SZÉCHENYI ISTVÁN EGYETEM

Automatizálási Tanszék

Mintavételes szabályozás mikrovezérlő segítségével

Budai Tamás
budai.tamas@sze.hu
http://maxwell.sze.hu/~budait

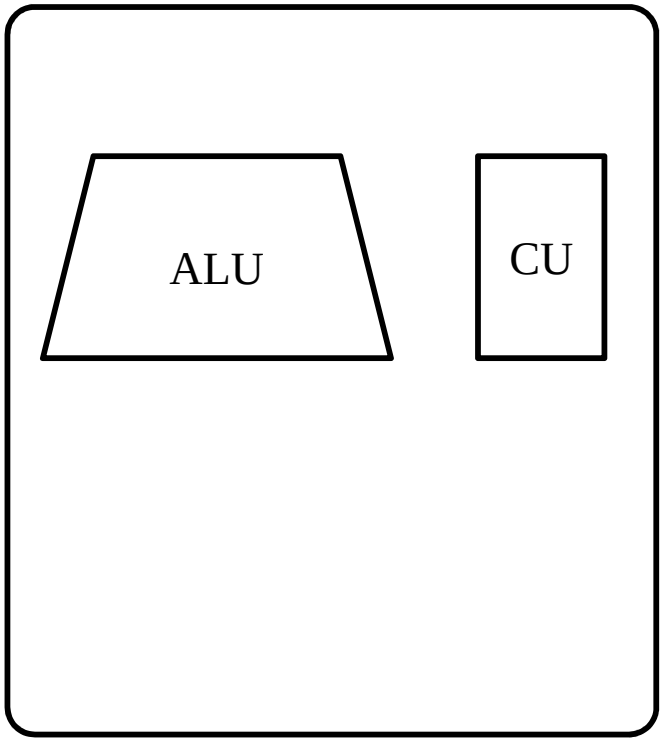
2015.03.24.

Tartalom

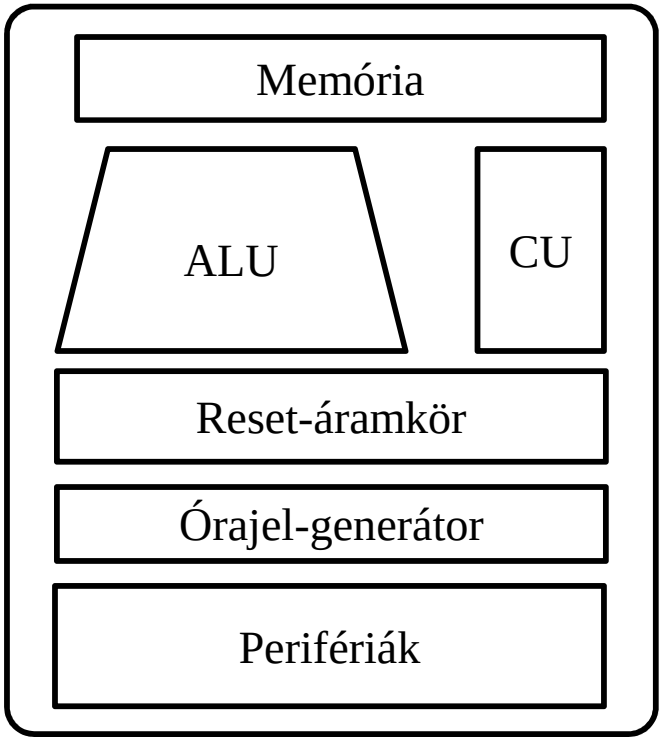
- Mikrovezérlőkről röviden
- Programozási alapismeretek – ismétlés
- Az Arduino család
- Mikrovezérlők programozása

Mikrovezérlőkről röviden

Mikroprocesszor



Mikrovezérlő



A mikrovezérlő (=mikrokontroller) : mikroszámítógép egy tokban

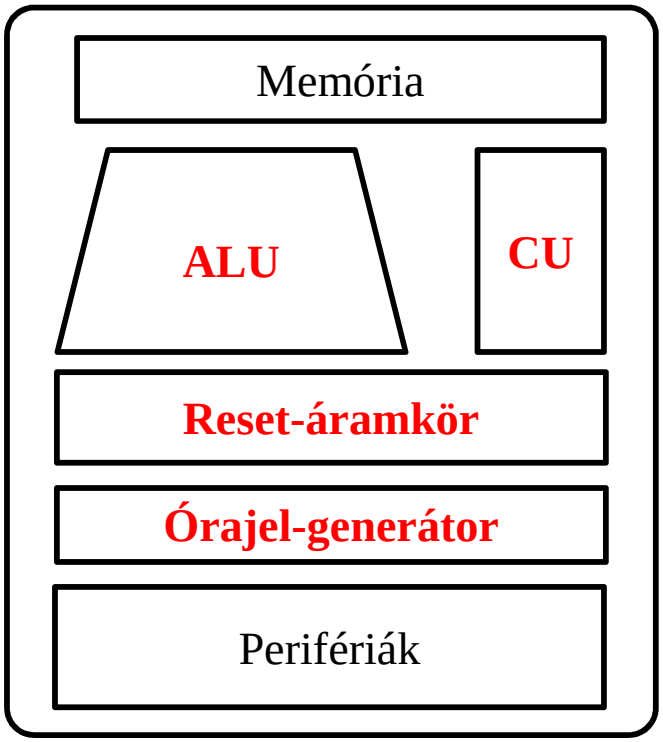
Mikrovezérlőkről röviden

CPU és tartozékai:

- Egyes családoknál* közös
- Meghatározza a számábrázolást:

