 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

- Ez a két regiszter szabja meg a kitöltést. A beírt érték ne legyen nagyobb a periódusnál (az **ICR1**-be írt számnál)!
- **OCR1A** a **D9** kivezetéshez tartozó jel kitöltését szabályozza
- **OCR1B** a **D10** kivezetéshez tartozó jel kitöltését szabályozza

**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 | |

# MF_breathingLED_16.ino

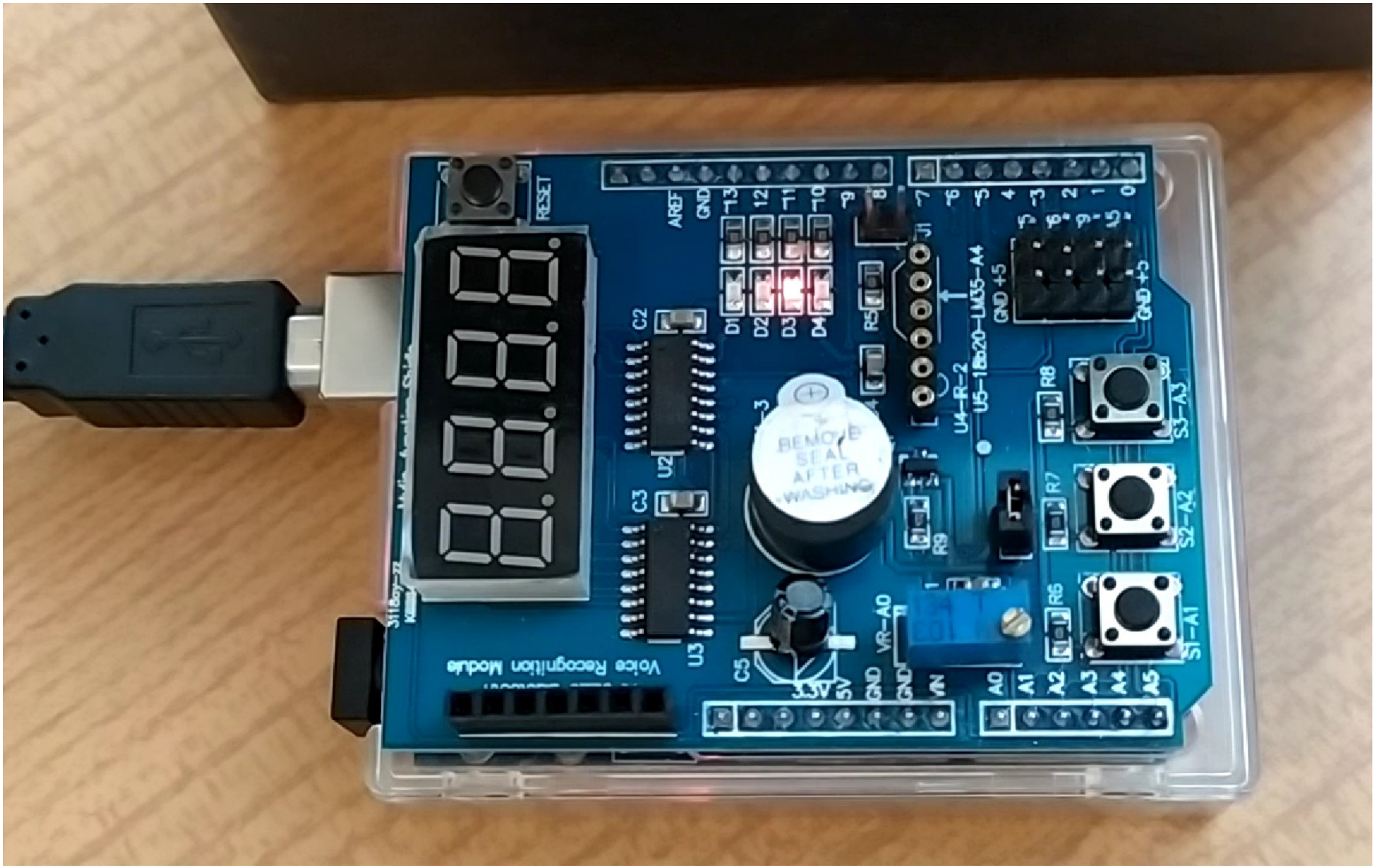
- Újabb „lélegző” LED, ezúttal 16 bites PWM-mel
- A Multifunkciós kártya **D10** kivezetését (*D4* LED katód) vezéreljük

```
const uint16_t MAX_PWM_VALUE = 65535U;

void setup() {
  setup_7seg();                          // Multifunkciós kártya inicializálás
  pinMode(10, OUTPUT);
  TCCR1A = bit(COM1B0) | bit(COM1B1) | // Inverting PWM (D10 katódvezérléshez)
           bit(WGM11);                   // Mode 14: Fast PWM, TOP=ICR1
  TCCR1B = bit(WGM13) | bit(WGM12) |
           bit(CS10);                    // Prescaler 1
  ICR1 = MAX_PWM_VALUE;                  // TOP counter value
}

void loop() {
  for (uint16_t i = 0; i < 1023; i++) {
    OCR1B = pow(i, 1.6002234458);
    delay(2);
  }
  for (uint16_t i = 0; i < 1023; i++) {
    OCR1B = pow(1023 - i, 1.6002234458);
    delay(2);
  }
}
```

- **pow(*alap, kitevő*)**
  az *alap* adott *kitevőjű* hatványát számolja ki
  ahol a kitevő törtszám is lehet

