**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

****

**BÁO CÁO TIỂU LUẬN**

**HỆ THỐNG NHÚNG**

**VÀ SDK FLUTTER**

Giảng viên: Đậu Trọng Hiển

Thành viên nhóm

Nguyễn Tiến Đạt 20139034

Trần Quốc Sĩ 19119215

Nguyễn Nhật Tùng 21139068

Tp.HCM, ngày 12 tháng 1 năm 2022

**Mục lục**

[**I. Phần cứng hệ thống** 1](#_Toc120801652)

[**1. Bộ xử lý đơn dụng (single purpose processor)** 1](#_Toc120801653)

[**2. Bộ xử lý đa dụng (general-purpose processor)** 5](#_Toc120801654)

[**3. Bộ xử lý chuyên dụng (Dedicated processor)** 6](#_Toc120801655)

[**4. UART, SPI, I2C, LAN** 10](#_Toc120801656)

[**4.1. UART** 10](#_Toc120801657)

[**4.2 SPI** 17](#_Toc120801658)

[**4.3 I2C** 21](#_Toc120801659)

[**4.4 LAN** 30](#_Toc120801660)

[**II. Phần mềm hệ thống nhúng** 37](#_Toc120801661)

[**1. Hệ điều hành nhúng RTOS, android, linux** 37](#_Toc120801662)

[**1.1 Hệ điều hành RTOS** 37](#_Toc120801663)

[**1.2 Hệ điều hành nhúng android** 40](#_Toc120801664)

[**1.3 Hệ điều hành linux** 42](#_Toc120801665)

[**2. Ngôn ngữ lập trình và SDK/IDE** 43](#_Toc120801666)

[**2.1 Lịch sử Flutter** 43](#_Toc120801667)

[**2.2 Flutter là gì?** 43](#_Toc120801668)

[**2.3 Giới thiệu ngôn ngữ Dart:** 44](#_Toc120801669)

[**2.4 Kiến trúc Flutter** 48](#_Toc120801670)

[**2.5 Ưu điểm của Flutter** 48](#_Toc120801671)

[**2.6 Nhược điểm của Flutter** 49](#_Toc120801672)

[**2.7 Hướng dẫn cài đặt Flutter** 50](#_Toc120801673)

[**III. Bài tập** 55](#_Toc120801674)

[**1. Chức năng phần mềm** 55](#_Toc120801675)

[**2. Lưu đồ giải thuật** 55](#_Toc120801676)

[**3. Code** 56](#_Toc120801677)

[**Tài liệu tham khảo** 77](#_Toc120801678)

# **I. Phần cứng hệ thống**

CPU (hay Bộ xử lý trung tâm) là bộ não của bất kỳ máy tính nào. Họ đã trải qua rất nhiều thay đổi và cải tiến kể từ lần đầu tiên được sản xuất. Tuy nhiên, các chức năng cơ bản của tất cả các CPU vẫn giống nhau, tức là tìm nạp, giải mã, thực thi và lưu trữ dữ liệu.

Trong kịch bản hiện tại, tất cả các công ty đều mong đợi cơ sở hạ tầng dựa trên đám mây để lưu trữ dữ liệu nhằm lưu cơ sở dữ liệu có giá trị của họ một cách an toàn. Nhưng câu hỏi đầy thách thức là: chọn kế hoạch nào để đáp ứng yêu cầu của họ và đạt được hiệu quả tối đa, và làm thế nào loại CPU hiện có có thể nâng cao trải nghiệm người dùng của bạn về mặt xử lý và quản lý cả khối lượng công việc và hoạt động.

Hãy bắt đầu bằng cách tìm hiểu riêng về từng loại CPU.

## 1. Bộ xử lý đơn dụng (single purpose processor)

Một bộ xử lý đơn dụng là một kỹ thuật số; mạch được thiết kế để thực hiện chính xác một chương trình. Một nhà thiết kế hệ thống nhúng có thể nhận được một số lợi ích bằng cách chọn sử dụng bộ xử lý đơn dụng tùy chỉnh để thực hiện tác vụ tính toán.

Một bộ xử lý cơ bản bao gồm một bộ điều khiển và một đường dẫn dữ liệu. Đường dẫn dữ liệu lưu trữ và thao tác dữ liệu của hệ thống. Đường dẫn dữ liệu chứa các đơn vị thanh ghi, đơn vị chức năng và kết nối như dây dẫn và bộ ghép kênh. Đường dẫn dữ liệu có thể được cấu hình để đọc dữ liệu từ các thanh ghi cụ thể cung cấp dữ liệu đó thông qua các đơn vị chức năng được cấu hình để thực hiện các hoạt động cụ thể như thêm hoặc dịch chuyển và lưu kết quả hoạt động trở lại vào các thanh ghi cụ thể. Bộ điều khiển thực hiện cấu hình như vậy của đường dẫn dữ liệu. Nó đặt các đầu vào điều khiển đường dẫn dữ liệu, như tải thanh ghi và tín hiệu chọn bộ ghép kênh, của các đơn vị thanh ghi, đơn vị chức năng và đơn vị kết nối để có được cấu hình mong muốn tại một thời điểm cụ thể.

Nó giám sát đầu vào điều khiển bên ngoài cũng như đầu ra điều khiển đường dẫn dữ liệu, được gọi là tín hiệu trạng thái, đến từ các đơn vị chức năng và nó cũng đặt đầu ra điều khiển bên ngoài.

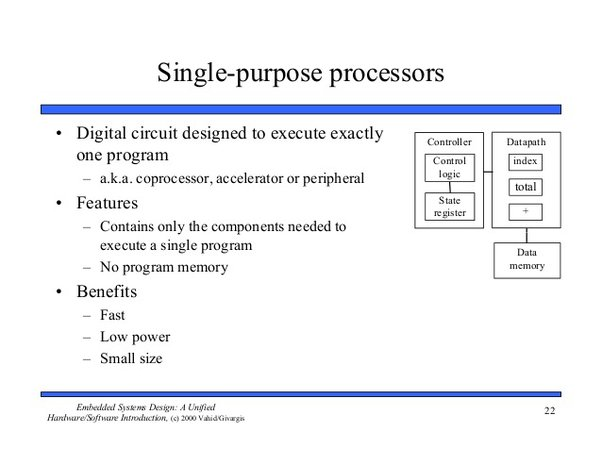
Các kỹ thuật thiết kế hệ thống kỹ thuật số như thiết kế logic tổ hợp và tuần tự bao gồm các kỹ thuật thiết kế đồng bộ và không đồng bộ có thể được áp dụng để xây dựng BỘ ĐIỀU KHIỂN và ĐƯỜNG DỮ LIỆU.

**Ưu điểm:**

1. Hiệu suất có thể nhanh hơn, do ít chu kỳ xung nhịp hơn do đường dẫn dữ liệu tùy chỉnh và do chu kỳ xung nhịp ngắn hơn do logic bộ điều khiển đơn giản hơn.
2. Kích thước có thể nhỏ hơn do đường dẫn dữ liệu đơn giản nhất và không có bộ nhớ chương trình.
3. Điện năng tiêu thụ có thể ít hơn do tính toán hiệu quả hơn.

**Nhược điểm:**

1. Không dễ thiết kế và thời gian thiết kế cao
2. Chi phí thiết kế do đó cao hơn
3. Lập trình lại rất khó
4. Tính linh hoạt hạn chế: không dễ thực hiện thay đổi, phù hợp với các tính năng





(chi phí có thể cao hơn do chi phí NRE cao. Ngoài ra thời gian đưa ra thị trường có thể lâu hơn.)

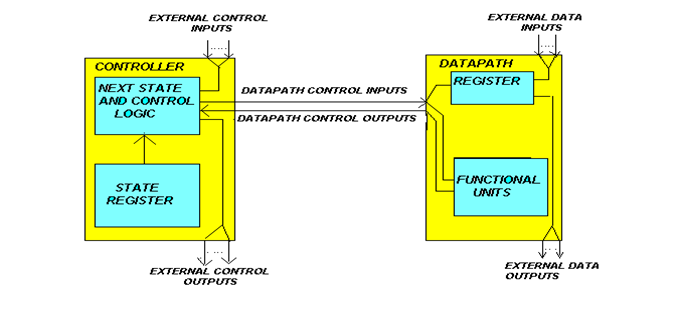
**Thiết kế của bộ xử lý đơn dụng:**

Có thể xem như thiết kế của một

Hệ thống gồm 2 thành phần:

• Đường dẫn dữ liệu thực thi hoạt động cần thiết đến hệ thống

• Đơn vị Điều khiển, tạo ra các lệnh cho đường dẫn dữ liệu trên cơ sở dữ liệu đầu vào và các điều kiện



Bộ xử lý đơn dụng

Trong thiết kế, chúng ta có thể tạo ra sự khác biệt giữa bộ xử lý đơn dụng tiêu chuẩn và bộ xử lý đơn dụng tùy chỉnh (ASSP). Bộ xử lý đơn dụng tiêu chuẩn (thiết bị ngoại vi phần cứng) được giải thích trong một chương khác Thiết bị ngoại vi phần cứng . Chúng được sử dụng trong nhiều ứng dụng cho cùng một nhiệm vụ.

* ví dụ: mạch tích hợp dành riêng cho ứng dụng (ASIC) , sản phẩm tiêu chuẩn dành riêng cho ứng dụng (ASSP)

Chúng được thiết kế để phù hợp với các yêu cầu cụ thể của một sản phẩm cuối cùng. Trong quá trình thiết kế, chúng sẽ được tối ưu hóa cả về sức mạnh và hiệu suất. ASIC được giới thiệu cho một sản phẩm thường sau khi sản phẩm đã ổn định trên thị trường (với giải pháp không phải ASIC), do thời gian và chi phí phát triển cao hơn. Sau khi ổn định, nhà sản xuất có thể tìm kiếm thêm lợi nhuận bằng cách làm cho bộ xử lý tốt nhất có thể cho ứng dụng của mình. Một bất lợi là khi bạn mắc lỗi, bạn sẽ cần tạo một con chip hoàn toàn mới, nhưng đối với bộ xử lý đa năng, bạn chỉ cần thay đổi mã và biên dịch lại. Bạn có thể sử dụng ASIC trong một thị trường ổn định, nơi công nghệ và các yêu cầu sẽ không thay đổi nhanh chóng.

## 2. Bộ xử lý đa dụng (general-purpose processor)

Bộ xử lý macro mục đích chung hoặc bộ tiền xử lý mục đích chung là bộ xử lý macro không được gắn với hoặc tích hợp với một ngôn ngữ hoặc phần mềm cụ thể. Bộ xử lý macro là một chương trình sao chép một luồng văn bản từ nơi này sang nơi khác, tạo ra một tập hợp thay thế có hệ thống như vậy.

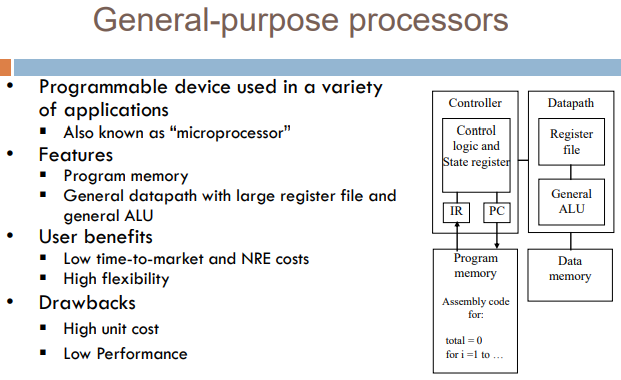
Người thiết kế hệ thống chỉ cần lập trình bộ nhớ của bộ xử lý để thực hiện chức năng được yêu cầu: phần mềm. Các loại này được tạo ra để sản xuất số lượng lớn cho nhiều ứng dụng.

**Ưu điểm:**

* Dễ thiết kế và sử dụng (chỉ lập trình bộ nhớ)
* Do đó thời gian thiết kế thấp
* Do đó chi phí thiết kế thấp
* Khả năng lập trình lại (thay đổi chức năng hoặc cải tiến hệ thống trở nên dễ dàng khi bạn chỉ cần thay đổi chương trình)

**Nhược điểm**:

* Hiệu suất không phải là rất tốt
* Kích thước lớn (vì chúng được xây dựng để sử dụng trong nhiều ứng dụng với các thông số kỹ thuật khác nhau và tất cả các thông số kỹ thuật này phải có trên bộ xử lý)
* Tiêu thụ nhiều năng lượng



Thông thường chúng được sử dụng để đưa vào một sản phẩm lần đầu tiên đưa ra thị trường, vì chi phí thiết kế rẻ và dễ sử dụng. Do việc sử dụng mã, một nhà thiết kế thường cố gắng giảm thiểu kích thước mã được biên dịch hơn là tối đa hóa hiệu suất.

* ví dụ: Bộ vi [điều khiển](https://en.wikipedia.org/wiki/Microcontroller" \o "w:Vi điều khiển) (MCU), [Bộ vi xử lý](https://en.wikipedia.org/wiki/Microprocessor" \o "w:Bộ vi xử lý)

## 3. Bộ xử lý chuyên dụng (Dedicated processor)

**CPU chuyên dụng là gì?**

CPU chuyên dụng có nghĩa là công ty của bạn sẽ có CPU hoặc lõi được chỉ định riêng cho bạn và chỉ cá nhân bạn mới có thể sử dụng chúng. CPU chuyên dụng là một đơn vị rất mạnh được tạo riêng cho các ứng dụng và tác vụ cần sức mạnh xử lý lớn. Các chức năng như phân tích dữ liệu, máy học, mã hóa và hoạt động tốc độ cao có thể xử lý trơn tru trên các CPU như vậy. Lợi ích quan trọng nhất của việc sử dụng một bộ CPU chuyên dụng là hiệu suất của CPU sẽ nhất quán và hiệu quả hơn trong suốt thời gian sử dụng.

Bạn không cần phải chia sẻ sức mạnh của bộ xử lý để chạy bất kỳ chức năng nào khác cho bất kỳ thiết bị nào khác. Trên thực tế, các CPU chuyên dụng sẽ chạy trên lõi chuyên dụng của riêng chúng. Những loại CPU này cung cấp quyền sở hữu và bảo mật dữ liệu và ứng dụng được cải thiện.

Tuy nhiên, trong CPU dùng chung, lõi và bộ xử lý sẽ được chia sẻ bởi nhiều người dùng dưới một máy chủ cụ thể. Việc sử dụng chung này có thể dẫn đến nguy cơ đe dọa bảo mật cao hơn và tốc độ xử lý rất chậm. Bằng cách sử dụng CPU chuyên dụng, những vấn đề này không xảy ra vì mỗi người có máy và bộ xử lý chuyên dụng để thực hiện các hoạt động.

**Ưu điểm của việc sử dụng CPU chuyên dụng**

● Thông qua các đơn vị CPU này, bạn có thể sử dụng riêng Core của CPU.

● Vì đơn vị CPU là dành riêng cho bạn nên nó làm trơn tru các hoạt động bằng cách cung cấp cho bạn toàn quyền kiểm soát.

● Do việc sử dụng các thiết bị này cực kỳ hạn chế nên các hệ thống này cung cấp bảo mật cấp cao và giảm rủi ro vi phạm dữ liệu.

● Thông qua các CPU này, bạn có được các tính năng xử lý nhanh và sao lưu dữ liệu.

**Tổ chức nào thích CPU chuyên dụng?**

Hãy phân tích nhanh xem loại hình doanh nghiệp nào thích sử dụng CPU chuyên dụng hiện nay:

● Dữ liệu lớn và các tổ chức dựa trên phân tích dữ liệu hoạt động theo cách tiếp cận 3V

● Các hãng máy tính khoa học lớn

● Các tổ chức sử dụng chuỗi công cụ tích hợp liên tục, phân phối liên tục và sử dụng máy chủ xây dựng

● Máy chủ trò chơi sử dụng nhiều CPU, như Team Fortress, Minecraft và Rust

● Truyền phát và chuyển mã âm thanh và video

● Các công cụ học máy như Apache, TensorFlow và PyTorch

● Doanh nghiệp truyền dữ liệu và lập trình

**Loại khối lượng công việc nào CPU chuyên dụng có thể xử lý?**

CPU chuyên dụng có thể xử lý khối lượng công việc như: -

● Tải của máy chủ web có lưu lượng truy cập trung bình đến cao

● Bảo trì các trang thương mại điện tử

● Xử lý cơ sở dữ liệu cỡ trung bình

● Chạy Phần mềm Doanh nghiệp dưới dạng Dịch vụ (SaaS)

Vì tất cả các chức năng này dẫn đến khối lượng công việc lớn trên máy của bạn và do đó, việc xử lý CPU chuyên dụng và bền vững là cần thiết mọi lúc có thể để vận hành trơn tru.

**Khi nào bạn cần một CPU chuyên dụng?**

Bạn có thể cần một CPU chuyên dụng:

● Nếu bạn đang chạy bất kỳ ứng dụng ngốn CPU hoặc ngốn bộ nhớ nào

● Nếu bạn nhận được thông báo "mức sử dụng cao" trong nhiều ngày liên tiếp và gặp phải sự chậm trễ trong quá trình xử lý trên máy của bạn.

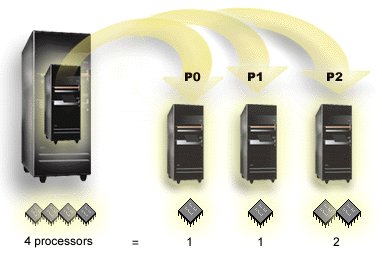
Sử dụng thông tin được đề cập ở trên và mô hình làm việc của công ty bạn, bạn có thể quyết định liệu một hệ thống CPU chuyên dụng có phải là lựa chọn phù hợp với bạn hay không.

**Giải quyết vấn đề**

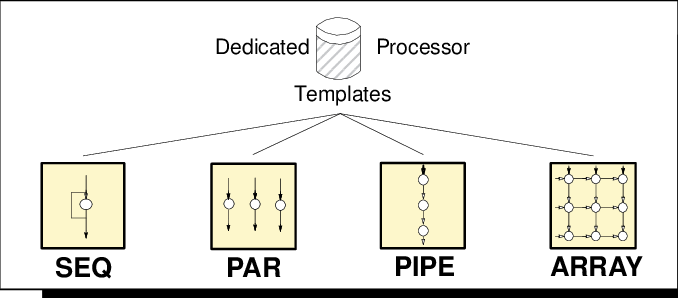
Bộ xử lý chuyên dụng là toàn bộ bộ xử lý được sử dụng riêng bởi phân vùng mà chúng được gán. Bộ xử lý chuyên dụng xử lý việc xử lý cho một phân vùng hợp lý cụ thể.

