TRƯỜNG ĐẠI HỌC SƯ PHẠM
KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

BÁO CÁO CUỐI KỲ

MÔN HỌC: HỆ THỐNG NHÚNG

GVHD: Ths. **Đậu Trọng Hiển**

SVTH: Nhóm 7

Trần Nguyễn Quang Lâm – 20139040

Nguyễn Trí Ban – 20139062

Trần Đức Hiếu – 20139074

Tô Gia Huy – 20139003

Phạm Thanh Hà – 20139073

TP Hồ Chí Minh, tháng 12 năm 2022

BẢNG PHÂN CÔNG NHIỆM VỤ

THỨ TỰ	HỌ TÊN	NHIỆM VỤ	KẾT QUẢ	KÝ TÊN
1	Phạm Thanh Hà	Phân công, tổng hợp và mục I - 1	Hoàn thành tốt	
2	Nguyễn Trí Ban	Mục I - 2, I - 3	Hoàn thành tốt	
3	Trần Đức Hiếu	Mục III - 2, mục II -2	Hoàn thành tốt	
4	Tô Gia Huy	Mục III	Hoàn thành tốt	
5	Trần Nguyễn Quang Lâm	Mục I – 4, mục II - 1	Hoàn thành tốt	

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

Ký tên

Ths. Đậu Trọng Hiển

MỤC LỤC

1. PHẦN CỨNG HỆ THỐNG NHÚNG

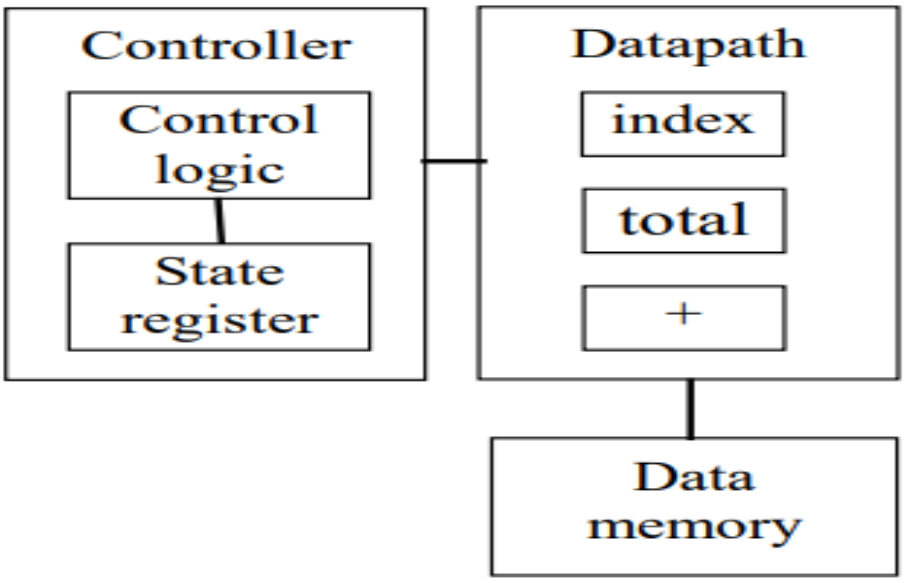
1.1 Bộ xử lý đơn dụng

Bộ xử lý đơn dụng(Single-purpose processors) là mạch kĩ thuật số, để thực hiện các tác vụ tính toán, thực hiện chính xác một chương trình. Bộ xử lý cơ bản bao gồm bộ điều khuyển và đường dẫn dũ liệu .

Đường dẫn dữ liệu lưu trữ và thao tác dữ liệu của hệ thống, 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.

Tương tự, bộ điều khuyển cấu hình bằng cách 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.

Bộ điều khuyển đảm nhận vai trò giám sát đầu vào bên ngoài cũng như đầu ra điều khiển đường dẫn dữ liệu, thể hiện qua các tín hiệu trạng thái.Ta có thể sử dụng kỹ thuật thiết kế hệ thống số, thiết kế Logic tổ hợp và tuần tự, thiết kế đồng bộ hoặc không đồng bộ để xây dựng bộ xử lý đơn dụng.



Đặc trưng của bộ xử lý đơn dụng:

Hiệu suất có thể tối ưu, vì bộ xử lý được tùy chỉnh cho tác vụ cụ thể đang diễn ra.

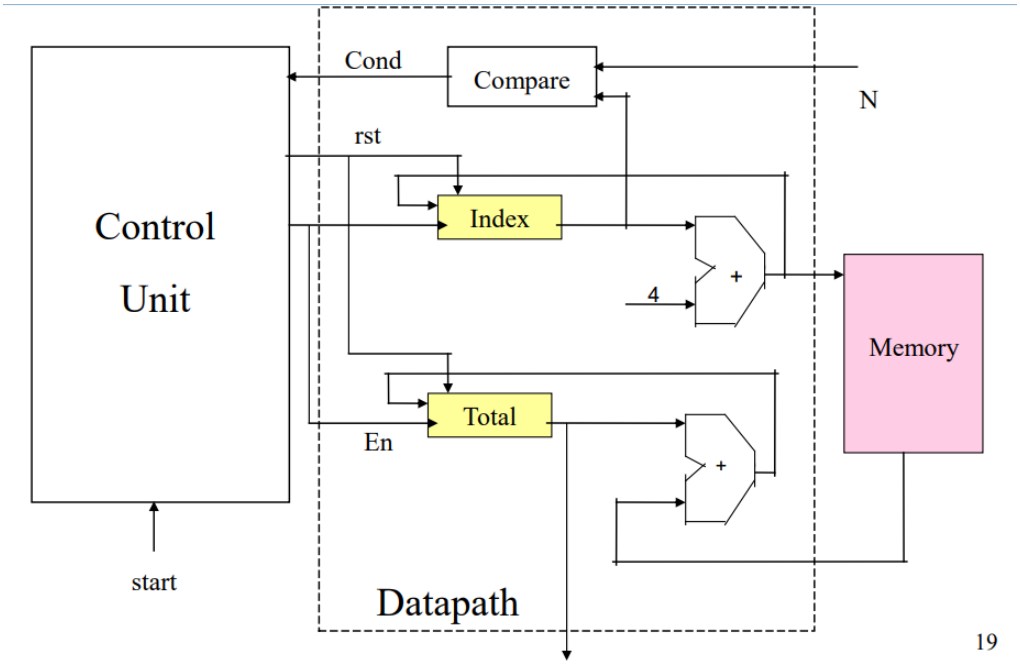
Các tác vụ có thể được thực thi với trong ít chu trình máy hơn, và chính các chu trình cũng ngắn hơn:

- Ít chu kỳ xung nhịp hơn có thể do nhiều thành phần đường dẫn dữ liệu hoạt động song song, từ các thành phần đường dẫn dữ liệu truyền dữ liệu trực tiếp cho nhau mà không cần các thanh ghi trung gian, hoặc do loại bỏ việc tìm nạp bộ nhớ chương trình.
- Các chu kỳ ngắn hơn có thể do các đơn vị chức năng đơn giản hơn, ít bộ ghép kênh hơn hoặc logic điều khiển đơn giản hơn.

Bộ xử lý đơn dụng không yêu cầu bộ nhớ chương trình. Ngoài ra, vì nó không cần hỗ trợ một tập lệnh lớn nên nó có thể có bộ điều khiển và đường dẫn dữ liệu đơn giản hơn.

Bộ xử lý đơn dụng có tốc độ nhanh, tuy nhiên công suất thấp,không linh hoạt, thời gian để đưa ra thì trường lâu và chi phí NRE cao.