`setup_7seg()` ugyanaz, mint az korábbi programokban, ezért itt nem részletezzük

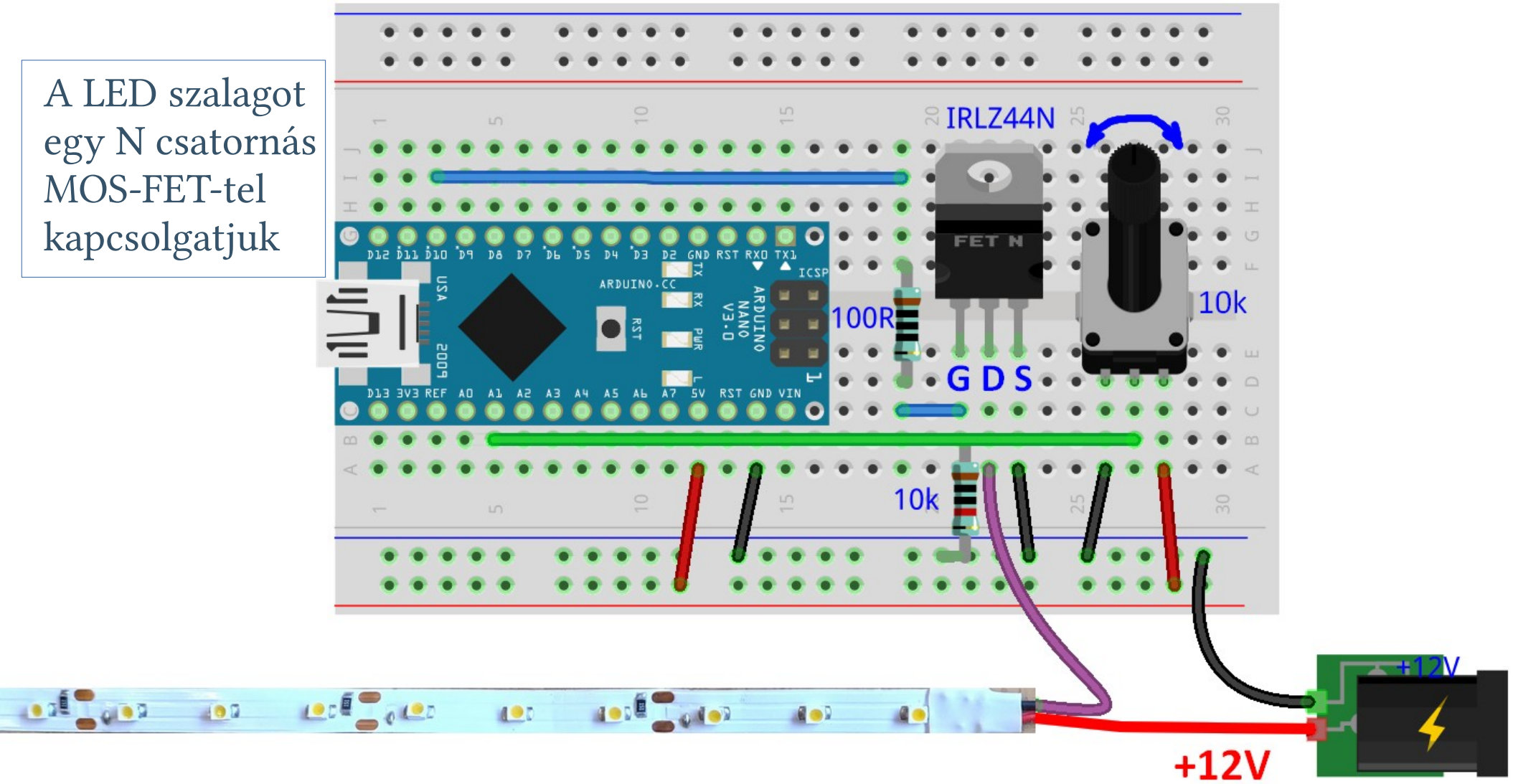
# MF_breathingLED_16 futási eredmény



# LED_PWM16

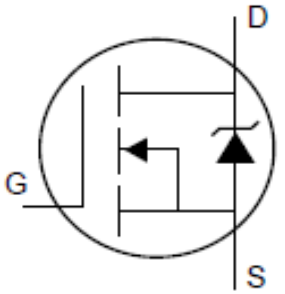
- Az **A1** bemenetre kötött potméterrel vezéreljük egy LED szalag fényerejét, a **D10**-es kimeneten, 16 bites **PWM**-mel

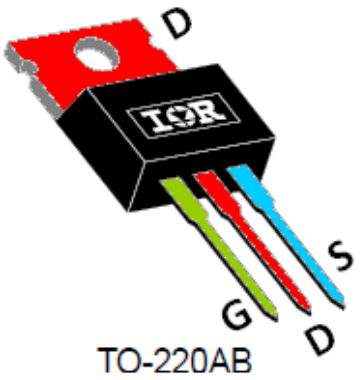
A LED szalagot egy N csatornás MOS-FET-tel kapcsolgatjuk



# IRLZ44N power FET

- Nagyteljesítményű: 55 V, 47 A HEXFET, n-csatornás, növekményes
- Kis maradékellenállású: $R_{DS(on)}$ = 0.022 Ω
- Kis nyitófeszültségű („logikai” FET): 4-5 V
- TO-220 tokozású (hűthető)
- Gyártó: **International Rectifier**



