

Laborator 1

Prezentarea platformei Nucleo și a mediului de dezvoltare

Secțiuni lucrare:

- [Ce e ARM?](#)
- [Placă de dezvoltare NUCLEO](#)
- [KEIL IDE](#)

Echipamente necesare lucrării:

- Placă de dezvoltare NUCLEO STM32F103RB
- Cablu mini usb

Ce e ARM?

Istoria ARM

ARM a fost fondată la sfârșitul anilor 90, ca o subdivize la o altă companie numită Acorn Computers. Acorn Computers realiza calculatoare personale de tip desktop pentru școlile din Marea Britanie. În 1982- 1986 o echipă mică din cadrul companiei Acorn Computers a avut ca principal obiectiv găsirea unui procesor pe piață care să îndeplinească cerințele impuse. După un timp au realizat că nu există un produs pe piață cu toate cerințele impuse de ei, prin urmare au decis să realizeze propriul procesor. În doar 18 luni au reușit să facă design-ul și implementarea unui nou procesor, care a fost folosit pe o masină Acorn Computers în 1995. Calculatorul purta numele de Acorn RISC Machine. După ce compania Acorn Computers a dat faliment, devizia care se ocupa de procesoare a creat o nouă companie numită **Advanced RISC Machine** în noiembrie 1990. Acum atât compania cât și procesoarele sunt cunoscute ca ARM. Compania ARM este cunoscută în special datorită gamei largi de procesoare RISC dar și pentru alte produse software și hardware pe care le dezvoltă alături de un număr mare de parteneri. Compania ARM nu realizează propriile cipuri. Ei fac design-ul cipurilor și vând proprietatea intelectulă către alte companii.

Unde se gasesc procesoarele ARM?

Procesoarele ARM se pot găsi într-o gamă variată de produse, pornind de la imprimante și hardisk-uri la device-uri de gamming și mașini de spălat, frigidere sau roboți (figura 1). Datorită acestei game foarte largi de folosire, s-au produs peste 30 de miliarde de procesoare ARM și numărul continuă să crească, ajungând la o cifră estimativă de 100 de miliarde de procesoare până în 2020.



Figura 1. Diverse dispozitive cu procesoare ARM

Familia ARM

Primul procesor din familia ARM a fost ARM7TDMI-S care este un procesor încorporat și în prezent în produse, dar pentru care ARM nu mai vinde licențe. Pornind de la primul procesor, s-au dezvoltat o serie de noi procesoare cu diverse îmbunătățiri putând fi folosite la o gamă cât mai mare de aplicații (figura 2). Momentan, procesoarele ARM se pot împărți în două mari categorii: procesoare dedicate aplicațiilor (Cortex-A) și procesoare dedicate sistemelor embedded (Cortex M, Cortex R). Atât familia de procesoare Cortex-M cât și familia de procesoare Cortex-R sunt destinate pentru a fi folosite în microcontrolere. Familia de microcontrolere Cortex-R se deosebește de Cortex-M prin capacitatea de a realiza mai multe calcule pe secundă, prin mecanismul său sofisticat de sincronizare (clock) dar și prin implementarea hardware a mai multor mecanisme care reduc timpul de răspuns la întreruperi. Procesoarele din familia Cortex-R sunt ideale pentru aplicații unde timpul este critic precum controlere de Hard-Disk sau sisteme care controlează motoarele cu combustie internă. Familia de procesoare Cortex-M este destinată pentru a fi folosită în microcontrolere la care se pune accent pe cost. Procesoarele din familia Cortex-A sunt folosite pentru platformele care necesită un sistem de operare precum Linux. Ele încorporează sisteme sofisticate de management a memoriei și un set extins de instrucțiuni pentru aplicații multimedia.

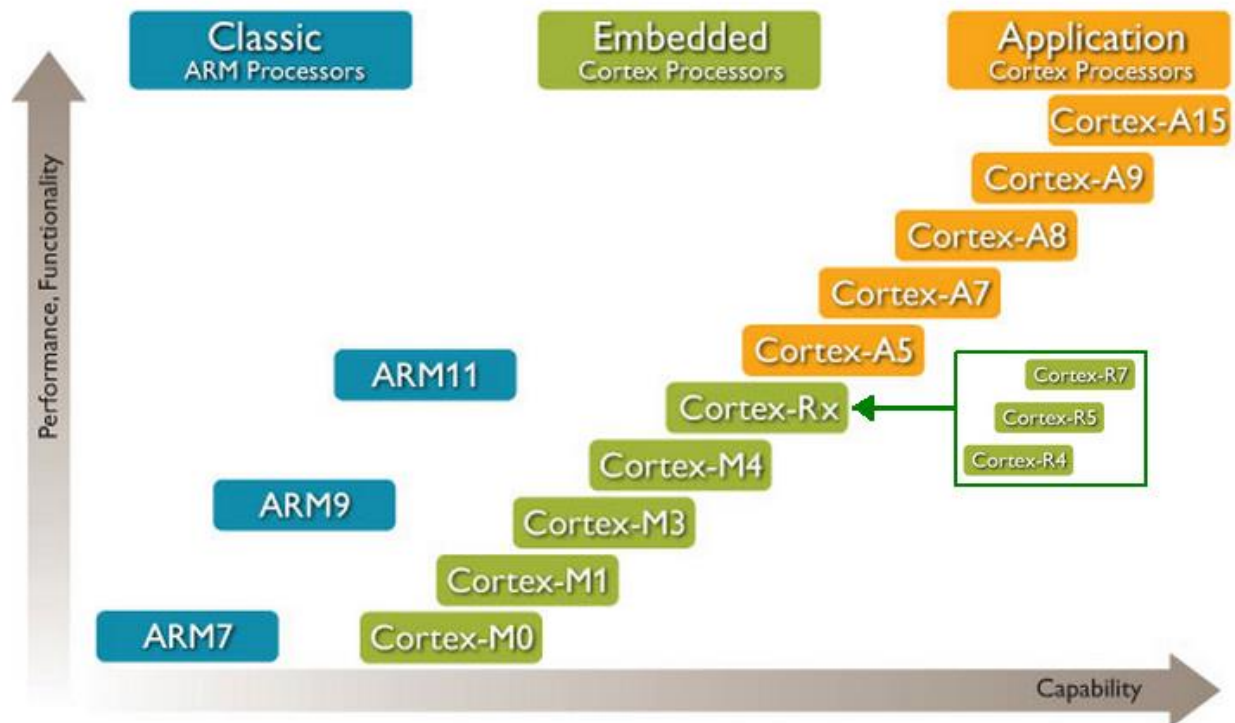


Figura 2. Categoriile de procesoare

ARM Cortex M

Familia de procesoare ARM Cortex M este o familie optimizată pentru aplicații embedded având un cost redus. Toate versiunile din familia Cortex M suportă doar setul de instrucțiuni Thumb.