Thiết kế cơ bản:

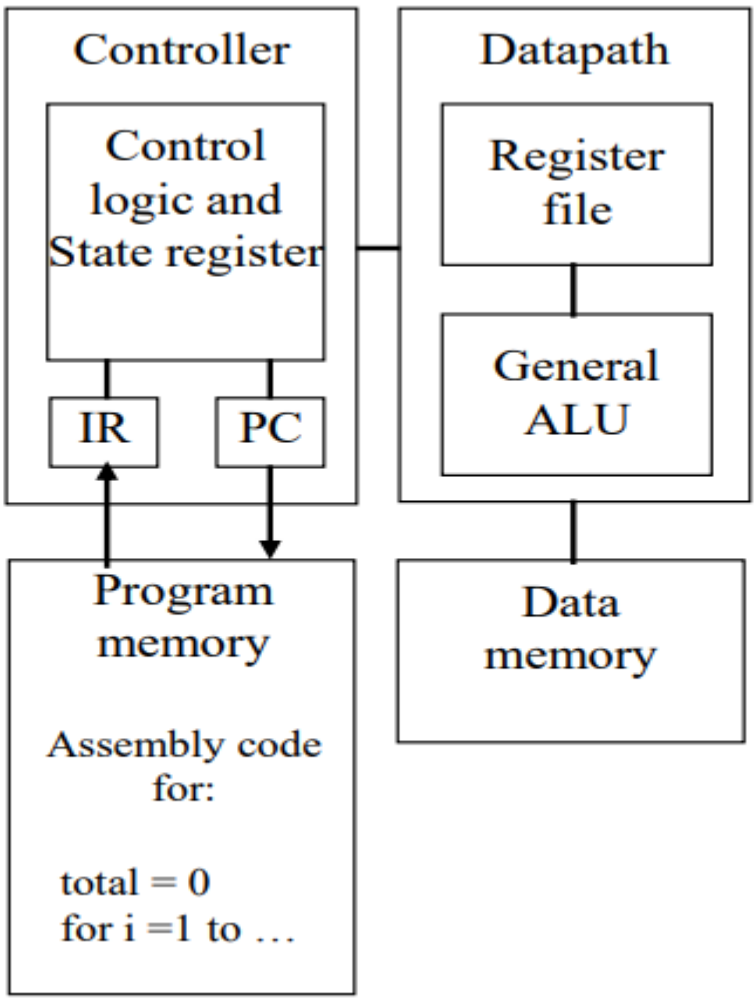


1.2 Bộ xử lý đa dụng

Bộ xử lý đa dụng (General-purpose processor) là bộ xủ lý đa năng, triển khai của trình thông dịch. Với tham chiếu là bộ đếm chương trình, được lưu trữ trong thanh ghi bộ

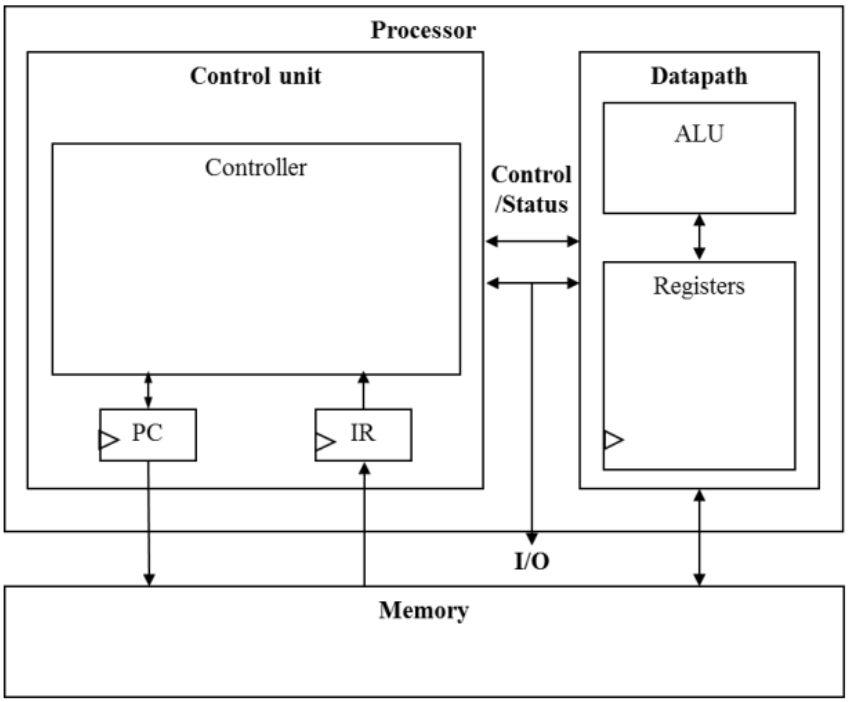
nhớ nhanh bên trong bộ xử lý. Bộ đếm chương trình chứa địa chỉ của vị trí bộ nhớ lưu lệnh tiếp theo của chương trình hiện tại. Tham chiếu môi trường của bộ xử lý bao gồm một phần của một lượng nhỏ bộ nhớ đượ

chương trình. Trong trường hợp thứ hai, trình xử lý ngắt có thể thực hiện một số công việc và sau đó trả lại quyền điều khiển cho chương trình ban đầu.



Bộ xử đa dụng được sử dụng trong nhiều loại ứng dụng với đường dẫn dữ liệu chung với tệp thanh ghi lớn và ALU chung. Với thời gian đưa sản phẩm ra thị trường nhanh chóng và chi phí NRE thấp. Tính linh hoạt cao, tuy nhiên chi phí đầu vào cao.

Thiết kế cơ bản:



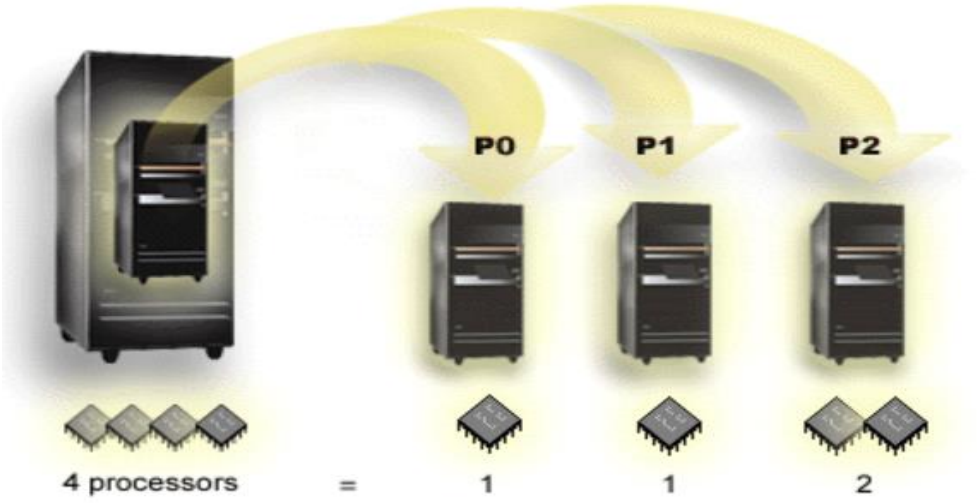
1.3 Bộ xử lý chuyển dụng

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 ,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.

Bộ xử lý chuyên dụng phù hợp để thực hiện các thuật toán cấp thấp thông thường, các phép đo hiệu suất và chi phí thậm chí có thể được tính toán một cách xác định, sử dụng phương pháp ước tính, dựa trên mô hình thuật toán và kiến thức liên quan đến chu kỳ xung nhịp cho các hoạt động cơ bản và truy cập bộ nhớ của bộ xử lý có thể lập trình.

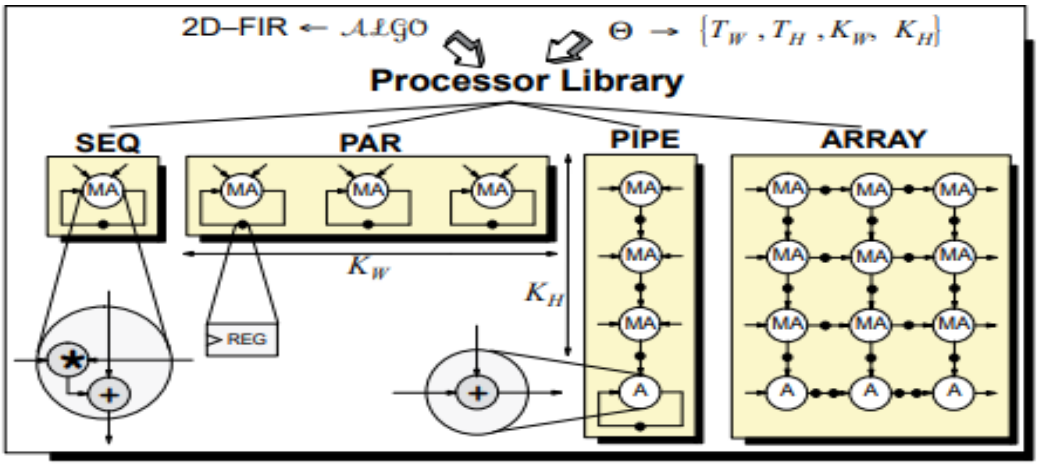


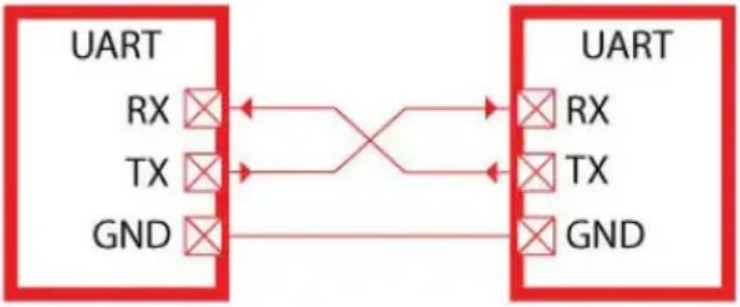
Fig. 6: Dedicated Processor Types for 2D-FIR Filtering

1.4 UART, SPI, I2C, LAN

1.4.1 UART

UART là một giao thức truyền thông phần cứng sử dụng giao tiếp nối tiếp không đồng bộ với tốc độ có thể định cấu hình. Không đồng bộ có nghĩa là không có tín hiệu đồng hồ để đồng bộ hóa các bit đầu ra từ thiết bị truyền đi đến bên nhận.