Nếu bạn chọn gán các bộ xử lý chuyên dụng cho một phân vùng logic, bạn phải gán ít nhất một bộ xử lý cho phân vùng đó. Tương tự như vậy, nếu bạn chọn xóa tài nguyên bộ xử lý khỏi một phân vùng chuyên dụng, bạn phải xóa ít nhất một bộ xử lý khỏi phân vùng đó.

Để điều chỉnh theo khối lượng công việc thay đổi, bạn có thể di chuyển các bộ xử lý chuyên dụng trong các giá trị tối thiểu/tối đa mà bạn thiết lập mà không cần phải khởi động lại phân vùng. Các giá trị này cho phép bạn thiết lập một phạm vi trong đó bạn có thể tự động di chuyển tài nguyên mà không cần phải khởi động lại phân vùng logic. Khi bạn thay đổi các giá trị tối thiểu/tối đa, nó yêu cầu bạn khởi động lại phân vùng. Các giá trị tối thiểu chỉ ra những gì cần thiết để khởi động lại phân vùng. Nếu giá trị tối thiểu không được đáp ứng cho tất cả các phân vùng hợp lý, chỉ phân vùng chính sẽ khởi động lại.



Ví dụ: một máy chủ có bốn bộ xử lý vật lý có thể có ba phân vùng logic, với hai phân vùng có một bộ xử lý chuyên dụng và một phân vùng có hai bộ xử lý chuyên dụng.



Các loại bộ xử lý chuyên dụng

## 4. UART, SPI, I2C, LAN

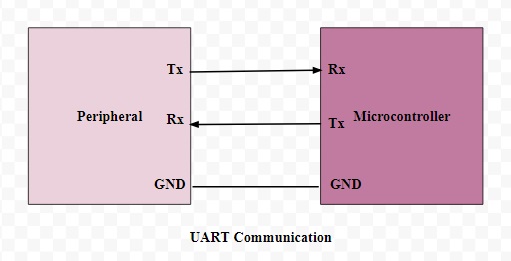
### **4.1. UART**

UART hay bộ thu-phát không đồng bộ đa năng là một trong những hình thức giao tiếp kỹ thuật số giữa thiết bị với thiết bị đơn giản và lâu đời nhất. Bạn có thể tìm thấy các thiết bị UART trong một phần của mạch tích hợp (IC) hoặc dưới dạng các thành phần riêng lẻ. Các UART giao tiếp giữa hai nút riêng biệt bằng cách sử dụng một cặp dẫn và một nối đất chung.

Nếu chúng ta nhớ các bộ phận máy tính cũ như máy in, chuột, bàn phím được liên kết với sự trợ giúp của các đầu nối. Quá trình giao tiếp giữa máy tính và các bộ phận này có thể được thực hiện bằng UART. Universal serial Bus (USB) đã thay đổi tất cả các loại nguyên tắc giao tiếp trên máy tính. Nhưng, UART vẫn được sử dụng trong các ứng dụng được khai báo ở trên. Tất cả các loại kiến trúc vi điều khiển đều có phần cứng UART tích hợp do giao tiếp nối tiếp và chỉ sử dụng hai cáp để liên lạc. Bài viết này thảo luận về UART, Cách thức hoạt động của UART, sự khác biệt giữa giao tiếp nối tiếp và song song, sơ đồ khối UART, giao tiếp UART, Ứng dụng, Ưu điểm và Nhược điểm.

**4.1.1 UART là gì?**

Các tên đầy đủ UART là “Universal Asynchronous Receiver / Transmitter”, và nó là một vi mạch sẵn có trong một vi điều khiển nhưng không giống như một giao thức truyền thông (I2C & SPI). Chức năng chính của UART là truyền dữ liệu nối tiếp. Trong UART, giao tiếp giữa hai thiết bị có thể được thực hiện theo hai cách là giao tiếp dữ liệu nối tiếp và giao tiếp dữ liệu song song.

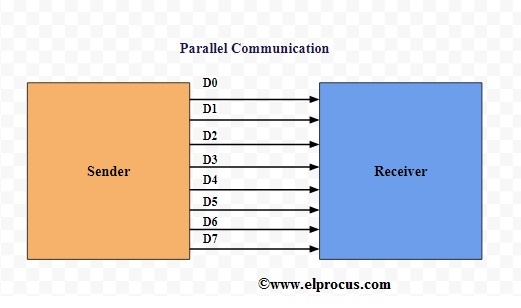


UART

**4.1.2 Truyền thông nối tiếp và song song**

Trong giao tiếp dữ liệu nối tiếp, dữ liệu có thể được truyền qua một cáp hoặc một đường dây ở dạng bit-bit và nó chỉ cần hai cáp. Truyền thông dữ liệu nối tiếp không đắt khi chúng ta so sánh với giao tiếp song song. Nó đòi hỏi rất ít mạch cũng như dây. Vì vậy, giao tiếp này rất hữu ích trong các mạch ghép so với giao tiếp song song.

Trong giao tiếp dữ liệu song song, dữ liệu có thể được truyền qua nhiều cáp cùng một lúc. Truyền dữ liệu song song tốn kém nhưng rất nhanh, vì nó đòi hỏi phần cứng và cáp bổ sung. Các ví dụ tốt nhất cho giao tiếp này là máy in cũ, PCI, RAM, v.v.



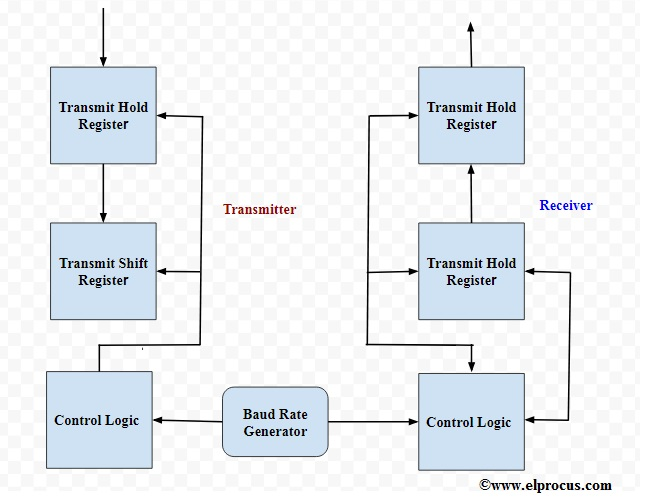
Giao tiếp song song

**4.1.3 Sơ đồ khối UART**

Sơ đồ khối UART bao gồm hai thành phần là máy phát và máy thu được hiển thị bên dưới. Phần máy phát bao gồm ba khối là thanh ghi giữ truyền, thanh ghi dịch chuyển và logic điều khiển. Tương tự, phần máy thu bao gồm một thanh ghi giữ, thanh ghi thay đổi và logic điều khiển. Hai phần này thường được cung cấp bởi một bộ tạo tốc độ baud. Trình tạo này được sử dụng để tạo tốc độ khi phần máy phát và phần máy thu phải truyền hoặc nhận dữ liệu.

Thanh ghi giữ trong máy phát bao gồm byte dữ liệu được truyền. Các thanh ghi thay đổi trong máy phát và máy thu di chuyển các bit sang phải hoặc trái cho đến khi một byte dữ liệu được truyền hoặc nhận. Một logic điều khiển đọc (hoặc) ghi được sử dụng để biết khi nào nên đọc hoặc viết.

Máy phát tốc độ baud giữa máy phát và máy thu tạo ra tốc độ dao động từ 110 bps đến 230400 bps. Thông thường, tốc độ truyền của vi điều khiển là 9600 đến 115200.

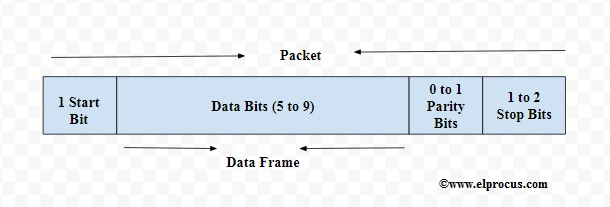
****

Sơ đồ khối UART

**4.1.4 Truyền thông UART**

Trong giao tiếp này, có hai loại UART có sẵn là truyền UART và nhận UART và giao tiếp giữa hai loại này có thể được thực hiện trực tiếp với nhau. Đối với điều này, chỉ cần hai cáp để giao tiếp giữa hai UART. Luồng dữ liệu sẽ từ cả hai chân truyền (Tx) và nhận (Rx) của UARTs. Trong UART, việc truyền dữ liệu từ Tx UART sang Rx UART có thể được thực hiện không đồng bộ (không có tín hiệu CLK để đồng bộ hóa các bit o / p).

Việc truyền dữ liệu của UART có thể được thực hiện bằng cách sử dụng bus dữ liệu ở dạng song song bởi các thiết bị khác như vi điều khiển, bộ nhớ, CPU, v.v. Sau khi nhận được dữ liệu song song từ bus, nó tạo thành gói dữ liệu bằng cách thêm ba bit như bắt đầu, dừng lại và trung bình. Nó đọc từng bit gói dữ liệu và chuyển đổi dữ liệu nhận được thành dạng song song để loại bỏ ba bit của gói dữ liệu. Tóm lại, gói dữ liệu nhận được bởi UART chuyển song song về phía bus dữ liệu ở đầu nhận.

****

Truyền thông UART

**Start-bit**

Start-bit còn được gọi là bit đồng bộ hóa được đặt trước dữ liệu thực tế. Nói chung, một đường truyền dữ liệu không hoạt động được điều khiển ở mức điện áp cao. Để bắt đầu truyền dữ liệu, truyền UART kéo đường dữ liệu từ mức điện áp cao (1) xuống mức điện áp thấp (0). UART thu được thông báo sự chuyển đổi này từ mức cao sang mức thấp qua đường dữ liệu cũng như bắt đầu hiểu dữ liệu thực. Nói chung, chỉ có một start-bit.

**Bit dừng**

Bit dừng được đặt ở phần cuối của gói dữ liệu. Thông thường, bit này dài 2 bit nhưng thường chỉ sử dụng 1 bit. Để dừng sóng, UART giữ đường dữ liệu ở mức điện áp cao.

**Bit chẵn lẻ**

Bit chẵn lẻ cho phép người nhận đảm bảo liệu dữ liệu được thu thập có đúng hay không. Đây là một hệ thống kiểm tra lỗi cấp thấp & bit chẵn lẻ có sẵn trong hai phạm vi như Chẵn lẻ – chẵn lẻ cũng như Chẵn lẻ – lẻ. Trên thực tế, bit này không được sử dụng rộng rãi nên không bắt buộc.

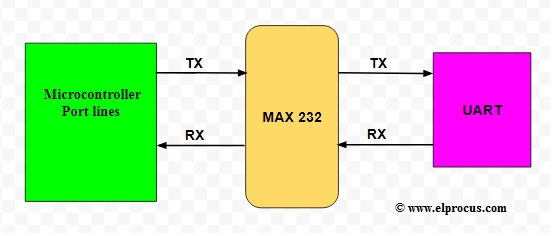
**Dữ liệu bit hoặc khung dữ liệu**

Các bit dữ liệu bao gồm dữ liệu thực được truyền từ người gửi đến người nhận. Độ dài khung dữ liệu có thể nằm trong khoảng 5 & 8. Nếu bit chẵn lẻ không được sử dụng thì chiều dài khung dữ liệu có thể dài 9 bit. Nói chung, LSB của dữ liệu được truyền trước tiên sau đó nó rất hữu ích cho việc truyền.

**4.1.5 Giao diện UART**

Hình dưới đây cho thấy UART giao tiếp với vi điều khiển. Giao tiếp UART có thể được thực hiện bằng ba tín hiệu như TXD, RXD và GND.

Bằng cách sử dụng điều này, chúng ta có thể hiển thị một văn bản trong máy tính cá nhân từ board vi điều khiển 8051 cũng như mô-đun UART. Trong board 8051, có hai giao diện nối tiếp như UART0 và UART1. Ở đây, giao diện UART0 được sử dụng. Chân Tx truyền thông tin đến chân PC & Rx nhận thông tin từ PC. Tốc độ Baud có thể được sử dụng để biểu thị tốc độ của cả vi điều khiển và PC. Việc truyền và nhận dữ liệu có thể được thực hiện đúng khi tốc độ truyền của cả vi điều khiển và PC là tương tự nhau.



Giao diện UART

**4.1.6 Các ứng dụng của UART**

UART thường được sử dụng trong các bộ vi điều khiển cho các yêu cầu chính xác và chúng cũng có sẵn trong các thiết bị liên lạc khác nhau như giao tiếp không dây, thiết bị GPS, mô-đun Bluetooth và nhiều ứng dụng khác.

Các tiêu chuẩn truyền thông như RS422 & TIA được sử dụng trong UART ngoại trừ RS232. Thông thường, UART là một IC riêng được sử dụng trong giao tiếp nối tiếp UART.

**4.1.7 Ưu và nhược điểm của UART**

Không có giao thức truyền thông nào là hoàn hảo, nhưng UART thực hiện khá tốt công việc của nó. Dưới đây là một số ưu và nhược điểm để giúp bạn quyết định xem nó có phù hợp với nhu cầu của bạn hay không:

**Ưu điểm**

* Chỉ sử dụng hai dây
* Không cần tín hiệu clock
* Có một bit chẵn lẻ để cho phép kiểm tra lỗi
* Cấu trúc của gói dữ liệu có thể được thay đổi miễn là cả hai bên đều được thiết lập cho nó
* Phương pháp có nhiều tài liệu và được sử dụng rộng rãi

**Nhược điểm**

* Kích thước của khung dữ liệu được giới hạn tối đa là 9 bit
* Không hỗ trợ nhiều hệ thống slave hoặc nhiều hệ thống master
* Tốc độ truyền của mỗi UART phải nằm trong khoảng 10% của nhau

### **4.2 SPI**

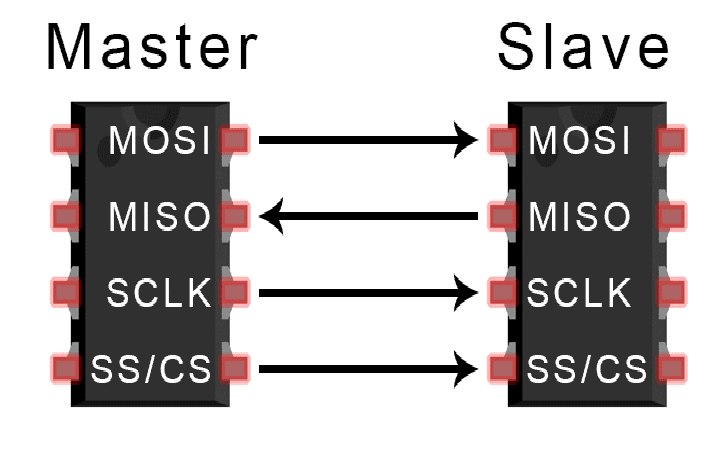
**4.2.1 Giới thiệu về giao tiếp SPI**

SPI (Serial Peripheral Bus) là một chuẩn truyền thông nối tiếp tốc độ cao do hãng Motorola đề xuất, một giao thức giao tiếp phổ biến được sử dụng bởi nhiều thiết bị khác nhau.

* Ví dụ, module thẻ SD, module đầu đọc thẻ RFID và bộ phát / thu không dây 2,4 GHz đều sử dụng SPI để giao tiếp với vi điều khiển.

Lợi ích duy nhất của SPI là dữ liệu có thể được truyền mà không bị gián đoạn. Bất kỳ số lượng bit nào cũng có thể được gửi hoặc nhận trong một luồng liên tục. Với I2C và UART, dữ liệu được gửi dưới dạng gói, giới hạn ở một số bit cụ thể. Điều kiện bắt đầu và dừng xác định điểm bắt đầu và kết thúc của mỗi gói, do đó dữ liệu bị gián đoạn trong quá trình truyền.

Các thiết bị giao tiếp qua SPI có quan hệ master - slave. Master là thiết bị điều khiển (thường là vi điều khiển), còn slave (thường là cảm biến, màn hình hoặc chip nhớ) nhận lệnh từ master. Cấu hình đơn giản nhất của SPI là hệ thống một slave, một master duy nhất, nhưng một master có thể điều khiển nhiều hơn một slave.



* OSI (đầu ra master / đầu vào slave) - đường truyền cho master gửi dữ liệu đến slave.
* MISO (đầu vào master / đầu ra slave) - đường cho slave gửi dữ liệu đến master.
* SCLK (clock) - đường cho tín hiệu xung nhịp.
* SS / CS (Slave Select / Chip Select) - đường cho master chọn slave nào để gởi tín hiệu.

**4.2.2 Cách hoạt động cuả SPI**

**Xung nhịp**

Tín hiệu xung nhịp đồng bộ hóa đầu ra của các bit dữ liệu từ master để lấy mẫu các bit của slave. Một bit dữ liệu được truyền trong mỗi chu kỳ xung nhịp, do đó tốc độ truyền dữ liệu được xác định bởi tần số của tín hiệu xung nhịp. Giao tiếp SPI được khởi tạo bởi master kể từ khi master cấu hình và tạo ra tín hiệu xung nhịp.

Bất kỳ giao thức giao tiếp nào mà các thiết bị chia sẻ tín hiệu xung nhịp thì đều được gọi là đồng bộ. SPI là một giao thức giao tiếp đồng bộ. Ngoài ra còn có các phương thức không đồng bộ không sử dụng tín hiệu xung nhịp. Ví dụ, trong giao tiếp UART, cả hai bên đều được đặt thành tốc độ truyền được cấu hình sẵn để chỉ ra tốc độ và thời gian truyền dữ liệu.

