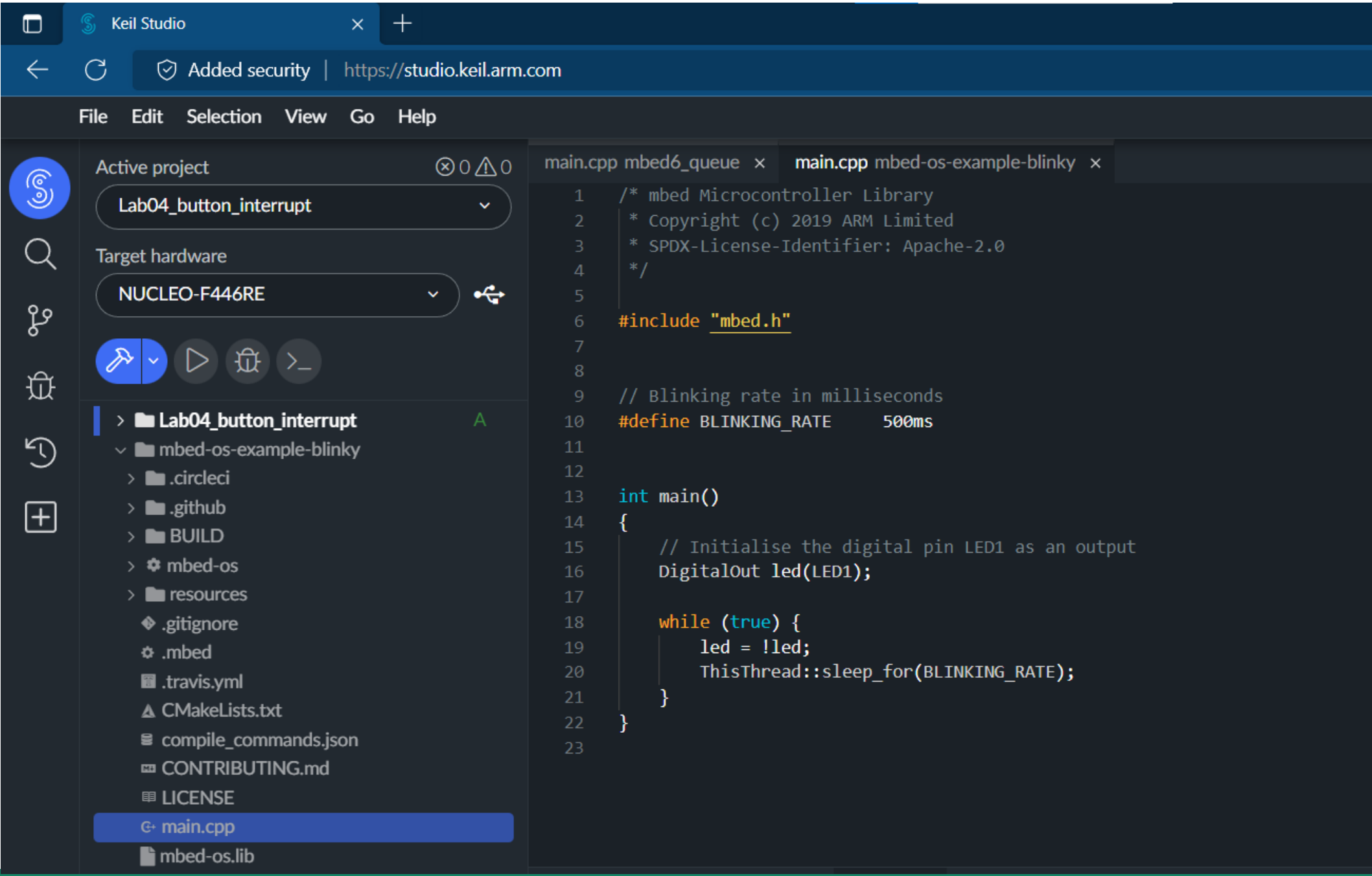
1. Mbed OS és a felhőalapú Keil Studio

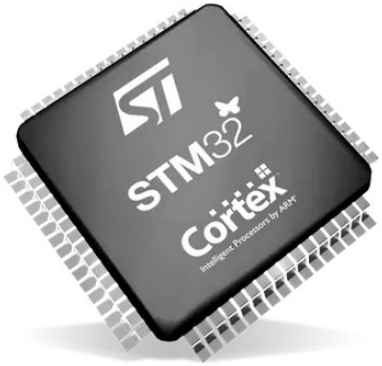


Felhasznált és ajánlott irodalom

- 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
- Cserny István: A FRDM-KL25Z kártya programozása mbed környezetben
- **ARM mbed honlap: https://os.mbed.com/**
- **ARM Keil Studio: https://studio.keil.arm.com/**
- ARM mbed OS Documentation: https://os.mbed.com/docs/mbed-os/
- Keil Studio manual: https://developer.arm.com/documentation/102 497/latest
- ARM mbed forráskód: https://github.com/ARMmbed/mbed-os

Adatlapok:

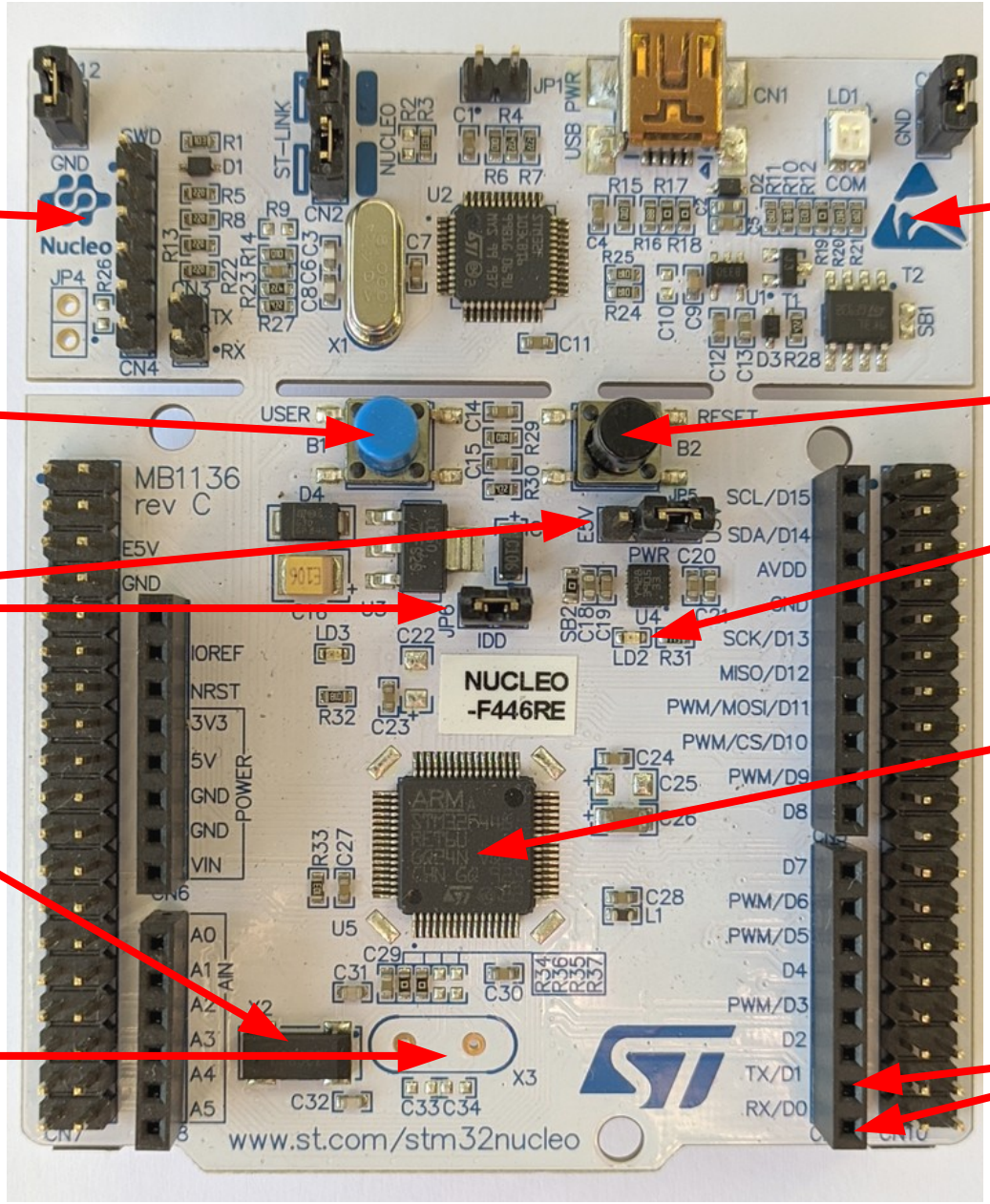
- **STM32F446RE** adatlap és termékinfo
- **STM32F446** Family Reference Manual



