

Hệ nhúng (Embedded Systems)

IT4210

Đỗ Công Thuần

Khoa Kỹ thuật máy tính, Trường CNTT&TT

Đại học Bách khoa Hà Nội

Email: thuandc@soict.hust.edu.vn

ONE LOVE. ONE FUTURE.

Giới thiệu môn học

- Tên học phần: **Hệ nhúng**
- Mã học phần: **IT4210 (3-0-1-6)**
- Thời lượng:
 - 16.5 buổi lý thuyết (3 tiết/buổi)
 - 3 buổi thực hành (5 tiết/buổi)
- Yêu cầu kiến thức nền tảng:
 - Kiến trúc máy tính
 - Vi xử lý
 - Lập trình C

Mục tiêu môn học

- Hiểu được kiến trúc tổng quan, đặc điểm và hoạt động của một hệ nhúng
- Biết thiết kế hệ nhúng cơ bản (nguyên lý thiết kế mạch, ...)
- Hiểu được kiến trúc vi điều khiển (Intel, ARM)
- Lập trình vi điều khiển từ cơ bản đến nâng cao với các dòng vi điều khiển phổ biến
- Lập trình với hệ điều hành nhúng

Đánh giá học phần

1. Đánh giá quá trình: **40%**

- Bài tập về nhà
- Chuyên cần
- Các bài thực hành, nhóm **4 SV/nhóm**

2. Đánh giá cuối kỳ: **60%**

- Làm project cuối kỳ, nhóm **4 SV/nhóm**
- Yêu cầu sinh viên tự chọn nhóm và đăng kí đề tài.
Chú ý: danh sách đề tài sẽ được cập nhật sau!

Tài liệu tham khảo

- Textbook/Lecture notes:

- Peter Marwedel, ***Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things***, Springer, 4th edition, 2021.
- Edward A. Lee and Sanjit A. Seshia, ***Introduction to Embedded Systems: A Cyber-Physical Systems Approach***, MIT Press, 2nd edition, 2017.
- Tammy Noergaard, ***Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers***, Elsevier, 2nd edition, 2013.
- Han-Way Huang, Leo Chartrand, ***Microcontroller: An Introduction to Software & Hardware Interfacing***, Cengage Learning, 2004.
- Lectures in Embedded Systems from *Univ. of Cincinnati (EECE 6017C)*, *Univ. of California, Berkeley (EECS 149)*, *Univ. of Pennsylvania (ESE 350)*, *Univ. of Kansas (EECS388)*.
- ...

- Manuals/Handbooks/Internet

- Atmel, Microchip, Texas Instruments, Keil...
- Keil ASM51
- Arduino IDE
- ...

Nội dung học phần

- Chương 1: Giới thiệu về Hệ nhúng
- Chương 2: Thiết kế phần cứng Hệ nhúng
- Chương 3: Lập trình với 8051
- Chương 4: Ghép nối ngoại vi với 8051
- Chương 5: Arduino
- Chương 6: Ghép nối nối tiếp
- Chương 7: Ghép nối với thế giới thực
- **Chương 8: Kiến trúc ARM**
- Chương 9: RTOS và FreeRTOS

Chương 8

Kiến trúc ARM

So sánh 8-bit và 32-bit MCU

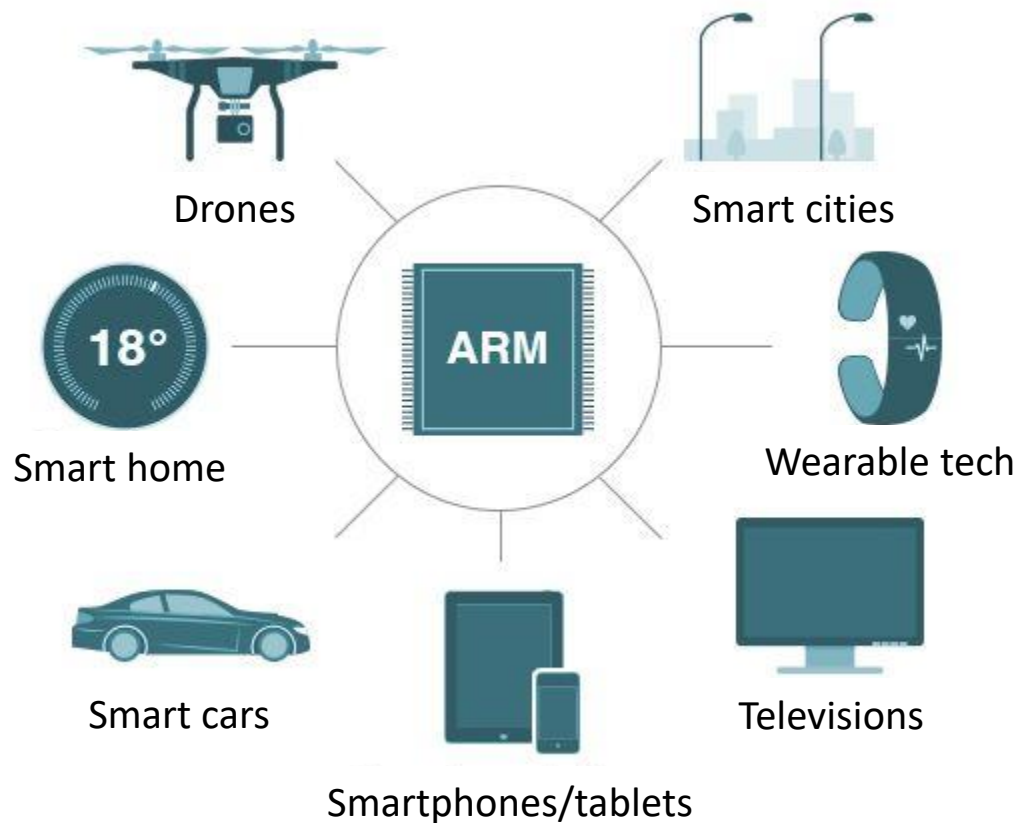
- 8-bit MCU: (8051, PIC, AVR, STM8...)
 - Rẻ tiền, dễ lập trình, tiết kiệm năng lượng.
 - Tốc độ thấp (10-20 MHz), bus dữ liệu nhỏ (8 bit), ít ngoại vi.
 - Phù hợp với các ứng dụng đơn giản, không đòi hỏi tính toán xử lý dữ liệu phức tạp.
- 32-bit MCU:
 - Kiến trúc tập lệnh 32 bit, bus dữ liệu 32 bit → khả năng xử lý dữ liệu vượt trội.
 - Tốc độ cao (100-200 MHz), nhiều ngoại vi tích hợp.
 - Phù hợp ứng dụng cần khối lượng tính toán lớn.

ARM Ltd.

