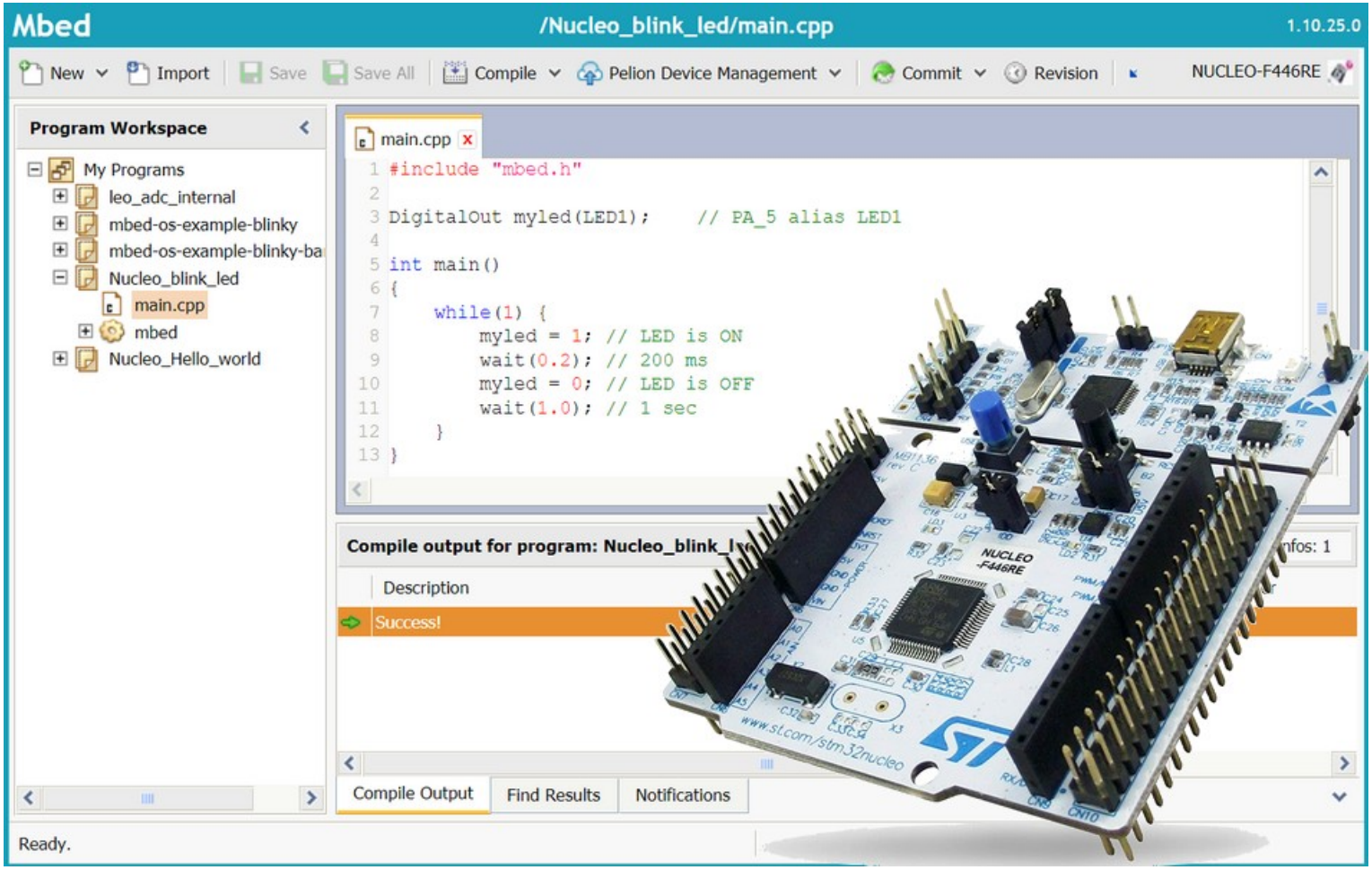
STM32 mikrovezérlők programozása ARM mbed környezetben



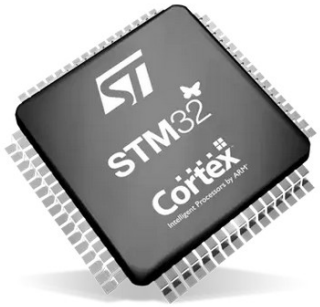
3. Analóg I/O

Felhasznált és ajánlott irodalom

- Cserny István: A FRDM-KL25Z kártya programozása mbed környezetben
- Rob Toulson and Tim Wilmhurst:
  Fast and Effective Embedded Systems Design: Applying the ARM mbed
- Perry Xiao: Designing Embedded Systems and the Internet of Things (IoT) with the ARM mbed
- Dogan Ibrahim: ARM-based Microcontroller Projects Using mbed
- **ARM mbed honlap: https://os.mbed.com/**
  - ARM mbed Compiler: https://ide.mbed.com/compiler/
  - ARM mbed 2 Handbook (elavult): https://os.mbed.com/handbook/Homepage
  - ARM mbed 2 Cookbook (elavult): https://os.mbed.com/cookbook/Homepage
  - ARM mbed forráskód: https://github.com/ARMmbed/mbed-os

Adatlapok:

- **STM32F446RE** adatlap és termékinfo
- **STM32F446** Family Reference Manual

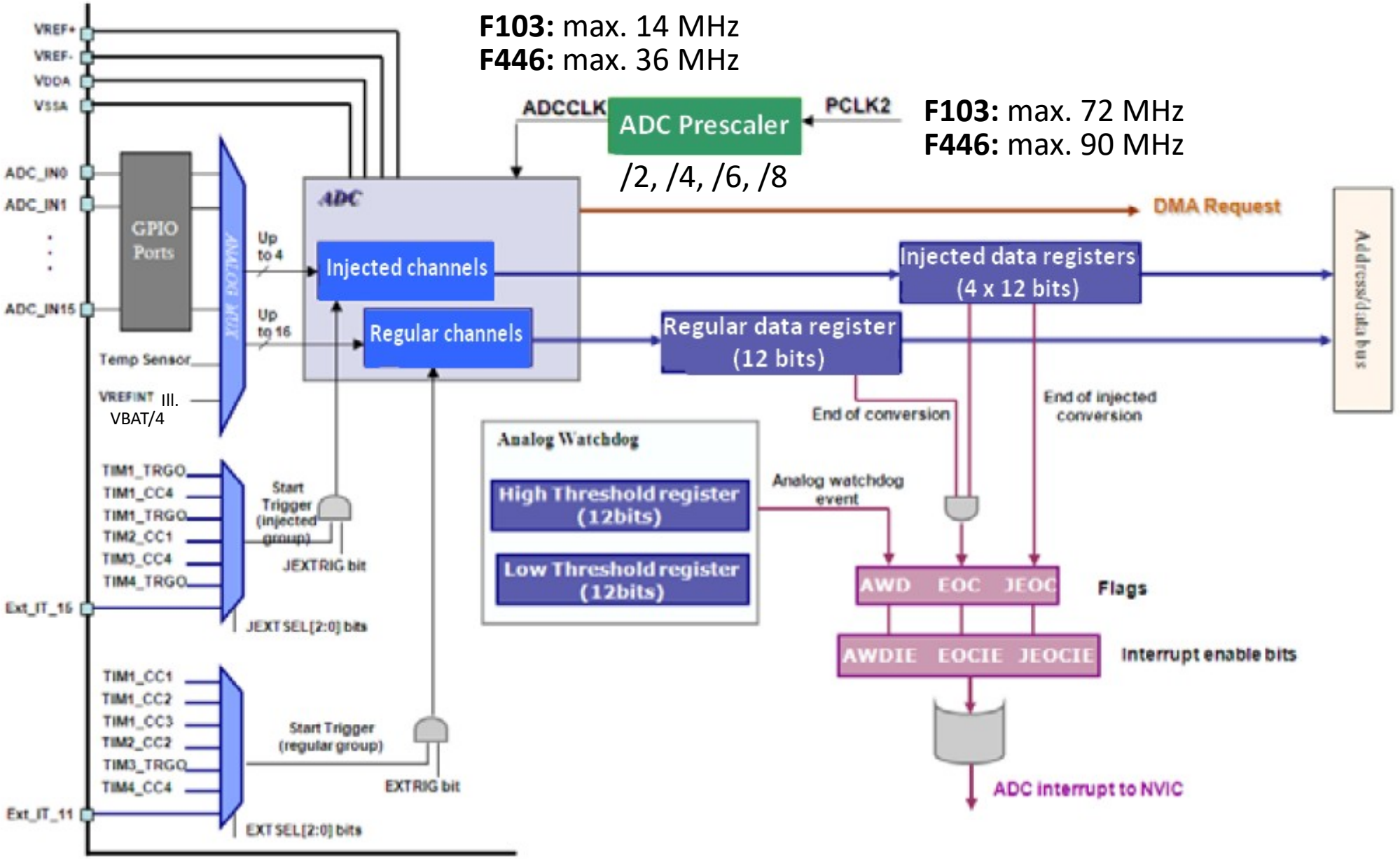


Az STM32 ADC-k főbb jellemzői

	STM32F446RE	STM32F103C8
ADC	3 db (ADC1, ADC2, ADC3)	2 db (ADC1, ADC2)
Felbontás	6, 8, 10, 12 bit	12 bit
Üzemmód	független, duál, tripla	független, duál
VDDA	1.7 – 3.6 V	2.4 – 3.6 V
VREF	(VDDA-1.2 V) – VDDA	2.4 V – VDDA
fADC (VDDA ≥ 2.4 V)	0.6 – 36 MHz (typ: 30)	0.6 – 14 MHz
tconv	0.5 µs @ 12 bit 0.3 µs @ 6 bit	1 µs @ 12 bit

- Szingli és folytonos konverziós módok, önkalibrálás
- Pásztázó mód több csatorna automatikus méréséhez
- Csatornánként beállítható sorrend és mintavételezési idő
- Injektált csatornák közbevetett mérése (max. 4 db)
- Külső triggerelés a reguláris és az injektált csatornák méréséhez
- DMA adatátviteli kérelem generálása konverzió végén

Az ADC-k blokkvázata



Az analóg bemenetek kiosztása

- Az alábbi táblázatban összefoglaltuk, hogy a **NUCLEO-F446RE** kártya esetén melyik analóg bemenet melyik GPIO portkivezetéshez csatlakozik
- A nem Arduino kompatibilis kivezetések a Morpho csatlakozókon érhetők el

ADC csatorna	Pin	Arduino	ADC csatorna	Pin	Arduino
Ain[0]	PA_0	**A0**	Ain[10]	PC_0	**A5**
Ain[1]	PA_1	**A1**	Ain[11]	PC_1	**A4**
Ain[2]	PA_2	D1	Ain[12]	PC_2	
Ain[3]	PA_3	D0	Ain[13]	PC_3	
Ain[4]	PA_4	**A2**	Ain[14]	PC_4	
Ain[5]	PA_5	D13	An[15]	PC_5	
Ain[6]	PA_6	D12			
Ain[7]	PA_7	D11			
Ain[8]	PB_0	**A3**			
Ain[9]	PB_1				

