**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN-ĐIỆN TỬ**

**----o0o----**

**Logo, company name

Description automatically generated**

**BÁO CÁO**

**THỰC TẬP NGOÀI TRƯỜNG**

**Sinh viên thực hiện:** Dương Quốc Hoàng

**MSSV:** 1711377

**Đơn vị thực tập:**

FPT Software

**Tp.Hồ Chí Minh, ngày 26 tháng 12 năm 2022**

NHẬN XÉT CỦA CÔNG TY

Sinh viên thực tập: Dương Quốc Hoàng

MSSV: 1711377

Trường: Đại học Bách Khoa TP. HCM

Địa điểm thực tập: FPT Software

Thời gian thực tập: 3 tháng (từ 18/08/2022 đến 18/11/2022)

# 1. Lời nhận xét của công ty

………………….. …………………... ………………….. ………………….. …………………..

…………………..

…………………..

# 2. Đánh giá thực tập (…/10)

Những nội dung trình bày trong báo cáo thực tập này là phù hợp với tình hình hiện tại của đơn vị. Chúng tôi đánh giá cao kết quả thực tập của em.

Kết quả thực tập: … /10

TP.HCM, ngày 26 tháng 12 năm 2022

Người hướng dẫn Đại diện công ty

Lời đầu tiên, em xin cảm ơn các thầy cô đã truyền đạt các kiến thức cho em để từ đó em có một nền tảng kiếm thức đi làm nó dễ dàng và thuận tiện hơn.

Lời cảm ơn

Tiếp đên, em xin cảm ơn chân thành đến Công ty FPT Software đã tạo điều kiện cho em thực tập để em có thêm các kỹ năng ngoài thực tế trong công việc sau này.

Em cũng xin cảm ơn các bạn trong quá trình thực tập đã hỗ trợ em và cùng nhau tạo một môi trường học tập tốt để cùng nhau cố gắng và giúp đỡ lẫn nhau.

Bản thân em trong quá trình thực tập và làm báo cáo thực tập, vì sự thiếu sót về cả kiến thức lẫn kinh nghiệm, rất khó tránh khỏi những sai sót. Rất mong nhận được sự góp ý và chia sẻ từ thầy cô và quý công ty để chúng em có thể ngày một cải thiện hơn nữa.

Em xin chân thành cảm ơn!

Mục lục

[Chương 1: Giới thiệu 1](#_Toc122960322)

[I. Tổng quan về đơn vị thực tập. 1](#_Toc122960323)

[II. Nhiệm vụ thực tập. 2](#_Toc122960324)

[1. Tìm hiểu về cách hoạt động của một hệ thống nhúng. 2](#_Toc122960325)

[2. Tìm hiểu về cấu tạo một vi điều khiển. 2](#_Toc122960326)

[3. Tìm hiểu về lập trình một vi điều khiển. 2](#_Toc122960327)

[III. Lịnh trình thực tập. 2](#_Toc122960328)

[Chương 2: Nội dung thực tập 3](#_Toc122960329)

[I. Tìm hiểu và cải thiện về lập trình C. 3](#_Toc122960330)

[1. Các bước để build một chương trình C. 3](#_Toc122960331)

[2. Tìm hiểu về các thành phần trong chương trình C. 4](#_Toc122960332)

[3. Quản lý bộ nhớ. 5](#_Toc122960333)

[4. Variable. 6](#_Toc122960334)

[5. Data structures and Algorithms. 7](#_Toc122960335)

[II. Tìm hiểu về Hệ thống nhúng. 10](#_Toc122960336)

[1. Giới thiệu. 10](#_Toc122960337)

[2. Vi điều khiển và vi xử lý. 11](#_Toc122960338)

[3. Các kiến trúc của vi điều khiển. 12](#_Toc122960339)

[4. Software Flow. 14](#_Toc122960340)

[III. Các công cụ hỗ trợ. 16](#_Toc122960341)

[1. Visual Studio. 16](#_Toc122960342)

[2. Keil uVision5 21](#_Toc122960343)

[3. Board KL46Z 24](#_Toc122960344)

[Chương 3: Kết luận 28](#_Toc122960345)

[Tài liệu tham khảo 29](#_Toc122960346)

# Chương 1: Giới thiệu

## Tổng quan về đơn vị thực tập.

FPT Software thành lập năm 1999, là công ty thành viên của FPT, Tập đoàn Công nghệ hàng đầu của Việt Nam. Sau 17 năm thành lập FPT Software hiện đang là công ty phần mềm lớn nhất của Việt Nam và đứng trong Top 100 Nhà cung cấp dịch vụ Outsourcing toàn cầu do International Association of Outsourcing Professionals (IAOP) đánh giá.

Là công ty chuyên xuất khẩu dịch vụ phần mềm, FPT Software cung cấp các dịch vụ phát triển phần mềm và bảo trì, triển khai ERP, QA, chuyển đổi ứng dụng, hệ thống nhúng, điện toán di động, điện toán đám mây… trong nhiều lĩnh vực như: Tài chính ngân hàng, Viễn thông, Y tế, Chế tạo, Công nghiệp xe hơi, Dịch vụ công… Hiện FPT Software đang tập trung nghiên cứu và phát triển dịch vụ CNTT dựa trên những nền tảng công nghệ mới như IoT, S.M.A.C, cho các lĩnh vực sản xuất máy bay, sản xuất ô tô, ngân hàng, truyền hình vệ tinh, viễn thông trên phạm vi toàn cầu. Công ty hiện đã và đang cung cấp dịch vụ cho khoảng 450 khách hàng là các tập đoàn lớn trên thế giới, trong đó có 43 khách hàng nằm trong danh sách Fortune Global 500.

Từ 13 lập trình viên với giấc mơ đưa trí tuệ Việt Nam đi khắp năm châu, đội quân FSOFT ở tuổi 18 với hơn 10.000 người, doanh thu hơn 230 triệu USD, tiếp tục biến giấc mơ biến Việt Nam trở thành điểm sáng về CNTT trên bản đồ thế giới thành hiện thực với mục tiêu đạt 1 tỉ USD doanh số năm 2020 và 30.000 người.