Programfejlesztés Mbed környezetben

- Tavaly bemutattuk a C/C++ alapú **ARM mbed** környezetet, amellyel a beágyazott rendszerek fejlesztése könnyen és hatékonyan végezhető (gyors prototípus készítés, *„proof of concept”*)
- Az **STM32F446RE** mikrovezérlő és a **NUCLEO-F446RE** kártya segítségével a gyakorlatban is kipróbálhattuk a bemutatott mintaprogramokat, illetve saját alkalmazásokat is fejleszthetünk rá
- A **NUCLEO-F446RE** kártyára épített ST-Link V2-1 eszközt nemcsak az STM32 mikrovezérlő programozására használhatjuk, hanem USB-soros átalakítóként a PC-vel történő kommunikációra, illetve megfelelő fejlesztői környezetben (pl. **Mbed Studio,** vagy a mai előadásban bemutatott **ARM Keil Studio**) hardveres nyomkövetésre is.
- A **NUCLEO-F446RE** kártyához csatlakoztatott kiegészítők (szenzorok, kijelzők, aktuátorok) segítségével *fizikai számítástechnikai* feladatokat végezhetünk, azaz olyan interaktív rendszereket hozhatunk létre hardver és szoftver segítségével, melyek képesek érzékelni a világban létrejövő jeleket és reagálni is tudnak rá

A NUCLEO-F446RE fejlesztői kártya



ST Link kimenet

Felhasználói nyomógomb [PC_13]

Átkötések

32 kHz kvarc

8 MHz kvarc nincs beépítve (az órajel az STLink-ből érkezik)

STLink V2-1 programozó/ hibavadász

RESET gomb

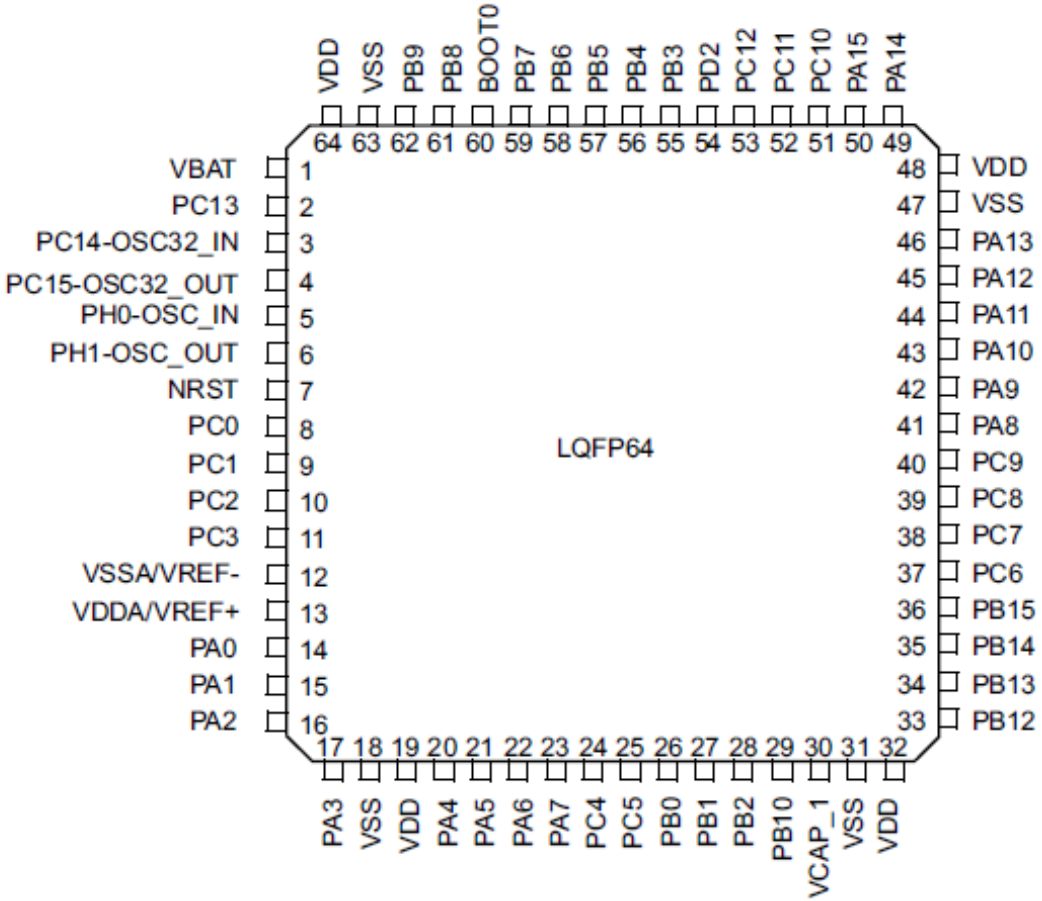
LED

STM32F446RE MCU

Soros port a PC-hez (STLinken át)

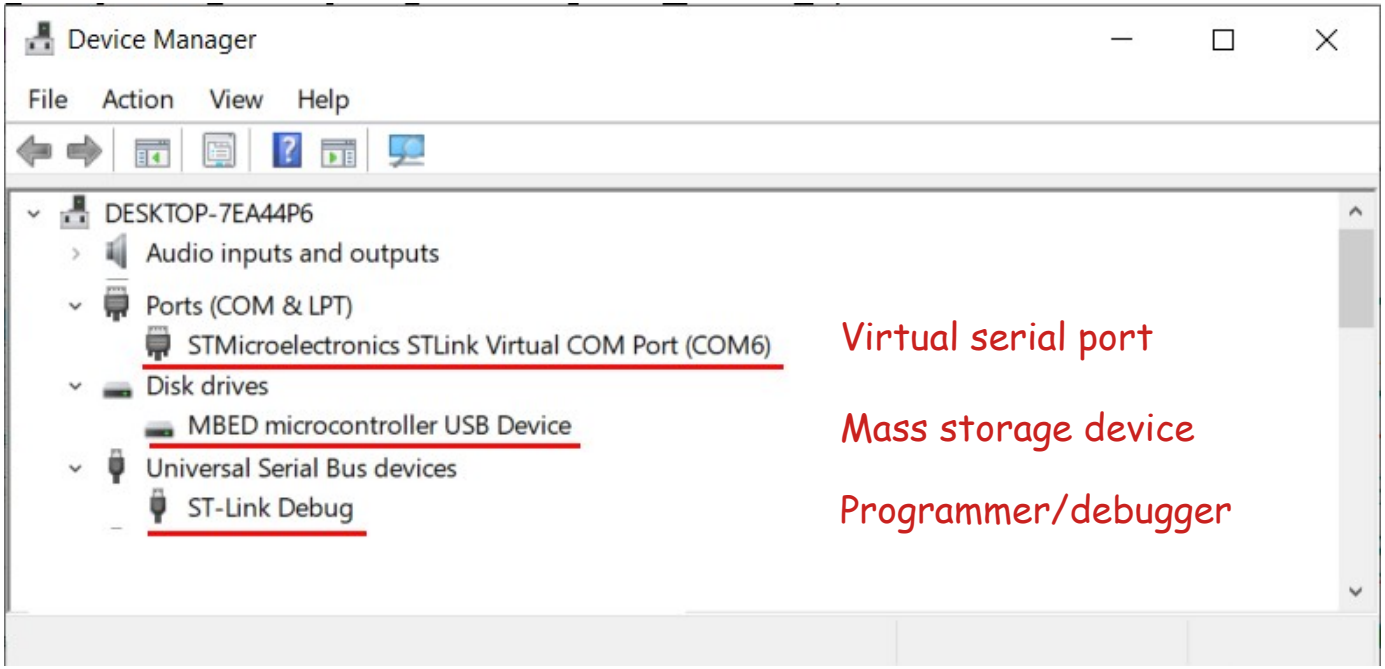
Az STM32F446RET6 mikrovezérlő

- 32-bit ARM Cortex-M4F CPU
- Flash: 512KB / RAM: 128KB
- Órajel: max. 180 MHz
- Tápfesz.: 3,3 V (1.8 V – 3.6 V)
- Digitális I/O: 50
- Analóg bemenet: 16
- 3db 12 bites ADC
- 2 db 12 bites DAC
- USB FS/HS, 4 SPI, 4 I2C,
  4 USART, 2 UART, 2 CAN
- 10+2+2 Timer
- RTC, SDIO



Meghajtó a NUCLEO-F446RE kártyához

- A **NUCLEO** kártyákon található **ST-Link v2-1** programozó USB-re csatlakozik (USB mini-B aljzat) és az SWD programozó illetve hibavadász funkció mellett USB MSD alapú programletöltést is támogat, illetve sorsos kommunikációt is biztosít a PC és a target MCU között (kompozit USB eszközként látszik)
- A kártya USB meghajtó programját az STM32Cube Programmer szoftverrel célszerű telepíteni



https://os.mbed.com

2. Itt lépj be!

Mbed

Rapid IoT device development

Mbed gives you a free open source IoT operating system with connectivity, security, storage, device management and machine learning. Build your next product with free development tools, thousands of code examples and support for hundreds of microcontroller development boards.

Sign up for free ← 1. Itt regisztráld magadat!