Az ADC-k üzemmódjainak áttekintése

- **Független módok:**
  - Egy csatorna, szingli konverzió ← mbed API alapértelmezett mód
  - Pásztázás (több csatorna), szingli konverzió
  - Egy csatorna, folyamatos mód
  - Pásztázás, folyamatos mód
  - Injektált csatornák
- **Duál/Tripla módok:**
  - Duál, reguláris, szimultán mód
  - Duál, gyors átlapolásos mód
  - Duál, lassú átlapolásos mód
  - Duál alternáló triggerelésű mód
  - Duál kombinált: reguláris/injektált szimultán mód
  - Duál kombinált: injektált szimultán + átlapolásos mód

Ajánlott olvasmány:

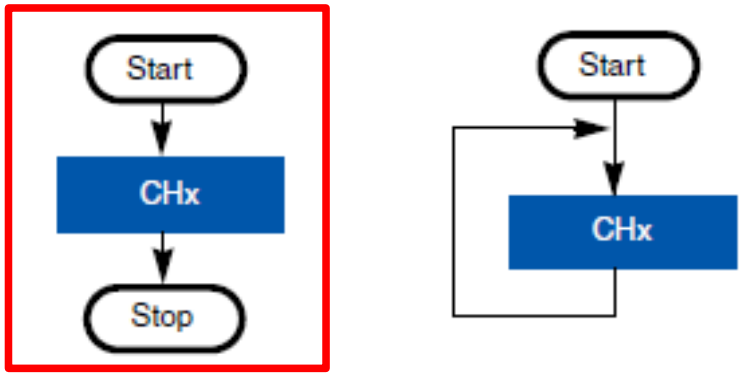
STmicro Application Note:
AN3116 – STM32's ADCs modes and their application

AN3116 mintaprogramok:
STSW-STM32028

Független módok

Egy csatorna

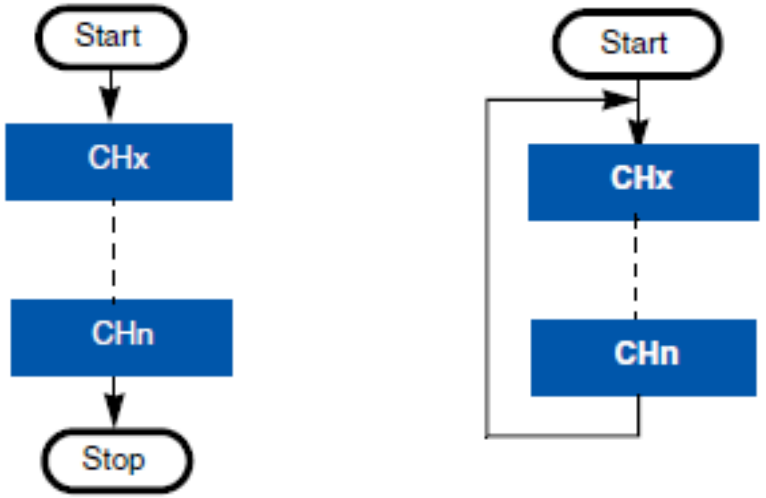
szingli konverzió folyamatos konverzió



mbed API alapértelmezett mód

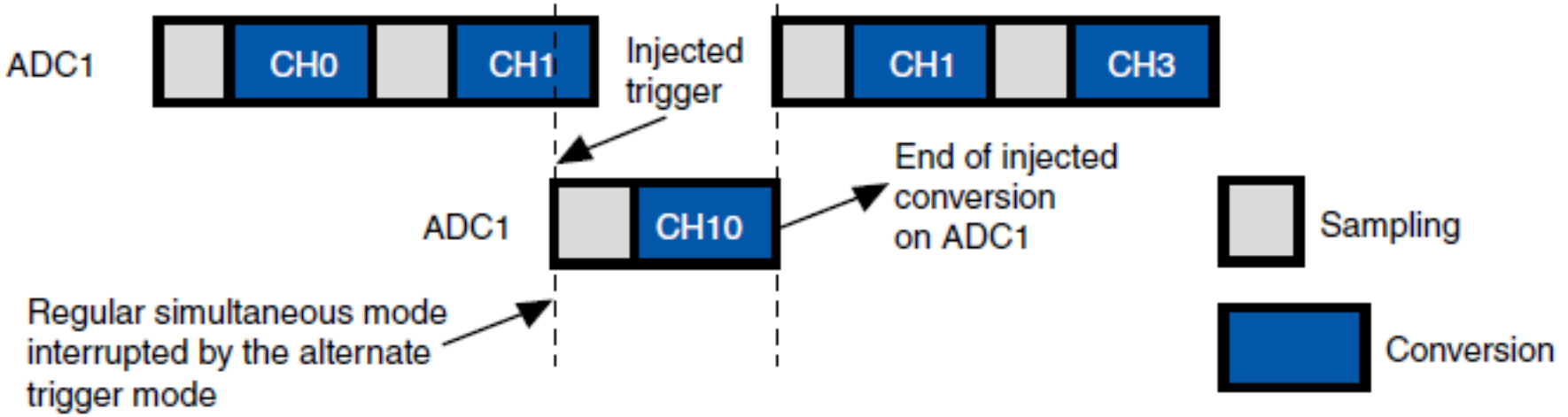
Pásztázás (több csatorna) (csak DMA-val lehetséges)

szingli konverzió folyamatos konverzió



Injektált konverziós mód

(legfeljebb 4 csatorna injektálható)



Az AnalogIn periféria-könyvtár

- Az **AnalogIn** objektumosztály az ADC konfigurálására és kezelésére szolgál. Része a standard **mbed API**-nak, így nem kell külön importálni
- Az **AnalogIn** objektumosztály tagfüggvényeit az alábbi táblázatban foglaltuk össze

Függvény	Használat
AnalogIn név(pin)	Létrehoz egy "*név*" nevű **AnalogIn** objektumot és a *pin* paraméterrel megadott kivezetést az ADC egyik analóg bemeneteként konfigurálja.
read()	Egy konverziót indít, majd visszatér az eredménnyel. A visszatérési érték 0 - 1.0 közötti float érték, amely a mért feszültséget a **VREFH** analóg referencia megfelelő hányadaként fejezi ki. Ha a mért feszültséget Voltban akarjuk kifejezni, akkor szorozzuk meg a kapott értéket **VREFH** értékével (3,3 V).
read_u16()	Egy konverziót indít, majd visszatér az eredménnyel. A visszatérési érték 0 - 0xFFFF közötti uint16 érték, amely a mért feszültséget **VREFH/65536** egységekben fejezi ki.
operator float()	Rövidített alak **read()** helyett

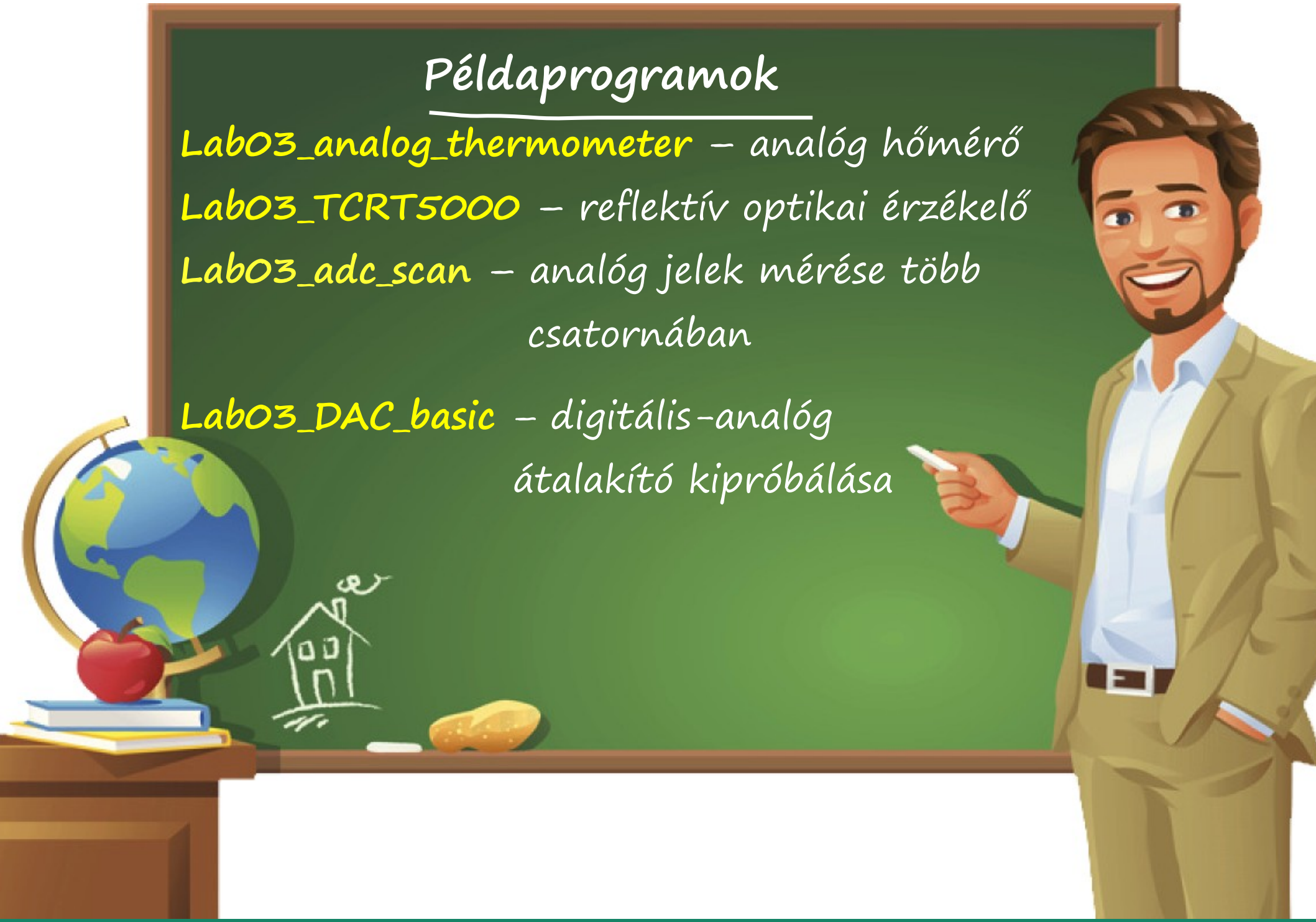
Példaprogramok

Lab03_analog_thermometer – analóg hőmérő

Lab03_TCRT5000 – reflektív optikai érzékelő

Lab03_adc_scan – analóg jelek mérése több csatornában

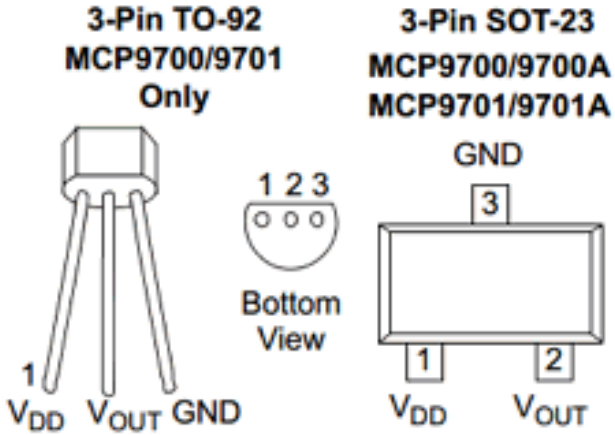
Lab03_DAC_basic – digitális-analóg átalakító kipróbálása