  8,16,32bit…

Mikrovezérlő

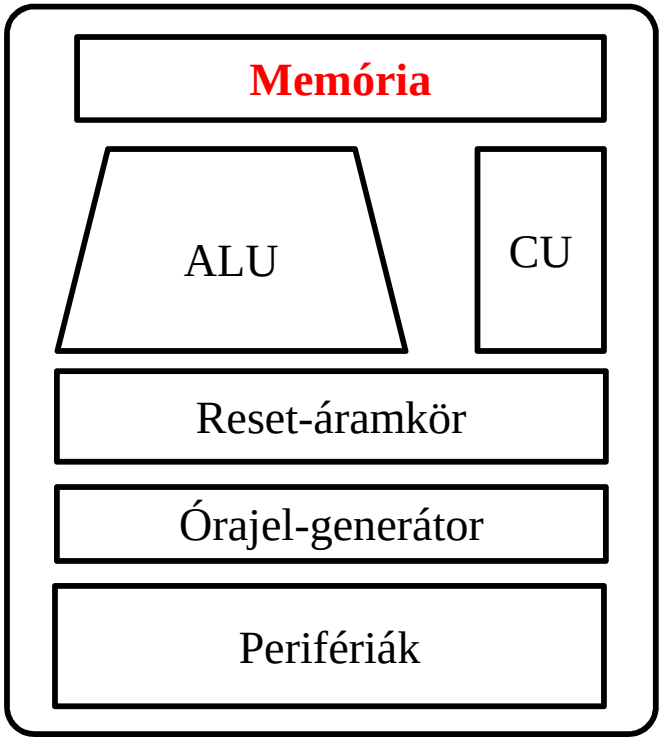


Mikrovezérlőkről röviden

Memória típusok:

- Program memória
  - Perzisztens, nem újraírható
  - Perzisztens, de újraírható
- Adatmemória
  - Nem perzisztens: RAM
  - Perzisztens: ROM

Mikrovezérlő

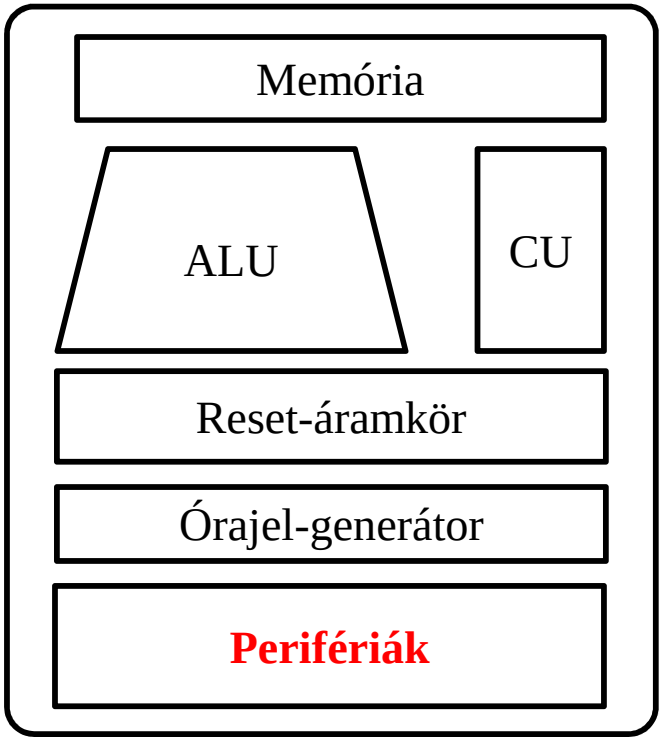


Mikrovezérlőkről röviden

Perifériák:

- Digitális be/kimenetek
- Digitális kapcsolatok: SPI, USB, CANBUS…
- A/D, D/A átalakítók
- PWM kimenet

Mikrovezérlő



Mikrovezérlők programozásához használt nyelvek:

- Assembly
- C/C++

Mikrovezérlők programozásához használt nyelvek:

- Assembly
- **C/C++**

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- Ciklusok
- Függvények

A **forráskód**:

- Egyszerű szövegfájl
- Ember által értelmezhető
- Utasítások sorozata
- **Kötött formátum**

Példa:

```c
#include <stdio.h>

#define ONE 1

int main()
{
  int j=5;
  int tanult = 1;

  if(tanult){
    printf("Az on erdemjegye: %d\n", j);
  }
  else{
    printf("Az on erdemjegye: %d\n", ONE);
  }

  return 0;
}

```

#include direktíva:

- más forráskódrészeket tudunk beágyazni
- ezek erősen függenek a használt környezettől

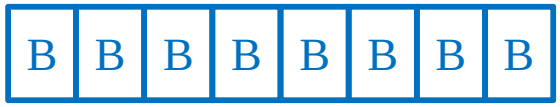
main() függvény:

- minden programban pontosan egy darabnak kell lennie
- itt kezdődik a programunk végrehajtása

C nyelv – alapismeretek:

- Programszerkezet
- **Változók, adattípusok**
- Operátorok
- Elágazások, Feltételes utasítások
- Ciklusok
- Függvények

A legkisebb egység: 1 bit 

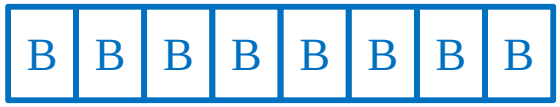
8 bit = **1 byte** 

void	„semmi” (ezt is jelölni kell valahogy)
bool	logikai (igaz/hamis)
char	egy karakter (1 byte)
int	(előjeles) egész szám
unsigned int	egész szám
float	lebegőpontos szám
double	dupla pontosságú float
string	char-okból álló tömb