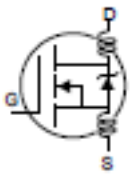
TO-220AB

**Absolute Maximum Ratings**

| | Parameter | Max. | Units |
|---|---|---|---|
| $I_D$ @ $T_C$ = 25°C | Continuous Drain Current, $V_{GS}$ @ 10V | 47 | A |
| $I_D$ @ $T_C$ = 100°C | Continuous Drain Current, $V_{GS}$ @ 10V | 33 | |
| $I_{DM}$ | Pulsed Drain Current ① | 160 | |
| $P_D$ @$T_C$ = 25°C | Power Dissipation | 110 | W |
| | Linear Derating Factor | 0.71 | W/°C |
| $V_{GS}$ | Gate-to-Source Voltage | ±16 | V |
| $E_{AS}$ | Single Pulse Avalanche Energy ② | 210 | mJ |
| $I_{AR}$ | Avalanche Current① | 25 | A |
| $E_{AR}$ | Repetitive Avalanche Energy① | 11 | mJ |
| dv/dt | Peak Diode Recovery dv/dt ③ | 5.0 | V/ns |
| $T_J$ | Operating Junction and | -55 to + 175 | °C |
| $T_{STG}$ | Storage Temperature Range | | |
| | Soldering Temperature, for 10 seconds | 300 (1.6mm from case) | |
| | Mounting torque, 6-32 or M3 screw. | 10 lbf•in (1.1N•m) | |

## Electrical Characteristics @ $T_J$ = 25°C (unless otherwise specified)

| | Parameter | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| $V_{(BR)DSS}$ | Drain-to-Source Breakdown Voltage | 55 | — | — | V | $V_{GS}$ = 0V, $I_D$ = 250µA |
| $\Delta V_{(BR)DSS}/\Delta T_J$ | Breakdown Voltage Temp. Coefficient | — | 0.070 | — | V/°C | Reference to 25°C, $I_D$ = 1mA |
| $R_{DS(on)}$ | Static Drain-to-Source On-Resistance | — | — | 0.022 | Ω | $V_{GS}$ = 10V, $I_D$ = 25A ④ |
| | | — | — | 0.025 | | $V_{GS}$ = 5.0V, $I_D$ = 25A ④ |
| | | — | — | 0.035 | | $V_{GS}$ = 4.0V, $I_D$ = 21A ④ |
| $V_{GS(th)}$ | Gate Threshold Voltage | 1.0 | — | 2.0 | V | $V_{DS}$ = $V_{GS}$, $I_D$ = 250µA |
| $g_{fs}$ | Forward Transconductance | 21 | — | — | S | $V_{DS}$ = 25V, $I_D$ = 25A |
| $I_{DSS}$ | Drain-to-Source Leakage Current | — | — | 25 | µA | $V_{DS}$ = 55V, $V_{GS}$ = 0V |
| | | — | — | 250 | | $V_{DS}$ = 44V, $V_{GS}$ = 0V, $T_J$ = 150°C |
| $I_{GSS}$ | Gate-to-Source Forward Leakage | — | — | 100 | nA | $V_{GS}$ = 16V |
| | Gate-to-Source Reverse Leakage | — | — | -100 | | $V_{GS}$ = -16V |
| $Q_g$ | Total Gate Charge | — | — | 48 | nC | $I_D$ = 25A |
| $Q_{gs}$ | Gate-to-Source Charge | — | — | 8.6 | | $V_{DS}$ = 44V |
| $Q_{gd}$ | Gate-to-Drain ("Miller") Charge | — | — | 25 | | $V_{GS}$ = 5.0V, See Fig. 6 and 13 ④ |
| $t_{d(on)}$ | Turn-On Delay Time | — | 11 | — | ns | $V_{DD}$ = 28V |
| $t_r$ | Rise Time | — | 84 | — | | $I_D$ = 25A |
| $t_{d(off)}$ | Turn-Off Delay Time | — | 26 | — | | $R_G$ = 3.4Ω, $V_{GS}$ = 5.0V |
| $t_f$ | Fall Time | | 15 | | | $R_D$ = 1.1Ω, See Fig. 10 ④ |
| $L_D$ | Internal Drain Inductance | — | 4.5 | — | nH | Between lead, 6mm (0.25in.) from package and center of die contact |
| $L_S$ | Internal Source Inductance | — | 7.5 | — | | |
| $C_{iss}$ | Input Capacitance | — | 1700 | — | pF | $V_{GS}$ = 0V |
| $C_{oss}$ | Output Capacitance | — | 400 | — | | $V_{DS}$ = 25V |
| $C_{rss}$ | Reverse Transfer Capacitance | — | 150 | — | | $f$ = 1.0MHz, See Fig. 5 |

