# AUTOSAR

## AUTOSAR

### AUTOSAR là gì?

***AUTOSAR*** *- viết tắt của* ***AUT****omotive* ***O****pen* ***S****ystem* ***AR****chitecture*, là một "tiêu chuẩn" dùng để thiết kế ECU trong ngành công nghiệp Automotive được quy định bởi một nhóm các công ty: BMW Group, BOSCH, Continental, Daimler, Ford, General Motors, PSA Group, Toyota, VOLKSWAGEN.

Trên đây là những thành viên chính (Core Members) của mối quan hệ đối tác AUTOSAR, những công ty/tập đoàn đã sáng lập ra chuẩn AUTOSAR.

**AUTOSAR** được triển khai theo mô hình phân lớp, giống như mô hình OSI. Các lớp trong mô hình này dùng để xử lý và trừu tượng hóa các hoạt động khác nhau của code. AUTOSAR được sử dụng cho các vi điều khiển với mục đích sử dụng cho các ứng dụng automotive - CAN, FlexRay, Ethernet, ...

### Lí do cần có AUTOSAR?

Thực tế thì làm gì cũng cần có tiêu chuẩn rõ ràng. Đối với việc viết phần mềm cho ECU, các developer có thể đối mặt với nhiều vấn đề khác nhau:

* Hệ thống nhúng là một lĩnh vực rất lớn và có rất nhiều nhà sản xuất chất bản dẫn khác nhau. Nền tảng phần cứng và phần mềm có thể được lựa chọn tùy thuộc vào từng yêu cầu ứng dụng khác nhau. Chính việc này có thể dẫn đến khó khăn trong việc phát triển cũng như tính di động của phần mềm. Chẳng hạn như việc phần cứng thay đổi, gần như toàn bộ phần mềm sẽ phải thay đổi theo, làm tăng chi phí và thời gian phát triển phần mềm.
* Automotive là một hệ thống phức tạp bao gồm nhiều hệ thống nhỏ hơn gọi là **ECUs - Electronic Control Unit**, nên việc bảo trì và phát triển phần mềm cho từng ECU là việc không dễ. Độ phức tạp càng tăng thêm khi các ECU khác nhau sử dụng các MCU khác nhau để đáp ứng các yêu cầu về mặt ứng dụng và chi phí. Vì vậy, có thể chúng ta sẽ cần phát triển rất nhiều phần mềm khác nhau cho các MCU này.
* Với cách viết phần mềm thông thường, để giao tiếp giữa nhiều ECU khác nhau, nhà phát triển sẽ cần tạo ra các giao thức, tiêu chuẩn giao tiếp, gọi là **Custom standard**. Việc này rất tốt, nhưng lần sau làm việc với chiếc xe khác, các ECU khác thì lại rất khó để maintain và tốn nhiều chi phí hơn.

AUTOSAR được thiết kế theo kiến trúc phân lớp, và tầng ứng dụng sẽ được viết độc lập với phần cứng, để cùng một code ứng dụng có thể chạy trên các nền tảng phần cứng khác nhau. AUTOSAR cung cấp một layer riêng để giao tiếp với phần cứng gọi là **MCAL** (Micro Controller Abstraction Layer).

### Đây có phải là một hạn chế của AUTOSAR

Trên thực tế đây không phải là một hạn chế, trong thế giới thay đổi nhanh chóng hiện nay, việc đáp ứng hoàn thành các dự án đúng deadline ngày càng nghiêm ngặt. Nên khi có AUTOSAR, việc đáp ứng nhu cầu phát triển phần mềm sẽ dễ dàng hơn. Mặc dùng AUTOSAR triển khai theo kiến trúc phân lớp và mọi thứ trông có vẻ có sẵn, nhưng chúng ta vẫn cần phải triển khai viết code cho các chức năng của SWC trong một **Runnable** của SWC.

### Nếu muốn sử dụng các thiết bị ngoài không hỗ AUTOSAR thì sao?

Nếu sử dụng các thiết bị không được AUTOSAR hỗ trợ, chúng ta có thể sử dụng tầng **CDD - Complex Device Drivers**. Tầng này cho phép truy cập trực tiếp vào lớp MCAL từ tầng Application và chúng ta có thể giao tiếp trực tiếp giữa thiết bị ngoài với ECU. Chúng ta có thể tự phát triển phần mềm cho các thiết bị này, tuy nhiên nó phụ thuộc vào phần cứng, nên không có khả năng tái sử dụng như SWC.