C nyelv – alapismeretek:

- Programszerkezet
- **Változók, adattípusok**
- Operátorok
- Elágazások, Feltételes utasítások
- Ciklusok
- Függvények

A legkisebb egység: 1 bit 

8 bit = **1 byte** 

void	„semmi” (ezt is jelölni kell valahogy)
bool	logikai (igaz/hamis)
char	egy karakter (1 byte)
int	**(előjeles) egész szám**
unsigned int	egész szám
float	**lebegőpontos szám**
double	dupla pontosságú float
string	char-okból álló tömb

C nyelv – alapismeretek:

- Programszerkezet
- **Változók, adattípusok**
- Operátorok
- Elágazások, Feltételes utasítások
- Ciklusok
- Függvények

Példa:

```
1   int a;
2   float b;
3   double c;
4
5   a=5;
6   b=3.14;
7   c = 6.53;
8
9   int n = 0;
10  float x = 1.2345;
```

Tömbök:

- Egy adott adattípusból álló több elemű halmaz
- A memóriában folytonosan helyezkedik el
- A tömb méretét előre meg kell mondanunk*

Tömbök:

- Egy adott adattípusból álló több elemű halmaz
- A memóriában folytonosan helyezkedik el
- A tömb méretét előre meg kell mondanunk*

```
1  int ints[6];
2  int pins[] = {2, 4, 8, 3, 6};
3  int vals[6] = {2, 4, -8, 3, 2};
4
5  printf("%d", pins[0]);
6  printf("%d", vals[3]);
```

Operátorok:

- Aritmetikai: `= + - * / % ( )`
- Összehasonlító: `== != < <= > >=`
- Logikai: `&& || !`
- Bitenkénti: `& | ^ ~ << >>`
- Helyben módosító: `++ -- += -= *= /= &= |=`
- Egyéb: `sizeof`

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- **Operátorok**
- Elágazások, Feltételes utasítások
- Ciklusok
- Függvények

Példa:

```
1  int a,b,c;
2  int d;
3  int e;
4
5  a=5;
6  b=2;
7  c=3;
8
9  d = a+b*c;
10
11 d= (a-b)*c;
12
```

Elágazások:

- (GOTO)
- if, if-else
- switch case

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- **Elágazások, feltételes utasítások**
- Ciklusok
- Függvények

if, if-else:

```
1  int x=0;
2  int val = 700;
3
4  if (val < 500)
5  {
6    x = 1;
7  }
8  else
9  {
10   x = 0;
11 }
```

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- **Elágazások, feltételes utasítások**
- Ciklusok
- Függvények

switch case:

```
1  int kerdes;
2  kerdes = 2;
3
4    switch (kerdes) {
5      case 1:
6        printf("Elso");
7        break;
8      case 2:
9        printf("Masodik");
10       break;
11     default:
12       printf("Egyik sem");
13   }
```

Ciklusok:

- for
- while
- do-while

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- **Ciklusok**
- Függvények

for:

```
1  int i;
2  i = 0;
3
4  for (; i <= 255; i++){
5    printf("%d ", i);
6  }
```

tömörebben:

```
1  for (int i=0; i <= 255; i++){
2    printf("%d ", i);
3  }
```

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- **Ciklusok**
- Függvények

while :

```
1  int var = 0;
2  while(var < 200){
3    printf("%d", var);
4    var++;
5  }
```

```
1  int var = 400;
2  while(var > 200){
3    printf("%d", var);
4    var--;
5  }
```

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- **Ciklusok**
- Függvények

do-while:

```
1  int x = 0;
2  do
3  {
4    printf("Ertek: %d",x);
5    x++;
6
7  } while (x < 100);
```

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- **Ciklusok**
- Függvények

Ciklusvezérlés:

- break
- continue
- return

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- **Ciklusok**
- Függvények

break:

```
1  int var = 0;
2  while(var < 200){
3
4    if(var>10){
5        printf("elertuk a limitet!");
6        break;
7    }
8
9    var++;
10 }
```

break:

```c
int var = 0;
while(var < 200){

  if(var>10){
      printf("elertuk a limitet!");
      break;
  }

  var++;
}
```

```c
int kerdes;
kerdes = 2;

  switch (kerdes) {
    case 1:
      printf("Elso");
      break;
    case 2:
      printf("Masodik");
      break;
    default:
      printf("Egyik sem");
  }
```

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- **Ciklusok**
- Függvények

continue:

```
1  int var = 0;
2  while(var < 200){
3    var++;
4
5    if((var%2) == 0){
6        continue;
7    }
8
9    printf("%d ",var);
10
11 }
```

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- **Ciklusok**
- Függvények

return:

```
1  if(err != 0){
2      printf("HIBA!");
3      return 1;
4  }
```

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- **Ciklusok**
- Függvények

return:

```c
if(err != 0){
    printf("HIBA!");
    return 1;
}
```

```c
#include <stdio.h>

#define ONE 1

int main()
{
  int j=5;
  int tanult = 1;

  if(tanult){
    printf("Az on erdemjegye: %d\n", j);
  }
  else{
    printf("Az on erdemjegye: %d\n", ONE);
  }

  return 0;
}

```

Függvények:

- mint a matematikában:
  - paraméter, visszatérési érték
- programszervezésre használhatók

Példa:

y = sin(x)

C nyelv – alapismeretek:

- Programszerkezet
- Változók, adattípusok
- Operátorok
- Elágazások, Feltételes utasítások
- Ciklusok
- **Függvények**