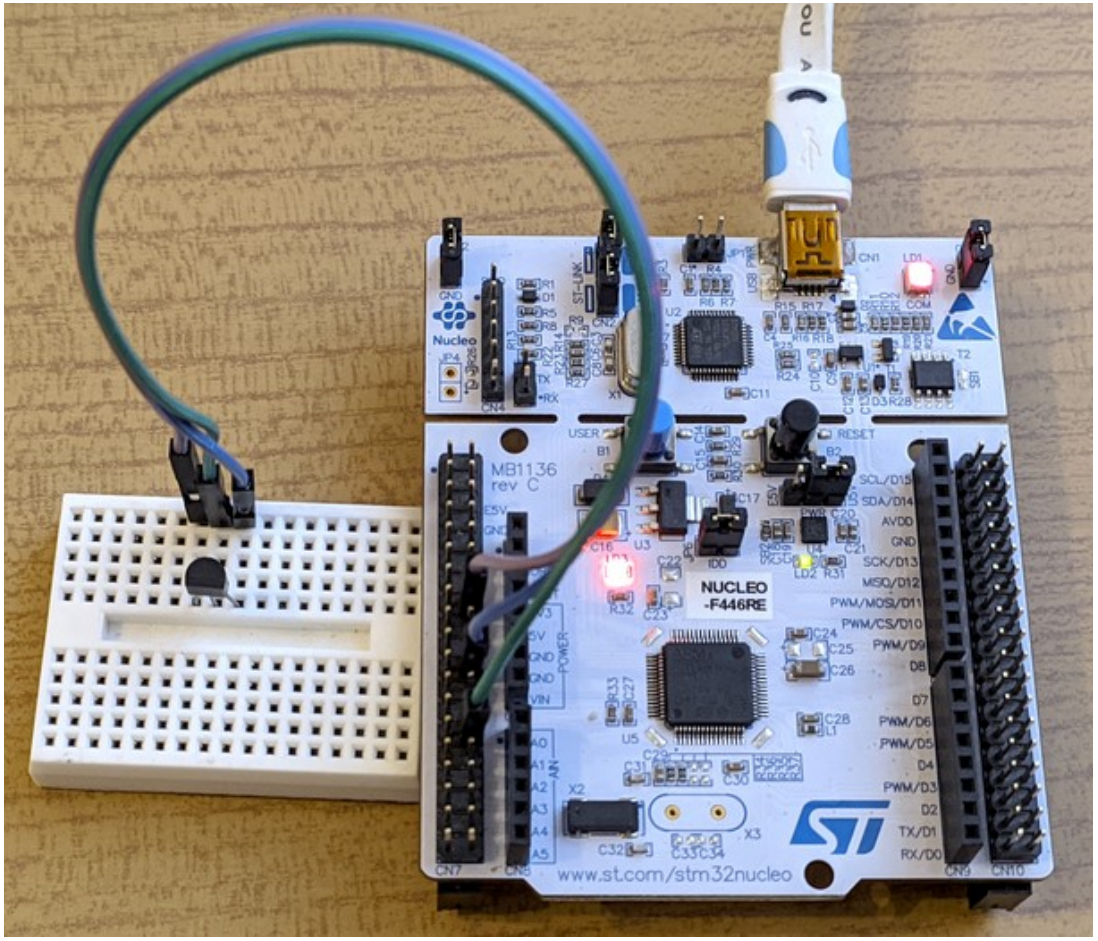


Analóg hőmérő használata

- Első mintapéldánkban (**Lab03_analog_thermometer** projekt):
  - Egy külső hőmérő jelét mérjük (**AIN0.** csatorna, **PA_0** bemenet)
  - Az eredményt kiíratjuk a soros porton keresztül
  - Több mérés átlagolásával csökkentjük a zavarjelek hatását
- Az analóg hőmérő esetünkben egy **MCP9700 (**vagy **TMP36)** melynek paraméterei:

VDD = 3,3 – 5 V
VOUT = 500 mV @ 0 ℃
slope = 10 mV/℃





Lab03_analog_thermometer/main.cpp

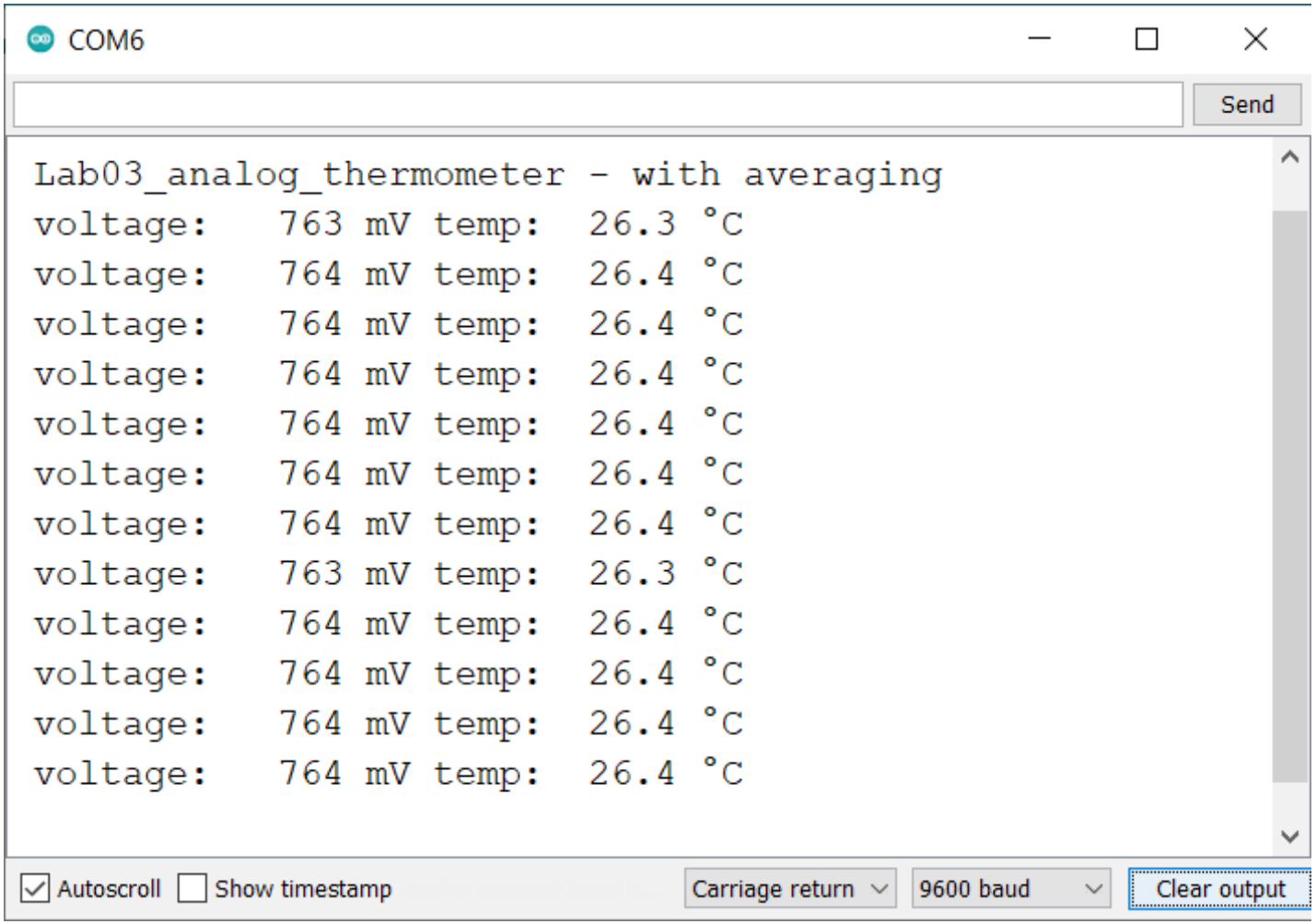
- Annyi ciklusban mérünk, amennyi VREFH mérőszáma, így nem kell sem VREFH-val szorozni, sem a mérések számával beosztani

Import into Compiler

```
#include "mbed.h"

AnalogIn ain(A0);                                   // Analog input at PA0
uint32_t mysum;                                     // Used for summation

int main()
{
    printf("\r\n Lab03_analog_thermometer - with averaging\r\n");
    while(1) {
        mysum = 0;
        for(int i=0; i<3300; i++) {
            mysum += ain.read_u16();                // sum up raw 16-bit data
        }
        float voltage = mysum>>16;                  // fast divide by 65536
        float tempC = (voltage - 500)/10;           // tempereature in Celsius
        printf(" voltage: %5.0f mV temp: %5.1f °C\r\n",voltage,tempC);
        wait(2);
    }
}
```

Lab03_analog_thermometer

- A program az alapértelmezett 9600 bps sebességgel küldi az adatokat, amelyeket egy terminálablakban nézhetjük meg



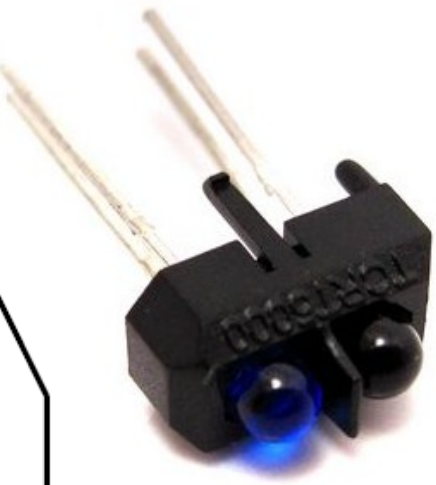
TCRT5000 reflektív optikai érzékelő

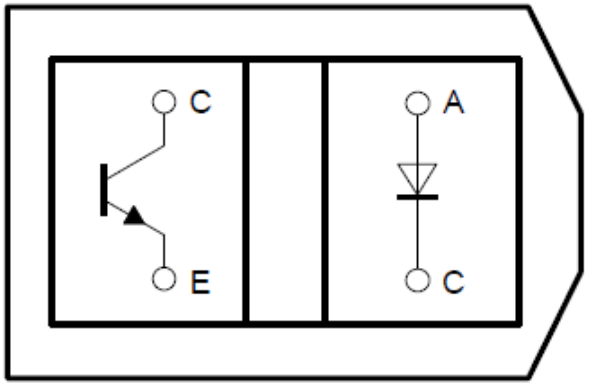
Egy IR LED-et és egy fototranzisztort tartalmaz.