- Mi az MBED? Az **MBED** gyorsabbá teszi az eszközfejlesztést, sokféle hardvert támogat (jelenleg 174 kártya támogatott)

MbedOS – C++ API, perifériakönyvtár, RTOS

Operating system

Fejlesztői környezet

Compiler & IDE

Mintapéldák

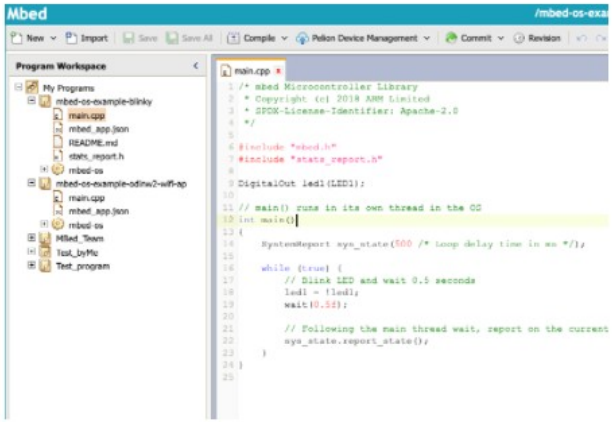
```
int main()
{
    while (true) {
        led = !led;
        thread_sleep_for
        (BLINKING_RATE_MS);
    }
}
```

MBED fejlesztői környezet(ek)

- Tavaly a felhőalapú Mbed Online Compiler -t használtuk, ami nem igényel telepítést és az **mbed 2** ill. **mbed 5** projekteket támogatta
- Az **Mbed OS 6** kiadáshoz tavaly csak az offline **Mbed Studio** létezett, ami a hardveres nyomkövetést is támogatja, de rendkívül erőforrás-igényes, ráadásul az **mbed 2** projekteket nem támogatta

Mbed Online Compiler

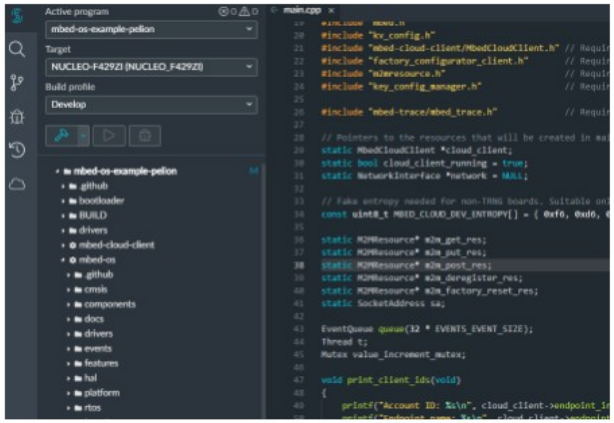
The easiest way to get started.



Jump into application development with Mbed OS directly without installing anything. Create an Mbed account to get started.

Mbed Studio

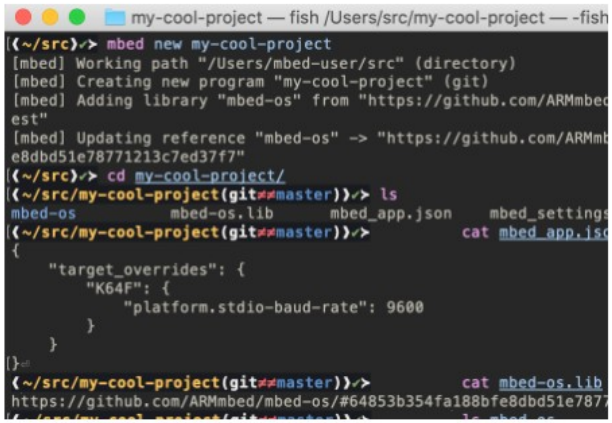
The desktop IDE for Mbed OS.



Mbed Studio is an IDE for application and library development. A single environment with everything you need to create, compile and debug your Mbed programs.

Mbed CLI

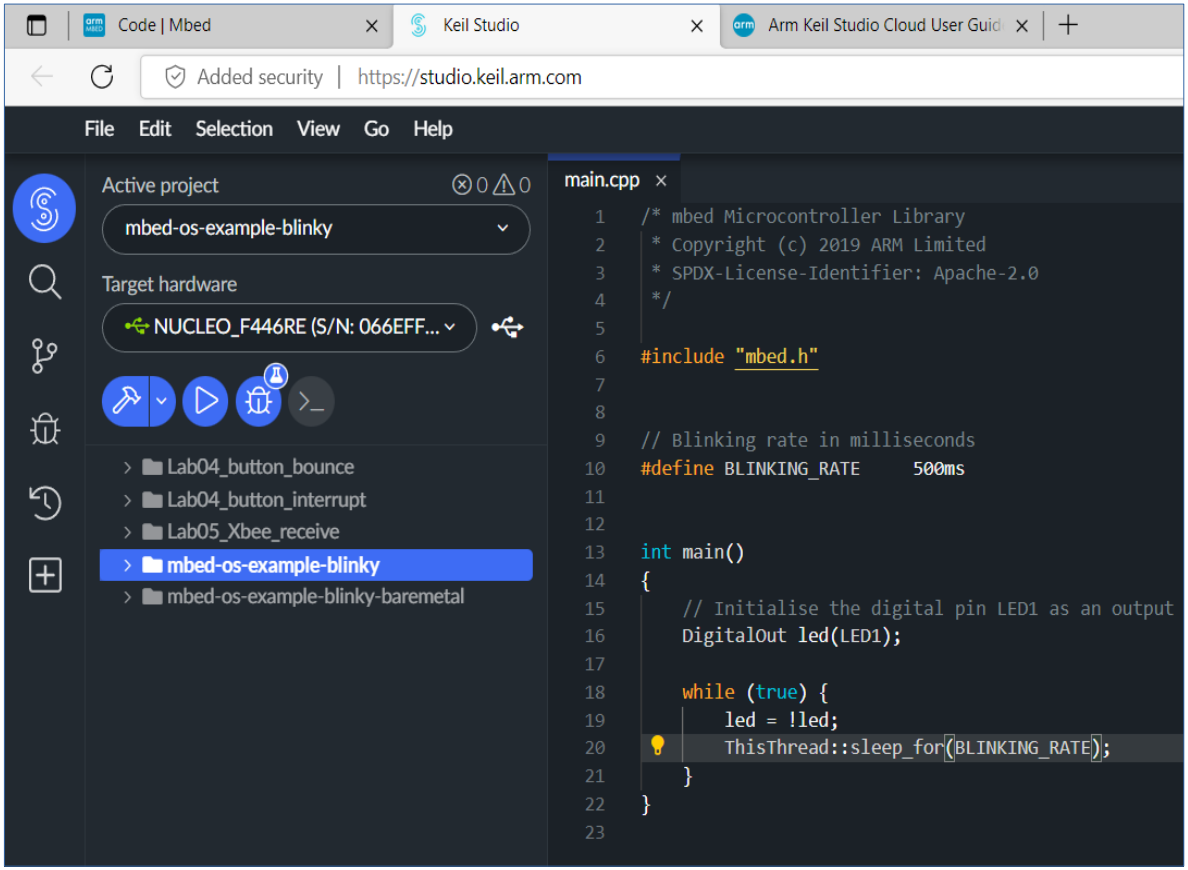
The command line tool for Mbed OS.



Integrate Mbed functionality into your preferred editor, or enhance your automation setup by using Mbed CLI, our command line interface.

Az ARM Keil Studio fejlesztői környezet

- Az **Arm Keil Studio** egy új, jelenleg még fejlesztés alatt álló, ingyenesen használható, böngésző alapú integrált fejlesztői környezet (IDE) beágyazott, IoT és Cortex-M eszközökhöz
- Kétféle projektet kezel: CMSIS és Mbed
- A **Keil Studio** az **Mbed Online Compiler** utódja, és lehetővé teszi az **Mbed OS 5** és **6** projektek fejlesztését a támogatott (Mbed Enabled) kártyákon
- A **Keil Studio** korlátozott támogatást nyújt az **Mbed 2**-höz is



