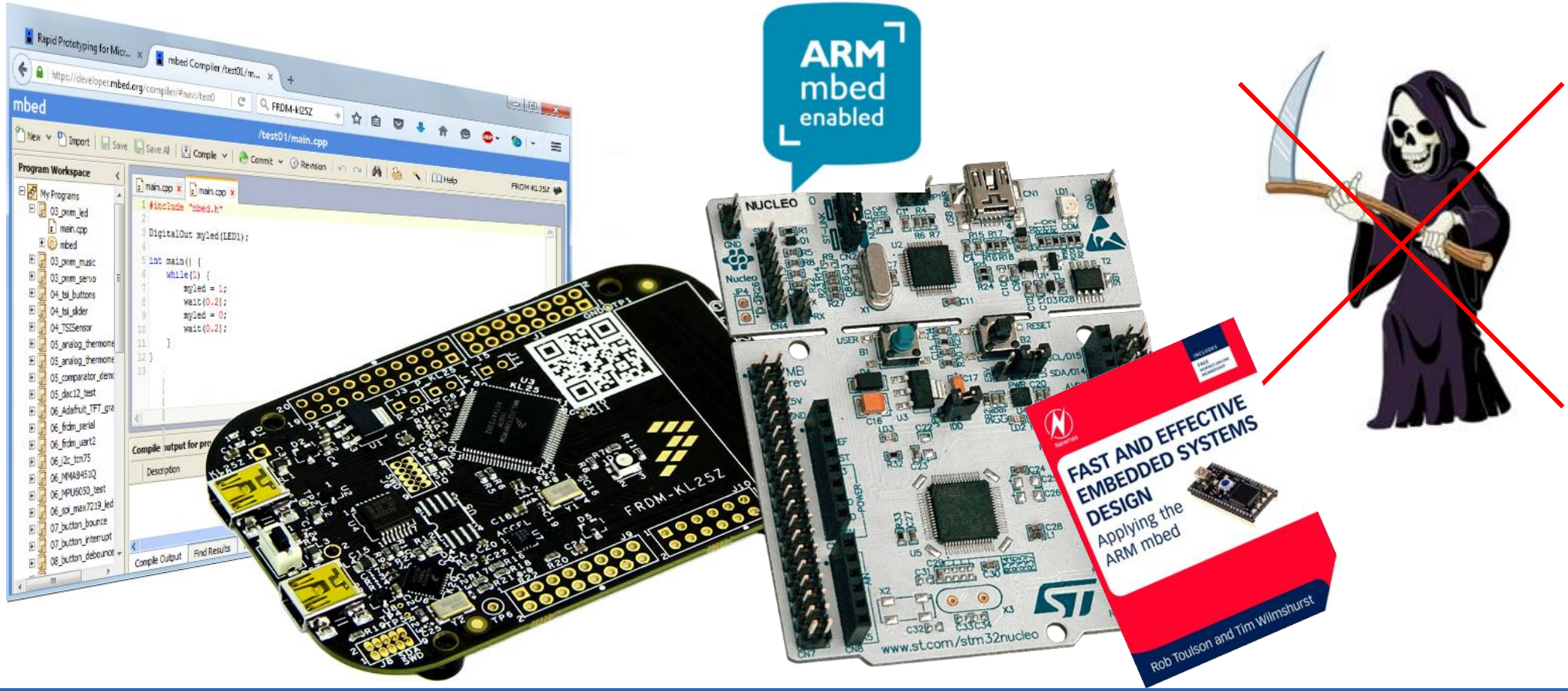
MBED – van-e élet a halál után?



Felhasznált és ajánlott irodalom

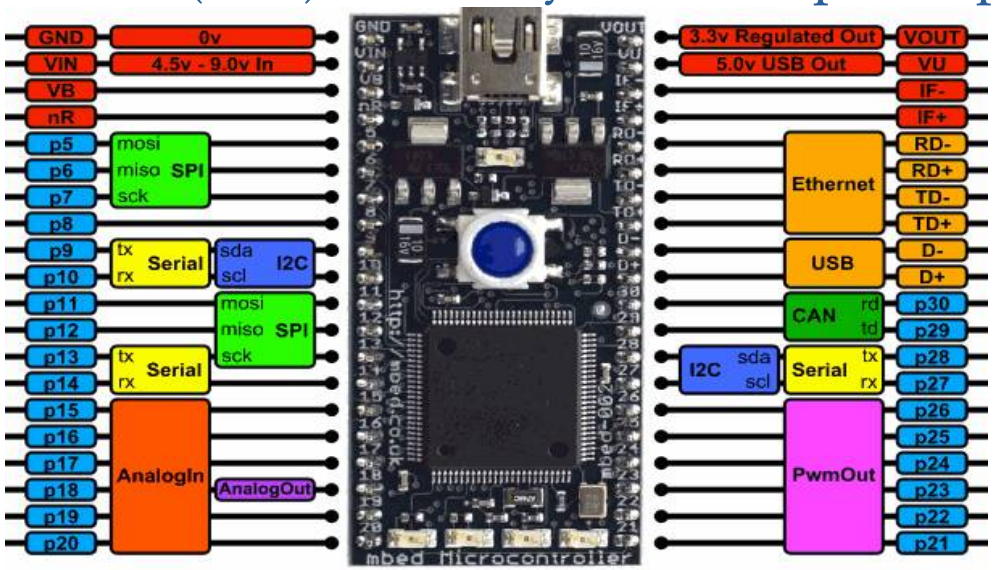
- Rob Toulson and Tim Wilmhurst: Fast and Effective Embedded Systems Design: Applying the ARM mbed
- Perry Xiao: Designing Embedded Systems and the Internet of Things (IoT) with the ARM mbed
- Cserny István: A FRDM-KL25Z kártya programozása mbed környezetben
- **ARM mbed honlap: https://os.mbed.com/**
- ARM mbed 2 Handbook (elavult): https://os.mbed.com/handbook/Homepage
- ARM mbed 2 Cookbook (elavult): https://os.mbed.com/cookbook/Homepage
- ARM mbed-os forráskód: https://github.com/ARMmbed/mbed-os
- Toyomasa Watarai: Mbed OS 2のサポートがいつ終わっても良いようにローカル環境を構築する方法【改訂版】