Sau 17 năm phát triển thị trường toàn cầu, đến nay FPT Software đã có 23 văn phòng tại 14 quốc gia bao gồm Việt Nam, Nhật Bản, Singapore, Mỹ, Pháp, Đức, Slovakia, Malaysia, Úc, Trung Quốc, Hàn Quốc, Myanmar, Philippines, Thái Lan.

FPT Software Academy là học viện công nghệ trực thuộc FPT Software dành cho học sinh, sinh viên, người đi làm có định hướng theo đuổi ngành Công nghệ.

Chúng tôi cung cấp các chương trình đào tạo định hướng ngành khác nhau về phát triển phần mềm, kiểm thử phần mềm để giúp học viên:

* Nâng cao năng lực chuyên môn.
* Có được kinh nghiệm thực tiễn.
* Cải thiện ngoại ngữ và kỹ năng mềm.

Các lĩnh vực hoạt động của công ty: Nhà máy thông minh, Nền tảng số, RPA, AI, IoT, Enterprise Mobility, Phát triển sản phẩm số, Đám mây, AR/VR, Ứng dụng doanh nghiệp, Dịch vụ ứng dụng, Dịch vụ được quản lý,..

## Nhiệm vụ thực tập.

### Tìm hiểu về cách hoạt động của một hệ thống nhúng.

### Tìm hiểu về cấu tạo một vi điều khiển.

### Tìm hiểu về lập trình một vi điều khiển.

## Lịnh trình thực tập.

|  |  |
| --- | --- |
| **Thời gian** | **Nội dung thực tập** |
| 18/08 – 18/09 | Làm quen với lập trình C |
| 18/09 – 24/09 | Làm một số bài Mock về lập trình C |
| 24/09 – 08/10 | Tìm hiểu về hệ thống nhúng |
| 08/10 – 08/11 | Lập trình module trên board MKL46 |
| 08/11 – 18/1 | Làm các bài Mock về module |

# Chương 2: Nội dung thực tập

## Tìm hiểu và cải thiện về lập trình C.

### Các bước để build một chương trình C.

Diagram

Description automatically generated

Quá trình build chương trình trải qua 4 bước:

Bước 1: Preprocessing – bộ tiền xử lý thay thế giá trị của các directive: #define, #include, #if, … trong file mã nguồn .h và .c.

Bước 2: Complication- trình biên dịch chuyển mã nguồn thành mã assembly.

Bước 3: Assembly – Chuyển từ mã assembly sang mã nhị phân (mã máy – machine code).

Bước 4: Linking – tiến hành liên kết các module và các thư viện liên kết thành file thực thi.

### Tìm hiểu về các thành phần trong chương trình C.

* Các cách khai báo biến và các loại dữ liệu.
* Toán tử số học (+,-,\*,/,%,++,--).
* Toán tử quan hệ (==, !=, >, <, >=, <=).
* Toán tử logic ( &&, ||, !).
* Toán tử bitwise ( &, |, ^, ~, <<, >>).
* Toán tử gán (=, +=, -=, \*=, /=, %=).
* Các câu lệnh điều kiện (if… else…., switch … case…).
* Các vòng lặp (for, while, do…while).
* Các cú pháp khác Return, goto, break, continue, exit.

### Quản lý bộ nhớ.



Bố cục của một chương trình đang chạy:

1. Text Segment: hay còn được gọi là đoạn mã (code), là một trong các phần của chương trình trong tệp đối tượng hoặc trong bộ nhớ chưa các lệnh thực thi, Được đặt bên dưới dùng heap hoặc stack để ngăn heap và tràn stack đè lên nó.
2. Initialized data: là vùng chưa các biến toàn cục và biến tĩnh do lập trình viên khởi tạo.
3. Uninitialized data(bss): Được khởi tạo bằng 0 trước khi chương trình bắt đầu thực thi.
4. Stack: liền kề với vùng heap và được lưu trữ theo hướng ngược lại. khi con trỏ Stack gặp con trỏ heap thì bộ nhớ đã hêt. Là một kiến trúc LIFO (last in first out) .
5. Heap: là nơi cấp phát bộ nhớ động được lưu trữ. Điểm bắt đầu của bộ nhớ là điểm kết thúc của BSS và được lưu trữ lớn hơn từ địa chỉ này.

### Variable.

1. Local variable: Khai báo trong một function( hàm) và chỉ được sử dụng trong hàm đó.
2. Formal parameters: Được khai báo khi định nghĩa function như parameters, hoạt động giống như local variable.
3. Global variable: Khai báo ngoài function và được sử dụng suốt quá trình thực thi chương trình.
4. Static variable: Biến tĩnh, được tạo ra bên trong một function, có khả năng lưu giữ giá trị của nó cho dù thoát khỏi hàm đó.
5. Extern variable: Nó là một biến, hàm cùng tên đã được định nghĩa ở bên ngoài. Nó chỉ khai báo chứ không định nghĩa. Biến phải được khai báo ở cấp độ cao nhất (toàn cục) và có thể nằm trong một file khác.
6. Register variable: là ta yêu cầu trình biên dịch ưu tiên dành vùng register tính toán để chưa biến đó. Tác dụng là nhằm làm tăng hiệu suât.
7. Volatile variable: Volatile có nghĩa là không dự đoán được. Một biến sử dụng với Volatile qualifier có nghĩa là nó nói với compiler rằng biến này có thể thay đổi ở bất kỳ chỗ nào.