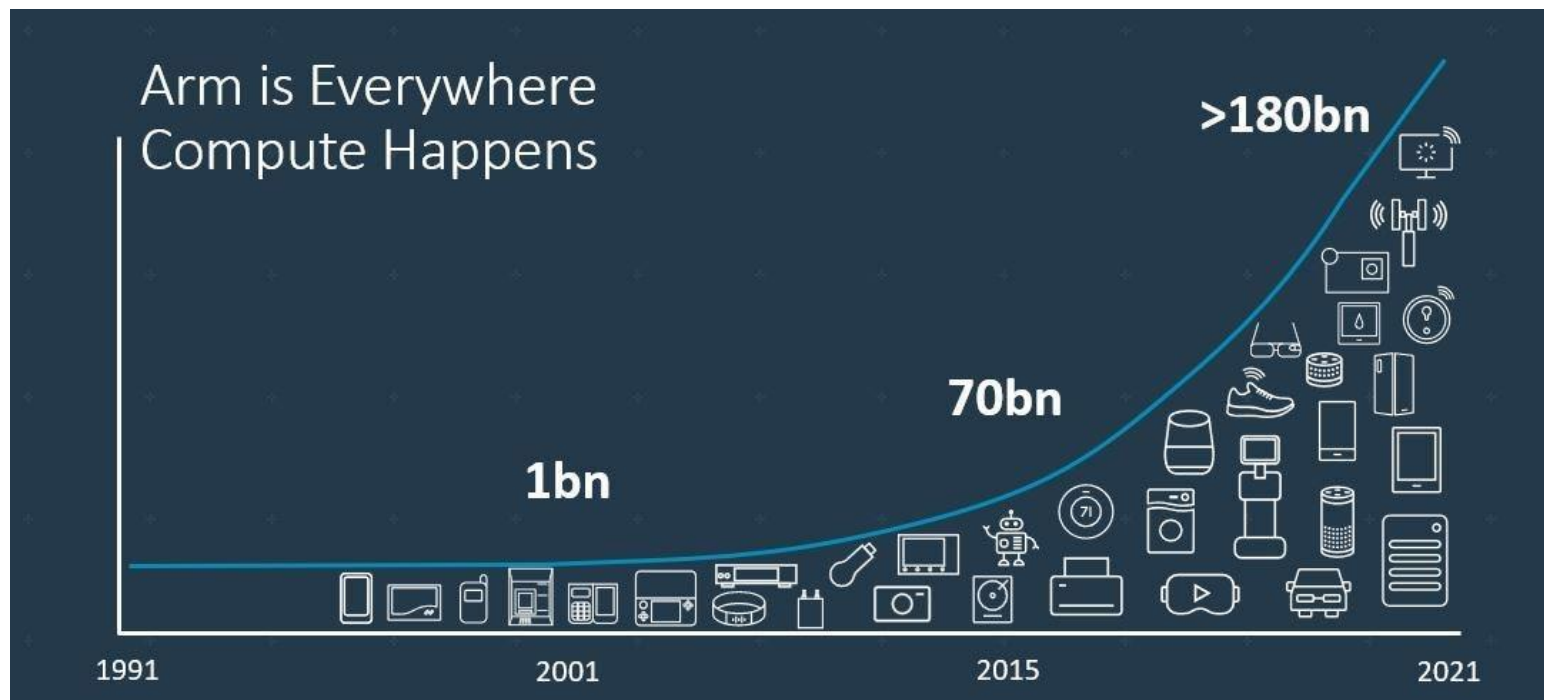
- Thành lập 11/1990
 - Spin-off từ Acorn Computers
- Thiết kế CPU ARM
- Cung cấp bản quyền sử dụng ARM core cho các công ty sản xuất CPU.
- Cung cấp các công cụ hỗ trợ xây dựng hệ thống



ARM Powered Products



Kỷ nguyên của ARM



Lượng chip ARM-based bán ra thị trường

ARM Cortex Processors

- ARM Cortex-**A** family:

Applications processors cho các hệ thống hiệu năng cao (Linux, Android...) với tần số clock > 1 GHz.

- ARM Cortex-**R** family:

Embedded processors cho ứng dụng real-time, điều khiển cần độ tin cậy cao, tần số clock khoảng 200 MHz – 1 GHz.

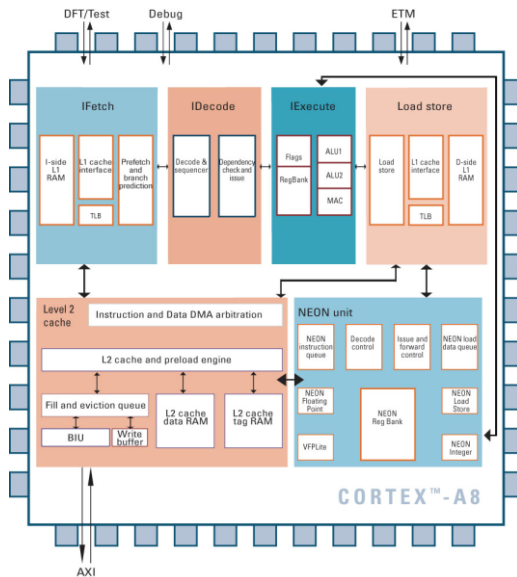
- ARM Cortex-**M** family:

Microcontroller trong các hệ nhúng, giá rẻ, tiết kiệm năng lượng, tần số clock < 200 MHz.

