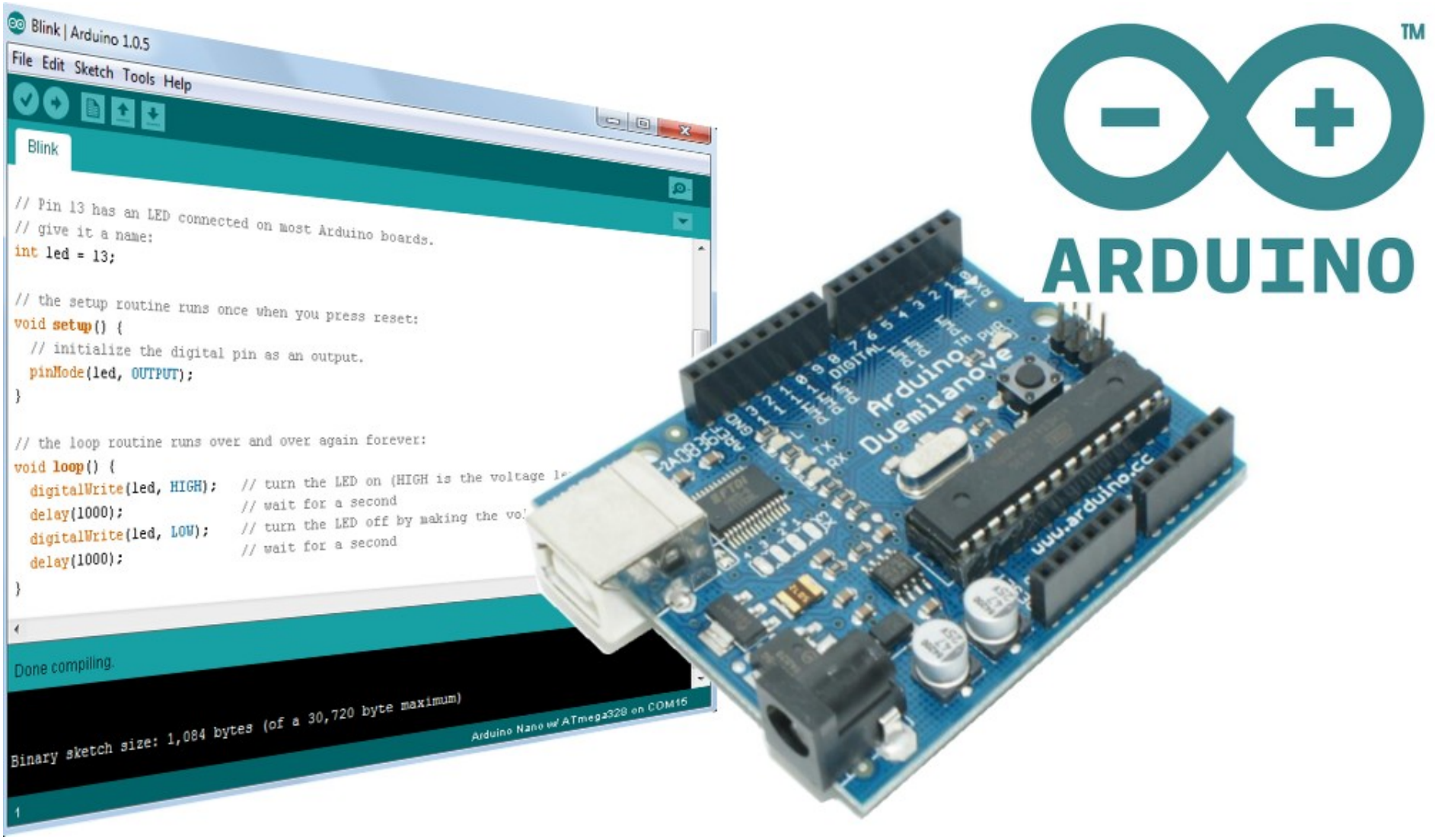
Arduino tanfolyam kezdőknek és haladóknak



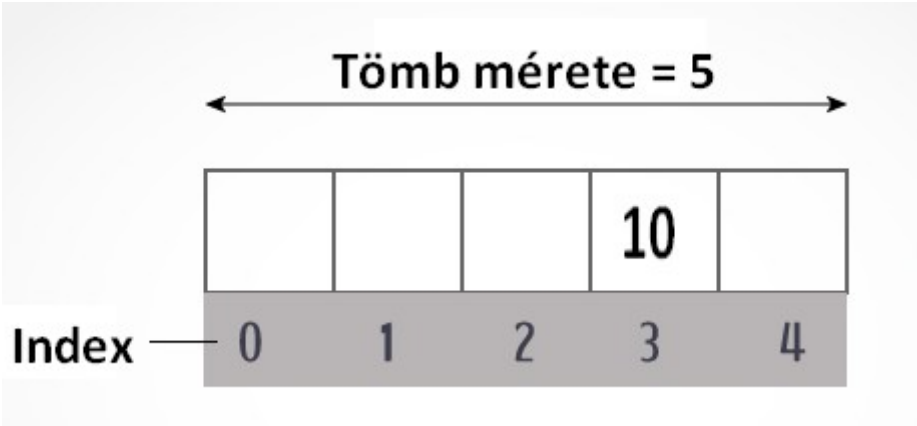
3. Tömbök, programciklusok, megszakítások

Tömbök

- A tömbváltozó közös névvel ellátott értékek/számok gyűjteménye, melyeket az index (sorszám) szerint érhetünk el. A tömbök indexelése a 0 sorszámmal kezdődik.
- A tömböt - a többi változóhoz hasonlóan - használat előtt deklarálni kell és lehetőség szerint a kezdőértékkel feltölteni, például:

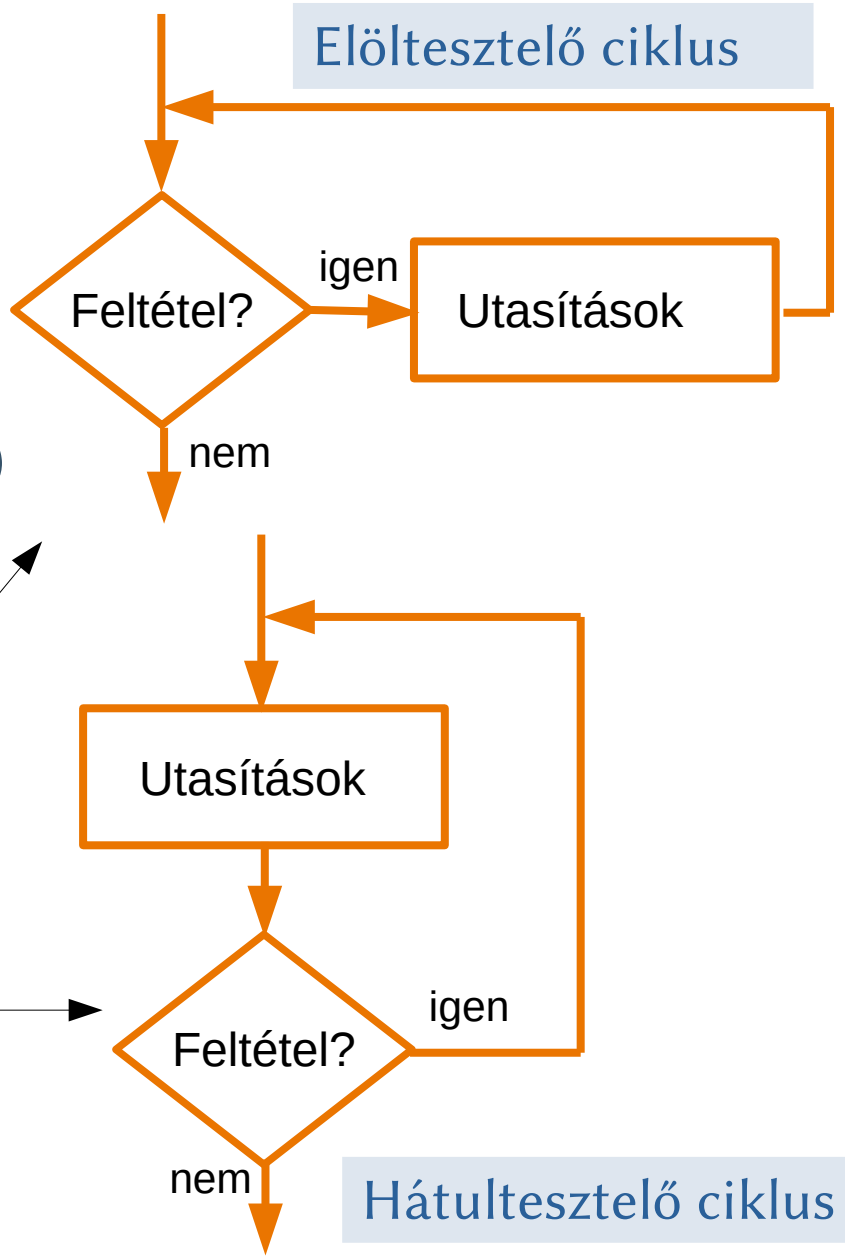
```
int a[5];      // Egy 5 elemű, int típusú tömböt deklarálunk
a[3] = 10;     // Az a tömb 4. eleme 10 lesz
int x = a[3];      // Az x változó értéke 10 lesz
```

- A tömbök egyik haszna az, hogy programciklusban egy számlálót növelgetve végig tudjuk futtatni a tömbelemek sorszámain (lásd majd a **for** ciklusnál!)
- A tömbök másik haszna az, hogy függvényeknek történő paraméter-átadásnál elegendő a tömb nevét és az elemek számát megadni



Programciklusok szervezése

- Ismétlődő feladatok esetén az utasításainkat „újrahasznosíthatjuk” **ciklusok** szervezésével, amikor egy utasításblokkot többször lefuttatunk
- Nem (csak) végtelen ciklust akarunk, ezért kell bennmaradási (vagy kilépési) **feltétel**
- A feltételt az utasításblokk végrehajtása előtt vagy után is vizsgálhatjuk:
  - Előltesztelő ciklus: **while, for**
  - Hátultesztelő ciklus: **do ... while**



Ciklusszervezés `for` és `while` utasítással

```
for (int i=0; i<5; i++) { utasítások }
```

A **for** utasítás is elöltesztelő ciklust szervez

1) Az első kifejezésben szereplő változó felveszi a kezdőértéket
2) Kiértékelésre kerül a második kifejezés, s teljesülés (nem nulla érték) esetén végrehajtásra kerül a ciklus törzse
3) A ciklustörzs lefutása után végrehajtásra kerül a harmadik kifejezésben szereplő művelet, majd visszatérünk a 2. ponthoz

Kezdőérték feltétel ciklusváltozó léptetés

```
for(int i = 0; i < 5; i++) {
  digitalWrite(LED,HIGH);
  delay(50);
  digitalWrite(LED,LOW);
  delay(500);
}
```

FOR ciklus

Kezdőérték
Feltétel

```
int i = 0;
while (i < 5) {
  digitalWrite(LED,HIGH);
  delay(50);
  digitalWrite(LED,LOW);
  delay(500);
  i++;
}
```

Ciklusváltozó léptetés

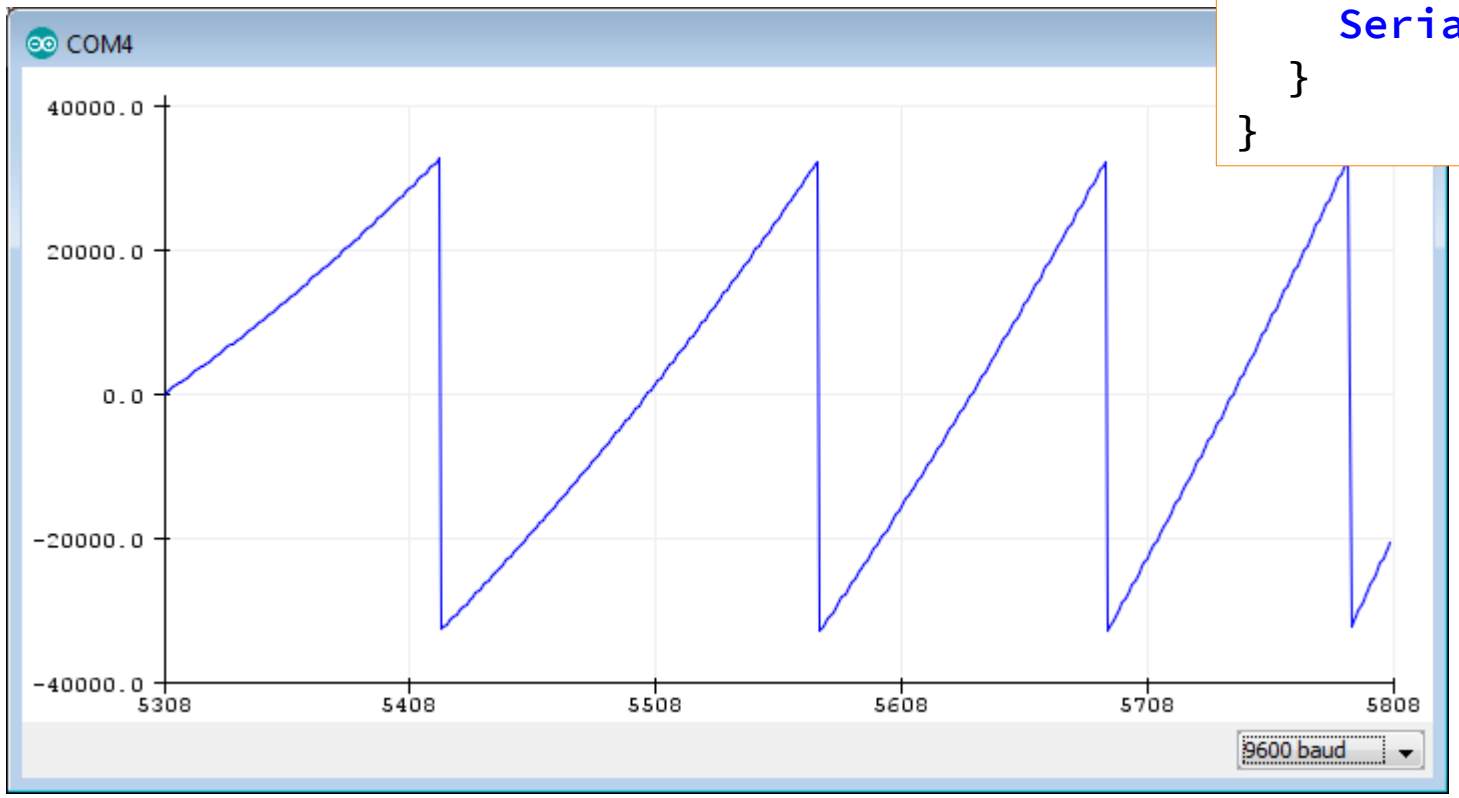
WHILE ciklus

Megjelenítés a serial plotter ablakban

- Az **USB – soros átalakítón** keresztül kapcsolatban vagyunk a számítógéppel, használjuk ki ezt a lehetőséget az eredmények megjelenítésére!
- Nyissuk meg a **Tools** menüben található **Serial Plotter** alkalmazást!