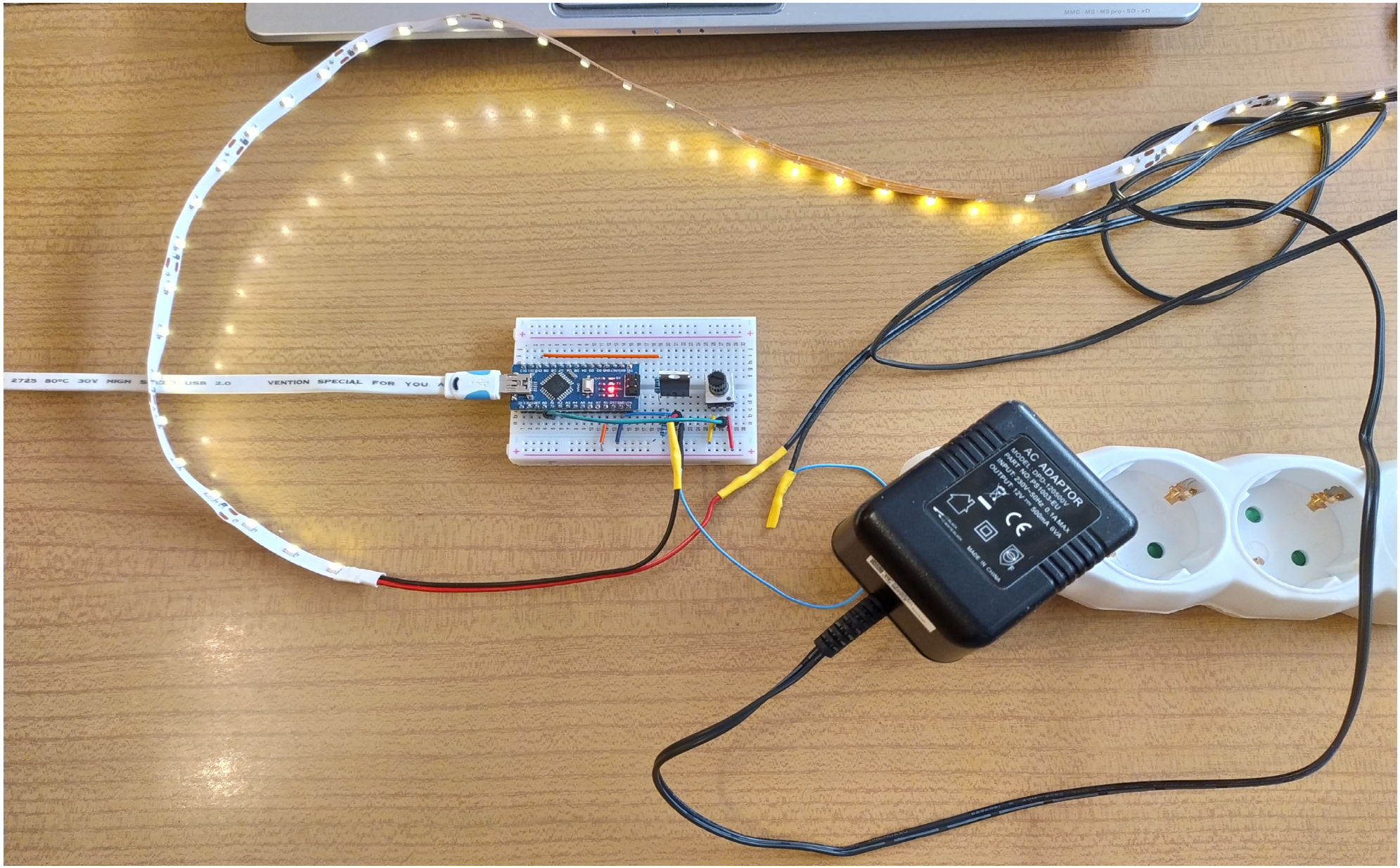


# LED_PWM16.ino

- A zaj kiátlagolása érdekében 256 mérés átlagát vesszük
- A hatványfüggvény nemlineáris vezérlést biztosít a fényforráshoz

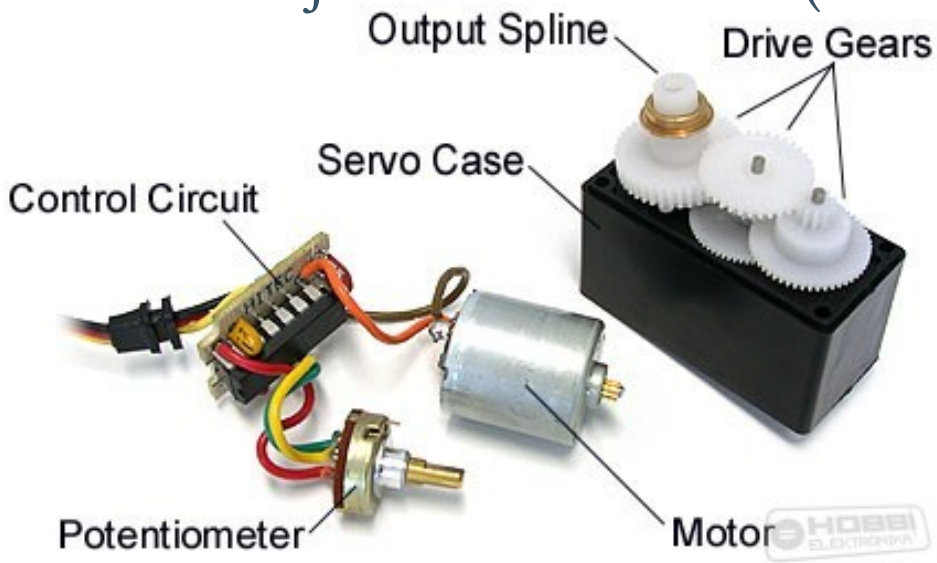
```
const uint16_t MAX_PWM_VALUE = 65535;

void setup() {
  pinMode(10, OUTPUT);
  TCCR1A = bit(COM1B1) |                  // Non-inv PWM
           bit(WGM11);                    // Mode 14: Fast PWM, TOP=ICR1
  TCCR1B = bit(WGM13) | bit(WGM12) |
           bit(CS10);                     // Prescaler 1
  ICR1 = MAX_PWM_VALUE;                   // TOP counter value
}

void loop() {
  uint32_t sum = 0;
  for (uint16_t i = 0; i < 256; i++) {
    sum += analogRead(A1);
  }
  uint16_t val = sum/256;
  OCR1B = pow(val, 1.6002234458);
  delay(20);
}
```