### Data structures and Algorithms.

1. Độ phức tạp của thuật toán:

* Trình tự câu lệnh đơn giản: s1, s2, …, sk -> độ phức tạp là O(1)
* Vòng lặp đơn giản: for(int i = 0; i < n; i++){s;} (s có O(1)) 🡪 Độ phức tạp là n.O(1) hay O(n)
* Vòng lặp không tuyến tính: while(h <= n){s;h = 2\* h} 🡪 Độ phức tạp là O(log(n))

1. Linked list:

Danh sách liên kết đơn là một tập hợp các Node được phân bố động, được sắp xếp theo cách sao cho mỗi Node chứa một giá trị (Data) và một con trỏ (Next). Con trỏ sẽ trỏ đến phần tử kế tiếp của danh sách liên kết đó. Nếu con trỏ mà trỏ tới NULL, nghĩa là đó là phần tử cuối cùng của linked list.

Graphical user interface, diagram

Description automatically generated

Ví dụ về một node

Diagram

Description automatically generated

Ví dụ về một linked list

1. Binary tree:

Cây nhị phân là một cấu trúc dữ liệu đặc biệt được sử dụng cho mục đích lưu trữ dữ liệu. Một cây nhị phân có một điều kiện đặc biệt là mỗi nút có thể có tối đa hai nút con. Một cây nhị phân tận dụng lợi thế của hai kiểu cấu trúc dữ liệu: một mảng đã sắp thứ tự và một danh sách liên kết (Linked List), do đó việc tìm kiếm sẽ nhanh như trong mảng đã sắp thứ tự và các thao tác chèn và xóa cũng sẽ nhanh bằng trong Linked List.

A picture containing clock

Description automatically generated

1. Queues:

Queue(hàng đợi) là một loại container, được thiết kế để hoạt động theo kiểu FIFO (First- in first – out) (vào trước ra trước), tức là một kiểu danh sách mà việc bổ sung được thực hiển ở cuối danh sách và loại bỏ ở đầu danh sách.

Shape, rectangle

Description automatically generated

1. Stack:

Stack là một loại container adaptor, được thiết kế để hoạt động theo kiểu LIFO (Last - in first - out) (vào sau ra trước), tức là một kiểu danh sách mà việc bổ sung và loại bỏ một phần tử được thực hiển ở cuối danh sách. Vị trí cuối cùng của stack gọi là đỉnh (top) của ngăn xếp.

Diagram

Description automatically generated

## Tìm hiểu về Hệ thống nhúng.

### Giới thiệu.

Hệ thống nhúng (tiếng Anh: embedded system) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị được nhúng vào trong một môi trường hay hệ thống mẹ. Đó là các hệ thống tích hợp cả phần cứng và phần mềm phục vụ các bài toán chuyên dụng trong nhiều lĩnh vực công nghiệp, tự động hoá điều khiển, quan trắc và truyền tin. Đặc điểm của các hệ thống nhúng là hoạt động ổn định và có tính năng tự động hoá cao.

Các tính năng:

* Hệ thống nhúng được thiết kế cho những nhiệm vụ đặc biệt.
* Chương trình được ghi cho hệ thống nhúng được gọi là “firmware”, và thường được lưu trữ ở vùng (rom/flash).
* firmware có thể chạy với giới hạn phần cứng.

### Vi điều khiển và vi xử lý.

1. Vi xử lý

Vi xử lý (tiếng Anh là microprocessor hay microprocessor unit, viết tắt là µP hay uP), đôi khi còn được gọi là bộ vi xử lý, là một linh kiện điện tử máy tính được chế tạo từ các transistor thu nhỏ tích hợp lên trên một vi mạch tích hợp đơn. Khối xử lý trung tâm (CPU) là một bộ vi xử lý được nhiều người biết đến nhưng ngoài ra nhiều thành phần khác trong máy tính cũng có bộ vi xử lý riêng của nó, ví dụ trên card màn hình (Graphic card) chúng ta cũng có một bộ vi xử lý.

Vi xử lý chứa: Arithmetic and logic unit (ALU), Register array và control unit.

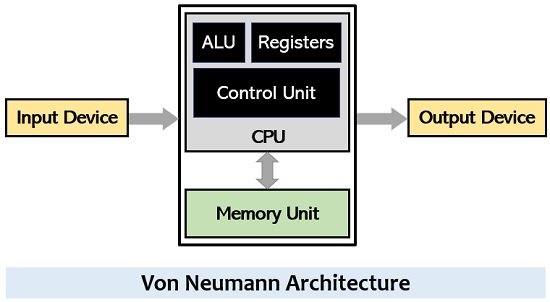
b. Vi điều khiển

Vi điều khiển là một máy tính được tích hợp trên một chip, nó thường được sử dụng để điều khiển các thiết bị điện tử. Vi điều khiển, thực chất, là một hệ thống bao gồm một vi xử lý có hiệu suất đủ dùng và giá thành thấp (khác với các bộ vi xử lý đa năng dùng trong máy tính) kết hợp với các khối ngoại vi như bộ nhớ, các module vào/ra, các module biến đổi số sang tương tự và tương tự sang số,... Ở máy tính thì các module thường được xây dựng bởi các chip và mạch ngoài.

Vi điều khiển chứa: Vi xử lý, bộ nhớ và các ngoại vi

### Các kiến trúc của vi điều khiển.

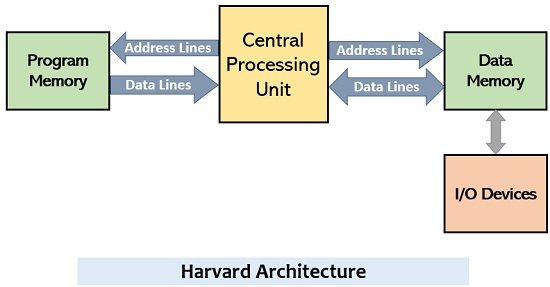
1. Kiến trúc Von neumann.



Thiết kế này đã phát triển lên có nghĩa là bất kỳ máy tính chương trình được lưu trữ nào trong đó việc tìm nạp lệnh và thao tác dữ liệu không thể xảy ra cùng một lúc vì chúng chia sẻ một bus chung. Điều này được gọi là nút cổ chai von Neumann và thường hạn chế hiệu suất của hệ thống.