- ARM Cortex M0 - Este un procesor cu număr redus de porți logice, ceea ce îl face foarte eficient din punct de vedere energetic.
- ARM Cortex M0+ - Este un procesor cu număr redus de porți logice, eficient din punct de vedere energetic care este gândit pentru microcontrolere și aplicații embedded care necesită atât optimizare din punct de vedere energetic cât și un set larg de configurări. Setul de instrucțiuni este compatibil cu ARM Cortex M0, prin urmare se pot folosi același compilator și unelte de debug.
- ARM Cortex M1 - Este dedicat aplicațiilor embedded care necesită un mic procesor integrat în FPGA.
- ARM Cortex M3 - Este un procesor low-power cu număr redus de porți logice, folosit în aplicații embedded unde este necesar un timp de răspuns scăzut la întreruperi, inclusiv în microcontrolere din industria auto și sisteme de control industrial.

- ARM Cortex M4 – Este un procesor low power, cu un număr redus de porți logice având un timp de răspuns scăzut la întreruperi. Cortex-M4F este un procesor cu aceleași capabilități ca și Cortex-M4, incluzând calcule cu numere cu virgulă flotantă. Aceste procesoare se folosesc în aplicații care necesită procesare de semnal.

Placa de dezvoltare NUCLEO

Sunt mai multe plăci din familia Nucleo, fiecare destinată pentru diverse aplicații embedded, pornind de la aplicații low power până la aplicații de procesare de semnal. În figura 3 se pot vedea diversele tipuri de plăci NUCLEO. Placa folosită în acest laborator face parte din grupul NUCLEO-64. Numărul de 64 vine de la numărul de pini ai procesorului folosit. NUCLEO-32 folosesc microcontrolere cu 32 de pini și NUCLEO 144, microcontrolere cu 144 de pini.

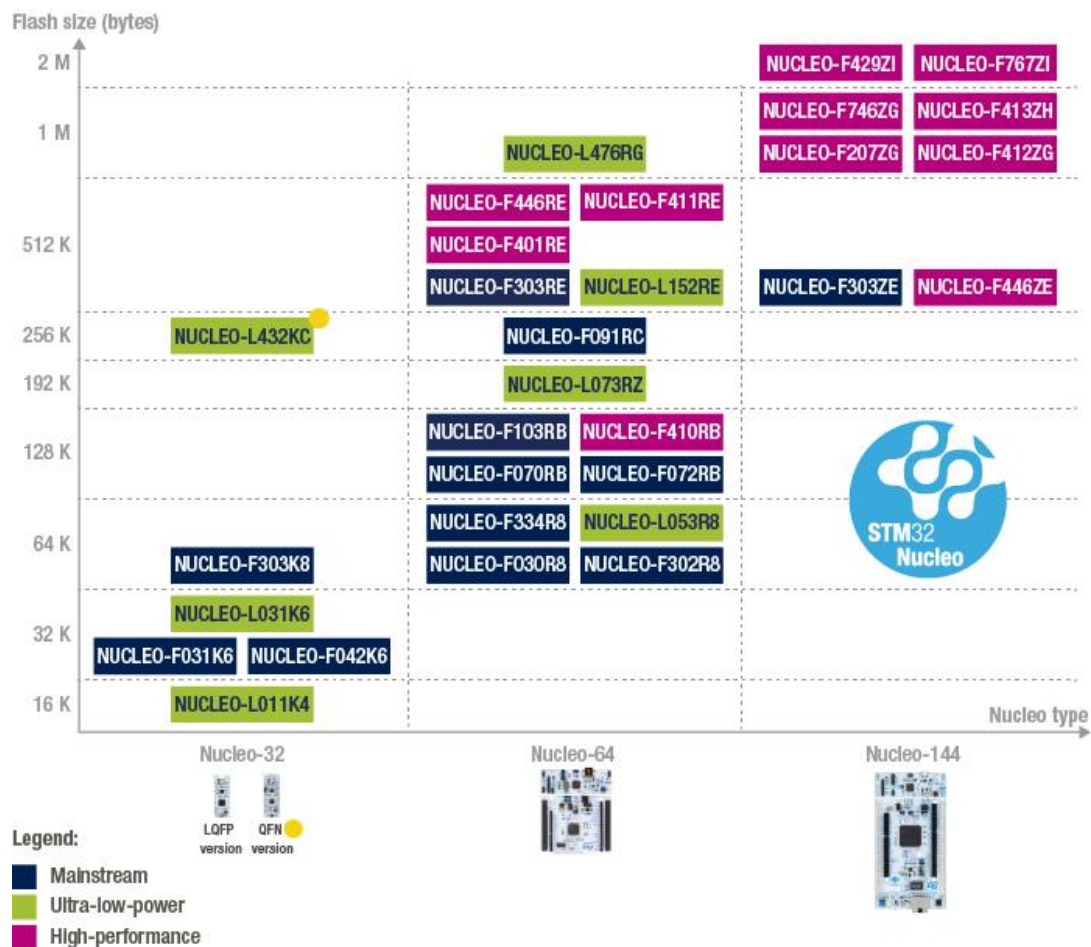


Figura 3. Plăci NUCLEO

Plăcile de dezvoltare STM32 Nucleo au un microcontroler din familia STM32 (o familie de microcontrolere pe 32 de biți dezvoltată de STMicroelectronics) care se bazează pe nucleele ARM Cortex M0, M3 sau M4. Familia de microcontrolere STM32F103xx sunt caracterizate de o frecvență de până la 72 MHz, 64sau 128 Kb de memorie Flash și 20 Kb de SRAM, 16 canale pentru conversia analog numerică pe 12 biți, timere, SPI, I2C, USART.