serial_plotter.ino

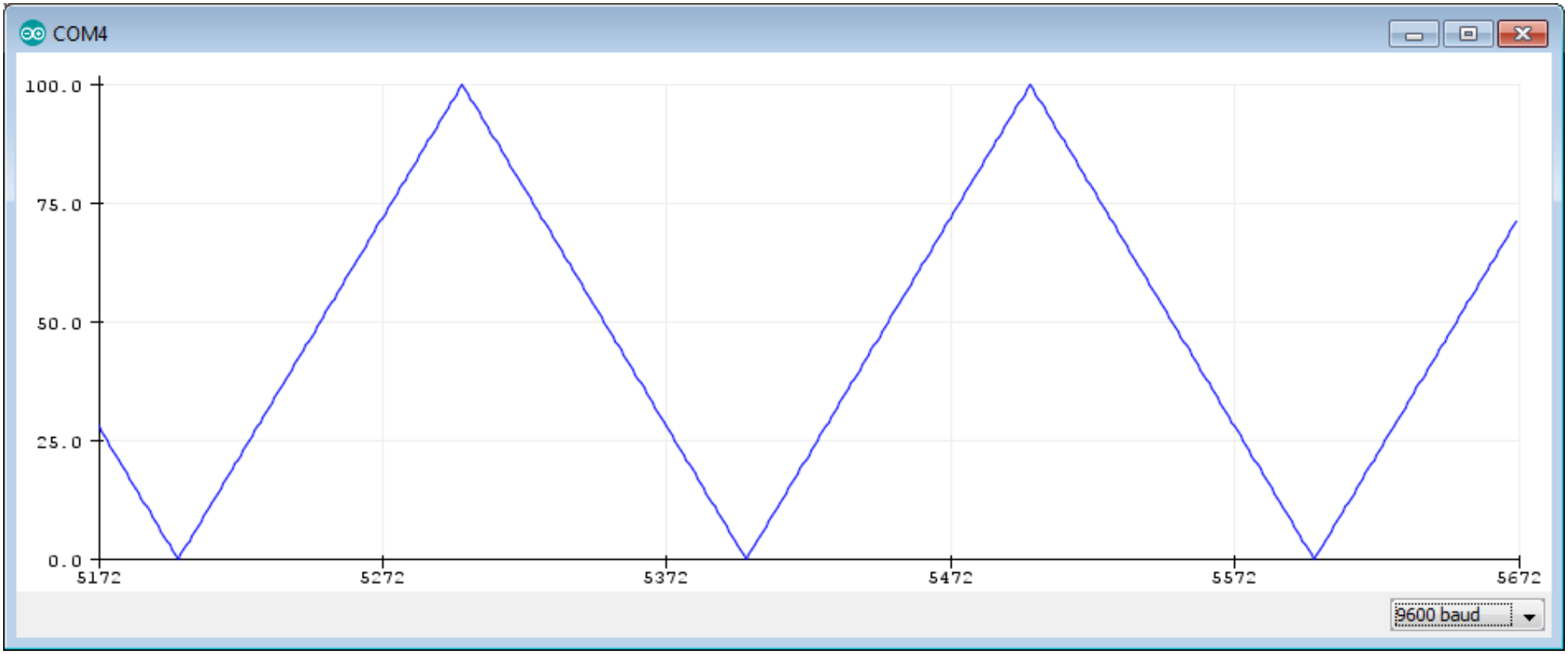
```
void setup() {
  Serial.begin(9600);
}

void loop() {
  for(int i=0; i<1000; i++) {
    int a = a + i;
    Serial.println(a);
  }
}
```



Tömbök és ciklusok: serial_plotter2.ino

```
int a[200];

void setup() {
  Serial.begin(9600);
  for(int i=0; i<100; i++) a[i] = i;
  for(int i=100; i<200; i++) a[i] = 200-i;
}
```

```
void loop() {
  for(int i=0; i<200; i++) {
    Serial.println(a[i]);
  }
}
```



ledblink_special.ino

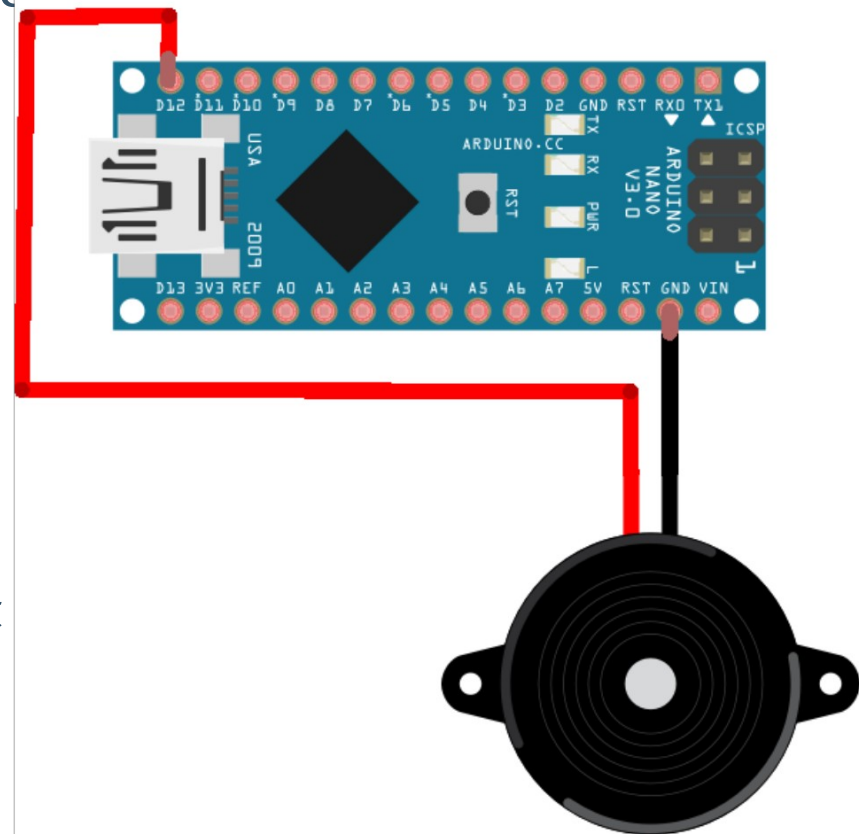
```
#define LED  13                    // A beépített LED-et használjuk
int delayTime = 1000;              // delayTime induló értéke 1000 legyen

void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  while(delayTime > 0) {           // amíg delayTime nagyobb, mint 0
    digitalWrite(LED, HIGH);
    delay(delayTime);
    digitalWrite(LED, LOW);
    delay(delayTime);
    delayTime = delayTime - 100; // fokozatosan csökkentjük az értéket
  }
  while(delayTime < 1000) {        // amíg delayTime kisebb, mint 1000
    delayTime = delayTime + 100; // ezzel kezdjük, delayTime pozitív legyen!
    digitalWrite(LED, HIGH);
    delay(delayTime);
    digitalWrite(LED, LOW);
    delay(delayTime);
  }
}
```