### Các phiên bản AUTOSAR

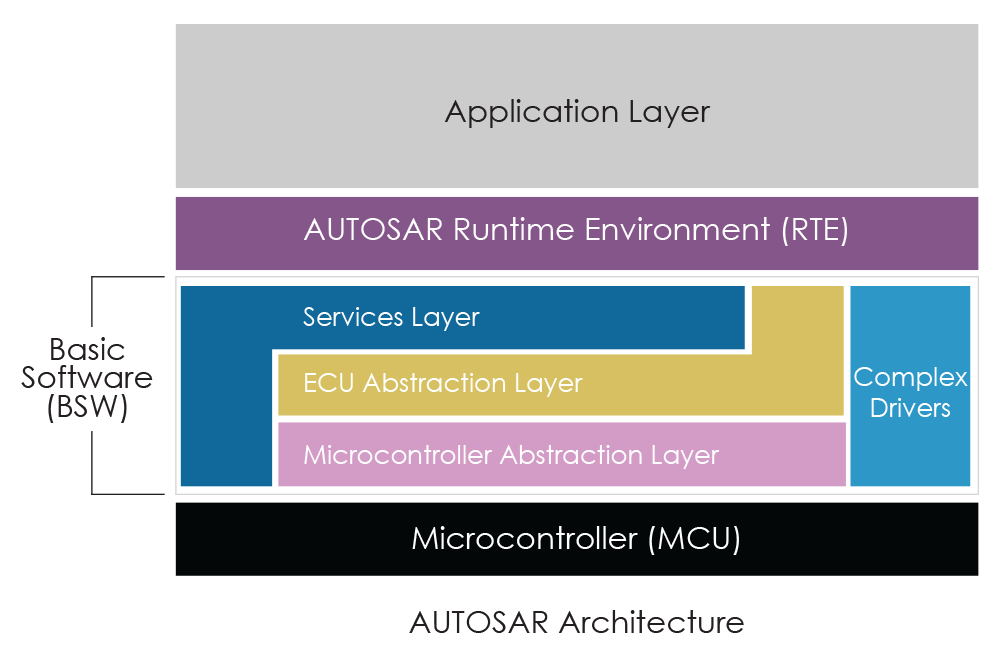
Có 2 loại kiến trúc AUTOSAR là: **Classic** và **Adaptive**. Bản Classic có tất cả các module cơ bản cần có cho các ứng dụng, trong khi bản Adaptive có thể config được và điều chỉnh theo các ứng dụng bằng cách loại bỏ các module không cần thiết.

Bản Classic hiện tại đang là **4.7.0**, còn bản Adaptive hiện tại là **21.11** **⇨** Phiên bản này chỉ đúng tại thời điểm viết bài và có thể không còn đúng vào thời điểm hiện tại.

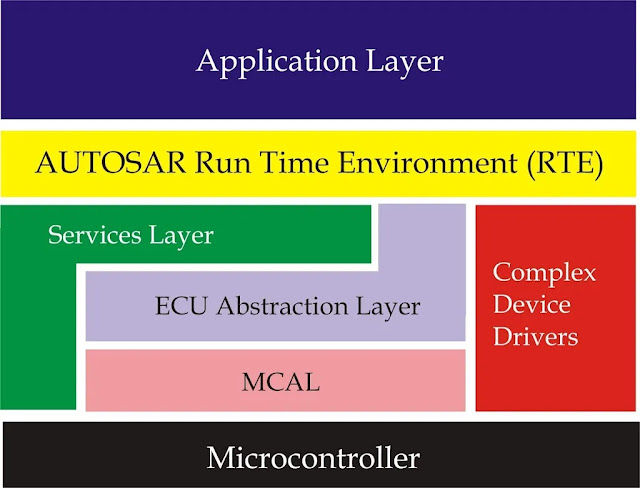
## AUTOSAR LAYER

### Cấu tạo của tiêu chuẩn AUTOSAR?

Các tiêu chuẩn của AUTTOSAR chia phần mềm ra là 5 lớp.



#### Application Layer



**Hình 1**: Kiến trúc phân lớp đơn giản nhất theo Autosar.

* **Application Layer:** Tầng này chứa code ứng dụng ở tầng cao nhất. Nó có thể bao gồm các khối ứng dụng khác nhau gọi là [**SWCs - Software Components**](https://www.laptrinhdientu.com/2023/06/autosar-software-component.html)cho từng tính năng mà ECU cần hỗ trợ tùy theo từng ứng dụng.