Cortex family

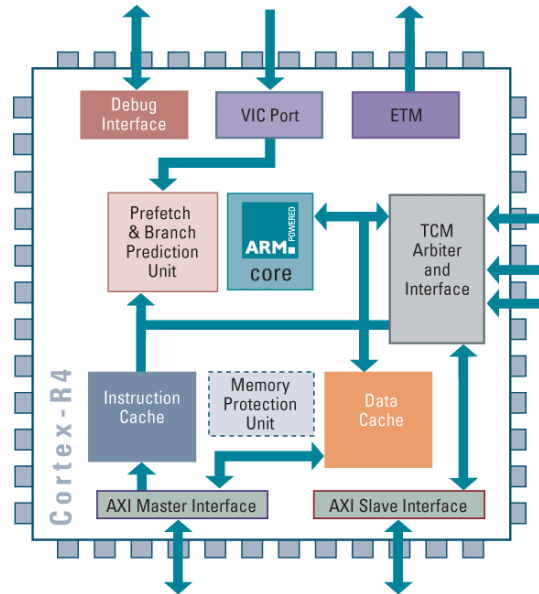
Cortex-A8

- Architecture v7A
- MMU
- AXI
- VFP & NEON support



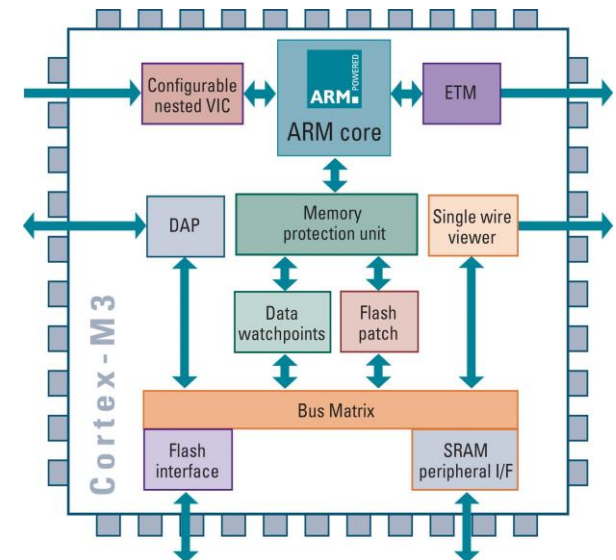
Cortex-R4

- Architecture v7R
- MPU (optional)
- AXI
- Dual Issue



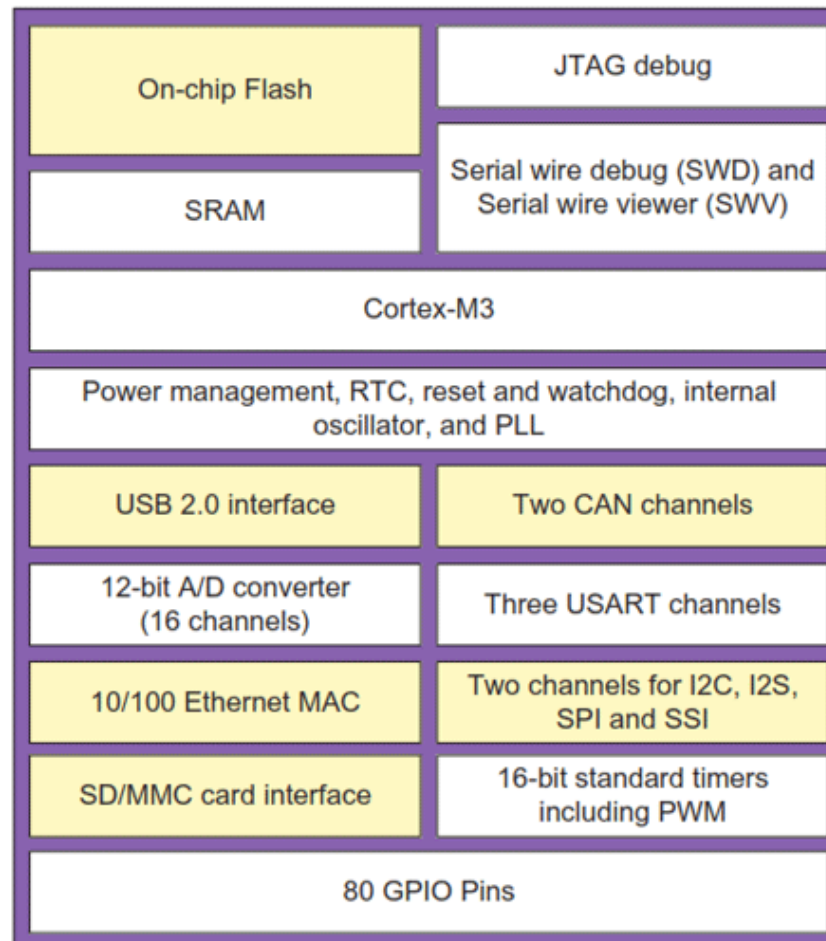
Cortex-M3

- Architecture v7M
- MPU (optional)
- AHB Lite & APB



ARM Cortex-M

- Các tài nguyên thường có trên ARM Cortex-M



VD: Một số chip sử dụng lõi Cortex-M4

- [Analog Devices](#) ADUCM4050
- [Cypress](#) 6200, FM4
- [Infineon](#) [XMC4000](#)
- [Maxim](#) Darwin
- [Microchip \(Atmel\)](#) SAM4C/4E/G5, SAMD5/E5x
- [Nordic](#) nRF52
- [Nuvoton](#) NuMicro M480
- [NXP](#) [LPC4000](#), [LPC4300](#) LPC54000
- [NXP \(Freescale\)](#) Kinetis K, V3, V4
- [Renesas](#) S3, S5, S7, RA4, RA6
- [Silicon Labs \(Energy Micro\)](#) [EFM32 Wonder](#)
- [ST](#) [STM32](#) F3, F4, L4, L4+, G4, WB
- [Texas Instruments](#) LM4F, TM4C, [MSP432](#), CC13x2R, CC1352P, CC26x2R
- [Toshiba](#) TX04

Processors implementing ARM ISAs

- Examples:
 - **Apple M1/M2**: a ARM-based SoC designed by Apple Inc., that implements the [ARMv8.5-A](#) ISA.
 - **Snapdragon 888/888+ 5G**: an ARM-based SoC made by Qualcomm that uses the ARM Cortex X1, A78, A55 cores and implements the ARMv8.2-A ISA.
 - **Exynos 2200**: an ARM-based SoC developed by Samsung Electronics that uses ARM Cortex cores and implements the ARMv8.2-A ISA.



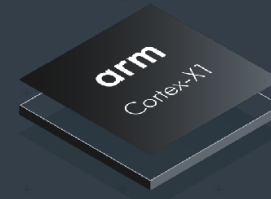
Processors implementing ARM ISAs

Design Philosophies



Arm Cortex-A78

- Area and power efficiency-first design
- Designed for sustained performance within a thermal envelope
- Optimize width and depth for most efficient performance



Arm Cortex-X1

- Performance-first design using Cortex-X Custom program engagement
- Designed for peak performance in shorter bursts
- Pushing the limits of width and depth, while maintaining max frequency capability

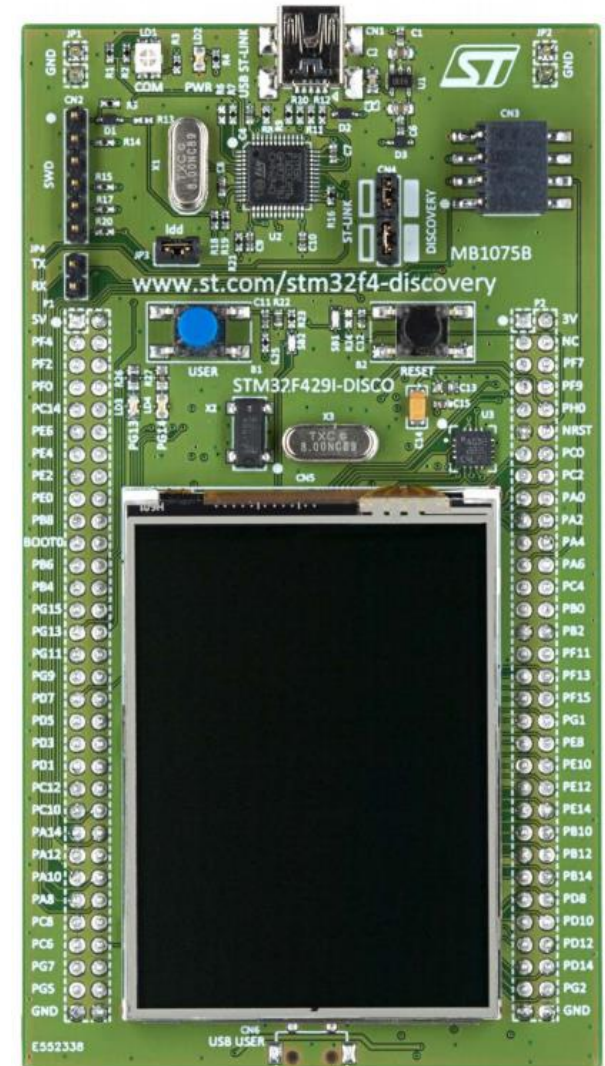
Contrasting the design philosophies of the ARM Cortex-A and Cortex-X series CPUs
(Reference [here](#))

Ví dụ: STM32F429Z

- 180 MHz max CPU (4-26 MHz crystal)
- 2 MB Flash, 256 KB SRAM
- FPU + DSP core
- Built-in Ethernet, USB
- 17 Timers (16/32 bits, 180 MHz)
- 3x12 bits ADCs (3x2.4 MSPS, 24 channels)
- 21 communication interfaces (SPI, I2C, I2S...)
- Camera interface, LCD interface, DRAM interface...
- RTC, CRC, Random generator...
- ...

Kit phát triển STM32F429I-DISC1

- STM32F429ZIT6 ARM Cortex-M4
- 2 MB flash, 256 KB SRAM
- 8MB SDRAM
- On-board ST-Link debugger
- 2.4" QVGA touch LCD
- 6 LEDs (2 user LEDs)
- 3 axis gyros