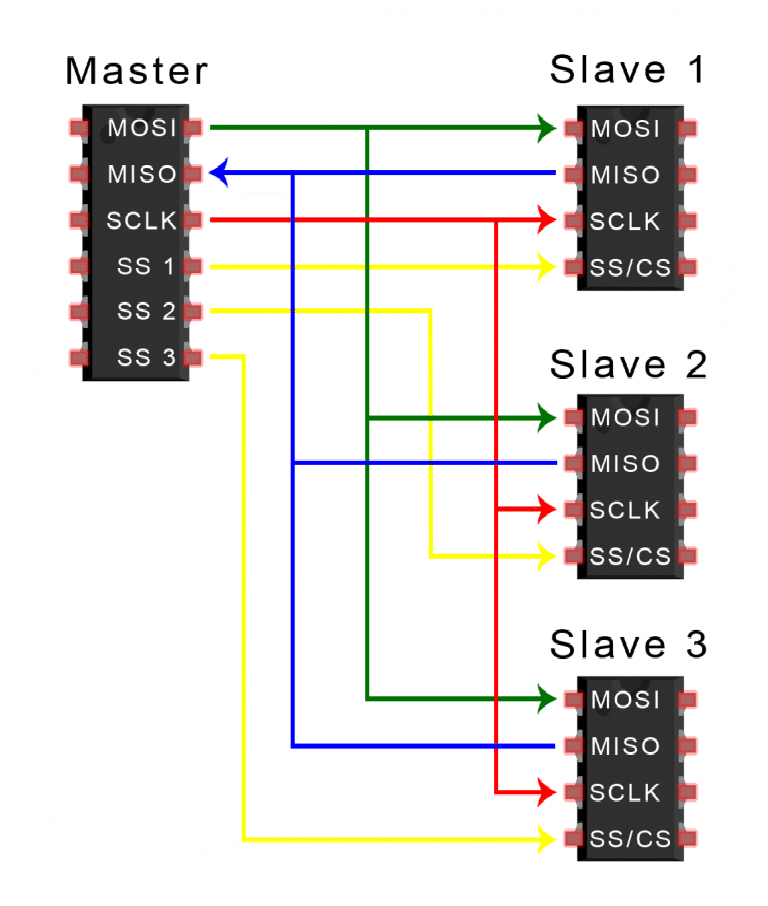
Tín hiệu xung nhịp trong SPI có thể được sửa bằng cách sử dụng các thuộc tính của phân cực xung nhịp và pha xung nhịp. Hai thuộc tính này làm việc cùng nhau để xác định khi nào các bit được xuất ra và khi được lấy mẫu. Phân cực xung nhịp có thể được thiết lập bởi master để cho phép các bit được xuất ra và lấy mẫu trên cạnh lên hoặc xuống của chu kỳ xung nhịp. Pha xung nhịp có thể được đặt để đầu ra và lấy mẫu xảy ra trên cạnh đầu tiên hoặc cạnh thứ hai của chu kỳ xung nhịp, bất kể nó đang tăng hay giảm.

**Slave Select**

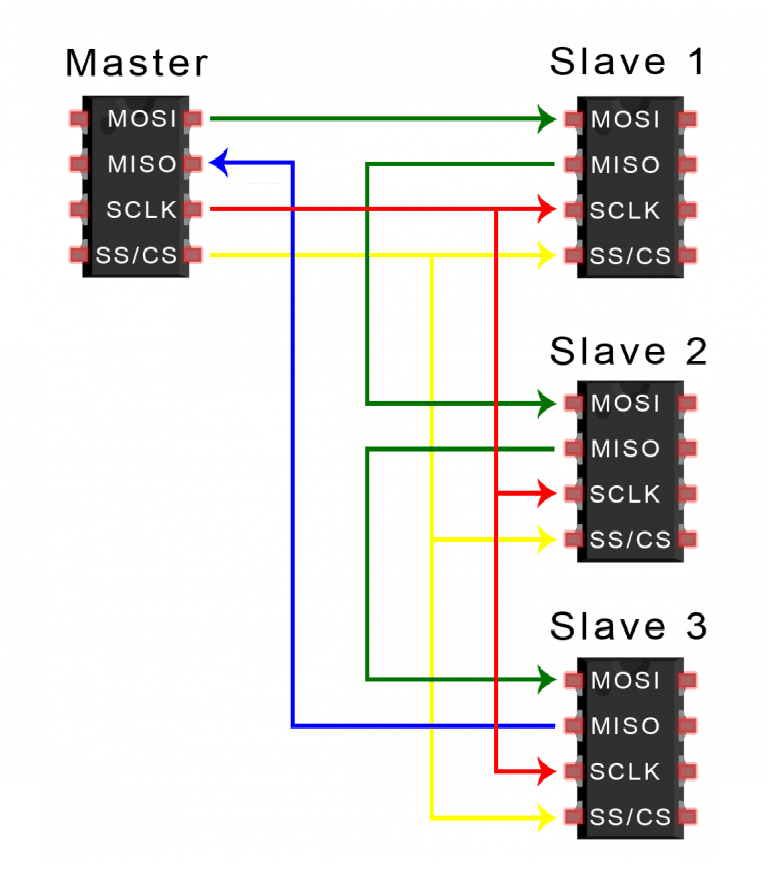
Master có thể chọn slave mà nó muốn giao tiếp bằng cách đặt đường CS / SS của slave ở mức điện áp thấp. Ở trạng thái idle, không truyền tải, dòng slave select được giữ ở mức điện áp cao. Nhiều chân CS / SS có thể có sẵn trên thiết bị master cho phép đấu dây song song nhiều slave. Nếu chỉ có một chân CS / SS, nhiều slave có thể được kết nối với master bằng cách nối chuỗi.

**Nhiều slave**

SPI có thể thiết lập để hoạt động với một master và một slave duy nhất, và nó có thể được thiết lập với nhiều slave do một master duy nhất điều khiển. Có hai cách để kết nối nhiều slave với master. Nếu master có nhiều chân slave select, các slave có thể được nối dây song song như thế này:



Nếu chỉ có một chân slave select, các slave có thể được nối chuỗi như sau:



**MOSI và MISO**

Master gửi dữ liệu đến slave từng bit, nối tiếp qua đường MOSI. Slave nhận dữ liệu được gửi từ master tại chân MOSI. Dữ liệu được gửi từ master đến slave thường được gửi với bit quan trọng nhất trước.

Slave cũng có thể gửi dữ liệu trở lại master thông qua đường MISO nối tiếp. Dữ liệu được gửi từ slave trở lại master thường được gửi với bit ít quan trọng nhất trước.

**4.2.3 Các bước truyền dữ liệu SPI**

1. Master ra tín hiệu xung nhịp.
2. Master chuyển chân SS / CS sang trạng thái điện áp thấp, điều này sẽ kích hoạt slave.
3. Master gửi dữ liệu từng bit một tới slave dọc theo đường MOSI. Slave đọc các bit khi nó nhận được.
4. Nếu cần phản hồi, slave sẽ trả lại dữ liệu từng bit một cho master dọc theo đường MISO. Master đọc các bit khi nó nhận được.

**4.2.4 Ưu và nhược điểm của SPI**

Có một số ưu và nhược điểm khi sử dụng SPI và nếu được lựa chọn giữa các giao thức giao tiếp khác nhau, bạn nên biết khi nào sử dụng SPI theo yêu cầu của dự án:

**Ưu điểm**

* Không có bit bắt đầu và dừng, vì vậy dữ liệu có thể được truyền liên tục mà không bị gián đoạn
* Không có hệ thống định địa chỉ slave phức tạp như I2C
* Tốc độ truyền dữ liệu cao hơn I2C (nhanh gần gấp đôi)
* Các đường MISO và MOSI riêng biệt, vì vậy dữ liệu có thể được gửi và nhận cùng một lúc

**Nhược điểm**

* Sử dụng bốn dây (I2C và UART sử dụng hai)
* Không xác nhận dữ liệu đã được nhận thành công (I2C có điều này)
* Không có hình thức kiểm tra lỗi như bit chẵn lẻ trong UART
* Chỉ cho phép một master duy nhất

### **4.3 I2C**

**4.3.1 Khái niệm**

I2C là tên viết tắt của cụm từ tiếng anh “Inter-Integrated Circuit”, là sự kết hợp các tính năng tốt nhất của SPI và UART.

Nó là một giao thức giao tiếp được phát triển bởi Philips Semiconductors để truyền dữ liệu giữa một bộ xử lý trung tâm với nhiều IC trên cùng một board mạch chỉ sử dụng hai đường truyền tín hiệu.

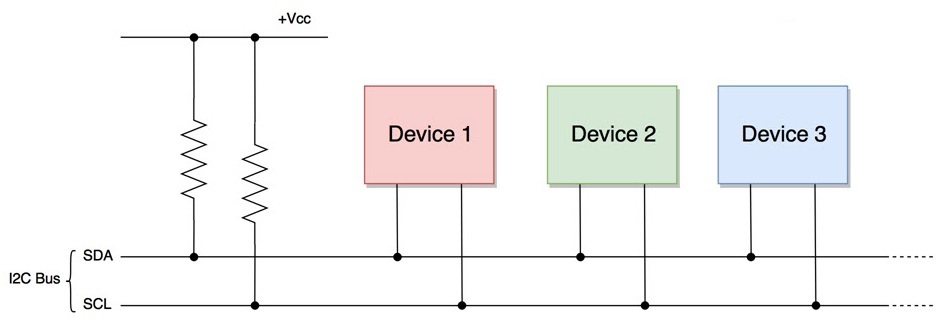
Do tính đơn giản của nó nên loại giao thức này được sử dụng rộng rãi cho giao tiếp giữa vi điều khiển và mảng cảm biến, các thiết bị hiển thị, thiết bị IoT, EEPROMs, v.v …

Đây là một loại giao thức giao tiếp nối tiếp đồng bộ. Nó có nghĩa là các bit dữ liệu được truyền từng bit một theo các khoảng thời gian đều đặn được thiết lập bởi một tín hiệu đồng hồ tham chiếu.

**4.3.2 PHẦN CỨNG**

**a. Bus vật lý I2C**

Bus I2C (dây giao tiếp) chỉ gồm hai dây và được đặt tên là Serial Clock Line (SCL) và Serial Data Line (SDA). Dữ liệu được truyền đi được gửi qua dây SDA và được đồng bộ với tín hiệu đồng hồ (clock) từ SCL. Tất cả các thiết bị / IC trên mạng I2C được kết nối với cùng đường SCL và SDA như sau:



Cả hai đường bus I2C (SDA, SCL) đều hoạt động như các bộ lái cực máng hở (open drain). Nó có nghĩa là bất kỳ thiết bị / IC trên mạng I2C có thể lái SDA và SCL xuống mức thấp, nhưng không thể lái chúng lên mức cao. Vì vậy, một điện trở kéo lên (khoảng 1 kΩ đến 4,7 kΩ) được sử dụng cho mỗi đường bus, để giữ cho chúng ở mức cao (ở điện áp dương) theo mặc định.

Lý do sử dụng một hệ thống cực máng hở (open drain) là để không xảy ra hiện tượng ngắn mạch, điều này có thể xảy ra khi một thiết bị cố gắng kéo đường dây lên cao và một số thiết bị khác cố gắng kéo đường dây xuống thấp.

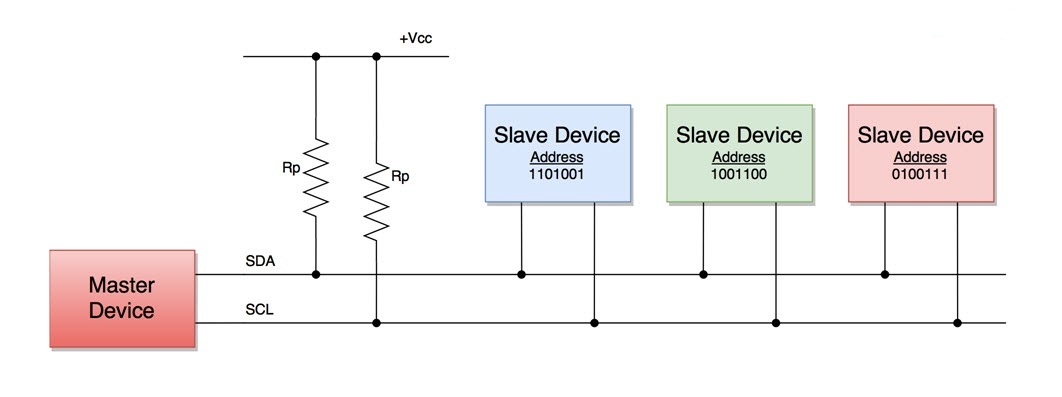
**b. Thiết bị chủ (Master) và tớ (Slave)**

Các thiết bị kết nối với bus I2C được phân loại hoặc là thiết bị Chủ (Master) hoặc là thiết bị Tớ (Slave). Ở bất cứ thời điểm nào thì chỉ có duy nhất một thiết bị Master ở trang thái hoạt động trên bus I2C. Nó điều khiển đường tín hiệu đồng hồ SCL và quyết định hoạt động nào sẽ được thực hiện trên đường dữ liệu SDA.

Tất cả các thiết bị đáp ứng các hướng dẫn từ thiết bị Master này đều là Slave. Để phân biệt giữa nhiều thiết bị Slave được kết nối với cùng một bus I2C, mỗi thiết bị Slave được gán một địa chỉ vật lý 7-bit cố định.

Khi một thiết bị Master muốn truyền dữ liệu đến hoặc nhận dữ liệu từ một thiết bị Slave, nó xác định địa chỉ thiết bị Slave cụ thể này trên đường SDA và sau đó tiến hành truyền dữ liệu. Vì vậy, giao tiếp có hiệu quả diễn ra giữa thiết bị Master và một thiết bị Slave cụ thể.

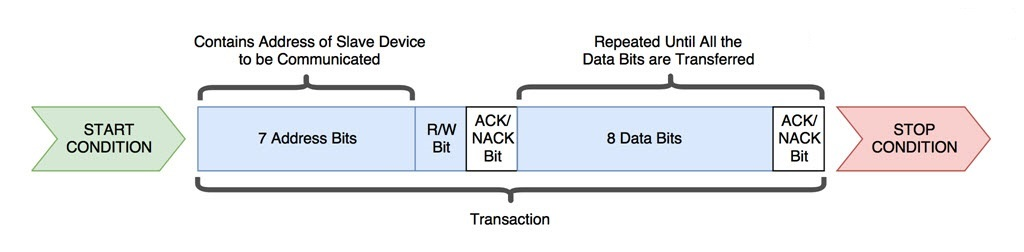
Tất cả các thiết bị Slave khác không phản hồi trừ khi địa chỉ của chúng được chỉ định bởi thiết bị Master trên dòng SDA.



**c. Giao thức truyền dữ liệu**

Giao thức sau đây (tập hợp các quy tắc) được theo sau bởi thiết bị Master và các thiết bị Slave để truyền dữ liệu giữa chúng.

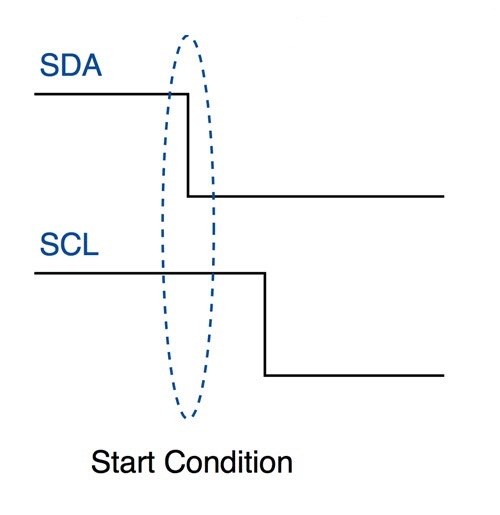
Dữ liệu được truyền giữa thiết bị Master và các thiết bị Slave thông qua một đường dữ liệu SDA duy nhất, thông qua các chuỗi có cấu trúc gồm các số 0 và 1 (bit). Mỗi chuỗi số 0 và 1 được gọi là giao dịch (transaction) và dữ liệu trong mỗi giao dịch có cấu trúc như sau:



**Điều kiện bắt đầu (Start Condition)**

Bất cứ khi nào một thiết bị chủ / IC quyết định bắt đầu một giao dịch, nó sẽ chuyển mạch SDA từ mức điện áp cao xuống mức điện áp thấp trước khi đường SCL chuyển từ cao xuống thấp.

Khi điều kiện bắt đầu được gửi bởi thiết bị Master, tất cả các thiết bị Slave đều hoạt động ngay cả khi chúng ở chế độ ngủ (sleep mode) và đợi bit địa chỉ.



**Khối địa chỉ**

Nó bao gồm 7 bit và được lấp đầy với địa chỉ của thiết bị Slave đến / từ đó thiết bị Master cần gửi / nhận dữ liệu. Tất cả các thiết bị Slave trên bus I2C so sánh các bit địa chỉ này với địa chỉ của chúng.

**Bit Read / Write**

Bit này xác định hướng truyền dữ liệu. Nếu thiết bị Master / IC cần gửi dữ liệu đến thiết bị Slave, bit này được thiết lập là ‘0’. Nếu IC Master cần nhận dữ liệu từ thiết bị Slave, bit này được thiết lập là ‘1’.

**Bit ACK / NACK**

ACK / NACK là viết tắt của Acknowledged/Not-Acknowledged. Nếu địa chỉ vật lý của bất kỳ thiết bị Slave nào trùng với địa chỉ được thiết bị Master phát, giá trị của bit này được set là ‘0’ bởi thiết bị Slave. Ngược lại, nó vẫn ở mức logic ‘1’ (mặc định).

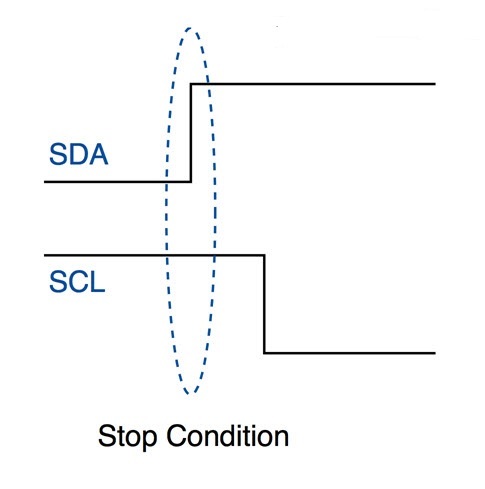
**Khối dữ liệu**

Nó bao gồm 8 bit và chúng được thiết lập bởi bên gửi, với các bit dữ liệu cần truyền tới bên nhận. Khối này được theo sau bởi một bit ACK / NACK và được set thành ‘0’ bởi bên nhận nếu nó nhận thành công dữ liệu. Ngược lại, nó vẫn ở mức logic ‘1’.

Sự kết hợp của khối dữ liệu theo sau bởi bit ACK / NACK được lặp lại cho đến quá trình truyền dữ liệu được hoàn tất.

**Điều kiện kết thúc (Stop condition)**

Sau khi các khung dữ liệu cần thiết được truyền qua đường SDA, thiết bị Master chuyển đường SDA từ mức điện áp thấp sang mức điện áp cao trước khi đường SCL chuyển từ cao xuống thấp.