Thiết kế của máy kiến ​​trúc von Neumann đơn giản hơn máy kiến ​​trúc Harvard - cũng là hệ thống lưu trữ nhưng có một bộ địa chỉ và bus dữ liệu chuyên dụng để đọc và ghi vào bộ nhớ, và một bộ bus địa chỉ và dữ liệu khác tìm nạp lệnh xử lý.

1. Kiến trúc Harvard.



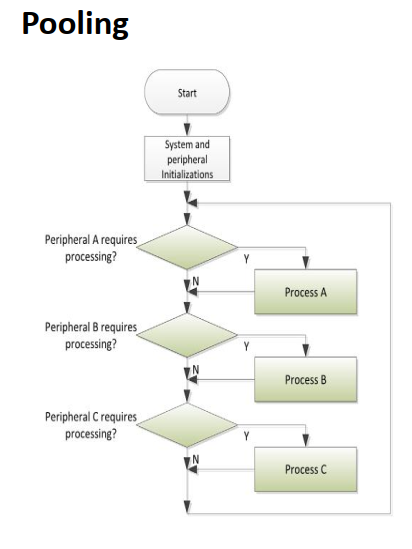
Kiến trúc Harvard được dùng để chỉ những kiến trúc máy tính mà trong đó phân biệt rõ ràng bộ nhớ dữ liệu và bộ nhớ chương trình, chúng có những đường truyền (bus) riêng để truy cập vào bộ nhớ dữ liệu và bộ nhớ chương trình.

### Software Flow.

1. Pooling:

Liên tục kiểm tra trạng thái của một ngoại vi, đọc dữ liệu từ đầu vào.

Tương đối đơn giản trong thiết kế với sự hi sinh của hiệu suất chương trình.



1. Interrupt:

Những sự kiện chỉ xảy ra khi ngắt được kích hoạt, Một xử lý ngắt được thực thi và CPU sẽ trở về lại hàm main.

Diagram

Description automatically generated

Interrupt process:

* CPU chờ tập lệnh hiện tại thực hiện xog.
* Lưu trạng thái hiện tại về thanh ghi của CPU, trạng thái thông tin ở bên trong đơn vị điều khiển( control unit).
* PC đọc địa chỉ của trình phục vụ ngắt( ISR – interrupt service routine).
* ISR được thực thi.
* Trở về chương trình từ ngắt.

## Các công cụ hỗ trợ.

### Visual Studio.

1. Icon

   Description automatically generatedGiới thiệu vê phần mềm:

Visual Studio là công cụ lập trình tốt nhất hiện nay được phát triển bởi Microsoft Coporation, được sử dụng phổ biến trong việc phát triển website, các phần mềm của Microsoft như Windows API, Windows Forms,... Visual Studio hỗ trợ lập trình trên nhiều ngôn ngữ giúp lập trình viên có thể lập trình được hệ thống một cách thuận tiện và nhanh chóng nhất.

Hỗ trợ việc Debug một cách mạnh mẽ và dễ dàng: Visual Studio là một trong những môi trường phát triển tích hợp (IDE) hỗ trợ tìm ra lỗi (debug) một cách dễ dàng và mạnh mẽ nhất nhờ vào Breakpoint. Ngoài cách xuất ra output những giá trị của biến thì Visual Studio có thể sử dụng Breakpoint để debug và kiểm tra từng giá trị trong mỗi câu lệnh.

1. Các bước sử dụng sử dụng debug:



* Các khái niệm:

+ Debug là quá trình tìm kiếm và fix bug (sửa lỗi) của chương trình. Quá trình debug thường mất nhiều thời gian hơn việc viết chương trình, vì những lỗi liên quan đến logic thường rất khó phát hiện. Visual Studio cung cấp nhiều công cụ trực quan để đơn giản hoá quá trình debug, tìm và sửa lỗi chương trình một cách dễ dàng hơn.

Text

Description automatically generated

+ Breakpoints là vị trí mà chương trình sẽ dừng lại để lập trình viên xem xét sự thay đổi của trạng thái chương trình, các biến qua từng dòng lệnh, từ đó phát hiện ra vị trí gây lỗi.

Graphical user interface, text, application

Description automatically generated

+ Cửa sổ Locals sẽ hiển thị tất cả các biến có liên quan đến dòng code hiện tại một cách tự động. Các biến hiển thị ở đây sẽ được thay đổi qua từng dòng code. Ngoài ra, màu sắc của các biến giúp ta phân biệt được những biến nào vừa thay đổi giá trị.

Text

Description automatically generated

+ Tương tự như Locals, cửa sổ Autos hiển thị các biến vừa được sử dụng trong các dòng code trước. Visual Studio sẽ tự động nhận diện biến nào không còn cần thiết và loại bỏ ra khỏi cửa sổ Autos.

Graphical user interface, application, Teams

Description automatically generated

+ Visual Studio không thể nhận diện được tất cả những gì lập trình viên cần. Trong một số trường hợp, cần theo dõi cụ thể một giá trị nào đó, chẳng hạn như phần tử thứ 10 trong một mảng số nguyên hay kí tự thứ 5 của chuỗi Stdio Tutorial, … Cần sử dụng đến cửa sổ Watch. Nó cho phép lập trình viên nhập vào tên biến, hàm, … cụ thể trong scope hiện tại. Giá trị của các biến, hàm sẽ được hiển thị bên cạnh tên biến.

A screenshot of a computer

Description automatically generated with medium confidence

* Debug chương trình:

Trước khi debug, cần tạo ra các breakpoints cần thiết để tìm và sửa lỗi chương trình. Để bắt đầu tiến hành debug một chương trình, vào menu Debug → Start Debugging hoặc nhấn phím F5 trên bàn phím. Visual Studio sẽ tiến hành Build chương trình. Sau khi Build xong và không có lỗi biên dịch (compile error), cửa sổ Debug sẽ xuất hiện. Nếu không có các lệnh dừng màn hình chờ nhập dữ liệu (như scanf, gets, …), chương trình sẽ dừng lại tại breakpoint đầu tiên.