Függvények:

```
1  #include <stdio.h>
2  #include <math.h>
3
4  float sinfok(int f){
5      return sin(f*M_PI/180);
6  }
7
8
9  int main(){
10     int x = 30;
11     printf("%f\n", sinfok(x));
12 }
```

Összefoglalás: A programozó „fegyvertára”

```
= + - * /  % ( ) == != < <= > >= ++ -- += -=
*= /= &= |= sizeof bool int float [] {} ;
#include
```

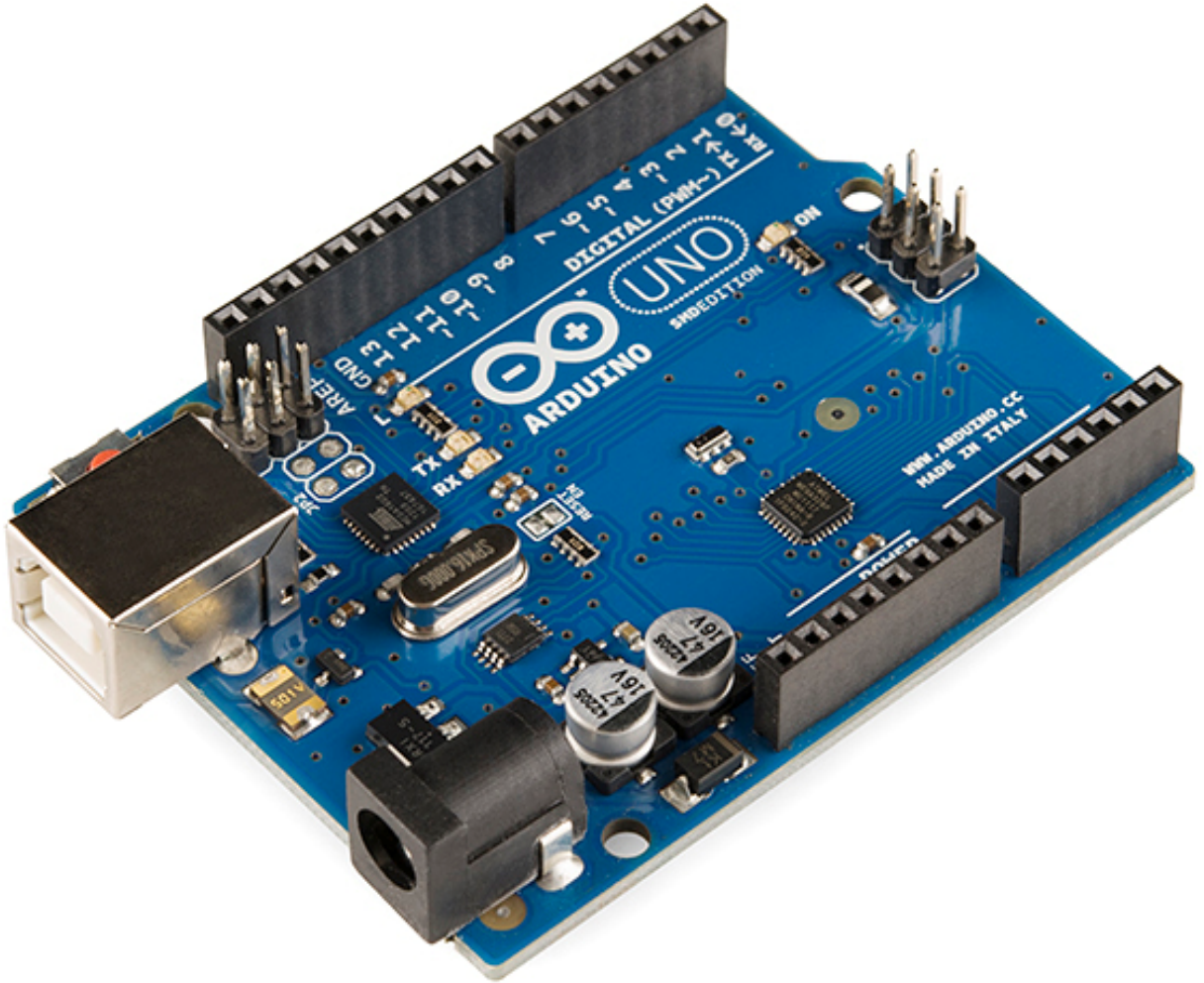
Mit hagytunk ki:

- előfordító direktívák, makrók
- mutatók, mutató aritmetika
- dinamikus memóriakezelés
- saját adattípusok, struktúrák
- függvénykönyvtárak
- hibakeresés
- …

Ajánlott olvasmányok:

- http://www.tutorialspoint.com/cprogramming/index.htm
- B. W. Kernighan - D. M. Ritchie : A C programozási nyelv
- http://arduino.cc/en/Reference/HomePage
- http://arduino.cc/en/Reference/Libraries

Az ARDUINO család



Az ARDUINO család

Arduino Uno

Arduino Leonardo

Arduino Due

Arduino Yún

Tartalom:

- Az Arduino IDE
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

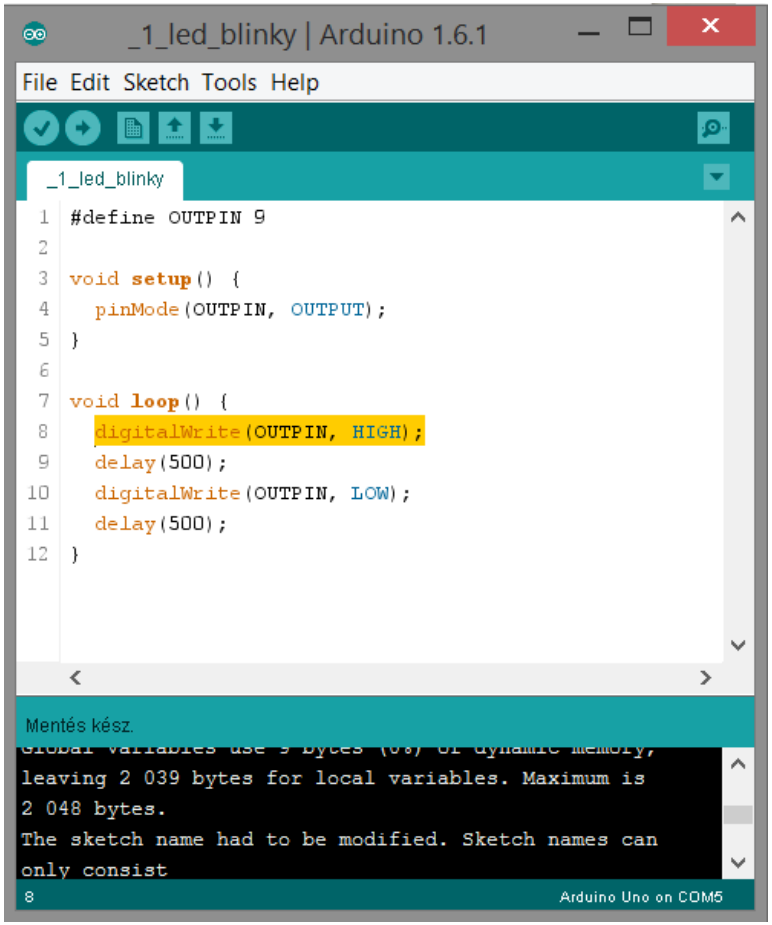
Az Arduino IDE

Ingyenes letöltés:

www.arduino.cc

- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

Az Arduino IDE



- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

Az Arduino IDE

Menü és ikonsor

- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

Az Arduino IDE

Menü és ikonsor

Forrásszerkesztő

- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

Az Arduino IDE

- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

Az Arduino IDE

- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

Az Arduino IDE

Verifikáció
(fordítás)

- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

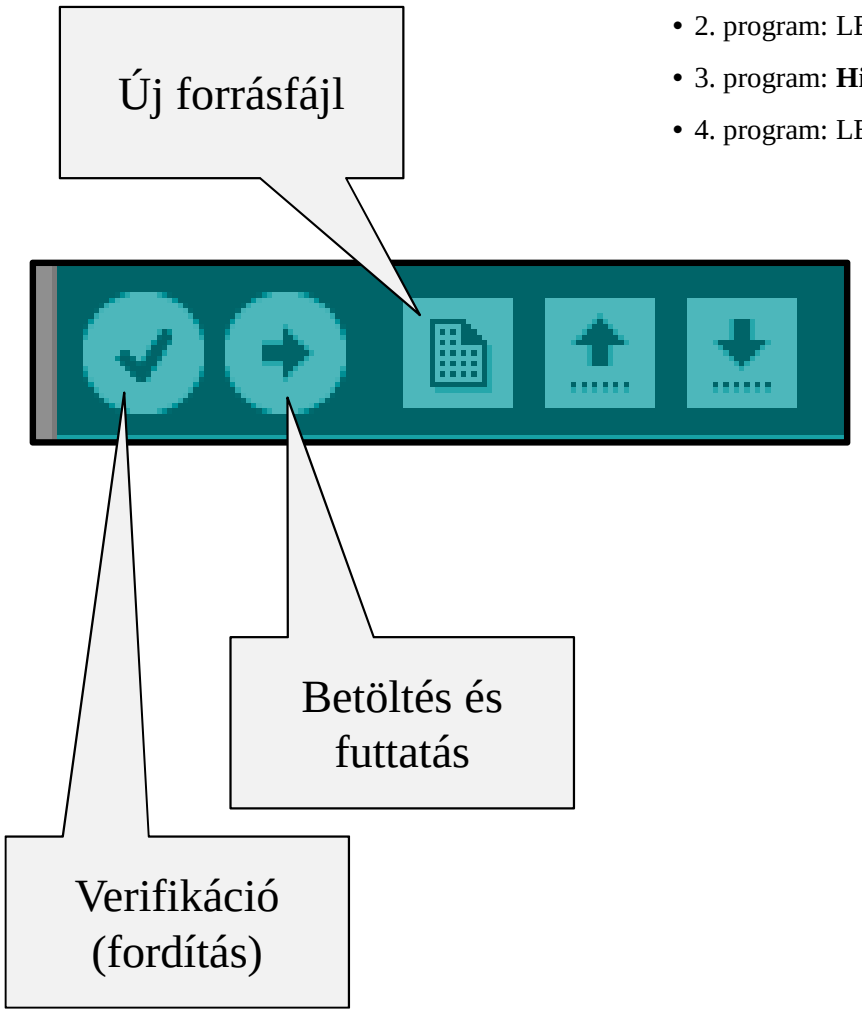
Az Arduino IDE

Betöltés és futtatás

Verifikáció (fordítás)