**d. Giao tiếp I2C hoạt động như thế nào?**

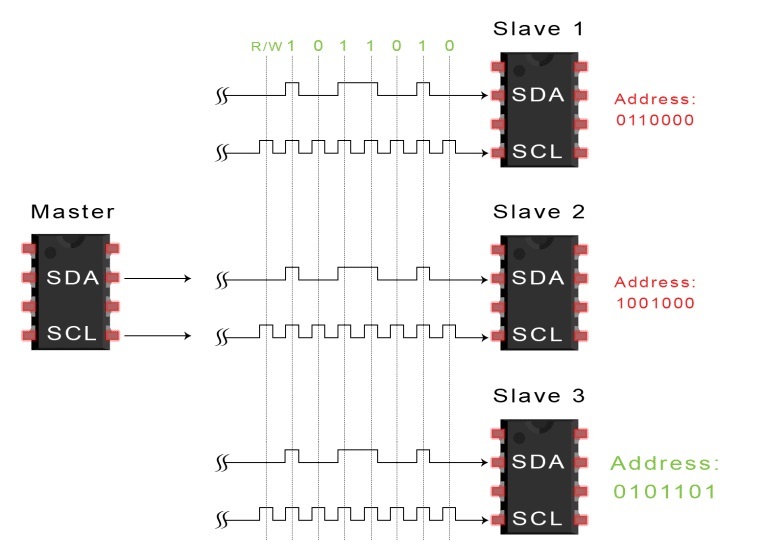
Giao tiếp I2C được bắt đầu bởi thiết bị Master hoặc để gửi dữ liệu đến thiết bị Slave hoặc nhận dữ liệu từ thiết bị đó. Chúng ta hãy tìm hiểu về cách làm việc của cả hai kịch bản một cách chi tiết.

**Gửi dữ liệu đến thiết bị Slave:**

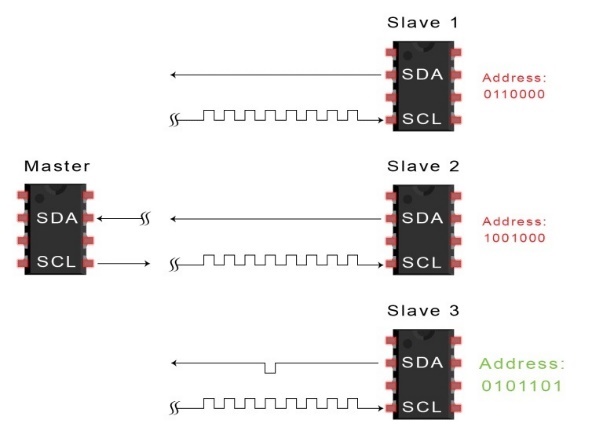
Trình tự hoạt động sau đây diễn ra khi một thiết bị Master gửi dữ liệu đến một thiết bị Slave cụ thể thông qua bus I2C:

Thiết bị Master gửi điều kiện bắt đầu đến tất cả các thiết bị Slave

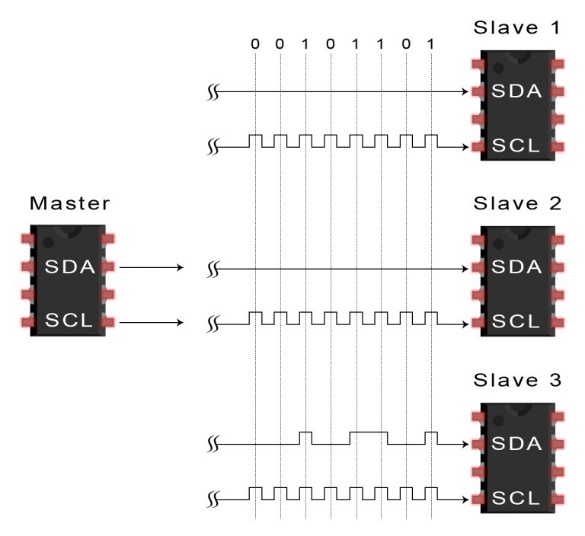
Thiết bị Master gửi 7 bit địa chỉ của thiết bị Slave mà thiết bị Master muốn giao tiếp cùng với bit Read/Write



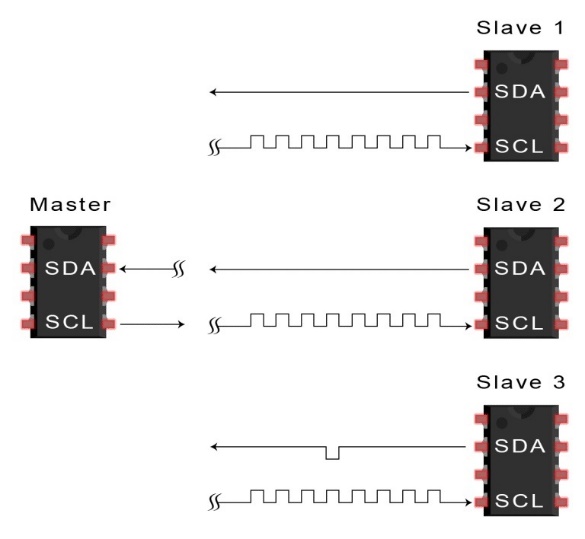
Mỗi thiết bị Slave so sánh địa chỉ được gửi từ thiết bị Master đến địa chỉ riêng của nó. Nếu địa chỉ trùng khớp, thiết bị Slave gửi về một bit ACK bằng cách kéo đường SDA xuống thấp và bit ACK / NACK được thiết lập là ‘0’,. Nếu địa chỉ từ thiết bị Master không khớp với địa chỉ riêng của thiết bị Slave thì đường SDA ở mức cao và bit ACK / NACK sẽ ở mức ‘1’ (mặc định).



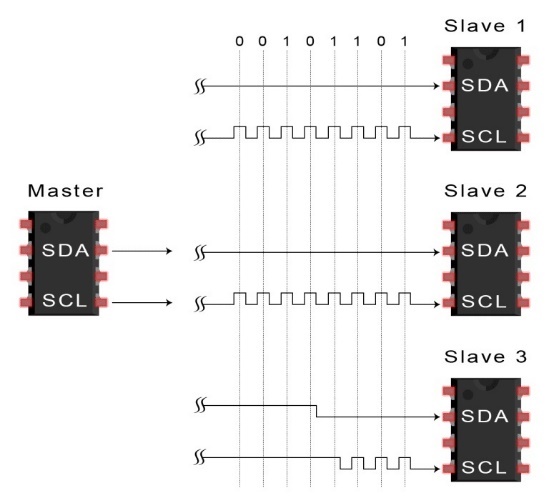
Thiết bị Master gửi hoặc nhận khung dữ liệu. Nếu thiết bị Master muốn gửi dữ liệu đến thiết bị Slave, bit Read / Write là mức điện áp thấp. Nếu thiết bị Master đang nhận dữ liệu từ thiết bị Slave, bit này là mức điện áp cao.



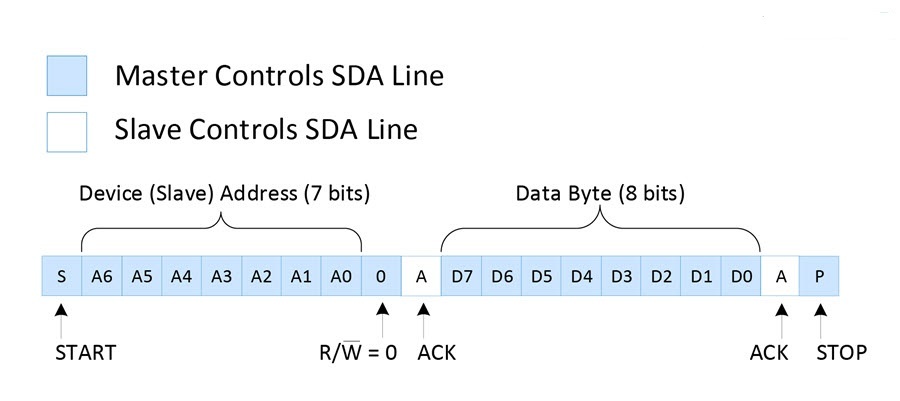
Nếu khung dữ liệu được thiết bị Slave nhận được thành công, nó sẽ thiết lập bit ACK / NACK thành ‘0’, báo hiệu cho thiết bị Master tiếp tục



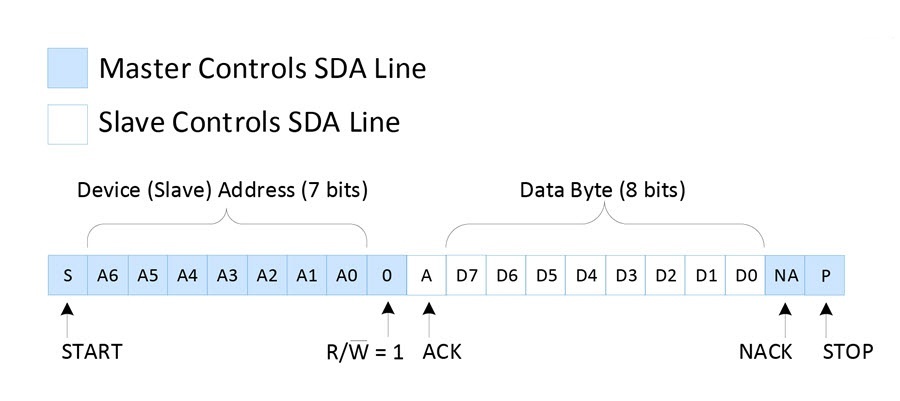
Sau khi tất cả dữ liệu được gửi đến thiết bị Slave, thiết bị Master gửi điều kiện dừng để báo hiệu cho tất cả các thiết bị Slave biết rằng việc truyền dữ liệu đã kết thúc.



Hình dưới đây thể hiện toàn bộ các bit dữ liệu được gửi trên đường SDA và thiết bị điều khiển chúng khi thiết bị Master gửi dữ liệu đến thiết bị Slave.



Hình dưới đây thể hiện toàn bộ các bit dữ liệu được gửi trên đường SDA và thiết bị điều khiển chúng khi thiết bị Master nhận dữ liệu đến thiết bị Slave.



**e. Các chế độ hoạt động của I²C**

Dựa vào tốc độ ta chia làm 2 loại

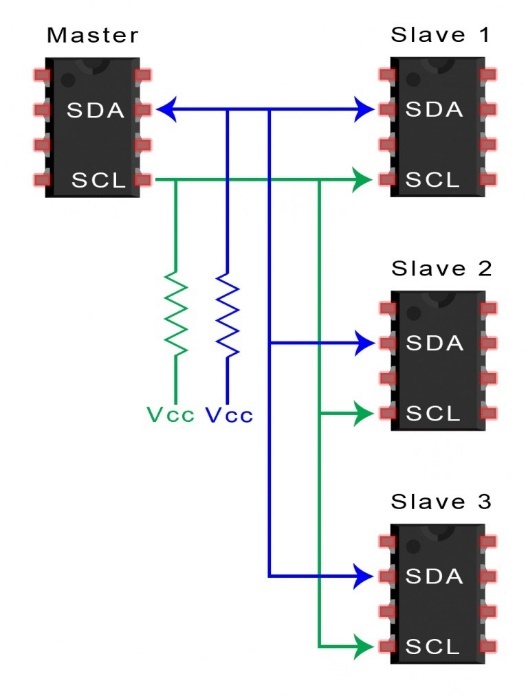
* Chế độ chuẩn (standard mode) hoạt động ở tốc độ 100 Kbit/s.
* Chế độ tốc độ thấp (low-speed mode) hoạt động ở tốc độ 10 Kbit/s.

Nếu chia theo quan hệ chủ tớ:

* Một Master một Slave.
* Một Master nhiều Slave.
* Nhiều Master nhiều Slave.

**Một Master nhiều Slave**

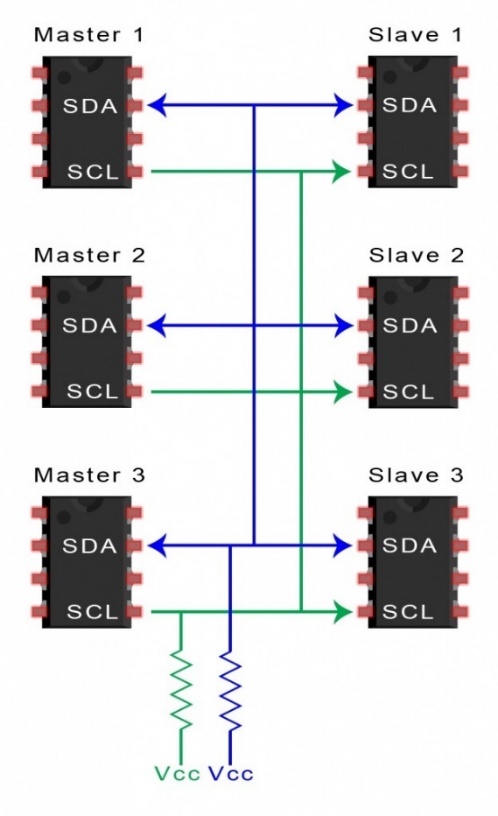
Bởi vì I2C sử dụng địa chỉ, nhiều thiết bị Slave có thể được điều khiển từ một thiết bị Master duy nhất. Với 7 bit địa chỉ tương ứng 128 (27) địa chỉ duy nhất có sẵn.  Để kết nối nhiều thiết bị Slave với một một thiết bị Master duy nhất, hãy nối dây như hình bên dưới, với các điện trở kéo lên 4.7 kΩ kết nối các đường SDA và SCL với Vcc:



* **Nhiều Master nhiều Slave.**

Nhiều thiết bị Master có thể được kết nối với một thiết bị Slave đơn hoặc nhiều thiết bị Slave. Vấn đề với nhiều thiết bị Master trong cùng một hệ thống xuất hiện khi hai thiết bị Master cố gắng gửi hoặc nhận dữ liệu cùng một lúc trên dòng SDA.

Để giải quyết vấn đề này, mỗi thiết bị Master cần phát hiện xem đường SDA thấp hay cao trước khi truyền một thông điệp. Nếu đường SDA thấp, điều này có nghĩa là một thiết bị Master khác có quyền điều khiển bus và thiết bị Master còn lại phải đợi để gửi tin nhắn. Nếu đường SDA cao thì an toàn để truyền tải thông điệp. Để kết nối nhiều thiết bị Master với nhiều thiết bị Slave, hãy sử dụng sơ đồ sau đây, với điện trở kéo lên 4.7 kΩ kết nối các đường SDA và SCL với Vcc:



**4.3.3 Ưu điểm và nhược điểm của I2C**

**Ưu điểm**

* Chỉ sử dụng hai dây
* Hỗ trợ nhiều master và nhiều slave
* Bit ACK / NACK xác nhận mỗi khung được chuyển thành công
* Phần cứng ít phức tạp hơn so với UART
* Giao thức nổi tiếng và được sử dụng rộng rãi

**Nhược điểm**

* Tốc độ truyền dữ liệu chậm hơn SPI
* Kích thước của khung dữ liệu bị giới hạn ở 8 bit
* Cần phần cứng phức tạp hơn để triển khai so với SP

### **4.4 LAN**

**a. LAN là gì?**

LAN là viết tắt của Local Area Network tạm dịch là mạng máy tính nội bộ, giao tiếp này cho phép các máy tính kết nối với nhau để cùng làm việc và chia sẻ dữ liệu. Kết nối này được thực hiện thông qua sợi cáp LAN hoặc Wifi (không dây) trong không gian hẹp, chính vì thế nó chỉ có thể sử dụng được trong một phạm vi giới hạn như phòng làm việc, trong nhà, trường học…

**b. Cổng mạng LAN (RJ45) trên máy tính, laptop là gì?**

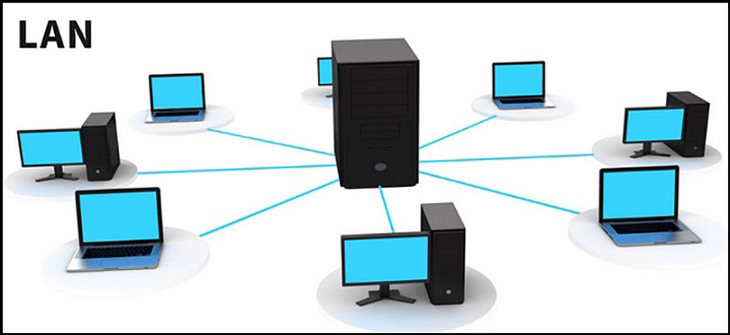
Trước khi tìm hiểu sâu hơn về kết nối LAN. Chúng ta cùng nói một chút về tên gọi "cổng mạng LAN" trên laptop hiện nay. Vì kết nối không dây ra mắt và được phổ biến khá lâu sau kết nối có dây (cáp) nên người ta thường gọi cổng kết nối cáp mạng RJ45 (Ethernet) trên laptop, máy tính là "cổng mạng LAN" hay "cổng LAN".

Hiện nay khi nói đến kết nối internet trên laptop, máy tính chúng ta có 2 loại là kết nối không dây qua wifi và kết nối có dây qua cáp mạng qua cổng RJ45 hay cổng mạng LAN.