# LED_PWM16: futási eredménye

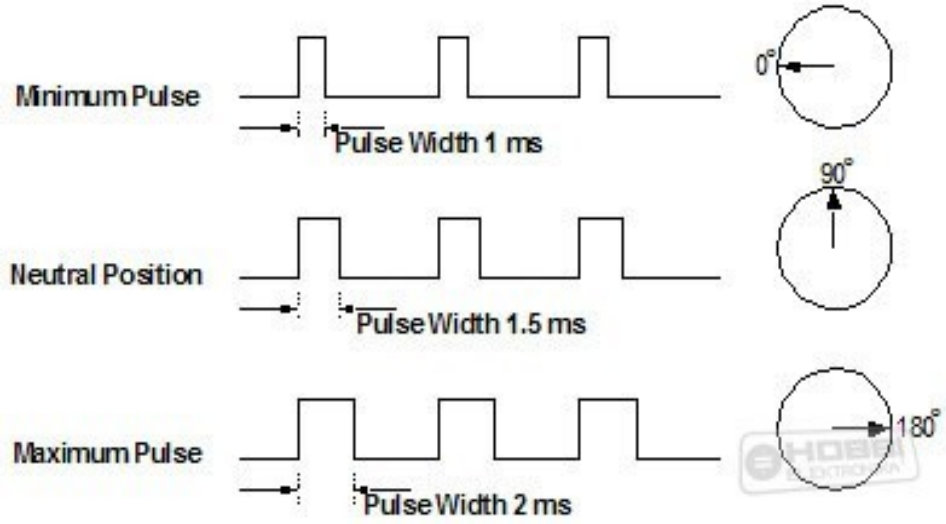


# Szervo motorok

- A szervo egy pozícionálható motor, amely „ismeri” az aktuális pozícióját, és a cél pozíciót. Feladata, hogy az aktuális pozícióból a kívánt pozícióba álljon
- Felépítése: vezérlő áramkör, mechanikusan összekapcsolt motor és potenciométer (a potméterrel leosztott feszültség jelzi a pozíciót)
- Általában 180 °-os tartományban mozgatható, 1 – 2 ms szélességű, 50 Hz-es jellel vezérelhető (PWM)