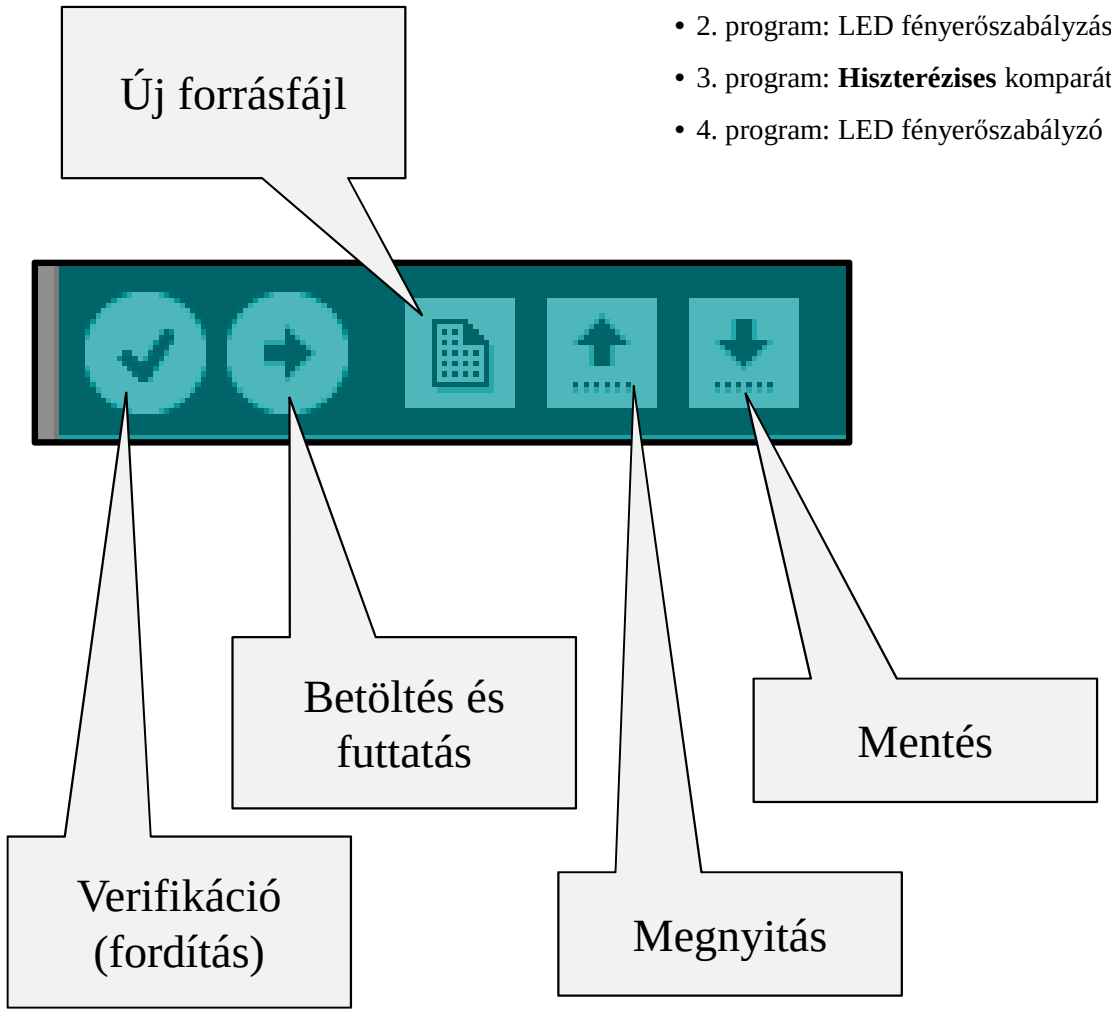
- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