**c. Yêu cầu để tạo kết nối LAN nội bộ là gì?**

Để tạo được mạng LAN nội bộ thì cần có một thiết bị làm máy chủ (sever), một số thiết bị hỗ trợ kết nối và cuối cùng là các máy khách

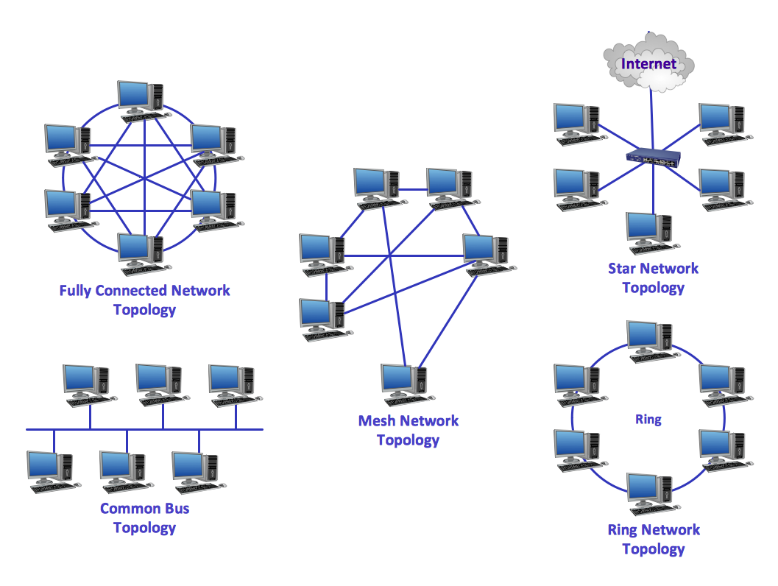


Trước khi tạo mạng LAN cần chắc chắn rằng tất cả các thiết bị đều được tích hợp sẵn card mạng NIC (Network Interface Card). Card mạng được xem là bộ thu phát tín hiệu mạng cho các thiết bị muốn kết nối với mạng LAN, card mạng thường được tích hợp sẵn trong laptop, máy tính,…

**d. Các kiểu Topology của mạng LAN**

Topology của mạng LAN thực chất là kiểu bố trí các phần tử trong cùng một mạng và là cách để chúng kết nối với nhau. Topology có rất nhiều loại:

* Star
* Ring
* Mesh
* Bus
* Tree

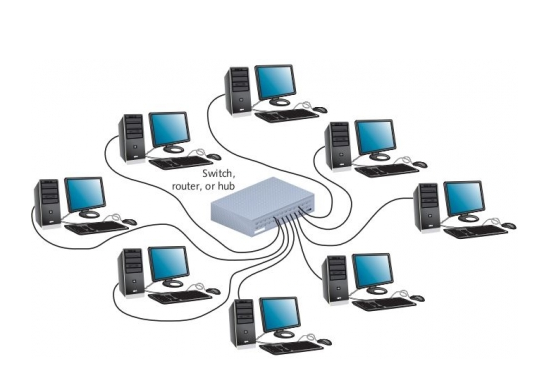


Nhưng đa số chúng được hình thành từ 3 loại Topology cơ bản sau: Mạng hình sao (Star Topology), mạng dạng vòng (Ring Topology) và mạng dạng tuyến (Linear Bus Topology).

**Mạng hình sao (Star Topology)**

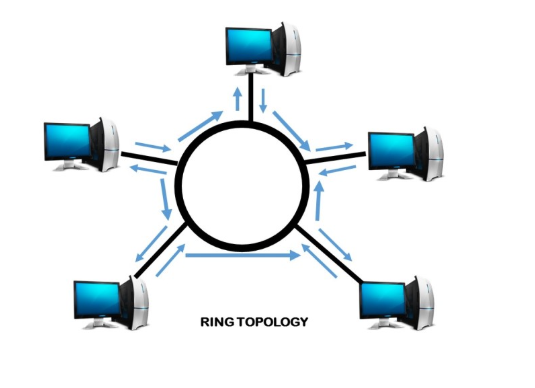
Mô hình mạng hình sao bao gồm một trung tâm và các máy trạm hay các thiết bị khác là các nút thông tin còn lại của mạng. Trong đó trung tâm đóng vai trò điều khiển tất cả các hoạt động của mạng:

* Thông báo các trạng thái của mạng.
* Xác định cặp địa chỉ gửi và địa chỉ nhận và cho phép chúng chiếm thông tin để liên lạc với nhau.
* Theo dõi và xử lý sai trong quá trình trao đổi thông tin, ...



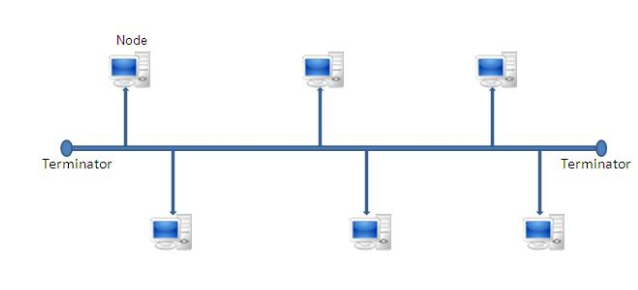
**Mạng dạng vòng (Ring Topology)**

Mạng được bố trí và lắp đặt theo kiểu một vòng tròn khép kín. Các tín hiệu được truyền theo một chiều nào đó, tuy nhiên mỗi một thời điểm các máy trạm chỉ được truyền tín hiệu qua một nút mà thôi và thông tin chuyển đi phải có đính kèm địa chỉ cụ thể của mỗi trạm tiếp nhận.



**Mạng dạng định tuyến (Linear Bus Topology)**

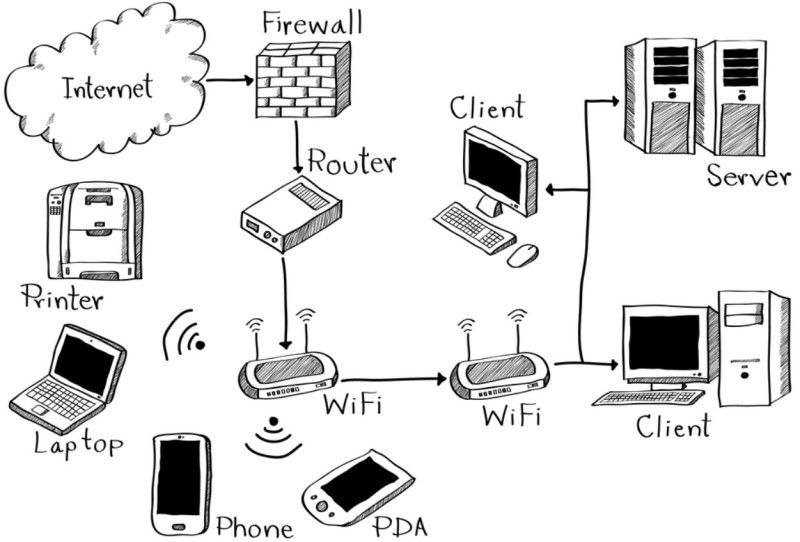
Mạng dạng tuyến được bố trí theo kiểu hành lang mà ở đó các thiết bị được ghép nối với nhau trên một đường trục chính để truyền tải dữ liệu và ở 2 đầu của trục chính được bịt kín bởi 2 thiết bị Terminator. Các dữ liệu được truyền trong mạng khi di chuyển lên hoặc xuống trong mạng đều cần mang theo địa chỉ nơi đến.



**h. Các thành phần của mạng LAN**

Hệ thống mạng LAN khá đa dạng thành phần, cụ thể như sau:

* **Máy chủ (Server)**: Đóng vai trò trung tâm, quản lý việc kết nối, chia sẻ thông tin giữa các máy trạm. Tuy nhiên, trong mạng LAN ngang hàng, không có máy chủ – máy khách, tất cả các máy đều ngang hàng, bình đẳng.
* **Máy khách/ máy trạm (client)**: Bao gồm các thiết bị *(PC, Laptop, Smartphone…)* kết nối với nhau và chịu sự quản lý của máy chủ.
* **Card mạng NIC (Network Interface Card)**: Là một bản mạch được cắm trong khe của bản mạch chính máy tính, làm nhiệm vụ cung cấp khả năng truyền thông mạng cho một máy tính. Thiết bị này được tích hợp sẵn trong các laptop hiện nay.
* **Cáp mạng (cable)**: Là phần cứng mạng, được làm từ kim loại hay hợp kim, có nhiệm vụ truyền dẫn tín hiệu giữa các thiết bị trong hệ thống.
* **Repeater**: Là bộ mở rộng wifi, có nhiệm vụ mở rộng vùng phủ sóng wifi. Giới hạn truyền trong hệ thống mạng cục bộ thường chỉ được 100m, nhưng có repeater thì khoảng cách sẽ được mở rộng.
* **Bộ định tuyến (router)**: Là thiết bị mạng giúp chuyển tiếp các gói dữ liệu giữa các mạng máy tính. Nói cách khác, nó có thể chia sẻ internet tới nhiều thiết bị khác nhau trong cùng 1 lớp mạng.
* **Bộ chuyển mạch (switch)**: Thiết bị này giống bridge, nhưng có nhiều cổng hơn, giúp liên kết nhiều segment lại với nhau.
* **Hub**: Chức năng tương tự như repeater, nhưng nó có nhiều cổng hơn, giúp khuếch đại tín hiệu từ một cổng đến nhiều cổng khác.
* **Cầu nối (bridge)**: Là cầu nối gắn kết nhiều mạng LAN lại với nhau, tạo thành một mạng LAN lớn hơn.
* **Cổng giao tiếp (gateway)**: Là thiết bị có khả năng nối ghép hai loại giao thức với nhau. Nói cách khác, nó là phương tiện giao tiếp giữa các giao thức, cho phép chúng có thể trao đổi thông tin trên cùng một thiết bị máy tính hoặc trong các hệ thống máy tính khác nhau.



**e. Công dụng của mạng LAN**

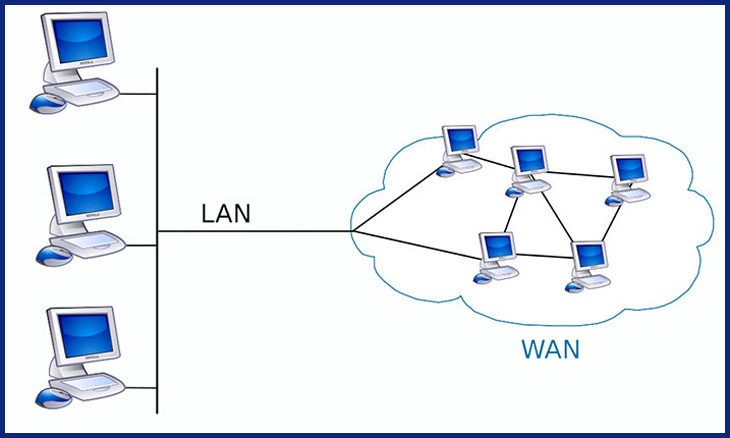
Mạng LAN cho phép các thiết bị laptop, điện thoại, máy tính bảng có thể kết nối với nhau. Các lợi ích công nghệ mạng LAN mang lại bao gồm:

* Cho phép truy cập vào các ứng dụng tập trung nằm trên các máy chủ
* Cho phép tất cả các thiết bị lưu trữ dữ liệu quan trọng của doanh nghiệp ở một vị trí tập trung
* Bảo vệ các thiết bị kết nối mạng LAN bằng các công cụ bảo mật mạng.
* Cho phép chia sẻ tài nguyên, bao gồm máy in, ứng dụng và các dịch vụ được chia sẻ khác
* Cho phép nhiều thiết bị trong một mạng LAN chia sẻ một kết nối internet duy nhất.

**f. Các loại kết nối trong mạng LAN**

Các thiết bị trong cùng mạng LAN có thể kết nối với nhau thông qua sợi cáp mạng. Các mạng LAN có thể kết nối với nhau tạo thành một mạng lưới rộng lớn hơn được gọi là WAN (Wide Area Network) và để giao tiếp với nhau, các thiết bị thường được kết nối với một hoặc vài bộ phát wifi (Router).

Ngoài ra, mạng LAN còn có thể được thiết lập bằng cổng kết nối không dây (Wireless) và được gọi chung là WLAN (Wireless LAN), hay chúng ta thường gọi là Wifi.



# **II. Phần mềm hệ thống nhúng**

## 1. Hệ điều hành nhúng RTOS, android, linux

### **1.1 Hệ điều hành RTOS**

**Định nghĩa**

RTOS (real-time operating systems)

Phần mềm điều khiển chuyên dụng thường được dùng trong những ứng dụng điện toán nhúng có tài nguyên bộ nhớ hạn chế và yêu cầu ngặt nghèo về thời gian đáp ứng tức thời, tính sẵn sàng cao và khả năng tự kiểm soát một cách chính xác.

Hầu hết các hệ thống thời gian thực không cung cấp các tính năng được tìm thấy trong một hệ thống máy tính nguyên nhân là vì:

* Các hệ thống thời gian thực thường chỉ thực hiện một nhiệm vụ đơn lẻ.
* Hệ thống thời gian thực không tiếp xúc trực tiếp với người dùng.
* Các tính năng tìm thấy trong một hệ thống pc yêu cầu phần cấu hình tối hiểu cao hơn những gì có trong một hệ thống thời gian thực.

**Ứng dụng**

* Động cơ xe
* Máy nhắn tin
* Điện thoại di động
* Thiết bị y tế
* Thiết bị đo lượng điều khiển công nghiệp

….

**Một số đặc trưng quan trọng của một hệ thống thời gian thực**

* Khả năng tách biết đối với các ứng dụng khác: nếu một chương trình gặp vấn đề trong việc thực thi thì RTOS sẽ nhanh chóng cô lập chương trình này và kích hoạt cơ chế phục hồi và bảo vệ các chương trình khác, tính năng tương tự cũng được ứng dụng để tránh tình trạng tràn bộ nhớ do bất cứ chương trình này gây ra.
* Cho phép đa tác cụ.
* Lịch trình các tác vụ có sự ưu tiên.
* Đồng bộ hóa truy cập tài nguyên.
* Giao tiếp liên tác vụ.
* Dự đoán thời gian.
* Ngắt xử lý.

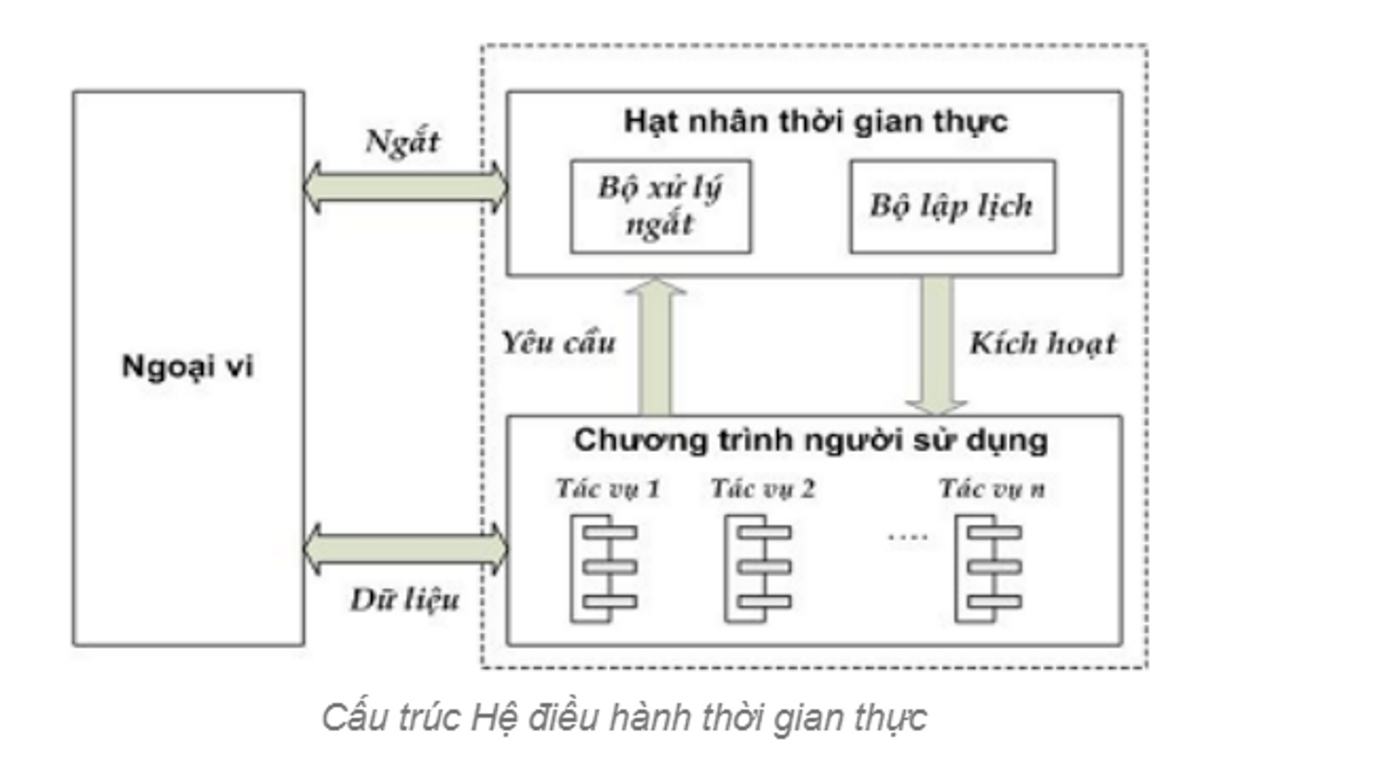
**Có 2 loại rtos**

-HARD REAL-TIME SYSTEM

→ Thời gian là vô cùng quan trọng và thơi hạn là không thể trễ ( nếu trễ deadline thì nó được coi là một lỗi vô cùng nghiêm trọng)

-SOFT REAL-TIME SYSTEM

→ Sự trễ thời hạn là một điều không nên được xảy ra, tuy nhiên nếu một số sự trễ sẽ không ảnh hưởng quá nghiêm trọng



**Những cách để đồng bộ hóa giữa các quá trình trong rtos**

* Semaphone
* Event
* Message Queue
* Message Mail Box

**RTOS Kernel**

KERNEL- là một phần của hệ thống đa nhiệm chịu trách nhiệm về quản lý các tác vụ (nghĩa là để quản lý thời gian của CPU) và giao tiếp giữa các tác vụ.

**RTOS Non Preemptive Kernel**

* RTOS Non Preemptive Kernel yêu cầu mỗi nhiệm vụ phải hoàn toàn không kiểm soát CPU
* Nhằm duy trì sự đồng thời của các task quá trình này phải được xảy ra thưỡng xuyên. Non-preemptive scheduling cũng được gọi là hợp tác đa nhiệm vụ để chia sẻ tài nguyên cpu với nhau
* Một trong nhưng lợi ích của quá trình này chí là khả năng gián đoạn vô cùng thấp
* Một trong những hạn chế quan trong nhất của quá trình này là khả năng đáp ứng bởi vì một nhiệm vụ ưu tiên hơn đã làm xong và sẵn sàng để chạy hoàn toàn có thể phải đợi rất lâu bởi vì một nhiệm vụ khác đang sử dụng cpu. Do vậy nên là thời gian thực thi của một nhiệm vụ ưu tiên sẽ không được xác định. Bạn không bao giờ thực sự biết khi nào nhiệm vụ ưu tiên cao nhất sẽ kiểm soát CPU nếu chương trình hiện tại không từ bỏ quyền kiểm soát cpu.