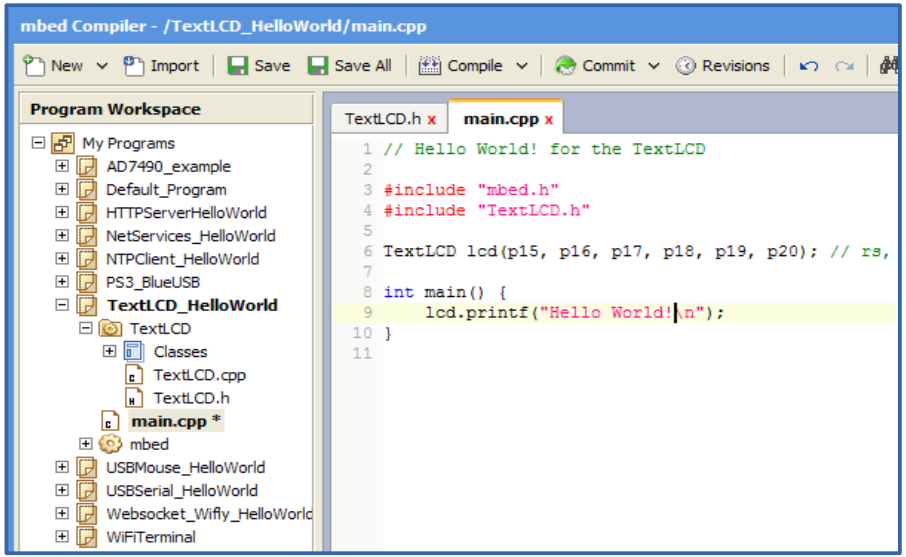
Bevezetés

- Az **Mbed** (későbbi verziója az **Mbed OS**) széles körben használt, mikrovezérlő-fejlesztési platform volt, amellyel a beágyazott rendszerek fejlesztése könnyen és hatékonyan végezhető (gyors prototípus készítés, „proof of concept”)
- Az ARM az **Mbed** hivatalos támogatását fokozatosan megszünteti, emiatt szolgáltatások szűntek meg, a korábban használt fejlesztőeszközök és munkamódszerek használhatatlanok vagy elérhetetlenek, a fejlesztői platform, gyakorlatilag „szétesett”
- **Előadásunk célja,** hogy áttekinthetően bemutassuk, mi működik még, és hogyan lehet régi **Mbed2** projekteket újra életre kelteni.
- Az előadás középpontban **a gyakorlati túlélési stratégiák állnak,** különösen a **gcc4mbed** és a saját buildrendszerek alkalmazása.
- **A bemutatott kísérleteket** és teszteket **Freescale FRDM-KL25Z** és **STM NUCLEO-F446RE** fejlesztőkártyákkal végeztük

MBED – ahogy elkezdődött...

- Az első kiadás: **2009. szeptember 21.**
- A támogatott hardver: **NXP LPC1768** alapú kártya (Cortex-M3 96 MHz,512k Flash/32k RAM, Ethernet, USB)
- USB Mass storage alapú programletöltés
- Online Compiler (cloud computing), magas szintű API (C++), hatékony HW+SW prototípus készítésére





Mi (volt) az Mbed

- **Operációs réteg**
  - Egységes futtatási környezet Cortex-M mikrovezérlőkhöz
  - **RTOS**-szolgáltatások vagy **bare-metal** runtime

- **Fordító és integrált fejlesztőkörnyezet**
  - Konzisztens build-folyamat (ARM, GCC, Clang)
  - Átlátható projektstruktúra, eszközök, profilok

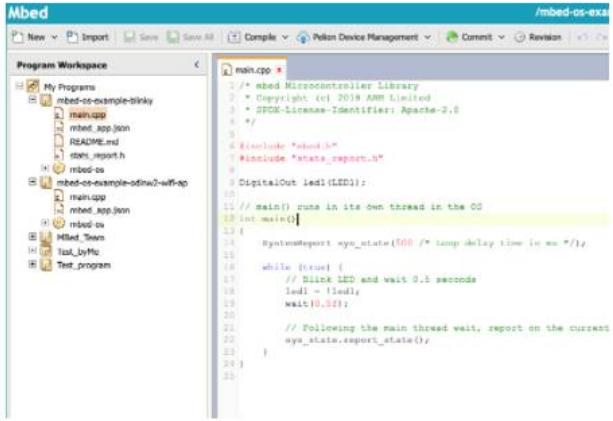
- **API-k és könyvtárak**
  - Hordozható C/C++ absztrakció GPIO, UART, SPI, I2C, PWM stb. fölött
  - Kiegészítő könyvtárak: RTOS, hálózat, fájlrendszer, USB, szenzorok, kijelzők
- **Mintaprogramok gyűjteménye**
  - Rövid, célzott példák az API használatához
  - Gyors belépési pont tanulóknak és fejlesztőknek

MBED fejlesztői környezetek (voltak)

- **Mbed Online Compiler:** 2022-ben megszűnt a szolgáltatás, lekapcsolták
- **Mbed Studio** (elvileg offline): a gyakorlatban már nem használható
- **Mbed CLI v1** (elvileg offline): a gyakorlatban már nem használható
- **Mbed CLI v2** (elvileg offline): elvileg még használható (Mbed OS 6.5-tel)
- **Keil Studio Cloud:** 2026. júliusától az Mbed támogatás megszűnik benne

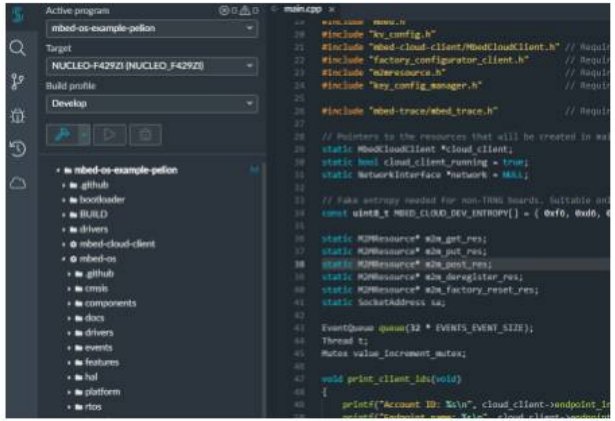
Mbed Online Compiler

The easiest way to get started.



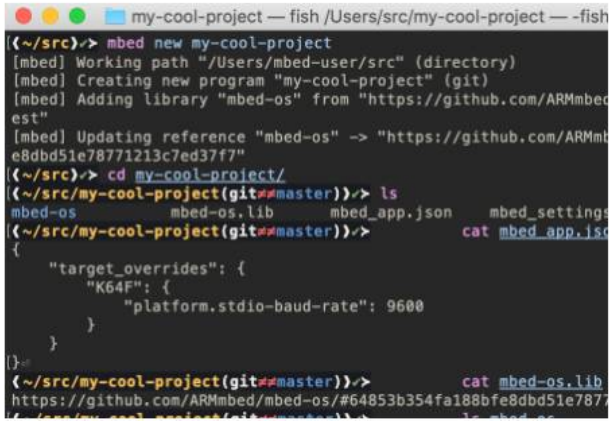
Mbed Studio

The desktop IDE for Mbed OS.