În acest laborator se va lucra cu placa NUCLEO STM32F103RB, bazată pe ARM Cortex M3 (figura 4). Aceasta are două tipuri de conectori: conectori mamă, compatibili cu placa de dezvoltare Arduino Uno și pini tată care permit accesul utilizatorului la toți cei 64 de pini ai microcontrolerului. ST-LINK este un programator/debugger cu conector SWD, încorporat în placa Nucleo dar care poate fi folosit independent. Programatorul/debugger-ul poate fi folosit și cu alte microcontrolere din familia STM32, fiind conceput să se poată desprinde de placa NUCLEO. Alimentarea se poate face fie prin conectorul USB, fie de la o sursă externă (3.3V-12V).

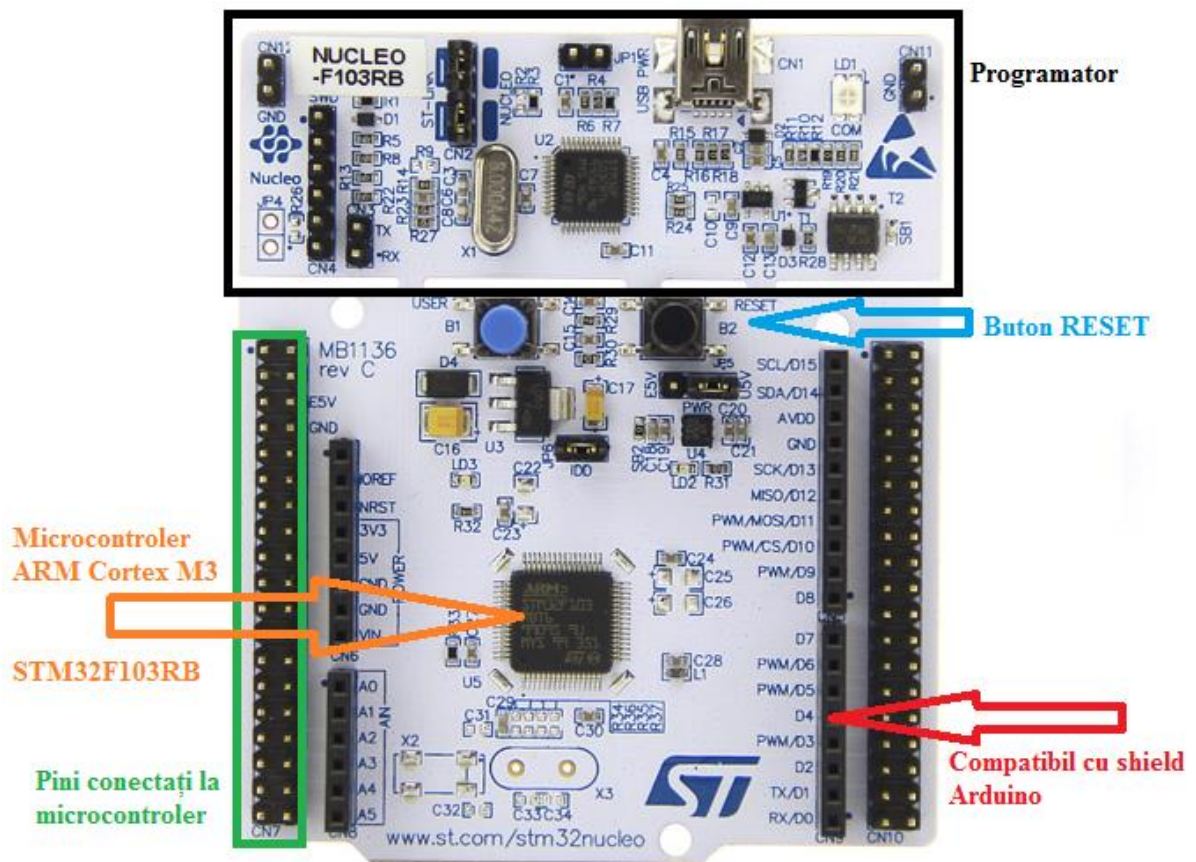


Figura 4. Nucleo STM32F103RB

KEIL IDE

Keil este un mediu de dezvoltare conceput de compania ARM, care include compilator de C/C++, cu un editor de text numit uVision și o suită de unelte de debug pentru diversele microcontrolere din familia ARM. În acest laborator se va folosi versiunea Keil uVision 5. Pentru downloadarea KEIL IDE accesați acest link <https://www.keil.com/download/product/>. Pentru a se putea lucra cu o placă precum NUCLEO, este necesară instalarea pe PC a unui software care să recunoască placa (driver). Software-ul se numeste ST-Link V2 și se poate downloada de la această adresă: <http://www.st.com/en/development-tools/st-link-v2.html>.

Proiect nou în Keil

Imediat după instalearea mediului Keil veți fi întâmpinat de fereastra modulului Pack Installer (Figura 5).