**RTOS Preemptive Kernel**

* RTOS Preemptive Kernel được sử dụng khi yều cầu về đáp ứng hệ thống là quan trọng nên là hầu hết các real-time kernels thương mại sẽ đều là preemptive.
* Nhiệm vụ có mức độ ưu tiên cao hơn sẽ chiếm được quyền kiểm soát CPU trước
* Khi xuất hiện 1 nhiệm vụ có ưu tiên cao hơn thì nhiệm vụ hiện tại sẽ bị preempted và nhường lại quyền kiểm soát cpu cho nhiệm vụ có ưu tiên cao hơn.
* Nếu một ISR sẵn sàng thực hiện một tác vụ có mức ưu tiên cao hơn, thì khi ISR hoàn thành, tác vụ bị gián đoạn sẽ bị tạm dừng và tác vụ có mức ưu tiên cao hơn mới được tiếp tục.

### **1.2 Hệ điều hành nhúng android**

**Embedded android là gì?**

Embedded Android về cơ bản làm việc dựa trên linux kernel, điều đó cũng đúng đối với Stock Android ( **Stock Android** không phải là một hệ điều hành nhúng cổ điển ở dạng ban đầu. Tuy nhiên, nó đang ngày càng trở nên phổ biến như một sự lựa chọn hành đầu. Embedded android có nhiều ưu điểm giống như embedded linux, đồng thời cung cấp thêm giao diện người dùng tinh tế và giao diện lập trình quen thuộc.)

**Hoạt động**

Khác với embedded linux, khi linux kernel giao tiếp trực tiếp vơi các phần cứng cơ bản thì chính các nhà sản xuất bo mạch và các nhà sản xuất chip hay cũng chính là người giao tiếp linux kernel, họ cũng sẽ cung cấp các drivers để phần cững có thể giao tiếp được với hệ điều hành. Còn đối với embedded android, android sẽ cung cấp 1 lớp trừu tượng giữa linux kernel với các thành phần khác của kiến trúc android. Lớp này được gọi là HAL và chịu trách nhiệm vưới linex kernel

**Ứng dụng**

* Chúng được sử dụng trong các bảng điều khiển kĩ thuật số
* Thiết bị đầu cuối tác các điểm bán hàng tự động

**Thách thức**

* Có quá nhiều thiết bị phần cứng có sẵn.
* Ngoài ra để làm cho android chạy được trên các thiết bị nhúng thì chúng ta cần có sự nỗ lực và kiến thức vô cùng sâu rộng trong việc điều chỉnh hệ điều hành android.
* Có quá nhiều loại thiết bị cần được hỗi trợ trên quy mô lớn
* Những sự hỗi trợ có sẵn còn ít

**Ưu điểm**

* Mang lại trải nghiệm người dùng mượt mà
* Giao diện người dùng rất quen thuộc và chất lượng cao
* Hỗ trợ màn hình cảm ứng gốc
* Hệ sinh thái ứng dụng Android phong phú
* Ngăn xếp giao tiếp tích hợp (WiFi, Bluetooth, ngăn xếp di động)
* Tối ưu hóa cho ARM (tiêu thụ điện năng thấp)
* Cơ chế cập nhật tích hợp
* Quản lý năng lượng được tối ưu hóa bằng pin tích hợp
* Tối ưu hóa cho máy tính bảng và thiết bị cảm ứng
* Sử dụng thư viện C của riêng nó có tên là Bionic, được điều chỉnh từ GNU C. Bionic đã được tối ưu hóa cho các thiết bị có ít năng lượng và bộ nhớ hơn so với các thiết bị Linux điển hình.
* Yêu cầu thời gian chạy Android để thực thi các ứng dụng

### **1.3 Hệ điều hành linux**

**Embedded linux là gì?**

Là một loại hệ điều hành linux kernel được thiết kế và cài đặt trên các thiết bị nhúng, chúng được tùy chỉnh và cài đặt riêng cho các hệ thống nhúng vì vậy chúng có kích thước nhỏ hơn đồng thời tiêu tốn ít điện năng hơn

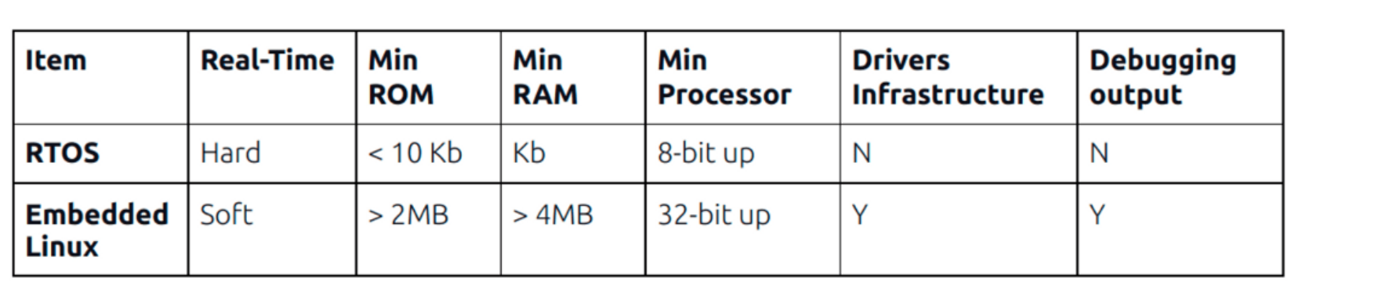
**Những thành phần quan trong của embedded linux:**

* Tool chain: chứa trình biên dịch và các công cụ cần thiết để tạo code cho thiết bị. Những thứ khác đều phụ thuộc vào toolchain.
* Bootloader: Nó cần thiết cho quá trình khởi tạo và tải, boo Linux kernel.
* Kernel: trái tim của hệ thống, có nhiệm vụ quản lý tài nguyên và giao tiếp với hardware.
* Root filesystem: chứa các thư viện và chương trình được chạy sau khi quá trình khởi tạo kernel hoàn thành.

**Ưu điểm so với embedded android**

* Vô số sự hỗi trợ có sẵn bảo đảm khả năng tương thích cao
* Dung lượng bộ nhớ tiêu chuẩn nhỏ hơn đáng kể
* Tối ưu hóa cho máy tính để bàn và máy tính xách tay
* Hỗ trợ nhiều kiến trúc
* Sử dụng thư viện C tiêu chuẩn
* Không cần thời gian chạy đặc biệt để chạy các ứng dụng
* Hệ điều hành thời gian thực có sẵn

**So sánh embedded linux với rtos**



## 2. Ngôn ngữ lập trình và SDK/IDE

### **2.1 Lịch sử Flutter**

Phiên bản đầu tiên của Flutter được gọi là"Sky"và chạy trên hệ điều hành Android. Nó được công bố tại hội nghị nhà phát triển Dart 2015, với dự định ban đầu để có thể kết xuất ổn định ở mức 120 khung hình trên giây. Trong bài phát biểu chính ở hội nghị Google Developer Days tại Thượng Hải, Google công bố phiên bản Flutter Release Preview 2, đây là phiên bản lớn cuối cùng trước Flutter 1.0. Vào ngày 4 tháng 12 năm 2018, Flutter 1.0 đã được phát hành tại sự kiện Flutter Live, là phiên bản"ổn định"đầu tiên của khung ứng dụng này.

**2.2 Flutter là gì?**

1. **SDK là gì?**

SDK viết tắt của *Software Development Kit* là các công cụ và phần mềm dùng để phát triển ứng dụng thông qua một nền tảng nhất định.

SDK được xây dựng tùy chỉnh sao cho tương thích với ngôn ngữ lập trình và các đặc điểm tương ứng. Theo đó, bạn sẽ có các loại SDK phổ biến sau:

* **Bộ công cụ SDK Android**: sử dụng ngôn ngữ Java, được dùng để lập trình các ứng dụng trên nền tảng Android.
* **Bộ công cụ SDK iOS**: sử dụng ngôn ngữ Swift, được dùng để hỗ trợ lập trình ứng dụng trên nền tảng iOS.
* **Bộ công cụ SDK Windows**: Windows yêu cầu phải có .NET Framework SDK đi kèm với .NET để lập trình các phần mềm chuyên dụng.
* **Bộ công cụ SDK VMware**: được dùng để tích hợp với nền tảng VMware (cho phép ảo hóa trên công nghệ điện toán đám mây).
* **Bộ công cụ SDK Bắc Âu**: được dùng để hỗ trợ tạo ra các sản phẩm Bluetooth hoặc không dây.

**SDK**lại đóng vai trò như một tập hợp các công cụ. Chương trình khác nhau có thể dùng để phát triển các phần mềm/ứng dụng mới trên một nền tảng cụ thể.

1. **Flutter là gì?**

Flutter là một SDK phát triển ứng dụng di động nguồn mở được tạo ra bởi Google. Nó được sử dụng để phát triển ứng ứng dụng cho Android và iOS, cũng là phương thức chính để tạo ứng dụng cho Google Fuchsia.

Flutter có thể sử dụng một ngôn ngữ lập trình và một cơ sở mã để tạo hai ứng dụng khác nhau (dành cho iOS và Android).

1. **Fluter gồm 2 phần quan trọng:**

**SDK (Software Development Kit):** Tập hợp các công cụ sẽ giúp bạn phát triển các ứng dụng của mình. Điều này bao gồm các công cụ để biên dịch mã của bạn thành mã máy gốc (mã cho iOS và Android).

**Framework (UI Library based on widgets):** Một bộ sưu tập các phần tử giao diện người dùng có thể sử dụng lại (nút, đầu vào văn bản, thanh trượt, v.v.) mà bạn có thể cá nhân hóa cho nhu cầu của riêng mình.

**2.3 Giới thiệu ngôn ngữ Dart:**

1. **Dart là gì ?**

Dart là một ngôn ngữ lập trình mã nguồn mở được sử dụng rộng rãi để phát triển ứng dụng di động, ứng dụng web hiện đại, ứng dụng máy tính để bàn và Internet of Things (IoT) bằng cách sử dụng khung Flutter. Nó cũng hỗ trợ một số khái niệm nâng cao như giao diện, mixin, lớp trừu tượng, tổng thể trường và giao diện kiểu.

Dart là một ngôn ngữ lập trình hiện đại có mục đích chung, cấp cao, được phát triển ban đầu bởi Google. Đây là ngôn ngữ lập trình mới xuất hiện vào năm 2011, nhưng phiên bản ổn định của nó đã được phát hành vào tháng 6 năm 2017. Dart không quá phổ biến vào thời điểm đó, nhưng nó đã trở nên phổ biến khi được sử dụng bởi Flutter.

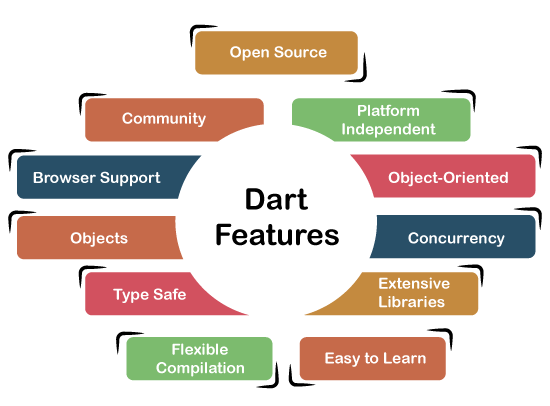
Dart là một ngôn ngữ lập trình động, dựa trên lớp, hướng đối tượng với phạm vi đóng và từ vựng. Về mặt cú pháp, nó khá giống với Java, C và JavaScript. Nếu bạn biết bất kỳ ngôn ngữ lập trình nào trong số này, bạn có thể dễ dàng học ngôn ngữ lập trình Dart.

Dart là một ngôn ngữ biên dịch và hỗ trợ hai loại kỹ thuật biên dịch.

* AOT (Ahead of Time) – Nó chuyển đổi mã Dart sang mã JavaScript được tối ưu hóa với sự trợ giúp của trình biên dịch dar2js và chạy trên tất cả các trình duyệt web hiện đại. Nó biên dịch mã tại thời điểm xây dựng.
* JOT (Just-In-Time) – Nó chuyển đổi mã byte trong mã máy (mã gốc), nhưng chỉ mã cần thiết.

1. **Tính năng của Dart**

Dart là một ngôn ngữ lập trình hướng đối tượng, mã nguồn mở, chứa nhiều tính năng hữu ích. Đây là ngôn ngữ lập trình mới và hỗ trợ một loạt các tiện ích lập trình như giao diện, bộ sưu tập, lớp, kiểu gõ động và tùy chọn. Nó được phát triển cho máy chủ cũng như trình duyệt. Dưới đây là danh sách các tính năng quan trọng của Dart



**Mã nguồn mở**

Dart là một ngôn ngữ lập trình mã nguồn mở, có nghĩa là nó có sẵn miễn phí. Nó được phát triển bởi Google, được phê duyệt bởi tiêu chuẩn ECMA và đi kèm với giấy phép BSD.

**Nền tảng độc lập**

Dart hỗ trợ tất cả các hệ điều hành chính như Windows, Linux, Macintosh, v.v. Dart có Máy ảo riêng được gọi là Dart VM, cho phép chúng tôi chạy mã Dart trong mọi hệ điều hành.

**Hướng đối tượng**

Dart là một ngôn ngữ lập trình hướng đối tượng và hỗ trợ tất cả các khái niệm oops như lớp, kế thừa, giao diện và các tính năng gõ tùy chọn. Nó cũng hỗ trợ các khái niệm nâng cao như mixin, abstract, các lớp, hệ thống kiểu chung được sửa đổi và mạnh mẽ.

**Đồng nhất**

Dart là một ngôn ngữ lập trình không đồng bộ, có nghĩa là nó hỗ trợ đa luồng sử dụng Isolates. Các vùng cách ly là các thực thể độc lập có liên quan đến các luồng nhưng không chia sẻ bộ nhớ và thiết lập giao tiếp giữa các quá trình bằng cách truyền thông điệp. Thông điệp nên được nối tiếp nhau để tạo hiệu quả truyền thông. Việc tuần tự hóa thông báo được thực hiện bằng cách sử dụng một ảnh chụp nhanh được tạo ra bởi đối tượng đã cho và sau đó truyền đến một vùng cách ly khác để giải mã.

**Thư viện mở rộng**

Dart bao gồm nhiều thư viện tích hợp hữu ích bao gồm SDK (Bộ phát triển phần mềm), lõi, toán học, không đồng bộ, toán học, chuyển đổi, html, IO, v.v. Nó cũng cung cấp cơ sở để tổ chức mã Dart thành các thư viện với không gian tên riêng. Nó có thể sử dụng lại bằng câu lệnh nhập.

**Dễ học**

Như chúng ta đã thảo luận trong phần trước, học Dart không phải là nhiệm vụ của Hercules vì ​​chúng ta biết rằng cú pháp của Dart tương tự như Java, C #, JavaScript, kotlin, v.v. nếu bạn biết bất kỳ ngôn ngữ nào trong số này thì bạn có thể học Dart dễ dàng.

**Biên dịch linh hoạt**

Dart cung cấp sự linh hoạt để biên dịch mã và nhanh chóng. Nó hỗ trợ hai loại quy trình biên dịch, AOT (Ahead of Time) và JIT (Just-in-Time). Mã Dart được truyền bằng ngôn ngữ khác có thể chạy trong các nhà sản xuất web hiện đại.

**Nhập An toàn**

Dart là ngôn ngữ an toàn kiểu, có nghĩa là nó sử dụng cả kiểm tra kiểu tĩnh và kiểm tra thời gian chạy để xác nhận rằng giá trị của một biến luôn khớp với kiểu tĩnh của biến, đôi khi nó được gọi là kiểu gõ âm thanh.

Mặc dù loại là bắt buộc, nhưng chú thích loại là tùy chọn vì loại nhiễu. Điều này làm cho mã dễ đọc hơn. Ưu điểm khác của ngôn ngữ an toàn kiểu chữ là khi chúng ta thay đổi phần mã, hệ thống sẽ cảnh báo chúng ta về sửa đổi mà chúng ta đã sửa trước đó.

**Các đối tượng**

Dart coi mọi thứ như một đồ vật. Giá trị gán cho biến là một đối tượng. Các hàm, số và chuỗi cũng là một đối tượng trong Dart. Tất cả các đối tượng kế thừa từ lớp Đối tượng.

**Hỗ trợ trình duyệt**

Dart hỗ trợ tất cả các trình duyệt web hiện đại. Nó đi kèm với trình biên dịch dart2js để chuyển đổi mã Dart thành mã JavaScript được tối ưu hóa phù hợp với tất cả các loại trình duyệt web.

**Cộng đồng**

Dart có một cộng đồng lớn trên toàn thế giới. Vì vậy, nếu bạn gặp vấn đề trong khi viết mã thì rất dễ dàng tìm được trợ giúp. Nhóm các nhà phát triển chuyên dụng đang làm việc để nâng cao chức năng của nó.

**2.4 Kiến trúc Flutter**

Để hiểu rõ hơn Flutter chúng ta phải tham khảo thêm kiến trúc của nó như sau:

* Với Flutter thì tất cả đều sẽ được quy về các widget và mỗi một widget phức hợp khác thường sẽ bao gồm những widget khác ở bên trong. Nhờ vào sự kết hợp linh hoạt trên mà người sử dụng có thể tạo ra được bất kỳ ứng dụng phức tạp nào khác.
* Mọi tính năng tương tác của Flutter đều sẽ được tích hợp vào bất kỳ thời điểm nào nhờ vào GestureDetector widget.
* Tất cả các trạng thái của widget đều sẽ được các quản lý cập nhật bởi StatefulWidget Widget.
* Flutter có khả năng cung cấp các thiết kế class để bất kỳ các lớp đều sẽ được lập trình sẵn tùy thuộc vào các mức độ phức tạp riêng của từng tác vụ.

**2.5 Ưu điểm của Flutter**

Flutter được đánh giá là có khả năng phát triển ứng dụng cực kỳ nhanh chóng với tính năng Hot Reload. Về cơ bản thì tính năng này sẽ hiển thị gần như là realtime (thời gian thực) cho những thay đổi trong thiết kế, hay những tùy chỉnh của bạn trên cùng một nền tảng.

Tạm hiểu thì bạn code gì, vẽ gì, thêm thắt gì, fix lỗi gì…. thì sẽ thấy ngay kết quả ở một cửa sổ bên cạnh. Điều này rõ ràng là nhanh hơn rất nhiều so với việc phải chạy[máy ảo Android](https://blogchiasekienthuc.com/thu-thuat-hay/cach-tao-may-tinh-ao-android.html) hay iOS.