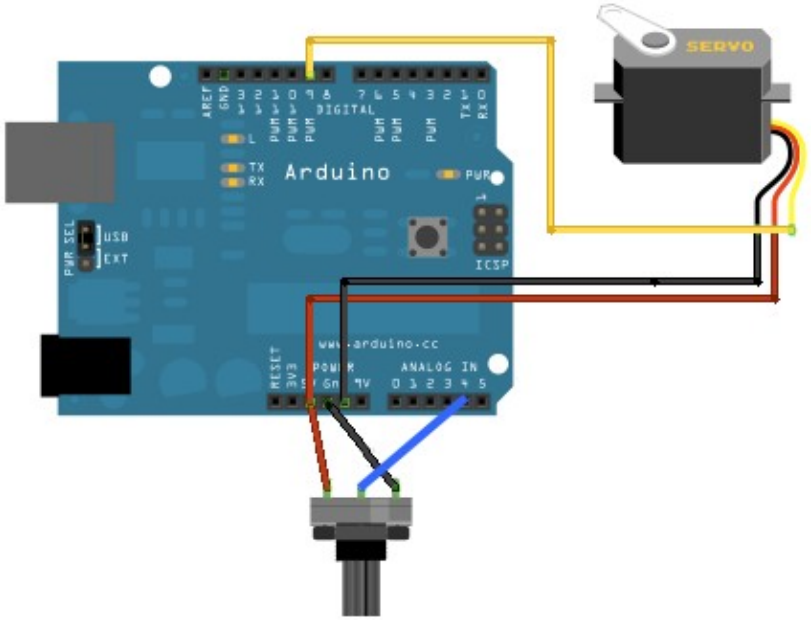


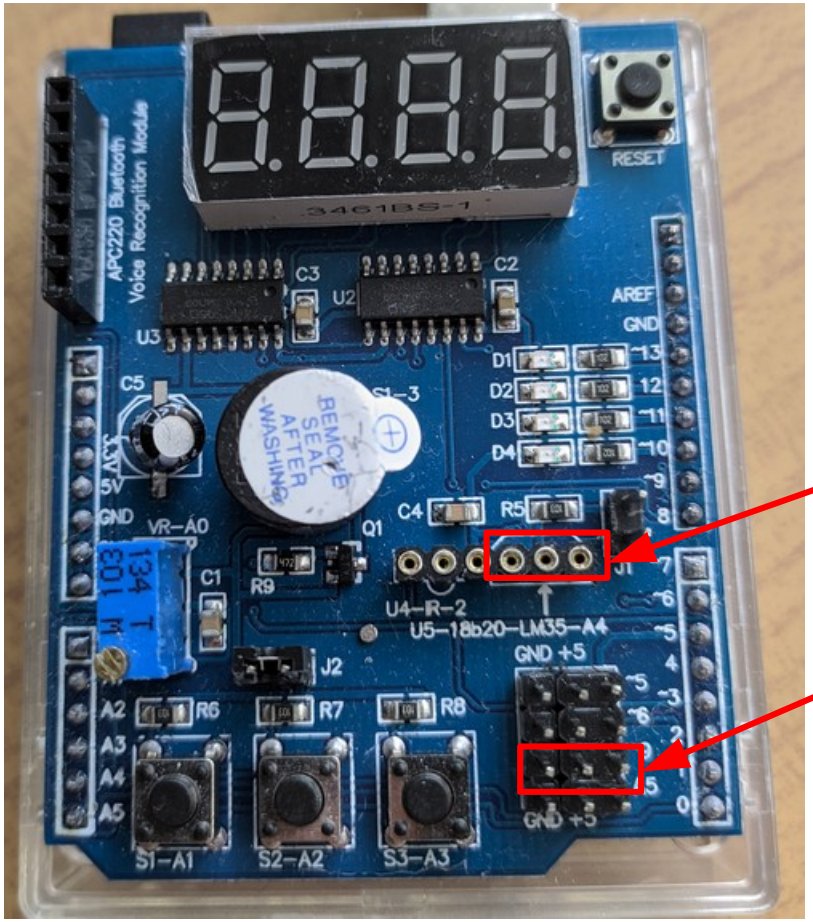

Felépítés




Jelalak

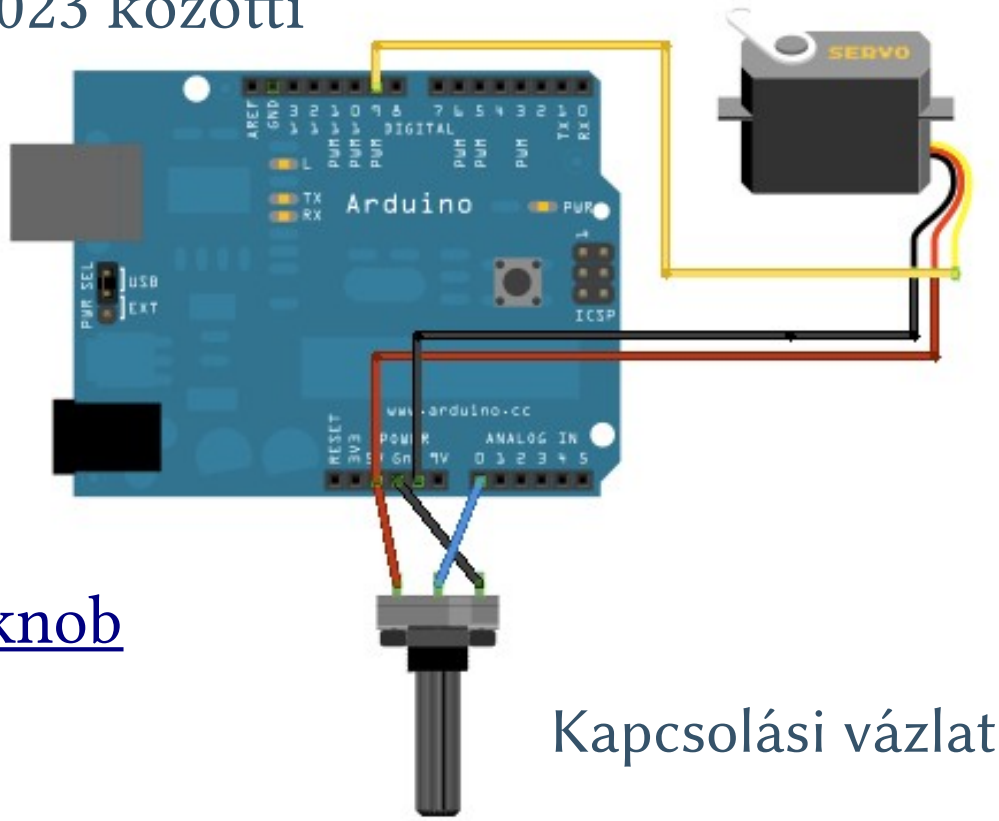
# Bekötési vázlat

# Servo vezérlés Timer1 PWM16 módban

- Ez a beépített mintaprogram (**File/Examples/Servo/Knob**) az **A0** bemenetre kötött potméterrel vezérli a szervót
- A 10 kΩ-os potméter csúszkáját az **A0** analóg bemenetre kötjük, a két végét padig **GND**-re, illetve az 5V-os tápfeszültségre
- Az **ADC** által visszaadott 0 – 1023 közötti számot át kell skálázni 1 – 180 közötti tartományba, ehhez a **map()** függvényt használjuk
- A szervo vezérléséhez a **Servo** könyvtárat használjuk
- Forrás: [arduino.cc/en/tutorial/knob](arduino.cc/en/tutorial/knob)