Változó ütemű késleltetés **while** ciklusok használatával
A késleltetési időt az első ciklusban csökkentgetjük, a második ciklusban növelgetjük

Hangkeltés mikrovezérlő segítségével

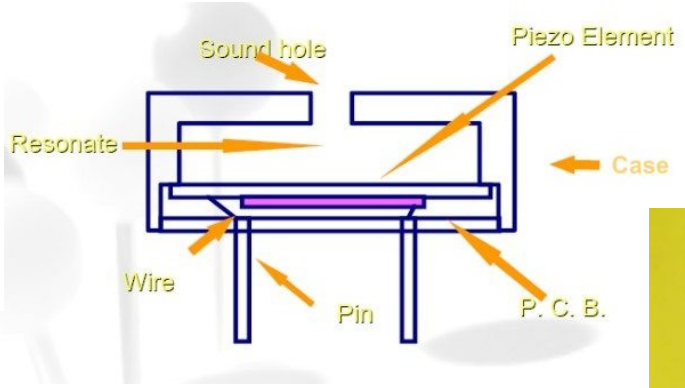
Mottó: „Nem élhetek muzsikaszó nélkül”

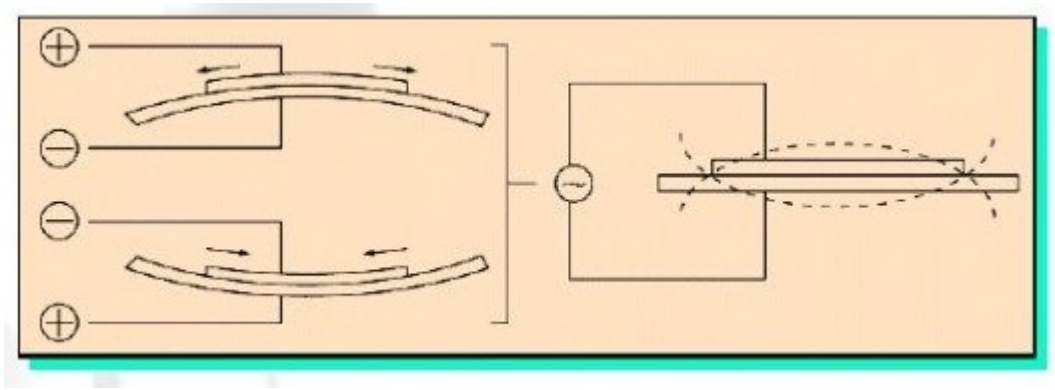
- **Egyszerű hangkeltési módszer:**
  - Négyszöghullámokat keltünk valamelyik kimeneten
  - A kimenetre hangszórót vagy piezo hangjelzőt kötünk (a felvett teljesítmény illesztésére ügyelve!)
- **Előnyök:**
  - Egyszerűség, kis erőforrás-igény
  - Olcsó
- **Hátrányok:**
- Gyenge hangminőség
  - A négyszögjel zavaró felharmonikusokat tartalmaz
  - A jel korlátozott információtartalmú

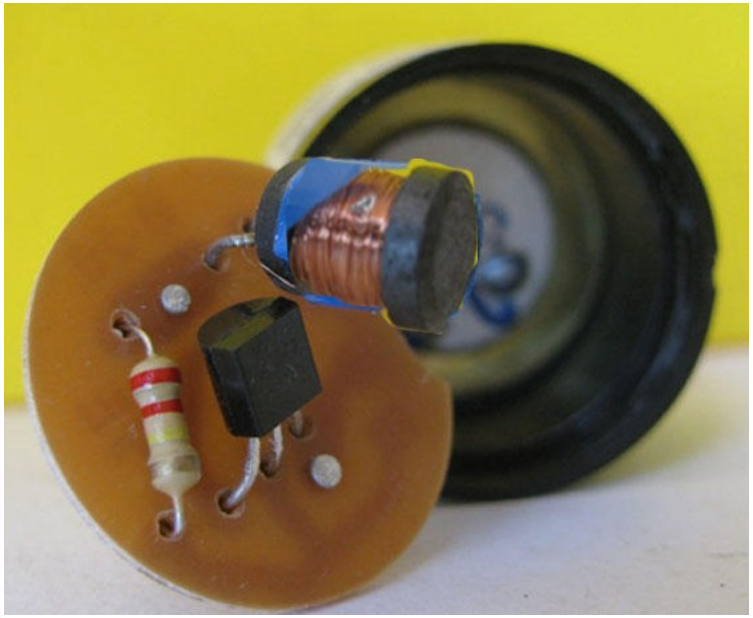


A piezo hangkeltő bemutatása

- Működése a piezoelektromos jelenségen alapul







Linkek: http://www.buzzer-speaker.com/manufacturer/introduction%20of%20piezo%20buzzer.htm
http://www.engineersgarage.com/insight/how-piezo-buzzer-works

A hangkeltéssel kapcsolatos függvények

tone()

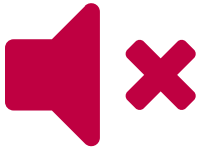
- Négyszögjelet kelt a megadott kimeneten a megadott frekvenciával (50 %-os kitöltéssel). Opcionálisan megadható egy időtartam, különben a hang addig szól, amíg egy **noTone()** függvényhívás le nem állítja a hangot. A hangkeltésre használt kivezetéshez hangszóró vagy piezo buzzer köthető.

Szintaxis:

- **tone**(kivezetés, frekvencia)
- **tone**(kivezetés, frekvencia, időtartam)

noTone()

- Leállítja a hangkeltést a megadott kivezetésen.