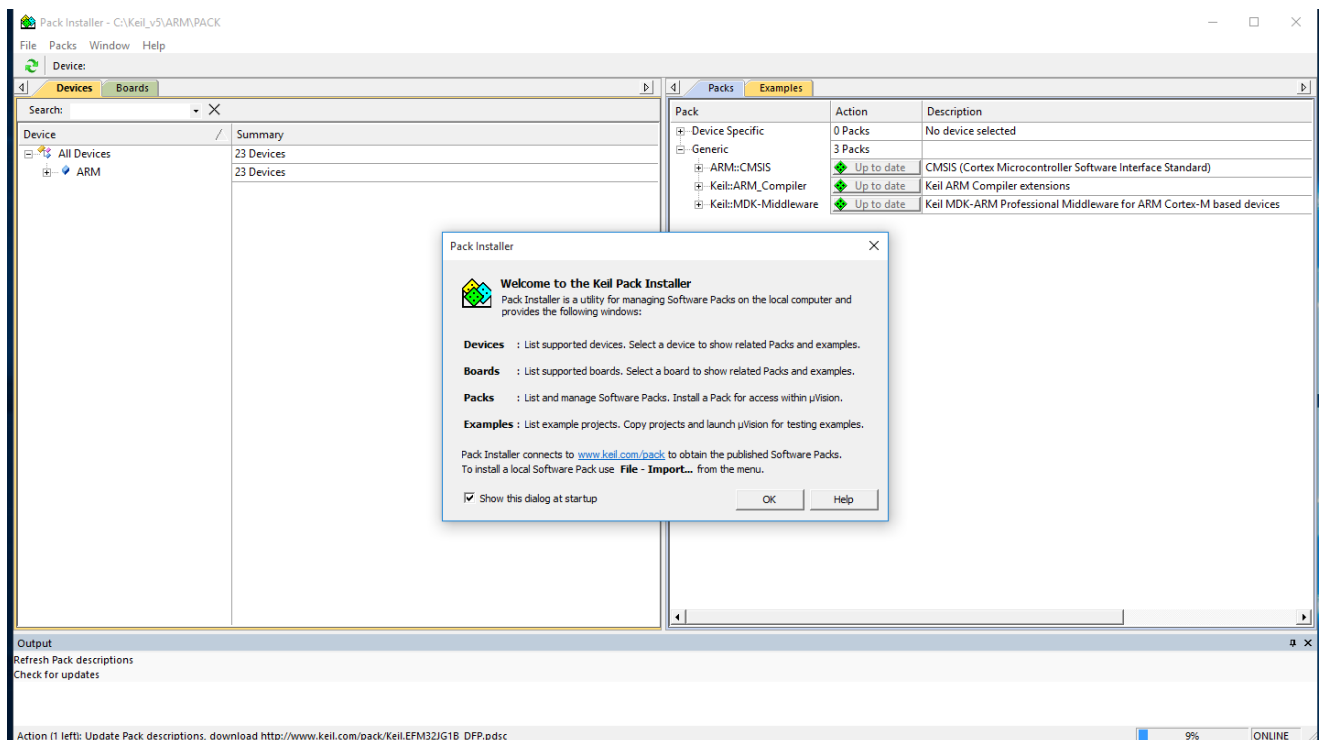


Figura 5. Instalare Keil

În partea din dreapta jos din figura 5, se poate vedea un progress bar. Keil downloadează o listă cu toate plăcile și microcontrolerele cu care acesta poate să lucreze, alături de exemple de cod și module software. După ce s-a încheiat acest pas, trebuie să căutați în lista de microcontrolere varianta de microcontroler cu care lucrați. În cazul acestui laborator se va lucra cu un microcontroler de la STMicroelectronics, STM32F103RB.

În imaginea de mai jos (figura 6), se poate vedea lista completă cu producătorii de microcontrolere. Din aceasta, a fost ales producatorul STMicroelectronics și microcontrolerul STM32F103. După ce microcontrolerul a fost ales (pasul 1), din cel de al doilea tab sub Packs (pasul 2) se vor instala KeilSTm32Fxxx_DFP și KeilSTM32NUCLEO_BOARD. Acestea vor downloada și instala biblioteci în C, specific create pentru microcontroler.

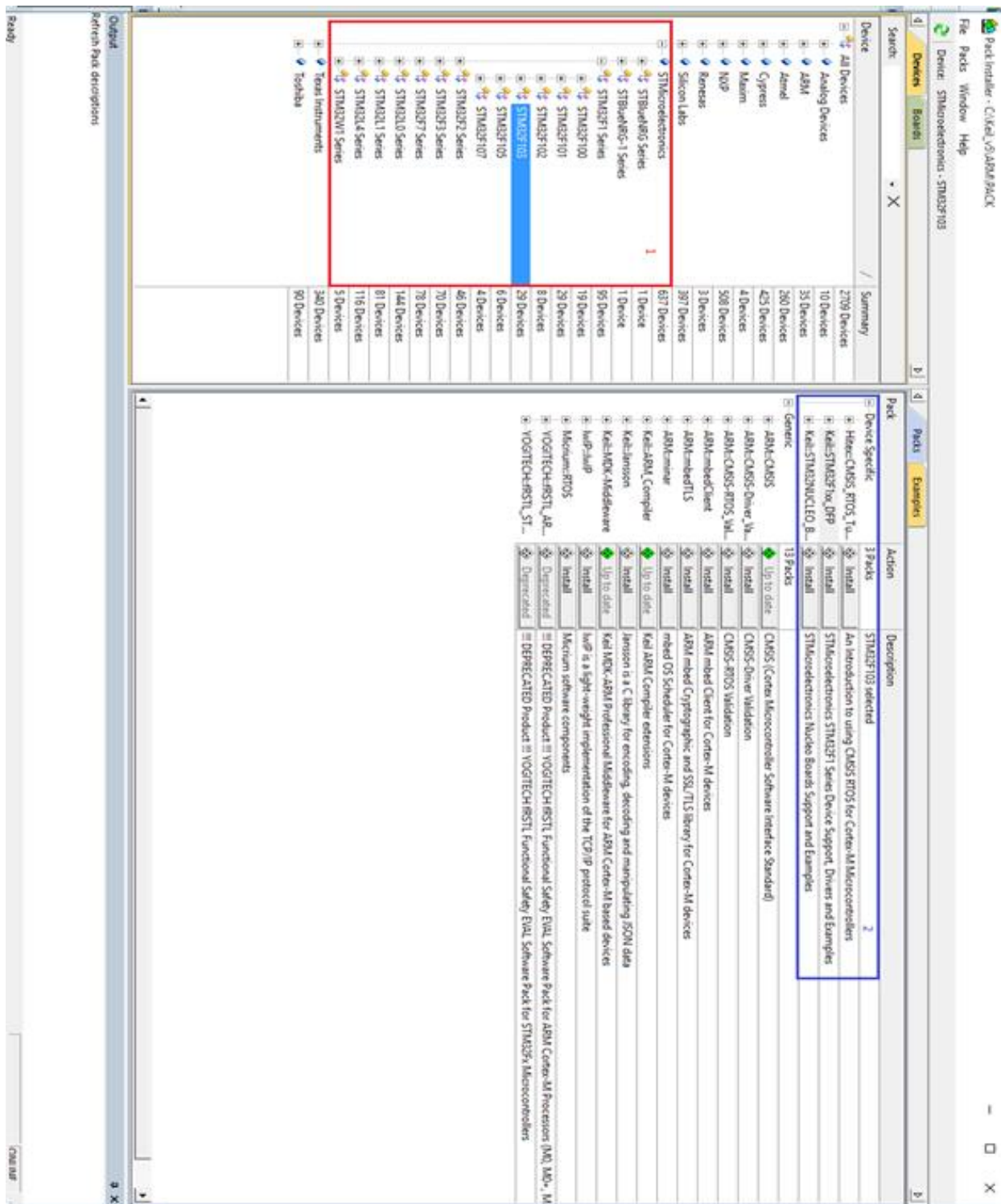


Figura 6. Pack Installer - Devices

Cearea unui proiect în Keil uVision

După deschiderea mediului se dă click pe Project -> New uVision Project. Se selectează microcontrolerul ales, în cazul nostru, STM32F103 -> STM32F103RB. Se dă click pe butonul OK, conform imaginii de mai jos (figura 7).