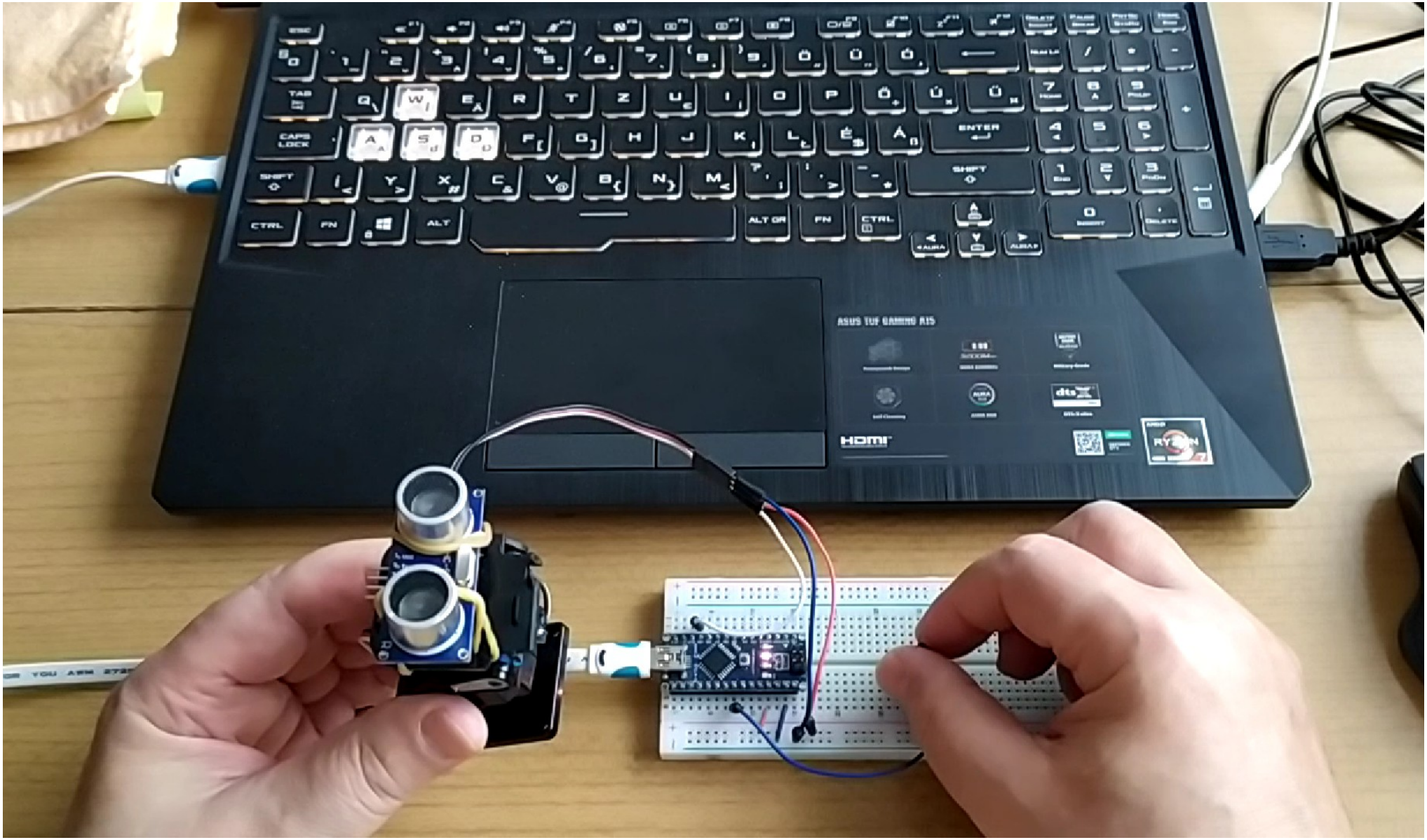

Kapcsolási vázlat

# servo_PWM16.ino

- Timer1 1:8 előosztással, 40 000-ig számlálva ad 20 ms periódust
- A szervók 1 – 2 ms jeléhez a kitöltés 2000 – 4000 legyen (néhány gyártó termékénél azonban 1000 – 5000 is kellhet)

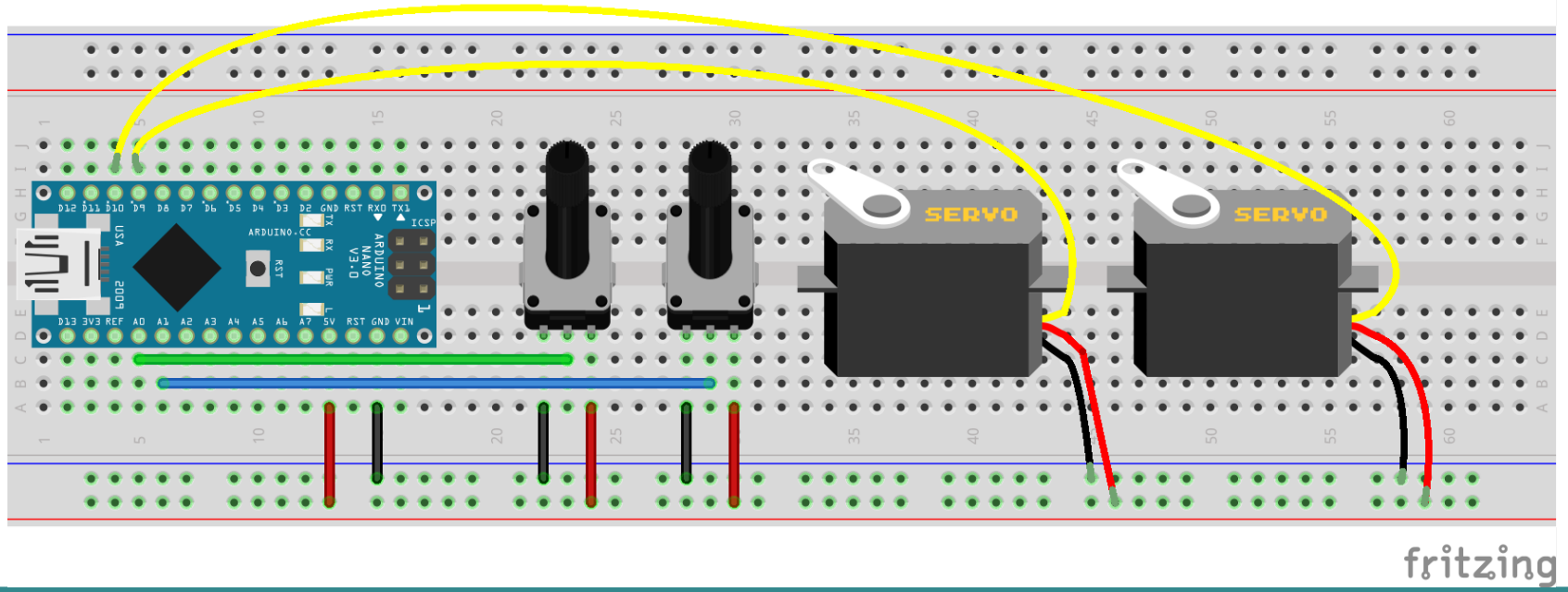
```
const uint16_t MAX_PWM_VALUE = 40000;  // Periódus = 20 ms

void setup() {
  pinMode(9, OUTPUT);
  TCCR1A = bit(COM1A1) |                // Non-inv PWM
           bit(WGM11);                  // Mode 14: Fast PWM, TOP=ICR1
  TCCR1B = bit(WGM13) | bit(WGM12) |
           bit(CS11);                   // Prescaler 1:8
  ICR1 = MAX_PWM_VALUE;                 // TOP counter value
}

void loop() {
  uint32_t sum = 0;
  for (uint16_t i = 0; i < 256; i++)  {
    sum += analogRead(A4);              //  Az A4-re kötött potméter jelét mérjük
  }
  OCR1A = sum / 128 + 2000;             //  2 * ADC + 2000
  sum = 0;
  delay(40);
}
```