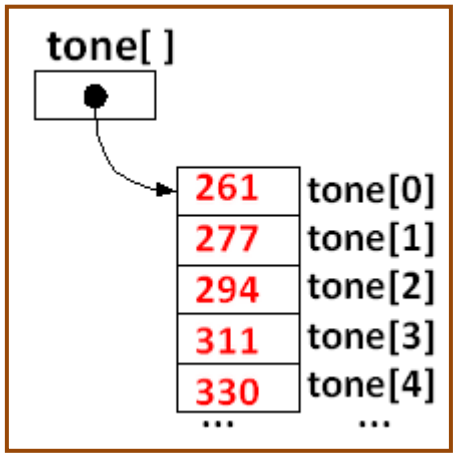


Szintaxis:

- **noTone**(kivezetés)

Skálázzunk! PlayingScale.ino

```
const int buzzerPin = 12;
int numTones = 10;
int tones[] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440 };
//            mid C   C#   D    D#   E    F    F#   G    G#   A
void setup()    {
    for (int i = 0; i < numTones; i++)  {
        tone(buzzerPin, tones[i]);
        delay(500);
    }
    noTone(buzzerPin);
}

void loop()    {
    // Nothing to do here. Press RESET to play again!
}
```

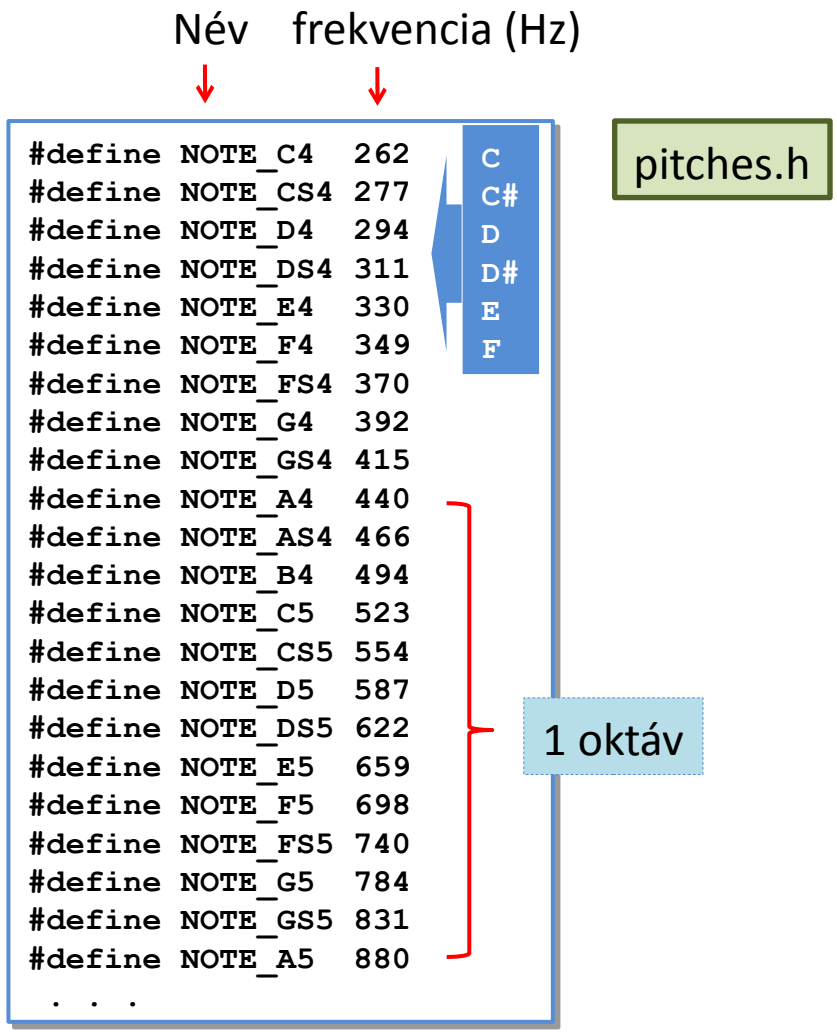


Link: http://learn.adafruit.com/adafruit-arduino-lesson-10-making-sounds/playing-a-scale

- A **tones[]** tömb változó bevezetése a beleírt 10 db szám összefogott kezelését teszi lehetővé. Enélkül nem tudnánk **for** ciklusba szervezni a hangok lejátszását, mindet ki kellene írni.

pitches.h – előre definiált frekvenciák

- A lejátszandó dallam hangjainak magasságát az előre definiált konstansokkal is megadhatjuk.
- A program elején ehhez be kell csatolni a **pitches.h** állományt
- Ha több projektben is használjuk, akkor az **Arduino\libraries** mappában célszerű elhelyezni:
  - Hozzunk létre egy új mappát, például **Notes** néven!
  - Másoljuk bele a **pitches.h** állományt!
  - A projektünkhöz adjuk hozzá a **Notes** programkönyvtárat a **Sketch → Include Library** menüpontban!



toneMelody.ino

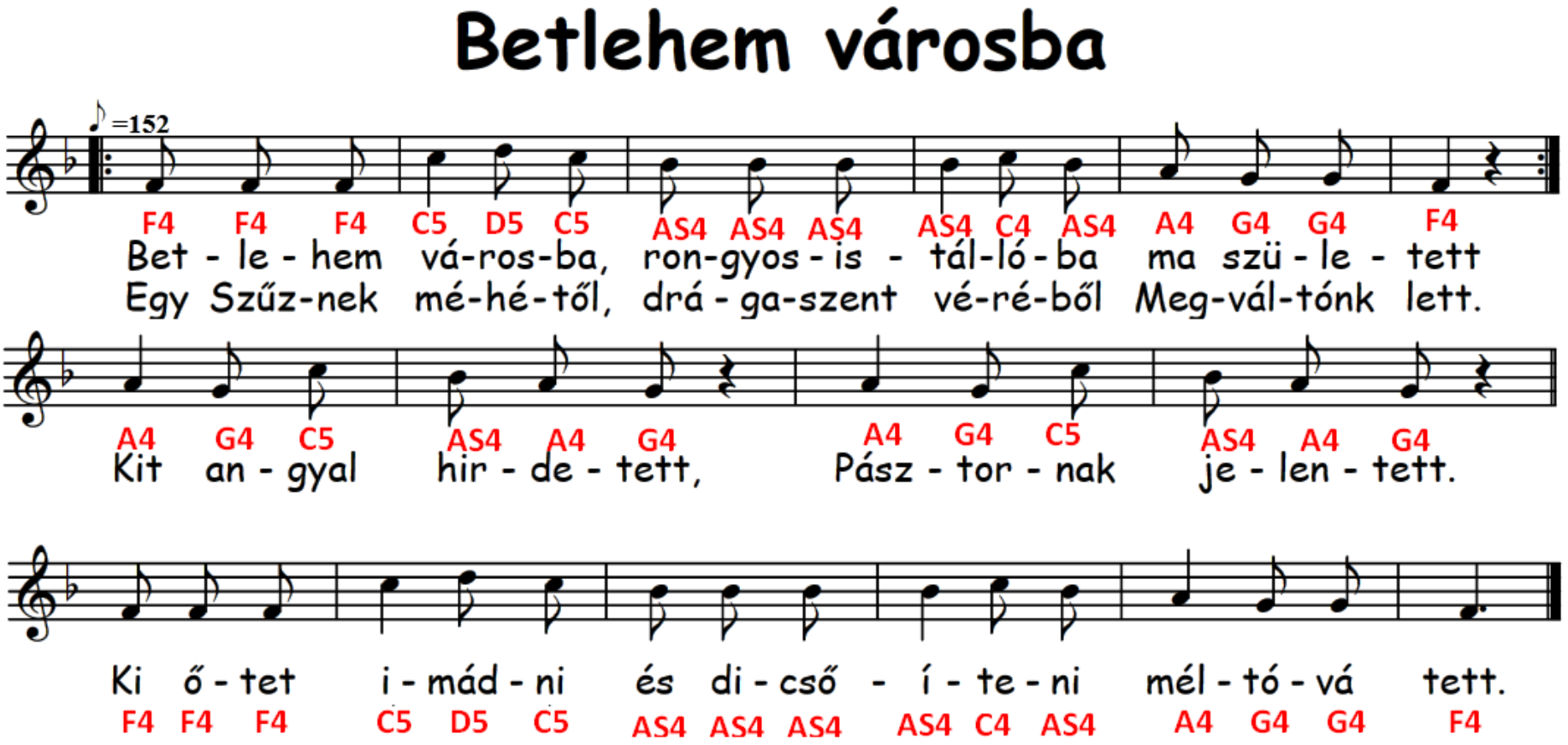
- Ez a gyári mintaprogram egyszerű dallamot játszik le a tömbökben tárolt hangmagasság és időtartam adatokból

```
#include <pitches.h>
//--- a dallam hangjegyei, megszólalás sorrendjében:
int melody[] = { NOTE_C5, NOTE_G4, NOTE_G4, NOTE_A4, NOTE_G4, 0, NOTE_B4, NOTE_C5 };
//--- hangjegyek hossza: 4 = negyed hang, 8 = nyolcad hang stb.:
int noteDurations[] = { 4, 8, 8, 4, 4, 4, 4, 4 };

void setup() {
  for (int thisNote = 0; thisNote < 8; thisNote++) {
    //--- a hangjegyek hosszának kiszámításához osszuk el a másodpercet arányosan,
    //--- pl. a negyed hang = 1000/4, a nyolcad = 1000/8, stb.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(12, melody[thisNote], noteDuration);
    //--- hogy elkülönítsük a hangjegyeket, tegyünk közéjük szünetet (pl. 30 %)
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
  }
  noTone(12);               // megállítjuk a lejátszást:
}

void loop() { }  //--- nem ismételjük meg a dallamot.
```