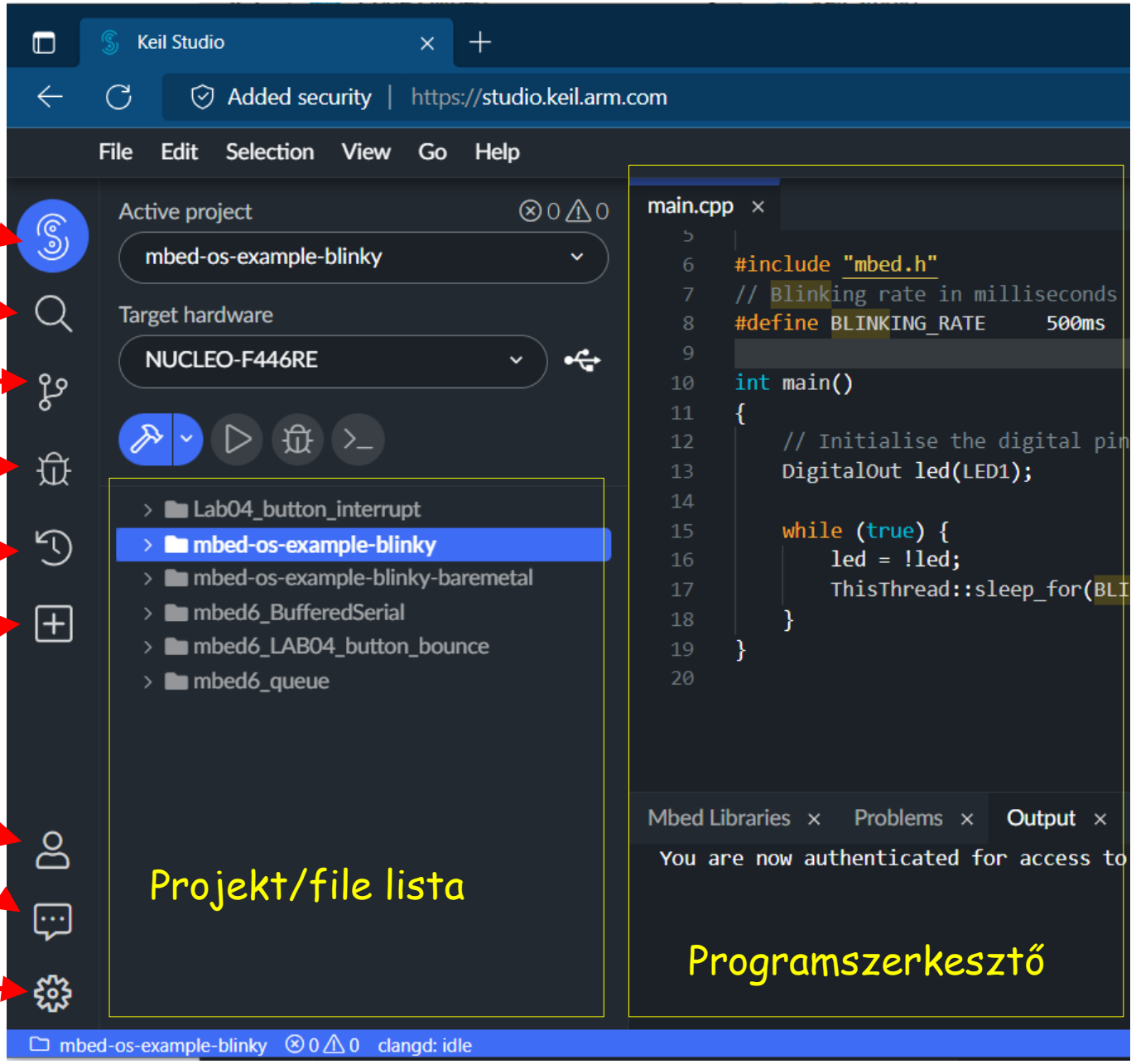
Mi kell a Keil Studio használatához?

- A **Keil Studio** használatához **Arm** vagy **Mbed** ingyenes belépőt kell regisztrálni (**Mbed** fejlesztéshez az utóbbi a célszerű)
- Ha a verziókezelést használni, vagy a projektünket közzétenni akarjuk, akkor a **GitHub**-on is regisztráljuk magunkat
- A **Keil Studio** kezelői felülete egy böngészőben jeleníthető meg, a **https://studio.keil.arm.com/** címre belépve
- **Megjegyzés: a hardveres nyomkövetés csak Microsoft Edge, vagy Google Chrome böngészőben fog működni!**
- A mikrovezérlő kártya felismeréséhez telepíteni kell a megfelelő meghajtószoftvert: például a **NUCLEO-F446RE** kártya esetén az **ST-Link v2-1** programozót kezelő szoftverre lesz szükség. Ez külön is beszerezhető, de célszerűbb az STM32Cube Programmer programmal együtt telepíteni, mert arra is szükségünk lehet
- A beépített soros terminál az **ST-Link v2-1**-gyel nem működik, külön terminálablakot (Putty, TeraTerm, stb.) kell használnunk

A kezelői felület

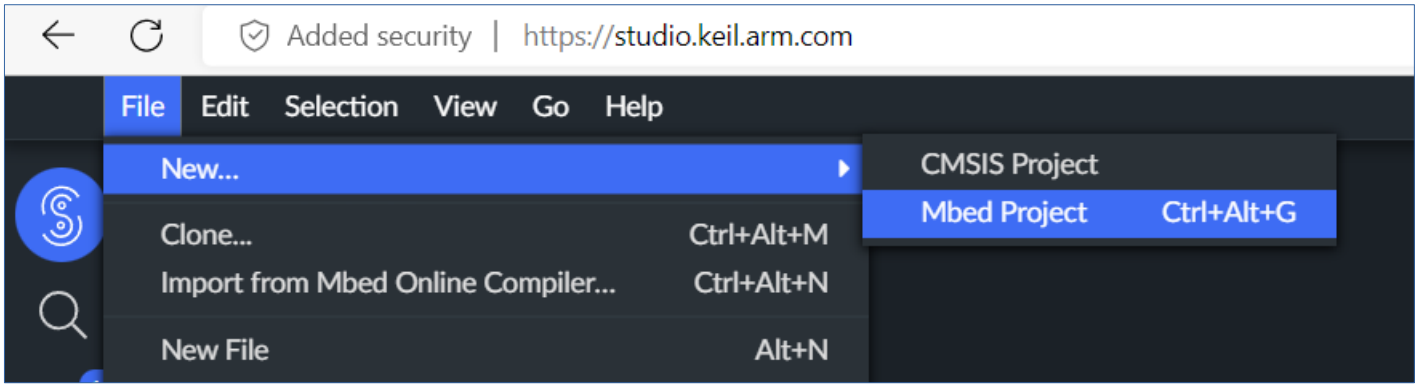
- Projekt nézet
- Keresés
- Verziókezelés (GIT)
- Nyomkövetés nézet
- Előzmények
- Bővítmények