Trong giao tiếp UART, hai UART giao tiếp trực tiếp với nhau. UART truyền chuyển đổi dữ liệu song song từ một thiết bị điều khiển như CPU thành dạng nối tiếp, truyền nó nối tiếp đến UART nhận, sau đó chuyển đổi dữ liệu nối tiếp trở lại thành dữ liệu song song cho thiết bị nhận.



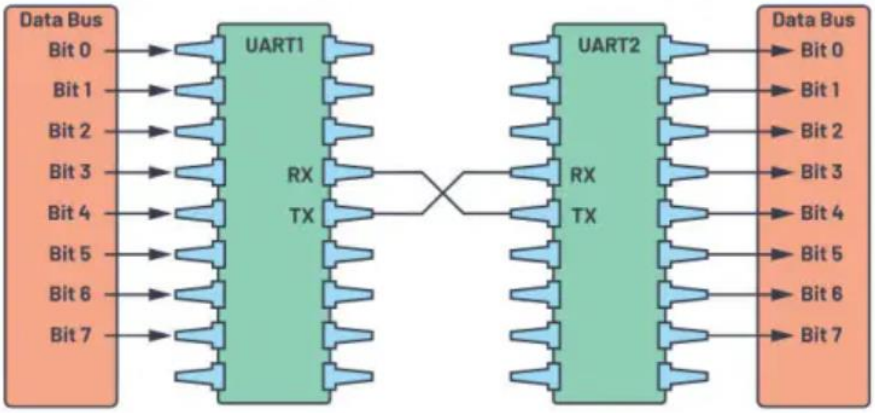
Hai đường dây mà mỗi thiết bị UART sử dụng để truyền dữ liệu đó là:

- Transmitter (Tx)
- Receiver (Rx)

UART truyền dữ liệu không đồng bộ, có nghĩa là không có tín hiệu đồng hồ để đồng bộ hóa đầu ra của các bit từ UART truyền đến việc lấy mẫu các bit bởi UART nhận.

Thay vì tín hiệu đồng hồ, UART truyền thêm các bit start và stop vào gói dữ liệu được chuyển. Các bit này xác định điểm bắt đầu và điểm kết thúc của gói dữ liệu để UART nhận biết khi nào bắt đầu đọc các bit.

Cách thức hoạt động của giao tiếp UART



UART sẽ truyền dữ liệu nhận được từ một bus dữ liệu (Data Bus). Bus dữ liệu được sử dụng để gửi dữ liệu đến UART bởi một thiết bị khác như CPU, bộ nhớ hoặc vi điều khiển. Dữ liệu được chuyển từ bus dữ liệu đến UART truyền ở dạng song song.

Sau khi UART truyền nhận dữ liệu song song từ bus dữ liệu, nó sẽ thêm một bit start, một bit chẵn lẻ và một bit stop, tạo ra gói dữ liệu. Tiếp theo, gói dữ liệu được xuất ra nối tiếp từng bit tại chân Tx. UART nhận đọc gói dữ liệu từng bit tại chân Rx của nó. UART nhận sau đó chuyển đổi dữ liệu trở lại dạng song song và loại bỏ bit start, bit chẵn lẻ và bit stop. Cuối cùng, UART nhận chuyển gói dữ liệu song song với bus dữ liệu ở đầu nhận.

Dữ liệu truyền qua UART được tập hợp thành gói (packet). Mỗi gói chứa 1 bit start, 5 đến 9 bit dữ liệu (tùy thuộc vào UART), một bit chẵn lẻ (parity bit) tùy chọn và 1 hoặc 2 bit stop.

Các bước truyền dữ liệu trong UART:

- UART truyền nhận dữ liệu song song từ bus dữ liệu.
- UART truyền thêm bit start, bit chẵn lẻ và bit dừng vào khung dữ liệu.
- Toàn bộ gói được gửi nối tiếp từ UART truyền đến UART nhận. UART nhận lấy mẫu đường dữ liệu ở tốc độ truyền được định cấu hình trước.
- UART nhận loại bỏ bit start, bit chẵn lẻ và bit stop khỏi khung dữ liệu.
- UART nhận chuyển đổi dữ liệu nối tiếp trở lại thành song song và chuyển nó đến bus dữ liệu ở đầu nhận.

Ưu điểm	Nhược điểm
• Chỉ sử dụng hai dây để truyền dữ liệu • Không cần xung clock • Có 1 bit chẵn lẻ để kiểm tra lỗi • Cấu trúc gói dữ liệu có thể thay đổi • Phương pháp truyền đơn giản, giá thành thấp.	• Kích thước của khung dữ liệu tối đa 9 bit • Không phù hợp với hệ thống nhiều thiết bị truyền và nhận • Tốc độ truyền của mỗi UART phải nằm trong khoảng 10%.

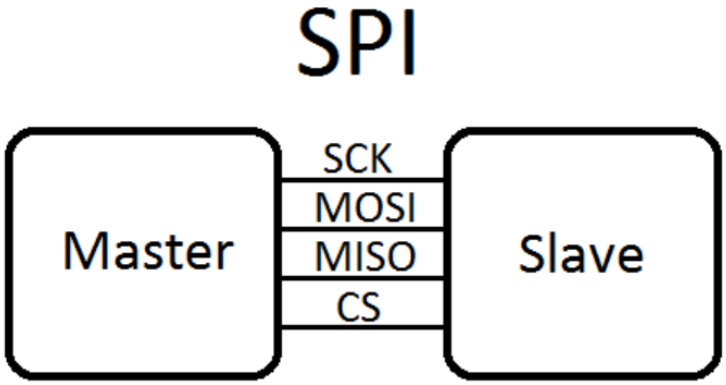
Ứng dụng:

Giao tiếp 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 truyền thông khác nhau như giao tiếp không dây, thiết bị GPS, module Bluetooth và nhiều ứng dụng khác.

1.4.2 SPI

SPI (Serial Peripheral Interface): là một chuẩn truyền thông nối tiếp đồng bộ được sử dụng để truyền dữ liệu trong khoảng cách ngắn, thường được sử dụng trong các hệ thống nhúng. SPI được phát triển bởi Motorola. Các ứng dụng tiêu biểu của SPI có thể kể đến như thẻ nhớ (Secure Digital cards) và giao tiếp màn hình LCD.

SPI là giao tiếp đồng bộ, bất cứ quá trình nào cũng đều được đồng bộ với xung clock sinh ra bởi thiết bị Master . Không cần phải lo lắng về tốc độ truyền dữ liệu.