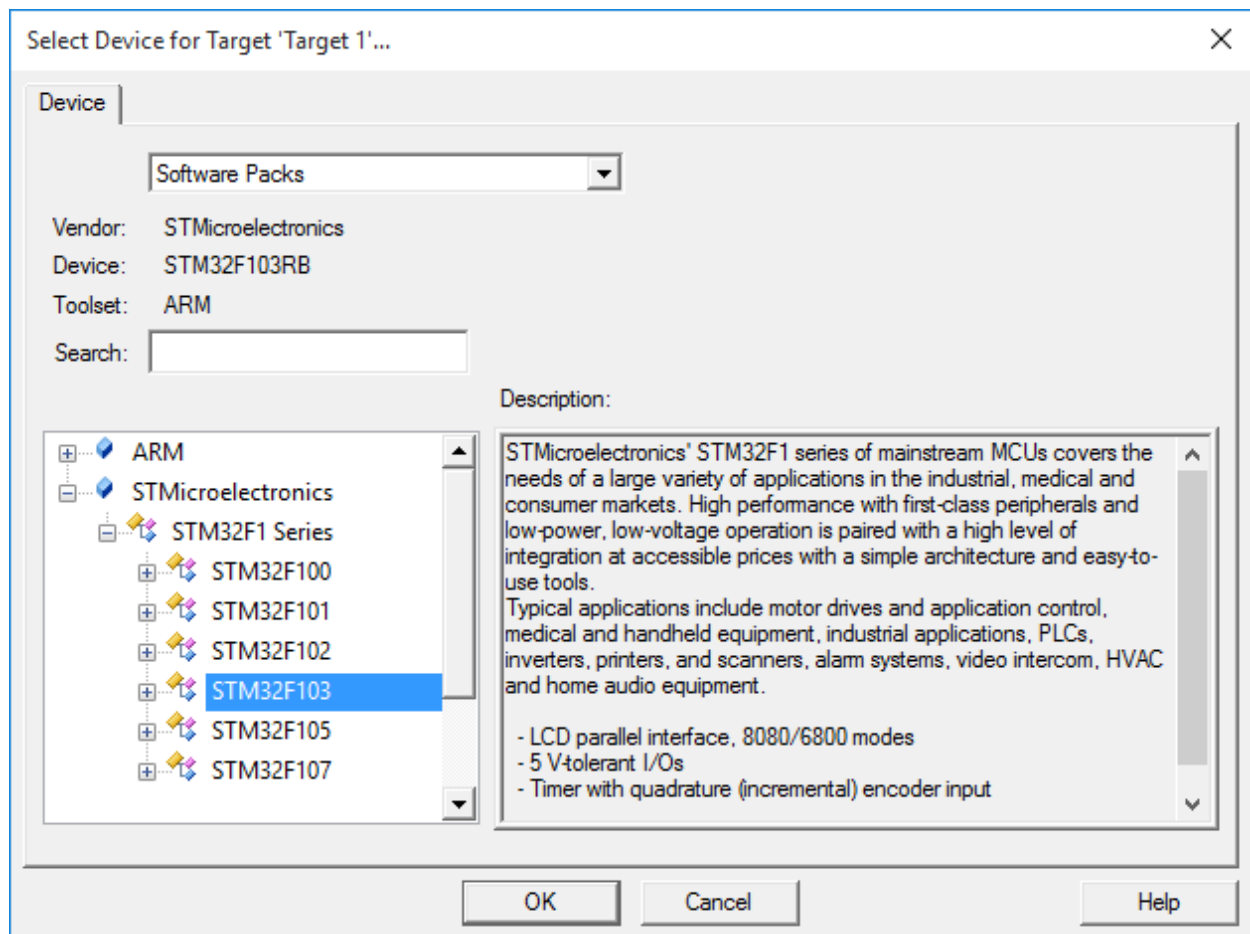



Figura 7. Selectarea dispozitivului

Următoarea fereastră care apare este Manage Run-Time Environment. Din această fereastră, se pot include diverse API-uri speciale pentru microcontrolerul ales sau pentru o placă precum NUCLEO, API-uri care să conțină metode pentru lucru cu led-uri, cu memoria sau cu diverse periferice. Pentru exemplul din acest laborator, se vor selecta modulele din imaginea de mai jos. Primul lucru care trebuie setat este Startup din tabul Device. Start-up realizează

inițializări pentru stivă, pentru semnalul de ceas și multe alte module, până a se apela funcția main()).

 Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Board Support		NUCLEO-F103RB	1.0.0	STMicroelectronics NUCLEO-F103RB Development Board
CMSIS				Cortex Microcontroller Software Interface Components
CORE	<input checked="" type="checkbox"/>		5.0.0	CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M
DSP	<input type="checkbox"/>		1.4.6	CMSIS-DSP Library for Cortex-M, SC000, and SC300
RTOS (API)			1.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
RTOS2 (API)			2.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
Compiler		ARM Compiler	1.1.0	Compiler Extensions for ARM Compiler ARMCC and ARMClang
Device				Startup, System Setup
DMA	<input type="checkbox"/>		1.2	DMA driver used by RTE Drivers for STM32F1 Series
GPIO	<input type="checkbox"/>		1.2	GPIO driver used by RTE Drivers for STM32F1 Series
Startup	<input checked="" type="checkbox"/>		1.0.0	System Startup for STMicroelectronics STM32F1xx device series
StdPeriph Drivers				
ADC	<input type="checkbox"/>		3.5.0	Analog-to-digital converter (ADC) driver for STM32F10x
BKP	<input type="checkbox"/>		3.5.0	Backup registers (BKP) driver for STM32F10x
CAN	<input type="checkbox"/>		3.5.0	Controller area network (CAN) driver for STM32F1xx
CEC	<input type="checkbox"/>		3.5.0	Consumer electronics control controller (CEC) driver for STM32F1xx
CRC	<input type="checkbox"/>		3.5.0	CRC calculation unit (CRC) driver for STM32F1xx
DAC	<input type="checkbox"/>		3.5.0	Digital-to-analog converter (DAC) driver for STM32F1xx
DBGMCU	<input type="checkbox"/>		3.5.0	MCU debug component (DBGMCU) driver for STM32F1xx
DMA	<input type="checkbox"/>		3.5.0	DMA controller (DMA) driver for STM32F1xx
EXTI	<input type="checkbox"/>		3.5.0	External interrupt/event controller (EXTI) driver for STM32F1xx
FSMC	<input type="checkbox"/>		3.5.0	Flexible Static Memory Controller (FSMC) driver for STM32F1xx
Flash	<input checked="" type="checkbox"/>		3.5.0	Embedded Flash memory driver for STM32F1xx
Framework	<input checked="" type="checkbox"/>		3.5.1	Standard Peripherals Drivers Framework
GPIO	<input checked="" type="checkbox"/>		3.5.0	General-purpose I/O (GPIO) driver for STM32F1xx
I2C	<input type="checkbox"/>		3.5.0	Inter-integrated circuit (I2C) interface driver for STM32F1xx
IWDG	<input type="checkbox"/>		3.5.0	Independent watchdog (IWDG) driver for STM32F1xx
PWR	<input type="checkbox"/>		3.5.0	Power controller (PWR) driver for STM32F1xx
RCC	<input checked="" type="checkbox"/>		3.5.0	Reset and clock control (RCC) driver for STM32F1xx
RTC	<input type="checkbox"/>		3.5.0	Real-time clock (RTC) driver for STM32F1xx
SDIO	<input type="checkbox"/>		3.5.0	Secure digital (SDIO) interface driver for STM32F1xx
SPI	<input type="checkbox"/>		3.5.0	Serial peripheral interface (SPI) driver for STM32F1xx
TIM	<input type="checkbox"/>		3.5.0	Timers (TIM) driver for STM32F1xx
USART	<input type="checkbox"/>		3.5.0	Universal synchronous asynchronous receiver transmitter (USART) driver for STM32F1xx
WWDG	<input type="checkbox"/>		3.5.0	Window watchdog (WWDG) driver for STM32F1xx
File System		MDK-Pro	6.9.0	File Access on various storage devices
Graphics		MDK-Pro	5.36.6	User Interface on graphical LCD displays

Figura 8. Manage Run-Time Environment

După ce toate elementele din figura 8 au fost setate, se poate crea fișierul care conține funcția main(). Din fereastra Project, se extinde tabul Target 1 și se dă click dreapta pe folderul Source Group 1. Se dă click pe Add new Item to Group Source Group 1. Acești pași se pot vedea și în figura de mai jos.

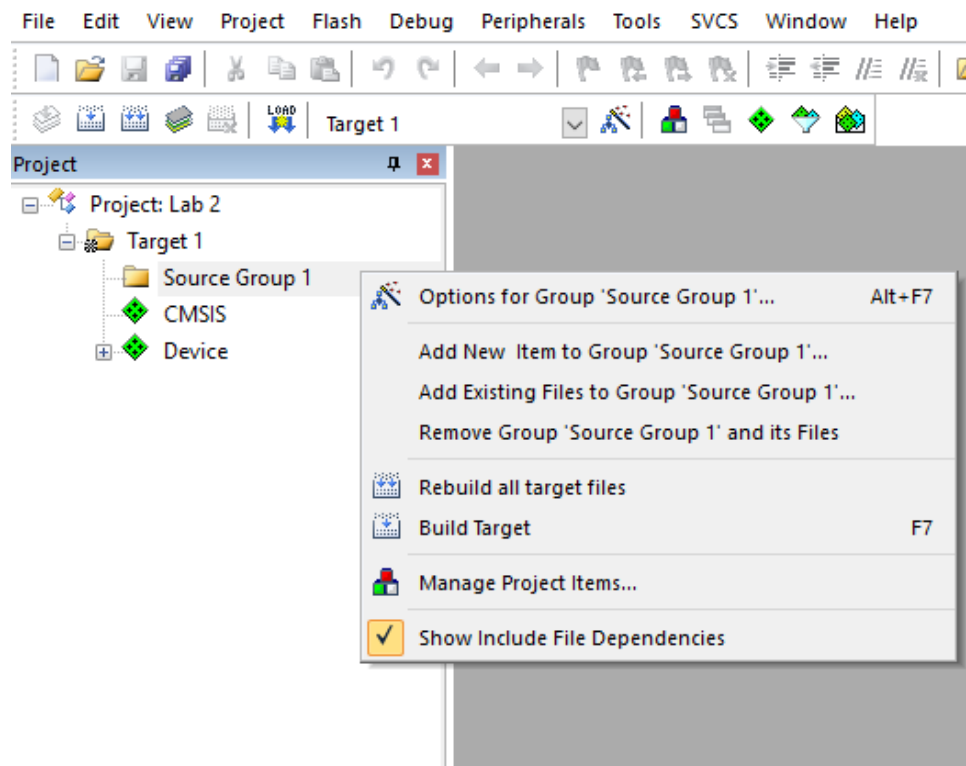


Figura 9. Adaugare fisiere

După acest pas, apare o fereastră asemănătoare cu cea din figura următoare (figura 10) în care trebuie să alegeți numele și tipul fișierului. Pentru acest exemplu, se va crea un fișier C file (.c) iar numele fișierului o să fie main.