- Felhasználó
- Visszajelzés és help
- Beállítások

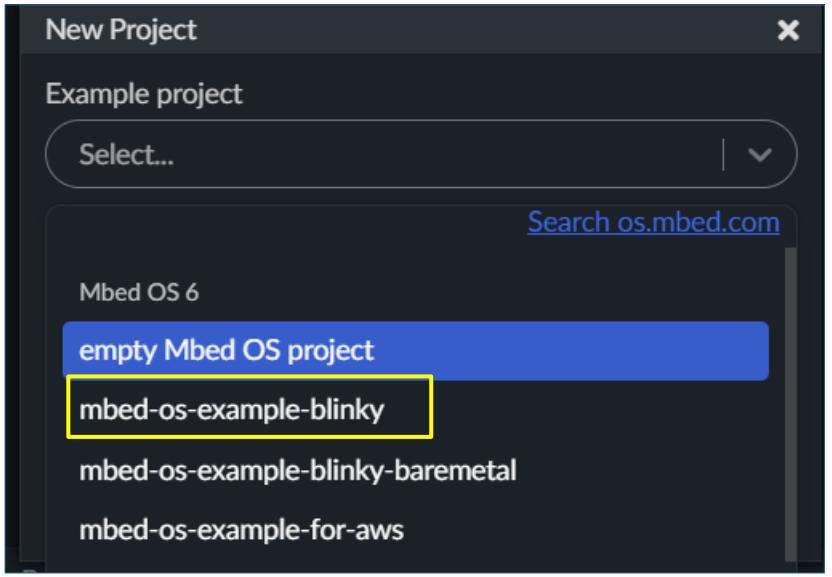


Új projekt létrehozásának lépései

- A **Keil Studio** menüjében: File⟶New⟶Mbed Project

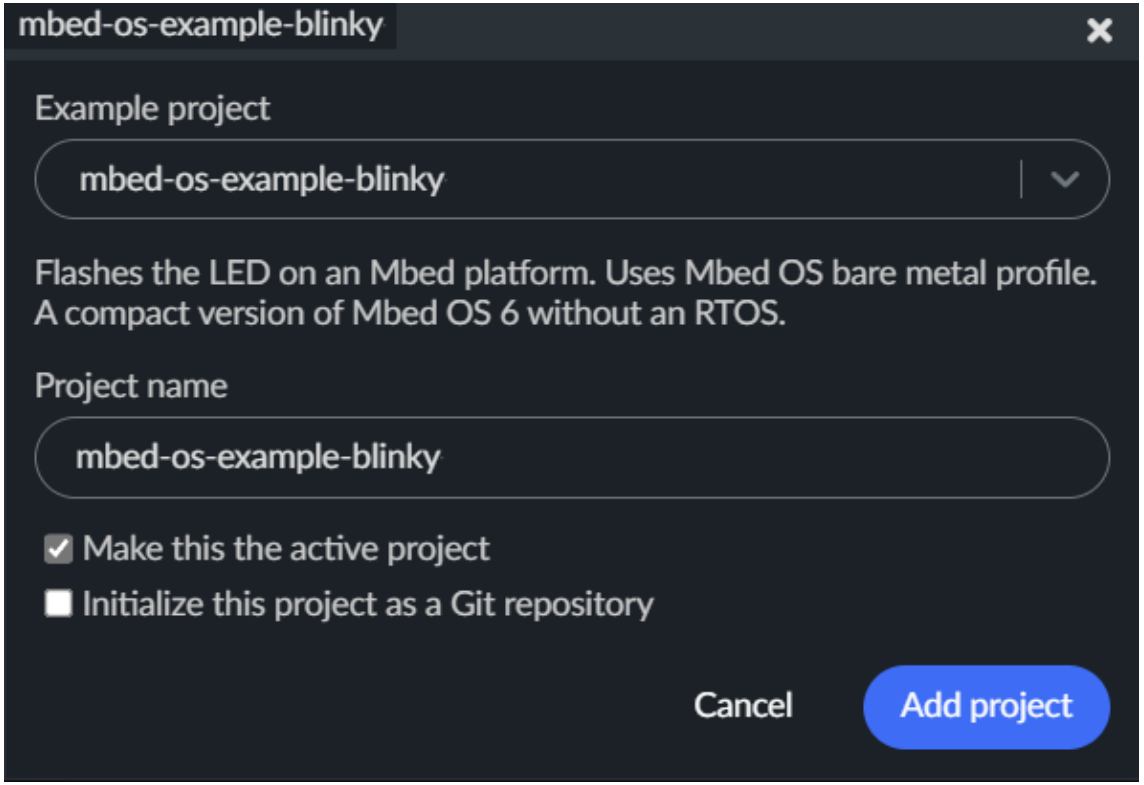


- Válasszuk ki azt a mintaprojektet (vagy „üres” projektet), amit majd átszabunk az új projektünkhöz
- Többnyire majd az „üres” projektből fogunk kiindulni, de most a ledvillogtató mintapéldát (mbed-os-example-blinky) válasszuk ki!



Új projekt létrehozásának lépései

- A **Keil Studio** menüjében: **File→New→Mbed Project**
- Az új projekt nevét átírhatjuk, de most változatlanul hagytuk
- Állítsuk be, hogy ez legyen az új aktív projekt
- A **GitHub** verziókövetés elől vegyük ki a pipát (utólag is publikálni tudjuk a projektet, ha szükségünk lesz rá)



mbed-os-example-blinky/main.cpp

- Nyissuk meg a projektből a **main.cpp** állományt!
- Az **Mbed OS** eleve többszálú (RTOS-t futtat), így blokkoló *wait()* várakozás helyett a futó programszálat *„altatjuk”* adott ideig
- Az időegységeket mértékegységgel együtt adhatjuk meg

```
#include "mbed.h"
// Blinking rate in milliseconds
#define BLINKING_RATE     500ms

int main()
{
    // Initialise the digital pin LED1 as an output
    DigitalOut led(LED1);  // LED1 predefined alias for PA_5

    while (true) {
        led = !led;
        ThisThread::sleep_for(BLINKING_RATE);
    }
}
```

DigitalOut

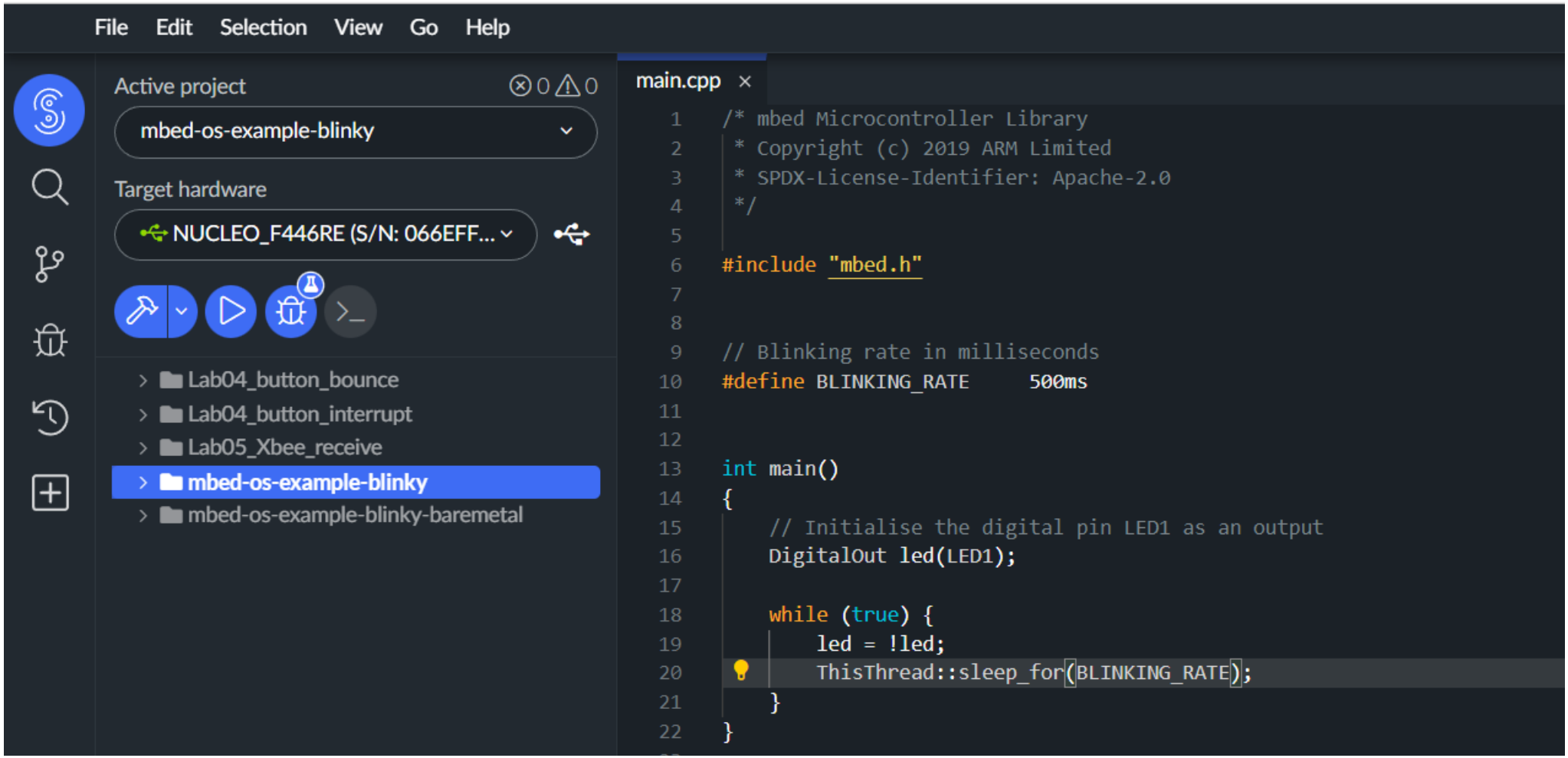
- A digitális be- és kimenetek kezelésére az mbed API a **DigitalIn**, **DigitalOut** és **DigitalInOut** komponenseket nyújtja
- A **DigitalOut** C++ objektumosztály általános célú *digitális kimenetek* konfigurálására és kezelésére szolgál
- A konstruktor hívásakor opcionálisan a kimenet kezdeti állapotát (0 vagy 1) is megadhatjuk: `DigitalOut név(pin,állapot);`

Függvény	Használat
DigitalOut név(pin)	Létrehoz egy "név" nevű **DigitalOut** objektumot és a pin paraméterrel megadott kimenethez rendeli
write(data)	A kimenet beállítása 0 vagy 1 állapotba
read()	Az int típusú visszatérési érték a kimenet állapotát adja meg (0 vagy 1)
operator =	Rövidített alak **write(data)** helyett `pl. led = 1;`
operator int()	Rövidített alak **read()** helyett `pl. ledstate = led;`

- A `write()` és `read()` tagfüggvények segítségével írhatunk a kimenetre vagy visszaolvashatjuk az állapotát. Pl. `led.write(1)` vagy `led.read()`, de használhatjuk a rövidebb formát is: `led = 1; ledstate = led;`