A picture containing graphical user interface

Description automatically generated

Một thanh công cụ Debug sẽ xuất hiện. Trong đó, có các nút công cụ quan trọng như sau:

Step Over: Chạy step by step, lướt qua hàm (chỉ nhận giá trị return của hàm).

Step Into: Chạy step by step, đi vào nội dung của các hàm con.

Step Out: "Nhảy" đến breakpoint kế tiếp. Nếu không còn breakpoint nào thì sẽ kết thúc debug. Ngoài ra nó còn có chức năng chạy lướt qua hàm con hiện tại.

Restart: trở lại ban đầu.

Stop debugging: Dừng debug.

Một số phím tắt trong debug:

Table

Description automatically generated

### Keil uVision5

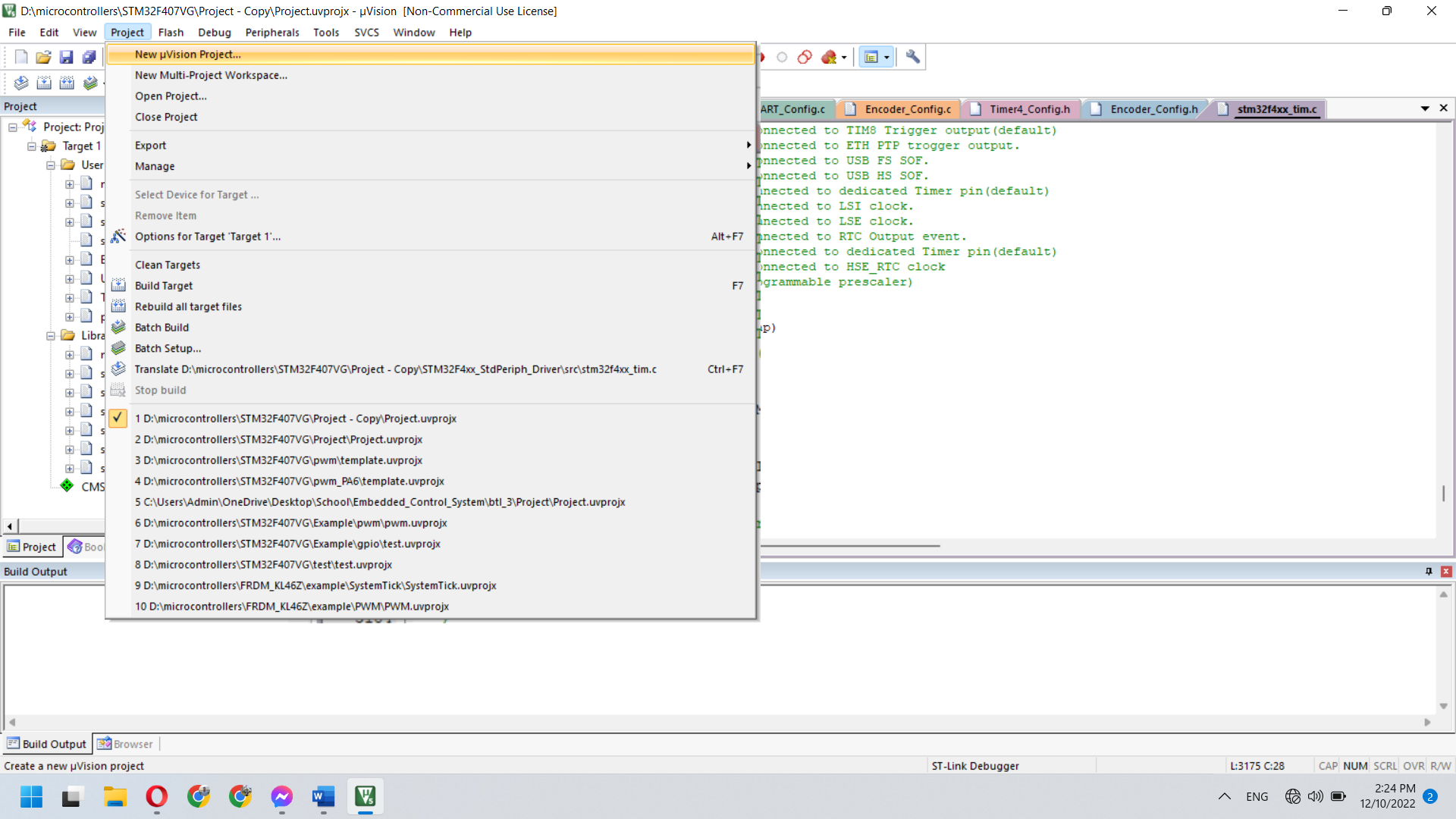
1. Giới thiệu phần mềm.

Keil C V5 là IDE được sử dụng nhiều nhất trong lập trình vi điều khiển, gắn bó lâu dài với sinh viên kỹ thuật trong những năm tháng đại học. Keil C không chỉ cho phép lập trình với các ngôn như khác nhau như C hay ASM, IDE này còn hỗ trợ rất nhiều dòng vi điều khiển, bạn có thể dễ dàng lựa chọn để lập trình.