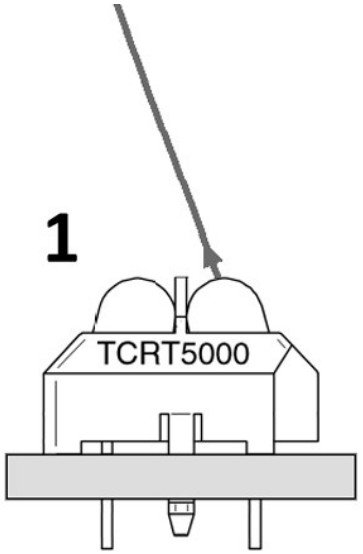
Működési elv:

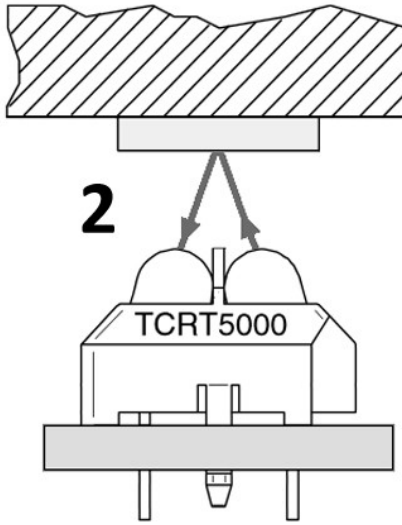
1. Ha nincs akadály, a fény nem verődik vissza, a tranzisztor nem érzékel fényt.

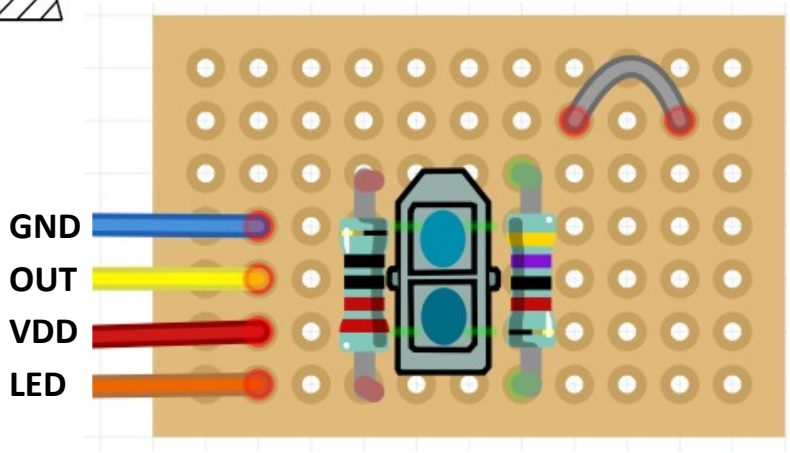
2. Akadály esetén a távolságtól és a reflexiós tényezőtől függ a fényvisszaverődési arány.

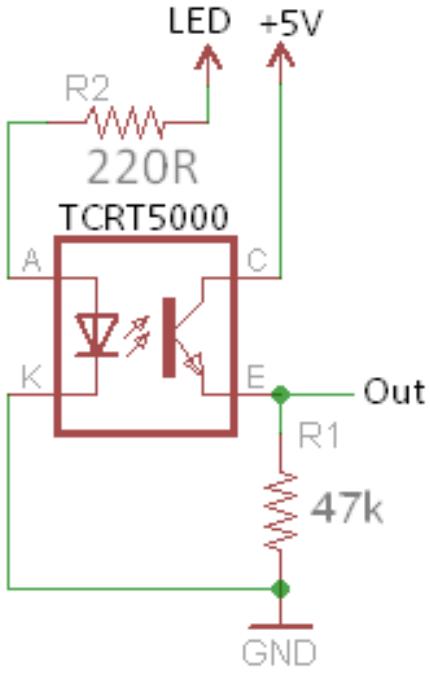












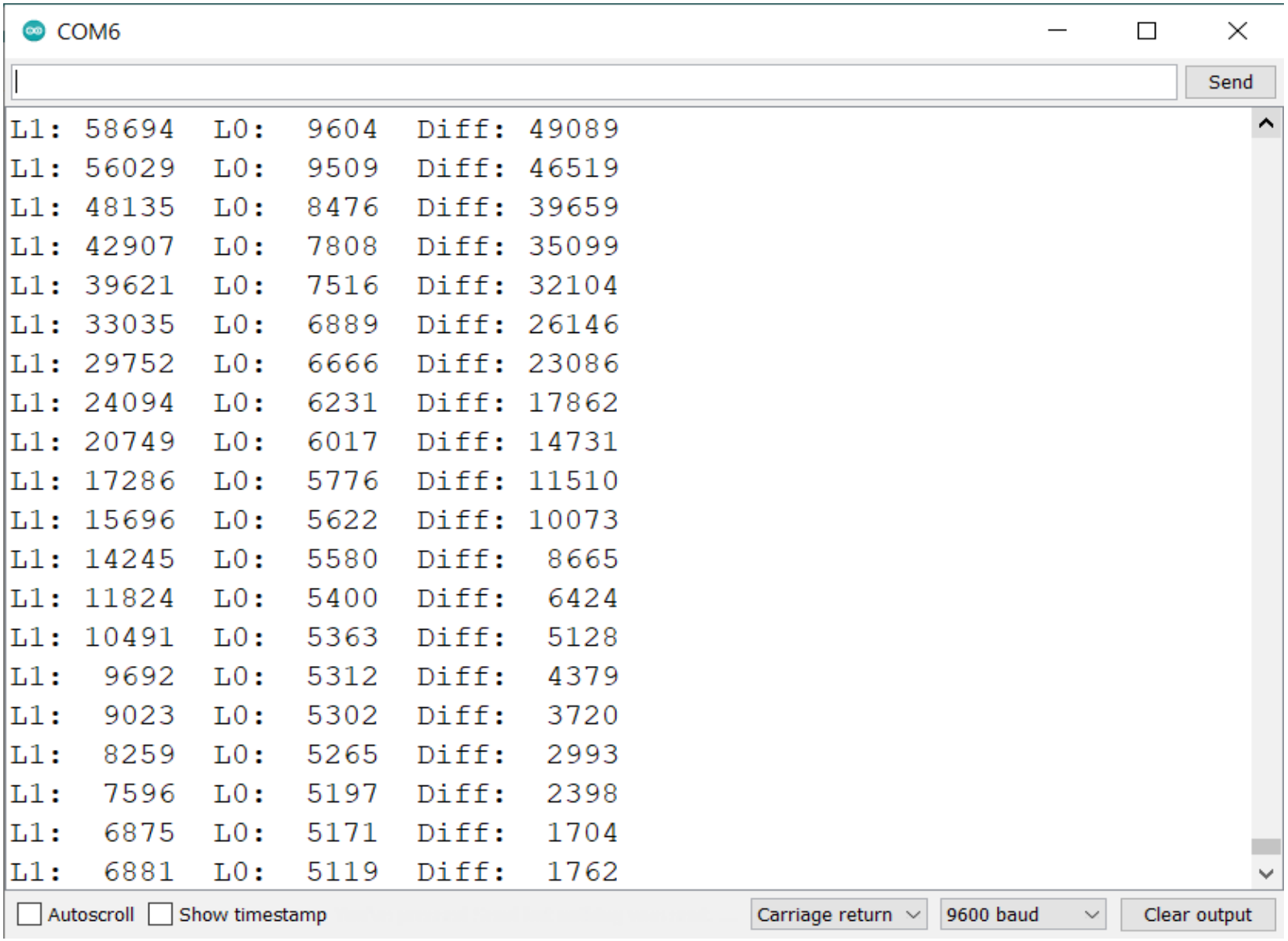
Lab03_TCRT5000/main.cpp

Import into Compiler

```
#include "mbed.h"
AnalogIn cds(A0);                               // Analog input at PA0
DigitalOut led(D10);                            // Led controlled by D10

uint32_t measLight(int nmeas) {
    uint32_t sum0, sum1, diff;
    led = 0; sum0 = 0;
    for (int i = 0; i < nmeas; i++) sum0 += cds.read_u16();
    led = 1; sum1 = 0;
    for (int i = 0; i < nmeas; i++) sum1 += cds.read_u16();
    if (sum0 > sum1) {
        diff= 0;
    } else {
        diff = (sum1 - sum0) / nmeas;
    }
    printf("L1: %5d  L0: %5d  Diff: %5d\r\n",sum1/nmeas, sum0/nmeas, diff);
    return diff;
}

int main() {
    printf("\r\n Lab03_light_sensor - with averaging\r\n");
    while(1) {
        measLight(1000);
        wait(2);
    }
}
```