Magyar változat: Harsányi Réka - Juhász Márton András,
Fizikai számítástechnika: elektronikai alapok és Arduino programozás

Hogyan lesz a kottából zene?

- Nézzük például az alábbi karácsonyi éneket!
  (Vitnyéd – Sopron környéki gyűjtés, Borsai Ilona, Sztanó Pál)
- Írjuk a hangjegyek alá az ABC-s nevüket és az oktáv sorszámát!



Bethlehem_varosba.ino

```
#include <pitches.h>
#define NOTE_P 0        // Szünet definiálása
#define buzzerPin 12    // D12-re kötjük a picergőt
// notes in the melody:
const int melody[] = {
    NOTE_F4,NOTE_F4, NOTE_F4, NOTE_C5,NOTE_D5,NOTE_C5,
    NOTE_AS4,NOTE_AS4,NOTE_AS4,NOTE_AS4,NOTE_C5,NOTE_AS4,
    NOTE_A4,NOTE_G4,NOTE_G4,NOTE_F4,NOTE_P,
    NOTE_F4,NOTE_F4, NOTE_F4, NOTE_C5,NOTE_D5,NOTE_C5,
    NOTE_AS4,NOTE_AS4,NOTE_AS4,NOTE_AS4,NOTE_C5,NOTE_AS4,
    NOTE_A4,NOTE_G4,NOTE_G4,NOTE_F4,NOTE_P,
    NOTE_A4,NOTE_G4,NOTE_C5, NOTE_AS4,NOTE_A4,NOTE_G4, NOTE_P,
    NOTE_A4,NOTE_G4,NOTE_C5, NOTE_AS4,NOTE_A4,NOTE_G4, NOTE_P,
    NOTE_F4,NOTE_F4, NOTE_F4, NOTE_C5,NOTE_D5,NOTE_C5,
    NOTE_AS4,NOTE_AS4,NOTE_AS4,NOTE_AS4,NOTE_C5,NOTE_AS4,
    NOTE_A4, NOTE_G4,NOTE_G4,NOTE_F4,NOTE_P
};

int num = 64; // total number of notes and spaces

// note durations: 4 = quarter note, 8 = eighth note, etc.:
const int noteDurations[] = {
    8,8,8,4,8,8,8,8,8,4,8,8,8,8,8,4,4,
    8,8,8,4,8,8,8,8,8,4,8,8,8,8,8,4,4,
    4,8,8,8,8,8,4,4,8,8,8,8,8,4,
    8,8,8,4,8,8,8,8,8,4,8,8,4,8,8,3
};
```

Betlehem városba,
rongyos istállóba
ma született,

Egy szűznek méhéből,
drága szent véréből,
megváltónk lett.

Kit angyal hirdetett,
pásztornak jelentett,

Ki őtet imádni
és dicsőíteni
méltóvá tett.

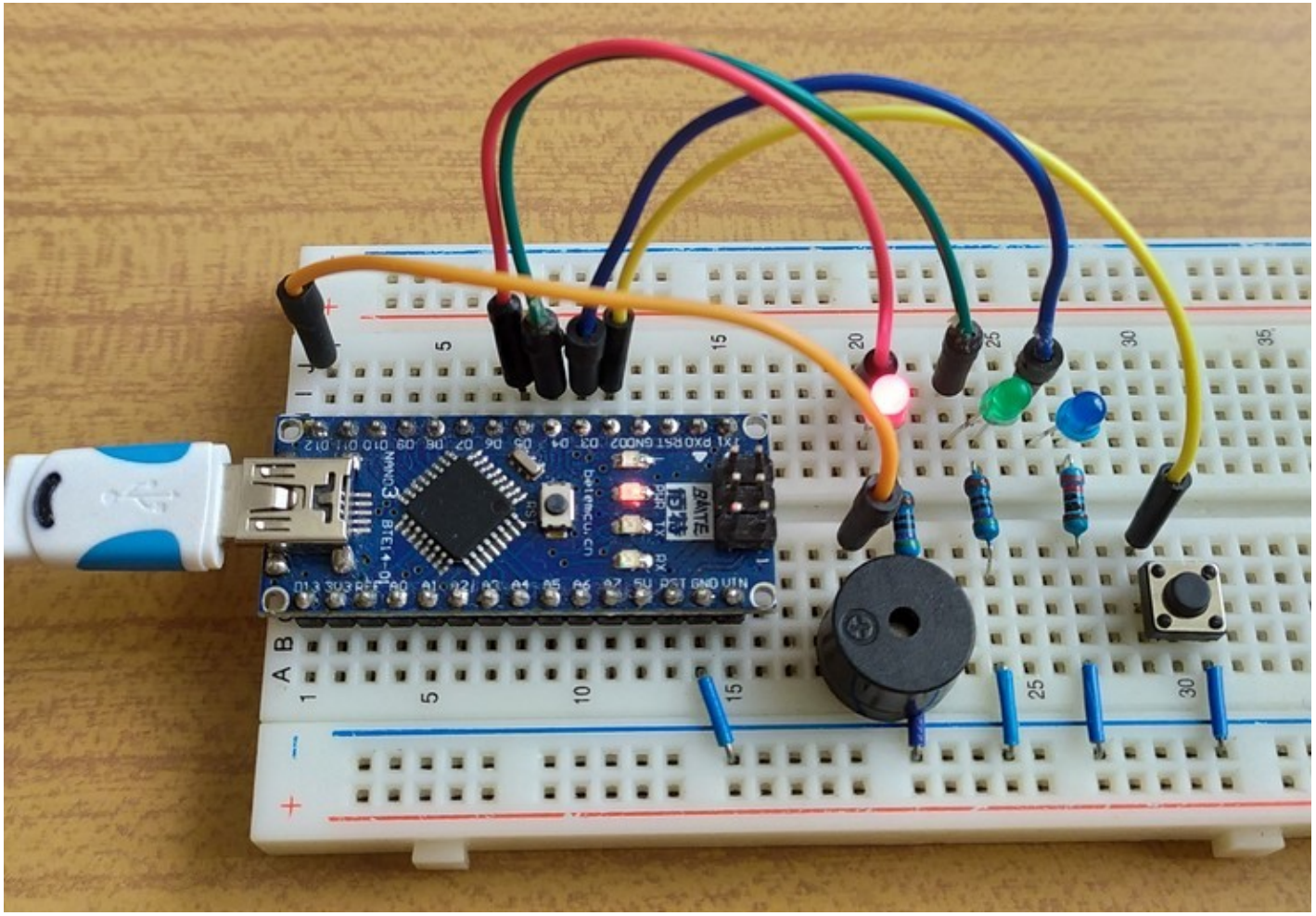
Bethlehem_varosban.ino

```
void setup() {

  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < num; thisNote++) {
    // to calculate the note duration, take one and half second
    // divided by the note type.
    //e.g. quarter note = 1500 / 4, eighth note = 1500/8, etc.
    int noteDuration = 1500/noteDurations[thisNote];
    tone(buzzerPin, melody[thisNote],noteDuration);
    int pauseBetweenNotes = noteDuration + 50;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(buzzerPin);
  }
}

void loop() {

  // no need to repeat the melody.
}
```