Az Arduino IDE



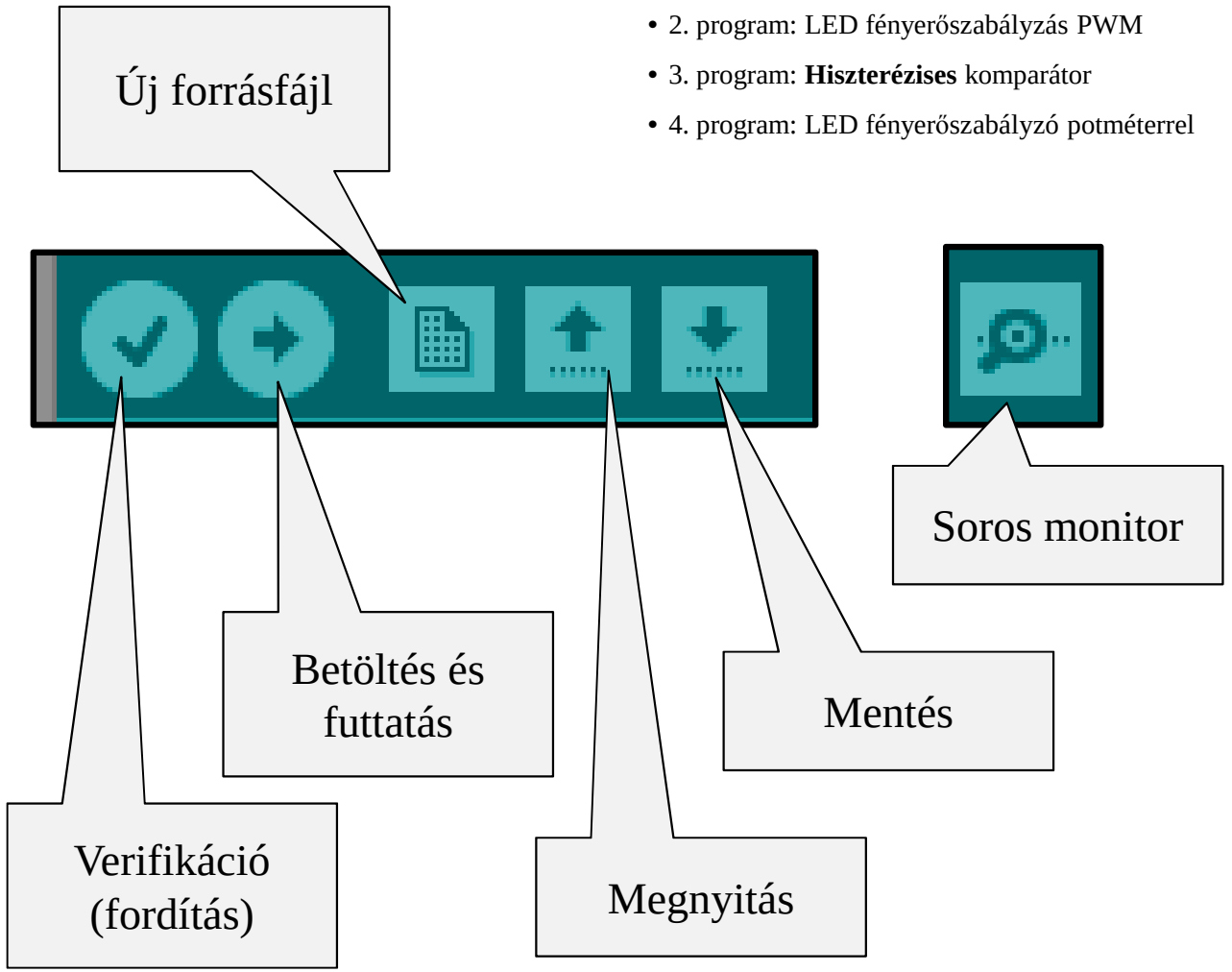
Az Arduino IDE

- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel



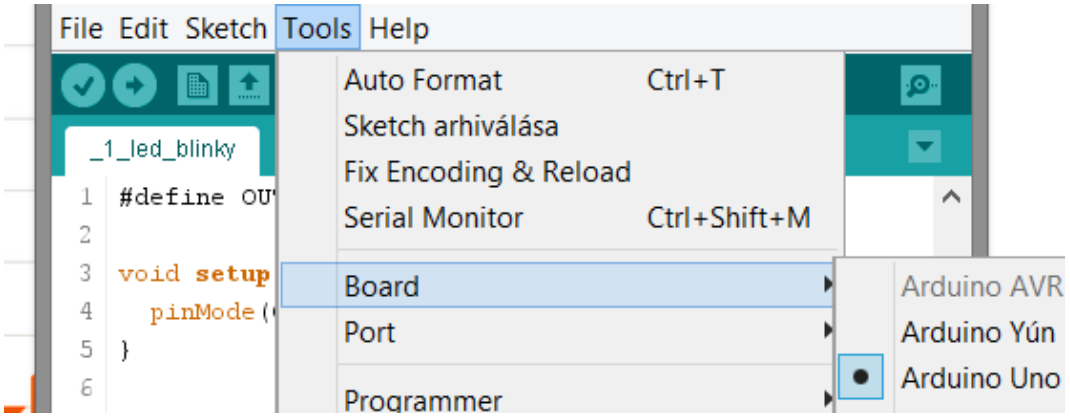
Az Arduino IDE

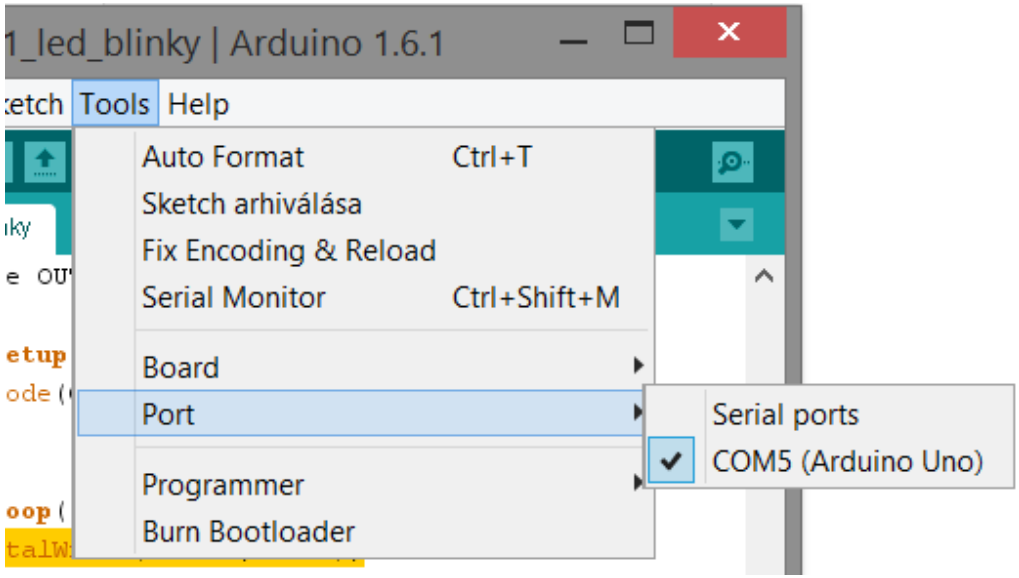
- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel



- **Az Arduino IDE**
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

Eszköz kiválasztása:





Összefoglalás:

- A fájlok nevébe ne tegyünk ékezetet!
- Mindig figyeljünk, hogy jó eszköz és port legyen kiválasztva
- Fordításkor figyeljük a naplót (hibák!)