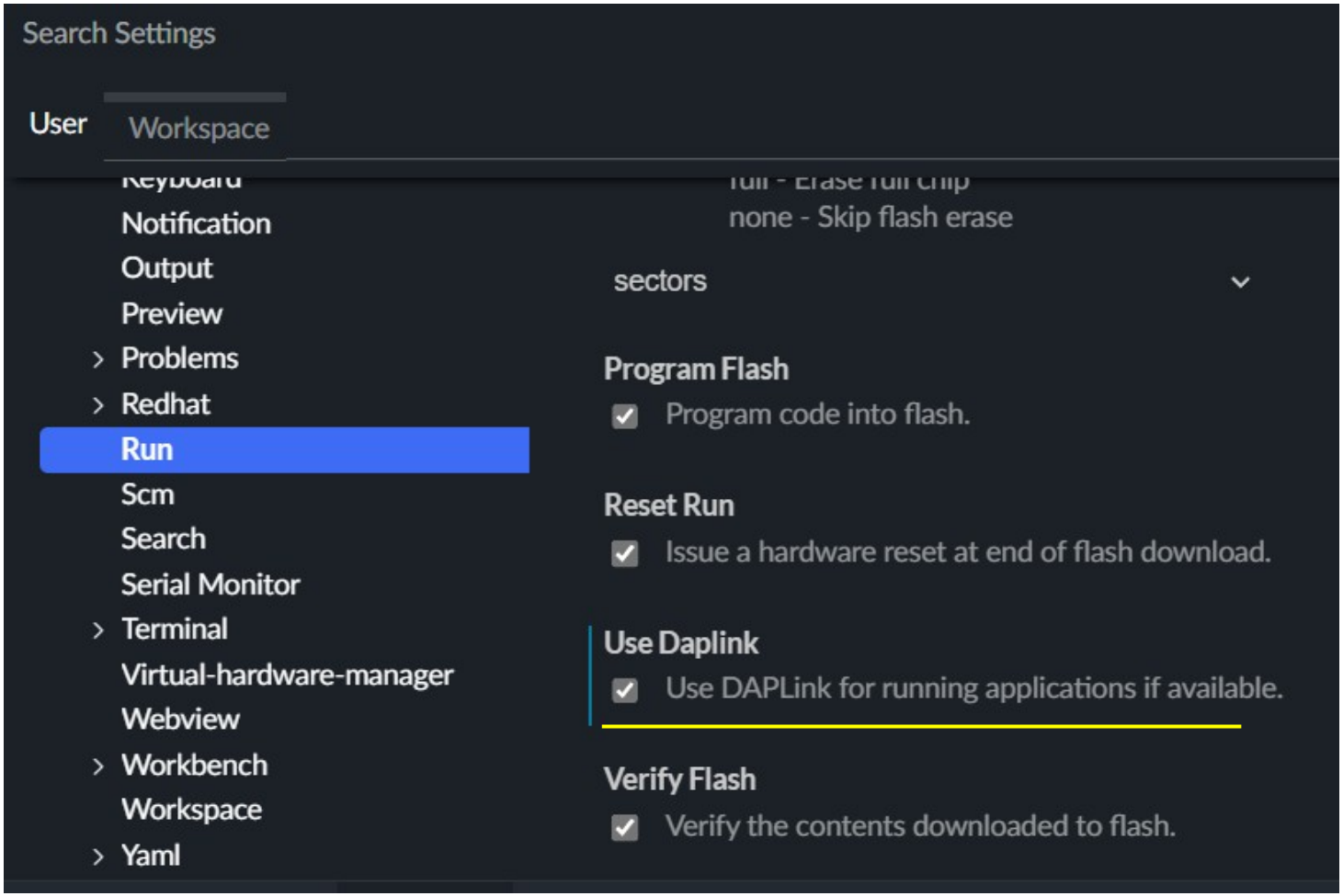
Fordítás, futtatás, nyomkövetés

- A kártya automatikus felismerésekor az USB szimbólum zöldre vált
- A fordítás, programletöltés/futtatás és a nyomkövetés az előzőleg aktívnak választott projekten történik



Egy fontos beállítás

- A **Settings** (fogaskerék ikon) → **Open preferences** → **Run** menü megnyitása után aktiváljuk (tegyünk pipát) az **Use Daplink** opciót (ezt csak egyszer kell beállítani)



Mbed OS baremetal profil

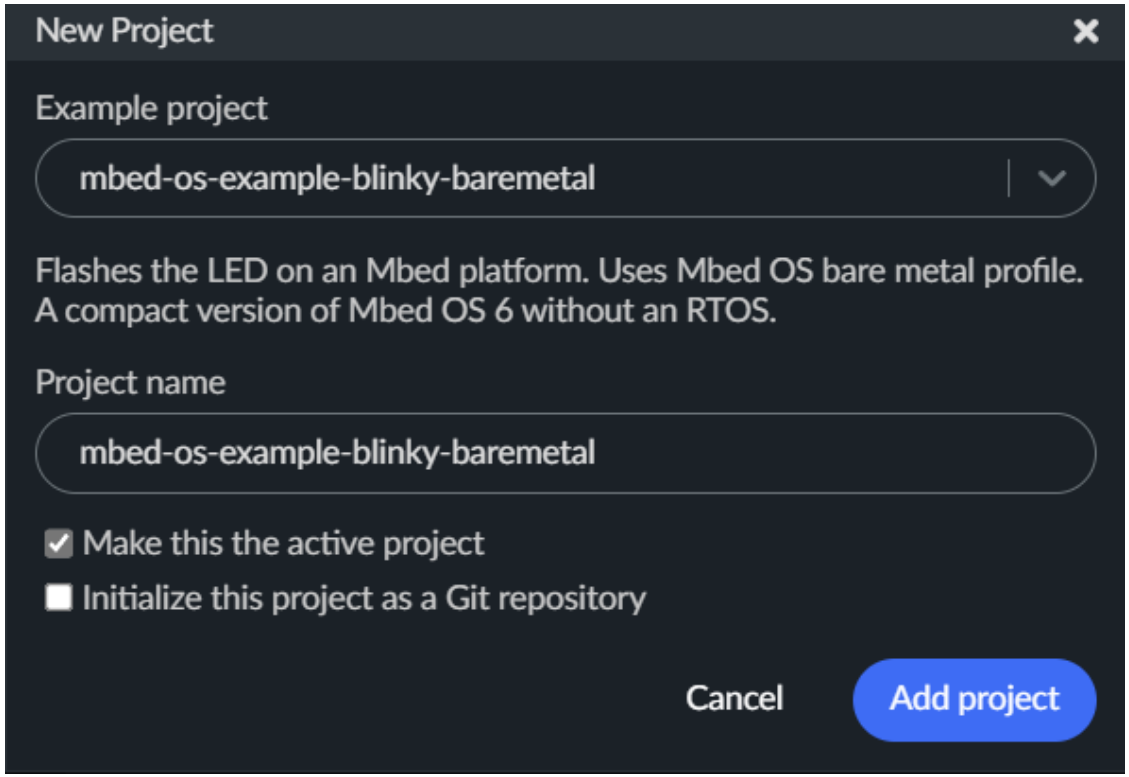
- Az **Mbed OS** projektek fordításakor láthatjuk, hogy sok fölösleges dolgot is lefordul, de ennél nagyobb probléma az, hogy az alapértelmezett RTOS fölösleges erőforrásokat foglal le a céleszközön
- A **baremetal profil** egy könnyített, RTOS nélküli rendszer, amely kis erőforrású célrendszerekhez, egyszerűbb projektekhez takarékos megoldást nyújt (olyan, mint az RTOS nélküli **mbed 2**)
- Az **RTOS** ütemezője helyett minden tevékenység lekérdezéses (polling) vagy megszakításvezérelt. Ez leegyszerűsíti az alkalmazáskódot, és lehetővé teszi olyan API-k használatát, amelyek nem szálbiztosak (non thread safe).
- Használhatja a C szabványkönyvtárak **microlib** és a **newlib-nano** méretre optimalizált verzióit, amelyek sokkal kisebbek, mint a teljes profilhoz szükséges szálbiztos könyvtárak.

Mbed OS baremetal profil

Összetevő	Támogatás a baremetal profilban
Drivers	✔
Events	✔ (manuálisan engedélyezhető) ← erre majd nézünk egy példát
HAL	✔ (manuálisan engedélyezhető)
Platform	✔
RTOS API	Ebből Semaphore, Mutex, EventFlags és ThisThread támogatott
Storage	✔ (manuálisan engedélyezhető)
WiFi	✗
LoRa	✔ (manuálisan engedélyezhető)
LwIP	✗
BLE	✔ (manuálisan engedélyezhető)
NFC	✔ (manuálisan engedélyezhető)
Mbed Crypto	✔ (manuálisan engedélyezhető)
Mbed TLS	✔ (manuálisan engedélyezhető)

mbed-os-example-blinky-baremetal

- A **Keil Studio** menüjében: **File⟶New⟶Mbed Project**
- Válasszuk ki az **mbed-os-example-blinky-baremetal** mintapéldát és hagyjuk meg ugyanezzel a névvel az új projektet
- Állítsuk be, hogy ez legyen az új aktív projekt
- A **GitHub** verziókövetés elől vegyük ki a pipát



