1.GIỚI THIỆU SƠ LƯỢC VỀ AUTOSAR CLASSIC

Chào các bạn, mình là Tùng Ngô và hôm nay mình sẽ giới thiệu về một chủ đề rất thú vị: AUTOSAR. Đây là một chuỗi video hướng dẫn chia sẻ kiến thức từ cơ bản đến nâng cao về AUTOSAR dành cho những người mới bắt đầu. Bài viết này sẽ giải thích chi tiết về AUTOSAR và tại sao nó quan trọng trong lập trình nhúng trên xe hơi. .

AUTOSAR là gì?

AUTOSAR (AUTomotive Open System ARchitecture) là một tiêu chuẩn kiến trúc mở cho phần mềm trên xe hơi. Tiêu chuẩn này được tạo ra để giải quyết vấn đề phức tạp ngày càng tăng của hệ thống điện tử và phần mềm trong xe hơi. Để dễ hiểu hơn, chúng ta sẽ dùng một số analogies (so sánh đơn giản) sau đây:

1. **Hệ thống phức tạp của xe hơi giống như một tòa nhà với nhiều phòng**: Mỗi phòng có một chức năng riêng (điều hòa, ánh sáng, an ninh, v.v.). Trong xe hơi, mỗi "phòng" này là một module phần mềm hoặc phần cứng điều khiển một chức năng cụ thể như động cơ, phanh, hoặc hệ thống giải trí.
2. **Chuẩn hóa giống như sử dụng cùng một loại phích cắm điện trong tất cả các phòng**: Trước khi có AUTOSAR, các hãng xe và nhà cung cấp phần mềm sử dụng các tiêu chuẩn riêng, giống như mỗi phòng trong tòa nhà sử dụng một loại phích cắm khác nhau. Điều này gây khó khăn khi cần sửa chữa hoặc nâng cấp. AUTOSAR cung cấp một "phích cắm" chung, giúp tất cả các module có thể dễ dàng kết nối và tương tác với nhau.

Lý do AUTOSAR ra đời

Ngày nay, hệ thống điện tử trong xe hơi ngày càng phức tạp và yêu cầu cao về an toàn, tốc độ, và kiểm soát khí thải. Do đó, AUTOSAR ra đời để chuẩn hóa và đơn giản hóa quá trình phát triển phần mềm nhúng trong xe hơi. Một số lý do chính bao gồm:

1. **Phức tạp của hệ thống tăng lên**: Giống như việc xây dựng một tòa nhà cao tầng cần có bản thiết kế chi tiết và chuẩn hóa để các tầng có thể hoạt động mượt mà với nhau, hệ thống điện tử trong xe cũng cần một kiến trúc chuẩn để quản lý các module khác nhau một cách hiệu quả.
2. **Yêu cầu an toàn và hiệu suất**: Giống như việc lắp đặt hệ thống báo cháy chuẩn trong tòa nhà để đảm bảo an toàn, AUTOSAR cung cấp các tiêu chuẩn giúp các module phần mềm tuân thủ yêu cầu an toàn và hiệu suất cao.
3. **Khả năng tái sử dụng và mở rộng**: Tương tự như việc sử dụng gạch chuẩn trong xây dựng để có thể mở rộng tòa nhà dễ dàng sau này, AUTOSAR cho phép các module phần mềm được tái sử dụng và dễ dàng nâng cấp.

Kiến trúc của AUTOSAR

Kiến trúc của AUTOSAR được chia thành ba lớp chính, giúp tách biệt các phần phụ thuộc vào phần cứng và phần mềm:

1. **Microcontroller Abstraction Layer (MCAL)**: Đây là lớp thấp nhất, tương đương với nền móng của tòa nhà, chịu trách nhiệm giao tiếp trực tiếp với phần cứng (vi điều khiển). MCAL cung cấp các hàm để điều khiển các thiết bị phần cứng cụ thể như chân GPIO, ADC, UART, v.v.
2. **Basic Software (BSW)**: Lớp này nằm trên MCAL và cung cấp các dịch vụ cơ bản, giống như hệ thống điện, nước trong tòa nhà. BSW bao gồm các module như OS, memory services, communication services, và system services.
3. **Runtime Environment (RTE)**: Lớp này giống như hệ thống quản lý trung tâm của tòa nhà, giúp các ứng dụng và dịch vụ khác nhau giao tiếp với nhau. RTE đảm bảo rằng các ứng dụng không cần phải biết về các chi tiết cụ thể của phần cứng mà chúng đang chạy trên.