Mbed CLI

The command line tool for Mbed OS.



MBED – ma még elérhető hasznos források

- **Mbed 2**
  - **Mbed Handbook** (Classic API) – Az **Mbed2 API** hivatalos dokumentációja
  - **Mbed Cookbook** – Periféria-kezelés példák, driverek, gyakorlati minták
  - Felhasználói projektek – például: https://os.mbed.com/users/cspista/code/
  - **mbed-dev** – Mbed 2.0 r163 – forráskód snapshot (GitHub repo)

- **Mbed OS 6.x**
  - **Mbed OS – Hivatalos dokumentáció**
  - **Mbed OS GitHub repository** – a teljes forráskód
  - **Mbed CLI v2** – Cmake workflow dokumentáció, telepítés, használat

Mentsük, ami menthető!

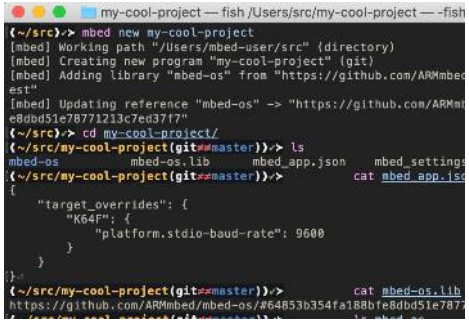
- Az **Mbed Online Compiler** felhőben tartja a projekteket. A platform jövője bizonytalan, a legfontosabb tehát, hogy a korábbi munkáinkat elmentsük
- Felhasználó projektek: **`https://os.mbed.com/users/<felhasználó>/code/`**
- Mivel tömeges export nincs, ezért egy Powershell scripttel végezzük a letöltést (**mbed_backup.ps1**), a letöltött ZIP csomagok pedig az **mbed_backup_<user>** mappába kerülnek elhelyezésre

mbed_backup.ps1

```
$User = "icserny"
$Base = "https://os.mbed.com/users/$User/code"
$Out  = "mbed_backup_$User"
mkdir $Out -ErrorAction Ignore

$page = 1; $Links = @()

while ($true) {
    $html = Invoke-WebRequest "$Base/?page=$page"
-UseBasicParsing
    $l = $html.Links | Where-Object { $_.href -match
"^/users/$User/code/.+?/$" }
    if ($l.Count -eq 0) { break }
    $Links += $l; $page++
}

foreach ($link in $Links) {
    $name = ($link.href -split "/")[-2]
    Invoke-WebRequest "https://os.mbed.com$
($link.href)archive/" `
        -OutFile "$Out/$name.zip" -UseBasicParsing
}
```

A túlélés lehetőségei

- **Mbed 2**
  - **gcc4mbed** (GCC ARM, Mbed API rev142)
  - **GCC ARM** + **saját scriptek** (Makefile/CMake/batch/shell/Python)
  - **Keil uVision** – csak korábban már exportált projektekhez
- **Mbed 6**
  - **Mbed CLI v2** (CMake + Ninja + GCC ARM)
  - **Keil Studio Cloud** – csak 2026 júliusig használható!
- **Mbed CE** (Mbed OS Community Edition)
  - **VS Code** + **CMake** + **Ninja** + **GCC ARM**
  - **mbed-cmake** (CE saját CLI)



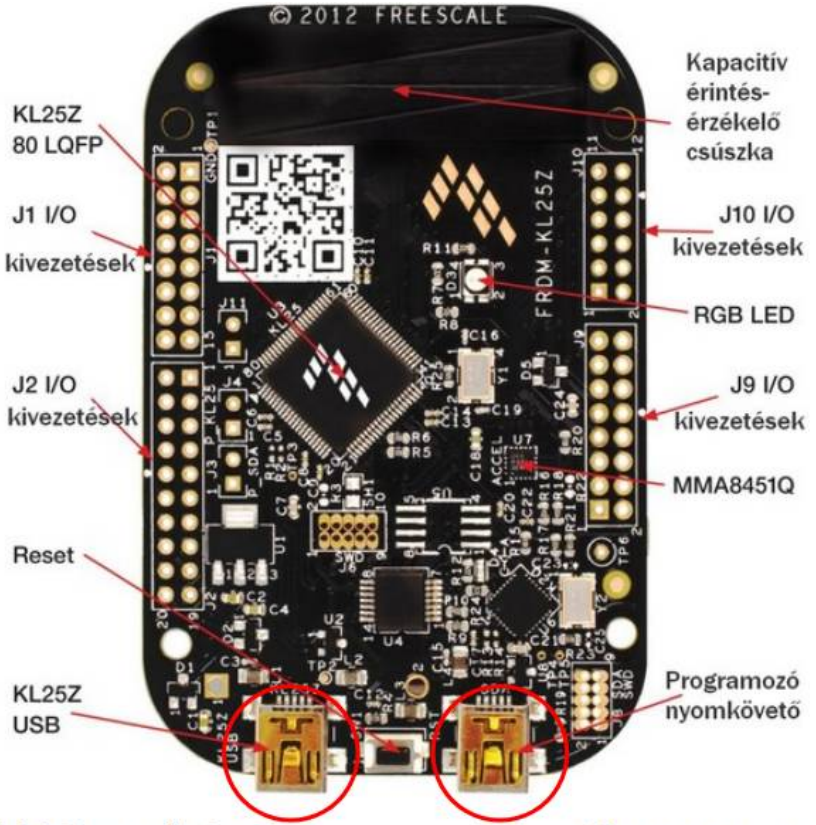
MBED CE

Freescale FRDM-KL25Z kártya

- Freescale MKL25Z128VLK4 MCU
- Nagy teljesítményű ARM Cortex-M0+
- 48MHz, 16KB RAM, 128KB FLASH
- USB (Host/Device)
- SPI (2) / I2C (2) / UART (3)
- PWM (TPM)
- ADC (16 bit) / DAC (1x12bit)
- Érintésérzékelő
- GPIO (66)
- MMA8451Q - 3-tengelyű gyorsulásmérő
- Kapacitív érintésérzékelő csúszka

A kártya műszaki adatai:

- méret: 81mm x 53mm
- 5V USB vagy 4.5-9V külső tápellátás
- Beépített USB FLASH
- "fogd és vidd" programozás