Extra szünet a hangok között

LED-es zenelejátszó

- Építsük meg **Jianan Li** LED-ekkel dekorált zenelejátszóját!
- A D12-re kötött piezo csipogó mellé kell 3 db LED (D3, D4, D5) és egy nyomógomb (D2 és GND közé kötve)



LEDMusicPlayer.ino 1. rész

```
#include <pitches.h>
const int piezoPin = 12; //piezo
const int rPin = 5;      //red LED
const int gPin = 4;      //green LED
const int bPin = 3;      //blue LED
const int pPin = 2;      //pushbutton
int ledState = 1;
int ledOn = false;

int melody[] = { NOTE_F5, NOTE_D5, NOTE_AS4, NOTE_D5, NOTE_F5, NOTE_AS5, NOTE_D6,
NOTE_C6, NOTE_AS5,NOTE_D5,NOTE_E5,NOTE_F5, NOTE_F5, NOTE_F5,NOTE_D6, NOTE_C6, NOTE_AS5,
NOTE_A5, NOTE_G5, NOTE_A5, NOTE_AS5, NOTE_AS5, NOTE_F5,NOTE_D5, NOTE_AS4, NOTE_D6,
NOTE_D6, NOTE_D6, NOTE_DS6, NOTE_F6,NOTE_F6,NOTE_DS6,NOTE_D6,NOTE_C6,NOTE_D6,NOTE_DS6,
NOTE_DS6, 0, NOTE_DS6, NOTE_D6, NOTE_C6, NOTE_AS5, NOTE_A5, NOTE_G5, NOTE_A5, NOTE_AS5,
NOTE_D5, NOTE_E5, NOTE_F5, NOTE_F5, NOTE_AS5,NOTE_AS5, NOTE_AS5, NOTE_A5, NOTE_G5,
NOTE_G5, NOTE_G5, NOTE_C6, NOTE_DS6, NOTE_D6, NOTE_C6, NOTE_AS5,NOTE_AS5,NOTE_A5,
NOTE_F5, NOTE_F5, NOTE_AS5, NOTE_C6, NOTE_D6, NOTE_DS6, NOTE_F6, NOTE_AS5, NOTE_C6,
NOTE_D6, NOTE_DS6, NOTE_C6, NOTE_AS5 };

// durations: 2 = half note, and 8/3,4,6,8,12
float noteDurations[] = { 6,12,4,4,4,2,6,12,4,4,4,2,8,8,8/2.9,8,4,2,8,8,4,4,4,4,4,
  6,12,4,4,4,2,8,8,4,4,4,2,8,8,8/2.9,8,4,2,8,8,4,4,4,2,4,4,4,8,8,4,4,4,4,8,8,8,8,4,4,
  8,8,8/2.9,8,8,8,2,8,8,4,4,4,2
};
```

LEDMusicPlayer.ino 2. rész

```
// calculates the number of elements in the melody array.
int musicLength=sizeof(melody)/sizeof('NOTE_F5');

void setup() {
  pinMode(pPin, INPUT_PULLUP);
  pinMode(rPin, OUTPUT);
  pinMode(gPin, OUTPUT);
  pinMode(bPin, OUTPUT);
}

void loop() {
  int pPinState=digitalRead(pPin);
  if(pPinState==LOW && ledState==1)   {//Ha lenyomásra várunk és lenyomást észlelünk
    ledState = 2;
    delay(20);                        //Pergésmentesítés
  }
  if (pPinState==HIGH && ledState==2) {//Ha felengedésre várunk és felengedést észlelünk
    ledState = 1;
    ledOn = not ledOn;                //Átbillentjük az állapotot
    delay(20);                        //Pergésmentesítés
  }
```

LEDMusicPlayer.ino 3. rész

```
if (ledOn && pPinState!=LOW) {
    for (int thisNote = 0; thisNote < musicLength; thisNote++) {
      // blink the three LEDs in sequence
      if (thisNote%3==0){
        digitalWrite(rPin, HIGH);
        digitalWrite(gPin, LOW);
        digitalWrite(bPin, LOW);
      }
      else if (thisNote%3==1){
        digitalWrite(rPin, LOW);
        digitalWrite(gPin, HIGH);
        digitalWrite(bPin, LOW);
      }
      else if (thisNote%3==2){
        digitalWrite(rPin, LOW);
        digitalWrite(gPin, LOW);
        digitalWrite(bPin, HIGH);
      }

      // calculate the note duration. change tempo by changing 2000 to other values
      int noteDuration = 2000/noteDurations[thisNote];
      tone(piezoPin, melody[thisNote],noteDuration);

      // to distinguish the notes, set a minimum time between them.
      // the note's duration + 30% seems to work well
      float pauseBetweenNotes = noteDuration * 1.30;
```

LEDMusicPlayer.ino 4. rész

```
      //split the delay into two parts and check to see
       //whether the pushbutton is pressed to turn off
       //the sound and light
       delay(pauseBetweenNotes/2);
       if(digitalRead(pPin)==LOW) {
         break;
       }
       delay(pauseBetweenNotes/2);
       if(digitalRead(pPin)==LOW) {
         break;
       }
     }
   }
   else if (not ledOn) {
     digitalWrite(rPin, LOW);
     digitalWrite(gPin, LOW);
     digitalWrite(bPin, LOW);
   }
}
```