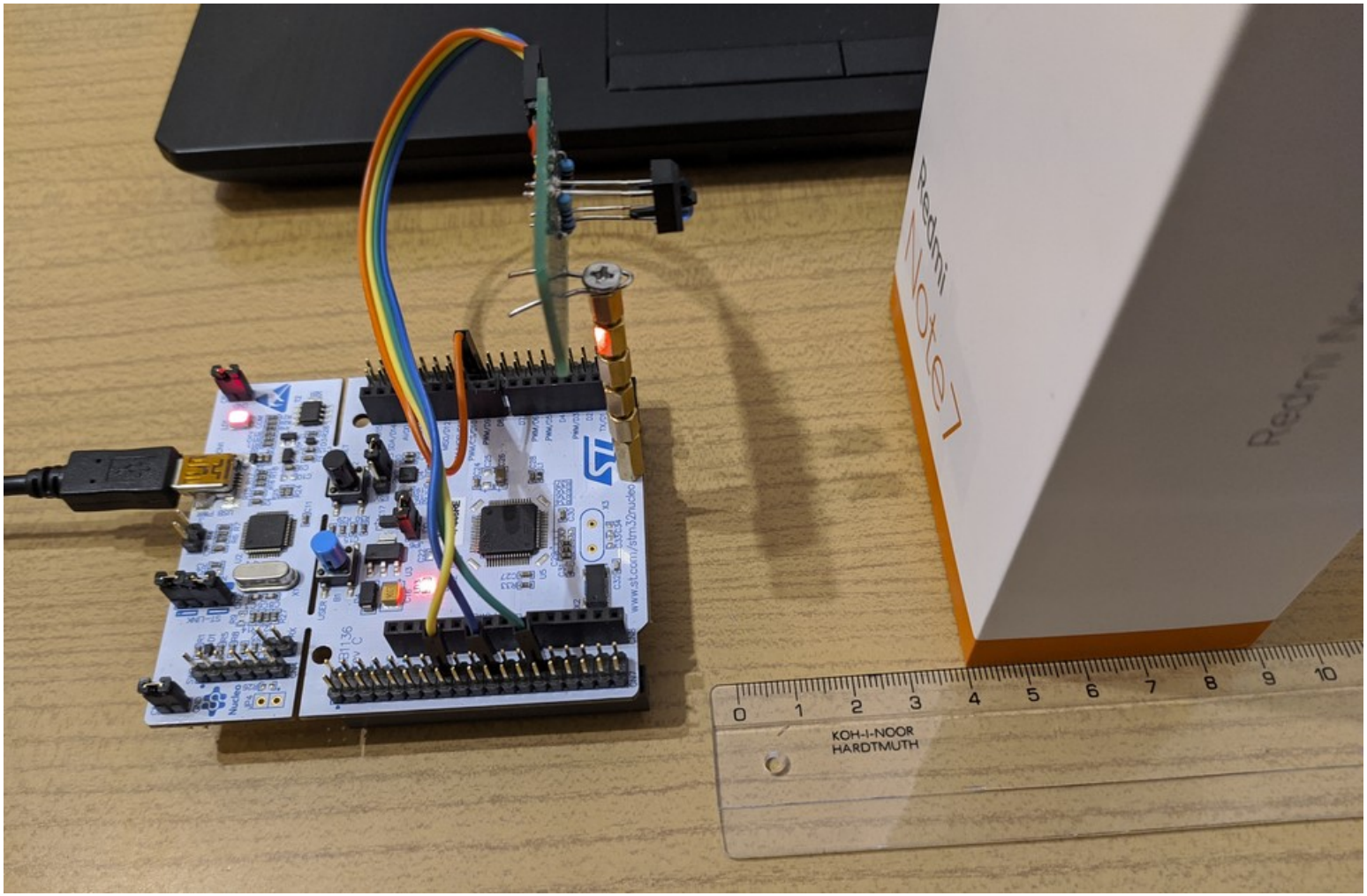
Lab03_TCRT5000

- A program egy futási eredménye az alábbi ábrán látható



Lab03_TCRT5000

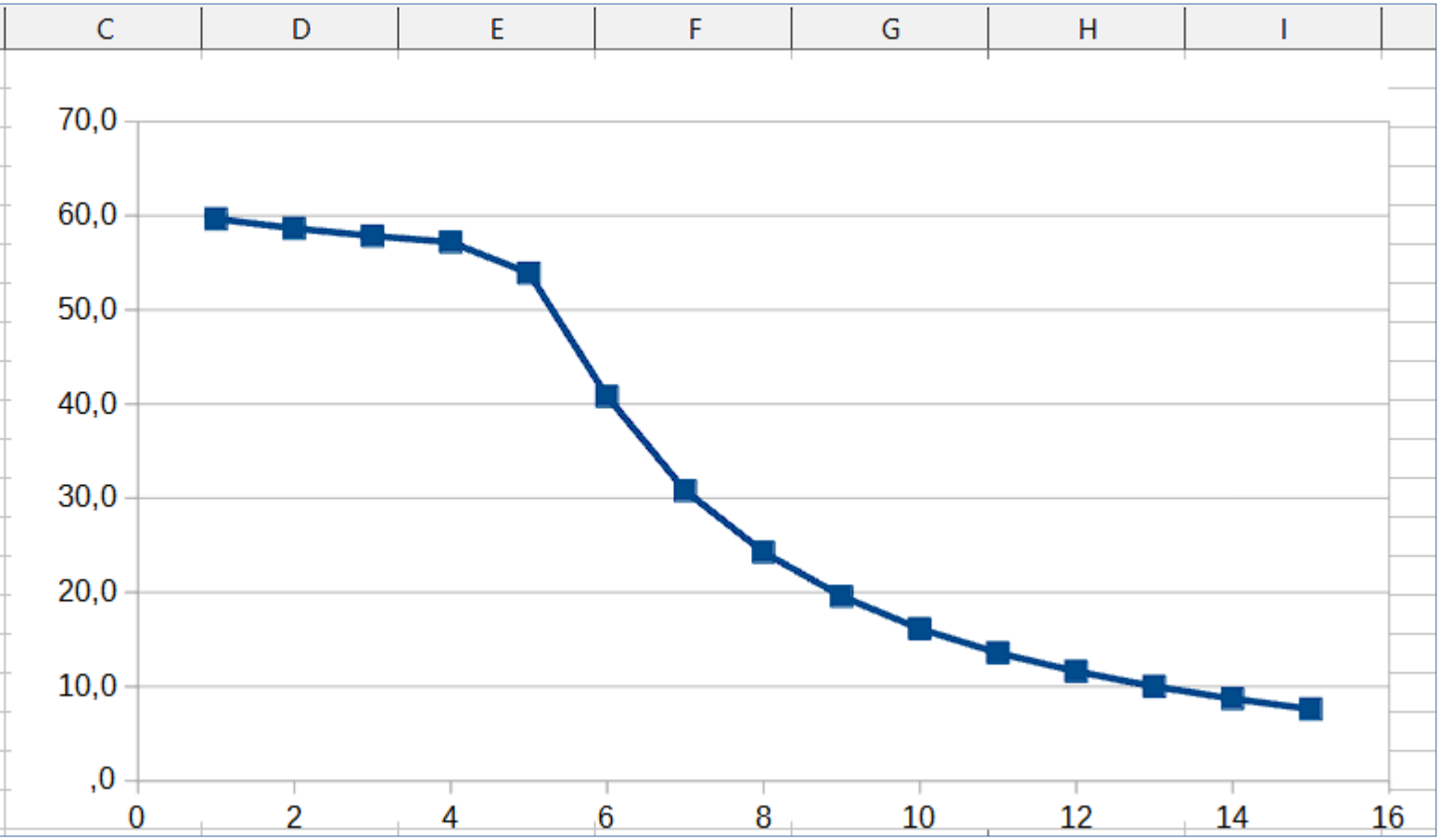
- A szenzor jelleggörbéjét a fényvisszaverő felület távolságának változtatásával vettük fel, a vonalzó mellett csúsztatva



Lab03_TCRT5000

- A táblázat és az ábra egy mérési sorozat eredményét mutatja
- A távolságot cm-ben mértük, a második oszlopban pedig az ADC által visszaadott 16 bites értékek láthatók 1000-rel beosztva

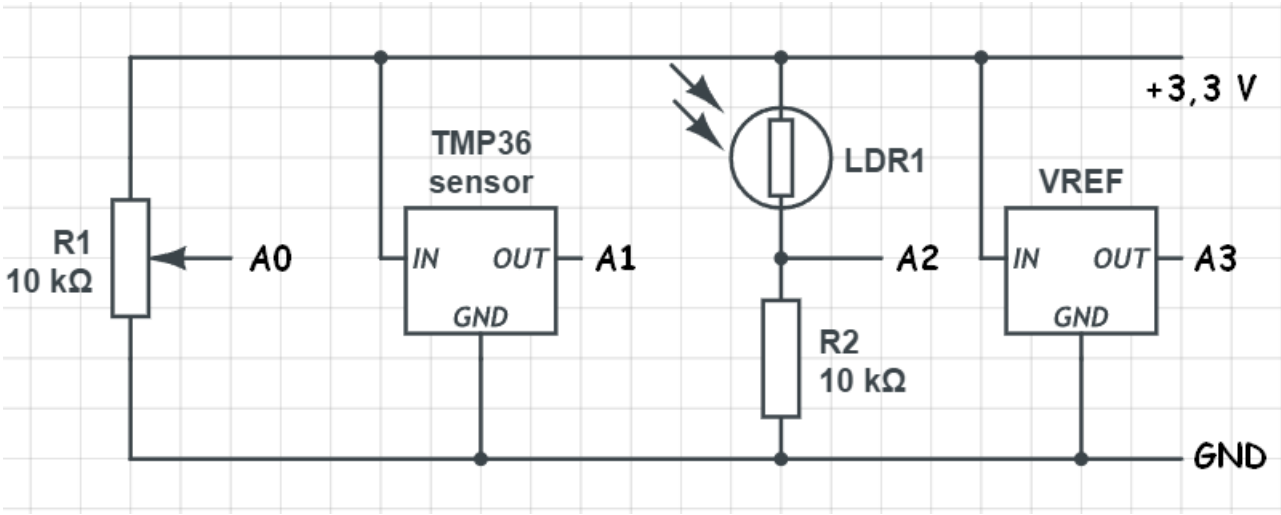
	A	B
1		
2	**d [cm]**	**diff**
3	1	59,7
4	2	58,7
5	3	57,8
6	4	57,2
7	5	53,9
8	6	40,8
9	7	30,8
10	8	24,3
11	9	19,6
12	10	16,1
13	11	13,6
14	12	11,6
15	13	10,0
16	14	8,7
17	15	7,6
18	100	1,5
19		
20		

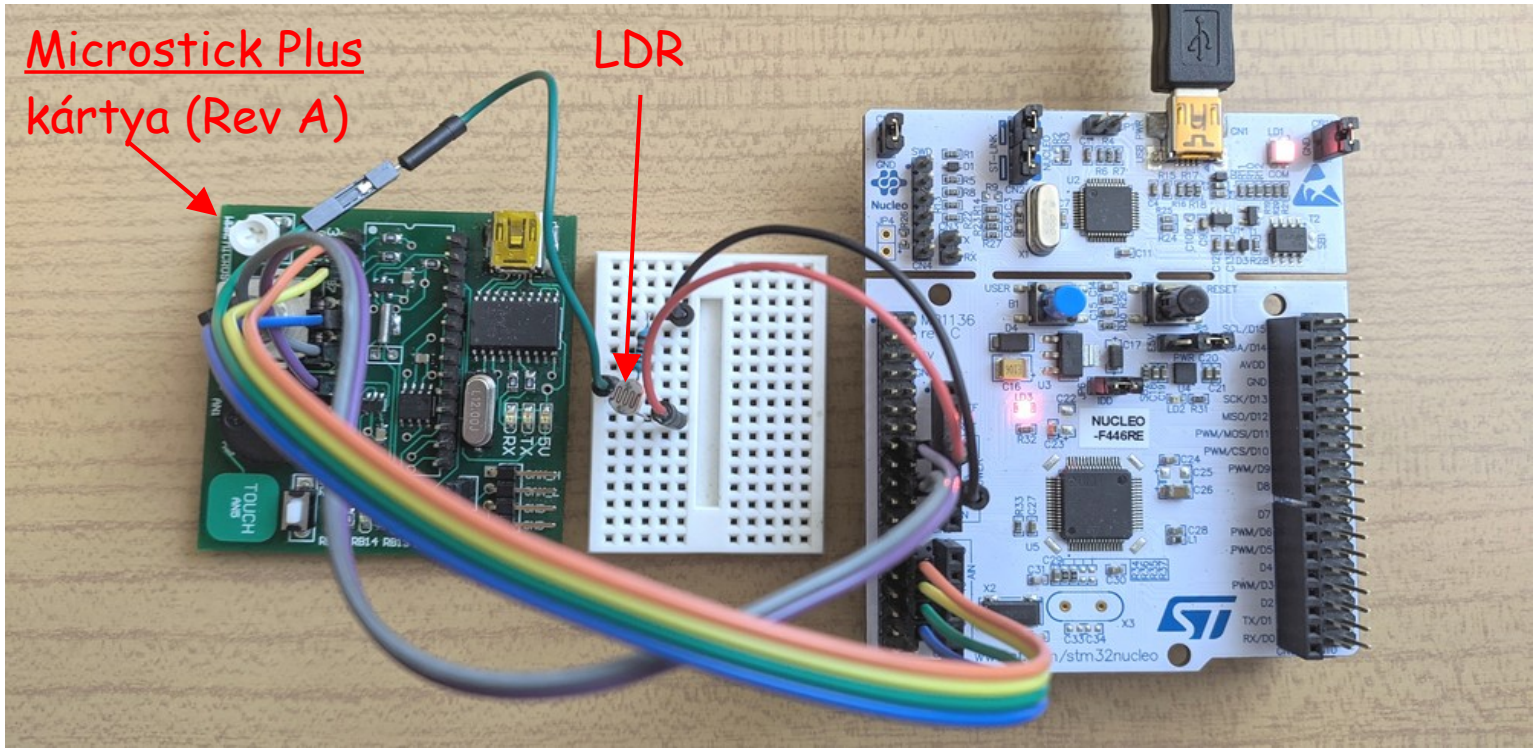


Lab03_adc_scan

- Ebben a projektben négy analóg csatorna jelét mérjük egymás után, 1000-szer ismételve (átlagolás)
- A mV-ra átszámított értékeket a soros porton kiíratjuk
- A mért jelek:

A0 – potméter
A1 – hőmérő
A2 – LDR osztó
A3 – 2,5V ref.