Lợi ích của AUTOSAR

- **Tăng tính tương thích**: Giúp các module phần mềm từ các nhà cung cấp khác nhau có thể làm việc cùng nhau một cách trơn tru.
- **Giảm thời gian phát triển**: Cho phép tái sử dụng các module phần mềm đã phát triển trước đó, giúp tiết kiệm thời gian và công sức.
- **Nâng cao chất lượng và độ tin cậy**: Đảm bảo các tiêu chuẩn an toàn và chất lượng cao, giúp giảm lỗi và tăng độ tin cậy của hệ thống.

2.GIỚI THIỆU TỔNG QUAN VỀ SOFTWARE ARCHITECTURE

Chào các bạn, chào mừng các bạn đến với video thứ hai trong loạt video về AUTOSAR. Trong video này, chúng ta sẽ đi sâu hơn vào việc phát triển phần mềm nhúng theo hướng AUTOSAR và sẽ có một số ví dụ cụ thể để minh họa. Trước tiên, mình sẽ tóm tắt nhanh lại một số khái niệm quan trọng từ video trước để mọi người có thể theo dõi dễ dàng hơn.

Tổng quan về AUTOSAR

AUTOSAR là một tiêu chuẩn kiến trúc mở cho phần mềm trên xe hơi, giúp chuẩn hóa và đơn giản hóa việc phát triển phần mềm nhúng phức tạp. Mục tiêu của AUTOSAR là tăng tính tương thích, giảm thời gian phát triển, và nâng cao chất lượng phần mềm.

Ví dụ về lập trình nhúng đèn giao thông

Trong ví dụ ở video trước, chúng ta đã thảo luận về việc sử dụng đèn giao thông với các trạng thái đèn xanh và đèn đỏ. Bây giờ, mình sẽ giới thiệu chi tiết hơn về cách triển khai ví dụ này theo phong cách AUTOSAR.

1. **Giải pháp truyền thống**: Khi bạn lập trình theo cách truyền thống, bạn có thể viết một đoạn mã như sau để điều khiển đèn:

```
void timer_event() {
    if (timer == GREEN_LIGHT_TIMER) {
        // Tắt đèn đỏ
        GPIO_Pin_Write(RED_LED_PIN, LOW);
        // Bật đèn xanh
        GPIO_Pin_Write(GREEN_LED_PIN, HIGH);
    } else if (timer == RED_LIGHT_TIMER) {
        // Tắt đèn xanh
        GPIO_Pin_Write(GREEN_LED_PIN, LOW);
        // Bật đèn đỏ
        GPIO_Pin_Write(RED_LED_PIN, HIGH);
    }
}
```

2. **Giải pháp theo AUTOSAR**: Khi áp dụng AUTOSAR, mã sẽ được tách ra thành các module riêng biệt, đảm bảo tính tái sử dụng và dễ bảo trì:

- **Application Layer**: Lớp ứng dụng sẽ chỉ định các hành động cụ thể như bật hoặc tắt đèn mà không quan tâm đến chi tiết phần cứng.

```
void switch_red_light(bool state) {
    if (state) {
        Rte_Write_Port_LED_RED(ON);
    } else {
        Rte_Write_Port_LED_RED(OFF);
    }
}
```

```
void switch_green_light(bool state) {
    if (state) {
        Rte_Write_Port_LED_GREEN(ON);
    } else {
        Rte_Write_Port_LED_GREEN(OFF);
    }
}

void timer_event() {
    if (timer == GREEN_LIGHT_TIMER) {
        switch_red_light(false);
        switch_green_light(true);
    } else if (timer == RED_LIGHT_TIMER) {
        switch_green_light(false);
        switch_red_light(true);
    }
}
```

- **MCAL Layer**: Lớp này chứa mã phần cứng cụ thể, được trừu tượng hóa bởi AUTOSAR.

```
void Rte_Write_Port_LED_RED(bool state) {
    GPIO_Pin_Write(RED_LED_PIN, state);
}

void Rte_Write_Port_LED_GREEN(bool state) {
    GPIO_Pin_Write(GREEN_LED_PIN, state);
}
```

Kiến trúc của AUTOSAR

AUTOSAR chia kiến trúc phần mềm thành các lớp để tách biệt các phần phụ thuộc vào phần cứng và phần mềm:

1. **Microcontroller Abstraction Layer (MCAL)**: Tương tự như nền móng của tòa nhà, MCAL cung cấp giao diện để điều khiển phần cứng.
2. **Basic Software (BSW)**: Giống như hệ thống điện và nước trong tòa nhà, BSW cung cấp các dịch vụ cơ bản.
3. **Runtime Environment (RTE)**: Hệ thống quản lý trung tâm, giúp các ứng dụng và dịch vụ giao tiếp với nhau mà không cần biết chi tiết phần cứng.