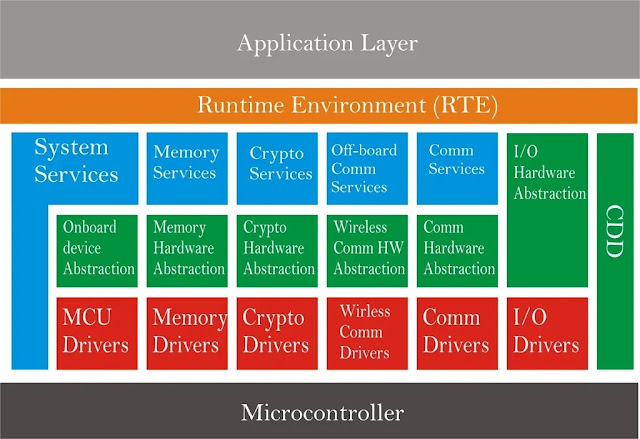
Ví dụ: Các chức năng như đèn báo trên tablo, hệ thống phanh, khí thải, điều hòa, đồng hồ đo nhiên liệu, vòng tua chính là các SWC riêng biệt.

* [**AUTOSAR RTE**](https://www.laptrinhdientu.com/2023/08/autosar-run-time-environment-rte.html)**:** Cung cấp khả năng giao tiếp giữa các SWC khác nhau và cả giữa các ECU khác nhau. Application layer sử dụng layer này trong khi giao tiếp với các layer bên dưới bằng các **ports**.

#### Basic Software là gì?

***BSW - Basic Software*** bao gồm các module Basic Software (BSWM) dưới dạng tập hợp các file phần mềm (code hoặc description), những file code này triển khai các hàm cơ bản của một ECU.

BSW được cấu hình bằng các phần mềm cấu hình như ***Vector Davinci Configurator*** là các cấu hình nằm dưới tầng RTE.



* **Services Layer:** Tầng này cung cấp các dịch vụ khác nhau cho các application sử dụng. Các dịch vụ như: System Services, Memory Services, Crypto Services, Off board communication Services, Communication Services.
* **ECU Abstraction Layer:** Tầng này cung cấp các tính năng liên quan đến abstraction (Trừu tượng hóa). Nó bao gồm các tầng abstraction như: I/O Hardware Abstraction layer, On board device abstraction, Memory hardware Abstraction, Crypto hardware abstraction, ... Mục đích của tầng này nhằm độc lập phần mềm ứng dụng với phần cứng nhằm để ReUser - Tức là code của tầng ứng dụng sẽ không cần thay đổi khi thay đổi phần cứng.
* **MCAL - Microcontroller Abstraction Layer**  
  Các khối màu đỏ đại diện cho tầng MCAL (Microcontroller Abstraction Layer), bao gồm các drivers của các peripherals. MCAL là *tầng thấp nhất của Basic Software*. Tầng này trực tiếp truy cập vào phần cứng nên rất phụ thuộc vào phần cứng, và sẽ cần thay đổi đối với các phần cứng khác nhau.
* **CDD - Complex Device Drivers**  
  Tầng CDD kết nối trực tiếp SWC - tầng Application với phần cứng MCU thông qua tầng RTE. Tầng này hữu ích cho việc viết các *functions/drivers* của các *peripheral/external devices* mà không được định nghĩa trong AUTOSAR, hoặc những functions yêu cầu ràng buộc cao về mặt thời gian.  
  Khác với tầng MCAL kết hợp với ECUAL, tầng CDD phụ thuộc nhiều vào phần cứng và khả năng sử dụng lại code sẽ kém hơn.

### Internal/External Driver trong AUTOSAR

Phụ thuộc vào loại peripheral được sử dụng, AUTOSAR có 2 loại driver: **Internal** và **External**.

Internal Driver được sử dụng để truy cập tới các thiết bị ngoại vi nội bên trong vi điều khiển như EEPROM, ADC, GPIO, ... Trong khi đó External Driver sử dụng để truy cập tới các thiết bị ngoại vi kết nối bên ngoài Vi điều khiển, như External Flash, SD Card, ...

Các Internal Driver nằm trong tầng MCAL, trong khi External Driver nằm trong tầng ECU Abstraction Layer. Một vài trường hợp ngoại lệ, với External Device như memory mapped memories, nó có thể được truy cập trực tiếp bằng vi điều khiển và nó nằm trong tầng MCAL.