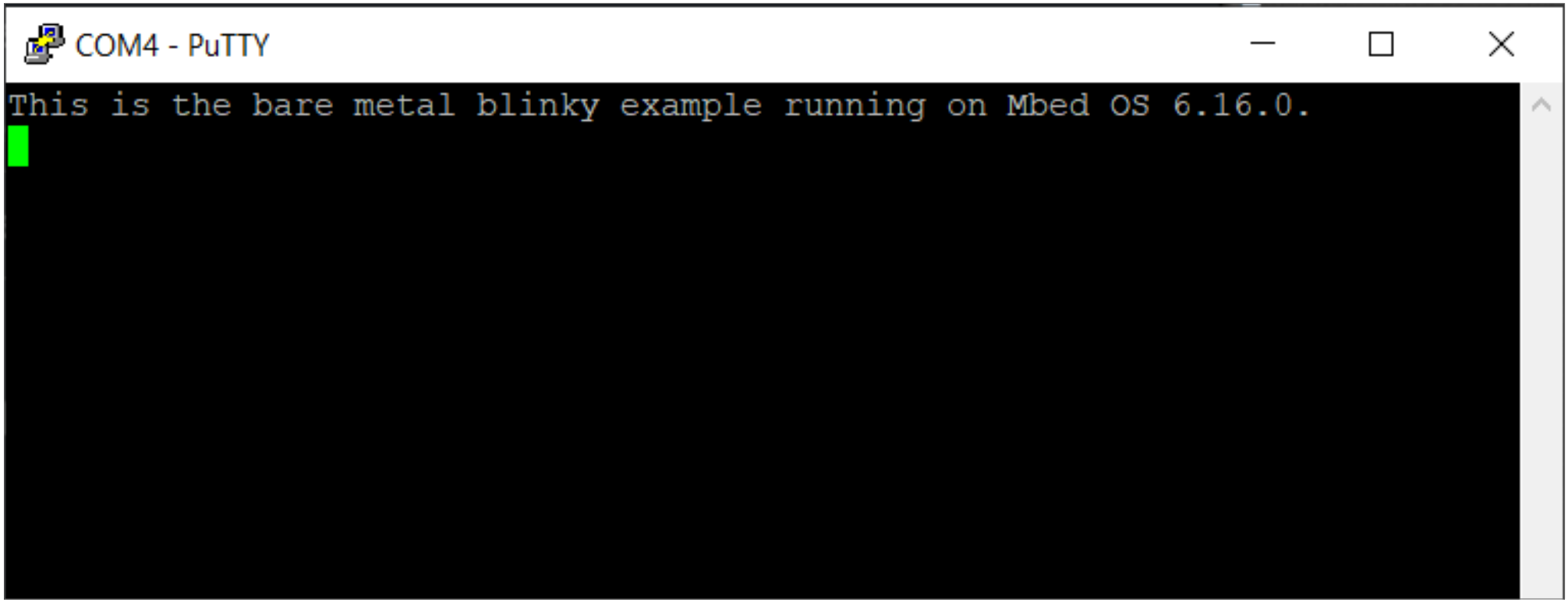
mbed-os-example-blinky-baremetal/main.cpp

- Nyissuk meg a projektből a **main.cpp** állományt!
- A **baremetal profil** is támogatja a **ThisThread** osztályt
- Az időegységet itt csak mértékegység nélkül, ms-ban adhatjuk meg

```
#include "mbed.h"

#define WAIT_TIME_MS 500
DigitalOut led1(LED1);

int main() {
    printf("This is the bare metal blinky example running on
    Mbed OS %d.%d.%d.\n", MBED_MAJOR_VERSION, MBED_MINOR_VERSION,
    MBED_PATCH_VERSION);

    while (true) {
        led1 = !led1;
        thread_sleep_for(WAIT_TIME_MS);
    }
}
```

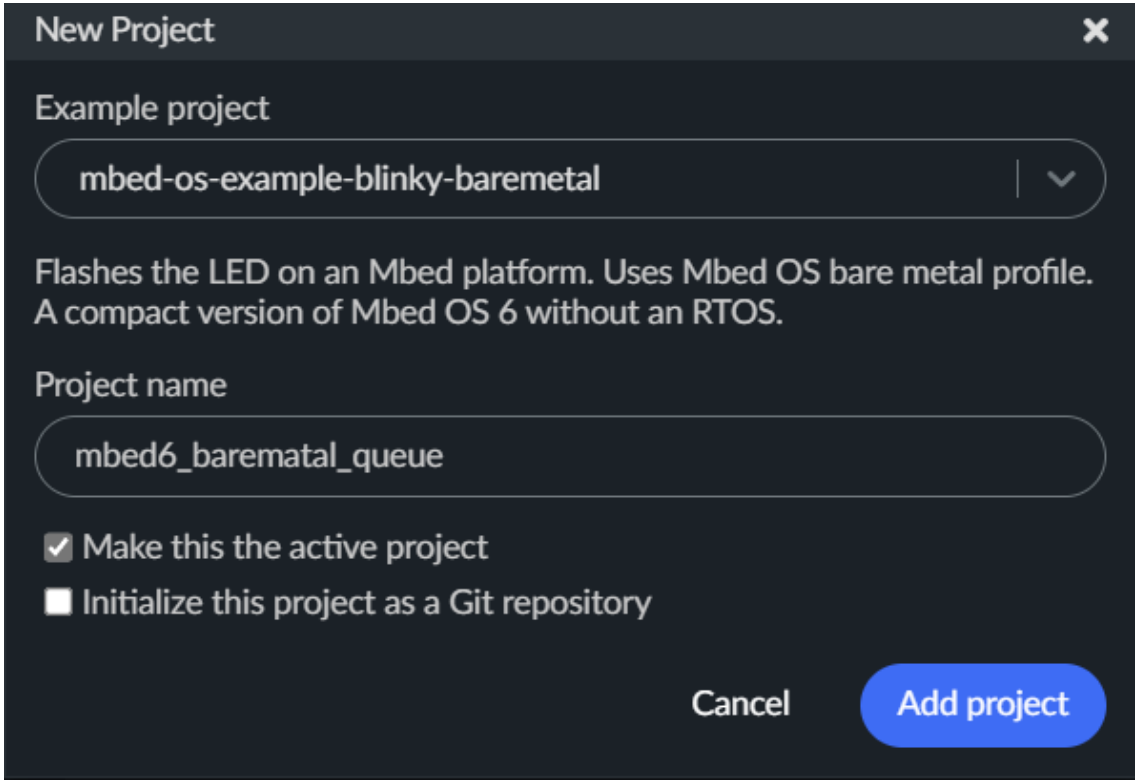
mbed-os-example-blinky-baremetal

- A **LED1** villogtatásán kívül most induláskor a soros terminálon is kiíratunk egy szöveget, a futó **Mbed OS** verziószámával együtt
- A **stdout** kimenet alapértelmezetten elérhető (printf/scanf), s az ST-Linken keresztül virtuális soros portként kapcsolódik a PC-hez
- Az alapértelmezett sebesség 9600 Baud



mbed6_baremetal_queue

- A **baremetal** profilnál maradva bemutatjuk a könyvtárak manuális engedélyezését és a stdio alapértelmezett sebességének beállítását
- A kiindulási projekt most is az **mbed-os-example-blinky-baremetal** mintapélda, ezt fogjuk módosítani, az új neve pedig legyen **mbed6_baremetal_queue** (egy eseménysort fog kezelni)



mbed6_baremetal_queue/main.cpp

- Az **EventQueue** metódusával azonnali (call), késleltetett(call_in), illetve periodikus (call_every) függvényhívást indíthatunk

```
#include "mbed.h"
#include <stdio.h>
using namespace std::chrono_literals;

int main() {
    // creates a queue with the default size
    EventQueue queue;

    // events are simple callbacks
    queue.call(printf, "called immediately\n");
    queue.call_in(2000ms, printf, "called in 2 seconds\n");
    queue.call_every(1000ms, printf, "called every 1 seconds\n");

    // events are executed by the dispatch_forever method
    queue.dispatch_forever();
}
```

Az mbed_app.json konfigurációs fájl

- Az **mbed_app.json** állományban engedélyezhetjük manuálisan az API könyvtárakat (itt most az events kell)
- Ugyanitt konfigurálhatjuk a könyvtárak tulajdonságait, illetve az **stdout** alapértelmezett paramétereit

```
{
    "requires": ["bare-metal","events"],
    "target_overrides": {
      "*": {
        "target.c_lib": "small",
        "target.printf_lib": "minimal-printf",
        "platform.minimal-printf-enable-floating-point": false,
        "platform.stdio-minimal-console-only": true,
        "platform.stdio-baud-rate": 115200
      }
    }
}
```

mbed6_baremetal_queue eredménye

- Újraindítás után lefut az azonnali, majd az 1s időközönként induló hívás, majd a 2s-mal késleltetett hívás, ezután csak az 1s időközönként induló hívás ismétlődik