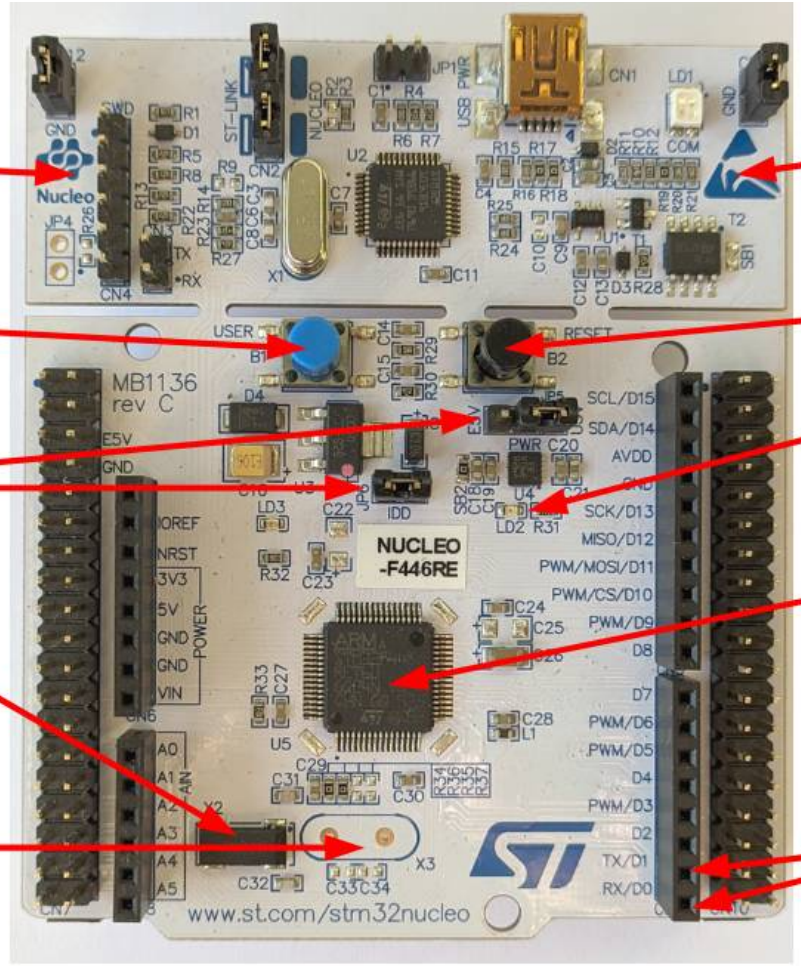
USB eszköz
Közvetlen csatlakozás a mikrovezérlőhöz

Programozó eszköz
Soros porton kommunikál a mikrovezérlővel (*Serial class*)

A FRDM-KL25Z kártya újraélesztése

- Ismert probléma, hogy a Windows 10 „elrontja” az OpenSDA bootloadert, a programletöltés nem működik, a kártya használhatatlanná válik
- **A megoldás (csak Windows 7 alatt működik stabilan):**
  - Az https://www.pemicro.com/opensda/ oldalról le kell tölteni az OpenSDA Firmware (MSD & Debug) csomagot (ehhez regisztrálni kell)
    A letöltött fájl: Pemicro_OpenSDA_Debug_MSD_Update_Apps_2023_12_12.zip
    Ebből csak a kétszeresen zip-pelt **BOOTUPDATEAPP_Pemicro_v111.SDA** kell
  - A https://os.mbed.com/handbook/Firmware-FRDM-KL25Z oldalról le kell tölteni a kártyához való mbed firmware-t (20 140 530_k20dx128_kl25z_if_opensda).
    A zip fájlból csomagoljuk ki a firware-t: **kl26z_opensda.s19**
  - Helyezzük a FRDM-KL25Z kártyát bootloader (programbetöltő) módba, majd a megjelenő BOOTLOADER meghajtóba másoljuk be a **BOOTUPDATEAPP_Pemicro_v111.SDA** és a **kl26z_opensda.s19** állományokat!
- **Lehetőleg csak Windows 7 alatt használjuk a kártyát!**

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)

A nehéz út

- A nehezebbik út, ha magunknak kell kitalálni, hogy a projektet hogyan kell lefordítani és összelinkelni az Mbed API könyvtáraival
  - Telepítsük a GCC ARM Embedded Toolchain 9-2019-q4-major (vagy korábbi) kiadását (Windows: gcc-arm-none-eabi-9-2019-q4-major-win32.exe)
  - Töltsük le az Mbed2 forrásállományait vagy az előfordított csomagot
  - Tegyük mellé egy projektmappát (benne a **main.cpp**)
  - Készítsünk a projekthez saját Makefile-t vagy batch fájlt amely konfigurálja és szervezi a fordítást
- Az alábbiakban azt mutatjuk be, hogy az Mbed2 r163 előfordított csomagját használva hogy tudjuk lefordítani a projektünket
  - Hozzuk létre a **blinky** mappát az **mbed-e95d10626187** mappa mellé!
  - Hozzuk létre a **blinky** mappában a **main.cpp** állományt!
  - Hozzuk létre, vagy másoljuk be a **build.bat** állományt, majd futtassuk!
  - A létrejött program,bin állományt másoljuk át az MBED meghajtóra

blinky/main.cpp

- Egyszerű LED villogtató program, ami a KL25Z kártya s LED-jét piros színnel villogtatja
- A **DigitalOut** kimenetté teszi a LED1 lábat, majd a végtelen ciklusban a program 200 ms-onként fel- és lekapcsolja, így a LED folyamatosan villog.

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}
```

build.bat – 2/1.

```
@echo off
setlocal ENABLEDELAYEDEXPANSION

REM ---- Paths ----
set MBED=..\mbed-e95d10626187
set TARGET=%MBED%\TARGET_KL25Z
set TOOLCHAIN=%TARGET%\TOOLCHAIN_GCC_ARM

REM ---- Toolchain ----
set CC=arm-none-eabi-gcc
set CXX=arm-none-eabi-g++
set LD=arm-none-eabi-g++
set OBJCOPY=arm-none-eabi-objcopy

REM ---- Includes ----
set INCLUDES=-I%MBED% ^
 -I%MBED%\drivers ^
 -I%MBED%\hal ^
 -I%MBED%\platform ^
 -I%TARGET% ^
 -I%TARGET%\TARGET_Freescale\TARGET_KLXX ^
 -I%TARGET%\TARGET_Freescale\TARGET_KLXX\TARGET_KL25Z ^
 -I%TARGET%\TARGET_Freescale\TARGET_KLXX\TARGET_KL25Z\device
```

Ez a batch fájl előkészíti a fordítókörnyezetet a KL25Z-hez.
Beállítja az mbed r163 könyvtár helyét, a KL25Z-hez tartozó target mappákat, majd megadja a GCC ARM fordítóprogramok nevét.
Ezután összerakja az include útvonalakat, hogy a fordító megtalálja az mbed és a KL25Z header fájljait.
A setlocal ENABLEDELAYEDEXPANSION csak azt biztosítja, hogy a változók friss értékei ciklusokban is helyesen kiértékelődjenek.

build.bat – 2/2.