Tương tác giữa các lớp trong AUTOSAR

AUTOSAR đảm bảo rằng các lớp khác nhau có thể giao tiếp thông qua các giao diện chuẩn hóa. Điều này giúp cho việc phát triển và bảo trì phần mềm trở nên dễ dàng hơn. Ví dụ, trong hệ thống đèn giao thông, lớp Application chỉ cần gọi các hàm từ lớp RTE mà không cần biết chi tiết cách các chân GPIO hoạt động.

Tính linh hoạt và mở rộng của AUTOSAR

Một trong những lợi ích lớn nhất của AUTOSAR là tính linh hoạt và khả năng mở rộng. Ví dụ, một module phần mềm có thể dễ dàng được cấu hình lại để hoạt động với các loại xe khác nhau mà không cần viết lại mã từ đầu.

3.GIỚI THIỆU TỔNG QUAN VỀ BASIC SOFTWARE BSW

Chào bạn! Hôm nay, chúng ta sẽ đi chi tiết hơn về cấu trúc và hoạt động của các lớp (layers) trong một hệ thống phần mềm, đặc biệt là liên quan đến việc điều khiển thiết bị và quản lý hệ thống. Chúng ta sẽ phân tích từng phần một cách cụ thể và dễ hiểu nhé.

1. Tổng Quan về Các Lớp trong Hệ Thống Phần Mềm

Trong một hệ thống phần mềm, đặc biệt là hệ thống điều khiển thiết bị như ô tô, chúng ta có thể chia ra nhiều lớp khác nhau. Mỗi lớp có một nhiệm vụ riêng và tương tác với các lớp khác để tạo ra một hệ thống hoàn chỉnh.

Các Lớp Chính Gồm Có:

- **Hardware Layer (Lớp Phần Cứng)**: Đây là lớp chứa các thiết bị vật lý như cảm biến, bộ điều khiển, và các phần cứng khác.
- **Driver Layer (Lớp Điều Khiển Thiết Bị)**: Các trình điều khiển giúp phần mềm giao tiếp với phần cứng.
- **Middleware Layer (Lớp Trung Gian)**: Các dịch vụ và thư viện hỗ trợ các chức năng chung cho hệ thống.
- **Application Layer (Lớp Ứng Dụng)**: Các ứng dụng cụ thể mà người dùng tương tác trực tiếp.

2. Chi Tiết Về Từng Lớp

2.1. Lớp Hardware (Phần Cứng)

Tưởng tượng rằng lớp phần cứng như là cơ thể của một con người, chứa các bộ phận như tay, chân, mắt, và tai. Những bộ phận này thực hiện các chức năng cơ bản như nhìn, nghe, cầm nắm.

2.2. Lớp Driver (Điều Khiển Thiết Bị)

Lớp này giống như hệ thần kinh trong cơ thể, giúp truyền tín hiệu từ não (phần mềm) tới các bộ phận cơ thể (phần cứng). Trình điều khiển (driver) là các phần mềm đặc biệt giúp giao tiếp giữa phần mềm và phần cứng. Ví dụ, khi bạn nhấn nút trên bàn phím, driver sẽ chuyển tín hiệu đó đến hệ điều hành để nó biết bạn đã nhấn nút gì.

2.3. Lớp Middleware (Trung Gian)

Lớp này có thể ví như hệ tiêu hóa và tuần hoàn, giúp phân phối và xử lý các chất dinh dưỡng, năng lượng trong cơ thể. Trong hệ thống phần mềm, middleware cung cấp các dịch vụ như quản lý bộ nhớ, bảo mật, và giao tiếp giữa các module khác nhau. Ví dụ, một middleware có thể giúp quản lý kết nối mạng, đảm bảo dữ liệu được truyền tải an toàn và hiệu quả.