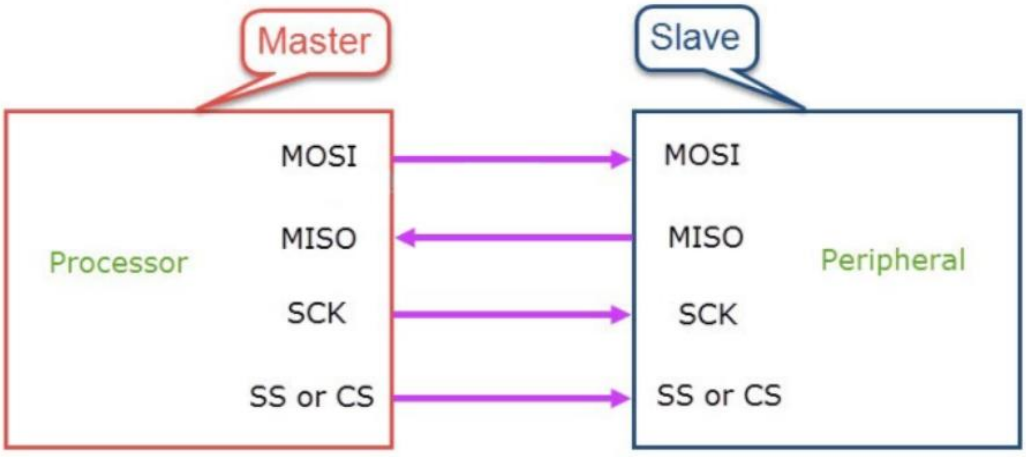


SPI thường được sử dụng giao tiếp với bộ nhớ EEPROM, RTC (Đồng hồ thời gian thực), IC âm thanh, các loại cảm biến như nhiệt độ và áp suất, thẻ nhớ như MMC hoặc thẻ SD hoặc thậm chí các bộ vi điều khiển khác.

Cấu tạo của SPI:

Sử dụng 4 đường dây giao tiếp:

- SCK (Serial Clock): Thiết bị Master tạo xung tín hiệu SCK và cung cấp cho Slave. Xung này có chức năng giữ nhịp cho giao tiếp SPI. Mỗi nhịp trên chân SCK báo 1 bit dữ liệu đến hoặc đi → Quá trình ít bị lỗi và tốc độ truyền cao
- MISO (Master Input Slave Output): Tín hiệu tạo bởi thiết bị Slave và nhận bởi thiết bị Master. Đường MISO phải được kết nối giữa thiết bị Master và Slave
- MOSI (Master Output Slave Input): Tín hiệu tạo bởi thiết bị Master và nhận bởi thiết bị Slave. Đường MOSI phải được kết nối giữa thiết bị Master và Slave.
- SS (Slave Select): Chọn thiết bị Slave cụ thể để giao tiếp. Để chọn Slave giao tiếp thiết bị Master chủ động kéo đường SS tương ứng xuống mức 0 (Low). Chân này đôi khi còn được gọi là CS (Chip Select). Chân SS của vi điều khiển (Master) có thể được người dùng tạo bằng cách cấu hình 1 chân GPIO bất kỳ chế độ Output.



Các bước truyền dữ liệu SPI:

- Master ra tín hiệu xung nhịp.
- Master chuyển chân SS / CS sang trạng thái điện áp thấp, điều này sẽ kích hoạt slave.
- Master gửi dữ liệu từng bit một tới slave dọc theo đường MOSI. Slave đọc các bit khi nó nhận được.
- 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.

Ưu điểm	Nhược 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	• 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 SPI

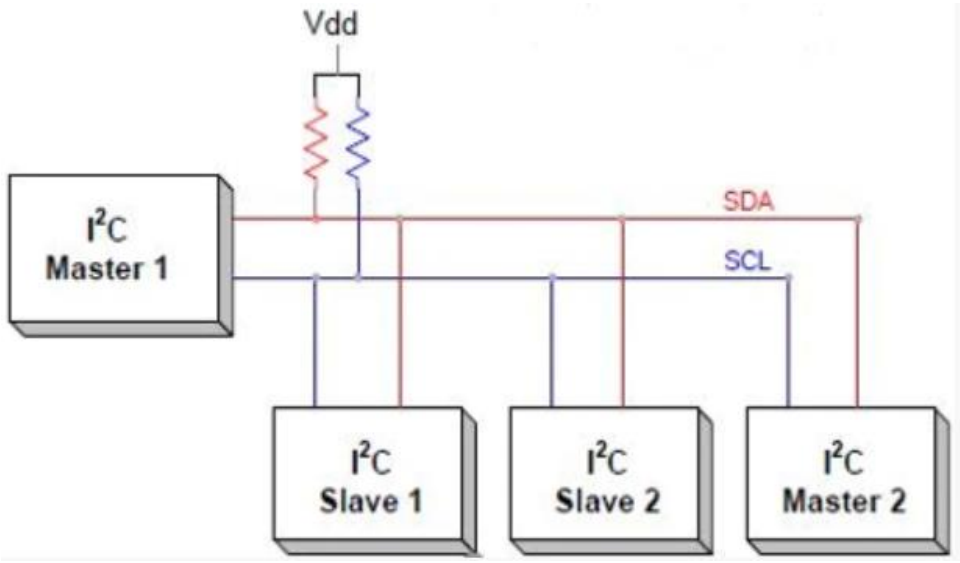
1.4.3 I2C

I2C (Inter – Integrated Circuit) là 1 giao thức giao tiếp nối tiếp đồng bộ được phát triển bởi Philips Semiconductors, sử dụng để truyền nhận dữ liệu giữa các IC với nhau chỉ sử dụng hai đường truyền tín hiệu.

Cấu tạo của I2C:

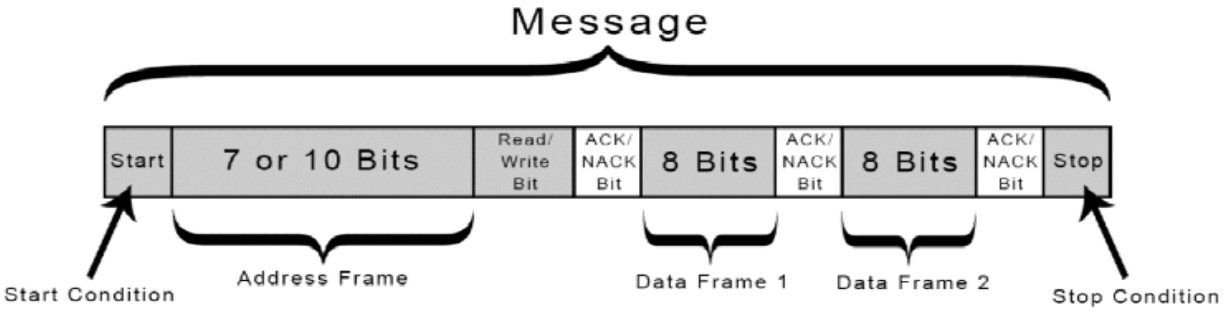
I2C sử dụng 2 đường truyền tín hiệu:

- SCL - Serial Clock Line : Tạo xung nhịp đồng hồ do Master phát đi
- SDA - Serial Data Line : Đường truyền nhận dữ liệu.



Giao tiếp I2C bao gồm quá trình truyền nhận dữ liệu giữa các thiết bị chủ tớ, hay Master - Slave. Thiết bị Master là 1 vi điều khiển, nó có nhiệm vụ điều khiển đường tín hiệu SCL và gửi nhận dữ liệu hay lệnh thông qua đường SDA đến các thiết bị khác. Các thiết bị nhận các dữ liệu lệnh và tín hiệu từ thiết bị Master được gọi là các thiết bị Slave. Các thiết bị Slave thường là các IC, hoặc thậm chí là vi điều khiển.

Cách hoạt động của I2C:



Với I2C, dữ liệu được truyền trong các tin nhắn. Tin nhắn được chia thành các khung dữ liệu. Mỗi tin nhắn có một khung địa chỉ chứa địa chỉ nhị phân của địa chỉ slave và một hoặc nhiều khung dữ liệu chứa dữ liệu đang được truyền. Thông điệp cũng bao gồm điều kiện khởi động và điều kiện dừng, các bit đọc / ghi và các bit ACK / NACK giữa mỗi khung dữ liệu:

- Khối bit địa chỉ: Thông thường quá trình truyền nhận sẽ diễn ra với rất nhiều thiết bị, IC với nhau. Do đó để phân biệt các thiết bị này, chúng sẽ được gắn 1 địa chỉ vật lý 7 bit cố định
- Bit Read/Write: Bit này dùng để xác định quá trình là truyền hay nhận dữ liệu từ thiết bị Master. Nếu Master gửi dữ liệu đi thì ứng với bit này bằng '0', và ngược lại, nhận dữ liệu khi bit này bằng '1'.
- Bit ACK/NACK: Dùng để so sánh bit địa chỉ vật lý của thiết bị so với địa chỉ được gửi tới. Nếu trùng thì Slave sẽ được đặt bằng '0' và ngược lại, nếu không thì mặc định bằng '1'.
- Khối bit dữ liệu: Gồm 8 bit và được thiết lập bởi thiết bị gửi truyền đến thiết bị nhân. Sau khi các bit này được gửi đi, lập tức 1 bit ACK/NACK được gửi ngay theo sau để xác nhận rằng thiết bị nhận đã nhận được dữ liệu thành công hay chưa. Nếu nhận thành công thì bit ACK/NACK được set bằng '0' và ngược lại.