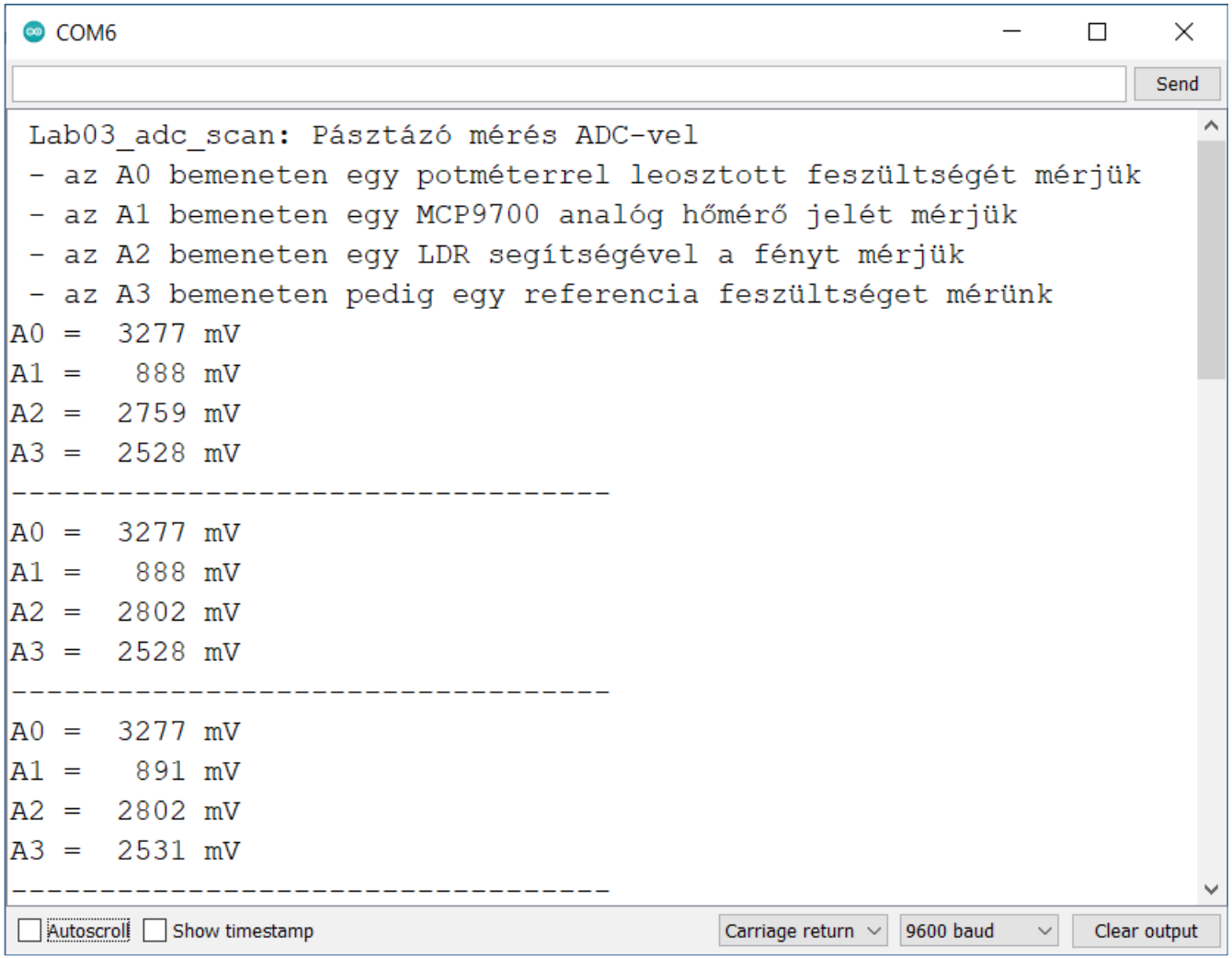
Lab03_adc_scan/main.cpp

Import into Compiler

```
#include "mbed.h"
const float VREFH = 3.3;     // VrefH referencia feszültség [V]
int data[4] = {0, 0, 0, 0}; // A mérési adatok tárolója
float v[4];                  // Feszültség mV-okban
AnalogIn potm(A0);           // Analog input at PA_0
AnalogIn homero(A1);         // Analog input at PA_1
AnalogIn ldr(A2);            // Analog input at PA_4
AnalogIn vref(A3);           // Analog input at PB_0

int main() {
    printf(" Lab03_adc_scan: Pásztázó mérés ADC-vel\r\n");
    while(1) {
        for(int i=0; i<4; i++) data[i]=0;
        for(int i=0; i<1000; i++) {
            data[0] += potm.read_u16();
            data[1] += homero.read_u16();
            data[2] += ldr.read_u16();
            data[3] += vref.read_u16();
        }
        for(int i=0; i<4; i++) {
            v[i] = VREFH*(data[i]>>16);        // V = NADC * VREFH / 2^16
            printf(" A%d = %5.0f mV\r\n",i,v[i]);
        }
        printf("--------------------------------\r\n");
        wait(2);
    }
}
```

Lab03_adc_scan

- Az ábrán a program egy futási eredménye látható

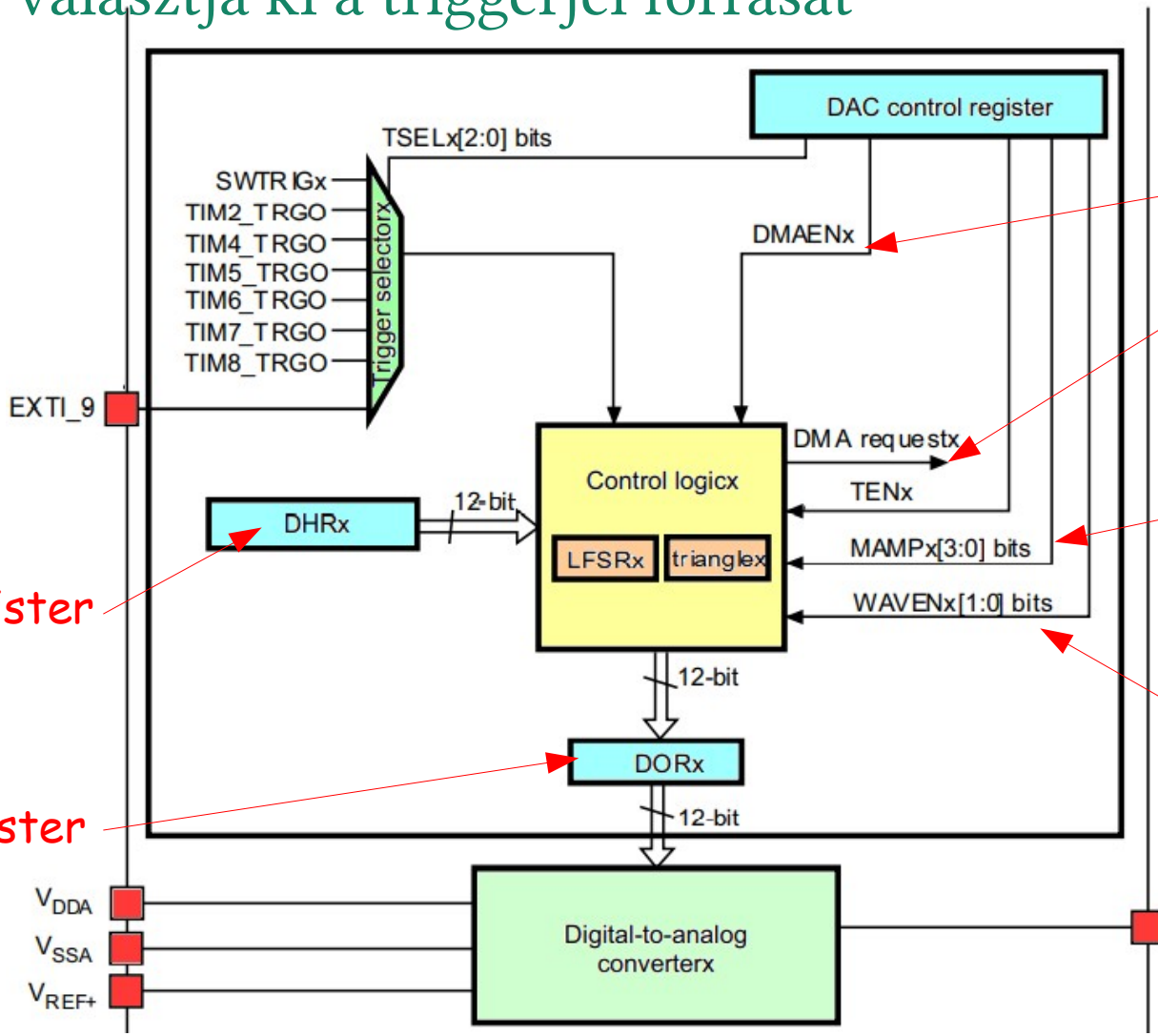


Digitális-analóg átalakító (DAC)

- Az **STM32F446RE** mikrovezérlő két, beépített 12 bites DAC csatornával rendelkezik, amelyek kimenetei a **PA_4** és **PA_5** kivezetések (nem áthelyezhető)
- A két **DAC** csatorna működhet függetlenül, vagy szinkronizáltan
- DMA-képes mindkét csatorna
- Külső és belső triggerjel is használható a konverzió indításához
- Feszültségreferencia bemenet (1.8 V ≤ VREF+ ≤ VDDA)
- Beépített zaj- illetve háromszögjel generátorok
- A DAC csatornák lehetőségeinek teljes kihasználása csak regiszter szintű programozással, vagy a HAL függvénykönyvtár használatával lehetséges. Ezekről részletes információt a **2020/2021 évi STM32 tanfolyam** 6. és 7. előadásában találhatunk