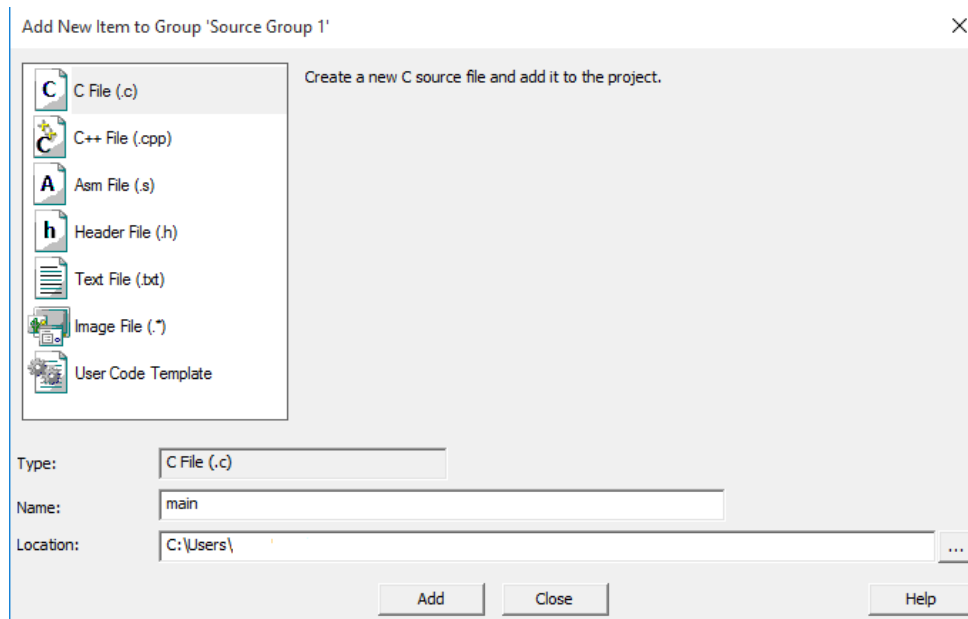


Figura 10. Salvarea noului fișier

În fișierul creat se copiază următorul cod iar modulele folosite vor fi explicate în detaliu în laboratoarele următoare.

```
#include "stm32f10x.h"
```

```
GPIO_InitTypeDef GPIO_InitStructure;
```

```
static __IO uint32_t TimingDelay;
```

```
void initGPIO()
```

```
{
```

```
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
```

```
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
```

```
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
```

```
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
```

```
    GPIO_Init(GPIOA, &GPIO_InitStructure);
```


```
}
```

```
void initTimer()
```

```
{
```

```
        SysTick_Config(SystemCoreClock / 1000);
    }
    void SysTick_Handler(void)
    {
        if (TimingDelay != 0x00)
        {
            TimingDelay--;
        }
    }
    void Delay(__IO uint32_t nTime)
    {
        TimingDelay = nTime;
        while(TimingDelay != 0);
    }
    int main()
    {
        initGPIO();
        initTimer();

        while(1)
        {
            GPIOA->BSRR = GPIO_Pin_5;
            Delay(1000);
            /* Reset PD0 and PD2 */
            GPIOA->BRR = GPIO_Pin_5;
            Delay(1000);
        }
    }
```


Codul de mai sus, aprinde și stinge led-ul de pe placă la interval de o secundă. Pentru a putea încărca codul pe placă, trebuie să setăm în mod corespunzător target-ul. Pentru aceasta, se dă click pe butonul Options for Target... . Din fereastra nou apărută se alege tabul Debug, și la câmpul Use se selectează ST-Link Debug, conform imaginii de mai jos.

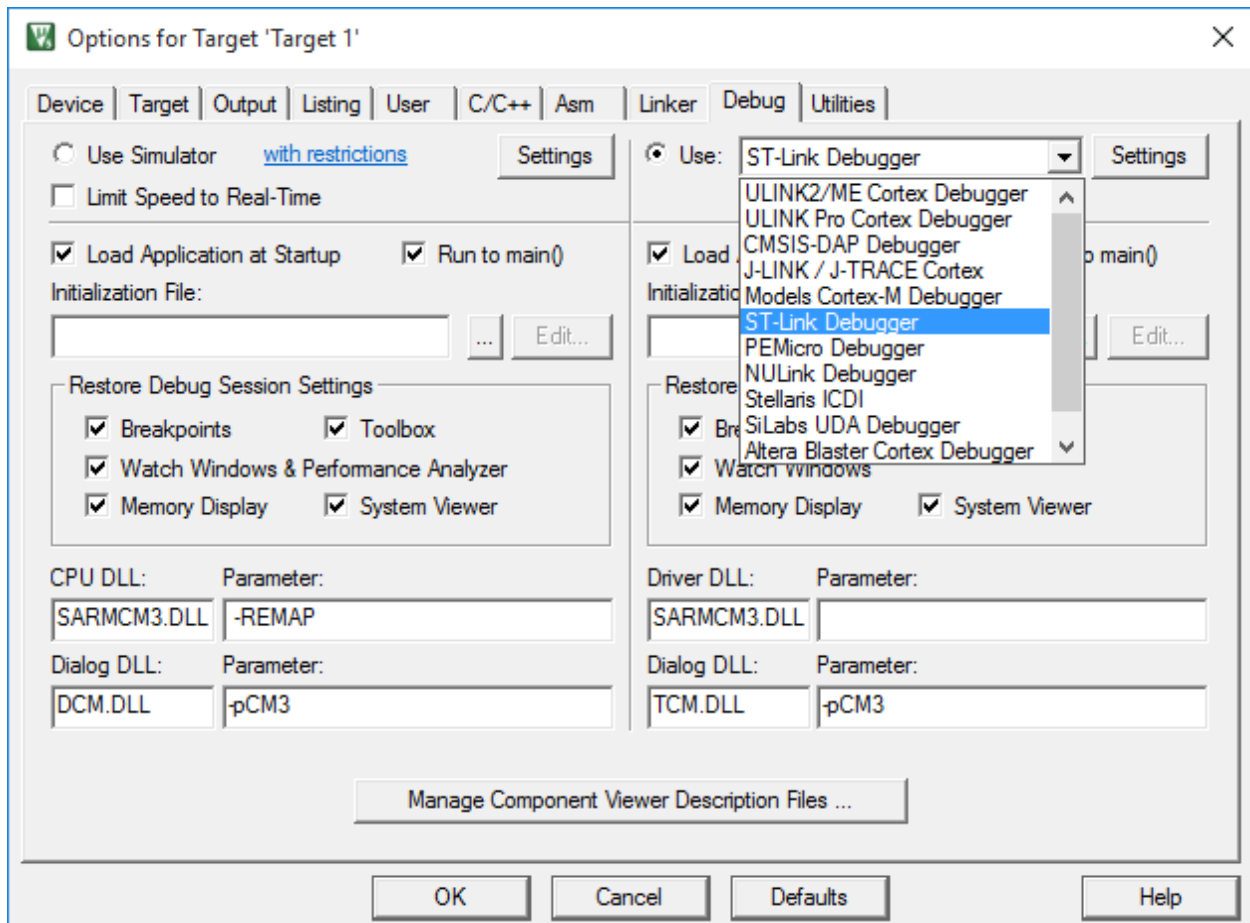


Figura 11. Target- Debug

După ce acest pas a fost făcut, se va apăsa butonul Settings din același tab, apoi Debug și se vor bifa de sub Download Option atât Verify Code Download cât și Download to Flash, așa cum se poate vedea în figura 12. Tot de aici se selectează tab-ul Flash Download și se bifează Reset and Run (figura 13).