Các bước truyền dữ liệu:

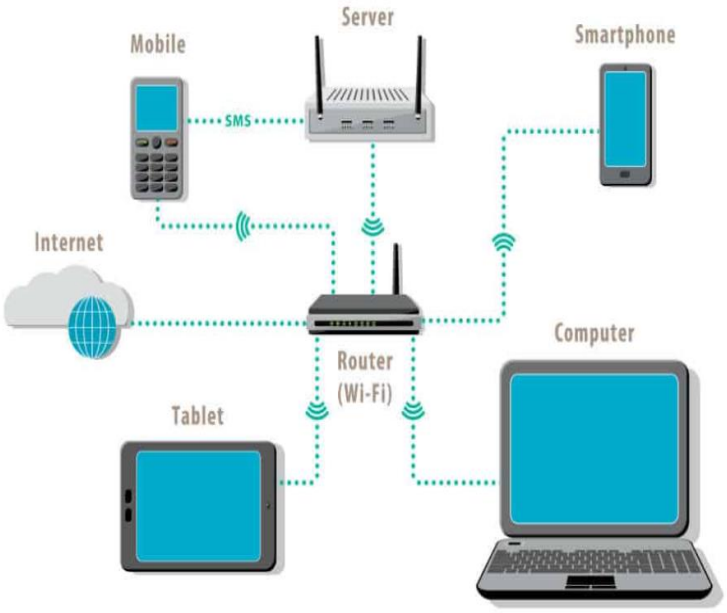
- Master gửi điều kiện khởi động đến mọi slave được kết nối bằng cách chuyển đường SDA từ mức điện áp cao sang mức điện áp thấp trước khi chuyển đường SCL từ mức cao xuống mức thấp.
- Master gửi cho mỗi slave địa chỉ 7 hoặc 10 bit của slave mà nó muốn giao tiếp, cùng với bit đọc / ghi.
- Mỗi slave sẽ so sánh địa chỉ được gửi từ master với địa chỉ của chính nó. Nếu địa chỉ trùng khớp, slave sẽ trả về một bit ACK bằng cách kéo dòng SDA xuống thấp cho một bit. Nếu địa chỉ từ master không khớp với địa chỉ của slave, slave rời khỏi đường SDA cao.

- Master gửi hoặc nhận khung dữ liệu.
- Sau khi mỗi khung dữ liệu được chuyển, thiết bị nhận trả về một bit ACK khác cho thiết bị gửi để xác nhận đã nhận thành công khung.
- Để dừng truyền dữ liệu, master gửi điều kiện dừng đến slave bằng cách chuyển đổi mức cao SCL trước khi chuyển mức cao SDA.

Ưu điểm	Nhược điểm
• Sử dụng chỉ hai dây • Hỗ trợ nhiều máy chủ chính và nhiều máy chủ nô lệ • Bit ACK / NACK xác nhận rằng mỗi khung hình đã được truyền thành công • Phần cứng không phức tạp như UART • Giao thức nổi tiếng và được sử dụng rộng rãi	• 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

1.4.4 LAN

LAN (Local Area Network) 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…



Cách hoạt động của LAN:

Việc kết nối các máy tính với một dây cáp được dùng như một phương tiện truyền tin chung cho tất cả các máy tính. Công việc kết nối vật lý vào mạng được thực hiện bằng cách cắm một card giao tiếp mạng NIC (Network Interface Card) vào trong máy tính và nối nó với cáp mạng. Sau khi kết nối vật lý đã hoàn tất, quản lý việc truyền tin giữa các trạm trên mạng tuỳ thuộc vào phần mềm mạng.

Khi một máy muốn gửi một thông điệp cho máy khác thì nó sẽ dùng một phần mềm trong máy nào đó đặt thông điệp vào một gói tin (packet) bao gồm dữ liệu thông điệp được bao bọc giữa tín hiệu đầu và tín hiệu cuối và dùng phần mềm mạng để gửi gói tin đó đến máy đích.

NIC sẽ chuyển gói tín hiệu vào mạng LAN, gói tín hiệu được truyền đi như một

dòng các bit dữ liệu, khi nó chạy trong cáp chung mọi máy đều nhận được tín hiệu này.

NIC ở mỗi trạm sẽ kiểm tra địa chỉ đích trong tín hiệu đầu của gói để xác định đúng địa chỉ đến, khi gói tín hiệu đi tới máy có địa chỉ cần đến, đích ở máy đó sẽ sao gói tín hiệu rồi lấy dữ liệu ra khỏi gói tin và đưa vào máy tính.

Ưu điểm	Nhược điểm
• Tốc độ cao • Có băng thông lớn, chạy được các ứng dụng trực tuyến được kết nối thông qua mạng như các cuộc hội thảo, chiếu phim… • Chi phí di trì thấp, cách thức quản trị mạng đơn giản. • Đa dạng về cấu trúc lắp đặt mạng như: Bus, ring, tree,…	• Đường truyền ngắn, chỉ có thể hoạt động trong một diện tích nhất định. • Chi phí lắp đặt ban đầu cao. • Một số topo mạng có tính bảo mật khá thấp, khó phát hiện lỗi khi gặp vấn đề.

Ứng dụng:

- Giúp các máy tính trong cùng không gian có thể sử dụng mạng internet
- Sử dụng chung các thiết bị ngoại vi như máy in, máy scan, camera,…
- Chia sẻ dữ liệu không cần thiết bị ngoại vi: ổ cứng di động, USB, thẻ nhớ,…
- Bảo mật tốt hơn vì dễ dàng thiết lập tường lửa.
- Sử dụng trong nhà, công ty, môi trường nhỏm,…

2. PHẦN MỀM HỆ THỐNG NHÚNG

2.1.Hệ điều hành nhúng

Về bản chất, một hệ điều hành nhúng về bản chất là một hệ điều hành bị tước bỏ với một số lượng hạn chế các tính năng. Nó thường được thiết kế cho các chức năng rất cụ thể để điều khiển một thiết bị điện tử. Nó xử lý tất cả các giao diện cơ bản và các tính năng của điện thoại.

Hệ điều hành nhúng có thể là hệ điều hành bằng văn bản tùy chỉnh cụ thể cho thiết bị hoặc một trong vô số các hệ điều hành có mục đích chung đã được sửa đổi để chạy trên đầu thiết bị. Các hệ điều hành nhúng phổ biến bao gồm RTOS, Android (điện thoại di động) và Linux.

2.1.1 Hệ điều hành thời gian thực- RealTime Operating Systems(RTOS)

RTOS là hệ điều hành thời gian thực (Real-time operating system) nhằm phục vụ các ứng dụng thời gian thực, với việc xử lý dữ liệu đầu vào mà không có sự chậm trễ của bộ đệm (buffer). Các yêu cầu về thời gian xử lý (bao gồm cả sự chậm trễ của hệ điều hành) được tính bằng phần mười của giây hoặc bằng thời gian ngắn hơn nữa.

Một hệ thống thời gian thực là một hệ thống giới hạn thời gian với các các ràng buộc thời gian cố định được định nghĩa rõ ràng. Quá trình xử lý phải được thực hiện trong một khoản thời gian cố định, nếu không thì hệ thống sẽ gặp sự cố. Việc này có thể được thực hiện thông qua cơ chế hướng sự kiện (event-driven) hoặc chia sẻ thời gian (time-sharing). Các hệ thống hướng sự kiện sẽ chuyển đổi giữa các tác vụ (task) nhiệm vụ dựa trên độ ưu tiên của chúng trong khi các hệ thống chia sẻ thời gian sẽ chuyển đổi các tác vụ dựa trên ngắt của xung nhịp.

Các chức năng cơ bản của một RTOS:

- Bộ lập lịch (Scheduler).
- Các dịch vụ thời gian thực (Realtime Services).
- Đồng bộ và xử lý thông điệp (Synchronization and Messaging).