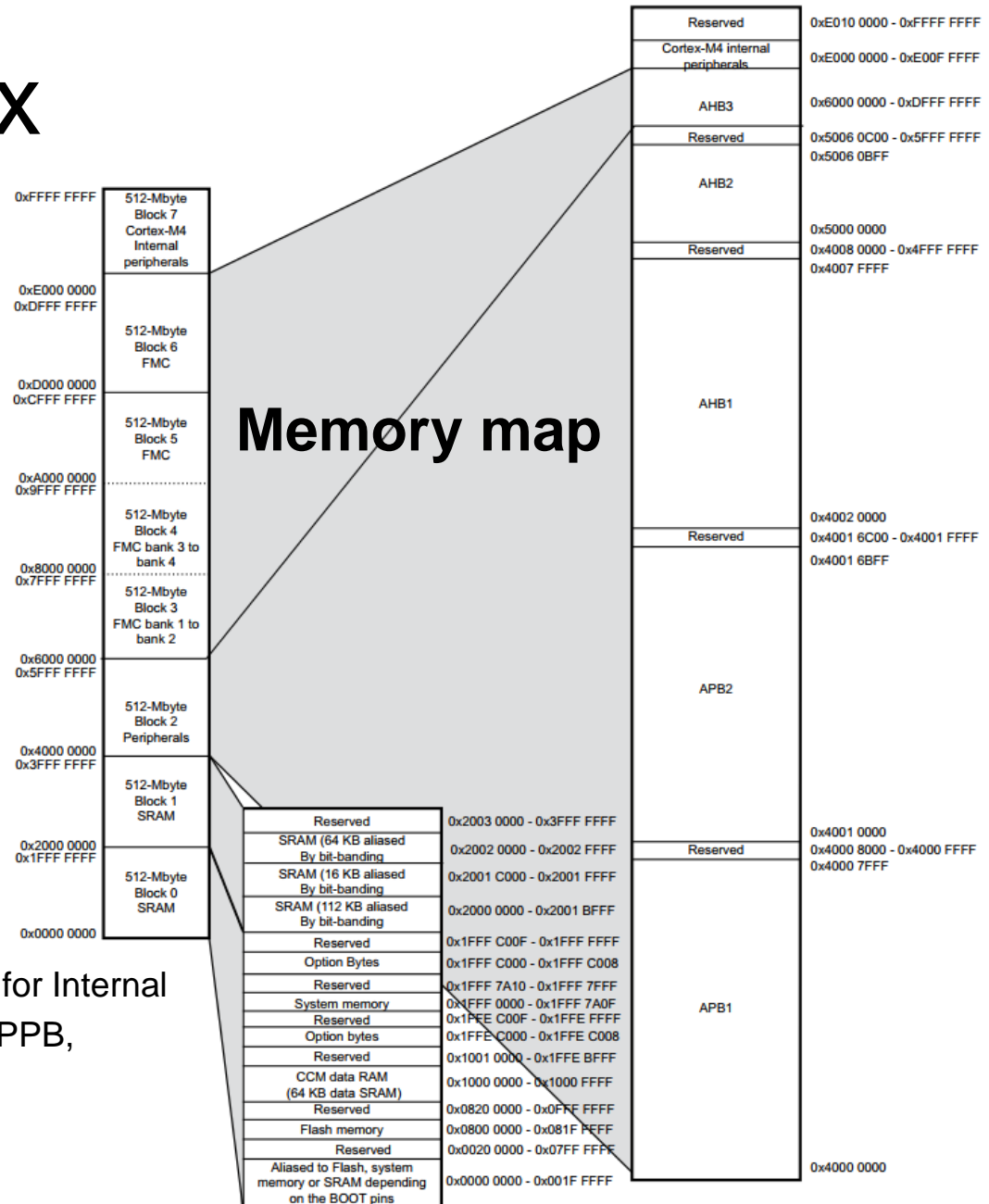


Block diagram



STM32F429xxx

- **Code** [0x00000000 – 0x1FFFFFFF]: for program code; data accesses allowed
- **SRAM** [0x20000000 – 0x3FFFFFFF]: for connecting SRAM, but no limitation of exact memory type; first 1MB is bit addressable
- **Peripherals** [0x40000000 – 0x5FFFFFFF]: for on-chip peripherals; first 1MB is bit addressable
- **RAM** [0x60000000 – 0x9FFFFFFF]: for off-chip memories; used for both code and data
- **Devices** [0xA0000000 – 0xDFFFFFFF]: for other (off-chip) peripherals
- **System** [0xE0000000 – 0xFFFFFFFF]: for Internal Private Peripheral Bus (PPB), External PPB, Vender-specific area



Một số vấn đề

- Clock rate rất cao → cần mô hình lập trình/xây dựng ứng dụng tận dụng hiệu năng của CPU
 - Polling trong main loop
 - Thực thi tác vụ trong hàm xử lý ngắt
 - Sử dụng hệ điều hành, tạo thread cho từng tác vụ
- Rất nhiều ngoại vi tích hợp trên chip → cần công cụ trợ giúp xây dựng hệ thống.
- Rất nhiều dòng chip tùy theo nhà sản xuất, nhưng chung kiến trúc tập lệnh ARM → cần tập thư viện mức thấp dùng chung

Môi trường lập trình

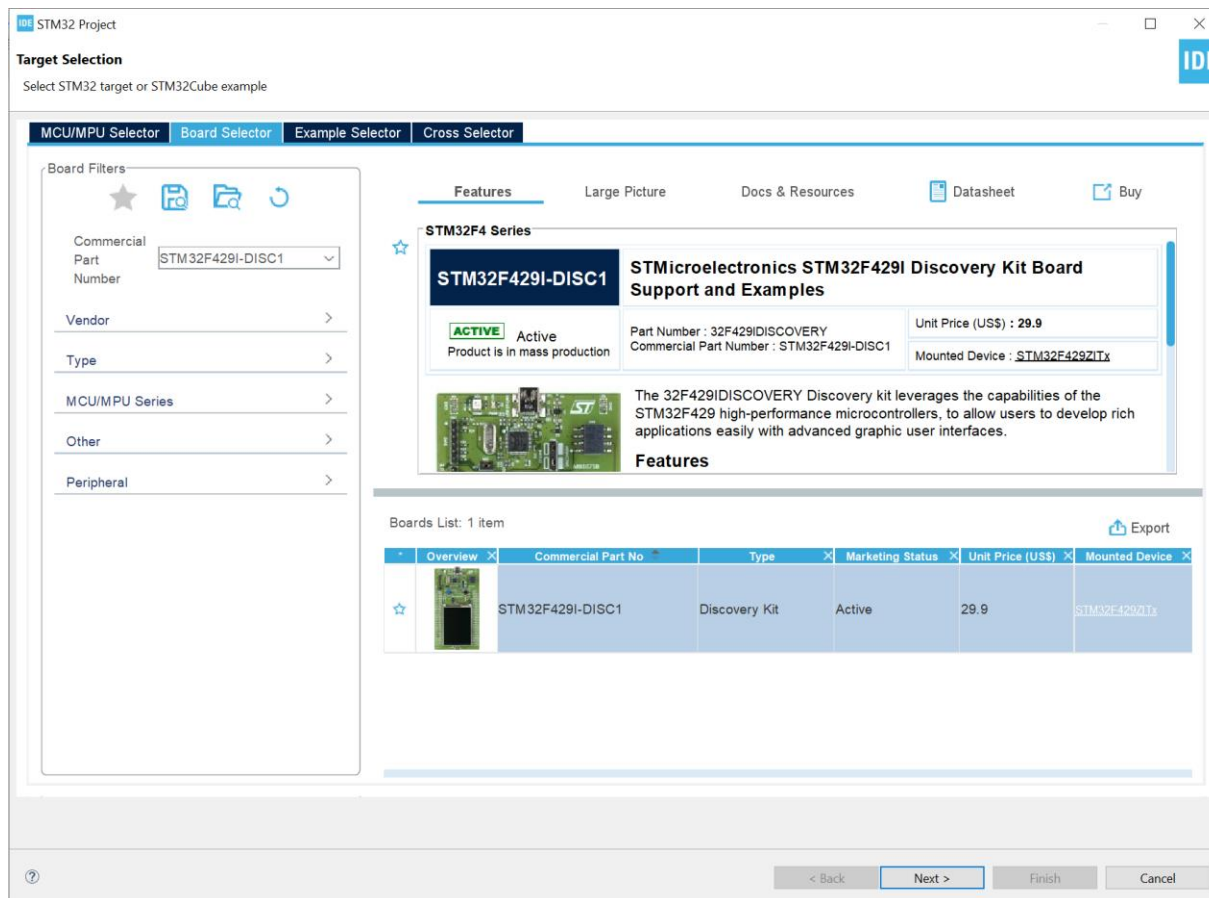
- Dành cho MCU của ST Microelectronics
- STM32 CubeIDE: IDE + compiler + debugger
- STM32CubeF4: driver + library
- ST-LINK009: USB driver