A digitális-analóg átalakító felépítése

- A **DMAEN*x*** bit engedélyezi a modul működését, a **TSEL*x***[2:0] bitcsoport választja ki a triggerjel forrását



Engedélyező bit

DMA átviteli kérelem

MAMPx – mask vagy amplitudó beállítás

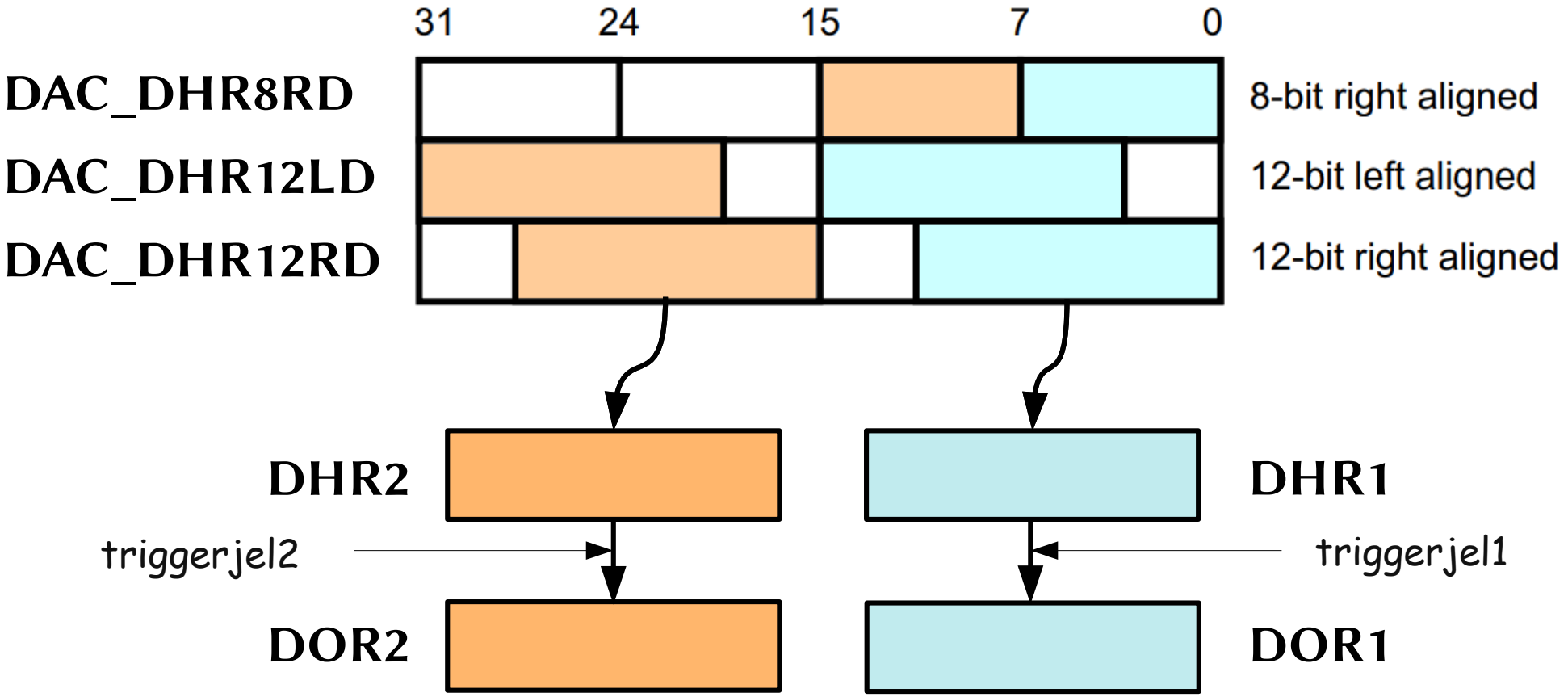
WAVENx – jelgenerátor mód választás

Data Holding Register (közvetlenül nem címezhető)

Data Output Register

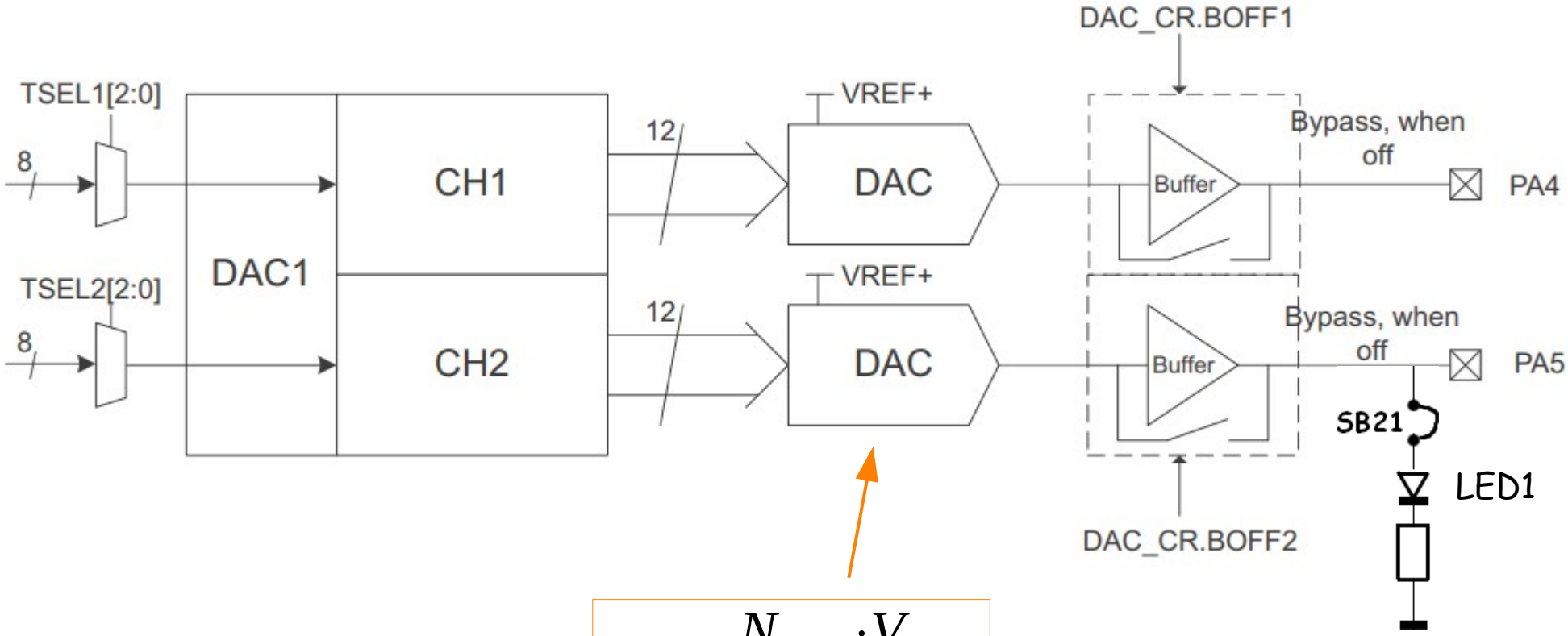
DAC adatformátum

- Az adatbeírás nem közvetlenül, hanem a jobbra-balra igazítást, ill. helyiérték-eltolást szervező regisztereken keresztül történik, amelyekből három készlet van (1. és 2. csatorna, illetve duál mód)
- Itt csak a duál módú beíráshoz tartozó regisztereket tüntettük fel



Kimeneti buffer

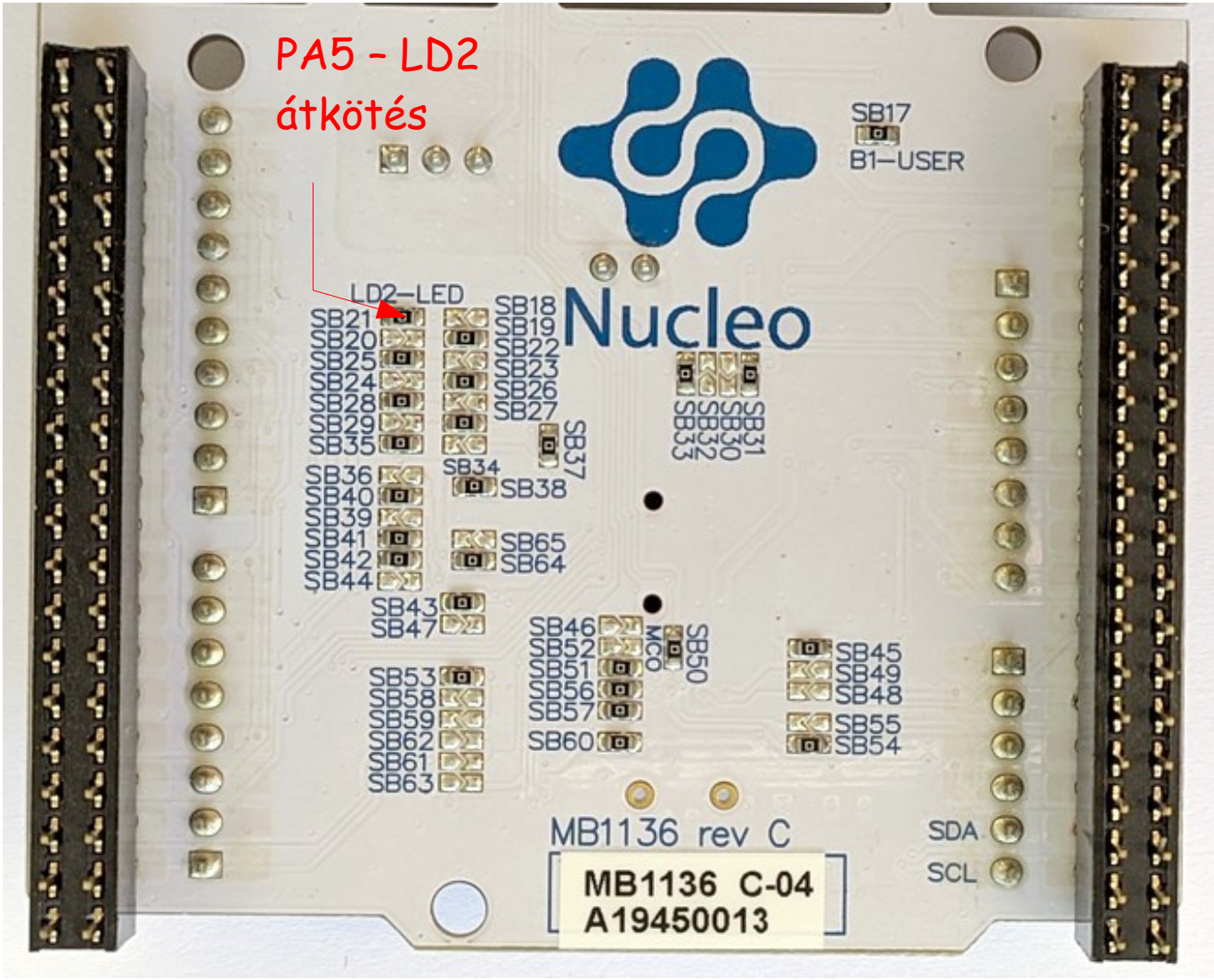
- A **DAC** kimeneteihez egy-egy buffer erősítő is tartozik, amelyek engedélyezhetők, vagy áthidalhatók, szerepük a kimeneti ellenállás csökkentése – a nagyobb áramfogyasztás árán



$$U_{KI} = \frac{N_{DAC} \cdot V_{REF}}{4096}$$

A DAC 2. csatorna „felszabadítása”