Các bộ UI với các hiệu ứng, hoạt ảnh cực kỳ tiện lợi. Do được trang bị rất nhiều UI, đi cùng với các bộ widget phong phú nên các nhà phát triển có thể dễ dàng “hình ảnh hóa” những gì mà mình muốn mà không gặp quá nhiều khó khăn.

Google cho biết, các ứng dụng được build bởi Flutter thì gần như không có sự khác biệt so với những ứng dụng được xây dựng bằng Android SDK, cả về giao diện lẫn hiệu suất. Hơn nữa, chỉ với một vài tùy chỉnh nhỏ thôi là chúng có thể chạy được trên thiết bị iOS rồi.

Đây là một nền tảng mã nguồn mở và lại do chính Google phát triển nên cộng đồng sử dụng chúng là rất lớn. Bạn có thể dễ dàng tìm tài liệu, cách sử dụng, cũng như các lỗi trong quá trình sử dụng trên những trang cộng đồng.

Flutter hỗ trợ tạo ra các ứng dụng đa nền tảng, chỉ với một số tùy chỉnh nhỏ thôi là bạn đã có thể tạo ra một ứng dụng chạy trên cả Android và iOS, với một lần code duy nhất.

Các ứng dụng Flutter có thể được phát triển bằng cách sử dụng [IntelliJ IDEA](https://blogchiasekienthuc.com/windows-server/cai-dat-intellij-ultimate-tren-windows.html), một IDE rất giống với [Android Studio](https://blogchiasekienthuc.com/lap-trinh/phan-mem-ho-tro-lap-trinh-ung-dung-android.html).

Flutter sử dụng ngôn ngữ lập trình Dart, một ngôn ngữ rất nhanh với nhiều tính năng hữu ích như mixin, generic, isolate, và static type.

Trong tương lai thì họ sẽ cải tiến để Flutter có thể tạo ra được một trang web hiện đại với nhiều tính năng hơn.

**2.6 Nhược điểm của Flutter**

Tuy là mạnh mẽ thật đấy nhưng thật khó để Flutter có thể đạt được những điều mà ngôn ngữ Java, C , JavaScript đã làm được. Đơn giản là vì nó còn khá mới mẻ!

Hiện nay nhu cầu nhân lực sử dụng nền tảng này để làm app vẫn còn khá hạn chế, gần như chỉ phù hợp với những ai làm bên ngoài, làm tự do ([Freelancer](https://blogchiasekienthuc.com/la-gi/freelancer-la-gi.html)) chứ còn trong các công ty chuyên nghiệp thì rất là ít.

Vì mới được phát triển nên các bộ thư viện để tạo nên tính năng trong ứng dụng vẫn còn hạn chế, không thể bằng những nền tảng đã phát triển lâu đời. Nhưng điều này trong tương lai hoàn toàn có thể khắc phục được.

Kích thước file lớn và đôi lúc sự tương thích là chưa ổn định:

Đồng ý là việc làm ra một ứng dụng chạy được trên cả Android và iOS là tuyệt vời thật đó, nhưng rõ ràng sự chuyên biệt luôn tốt hơn những thứ tổng hợp. Nên đôi khi ứng dụng sẽ gặp một vài lỗi vặt trên 2 nền tảng này và kích thước file của chúng còn khá lớn.

Ngôn ngữ Dart vẫn còn lạ lẫm với nhiều anh em dev, gần như không có quá nhiều coder biết đến ngôn ngữ này. Tài liệu về chúng cũng không được nhiều như các ngôn ngữ khác. Vậy nên dù có thể là dễ học, dễ thực hiện nhưng so với các ngôn ngữ thịnh hành thì đó là một trở ngại.

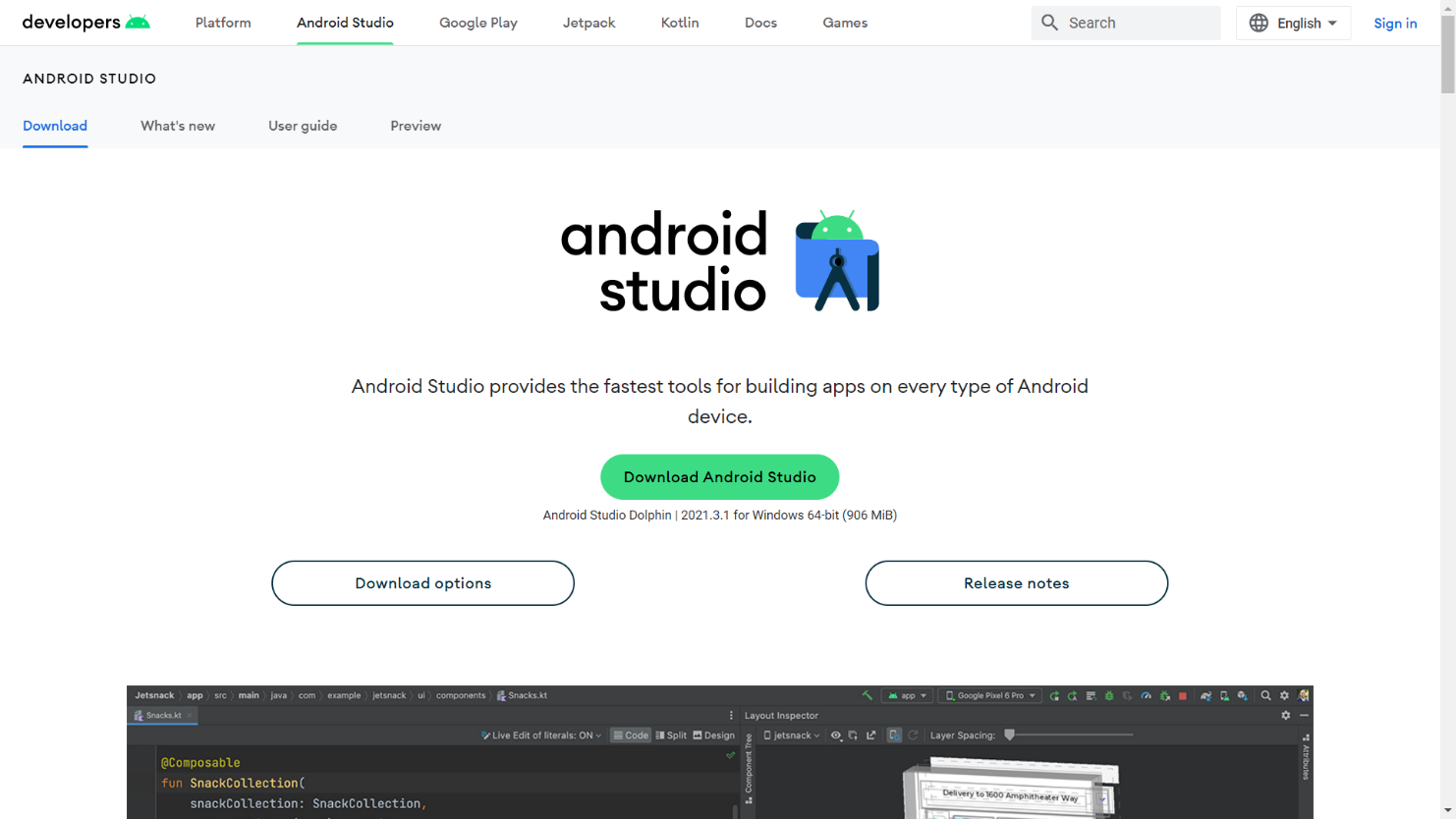
Thiết bị di động ngày càng phát triển, kéo theo nhu cầu về những ứng dụng di động ngày càng cao.

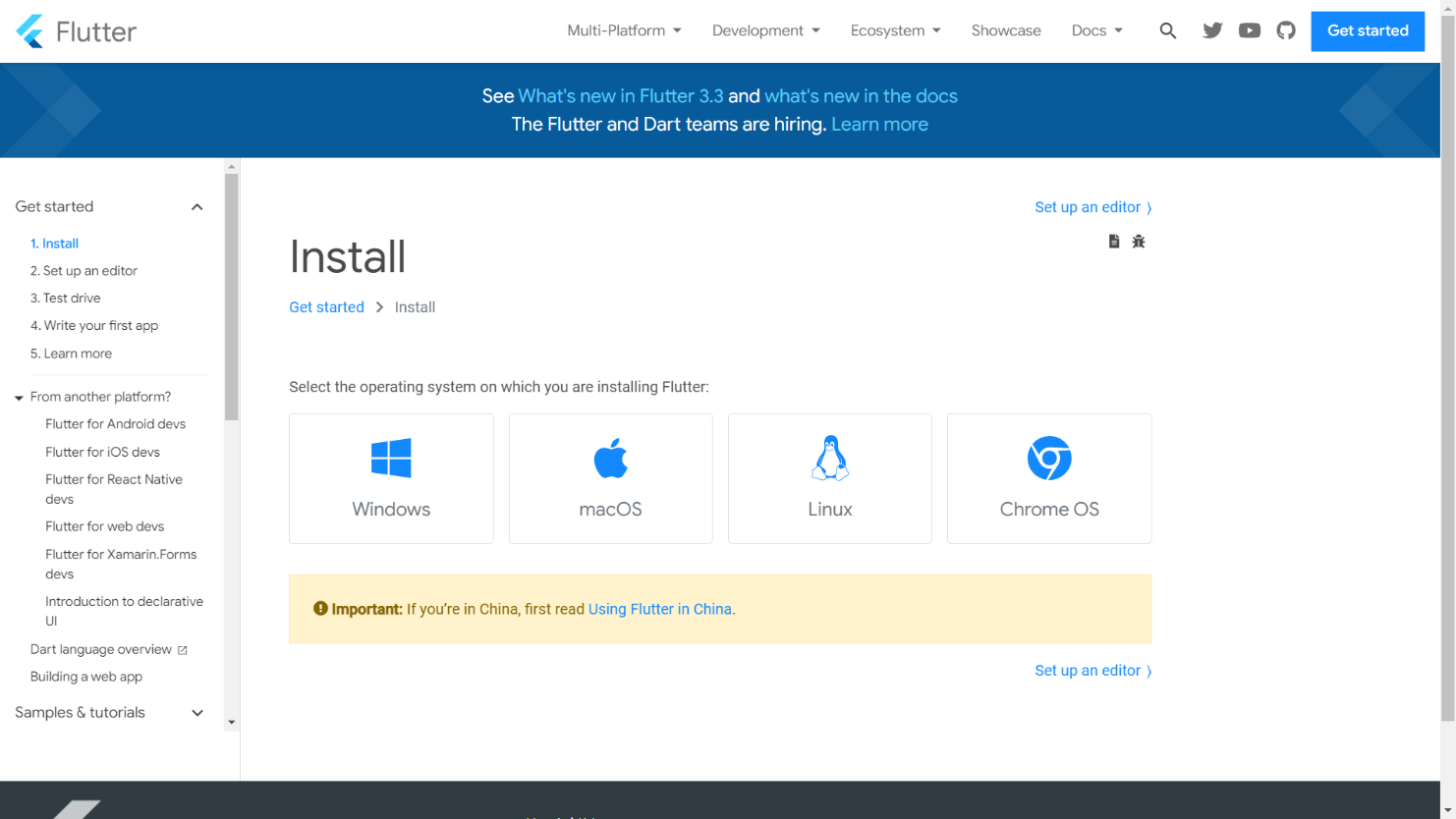
Đối với một công ty phát triển ứng dụng/ phần mềm, hoặc là một cá nhân làm tự do thì việc làm sao để tối ưu được tốc độ phát triển ứng dụng nhưng vẫn đảm bảo hiệu năng hoạt động luôn là điều mà họ hướng tới.

Đó cũng chính là lý do mà Flutter ra đời. Sẽ còn quá sớm để nói nó sẽ cạnh tranh được với các ông lớn như Java, JavaScript… nhưng mình tin framework này sẽ một xu hướng trong tương lai tới.

**2.7 Hướng dẫn cài đặt Flutter**

**Bước 1:** Thực hiện download *android studio* ở trang: *https://developer.android.com/studio*.



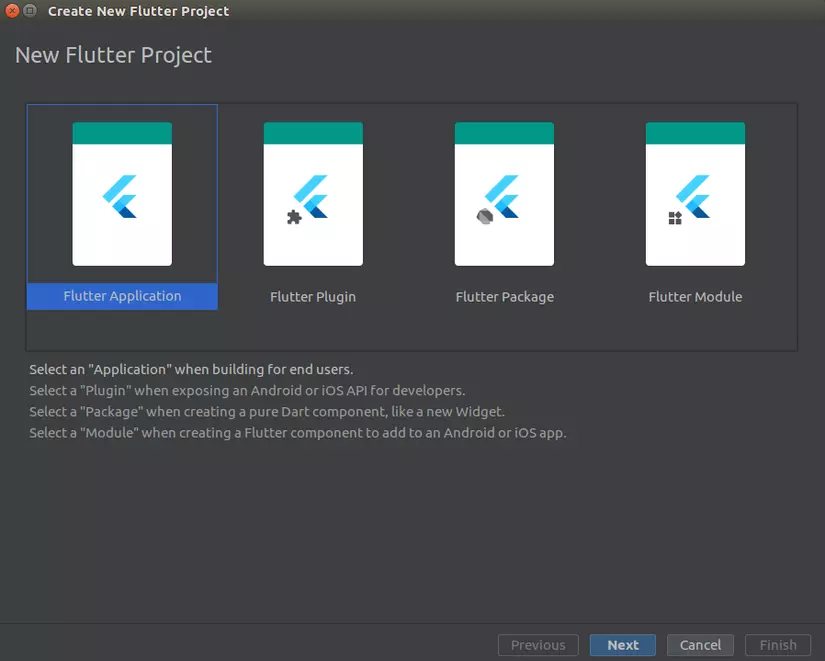
**Bước 2:** Tiến hành Download Flutter SDK ở trang: *https://docs.flutter.dev/get-started/install*

* 1. **Tiến hành giải nén SDK flutter.**

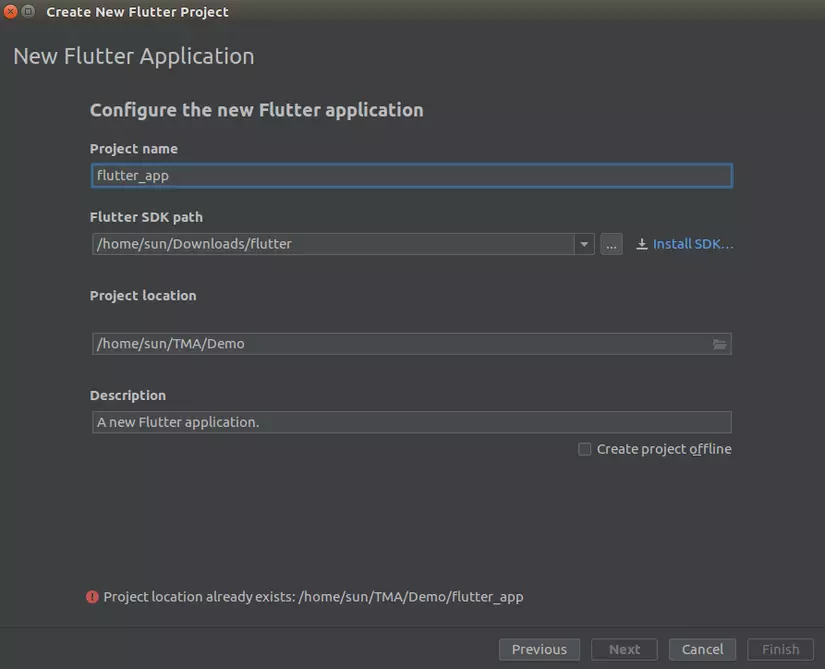
**Bước 3:** Tiến hành mở **Android Studio** rồi chọn **Start a new Flutter project**.



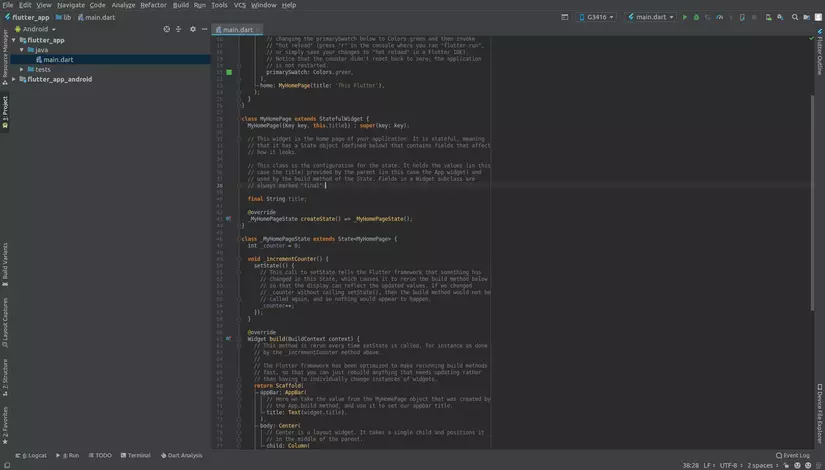
* 1. **Chọn Flutter project**

****

**Bước 4:** Cài đặt Flutter SDK path (đường dẫn flutter mình vừa giải nén) 🡪 **click Next**

****

* 1. **Đợi Android Studio thực hiện build ứng dụng…Kết quả.**

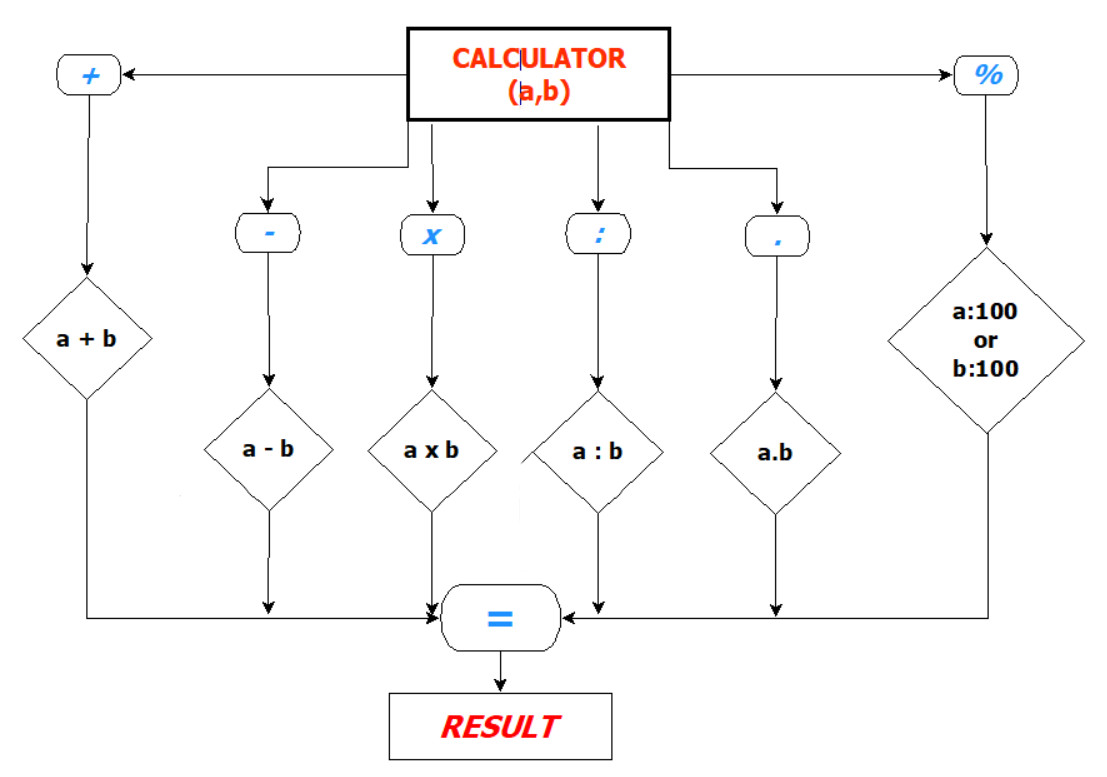
****

# **III. Bài tập**

**1. Chức năng phần mềm**

Ứng dụng với công cụ tính toán thông minh, giúp bạn đưa ra đáp án một cách chính xác, nhanh chóng. Bạn sẽ không còn lo mình tính sai tiền khi mua một món đồ nào đó hay tính toán chi tiêu trong gia đình nữa.