Ví dụ 1: Hello World

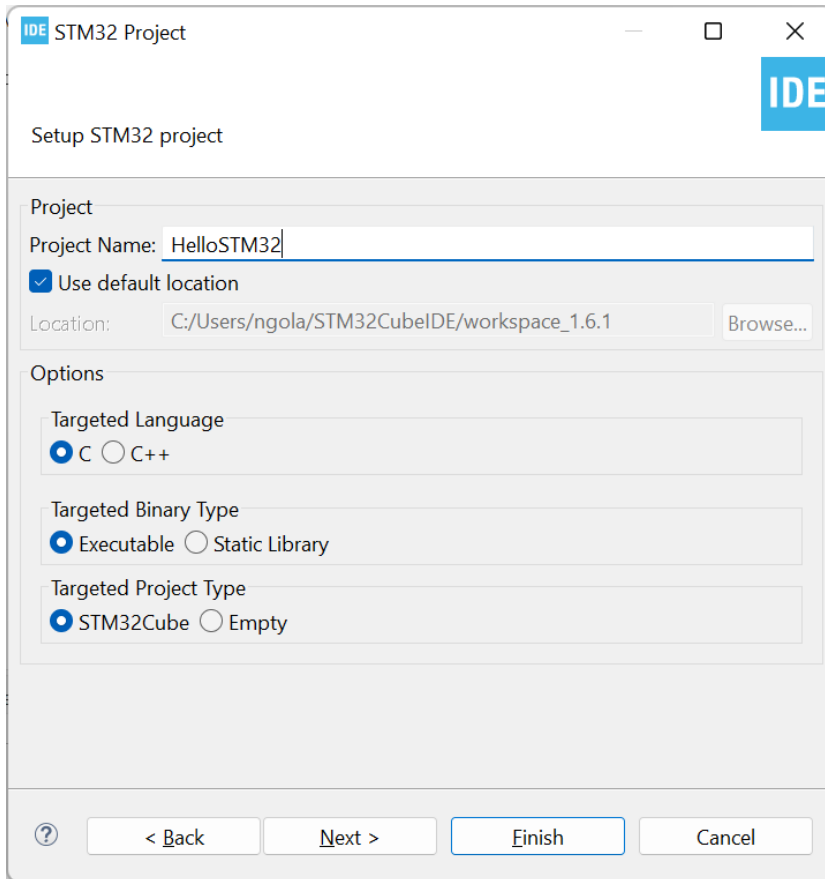
- Mục tiêu:
 - Tạo project lập trình cho STM32F4
 - Cấu hình GPIO và bật 1 đèn LED
- Đọc file manual của kit, tìm hiểu các ngoại vi sau nối vào chân nào của CPU
 - Các đèn LED3, LED4
 - Nút bấm USER_BUTTON

Tạo project với STM32CubeIDE

- Chọn New → STM32 Project → Board Selector
 - Nhập STM32F429I-DISC1



Tạo project



IDE STM32 Project

Setup STM32 project

Project

Project Name:

☒ Use default location

Location: [Browse...](#)

Options

Targeted Language

☒ C ☐ C++

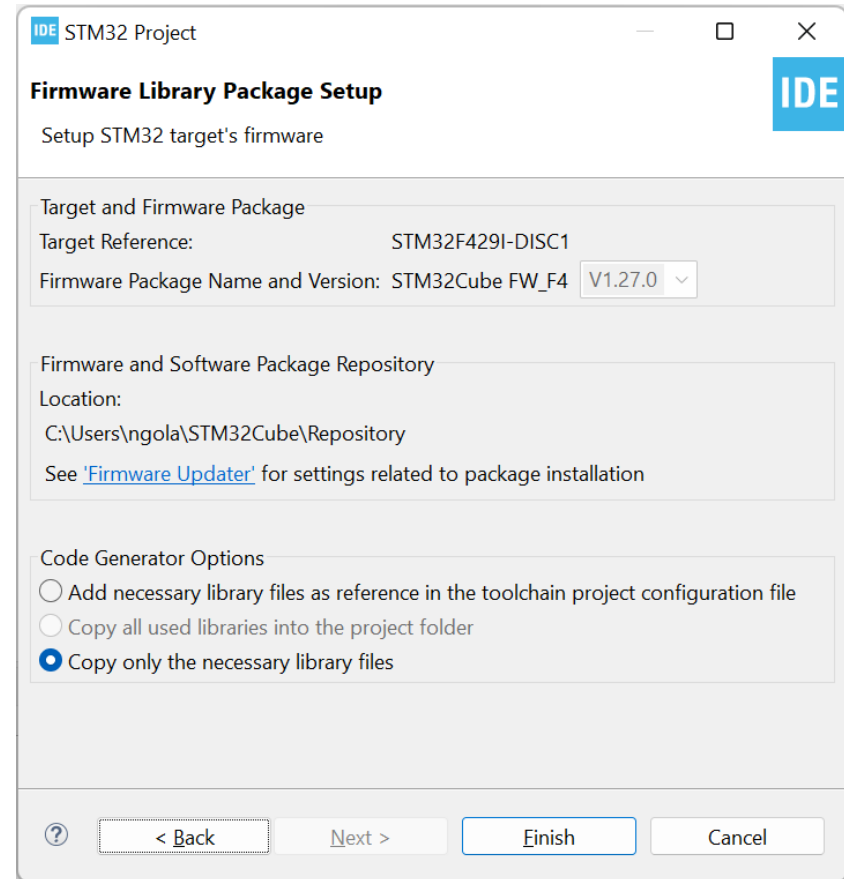
Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)



IDE STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference: STM32F429I-DISC1

Firmware Package Name and Version: STM32Cube FW_F4

Firmware and Software Package Repository

Location: C:\Users\ngola\STM32Cube\Repository

See ['Firmware Updater'](#) for settings related to package installation

Code Generator Options

☐ Add necessary library files as reference in the toolchain project configuration file

☐ Copy all used libraries into the project folder

☒ Copy only the necessary library files

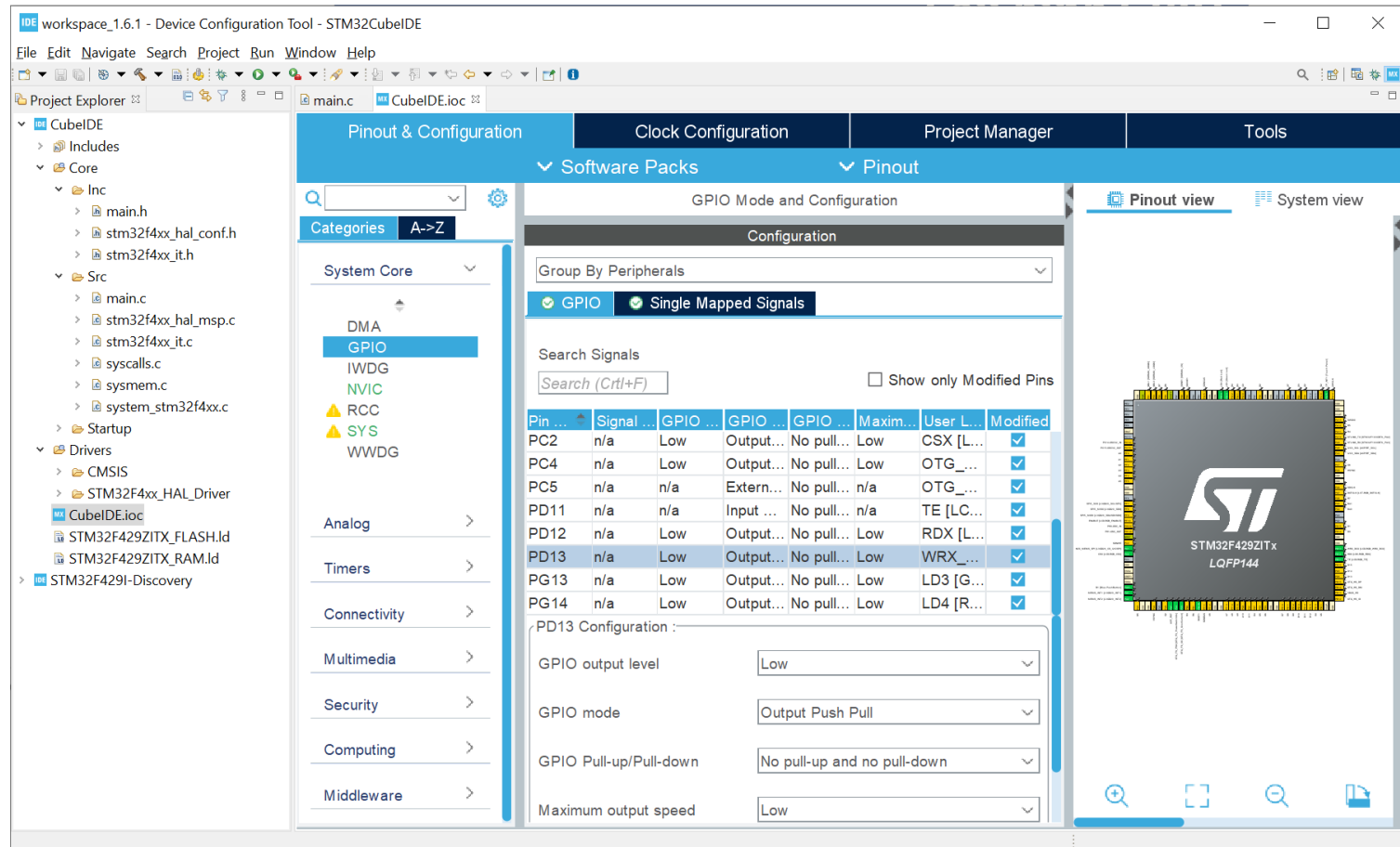
[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Cấu hình chân chip