Bộ lập lịch:

Để lập lịch cho Task, có 3 kỹ thuật được áp dụng:

- Co-operative scheduling: Mỗi task được thực thi đến khi kết thúc quá trình thì task tiếp theo mới được thực thi.
- Round Robin Scheduling: Mỗi task được chia cho một khe thời gian cố định, nếu trong khoảng thời gian được chia đó mà task chưa thực hiện xong thì sẽ bị tạm dừng, chờ đến lượt tiếp theo để thực hiện tiếp công việc sau khi hệ thống xử lý hết một lượt các task.
- Preemptive Scheduling: Phương pháp này ưu tiên phân bổ thời gian cho các task có mức ưu tiên cao hơn. Mỗi task được gán 1 mức ưu tiên duy nhất. Có thể có 256 mức ưu tiên trong hệ thống, và có thể có nhiều task có cùng mức ưu tiên.

Các dịch vụ thời gian thực :

Kernel là trái tim của mỗi hệ điều hành. Nó thực hiện chức năng quản lý và lập lịch các quá trình sử dụng CPU và điều phối tài nguyên. Mỗi task chỉ được thực thi bởi CPU trong một khoảng thời gian, trong các khoảng thời gian còn lại thì task được quản lý bởi các dịch vụ quản lý của hệ điều hành. Các dịch vụ của RTOS bao gồm:

- Xử lý ngắt (Interrupt handling services).
- Dịch vụ quản lý thời gian (Time services).
- Dịch vụ quản lý thiết bị (Device management services).
- Dịch vụ quản lý bộ nhớ (Memory management services).
- Dịch vụ quản lý các kết nối Vào - Ra (IO services).

Đồng bộ và xử lý thông điệp:

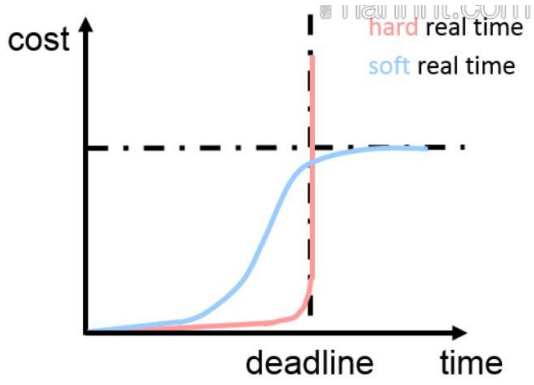
Các thông điệp sử dụng để trao đổi thông tin giữa các hệ thống khác nhau hoặc giữa các task. Các dịch vụ quản lý thông điệp bao gồm:

- Semaphores: Dùng để đồng bộ hóa quyền truy cập vào các tài nguyên dùng chung.
- Event Flags: Dùng để đồng bộ hóa các hoạt động cần có sự phối hợp của nhiều task.
- Mailboxes, Pipes, Message queues: Dùng để quản lý các thông điệp gửi đi - đến giữa các task.

Các dạng thời gian thực :

Một hệ thống có thể chia làm 2 loại thời gian thực dựa vào mức độ trễ - được gọi là thời hạn lập danh mục (scheduling deadline) :

- Hard - Realtime: hệ thống phải tiếp nhận và nắm bắt được scheduling deadline của nó tại mỗi và mọi thời điểm. Sự sai sót trong việc tiếp nhận deadline có thể dẫn đến hậu quả nghiêm trọng.
- Soft - Realtime: scheduling deadline có dễ thở hơn chút ít, với softrealtime thì không có gì quá nghiêm trọng xảy ra nếu hệ thống thỉnh thoảng bị trễ. Lỗi về mặt thời gian có thể chỉ đơn giản là dẫn đến hậu quả giảm độ tin cậy của đối tượng đối với hệ thống mà không dẫn đến hậu quả nghiêm trọng nào khác xảy ra.



2.1.2 Hệ điều hành Android

Android là một hệ điều hành dựa trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính

bảng..Android là một hệ điều hành được viết từ JAVA, mã nguồn mở hoàn toàn giúp cho các lập trình viên cài đặt các ứng dụng trên thiết bị do chính mình viết ra.

Kiến trúc của Android:

Android bao gồm 4 tầng từ dưới lên trên là tầng hạt nhân Linux, tầng Libraries & Android runtime, tầng Application Framework và tầng Application.

Tầng hạt nhân Linux

Lớp Linux cung cấp 1 cấp độ trừu tượng giữa phần cứng của thiết bị và các thành trình điều khiển phần cứng thiết yếu như máy ảnh, bàn phím, màn hình hiển thị... Đồng thời, hạt nhân (kernel) còn xử lý tất cả các thứ mà Linux có thể làm tốt như mạng kết nối và 1 chuỗi các trình điều khiển thiết bị, giúp cho giao tiếp với các thiết bị ngoại vi dễ dàng hơn.

Tầng Libraries & Android runtime

Tập các thư viện bao gồm WebKit - trình duyệt Web mã nguồn mở, được biết đến như thư viện libc, cơ sở dữ liệu SQLite - hữu dụng cho việc lưu trữ và chia sẻ dữ liệu ứng dụng, các thư viện chơi và ghi âm audio, video, hay các thư viện SSL chịu trách nhiệm bảo mật Internet...

Tầng Application Framework

Lớp Android Framework cung cấp các dịch vụ cấp độ cao hơn cho các ứng dụng dưới dạng lớp Java. Các nhà phát triển ứng dụng được phép sử dụng các dịch vụ này trong ứng dụng của họ.

Android Framework bao gồm các dịch vụ chính sau: Activitty Manager, Content Providers, Resource Manager , Content Providers , Resource Manager, Notifications Manager, View System

Tầng Application

Lớp trên cùng của kiến trúc là Application. Các ứng dụng bạn tạo ra sẽ được cài đặt trên lớp này. Ví dụ như: Danh bạ, nhắn tin, trò chơi...

Các chức năng của Android:

Android sở hữu giao diện người dùng mặc định gồm có những thao tác sử dụng trực tiếp trên màn hình như vuốt, chạm hay Hơn, hệ thống thông báo rung để phản hồi các nút điều hướng, khởi động thiết bị, biểu tượng,…

Android OS cũng bao gồm các tính năng để tiết kiệm pin sử dụng. Android ngoài ra còn có thể quản lý bộ nhớ, tự động đóng các quá trình không hoạt động được lưu trữ trong bộ nhớ của nó,hỗ trợ các tính năng: Bluetooth, Edge, 3G, Wifi, Tin nhắn SMS và MMS,Máy quay Video, máy ảnh kỹ thuật số,GPS,La bàn,Máy đo gia tốc,Đồ họa 3D tăng tốc,Ứng dụng đa nhiệm...

2.1.3 Hệ điều hành Linux

Linux là một hệ điều hành mã nguồn mở (open-source OS) chạy trên hầu hết các kiến trúc vi xử lý, bao gồm dòng vi xử lý ARM. Linux được hỗ trợ bởi một cộng đồng mã nguồn mở (GNU), chính điều này làm cho Linux rất linh hoạt và phát triển rất nhanh với nhiều tính năng không thua kém các hệ điều hành khác hiện nay.

Hầu hết các bản Linux đều hỗ trợ rất nhiều ngôn ngữ lập trình, đặc biệt công cụ GCC cho phép người lập trình có thể biên dịch và thực thi ứng dụng viết bằng nhiều ngôn ngữ lập trình khác nhau: C/C++, Java, .v.v... Ngoài ra, Linux còn hỗ trợ ngôn ngữ lập trình để phát triển các ứng dụng đồ họa như: JTK+, Qt, .v.v...

Ngoài các thiết kế hệ điều hành dành cho máy tính desktop và máy chủ (server), các nhà cung cấp hệ điều hành linux còn thiết kế ra các bản chuyên biệt cho các mục đích khác nhau như: hỗ trợ kiến trúc máy tính, hệ thống nhúng (embedded Linux system), các hệ thống bảo mật, an ninh mạng, .v.v…

Do chi phí thấp, khả năng thay đổi tương thích với phần cứng dễ dàng, nên nhúng Linux thường được sử dụng rất nhiều trong các hệ thống nhúng (embedded system)

Kiến trúc của Linux

Thông thường, kiến trúc Linux được chia làm 3 thành phần chính, đó là: Kernel, Shell, Applications.

Kernel:

Đây là thành phần quan trọng của mọi hệ điều hành, và được ví như trái tim của HĐH, kernel sẽ chứa các module hay các thư viện để quản lý và giao tiếp giữa phần cứng máy tính và các ứng dụng.