- A **DAC** 2. csatornája arra a **PA_5** kivezetésre csatlakozik, amelyre az **LD2** LED is rá van kötve. A LED-et az **SB21** átkötés leforrasztásával tudjuk leválasztani
- Az **SB** elnevezés egyébként az angol *solder bridge* rövidítése



Az AnalogOut periféria-könyvtár

- Az **AnalogOut** objektumosztály a DAC konfigurálására és kezelésére szolgál. Mivel része a standard mbed API-nak, így nem kell külön importálni
- Az **AnalogOut** objektumosztály tagfüggvényeit az alábbi táblázatban foglaltuk össze

Függvény	Használat
AnalogOut név(pin)	Létrehoz egy "*név*" nevű **AnalogOut** objektumot és a *pin* paraméterrel megadott kivezetést a DAC analóg kimeneteként konfigurálja. Esetünkben a pin értéke csak **PA_4**, vagy **PA_5** lehet
write(data**)**	A DAC kimenetet beállítja a *data*VREFH* feszültségre. A paraméter 0 - 1.0 közötti float érték, amely a beállítandó feszültséget a **VREFH** analóg referencia megfelelő hányadaként fejezi ki.
write_u16(data**)**	A DAC kimenetét beállítja a data/0xFFFF*VREFH feszültségre. A paraméter értéke 0 - 0xFFFF közötti uint16 érték, amely a mért feszültséget **VREFH/65535** egységekben fejezi ki.
read()	Visszaolvassa a legutoljára beállított értéket (0 - 1.0 közötti float érték)
operator = data	Rövidített alak **write(**data**)** helyett
operator float()	Rövidített alak **read()** helyett

Lab03_DAC_basic

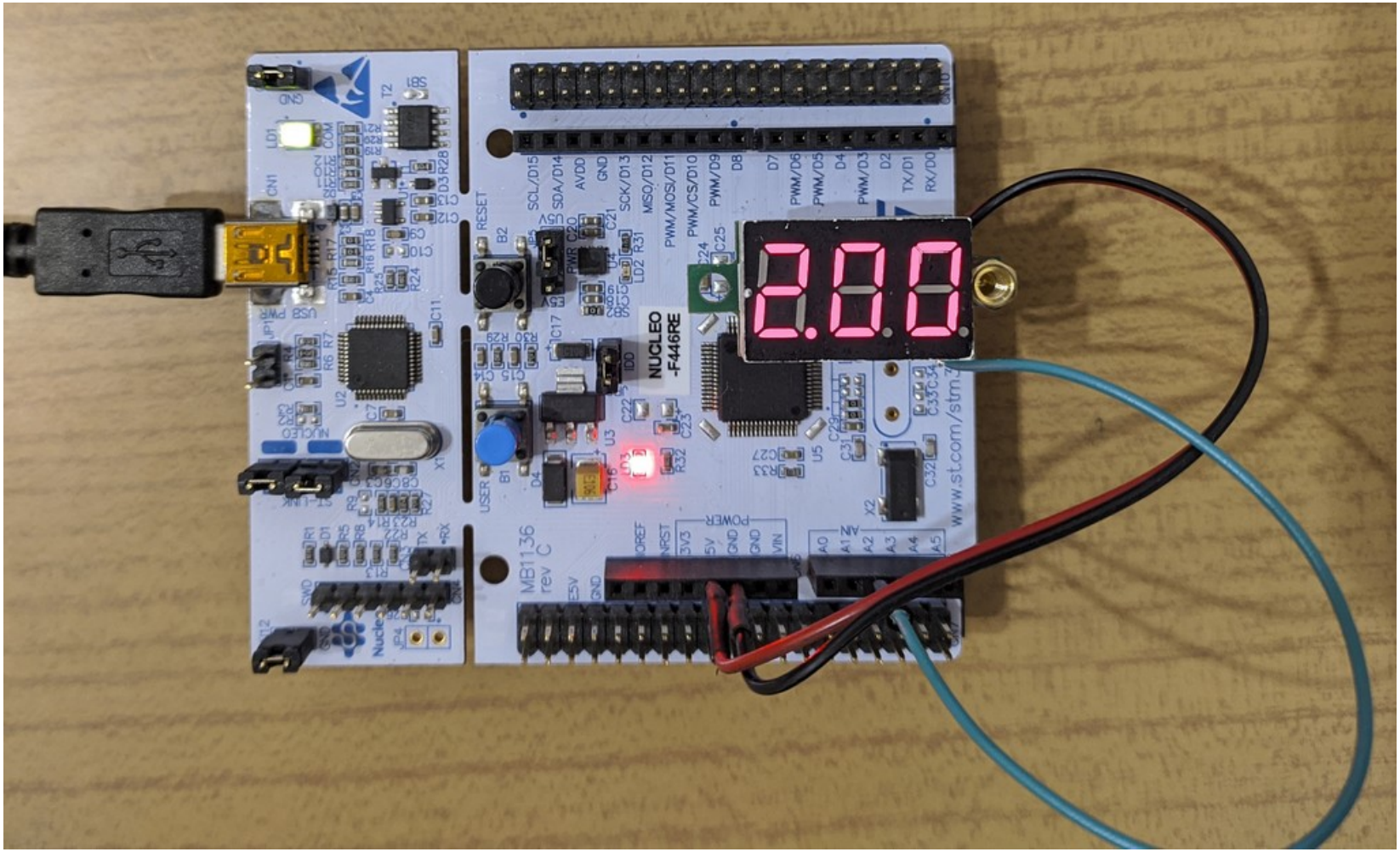
- Az alábbi programban a DAC_OUT1 (**PA_4**) kimenetet használva ciklikusan ismételve 0V, 1.0V, 2.0V és 3,0V feszültséget állítunk be, s az egyes lépések között 5 másodperc szünetet tartunk
- A kimenő feszültséget kéziműszerrel ellenőrizhetjük

Import into Compiler

```
#include "mbed.h"

AnalogOut dac(PA_4);

int main()
{

    while(1) {
        for (int i = 0; i < 4; i++) {
            dac.write(float(i)/3.3f);
            printf(" DAC: %4.1f\n", dac.read());
            wait(5);
        }
    }
}
```

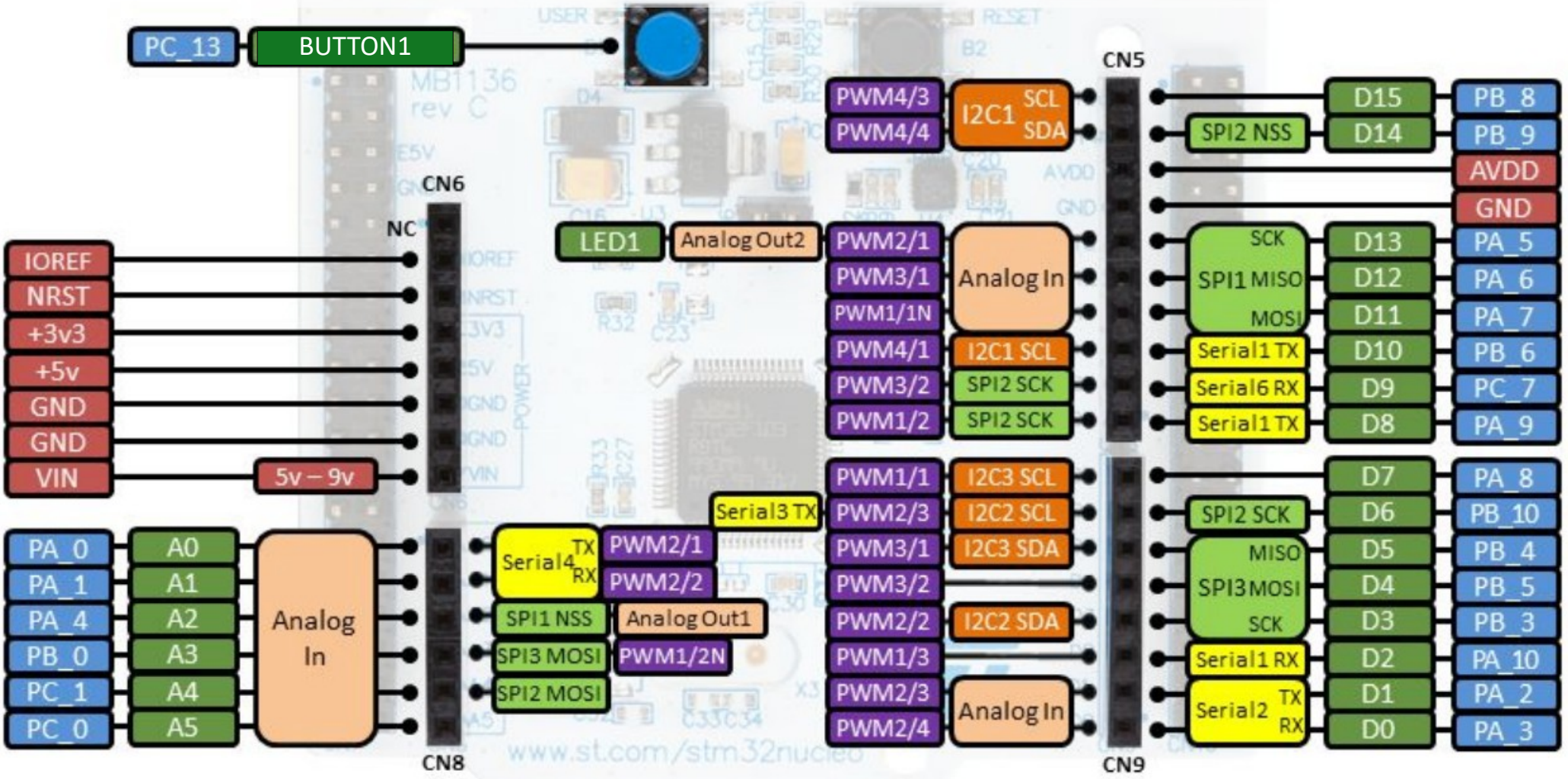
Lab03_DAC_basic

- Az alábbi árán egy filléres digitális voltmérőt kötöttünk a **PA_4** kivezetésre



Arduino kompatibilis kivezetések

- Kivezetés azonosításra a kék, illetve a sötétzöld címkék használhatók (pl. PA_5, D13, vagy LED1)



Morpho kivezetések

Nucleo F446RE

Morpho Headers