### Khái niệm Interfaces, Handlers, Manager

➤ **Interfaces** là khái niệm nằm trong tầng **E*CU Abstraction***, cung cấp một chức năng để trừu tượng hóa các module cấp thấp (Low-level module) và cung cấp một số API có thể được sử dụng ở các lớp trên.

***Interfaces*** còn cung cấp khả năng truy cập vào một số thiết bị cụ thể bất kể số lượng thiết bị hiện có cùng loại, và không phụ thuộc vào Hardware.

➤ **Handlers** có nhiệm vụ kiểm soát các truy cập đồng thời, các truy cập liên tiếp và bất đồng bộ của một hay nhiều clients, Khi có nhiều truy cập xảy ra thì Handlers sẽ đưa chúng vào các cơ chế như Buffer, [**Queue**](http://www.laptrinhdientu.com/2022/05/Queue.html), ... để tránh mất sự kiện.

Các chức năng của ***Handlers*** thường được triển khai trong Driver hoặc Interfaces.

➤ **Manager** là khái niệm nằm trong tầng ***Services***, nó cung cung các dịch vụ cụ thể cho nhiều clients khác nhau. Nó được yêu cầu trong mọi trường hợp mà chỉ mình ***Handlers*** là không đủ để xử lý các yêu cầu.

Ví dụ. ***NVRAM Manager*** điều khiển việc truy cập vào bộ nhớ [**Internal/External Flash**](http://www.laptrinhdientu.com/2022/01/EmbeddedMemory.html).

### Chi tiết các tầng của BSW

#### MCAL (Microcontroller Abstraction Layer)

Tầng MCAL bao gồm những Layer sau:

* ***Microcontroller Drivers*** - Cung cấp các function để truy cập đến các ngoại vi nội như Watchdog, Timer, ...
* ***Memory Drivers*** - Cung cấp các function để truy cập đến các bộ nhớ nội như Internal Flash, Internal EEPROM, và các bộ nhớ ngoài như External Flash.
* ***Crypto Drivers*** - Cung cấp các function để truy cập đến Internal Crypto như SHE, HSM, ...
* ***Wireless Communication Drivers*** - Cung cấp các function cho các hệ thống mạng không dây (Giao tiếp trong hoặc ngoài xe).
* ***Communication Drivers*** - Cung cấp các function cho các giao tiếp on board như [**UART**](http://www.laptrinhdientu.com/2021/09/STM11.html), [**I2C**](http://www.laptrinhdientu.com/2021/09/8051Bai21.html), [**SPI**](http://www.laptrinhdientu.com/2021/08/8051Bai19.html), ... và các giao tiếp Vehicle như CAN, LIN, FlexRay, ...
* ***I/O Drivers*** - Cung cấp các function để truy cập và sử dụng các chân I/O của MCU như chân Digital, Analog, PWM, ...

#### CDD - Complex Device Drivers

Module này thích hợp trong việc triển khai các chức năng ***non-standard*** (không theo tiêu chuẩn của AUTOSAR) trong BSW stack. Trên thực tế AUTOSAR là một tiêu chuẩn chung cho phần mềm phục vụ cho lĩnh vực Automotive, nên nó không thể cover hết được những trường hợp riêng biệt của từng hãng làm phần mềm.

Chính vì vậy có nhiều trường hợp mà các hãng sẽ triển khai một số chức năng mà AUTOSAR không hỗ trợ, CDD sẽ được sử dụng trong những trường hợp như vậy.

Một trường hợp khác, một số chức năng yêu cầu ràng buộc về mặt thời gian, và yêu cầu thời gian thấp hơn thời gian tối thiểu của AUTOSAR OS cũng cần đặt vào tầng CDD, vì CDD giúp kết nối MCU với ứng dụng.

➤ Nhược điểm lớn nhất của tầng CDD là không theo tiêu chuẩn, và phụ thuộc vào application nhiều hơn là MCU, vì vậy code ở tầng này sẽ ***khó porting*** giữa các dòng chip khác nhau.

#### ECU Abstraction Layer

Tầng Abstraction giúp việc viết phần mềm cho ECU sẽ độc lập với phần cứng (Cả về các thành phần MCU cũng như các thiết bị ngoại vi bên ngoài kết nối với ECU). Tức là Application sẽ được Abstraction cung cấp các API để sử dụng và không cần quan tâm đến sự thay đổi của phần cứng (ECU hay MCU).

