**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);

}

}

1. **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.