Zenelejátszás közben itt vizsgáljuk, hogy történt-e gomblenyomás

Gomblenyomás esetén megszakítjuk a for ciklust

Programmegszakítások

- **Arduino Uno,** illetve **Arduino Nano** esetén két külső megszakítási forrás használható:
  INT 0 a **D2** bemeneten
  INT 1 a **D3** bemeneten
- **attachInterrupt**(*interrupt, ISR, mode*) – hozzárendel egy megszakítást kiszolgáló eljárást (ISR) a megnevezett megszakításhoz (0, vagy 1) és beállítja a megszakítás módját (LOW, FALLING, RISING, CHANGE)
- **detachInterrupt**(*interrupt*) – megszünteti a korábbi hozzárendelést
- **Megjegyzés:** az *interrupt* paraméternél a megszakítás sorszámának közvetlen megadása nem javasolt, elavult. Helyette használjuk a **digitalPinToInterrupt**(*pin*) konverziós függvényt, ahol *pin* a felhasznált kivezetés sorszáma (esetünkben 2, vagy 3)

ledswitch_interrupt.ino

- Kapcsolgassuk a beépített LED-et egy hardveresen **pergésmentesített** nyomógombbal, megszakításban!
- Adatátadáskor *volatile* típusú globális változókat használjunk

```
#define ledPin 13                       // A beépített LED-et használjuk
#define interruptPin 2                  // Ide kötjük a nyomógombot
volatile boolean state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, FALLING);
}

void loop() {                           // Most nincs mit tenni ...

}

void blink() {                          // Ez a megszakítást kiszolgáló program
  state = !state;
    digitalWrite(ledPin, state);
}
```

LEDMusicPlayer_interrupt.ino

- Írjuk át a zenelejátszót úgy, hogy a ki-bekapcsolást végző nyomógombot megszakításban kezeljük!
- A nyomógomb lenyomásakor a **ledOn** állapotváltozót billentjük ellenkező állapotba. Ez a változó legyen *volatile* típusú!
- Deklarációs rész:

```
#include <pitches.h>

const int piezoPin = 12; //piezo
const int rPin = 5;      //red LED
const int gPin = 4;      //green LED
const int bPin = 3;      //blue LED
const int pPin = 2;      //pushbutton

volatile boolean ledOn = false;
```

- Setup() kiegészítése:

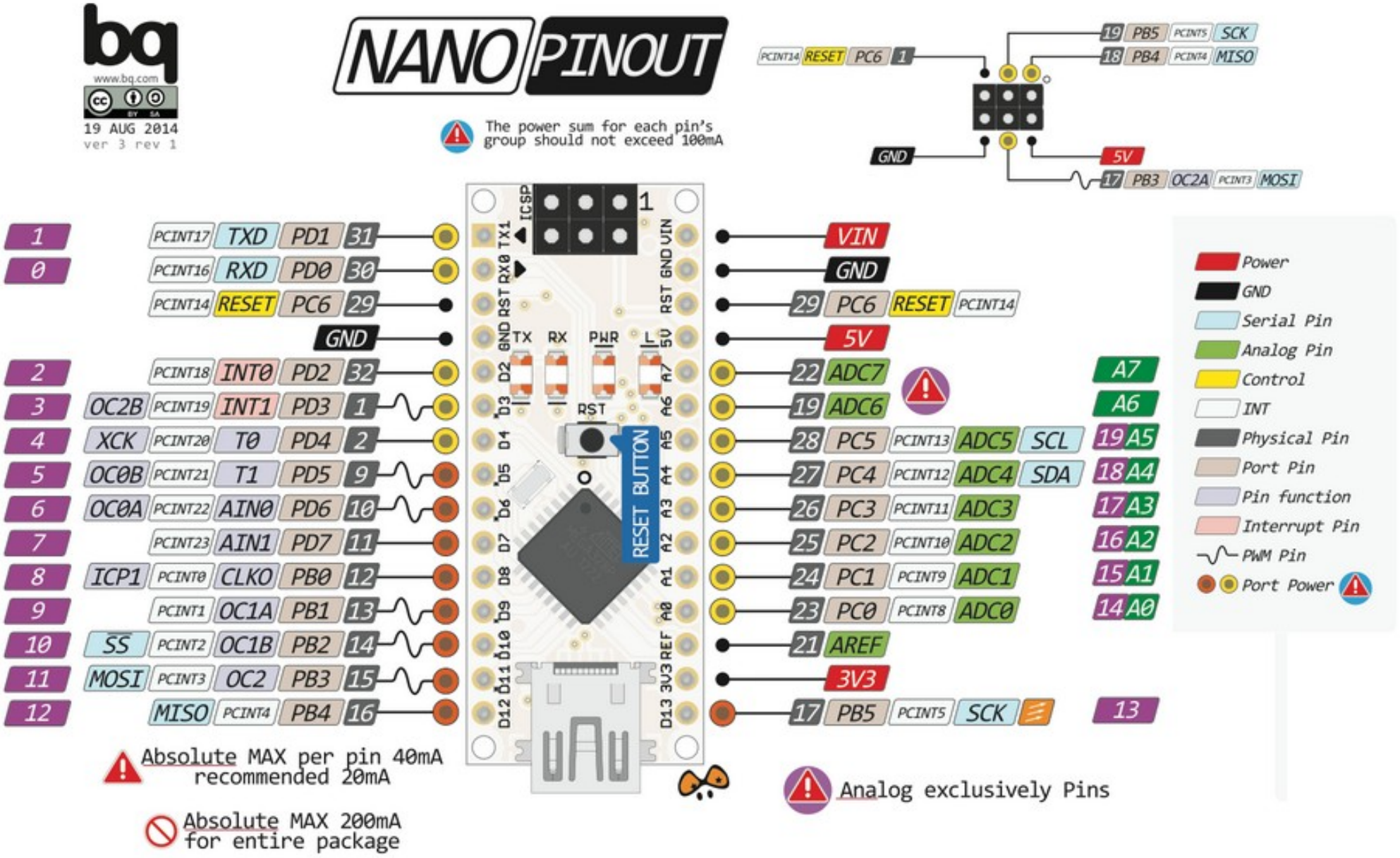
```
  attachInterrupt(digitalPinToInterrupt(pPin), buttonIsr, FALLING);
```

LEDMusicPlayer_interrupt.ino

- Az áttekinthetőség kedvéért a LED-ek kezelését itt most nem mutatjuk (ugyanaz maradt, mint az eredeti programban)

```
void loop() {
  if (ledOn) {
    for (int thisNote = 0; thisNote < musicLength; thisNote++) {
      // A három LED villogtatása
      // itt most nem részletezzük ...
      int noteDuration = 2000/noteDurations[thisNote];
      tone(piezoPin, melody[thisNote],noteDuration);
      float pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);
      if( not ledOn) {
        break;                          // Kilépés a lejátszásból, ha ledOn megváltozott
      }
    }
  }
  else if (not ledOn) {
     // LED-ek lekapcsolása
     // itt most nem részletezzük ...
  }
}

void buttonIsr() {                      // Ez a megszakítást kiszolgáló program
  ledOn = !ledOn;                       // Pergésmentesített nyomógombot feltételezünk
}
```

Az Arduino nano kártya kivezetései



Ellenállás színkódok