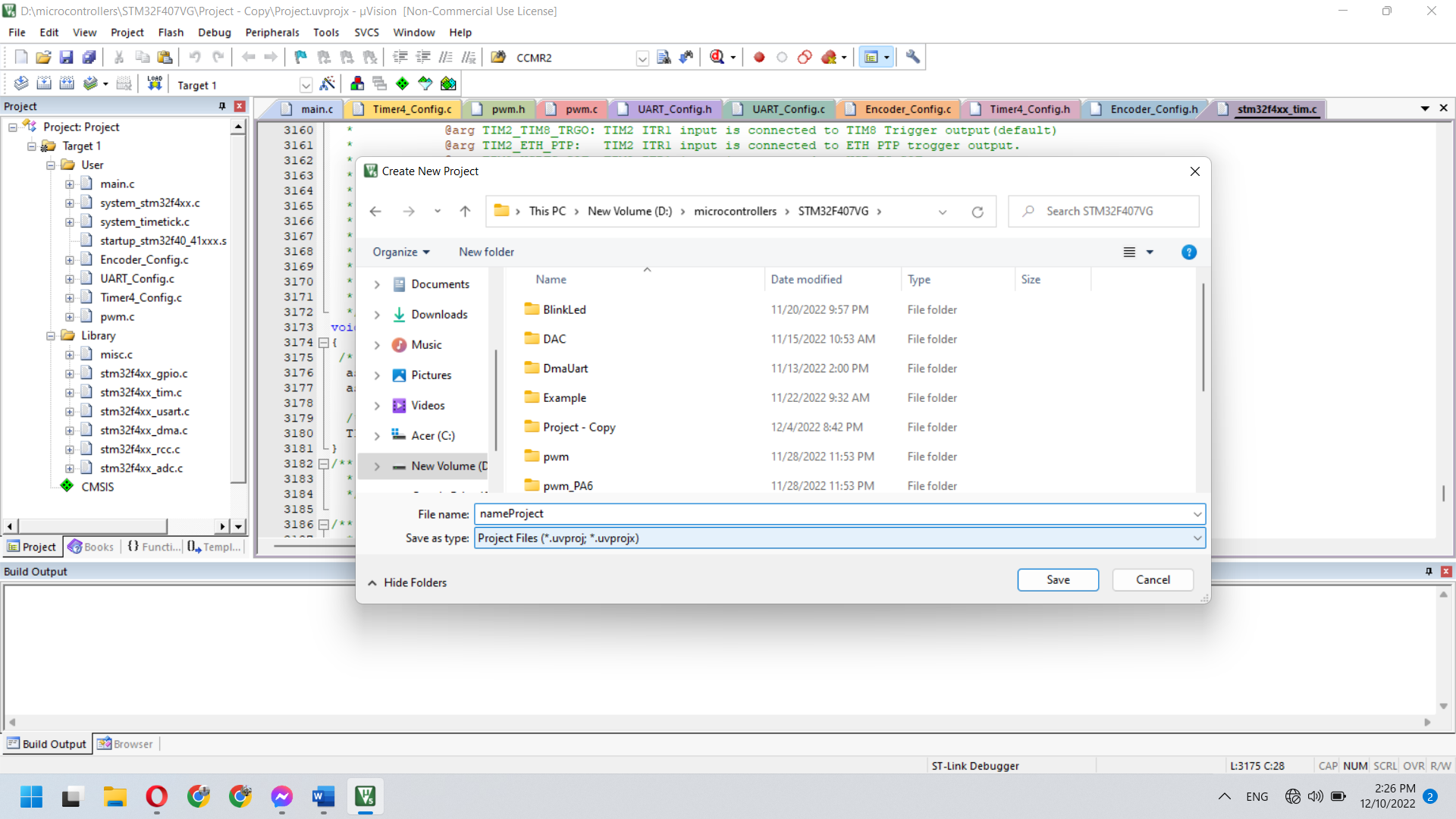
1. Sử dụng phần mềm.

Cách tạo project mới:

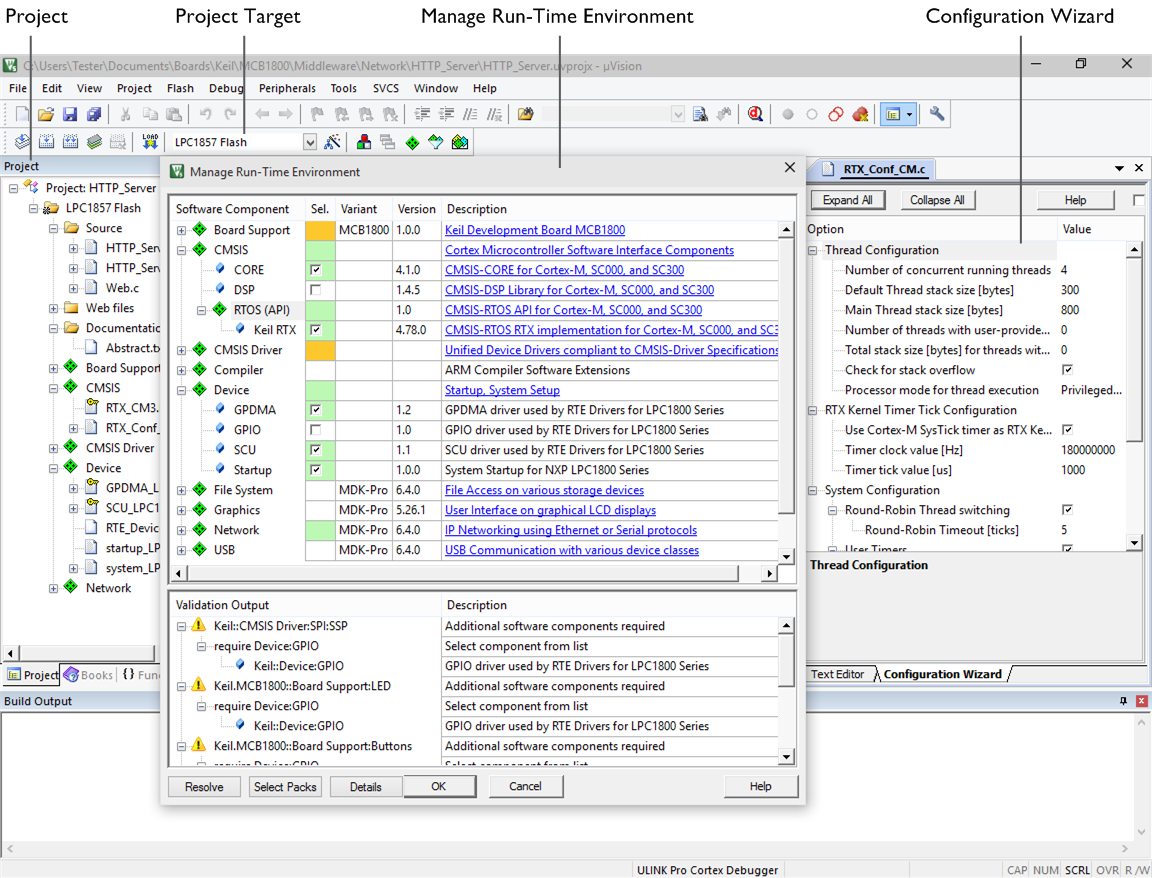
Bước 1: vào Project -> new Project…



Bước 2: chọn khu vực lưu project và đặt tên sau đó nhấn save.