```
REM ---- Flags ----
set CFLAGS=-mcpu=cortex-m0plus -mthumb -Os -ffunction-sections -fdata-sections -Wall -Wextra %INCLUDES%
set LDFLAGS=-mcpu=cortex-m0plus -mthumb -Wl,--gc-sections --specs=nosys.specs -T%TOOLCHAIN%\MKL25Z4.ld
REM ---- Collect TARGET object files ----
set OBJLIST=
for %%f in (%TOOLCHAIN%\*.o) do (
    set OBJLIST=!OBJLIST! %%f
)
REM ---- Build ----
%CXX% %CFLAGS% -c main.cpp -o main.o
if errorlevel 1 goto :error

%LD% %LDFLAGS% main.o !OBJLIST! %TOOLCHAIN%\libmbed.a -o program.elf
if errorlevel 1 goto :error

%OBJCOPY% -O binary program.elf program.bin
if errorlevel 1 goto :error
goto :end
:error
echo.
echo Build FAILED.
:end
endlocal
```

CFLAGS a fordító opciói: Cortex-M0+ CPU, Thumb mód, optimalizálás, fölösleges szekciók eldobása, valamint az include könyvtárak hozzáadása.
LDFLAGS a linker beállításai: ugyanaz a CPU, fölösleges kódrészek kiszedése, az alapértelmezett nosys környezet, és a KL25Z-hez tartozó linker script (MKL25Z4.ld) megadása.

A **for** ciklus összegyűjti a KL25Z-hez tartozó előre lefordított objektumfájlokat (*.o).
Ezután a script lefordítja a main.cpp-t, majd a linkerrel összefűzi a saját objektumot, a target objektumokat és a **libmbed.a** könyvtárat.
Végül az **objcopy** elkészíti a **program.bin** fájlt.

A cselekmény bonyolódik...

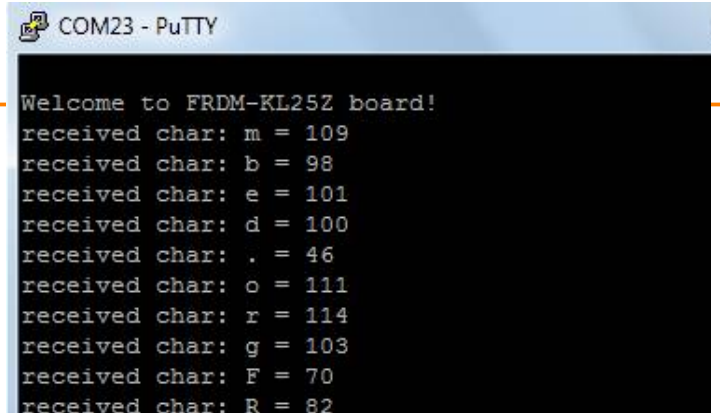
- A **blinky** projekt még gond nélkül működött, mert csak az alap mbed-modulokra volt szükség.
- A Serial/I2C/SPI stb. perifériák használata azonban már nem automatikus: az **mbed2 csak akkor építi be** a periféria kódját, ha definiáljuk hozzá a szükséges makrót
- A makrókat többféle módon is meg lehet adni, mi egy **mbed_config.h** fájlban helyezzük el a projekt mappában
- A **build.bat** fáljban az **Includes** szekcióban szükség lesz egy `-I. ^` sorra is
- Az **mbed-e95d10626187** mappában pedig az mbed.h fejlécállomány elejére írjuk be: #include "mbed_config.h"
- A következő példához elegendő csupán a DEVICE_SERIAL makrót 1-be állítani

```
#ifndef MBED_CONFIG_H
#define MBED_CONFIG_H

#define DEVICE_SERIAL        1
#define DEVICE_INTERRUPTIN   1
#define DEVICE_SLEEP         1
#define DEVICE_ANALOGIN      1
#define DEVICE_PWMOUT        1
#define DEVICE_I2C           1
#define DEVICE_SPI           1

#endif
```

serial/main.cpp

- A program egy Serial objektumot inicializál az OpenSDA-n keresztüli UART0-hoz, induláskor kiír egy üdvözlő szöveget a PC felé
- A fő ciklusban folyamatosan vár egy bejövő karakterre, és minden egyes fogadott bájtot visszaküld, kiírva annak karakter- és decimális kódját is

```
#include "mbed.h"
#include "Serial.h"

