BỘ CÔNG THƯƠNG

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH**



**Võ Linh Trúc**

Thiết kế và triển khai công cụ mã nguồn mở RTE generator cho phần mềm nhúng xe ô tô

Chuyên ngành: KỸ THUẬT ĐIỆN TỬ

Mã chuyên ngành: 8520203

**ĐỀ CƯƠNG LUẬN VĂN THẠC SĨ**

thành phố hồ chí minh, NĂM 2023

**THÔNG TIN CHUNG**

Họ và tên học viên : Võ Linh Trúc MSHV: 20000411

Lớp : CHDT10A Khóa: 10

Chuyên ngành : Kỹ thuật điện tử Mã chuyên ngành: 8520203

SĐT :0388488608

Email : trucvo2601@gmail.com

Địa chỉ liên hệ : 512 Trường Chinh, phường 13, quận Tân Bình, TPHCM

Tên đề tài : Thiết kế và triển khai công cụ mã nguồn mở RTE generator cho phần mềm nhúng xe ô tô

Người hướng dẫn : PGS.TS. Nguyễn Ngọc Sơn

SĐT : 0976645726

Email : nguyenngocson@iuh.edu.vn

Cơ quan công tác : Trường Đại học Công nghiệp Tp.HCM

Địa chỉ liên hệ : Phòng X8.1A, 12 Nguyễn Văn Bảo, p4, Gò Vấp, TPHCM

*Tp. Hồ Chí Minh, ngày tháng năm 2023*

|  |  |
| --- | --- |
| **Người hướng dẫn**  *(Ký và ghi rõ họ tên)* | **Học viên**  *(Ký và ghi rõ họ tên)* |

MỤC LỤC

[MỤC LỤC 1](#_Toc128570692)

[DANH MỤC HÌNH ẢNH 2](#_Toc128570693)

[DANH MỤC BẢNG BIỂU 3](#_Toc128570694)

[DANH MỤC TỪ VIẾT TẮT 4](#_Toc128570695)

[MỞ ĐẦU 5](#_Toc128570696)

[CHƯƠNG 1 TỔNG QUAN VỀ LĨNH VỰC NGHIÊN CỨU 9](#_Toc128570697)

[CHƯƠNG 2 CƠ SỞ LÝ THUYẾT 10](#_Toc128570698)

[2.1 Runtime Environment (RTE) và Software Component (SWC) 10](#_Toc128570699)

[2.2 Khái niệm Interface trong AUTOSAR 11](#_Toc128570700)

[2.3 Các thành phần của SWC 11](#_Toc128570701)

[2.3.1 Port 11](#_Toc128570702)

[2.3.2 Runnable 12](#_Toc128570703)

[2.3.3 Event 13](#_Toc128570704)

[2.4 AUTOSAR RTE Generator 15](#_Toc128570705)

[2.5 Hệ điều hành thời gian thực FreeRTOS 15](#_Toc128570706)

[2.6 Các khái niệm trong FreeRTOS: 16](#_Toc128570707)

[2.6.1 Task 16](#_Toc128570708)

[2.6.2 Cấu trúc dữ liệu 18](#_Toc128570709)

[2.6.3 Scheduler 19](#_Toc128570710)

[CHƯƠNG 3 ÁP DỤNG KẾT QUẢ NGHIÊN CỨU 21](#_Toc128570711)

[3.1 Kết quả nghiên cứu ở phạm vi bài nghiên cứu 21](#_Toc128570712)

[3.2 Kết quả nghiên cứu ở phạm vi rộng 21](#_Toc128570713)

[KẾT LUẬN VÀ KIẾN NGHỊ 22](#_Toc128570714)

[TÀI LIỆU THAM KHẢO 23](#_Toc128570715)

DANH MỤC HÌNH ẢNH

[Hình 2.1 RTE và các SWC 10](#_Toc128570659)

[Hình 2.2 Ký hiệu các loại Interface 11](#_Toc128570660)

[Hình 2.3 Ví dụ về SWC 15](#_Toc128570661)