# servo_PWM16: futási eredmény



# two_servos_PWM16.ino

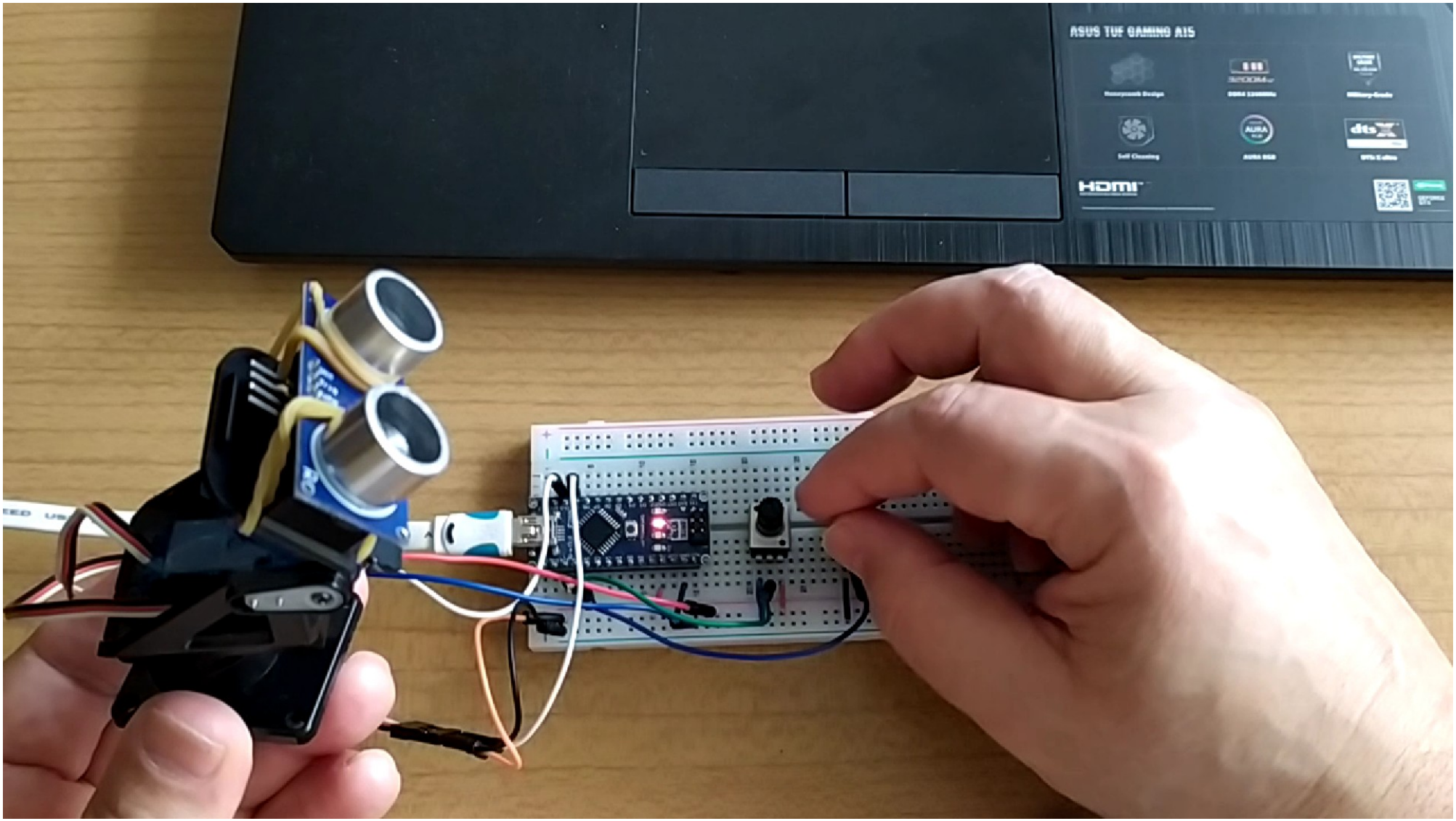
- Két szervót (**D9** és **D10**) vezérlünk két potméterrel (**A0** és **A1**)
- A 16 bites **PWM** kezeléséhez függvényeket készítünk:
  - **setupPWM16()** - **Timer1** 16 bites **PWM** módjának konfigurálása, valamint **D9** és **D10** kimenetre állítása
  - **analogWrite16(***pin, value***)** – a megadott kivezetésen (9 vagy 10) beállítja a kitöltés (duty) értékét
- Felhasznált irodalom:
  T.K. Hareendran: **Arduino & Advanced 16-bit PWM**



# two_servos_PWM16.ino

```
void setup() { setupPWM16(); }

void loop() {
  uint16_t duty = 0;
  duty = analogRead(A0) * 2 + 2000;
  analogWrite(9, duty);
  delay(25);
  duty = analogRead(A1) * 2 + 2000;
  analogWrite(10, duty);
  delay(25);
}

void setupPWM16() {
  DDRB |= bit(PB1) | bit(PB2);        // D9 & D10 output
  TCCR1A = bit(COM1A1) | bit(COM1B1) // Non-Inv PWM
         | bit(WGM11);                // Mode 14: Fast PWM, TOP=ICR1
  TCCR1B = bit(WGM13) | bit(WGM12)
         | bit(CS11);                 // Prescaler 1:8
  ICR1 = 40000;                       // TOP counter value (Relieving OCR1A*)
}

void analogWrite16(uint8_t pin, uint16_t val) {
  switch (pin) {
    case 9: OCR1A = val; break;
    case 10: OCR1B = val; break;
  }
}
```

# two_servos_PWM16: futási eredmény



# Az Arduino nano kártya kivezetései