2.4. Lớp Application (Ứng Dụng)

Đây là lớp cuối cùng mà người dùng tương tác trực tiếp, giống như là các hành động mà bạn thực hiện hàng ngày như đi bộ, ăn uống, nói chuyện. Các ứng dụng cụ thể như ứng dụng điều khiển xe ô tô, ứng dụng di động, v.v. được phát triển trên lớp này.

3. Ví Dụ Minh Họa

Để làm rõ hơn, hãy tưởng tượng về một hệ thống quản lý xe ô tô:

- **Hardware Layer**: Cảm biến tốc độ, cảm biến nhiệt độ, bộ điều khiển động cơ.
- **Driver Layer**: Các driver cảm biến tốc độ và nhiệt độ, driver điều khiển động cơ.
- **Middleware Layer**: Quản lý dữ liệu từ các cảm biến, bảo mật dữ liệu, quản lý giao tiếp giữa các module.
- **Application Layer**: Ứng dụng điều khiển xe, hiển thị thông tin tốc độ và nhiệt độ lên bảng điều khiển.

4. Những Khái Niệm Quan Trọng

4.1. Interface (Giao Diện)

Giao diện trong hệ thống phần mềm giống như các cổng giao tiếp trong cơ thể. Ví dụ, miệng là giao diện để ăn uống, tai là giao diện để nghe. Trong phần mềm, giao diện (interface) giúp các module giao tiếp với nhau mà không cần biết chi tiết về cách hoạt động bên trong của nhau.

4.2. API (Application Programming Interface)

API là một tập hợp các hàm, giao thức, và công cụ giúp các ứng dụng giao tiếp với nhau. Tưởng tượng rằng API giống như là một quyển hướng dẫn sử dụng giúp bạn biết cách sử dụng một thiết bị nào đó.

4.TỔNG HỢP KIẾN THỨC VỀ CÁC KHÁI NIỆM TRONG HỆ THỐNG TRUYỀN DỮ LIỆU

1. Mô hình OSI (Open Systems Interconnection):

Mô hình OSI là một khung lý thuyết được chuẩn hóa bởi ISO (Tổ chức Tiêu chuẩn hóa Quốc tế), được thiết kế để hướng dẫn và cải thiện khả năng tương tác giữa các hệ thống mạng khác nhau. Mô hình này chia quá trình truyền dữ liệu thành 7 lớp, mỗi lớp đảm nhận một chức năng riêng biệt.

- **Lớp Vật Lý (Physical Layer):** Xử lý truyền tải bit đơn qua phương tiện truyền dẫn vật lý như cáp đồng, sợi quang, hoặc sóng vô tuyến.
    - **Ví dụ:** Truyền tín hiệu điện qua cáp mạng Ethernet.
- **Lớp Liên Kết Dữ Liệu (Data Link Layer):** Đảm bảo rằng dữ liệu được truyền chính xác qua một liên kết vật lý. Chức năng chính bao gồm phát hiện và sửa lỗi.
    - **Ví dụ:** Giao thức Ethernet tại lớp Liên Kết Dữ Liệu đảm bảo dữ liệu được truyền và nhận chính xác qua cáp mạng.
- **Lớp Mạng (Network Layer):** Quản lý định tuyến và chuyển tiếp các gói dữ liệu từ nguồn đến đích qua các mạng trung gian.
    - **Ví dụ:** Giao thức IP (Internet Protocol) quyết định đường đi tốt nhất để gói dữ liệu tới đích.
- **Lớp Giao Vận (Transport Layer):** Quản lý việc chia nhỏ dữ liệu thành các đoạn (segment), kiểm soát lưu lượng và đảm bảo tính toàn vẹn của dữ liệu.
    - **Ví dụ:** Giao thức TCP (Transmission Control Protocol) chia nhỏ dữ liệu thành các đoạn nhỏ và đảm bảo tất cả các đoạn đến đích đầy đủ và đúng thứ tự.
- **Lớp Phiên (Session Layer):** Quản lý các phiên giao tiếp và đảm bảo rằng các phiên này được thiết lập, duy trì và kết thúc một cách hợp lý.
    - **Ví dụ:** Trong một cuộc gọi video, lớp Phiên đảm bảo rằng phiên giao tiếp giữa hai người dùng được duy trì ổn định.
- **Lớp Trình Bày (Presentation Layer):** Chuyển đổi dữ liệu giữa định dạng được sử dụng bởi ứng dụng và định dạng được sử dụng để truyền tải. Đây là lớp mà mã hóa và nén dữ liệu diễn ra.
    - **Ví dụ:** Chuyển đổi dữ liệu từ định dạng văn bản ASCII sang định dạng nhị phân để truyền tải.
- **Lớp Ứng Dụng (Application Layer):** Cung cấp giao diện trực tiếp với người dùng và các ứng dụng mạng.
    - **Ví dụ:** Giao thức HTTP (HyperText Transfer Protocol) cho phép trình duyệt web tải trang web từ máy chủ.