- Cần cấu hình chân chip (PG13 và PG14) để điều khiển 2 đèn LED3 và LED4.
- Chọn System Core, mục GPIO:
 - Cấu hình chân PG13, PG14 là GPIO_Output
- Bấm Ctrl+S → lưu thành file *.ioc và tự sinh mã nguồn

Cấu hình GPIO

- Output, push-pull, Low

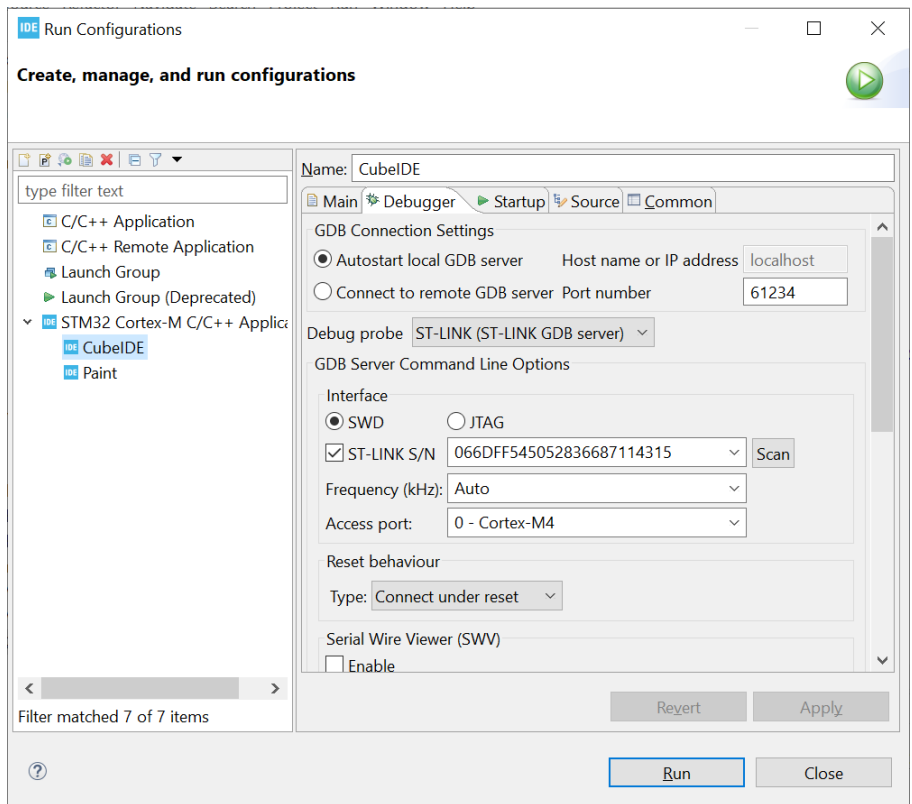
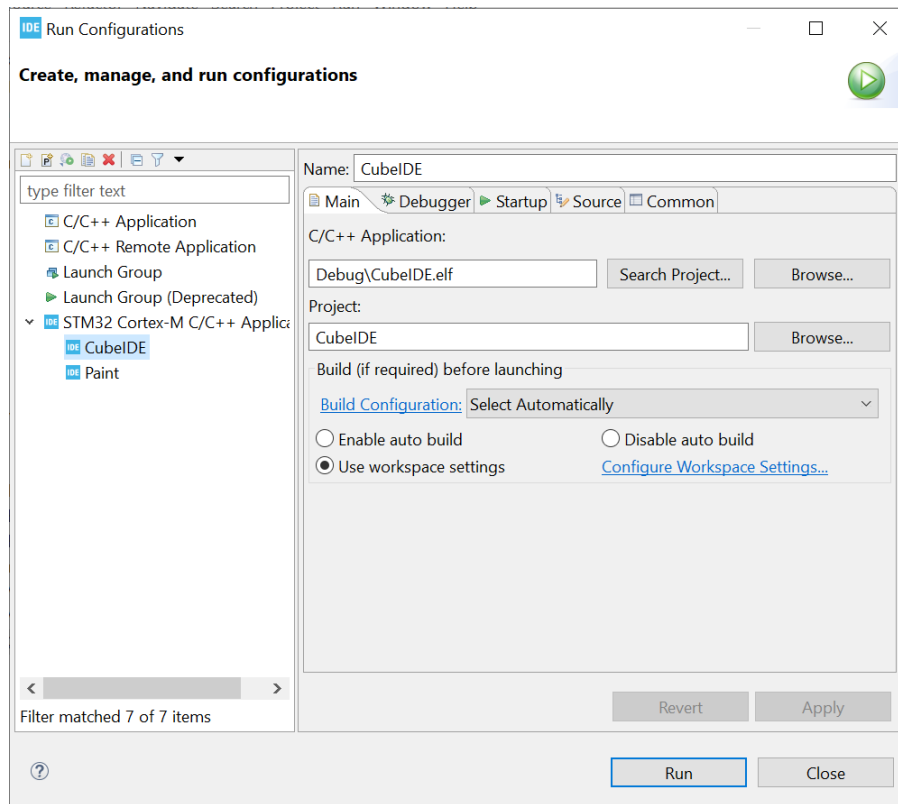


Thêm mã nguồn cho hàm main

```
/* Reset of all peripherals, Initializes the Flash  
interface and the Systick. */  
HAL_Init();  
/* Initialize all configured peripherals */  
MX_GPIO_Init();  
/* Write 1 to GPIO → turn LED ON */  
HAL_GPIO_WritePin(LD4_GPIO_Port, LD4_Pin, GPIO_PIN_SET);
```

Tải và chạy chương trình

- Run → Run as STM32 CortexM C/C++ Application
- Cắm mạch vào cổng USB của máy tính, chọn Debugger là SWD, ST-LINK, bấm Scan để tìm S/N



STM32 HAL library

- Cung cấp tập API và driver giao tiếp, điều khiển phần cứng có trên chip STM32
- Cross-device support: có thể dùng chung giữa các chip STM32 khác nhau → giảm chi phí phát triển phần mềm khi nâng cấp thay đổi phần cứng
- Tham khảo chi tiết: ***Description of STM32F4xx HAL drivers***
- Tìm hiểu các hàm HAL_GPIOxxx

Ví dụ 2: nháy LED

- Mục tiêu
 - Giống VD 1
 - Cấu hình clock để đặt chính xác system clock
 - Sử dụng hàm delay để bật/tắt LED theo chu kỳ

Cấu hình clock

- Mở file .ioc, đặt chế độ hoạt động của HSE

The screenshot shows the CubeIDE Pinout & Configuration window. The 'Clock Configuration' tab is selected. The 'RCC Mode and Configuration' section shows the following settings:

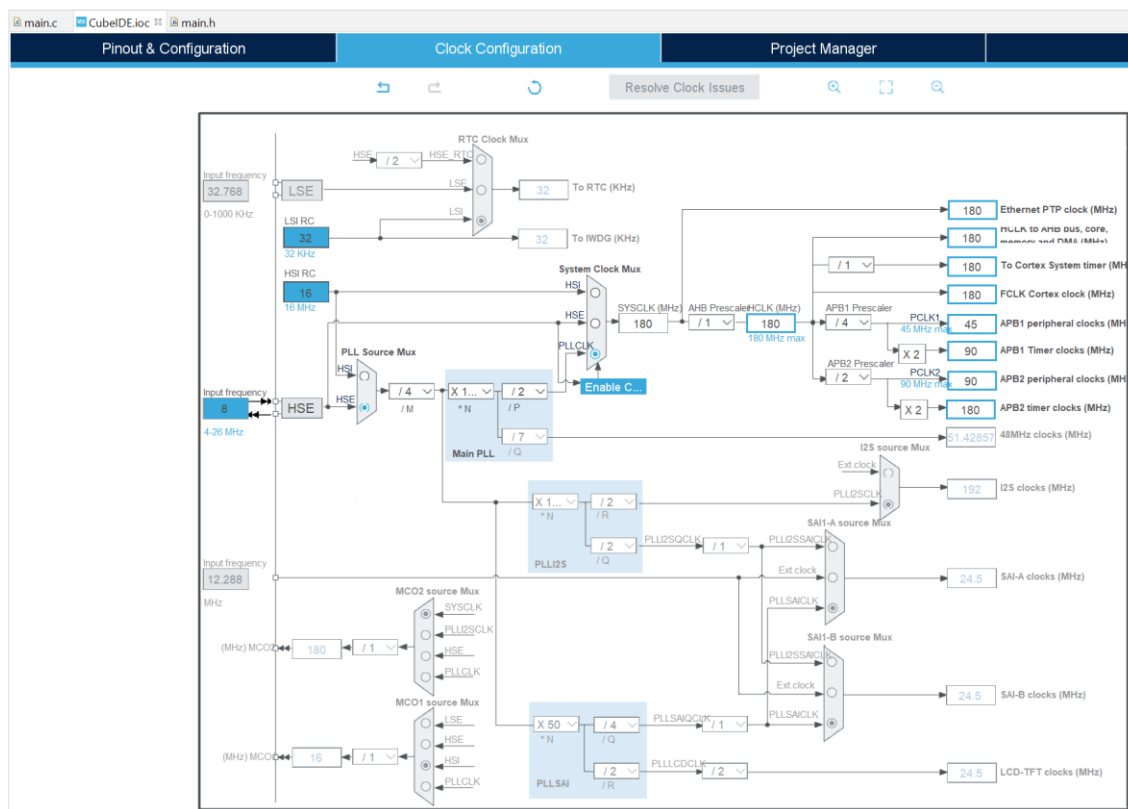
- High Speed Clock (HSE): Crystal/Ceramic Resonator
- Low Speed Clock (LSE): Disable
- Master Clock Output 1: ☐
- Master Clock Output 2: ☐
- Audio Clock Input (I2S_CKIN): ☐

The 'Configuration' section shows a table of pin configurations:

Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-u...	Maximum o...	User
PH0/OSC_IN	RCC_OSC_IN	n/a	n/a	n/a	n/a	PH0-O
PH1/OSC_...	RCC_OSC_...	n/a	n/a	n/a	n/a	PH1-O

Cấu hình clock

- Chọn System Clock Mux là PLLCLK
- Đặt HCLK là 180 → các thông số sẽ được tự sinh
- Bấm Ctrl + S để tự sinh code và cập nhật vào project



Mã nguồn main loop

```
while (1)
{
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
}
```

Ví dụ: Nháy lần lượt 2 LED

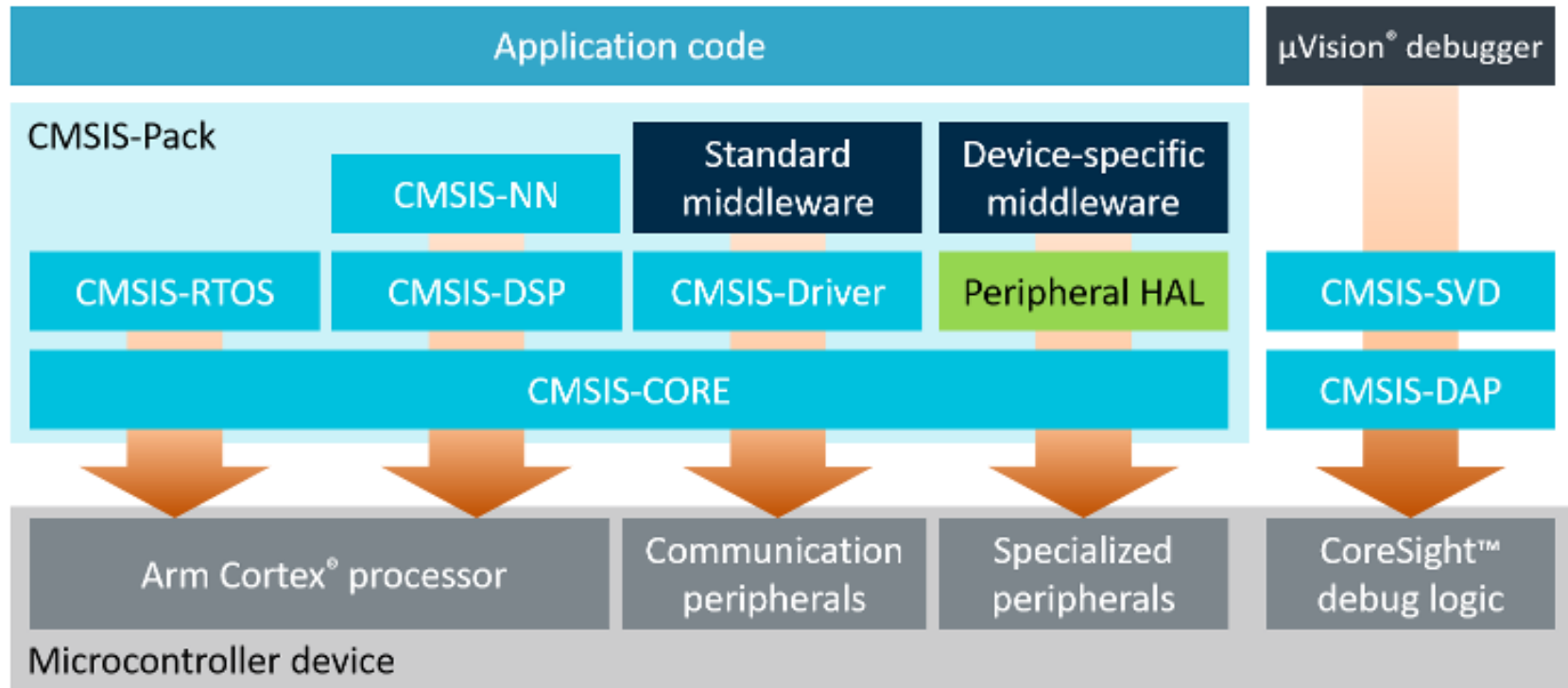
Ví dụ 3: Interrupt

- Tạo project cho board STM32F429I-DISC1 (để có các chân vào ra được set up sẵn)
- Cấu hình clock
- Cấu hình ngắt cho Push button
- Cấu hình Timer6
- Lập trình các hàm xử lý ngắt để bật/tắt LED3 và nhấp nháy LED4
 - void EXTI0_IRQHandler(void)
 - void TIM6_DAC_IRQHandler(void)