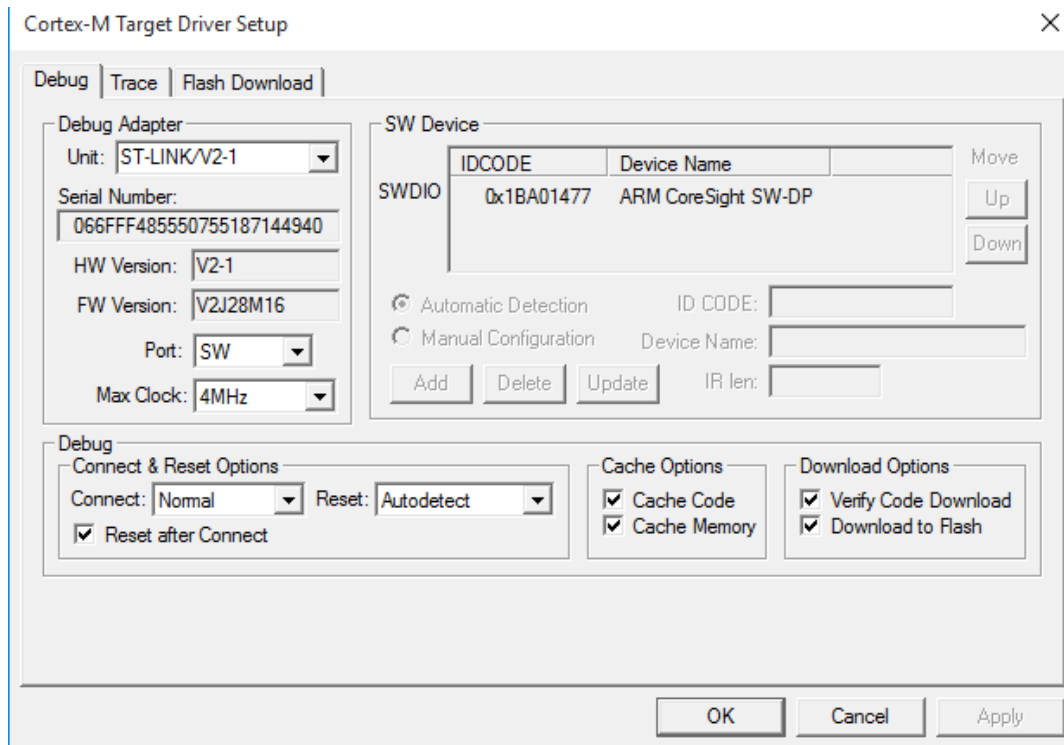


Figura 12. Tab-ul Debug

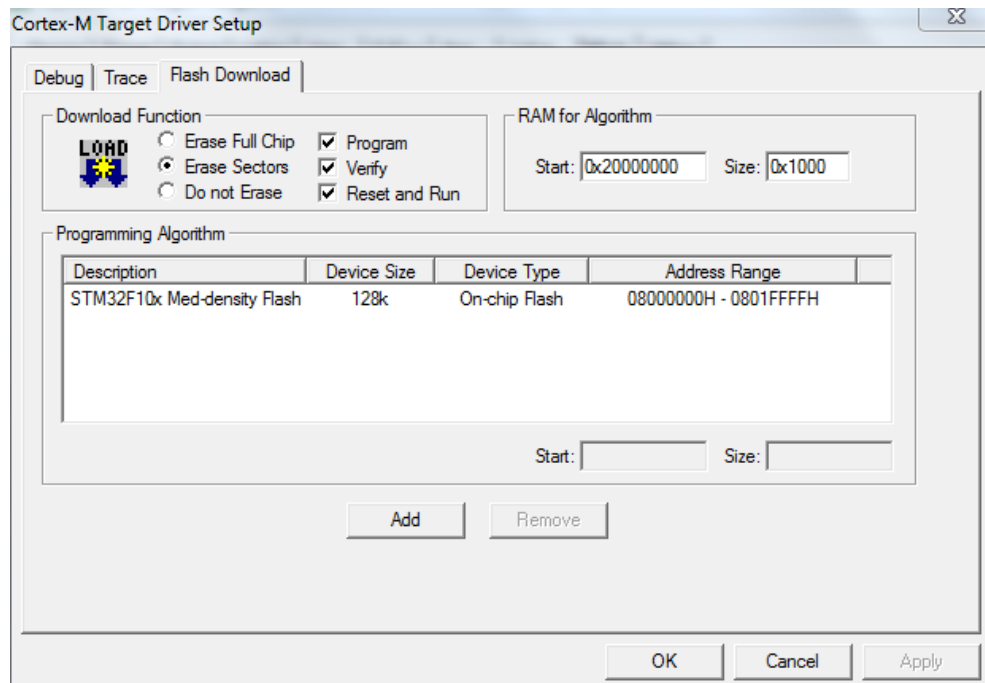


Figura 13. Tab-ul Flash Download

După acest pas, se va apăsa butonul OK. Pentru a încărca codul pe placă, trebuie mai înainte să îl compilăm. Compilarea se realizează prin apăsarea butonului Build locat în partea din stânga sus. După ce procesul de Build a fost realizat cu succes, și placa este conectată la un port USB al calculatorului, se poate descărca fișierul creat în memoria flash a plăcii prin apăsarea butonului LOAD (figura 14).

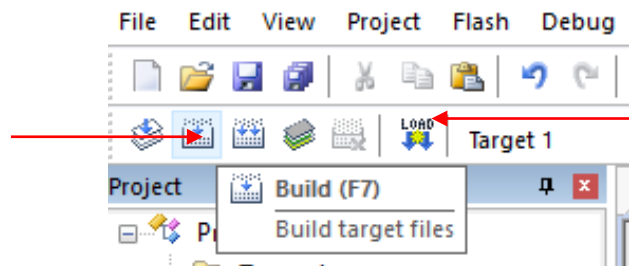


Figura 14. Butoanele Build și Load