2. Khái niệm SDU và PDU:

- **SDU (Service Data Unit):** Đây là đơn vị dữ liệu mà một lớp nhận từ lớp trên. SDU là dữ liệu gốc trước khi được xử lý bởi lớp hiện tại.
    - **Ví dụ:** Khi một ứng dụng gửi dữ liệu xuống lớp giao vận, dữ liệu đó là SDU của lớp giao vận.

- **PDU (Protocol Data Unit):** Đây là đơn vị dữ liệu được truyền qua mạng bởi một giao thức cụ thể. PDU bao gồm SDU và PCI.
    - **Ví dụ:** Tại lớp mạng, khi một gói dữ liệu IP (gồm cả phần đầu và phần dữ liệu) được tạo ra, đó là PDU của lớp mạng.

3. PCI (Protocol Control Information):

- **PCI:** Là thông tin điều khiển thêm vào SDU để tạo thành PDU. PCI chứa các thông tin như địa chỉ nguồn và đích, số thứ tự, và các tham số kiểm soát khác.
    - **Ví dụ:** Một gói TCP bao gồm PCI như số thứ tự gói, số ACK, và các cờ điều khiển để đảm bảo truyền dữ liệu chính xác.

4. Fragmentation (Phân mảnh dữ liệu):

- **Fragmentation:** Khi dữ liệu quá lớn để truyền trong một lần, nó được chia nhỏ thành nhiều mảnh nhỏ hơn (segments hoặc fragments). Mỗi mảnh sẽ có PCI riêng và được truyền độc lập.
    - **Ví dụ:** Một tệp lớn gửi qua mạng có thể bị phân mảnh thành nhiều đoạn nhỏ, mỗi đoạn có thể được gửi qua các đường truyền khác nhau và sau đó tái hợp tại đích.

5.INTERFACES TRONG AUTOSAR

Trong phần này, chúng ta sẽ tìm hiểu chi tiết về các loại interface khác nhau được sử dụng trong hệ thống truyền dữ liệu. Các loại interface này bao gồm Autosar Interface, Standard Interface và một số khái niệm khác liên quan.

1. Autosar Interface (Autosar Interface ở mức cao nhất):

- **Autosar Interface** là loại giao diện ở mức cao nhất, được thiết kế để giao tiếp giữa các module phần mềm trong hệ thống Autosar (Automotive Open System Architecture).
- **Đặc điểm:**
  - Không phụ thuộc vào một phần cứng cụ thể.
  - Được sinh ra trong quá trình chạy công cụ để tạo ra các module phần mềm.
  - Độc lập giữa các ECU (Electronic Control Unit), nghĩa là có thể thiết kế giao diện mà không quan tâm đến nơi mà giao diện đó sẽ được sử dụng.

Ví dụ: Autosar Interface có thể được sử dụng để giao tiếp giữa các hệ thống điều khiển động cơ và hệ thống phanh trong một chiếc xe ô tô.

2. Standard Interface (Chuẩn hóa bởi Autosar):

- **Standard Interface** là các hàm API được chuẩn hóa bởi Autosar. Các API này có cú pháp, loại dữ liệu trả về và các tham số đầu vào đã được quy định sẵn.
- **Đặc điểm:**
  - Được sử dụng trực tiếp trong các module phần mềm của hệ thống Autosar.
  - Không cần phải thông qua một lớp trung gian nào.
  - Có thể được gọi bởi các phần mềm ứng dụng để thực hiện các chức năng cụ thể.

Ví dụ: Một hàm API như `Com_SendSignal()` trong Autosar được sử dụng để cập nhật giá trị của một tín hiệu trong hệ thống.

3. Intermediate Interface (Các interface trung gian):

- **Intermediate Interface** là loại giao diện nằm giữa Autosar Interface và Standard Interface, cung cấp một sự kết hợp giữa cả hai loại.
- **Đặc điểm:**
  - Sử dụng để cung cấp các API cho các ứng dụng cụ thể trong hệ thống.
  - Kết hợp tính linh hoạt của Autosar Interface và tính chuẩn hóa của Standard Interface.