Hướng dẫn sử dụng:

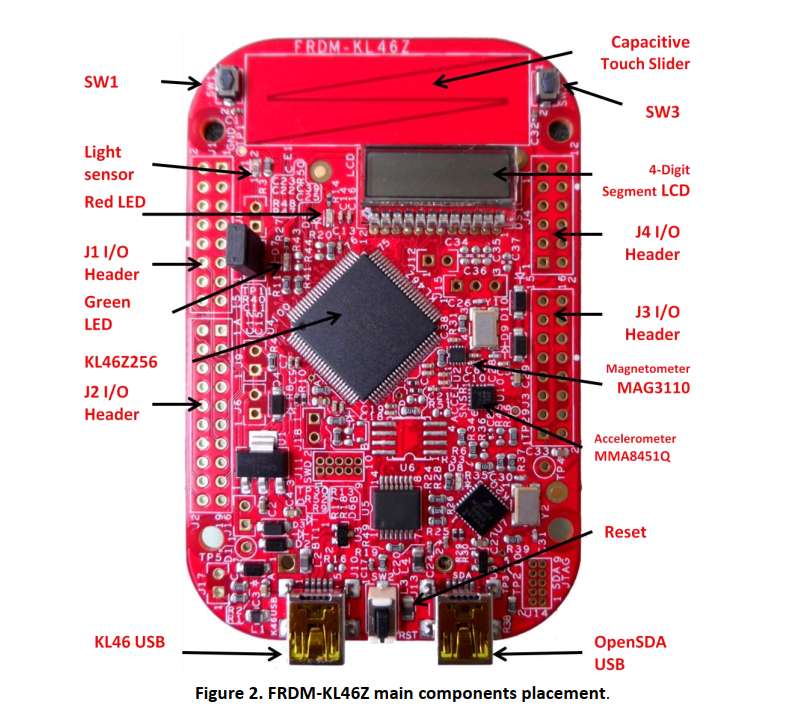


* Phân biệt các vùng trong cửa sổ phần mềm

Cách debug tương tự như với Visual Studio nhưng cần phải kết nối với vi điều khiển và cần cài đặt một số driver cần thiết cho kết nối và debug.

### Board KL46Z

1. Giới thiệu:



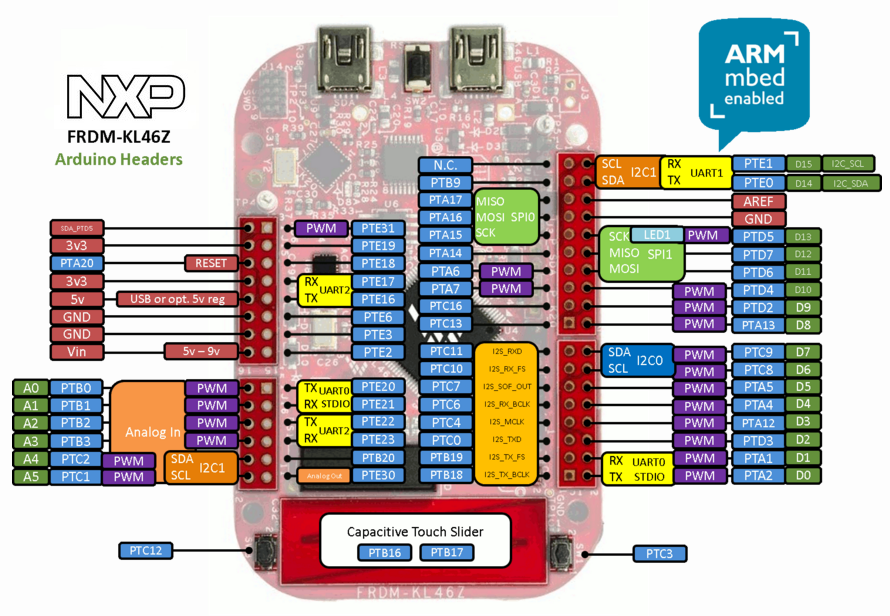
Nền tảng phát triển FRDM-KL46Z của hãng Freescale (nay là NXP) là một bộ công cụ gồm phần mềm và phần cứng để đánh giá và phát triển các ứng dụng nhúng dựa trên dòng vi điều khiển Kinetis L. Phần cứng của nền tảng, bo mạch FRDM-KL46Z, có thiết kế tinh vi với bộ vi điều khiển MKL46Z256VLL4 là bộ vi điều khiển công nghiệp đầu tiên được xây dựng trên lõi 32-bit ARM® Cortex ™ -M0+.

Bo mạch FRDM-KL46Z được giới thiệu lần đầu tiên ở Việt Nam vào năm 2013, tuy nhiên đến nay thì bo mạch này mới được bán rộng rãi trên thị trường.