Különbségek:

- Kisebb tárolási egységek (számábrázolás!)
- Lassabb órajel → minden utasítás számít
- nincsenek klasszikus be és kimeneti perifériák (billentyűzet, képernyő)
- a mikrovezérlőn általában nem egyszer futtatjuk a programot, hanem folyamatosan (végtelen ciklus)

Különbségek:

- Függvénytár

`pinMode(láb, mód)`	beállítja a <láb>-at a megfelelő <mód>-ba. (INPUT,OUTPUT)	`pinMode(8, OUTPUT)`
`digitalWrite(láb, érték)`	beállítja a <láb>-on a kimenet <érték>-ét (LOW/HIGH)	`digitalWrite(8, LOW)`
`delay(idő)`	várakozik <idő>-nyi milliszekundumig	`delay(500)`
`analogRead(láb)`	beolvassa a <láb>-ról a feszültség értékét* (A/D)	`analogRead(A1);`
`analogWrite(láb, érték)`	beállítja a <láb>-on az analóg értéket**	`analogWrite(8,214);`

`analogRead(láb)`:

- 10 bites A/D átalakító

egész számot ad vissza, ami arányos a bemeneti feszültséggel:

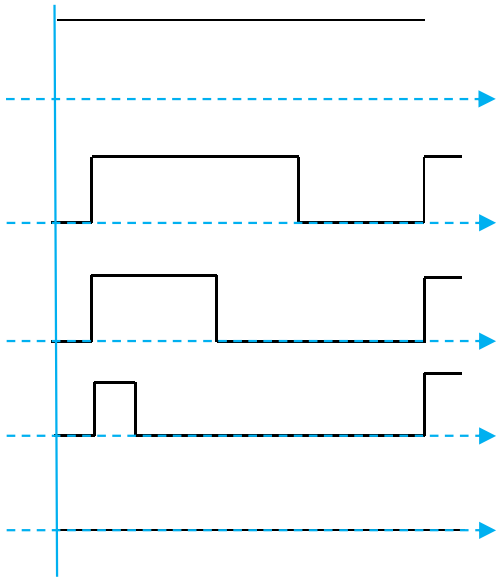
Feszültségszint:	Beolvasott érték
5V	1023
…..	…..
…	….
2.5V	511
..	..
.	.
0V	0

`analogWrite(láb, érték)`:

- 8 bites PWM modul*

egész számot vár 0-255 között, ezzel arányosan állítja be a négyszögjel kitöltési tényezőjét:

bementi paraméter:	**kitöltési tényező:**
255	100%
…..	…..
…	….
128	50%
..	..
.	.
0	0%



Blinky, a villogó LED

- Az Arduino IDE
- különbségek az „asztali” programokhoz képest
- **1. program: blinky, a villogó LED**
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

```
1  #define OUTPIN 9
2
3  void setup() {
4    pinMode(OUTPIN, OUTPUT);
5  }
6
7  void loop() {
8    digitalWrite(OUTPIN, HIGH);
9    delay(500);
10   digitalWrite(OUTPIN, LOW);
11   delay(500);
12 }
```

LED fényerőszabályzás PWM

- Az Arduino IDE
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- **2. program: LED fényerőszabályzás PWM**
- 3. program: **Hiszterézises** komparátor
- 4. program: LED fényerőszabályzó potméterrel

```
1  #define OUTPIN 9
2
3  void setup() {
4    pinMode(OUTPIN, OUTPUT);
5  }
6
7  int i=0;
8  void loop() {
9    if(i > 255){
10     i=0;
11   }
12   i++;
13   // PWM
14   analogWrite(OUTPIN,i);
15   delay(20);
16 }
```

Hiszterézises komparátor

```
1  #define INPIN A0
2  #define OUTPIN 9
3
4  void setup() {
5    pinMode(OUTPIN, OUTPUT);
6    pinMode(INPIN, INPUT);
7    digitalWrite(OUTPIN, LOW);
8    Serial.begin(9600);
9  }
10
11 int ival;
12 float ivoltage;
13
14 void loop() {
15   ival = analogRead(INPIN);
16   // bemenet: 10bit A/D. Alakitsuk at feszultsegszintre!
17   ivoltage = (ival/1023.0) * 5.0;
18   if(ivoltage > 3.8){
19     digitalWrite(OUTPIN, HIGH);
20   }
21   if(ivoltage < 1.2){
22     digitalWrite(OUTPIN, LOW);
23   }
24   Serial.println(ivoltage);
25 }
```

- Az Arduino IDE
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- **3. program: Hiszterézises komparátor**
- 4. program: LED fényerőszabályzó potméterrel

LED fényerőszabályzó potméterrel

```
1  #define INPIN A0
2  #define OUTPIN 9
3
4  void setup() {
5    pinMode(OUTPIN, OUTPUT);
6    pinMode(INPIN, INPUT);
7    Serial.begin(9600);
8  }
9
10 int ival;
11 float ivoltage;
12 int outval;
13 float outvoltage;
14 void loop() {
15
16   ival = analogRead(INPIN);
17   // bemenet: 10bit A/D. Alakitsuk at feszultsegszintre!
18   ivoltage = (ival/1023.0) * 5.0;
19   outvoltage = ivoltage;
20   // alakitsuk at a szintet kitoltesi tenyezore a PWM-hez!
21   outval = (outvoltage/5)*255;
22   analogWrite(OUTPIN,outval);
23   Serial.println(ivoltage);
24   //delay(1);
25 }
```

- Az Arduino IDE
- különbségek az „asztali” programokhoz képest
- 1. program: blinky, a villogó LED
- 2. program: LED fényerőszabályzás PWM
- 3. program: **Hiszterézises** komparátor
- **4. program: LED fényerőszabályzó potméterrel**