ARM CMSIS v5

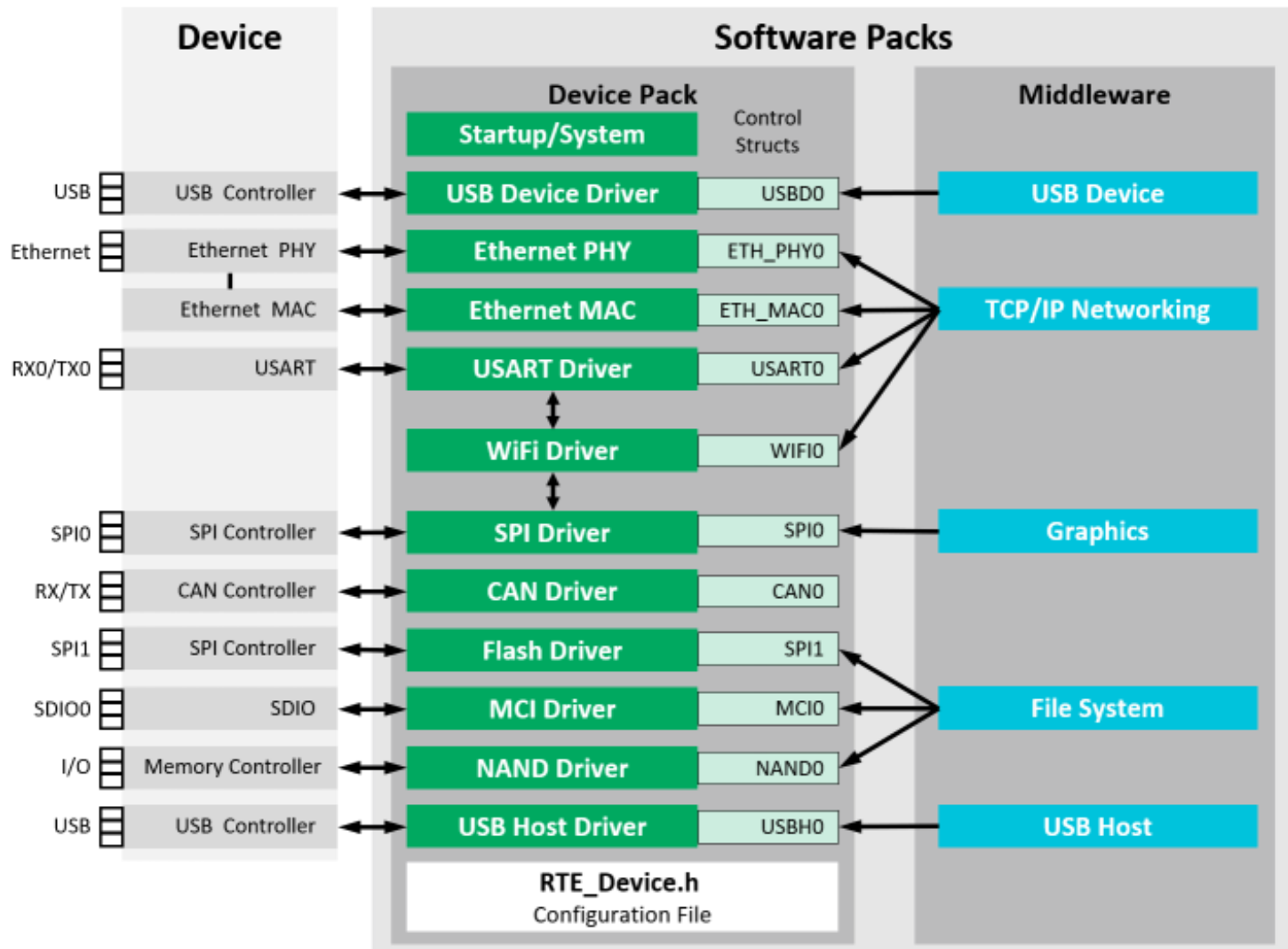
- ARM CPU/MCU
 - Năng lực tính toán lớn
 - Nhiều tài nguyên: bộ nhớ lớn, nhiều ngoại vi
 - Thích hợp cho các ứng dụng hiệu năng cao
 - Trở ngại:
 - Lập trình phức tạp hơn (nhiều) so với các CPU/MCU 8 bit
 - Yêu cầu code có khả năng tương thích và khả năng chuyển tốt hơn
- ➔ Cần các API, library

Mô hình lập trình với ARM Cortex



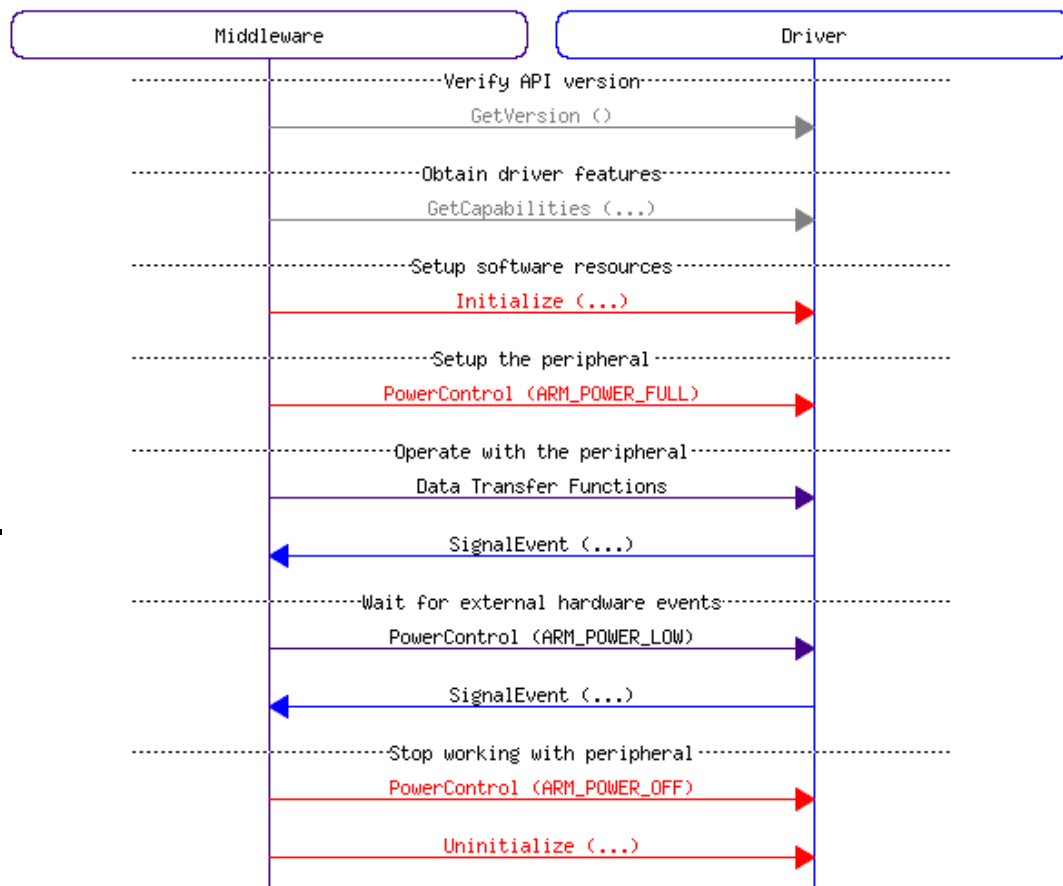
- CMSIS: vendor independent components provided by ARM
- Peripheral HAL: provided by CPU vendor
- Middleware: from 3rd party software vendors

CMSIS driver



Các hàm mặc định

- `GetVersion()`.
- `GetCapabilities()`.
- `Initialize()`.
- `SignalEvent()`.
- `PowerControl()`
 - `ARM_POWER_FULL`.
 - `ARM_POWER_LOW`.
 - `ARM_POWER_OFF`.
- `Uninitialize()`.
- `Control()`.



VD driver cho SPI

```
typedef struct _ARM_DRIVER_SPI {  
    ARM_DRIVER_VERSION (*GetVersion) (void);  
    ARM_SPI_CAPABILITIES (*GetCapabilities) (void);  
    int32_t (*Initialize) (ARM_SPI_SignalEvent_t cb_event);  
    int32_t (*Uninitialize) (void);  
    int32_t (*PowerControl) (ARM_POWER_STATE state);  
    int32_t (*Send) (const void *data, uint32_t num);  
    int32_t (*Receive) (void *data, uint32_t num);  
    int32_t (*Transfer) (const void *data_out,  
                        void *data_in, uint32_t num);  
    uint32_t (*GetDataCount) (void);  
    int32_t (*Control) (uint32_t control, uint32_t arg);  
    ARM_SPI_STATUS (*GetStatus) (void);  
} const ARM_DRIVER_SPI;
```

```
ARM_DRIVER_SPI Driver_SPI1; //access functions for SPI1 interface
```

CMSIS DSP lib

- Basic math functions
- Fast math functions
- Complex math functions
- Filtering functions
- Matrix functions
- Transform functions
- Motor control functions
- Statistical functions
- Support functions
- Interpolation functions
- Support Vector Machine functions (SVM)
- Bayes classifier functions
- Distance functions

Bài tập

- Tạo project với BSP example
- Lập trình hiển thị giá trị lấy từ cảm biến gyroscope theo 3 trục x, y, z lên màn hình