**2. Lưu đồ giải thuật**



**Lưu đồ app Calculator**

Giải thích lưu đồ

Bước 1: Đầu tiên, ta nhập 2 giá trị a, b vào từ bàn phím

Bước 2: Sau đó ta nhấn 1 nút thể hiện cho 1 phép tính cụ thể (ví dụ cộng, trừ, nhân, chia) mà ta muốn

Bước 3: Khi ta nhấn phím “=” , phép tính giữa 2 số a và b sẽ được thực hiện

Bước 4: Giá trị sau cùng của phép tính được in ra màn hình

**3. Code**

import 'package:flutter/material.dart';

void main() => runApp(new MyApp());

//////////////////////////// W I D G E T S //////////////////////////////

class HomePage extends StatefulWidget {

@override

State createState() => new HomePageState();

}

class HomePageState extends State<HomePage> {

List<String> operators = ["+", "-", "×", "÷"];

List<String> hist = [];

var history = "", output = "", answer = 0.0;

/\* -----------click function button------------------\*/

void click1() {

setState(() {

if (double.parse(output) != 0.0) {

output += "1";

} else {

output = "1";

}

});

}

void click2() {

setState(() {

if (double.parse(output) != 0.0) {

output += "2";

} else {

output = "2";

}

});

}

void click3() {

setState(() {

if (double.parse(output) != 0.0) {

output += "3";

} else {

output = "3";

}

});

}

void click4() {

setState(() {

if (double.parse(output) != 0.0) {

output += "4";

} else {

output = "4";

}

});

}

void click5() {

setState(() {

if (double.parse(output) != 0.0) {

output += "5";

} else {

output = "5";

}

});

}

void click6() {

setState(() {

if (double.parse(output) != 0.0) {

output += "6";

} else {

output = "6";

}

});

}

void click7() {

setState(() {

if (double.parse(output) != 0.0) {

output += "7";

} else {

output = "7";

}

});

}

void click8() {

setState(() {

if (double.parse(output) != 0.0) {

output += "8";

} else {

output = "8";

}

});

}

void click9() {

setState(() {

if (double.parse(output) != 0.0) {

output += "9";

} else {

output = "9";

}

});

}

void click0() {

setState(() {

if (double.parse(output) != 0.0) {

output += "0";

} else {

output = "0";

}

});

}

void clickDot() {

setState(() {

output += ".";

});

}

void clear() {

setState(() {

history = "";

output = "0";

answer = 0.0;

hist = [];

});

}

void sign() {

setState(() {

if (double.parse(output) == 0.0) {

} else {

if (output[0] == '-') {

output = output.substring(1);

} else {

output = '-' + output;

}

}

});

}

void percent() {

setState(() {

double percent = 0.0;

percent = answer / 100;

history = answer.toString() + " ÷ 100 =";

output = percent.toString();

});

}

/\*----------------------------------------------------\*/

String getTape() {

return hist.join(" ");

}

bool isOperator(String s) {

return (operators.contains(s));

}

bool isNumeric(String s) {

if (s == null) {

return false;

}

return double.parse(s) != null;

}

/\* xuat ra ket qua \*/

void equals() {

setState(() {

if (hist.length <= 3) {

hist.add(output);

}

history = getTape() + " =";

var opr1, opr2, op;

opr1 = double.parse(hist.removeAt(0));

op = hist.removeAt(0);

opr2 = double.parse(hist.removeAt(0));

switch (op) {

case "+":

answer = opr1 + opr2;

break;

case "-":

answer = opr1 - opr2;

break;

case "×":

answer = opr1 \* opr2;

break;

case "÷":

answer = opr1 / opr2;

break;

default:

}

output = answer.toString();

hist.insert(0, answer.toString());

});

}

//Ham cong

void add() {

setState(() {

answer = double.parse(output);

hist.add(output);

hist.add("+");

if (hist.length >= 3) {

output = "0";

equals();

}

output = "0";

history = getTape();

});

}

// ham tru

void sub() {

setState(() {

answer = double.parse(output);

hist.add(output);

hist.add("-");

if (hist.length >= 3) {

output = "0";

equals();

}

output = "0";

history = getTape();

});

}

//ham chia

void div() {

setState(() {

answer = double.parse(output);

hist.add(output);

hist.add("÷");

if (hist.length >= 3) {

output = "0";

equals();

}

output = "0";

history = getTape();

});

}

// Ham nhan

void mul() {

setState(() {

answer = double.parse(output);

hist.add(output);

hist.add("×");

if (hist.length >= 3) {

output = "0";

equals();

}

output = "0";

history = getTape();

});

}

/\* Giao dien \*/

@override

Widget build(BuildContext context) {

return new Scaffold(

appBar: new AppBar(

centerTitle: true,

backgroundColor: Colors.deepOrangeAccent.shade400,

title: Text('Calculator'),

),

body: new Container(

child: new Column(

mainAxisAlignment: MainAxisAlignment.spaceAround,

crossAxisAlignment: CrossAxisAlignment.stretch,

children: <Widget>[

new Padding(

padding: new EdgeInsets.only(top: 25.0, right: 15.0),

child: new Text(

"$history",

overflow: TextOverflow.fade,

maxLines: 1,

style: TextStyle(

fontSize: 25.0,

fontWeight: FontWeight.w200,

),

textAlign: TextAlign.end,

),

),

new Padding(

padding:

new EdgeInsets.only(top: 15.0, right: 15.0, bottom: 15.0),

child: new Text(

"$output",

overflow: TextOverflow.fade,

maxLines: 1,

style: TextStyle(

fontSize: 60.0,

fontWeight: FontWeight.w100,

),

textAlign: TextAlign.end,

),

),

new Padding(

padding: new EdgeInsets.only(top: 15.0, left: 5.0),

child: new Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

new RawMaterialButton(

onPressed: clear,

child: new Icon(

Icons.block,

size: 35.0,

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.deepOrangeAccent.shade400,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: sign,

child: new Text(

"±",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.deepOrangeAccent.shade400,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: percent,

child: new Text(

"%",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.deepOrangeAccent.shade400,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: div,

child: new Text(

"÷",

style: TextStyle(

fontSize: 35.0,

color: Colors.deepOrangeAccent.shade400,

fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.white,

highlightColor: Colors.deepOrangeAccent.shade400,

splashColor: Colors.red[100],

padding: const EdgeInsets.all(15.0),

)

]),

),

new Padding(

padding: new EdgeInsets.only(top: 15.0, left: 5.0),

child: new Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

new RawMaterialButton(

onPressed: click1,

child: new Text(

"1",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: click2,

child: new Text(

"2",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: click3,

child: new Text(

"3",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: mul,

child: new Text(

"×",

style: TextStyle(

fontSize: 35.0,

color: Colors.red[700],

fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.white,

highlightColor: Colors.red[100],

splashColor: Colors.red[100],

padding: const EdgeInsets.all(15.0),

)

]),

),

new Padding(

padding: new EdgeInsets.only(top: 15.0, left: 5.0),

child: new Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

new RawMaterialButton(

onPressed: click4,

child: new Text(

"4",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: click5,

child: new Text(

"5",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: click6,

child: new Text(

"6",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: sub,

child: new Text(

"-",

style: TextStyle(

fontSize: 35.0,

color: Colors.red[700],

fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.white,

highlightColor: Colors.red[100],

splashColor: Colors.red[100],

padding: const EdgeInsets.all(15.0),

)

]),

),

new Padding(

padding: new EdgeInsets.only(top: 15.0, left: 5.0),

child: new Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

new RawMaterialButton(

onPressed: click7,

child: new Text(

"7",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: click8,

child: new Text(

"8",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: click9,

child: new Text(

"9",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: add,

child: new Text(

"+",

style: TextStyle(

fontSize: 35.0,

color: Colors.red[700],

fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.white,

highlightColor: Colors.red[100],

splashColor: Colors.red[100],

padding: const EdgeInsets.all(15.0),

)

]),

),

new Padding(

padding: new EdgeInsets.only(top: 15.0, left: 5.0, bottom: 6.0),

child: new Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

new RawMaterialButton(

onPressed: click0,

child: new Text(

"0",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

constraints: BoxConstraints.tightFor(width: 170.0),

shape: new RoundedRectangleBorder(

borderRadius: BorderRadius.circular(45.0)),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.only(

left: 18.0, top: 15.0, bottom: 15.0, right: 15.0),

),

new RawMaterialButton(

onPressed: clickDot,

child: new Text(

".",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.black45,

padding: const EdgeInsets.all(15.0),

),

new RawMaterialButton(

onPressed: equals,

child: new Text(

"=",

style: TextStyle(

fontSize: 35.0, fontWeight: FontWeight.w500),

),

shape: new CircleBorder(),

elevation: 2.0,

fillColor: Colors.deepOrangeAccent.shade400,

padding: const EdgeInsets.all(15.0),

)

]),

),

]),

),

);

}

}

///////////////////////////// A P P W I D G E T ///////////////////////////

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return new MaterialApp(

title: "Calcon",

theme: new ThemeData(

primarySwatch: Colors.red,

accentColor: Colors.deepOrange,

brightness: Brightness.dark),

home: new HomePage(),

);

}

}

# **Tài liệu tham khảo**

**Bộ xử lý đơn dụng (single-use processor)**

[1] <https://en.wikibooks.org/wiki/Embedded_Control_Systems_Design/Processors>

[2] <http://easy2teach.blogspot.com/2011/12/what-is-single-purpose-processor.html>

[3]<http://utenti.dieei.unict.it/users/gascia/COURSES/sist_emb_14_15/download/SE03_Processor%20technology.pdf>

[4] <https://www.quora.com/What-is-a-difference-between-general-purpose-processor-and-single-purpose-processor>

[5] <https://slideplayer.com/slide/5128471/>

**Bộ xử lý đa dụng (general-purpose processor)**

[6] [https://www.sciencedirect.com/topics/computer-science/general-purpose-processor#:~:text=GPPs%20are%20the%20processors%20that,from%20a%20variety%20of%20manufacturers](https://www.sciencedirect.com/topics/computer-science/general-purpose-processor" \l ":~:text=GPPs%20are%20the%20processors%20that,from%20a%20variety%20of%20manufacturers).

[7] <https://www.european-processor-initiative.eu/general-purpose-processor/>

[8] <https://web.itu.edu.tr/yalcinmust/EHB432/msd_not_4.pdf>

[9] <https://www.javatpoint.com/embedded-system-processors>

[10] <https://en.wikipedia.org/wiki/History_of_general-purpose_CPUs>

[11] [https://www.itwissen.info/en/general-purpose-processor-GPP-122865.html#gsc.tab=0](https://www.itwissen.info/en/general-purpose-processor-GPP-122865.html" \l "gsc.tab=0)

[12] <https://www.igi-global.com/dictionary/rfmicrowave-instruments-evolution/110330>

[13] <https://www.directindustry.com/industrial-manufacturer/general-purpose-processor-210389.html>

[14] <https://www.researchgate.net/figure/General-purpose-processor-architecture-vs-custom-hardware-architecture_fig5_317091879>

[15] <https://electronics.stackexchange.com/questions/159293/why-is-it-called-general-purpose-processor>

[16] <https://www.tutorialspoint.com/embedded_systems/es_processors.htm>

[17] <https://personalpages.manchester.ac.uk/staff/p.dudek/papers/dudek-tcasi2005.pdf>

[18] <https://ieeexplore.ieee.org/document/808589>

[19] <https://slideplayer.com/slide/15417552/>

[20] <https://courses.cs.washington.edu/courses/cse477/99wi/lectures/02-MicrocontrollersI/sld006.htm>

[21] <https://cacm.acm.org/magazines/2014/12/180775-no-such-thing-as-a-general-purpose-processor/abstract>

[22] <http://esd.cs.ucr.edu/toc.html>

[23] <https://www.futurelearn.com/info/courses/embedded-systems/0/steps/64771>

**Bộ xử lý chuyên dụng (Dedicated processor)**

[24] <https://www.e2enetworks.com/blog/dedicated-cpu-vs-shared-vcpus>

[25] <https://www.ibm.com/support/pages/what-dedicated-processor#:~:text=Dedicated%20processors%20are%20whole%20processors,one%20processor%20to%20that%20partition>.

[26] <https://www.ibm.com/docs/en/power8?topic=processors-dedicated>

[27] <https://winnervps.com/index.php?rp=/knowledgebase/108/Why-and-When-do-I-need-a-Dedicated-CPU.html>

[28] <https://www.linode.com/docs/guides/platform/dedicated-cpu/>

[29] <https://www.linode.com/docs/guides/comparing-shared-and-dedicated-cpus/>

[30] <https://www.researchgate.net/figure/Dedicated-Processor-Types_fig3_3667566>  
[31] <https://www.sciencedirect.com/science/article/pii/S1474667017575126>

[32] <https://www.linode.com/products/dedicated-cpu/>

[33] <https://ieeexplore.ieee.org/document/238375>

[34] <https://patents.google.com/patent/US5280621>

**UART**

[35] <https://dientutuonglai.com/giao-tiep-uart-la-gi.html>

[36] <https://advancecad.edu.vn/khai-niem-co-ban-ve-truyen-thong-uart-so-do-khoi-ung-dung/#Uu_diem_va_nhuoc_diem_cua_UART>

[37] <https://dientuviet.com/kien-thuc-co-ban-ve-giao-tiep-uart/>

[38] <https://benh.edu.vn/giao-tiep-uart-la-gi/>

[39] <https://kysungheo.com/chuan-giao-tiep-uart/>

[40] <https://ktktdl.edu.vn/uart-la-gi/>

[41] <https://tudienso.com/thuat-ngu/index.php/universal-asynchronous-receiver-transmitter-uart-la-gi/>

[42] <https://tapit.vn/luoc-ly-thuyet-ve-chuc-nang-uart-va-mot-thanh-ghi-trong-chip-stm32f103c8t6/>

[43] https://deviot.vn/tutorials/pic.22296474/giao-tiep-uart.93780684

[44] <https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter>

[45] https://vi.wikipedia.org/wiki/Truyền\_thông\_nối\_tiếp

**SPI**

[46] <https://dientutuonglai.com/chuan-giao-tiep-spi-la-gi.html>

[47] <https://deviot.vn/blog/giao-tiep-spi.74706311>

[48] <https://dammedientu.vn/chuan-giao-tiep-spi-va-cach-ket-noi>

[49] <https://vi.wikipedia.org/wiki/Serial_Peripheral_Interface>

[50] <http://arduino.vn/bai-viet/1081-arduino-va-giao-tiep-spi>

[51] <https://en.wikipedia.org/wiki/Serial_Peripheral_Interface>

**I2C**

[52] <https://dientutuonglai.com/chuan-giao-tiep-i2c-la-gi.html>

[53] <https://deviot.vn/blog/giao-tiep-i2c.05019305>

[54] <https://dammedientu.vn/gioi-thieu-chuan-giao-tiep-i2c>

[55] <https://kysungheo.com/chuan-giao-tiep-i2c/>

[56] <https://esp32.vn/idf/i2c.html>

[57] https://vi.wikipedia.org/wiki/I²C

[58] <http://arduino.vn/bai-viet/1053-giao-tiep-i2c-voi-nhieu-module>

[59] <https://tapit.vn/giao-thuc-i2c-va-giao-tiep-voi-cam-bien-nhiet-hong-ngoai-mlx90614/>

[60] <http://kdientu.duytan.edu.vn/media/109456/gioi-thieu-giao-thuc-i2c.pdf>

[61] https://en.wikipedia.org/wiki/I²C

[62] <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>

**LAN**

[63] <https://www.dienmayxanh.com/kinh-nghiem-hay/lan-la-gi-596356>

[64] <https://capquangviettel.vn/mang-lan-la-gi-cong-dung-cua-mang-lan-nhu-nao/>

[65] <https://www.totolink.vn/article/151-lan-la-gi-tong-quan-kien-thuc-ve-lan-ma-ban-nen-biet.html>

[66] <https://wifi.fpt.net/mang-lan-la-gi-tat-tan-tat-ve-mang-lan/>

[67] <https://mediamart.vn/tu-van-may-tinh/lan-la-gi-cac-kieu-ket-noi-va-cong-dung-cua-mang-lan>

[68] https://vi.wikipedia.org/wiki/Mạng\_cục\_bộ

[69] <https://fpttelecom.com/blog/mang-lan-la-gi/>

[70] <https://bacsitinhoc.com.vn/tin-tuc/tim-hieu-ve-mang-lan-la-gi-va-ung-dung-cua-mang-lan/>

[71] <https://tip.com.vn/lan-la-gi-tong-quan-ve-mang-may-tinh-mang-lan.html>