Serial pc(USBTX,USBRX);     //UART0 via OpenSDA

int main()
{
    pc.printf("\r\nWelcome to FRDM-KL25Z board!\r\n");
    while(1) {
        char c = pc.getc(); //Read one character
        pc.printf("received char: %c = %d\r\n",c,c);
    }
}
```



gcc4mbed – a „királyi út”

adamgreen/
gcc4mbed

01000001
01000111

Project to allow GCC compilation of code using mbed SDK libraries.

15 Contributors | 0 Issues | 172 Stars | 67 Forks

github.com/adamgreen/gcc4mbed

Files

master

Go to file

- build
- external
  - mbed-os
  - osx64
  - win32
  - .gitignore
- mbedUpdater
- mri
- notes
- samples
- src
- .gitignore
- README.creole
- linux_install
- mac_install
- win_install.cmd

Archived - June 10th, 2021

This project is no longer under active development.

Current Version

GCC Version: GCC ARM Embedded 6-2017-q1-update
mbed SDK Version: Revision 142

Quick Start

These are the quick steps to get gcc4mbed installed on your computer:

- Download the gcc4mbed compressed archive from https://github.com/adamgreen/gcc4mbed/zipball/master
- Decompress and extract the contents of this archive to the location where you want gcc4mbed and the GNU Tools for ARM Embedded Processors to be installed.
- Run the install script appropriate for your platform:
  - Windows: win_install.cmd
  - macOS: mac_install
  - Linux: linux_install
- You can then run the BuildShell script which will be created during the install to properly configure the PATH environment variable to run the GNU tools just installed within the gcc4mbed directory. You may want to edit this script to further customize your development environment.

Mi a gcc4mbed?

- A **gcc4mbed** (GCC for Mbed) egy offline fejlesztői csomag, amely lehetővé teszi a **Mbed 2** projektek fordítását és karbantartását
- A GitHubról ZIP fájl formájában letölthető **gcc4mbed** csomag már tartalmazza az Mbed forráskód rev142 kiadását (ez az Mbed OS 5.4.5 kiadása), és telepítéskor automatikusan letölti az GCC ARM Embedded 6-2017-q1-update toolkitet
- A **gcc4mbed** csomag néhány windowsos segédprogramot is tartalmaz, köztük a **GNU Make 3.81** verzióját, amely a fordítás és összeállítás folyamatát irányítja. A végeredmény egy *.bin* fájl, ami „fogd és dobd” módszerrel a kártyára másolható *(drag and drop programming)*
- Az **mbed.org** korábbi online IDE-jével szemben a projektek konfigurálása nem kattintgatásokkal, hanem egy rövid és egyszerű **Makefile** megírásával történik
- Az ARM Mbed szerveréről letöltött régi projektek konvertálását és a konfiguráló **Makefile** állományok elkészítését mi egy **Powershell** scripttel automatizáltuk

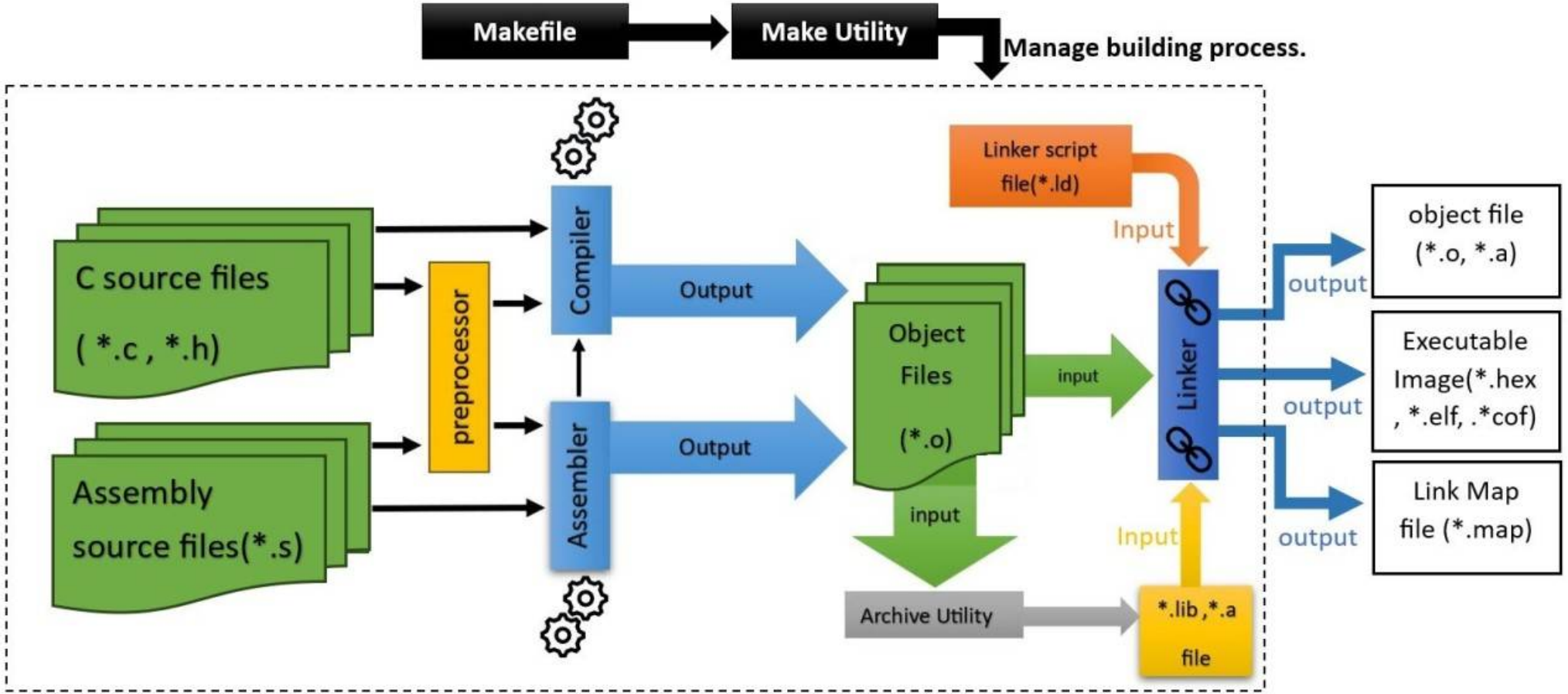
A gcc4mbed toolkit telepítése

- Töltse le a **gcc4mbed** csomagot a github.com/adamgreen/gcc4mbed/zipball/master címről
- Bontsa ki és helyezze a tartalmát abba a mappába, ahová a gcc4mbed-et és az ARM beágyazott processzorokhoz készült GNU eszközöket telepíteni szeretné
- A telepítési könyvtárban egy parancsablakból indítsa el a telepítő scriptet: Windows alatt **win_install.cmd**, macOS-en ./mac_install, Linuxon ./linux_install
- A telepítő automatikusan és sokáig dolgozik: letölti a teljes ARM GCC toolchain-t, majd ellenőrzésként négy különböző mbed-es célplatformra (KL25Z, K64F, LPC1768, NRF51_DK) a ***samples*** almappában lefordít néhány mintaprojektet.
  Ez a lépés nagyon időigényes mert első alkalommal le kell fordítania a teljes **mbed** forráskódot is
- A telepítés „hordozható”: nem módosítja a rendszer PATH-ját, nem telepít semmit a rendszerbe, minden a saját könyvtárában marad, és nyom nélkül törölhető

A gcc4mbed könyvtárstruktúrája