Vì được phát hành với bản quyền GNU - General Public License. Do đó mà bất cứ ai cũng có thể tải và xem mã nguồn của Linux, thậm chí việc chỉnh sửa và phát triển một distro riêng cho mình. Và tính tới thời điểm bài viết này, đã có hơn 300 distro được xuất bản.

Shell:

Nằm trên Kernel đó chính là Shell. Đây là một chương trình có chức năng thực thi các lệnh (command) từ người dùng hoặc từ các ứng dụng yêu cầu, chuyển đến cho Kernel xử lý. Có thể hiểu Shell chính là trung gian nằm giữa Kernel và Application, có nhiệm vụ "phiên dịch" các lệnh từ Application gửi đến Kernel để thực thi.Các loại shell như sau:

- sh (the Bourne Shell): đây là shell nguyên thủy của UNIX được viết bởi Stephen Bourne vào năm 1974. Đến nay shell sh vẫn sử dụng rộng rãi.
- bash(Bourne-again shell): đây là shell mặc định trên linux.
- csh (C shell): shell được viết bằng ngôn ngữ lập trình C, được viết bởi Bill Joy vào năm 1978.

Ngoài ra còn có các loại shell khác như: ash (Almquist shell), tsh (TENEX C shell), zsh (Z shell).

Appication:

Application chính là các ứng dụng, phần mềm, và tiện ích mà người dùng cài đặt trên máy và sử dụng nó hằng này. Trên Kali, các công cụ chính là các Application, trình duyệt cũng là Application, hay chính giao diện GNOME của bạn đang sử dụng cũng là Application..

Chức năng của Linux :

Tương tự như các hệ điều hành khác, Linux cũng cấp môi trường trung gian để người dùng có thể giao tiếp với phần cứng máy tính, thực hiện các công việc của mình.

Bên cạnh đó, nhờ ứng dụng mã nguồn mở mà Linux đem lại nhiều sự thoải mái hơn cho người dùng, đặc biệt các lập trình viên, nhà phát triển.

Các phiên bản của Linux :

Hệ điều hành Linux có rất nhiều phiên bản do người dùng tạo ra : Ubuntu:, Linux Mint, Debian, Fedora, CentOS, LinuxOpenSUSE/SUSE Linux EnterMagei...

2.2 Ngôn ngữ lập trình và SDK

SDK (Software Development Kit) là các công cụ và phần mềm dùng để phát triển ứng dụng thông qua một nền tảng nhất định. SDK cung cấp các thư viện, tài liệu, mẫu template, sample code, tiện ích gỡ rối (debugging), các ghi chú hỗ trợ (documentation) hoặc các tài liệu bổ sung,… để nhà phát triển có thể tích hợp thêm vào phần mềm/ứng dụng của mình. SDK được xây dựng tùy chỉnh sao cho tương thích với ngôn ngữ lập trình và các đặc điểm tương ứng.

Flutter là một bộ SDK đa nền tảng được google phát triển và phát hành vào tháng 5 năm 2017. Flutter sử dụng ngôn ngữ lập trình Dart của Google và đi kèm với thư viện đồ họa và thiết kế material design của riêng mình, cho phép phát triển ứng dụng nhanh hơn và cho ra một sản phẩm hoàn thiện hơn.Các ứng của Flutter có thể hoạt động trên cả các nền tảng iOS, Android, windows,...

- Flutter sử dụng Dart, một ngôn ngữ nhanh, hướng đối tượng với nhiều tính năng hữu ích như mixin, generic, isolate, và static type.
- Flutter có các thành phần UI của riêng nó, cùng với một cơ chế để kết xuất chúng trên nền tảng Android và iOS. Hầu hết các thành phần giao diện người dùng, đều sẵn dùng, phù hợp với các nguyên tắc của Material Design.

- Các ứng dụng Flutter có thể được phát triển bằng cách4 sử dụng IntelliJ IDEA, một IDE rất giống với Android Studio.
- Fast Development: Tính năng Hot Reload hoạt động trong milliseconds để hiện thị giao diện tới bạn. Sử dụng tập hợp các widget có thể customizable để xây dựng giao diện trong vài phút. Ngoài ra Hot Reload còn giúp bạn thêm các tính năng, fix bug tiết kiệm thời gian hơn mà không cần phải thông qua máy ảo, máy android hoặc iOS.
- Expressive and Flexible UI: Có rất nhiều các thành phần để xây dựng giao diện của Flutter vô cùng đẹp mắt theo phong cách Material Design và Cupertino, hỗ trợ nhiều các APIs chuyển động, smooth scrolling...
- Native Performance: Các widget của fluter kết hợp các sự khác biệt của các nền tảng ví dụ như scrolling, navigation, icons, font để cung cấp một hiệu năng tốt nhất tới iOS và Android.

IDE (Integrated Development Environment) là môi trường tích hợp dùng để viết code để phát triển ứng dụng. Ngoài ra IDE tích hợp các tool hỗ trợ khác như trình biên dịch (Compiler), trình thông dịch (Interpreter), kiểm tra lỗi (Debugger), định dạng hoặc highlight code, tổ chức thư mục code, tìm kiếm code…

Visual Studio Code (VS Code) là IDE cho phép biên tập, soạn thảo các đoạn code để hỗ trợ trong quá trình thực hiện xây dựng, thiết kế website một cách nhanh chóng. Visual Studio Code hay còn được viết tắt là VS Code. Trình soạn thảo này vận hành mượt mà trên các nền tảng như Windows, macOS, Linux. Hơn thế nữa, VS Code còn cho khả năng tương thích với những thiết bị máy tính có cấu hình tầm trung vẫn có thể sử dụng dễ dàng.

Visual Studio Code hỗ trợ đa dạng các chức năng Debug, đi kèm với Git, có Syntax Highlighting. Đặc biệt là tự hoàn thành mã thông minh, Snippets, và khả năng cải tiến mã nguồn. Nhờ tính năng tùy chỉnh, Visual Studio Code cũng cho phép các lập trình

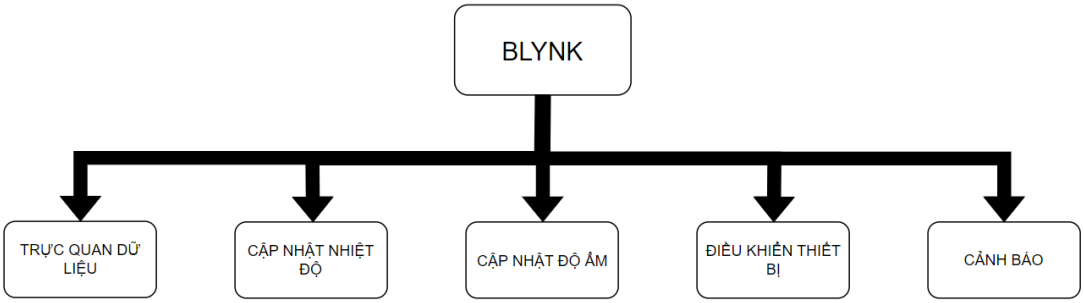
viên thay đổi Theme, phím tắt, và đa dạng các tùy chọn khác. Mặc dù trình soạn thảo Code này tương đối nhẹ, nhưng lại bao gồm các tính năng mạnh mẽ.

Visual Studio Code cũng luôn có những cải tiến và tạo ra đa dạng các tiện ích đi kèm từ đó giúp cho các lập trình viên sử dụng dễ dàng hơn. Trong đó có thể kể đến những ưu điểm sau:

- Đa dạng ngôn ngữ lập trình giúp người dùng thỏa sức sáng tạo và sử dụng như HTML, CSS, JavaScript, C++,…
- Ngôn ngữ, giao diện tối giản, thân thiện, giúp các lập trình viên dễ dàng định hình nội dung.
- Các tiện ích mở rộng rất đa dạng và phong phú.
- Tích hợp các tính năng quan trọng như tính năng bảo mật (Git), khả năng tăng tốc xử lý vòng lặp (Debug),…
- Đơn giản hóa việc tìm quản lý hết tất cả các Code có trên hệ thống.