A soros port kezelése

- Mbed OS alatt az alapértelmezett soros port háromféle módon érhető el:
  - **stdio console** elérése **printf, scanf** függvényekkel
  - **BufferedSerial** osztály **write, read** függvényekkel
    (némi ügyeskedéssel az **stdio console** átirányítható egy **BufferedSerial** típusú soros portra, ekkor **printf, scanf** is használható)
  - **UnbufferedSerial** osztály megszakításban kezelt **read, write**
- Bővebb információ és mintapéldák:
  https://os.mbed.com/docs/mbed-os/v6.15/apis/serial-uart-apis.html

mbed6_BufferedSerial/main.cpp

- A program a soros porton érkező karaktereket visszatükrözi

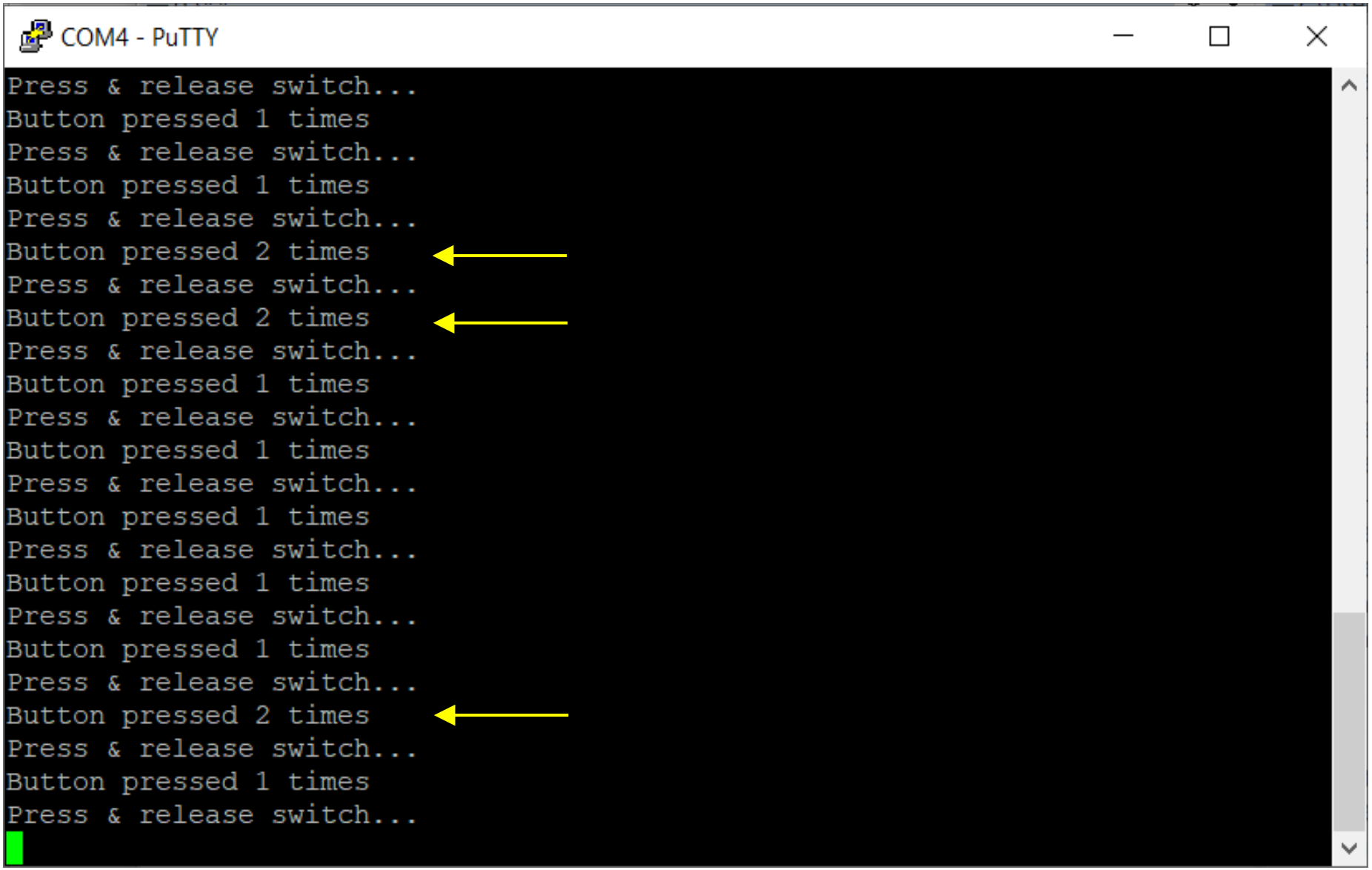
```
#include "mbed.h"

#define MAXIMUM_BUFFER_SIZE 32
static DigitalOut led(LED1);
static BufferedSerial serial_port(USBTX, USBRX);

int main(void) {
    serial_port.set_baud(9600);
    serial_port.set_format(8, BufferedSerial::None,1);
    char buf[MAXIMUM_BUFFER_SIZE] = {0};

    while (1) {
        if (uint32_t num = serial_port.read(buf, sizeof(buf))) {
            // Toggle the LED.
            led = !led;

            // Echo the input back to the terminal.
            serial_port.write(buf, num);
        }
    }
}
```

mbed6_Lab04_button_interrupt/main.cpp

```cpp
#include "mbed.h"          // Ez egy tavalyi mintapélda átirata

DigitalIn mybutton(BUTTON1,PullUp);  // Pushbutton input (PC_13)
InterruptIn button(BUTTON1);         // Pusbutton interrupt
volatile uint16_t counts;            // counter variable

void button_pressed() {
    counts++;                        // counts button presses
}

int main() {
    button.mode(PullUp);             // Enable internal pullup
    button.fall(&button_pressed);    // Attach function to falling edge
    while (true) {
        counts = 0;                  // Clear counter
        printf("Press & release switch... \r\n");
        while (mybutton);            // Wait for button press
        ThisThread::sleep_for(20ms); // Debounce delay
        while (!mybutton);           // Wait for button release
        ThisThread::sleep_for(20ms); // Debounce delay
        printf("Button pressed %d times\r\n",counts);
    }
}
```

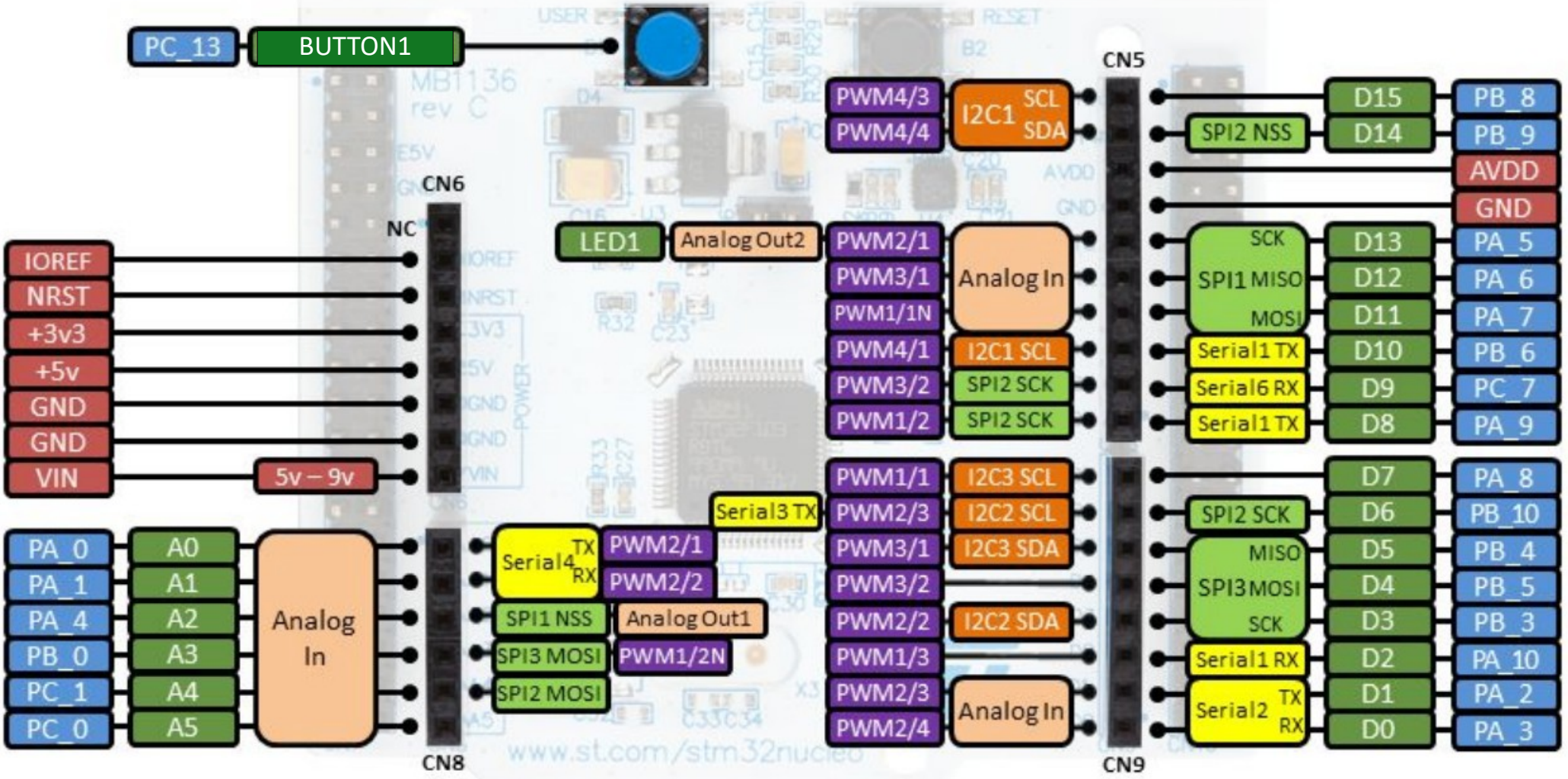
mbed6_Lab04_button_interrupt eredménye

- Az 1-től különböző értékek a kontaktus pergését jelzik



Arduino kompatibilis kivezetések

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



Morpho kivezetések

Nucleo F446RE
Morpho Headers