```
gcc4mbed/
├─ build/                    ← Buildrendszer target-specifikus .mk fájljai és a cgcc4mbed.mk
├─ external/
│   ├─ mbed-libs/            ← Kiegészítő könyvtárak helye (ezt majd mi hozzuk létre)
│   ├─ mbed-os/              ← Mbed rev142 (Mbed OS 5.4.5) teljes forrása
│   ├─ osx64/                ← macOS segédprogramok (gdb, make)
│   └─ win32/                ← Windows segédprogramok (make, curl, bsdtar, md5sum)
├─ gcc-arm-none-eabi/        ← A letöltött ARM GCC toolchain (fordító, linker, binutils)
├─ mbedUpdater/              ← Mbed frissítő és verziókezelő segédlet
├─ mri/                      ← MRI debug stub (soros hibakeresés támogatása)
├─ notes/                    ← Használati útmutatók
├─ samples/                  ← Mintaprojektek (HelloWorld, StdIO, stb.)
├─ src/                      ← A gcc4mbed saját forrásai (buildrendszer, eszközök)
├─ linux_install             ← Linux telepítő script
├─ mac_install               ← macOS telepítő script
├─ win_install.cmd           ← Windows telepítő script
├─ win_install.log           ← Telepítés naplója
├─ BuildShell.cmd            ← Windows build-környezet indítása
├─ BuildShellDebug.cmd       ← Debug-módú build-környezet
└─ BuildShellDevelop.cmd     ← Fejlesztői build-környezet
```

Makefile-alapú ARM GCC fordítási folyamat



gcc4mbed – az első lépések

- A **BuildShell.cmd** egy előre beállított parancssori környezetet indít, amelyben a **gcc4mbed** fordítórendszer minden szükséges útvonala és eszköze elérhető.
  A fordítást (make parancs) mindig ebből a parancssori környezetből kell indítani
- **Indítás:** fájl ablakból duplakattintással, de parancsikont is rendelhetünk hozzá
- A **HelloWorld** mintapélda (a beépített **LED1** villogtatása) lefordításához:
  - navigáljunk a **samples\HelloWorld** mappába
  - Konfiguráljunk a **Makefile** átszabásával
  - Majd adjuk ki a **make** parancsot
- A **DEVICES** paraméternél, szóközzel elválasztva több targetet is megadhatunk, így egyetlen **make** paranccsal több targetre is lefordíthatjuk a programot

```
PROJECT          := HelloWorld
DEVICES          := KL25Z \
                    NUCLEO_F446RE
GCC4MBED_DIR     := ../..
GCC4MBED_TYPE    := Release
MBED_OS_ENABLE   := 0
NEWLIB_NANO      := 1
NO_FLOAT_SCANF   := 1
NO_FLOAT_PRINTF  := 1

include $(GCC4MBED_DIR)/build/gcc4mbed.mk
```

gcc4mbed fordítási opciók

A Makefile állományokban leggyakrabban az alábbi paramétereket használjuk:

- **PROJECT** – A projekt neve; ezt a nevet kapják a kimeneti fájlok (.elf, .bin, .hex).
- **DEVICES** – A célplatformok listája, amelyekre a projektet lefordítani akarjuk
- **GCC4MBED_DIR** – A **gcc4mbed** gyökérkönyvtárának elérési útja
- **GCC4MBED_TYPE** – Build típusa: Debug, **Release** vagy Develop
- **MBED_OS_ENABLE** – **1:** mbed +RTOS, **0:** a klasszikus mbed 2 (RTOS nélkül)
- **NEWLIB_NANO** – **1:** kisméretű *newlib-nano*, **0:** teljes *newlib* könyvtár
- **NO_FLOAT_SCANF** – **1:** scanf nem támogat lebegőpontos értékeket (kisebb kódméret)
- **NO_FLOAT_PRINTF** – 1: printf nem támogat lebegőpontos értékeket (kisebb kódméret)
- **USER_LIBS** – Felhasználói könyvtárak listája; **!** előtaggal tiltott rekurzió az include-okra
- **DEFINES** – A fordítónak átadott előfeldolgozó makrók (pl. `DEFINES := DEBUG=1`)

A további Makefile-paraméterek ismertetése a **makefile.creole** dokumentumban található

Automatikus projektkonvertálás – 2/1.

- Ez a Powershell szkript minden, az Mbed online archívumból lementett projekt ZIP állományból létrehoz egy projektmappát, kimásolja belőle a **main.cpp**-t, beállítja az **MBED_OS_ENABLE** értékét (**1:** ha a projektnév tartalmazza az **„rtos”** szót, egyéként pedig **0**), majd generál egy **gcc4mbed**-hez való **Makefile**-t a projekthez

convert_projects.ps1

```
# projects mappa létrehozása
$projectsDir = "projects"
New-Item -ItemType Directory -Path $projectsDir -Force | Out-Null

# ZIP-ek feldolgozása
Get-ChildItem *.zip | ForEach-Object {
    $name = $_.BaseName
    $proj = Join-Path $projectsDir $name
    New-Item -ItemType Directory -Path $proj -Force | Out-Null

    # ideiglenes kicsomagolás
    $tmp = Join-Path $env:TEMP ([IO.Path]::GetRandomFileName())
    New-Item -ItemType Directory -Path $tmp | Out-Null
    Expand-Archive $_.FullName $tmp
```

Automatikus projektkonvertálás – 2/2.

```
    # main.cpp másolása
    $main = Get-ChildItem $tmp -Recurse -Filter main.cpp | Select-Object -First 1
    if ($main) {
        Copy-Item $main.FullName (Join-Path $proj "main.cpp") -Force
    } else { Write-Host "Nincs main.cpp a(z) $name projektben!"  }

    # MBED_OS_ENABLE automatikus beállítása
    $mbed = if ($name -match "rtos") { 1 } else { 0 }

    # Makefile generálása
@"
PROJECT         := $name
DEVICES         := KL25Z \
                   NUCLEO_F446RE
GCC4MBED_DIR    := ../..
GCC4MBED_TYPE   := Release
MBED_OS_ENABLE  := $mbed
NEWLIB_NANO     := 1
NO_FLOAT_SCANF  := 1
NO_FLOAT_PRINTF := 1

include \$(GCC4MBED_DIR)/build/gcc4mbed.mk
"@ | Set-Content -Encoding ASCII (Join-Path $proj "Makefile")
}
```

Kiegészítő mbed könyvtárak

```
mbed-libs
├── BufferedSerial
│   ├── BufferedSerial.cpp
│   └── BufferedSerial.h
├── ComparatorIn
│   ├── ComparatorIn.cpp
│   └── ComparatorIn.h
├── Hotboards_keypad
│   ├── Hotboards_keypad.cpp
│   ├── Hotboards_keypad.h
│   ├── Key.cpp
│   ├── Key.h
│   └── README
├── MMA8451Q
│   ├── MMA8451Q.cpp
│   └── MMA8451Q.h
├── TSI
│   ├── TSISensor.cpp
│   └── TSISensor.h
└── tsi_sensor
    ├── readme
    ├── tsi_sensor.cpp
    └── tsi_sensor.h
```

- Ezek a könyvtárak korábbi Mbed-projektjeimből származnak: a projektekben található .lib fájlok valójában URL-hivatkozásokat tartalmaznak, és ezek alapján tölthetők le a hozzájuk tartozó eredeti Mbed-könyvtárak
- A **gcc4mbed** környezetben is használhatók ezek a könyvtárak különféle kiegészítő funkciók (bufferelt soros kommunikáció, érzékelők, keypad stb.) biztosítására, csak hivatkozni kell rájuk a projektben és az elérhetőségüket mag kell adni a projekthez tartozó **Makefile**-ban
- A `USER_LIBS` olyan könyvtárakat sorol fel, amelyekben a gcc4mbed automatikusan megkeresi a .cpp fájlokat, és hozzáadja őket a fordításhoz. Például:

```
USER_LIBS :=  ../../external/mbed-libs/TSI \
              ../../external/mbed-libs/BufferedSerial \
              ../../external/mbed-libs/MMA8451Q
```

Projektek tömeges fordítása

- **Mbed** projektek tömeges fordításához készítettük az alábbi Makefile-t, amelyet a projektek gyűjtőmappájában kell elindítani. Végignézi a projektek neveit, kiválogatja azokat a projektkönyvtárakat, amelyek neve a megadott minták valamelyikével kezdődik (pl. 04), majd mindegyiket külön alfolyamatban lefordítja
- A fordítás több processzormagon is folyhat (ez különösen akkor hasznos, amikor sok fáljt kell fordítani), ekkor a make parancs -j opcióját kell használni Például: **make all -j16**

```
MAKEFLAGS += -rR

# Projekt lista lekérése és szűrése
ALL_PROJECTS := $(shell dir /b /ad)
ALLOWED_NUMS := 02 04 05 09
PATTERNS := $(addsuffix %, $(ALLOWED_NUMS))
FILTERED_PROJECTS := $(filter $(PATTERNS), $
(ALL_PROJECTS))

# Target-alapú felbontás
BUILD_TARGETS := $(addprefix build-, $
(FILTERED_PROJECTS))
CLEAN_TARGETS := $(addprefix clean-, $
(FILTERED_PROJECTS))

.PHONY: all clean $(BUILD_TARGETS) $(CLEAN_TARGETS)

all: $(BUILD_TARGETS)
$(BUILD_TARGETS): build-%:
     @cd $* && $(MAKE)

clean: $(CLEAN_TARGETS)
$(CLEAN_TARGETS): clean-%:
     @cd $* && $(MAKE) clean
```

A korábbi tananyag mintapéldáinak újraélesztése

- A **FRDM KL25Z** kártya **ARMmbed** környezetben történő programozásáról 2016. tavaszán készült tananyag mintapéldáit „újraéleszthetjük” a **gcc4mbed** csomag segítségével

- Az archivált projektek innen tölthetők le: os.mbed.com/users/icserny/code/ illetve os.mbed.com/teams/Megtestesules-Plebania-hobbielektronika-/code/

- A csomagok csoportos letöltése, kibontása és átkonvertálása „kézzel”, vagy az előző oldalakon bemutatott Powershell scriptekkel is elvégezhető

- A következőkben néhány mintapéldán keresztül bemutatjuk, hogy a projektek konfigurálásánál mire kell ügyelni

02_ledblink_other_way

- Az „alap” Makefile: **newlib-nano,** nincs lebegőpontos scanf/printf, nem használjuk az RTOS ütemezőt, **Release** build mód

main.cpp

```
#include "mbed.h"
DigitalInOut myled(LED_GREEN);

int main() {
    while(1) {
        myled.output();
        myled = 0;
        wait(0.2);
        myled.input();
        wait(0.2);
    }
}
```

Makefile

```
PROJECT          := 02_ledblink_other_way
DEVICES          := KL25Z
GCC4MBED_DIR     := ../..
GCC4MBED_TYPE    := Release
MBED_OS_ENABLE   := 0
NEWLIB_NANO      := 1
NO_FLOAT_SCANF   := 1
NO_FLOAT_PRINTF  := 1

include $(GCC4MBED_DIR)/build/gcc4mbed.mk
```

04_TSISensor

- A programban a TSI kiegészítő könyvtárat használjuk, ennek elérhetőségét meg kell adni a Makefile-ban

main.cpp

```
#include "mbed.h"
#include "TSISensor.h"

int main(void) {
    PwmOut led(LED_BLUE);
    TSISensor tsi;
    led = 1.0;
    while (true) {
      float s = tsi.readPercentage();
        if (s > 0.01) led = 1.0 - s;
        wait(0.1);
    }
}
```

Makefile

```
PROJECT          := 04_TSISensor
DEVICES          := KL25Z
GCC4MBED_DIR     := ../..
GCC4MBED_TYPE    := Release
USER_LIBS := ../../external/mbed-libs/TSI
MBED_OS_ENABLE   := 0
NEWLIB_NANO      := 1
NO_FLOAT_SCANF   := 1
NO_FLOAT_PRINTF  := 1

include $(GCC4MBED_DIR)/build/gcc4mbed.mk
```

05_ADC_extra

- A programban lebegőpontos (float) kiírást használunk, ehhez a megfelelő könyvtár becsatolásához `NO_FLOAT_PRINTF := 0` beállítás kell

```
#include "mbed.h"
AnalogIn adc(A0);
...
int main(void) {
  while (true) {
    uint16_t a1 = adc_read(8);
    uint16_t a2 = adc_read(26);
    uint16_t a3 = adc_read(27);
    float v1 = a1*3.3f/65536;
    float v2 = a2*3300.0f/65536;
    float temp = 25.0f-(v2-v25)*1000.0f/m;
    float v3 = a3*3.3f/65536;
    printf("A0 = %6.3f V  Temp = %5.2f C  Bgap = %6.3f V\n",v1,temp,v3);
    wait_ms(2000);
  }
}
```

Az **adc_read()** függvényt most nem részletezzük

main.cpp

Makefile

```
DEVICES          := KL25Z
PROJECT          := 05_ADC_extra
GCC4MBED_DIR     := ../..
GCC4MBED_TYPE    := Release
MBED_OS_ENABLE   := 0
NEWLIB_NANO      := 1
NO_FLOAT_SCANF   := 1
NO_FLOAT_PRINTF  := 0
include $(GCC4MBED_DIR)/build/gcc4mbed.mk
```

09_rtos_basic

- Be kell csatolni az ”rtos.h” fejléc állományt, a Makefile-ban pedig `MBED_OS_ENABLE := 1` beállítás kell

main.cpp

```
#include "mbed.h"
#include "rtos.h"

...

int main() {
    Thread thread2(led2_thread);
    Thread thread3(led3_thread);

    while (true) {
        led1 = !led1;
        Thread::wait(1000);
    }
}
```

Makefile

```
PROJECT             := 09_rtos_basic
DEVICES             := KL25Z \
                       NUCLEO_F446RE
GCC4MBED_DIR        := ../..
GCC4MBED_TYPE       := Release

MBED_OS_ENABLE      := 1
NEWLIB_NANO         := 1
NO_FLOAT_SCANF      := 1
NO_FLOAT_PRINTF     := 1

include $(GCC4MBED_DIR)/build/gcc4mbed.mk
```