3. BÀI TẬP

3.1. Chức năng phần mềm:

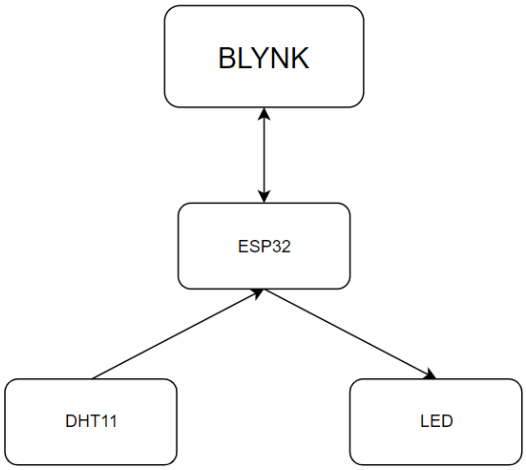


Mục đích chính của code gồm có:

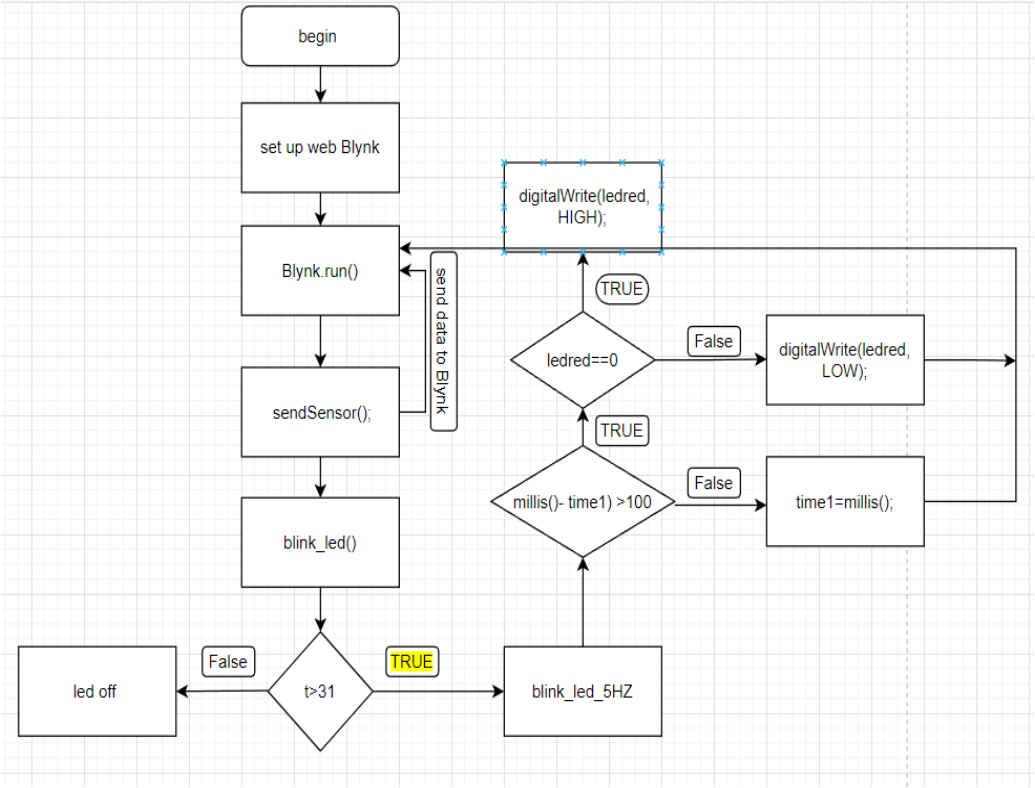
1. Điều khiển từ xa (ở bất cứ nơi nào, chỉ cần thiết bị điều khiển kết nối wifi là được) 2 công tắt online qua wifi (khi esp32 được kết nối wifi).
2. Cập nhật cảm biến DHT11 về nhiệt độ và độ ẩm liên tục, dữ liệu sẽ được gửi lên trang web Blynk, từ đó ta sẽ vẽ đồ thị để dễ nhìn và đánh giá dữ liệu.
3. Đèn led chớp tắt liên tục khi nhiệt độ lớn hơn 31 độ. Trong quá trình chớp tắt led, chương trình vẫn sẽ hoạt động bình thường mà không bị delay. Các công tắt điều khiển led vẫn hoạt động bình thường.

3.2. Lưu đồ thuật toán

Sơ đồ hệ thống Blynk - ESP:



Lưu đồ thuật toán chương trình:



3.3. Code:

Code	Giải thích

`#define BLYNK_TEMPLATE_ID "TMPL866kA9tK"`
`#define BLYNK_DEVICE_NAME "Quickstart Template"`
`#define BLYNK_AUTH_TOKEN "5kw7sHgRaM9TtjzW-5fvGT5OqIvzxYs-"`
 	
`#define BLYNK_PRINT Serial`
`#include <WiFi.h>`
`#include <WiFiClient.h>`
`#include <BlynkSimpleEsp32.h>`
	Thiết lập cho firmware để kết nối với device đã tạo trên Blynk Cloud với ID, tên device và Token tương ứng
	
	Các thư viện cần thiết cho việc kết nối với Blynk, WiFi và WiFiClient và BlynkSimpleEsp32 nằm trong gói thư viện

`#include "DHT.h"`	Blynk cài trên Arduino Thư viện cảm biến DHT

`char ssid[] = "Lau2";`
`char pass[] = "03012004";`
 	
`#define DHTTYPE DHT11`
`#define DHTPIN 4`
`#define ledred GPIO_NUM_18`
 	
`DHT dht(DHTPIN, DHTTYPE);`
`BlynkTimer timer;`
	-Thiết lập chuỗi SSID và pass tương ứng với mạng Wifi mà Esp32 sẽ truy cập để có thể kết nối

`void blink_led()`
`{`
`  if (t>31)`
`  {`
`    if ( (unsigned long) (millis()-time1) >100)`
`    {`
`      if (digitalRead(ledred)==0)`
`    {`
`        digitalWrite(ledred, HIGH);`
`    }`
`      else`
`    {`
	-Khi nhiệt độ tăng quá 31°C ESP sẽ truyền tín hiệu lên Blynk
	-Blynk lúc này sẽ gửi tín hiệu xuống ESP để nháy đèn cảnh báo

digitalWrite(ledred, LOW);
}
time1=millis();
}
}
if (t<=31) {
digitalWrite(ledred, LOW);
}
}	

void sendSensor()
{
float h = dht.readHumidity();
 	
float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
 	
if (isnan(h) || isnan(t)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
Blynk.virtualWrite(V5, t);
Blynk.virtualWrite(V6, h);
}
	Hàm Sensor
	
	-Biến float h được khai báo như một đối tượng của class DHT, sử dụng method readHumidity() để đọc chỉ số về độ ẩm từ cảm biến
	-Biến float t tương tự sử dụng method readTemperature() để đọc chỉ số nhiệt độ từ cảm biến
	
	-Điều kiện nếu
	Thì in ra màn hình dòng chữ “Failed to read from the DHT sensor!”
	
	Ghi giá trị từ 2 biến t và h vào các ngõ ra tương ứng

void setup()
{
// Debug console
Serial.begin(9600);
	-Hàm setup, thiết lập các thông số, chuẩn bị cho chương trình chạy
	
	-Thiết lập giao tiếp nối giữa esp32 và cổng

dht.begin();
timer.setInterval(1000L, sendSensor);
Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
pinMode(ledred, OUTPUT);
time1=millis();
}
	truyền dữ liệu với baud rate 9600
	-Gọi đến hàm dht, chạy hàm
	-Setup hàm được gọi mỗi 1000ms = 1s
	-Chạy Blynk với các tham số đầu vào tương ứng là token của device trên blink, ssid và pass của wifi

void loop()
{
sendSensor();
Blynk.run();
Blink_led();
}
	Chương trình chính chạy ở đây
	
	-Chạy hàm sendSensor
	-Chạy Blynk

TÀI LIỆU THAM KHẢO

[1] “DevIoT,” [Trực tuyến]. Available: https://deviot.vn/. [Đã truy cập 2 12 2022].

[2] “IBM,” 18 12 2019. [Trực tuyến]. Available: https://www.ibm.com/support/pages/what-dedicated-processor. [Đã truy cập 2 12 2022].

[3] “Easy2teach,” 28 12 2011. [Trực tuyến]. Available: http://easy2teach.blogspot.com/2011/12/what-is-single-purpose-processor.html. [Đã truy cập 2 12 2022].

[4] “Sciencedirect,” [Trực tuyến]. [Đã truy cập 2 12 2022].

[5] “VINAFE,” [Trực tuyến]. Available: https://dientutuonglai.com/chuan-giao-tiep-i2c-la-gi.html. [Đã truy cập 2 12 2022].