Kit phát triển FRDM-KL46Z là một kit sử dụng lõi vi xử lý ARM® Cortex™-M0+ có công suất tiêu thụ thấp, giá thành rẻ do công ty Freescale (hiện nay là NXP) sản xuất. Kit có kích thước khá nhỏ, cỡ nhỏ hơn một thẻ ATM. Nhưng trên kit tích hợp khá nhiều tính năng hấp dẫn, bao gồm:

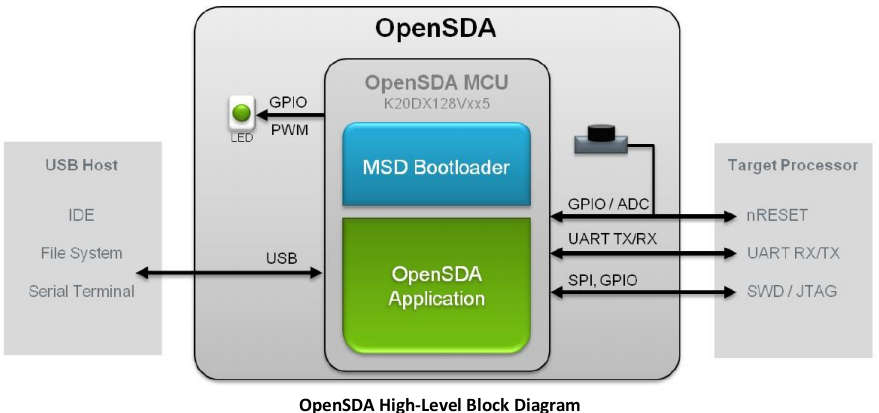
Chip vi điều khiển (MCU: Microcontroller Unit)) ARM MKL46Z256VLL4 sử dụng lõi vi xử lý ARM® Cortex™-M0+ hiệu năng cao, công suất tiêu thụ thấp, có tần số hoạt động cực đại là 48MHz, tích hợp bộ nhớ flash 256kB và SRAM 32kB, tích hợp bộ điều khiển segment LCD, USB OTG, …

* Cảm ứng điện dung (Capacitive touch slider) với bộ điều khiển cảm biến được tích hợp sẵn trên chip ARM;
* Cảm biến gia tốc (Accelerometer) MMA8451Q;
* Cảm biến từ trường (Magnetometer) MAG3110N;
* Cảm biến ánh sáng ALS-PT19-315C/L177/TR8;
* Ngoại vi I/O có thể lập trình: 2 LEDs và 2 nút bấm kiểu Switch;
* LCD-S401M16KR với 4 chữ số kiểu 8-segment
* Nguồn nuôi có thể được lựa chọn linh hoạt từ 5V bên ngoài hoặc từ cổng USB máy tính hoặc hoặc từ pin 3V3;
* Các đầu cắm mở rộng chức năng (Header) tương thích hoàn toàn với chuẩn Arduino ™ R3 do đó dễ dàng sử dụng với các mô-đun mở rộng chức năng của Arduino;
* Tích hợp sẵn bộ Debug OpenSDA trên kit, có thể sử dụng Open SDA để nạp chương trình, chạy debug và giao tiếp chuẩn RS232 với máy tính.

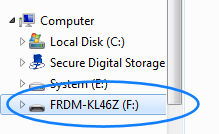


Hình Bố trí các đầu cắm mở rộng (Expand Header) trên kit FRDM-KL46Z.

1. Giao diện nạp chương trình OpenSDA tích hợp sẵn trên Kit



Khi được cấp nguồn và ghép nối với máy tính, bo mạch FRDM hoạt động ở chế độ “Thiết bị lưu trữ khối MSD” (Mass Storage Device) cho phép bo mạch (hay chính xác hơn là vi điều khiển điều khiển) được nhận diện như một ổ nhớ USB trên máy tính, do đó người dùng có thể kéo và thả các chương trình đã được biên dịch lên đó. Khi thao tác này xảy ra, bộ vi điều khiển điều khiển tự động reset bộ vi điều khiển đích (tức là bộ vi điều khiển Kinetis KL46Z256VLL4) và lập trình lại nó rồi cho phép vi điều khiển đích hoạt động.

Kit FRDM-KL46Z được nhận diện như một ổ USB trên máy tính.

# Chương 3: Kết luận

Trong quá trình đào tạo, em đã được học hỏi rất nhiều điều về cách tìm kiếm tài liệu và cách đọc tài liệu một cách hiệu quả nhất, còn có cả cách hoạt động của một công việc theo yêu cầu được đề ra.

Bên cạnh đó, một số phần mềm và các kỹ năng cần thiết khi ra ngoài làm cũng được anh chị hướng dẫn tận tình và cho chúng em cơ hội thực hành bằng việc áp dụng kiến thức đã học vào tính toán và thiết kế cho dự án thực tế.

Và ấn tượng không kém đối với chúng em chính là những kinh nghiệm đúc rút từ thực tế làm việc đã được anh chị chia sẻ một cách tận tình. Điều đó thực sự vô cùng quý giá, cho chúng em cái nhìn thực tế để định hướng công việc sau này lẫn những kiến thức, kỹ năng mà bản thân cần trau dồi nhằm trở thành một người lao động có giá trị.

Qua quá trình đào tạo tại Công ty, em nhận thấy mỗi người sinh viên trau dồi kiến thức một cách chỉnh chu và cả trải nghiệm công việc một cách thực tế công việc để có kinh nghiệm làm việc và biết cách làm việc hiệu quả nhất và tốt nhất cho mỗi người.

Một lần nữa, em chân thành cảm ơn Thầy cô, Quý công ty và tất cả các anh chị mà em không có may mắn biết tên đã giúp đỡ chúng em có cơ hội thực tập quý giá này.

# Tài liệu tham khảo

[1]  <https://www.geeksforgeeks.org/memory-layout-of-c-program/>

[2]  https://vi.wikipedia.org/

[3]  https://www.iostream.vn/article/co-ban-debug-trong-visual-studio-F1Yf3U

[4] https://codelearn.io/learning

[5]  https://circuitglobe.com/difference-between-von-neumann-and-harvard-architecture

[6]  KL46 Sub-Family Reference Manual (Docs/KL46P121M48SF4RM.pdf)

[7] Quick Start Guide for FRDM-KL46Z (Quick Start Guide for Freedom KL46Z Rev1.pdf)

[8]  FRDM-KL46Z User’s Manual (FRDM-KL46Z\_UM.pdf)

[9] FRDM-KL46Z Schematic (FRDM-KL46Z\_SCH.pdf)

[10] FRDM-KL46 OpenSDA applications