Ví dụ: Một API trong hệ thống quản lý sự kiện (Event Manager) có thể được sử dụng để kiểm tra trạng thái của một mạng con trong hệ thống.

Tổng Quát và Ví Dụ Minh Họa

Ví dụ Cụ Thể về Giao Diện và API trong Hệ Thống Autosar:

1. **Autosar Interface:**
    - **Khái niệm:** Được sử dụng để giao tiếp giữa các module phần mềm, không phụ thuộc vào phần cứng cụ thể.
    - **Ví dụ:** Giao diện giữa hệ thống điều khiển động cơ và hệ thống phanh.
2. **Standard Interface:**
    - **Khái niệm:** Là các hàm API được chuẩn hóa, có thể gọi trực tiếp bởi các module phần mềm.
    - **Ví dụ:** Hàm `Com_SendSignal()` dùng để gửi tín hiệu trong hệ thống.
3. **Intermediate Interface:**
    - **Khái niệm:** Kết hợp giữa Autosar Interface và Standard Interface, cung cấp API cho các ứng dụng cụ thể.
    - **Ví dụ:** API trong hệ thống quản lý sự kiện để kiểm tra trạng thái của một mạng con.

Sơ Đồ Tổng Quát

Một sơ đồ tổng quát của các loại interface có thể trông như sau:

- **Autosar Interface (màu xanh):** Giao tiếp với nhau thông qua cơ chế không phụ thuộc vào phần cứng cụ thể.
- **Standard Interface (màu vàng):** Giao tiếp trực tiếp với các module phần mềm.
- **Intermediate Interface (màu cam):** Kết hợp cả hai loại trên và cung cấp API cho các ứng dụng cụ thể.

6.VIRTUAL FUNCTIONAL BUS

1. Virtual Functional Bus (VFB)

Khái niệm:

- **VFB (Virtual Functional Bus)** là một khái niệm quan trọng trong hệ thống Autosar, cho phép tách biệt thiết kế phần mềm và phần cứng. Nó cung cấp một cơ chế ảo để các module phần mềm giao tiếp với nhau mà không cần biết đến phần cứng thực tế hay các chi tiết triển khai cụ thể.

Đặc điểm:

- **Độc lập với phần cứng:** VFB cho phép các module phần mềm giao tiếp với nhau mà không cần biết đến vị trí vật lý hoặc phần cứng cụ thể.
- **Thiết kế linh hoạt:** Các nhà phát triển có thể tập trung vào việc thiết kế và phát triển chức năng mà không cần quan tâm đến việc triển khai phần cứng.

Ví dụ:

- Một hệ thống điều khiển động cơ có thể gửi dữ liệu đến hệ thống phanh thông qua VFB mà không cần biết đến cách kết nối vật lý giữa các ECU (Electronic Control Units).

2. Autosar Layered Architecture (Kiến trúc lớp của Autosar)

Khái niệm:

- Kiến trúc lớp của Autosar được thiết kế để tách biệt các chức năng khác nhau trong hệ thống, giúp dễ dàng phát triển, bảo trì và mở rộng hệ thống.

Các lớp chính:

- **Application Layer:** Chứa các ứng dụng phần mềm cụ thể.
- **Runtime Environment (RTE):** Cung cấp cơ chế giao tiếp giữa các module phần mềm trong Application Layer và Basic Software Layer.
- **Basic Software Layer:** Chứa các module phần mềm cơ bản cung cấp các dịch vụ chung như giao tiếp, quản lý bộ nhớ và quản lý thời gian thực.

Ví dụ:

- **Application Layer:** Chứa các ứng dụng như hệ thống điều khiển động cơ, hệ thống phanh.
- **Basic Software Layer:** Chứa các dịch vụ chung như quản lý bộ nhớ, giao tiếp CAN (Controller Area Network).

3. Types of Interfaces (Các loại giao diện)

Autosar Interfaces:

- **Application Interface (API):** Cung cấp các hàm chuẩn hóa để các ứng dụng phần mềm giao tiếp với nhau.
- **Standard Interface:** Được chuẩn hóa bởi Autosar, cho phép các module phần mềm giao tiếp mà không phụ thuộc vào phần cứng cụ thể.

Ví dụ:

- **API:** Các hàm như Com_SendSignal() để gửi tín hiệu trong hệ thống.
- **Standard Interface:** Cho phép giao tiếp giữa các module phần mềm trong hệ thống.