* ***I/O Hardware Abstraction*** - Các thiết bị ngoại vi I/O có thể nằm trên chip hoặc trên board, việc cấu hình các chân của MCU cũng khá phức tạp. Tuy nhiên tầng Abstraction giúp đơn giản hóa việc cấu hình của tầng Application, bằng cách cung cấp các API để sử dụng I/O, Ví dụ - [***Pin\_SetMode***](http://www.laptrinhdientu.com/2021/09/STM6.html)***(pin, mode)***.   
  Tầng này sẽ cung cấp các API không đổi như trên, nhưng nội dung của nó sẽ thay đổi theo phần cứng của ECU (Ví dụ các MCU khác nhau thì sẽ có số lượng chân khác nhau, tên các module quản lý chân khác nhau như GPIO, PORT, DIO, ...).
* ***Communication Hardware Abstraction*** - Đối với các module truyền thông (như CAN, LIN, FlexRay, ...), tầng Application chỉ cần quan tâm việc truyền nhận dữ liệu, không cần quan tâm chân nào được sử dụng để kết nối, kết nối trên chip hay trên board, thậm chí không cần quan tâm loại bus giao tiếp nào được sử dụng.   
  Module này đảm bảo việc độc lập phần cứng như nói trên, cung cấp các API cho việc truyền/nhận dữ liệu.
* ***Crypto Hardware Abstraction*** - Module này trừu tượng hóa chức năng Crypto bằng cách ẩn các thông tin về Crypto được sử dụng (Các thiết bị Internal/External hoặc các phần mềm cơ sở). Bởi vì Application không cần quan tâm đến việc loại Crypto nào được sử dụng, trên chip, board hay mã hóa bằng phần mềm.
* ***Memory Hardware Abstraction*** - Module này trừu tượng hóa vị trí của thiết bị bộ nhớ được sử dụng. Application chỉ cần quan tâm đến dữ liệu, không có quyền kiểm soát đối với thiết bị bộ nhớ được chọn là Internal/External, ROM/Flash/SDCard, ...
* ***Onboard Device Abstraction*** - Module này trừu tượng hóa các thiết bị trên board mạch cụ thể, cung cấp các API để giao tiếp với các thiết bị trên Board như [**Sensors**](http://www.laptrinhdientu.com/2022/06/Sensor-Actuator.html), [**Actuators**](http://www.laptrinhdientu.com/2022/06/Sensor-Actuator.html).

#### Services Layer

* ***Communication Services*** - Đây là một nhóm các module dành cho các giao thức truyền thông mạng trên xe, cung cấp giao diện thống nhất cho mạng trên xe để giao tiếp dữ liệu, đồng thời ẩn các thuộc tính giao thức và thuộc tính message khỏi tầng Application.  
  Communication Service Interfaces với Communication Drivers (MCAL), cùng với sự giúp đỡ của Communication Hardware Abstraction. Đây là sự độc lập phần cứng với ECU và MCU, nhưng phụ thuộc vào bus type. Vì vậy, một phần của tầng Service có thể thay đổi nếu bus type thay đổi (CAN, FlexRay, ...).
* ***Off board Communication Services*** - Đây là một nhóm các module dành cho phương tiện để giao tiếp với các clients bên ngoài thông qua mạng không dây. Nó bao gồm 3 khối được sử dụng với những chức năng khác nhau. Module này cung cấp một giao diện thống nhất cho mạng Ethernet không dây bằng cách ẩn các thuộc tính giao thức và message.
* ***Memory Services*** - Service này bao gồm một module, *NVRAM Manager*. Nó chịu trách nhiệm quản lý dữ liệu non-volatile (đọc/ghi từ các memory drivers khác nhau). Application yêu cầu lưu trữ dữ liệu trong bộ nhớ để sử dụng sau này, vì vậy module này được sử dụng để triển khai điều này theo một cách thống nhất và cung cấp khả năng trừu tượng hóa từ các vị trí bộ nhớ cấp thấp hơn.  
  Memory Services cung cấp cơ chế để lưu trữ, load, checksum, ... dữ liệu non-volatile, có khả năng cấu hình cao và độc lập với phần cứng ECU.
* ***System Services*** - Đây là một nhóm các module có thể được sử dụng bởi các module của tất cả các layer, ví dụ như RTOS, Error Messager. Các services này có thể phụ thuộc vào một số MCU hoặc có thể hỗ trợ các tính năng đặc biệt của MCU (như Time Service), một phần phụ thuộc vào phần cứng ECU và Application.

Trên đây là các thuật ngữ về BSW, có một số thuật ngữ sẽ được giải thích chi tiết sau này.