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

**AUTOSAR** Automotive Open System Architecture

**arxml** AUTOSAR XML file

**ECU** Electronic Control Unit

**FreeRTOS** Free Real-Time Operating System

**RTE** Run-Time Environment

**SWC** Software component

MỞ ĐẦU

**1. Đặt vấn đề**

Trong những năm gần đây, công nghệ xe hơi đã có sự phát triển mạnh mẽ với sự ra đời của nhiều công nghệ mới như xe tự lái, kết nối thông minh và đặc biệt là hệ thống điện tử nâng cao. Cùng với sự phát triển này, nhu cầu để xây dựng các phần mềm tích hợp cho các hệ thống điện tử trên xe đã trở nên ngày càng quan trọng. Tuy nhiên, việc phát triển phần mềm cho hệ thống điện tử trên xe là một công việc rất phức tạp và tốn nhiều thời gian. Để giải quyết vấn đề này, AUTOSAR (Architecture for Automotive Software) đã được tạo ra để định nghĩa một kiến trúc phần mềm cho hệ thống điện tử trên xe. Nó giúp cho việc phát triển phần mềm cho hệ thống điện tử trở nên dễ dàng hơn và hiệu quả hơn[1].

Tuy nhiên, trong quá trình phát triển phần mềm cho hệ thống điện tử trên xe sử dụng AUTOSAR, việc tạo ra một RTE (Run-Time Environment) để quản lý các sự kiện trong hệ thống trở nên rất quan trọng và tốn nhiều thời gian. Việc sử dụng một AUTOSAR RTE Generator sẽ giúp cho việc tạo ra RTE trở nên dễ dàng hơn và giảm thiểu thời gian và công sức cần thiết.

Tính đến hiện tại, các công ty lớn thường bán AUTOSAR RTE Generator với giá rất cao, dẫn đến khó khăn trong việc tiếp cận với công nghệ này. Điều này gây ra một khó khăn lớn trong việc tạo ra nguồn nhân lực chất lượng trong mảng phần mềm nhúng ô tô, làm chậm quá trình phát triển và cản trở sự tiến bộ của ngành công nghiệp này[2].

Vì vậy, đề tài của chúng tôi hướng đến việc phát triển một AUTOSAR RTE Generator mã nguồn mở, nhằm giúp cho các nhà phát triển, sinh viên và học viên có thể tiếp cận với công nghệ AUTOSAR một cách dễ dàng và thuận tiện. Đồng thời, chúng tôi cũng hy vọng rằng đề tài này sẽ giúp tạo ra một nguồn nhân lực chất lượng, thành thạo trong việc sử dụng AUTOSAR, giúp thúc đẩy sự phát triển của ngành công nghiệp phần mềm nhúng ô tô.

Ví dụ cụ thể, chúng tôi sẽ xây dựng một AUTOSAR RTE Generator cho phép tạo ra các module phần mềm phù hợp với tiêu chuẩn AUTOSAR. Đây là một công cụ hữu ích cho các nhà phát triển phần mềm, giúp cho họ có thể nhanh chóng tạo ra các module phần mềm có tính tái sử dụng cao, giảm thiểu thời gian và chi phí trong việc phát triển sản phẩm.

Trong quá trình xây dựng AUTOSAR RTE Generator, chúng tôi sẽ tập trung vào việc phát triển một công cụ dễ sử dụng, có tính linh hoạt cao và đáp ứng được nhu cầu của các nhà phát triển và học viên. Ngoài ra, chúng tôi cũng sẽ đưa ra các tài liệu và hướng dẫn chi tiết để giúp người sử dụng có thể sử dụng công cụ này một cách hiệu quả và dễ dàng.

**2. Mục tiêu nghiên cứu**

* Xây dựng một công cụ AUTOSAR RTE Generator mã nguồn mở: Mục tiêu chính của đề tài là tạo ra một phần mềm AUTOSAR RTE Generator mã nguồn mở giúp giảm thiểu chi phí cho các công ty trong việc triển khai các dự án phần mềm nhúng ô tô. Điều này sẽ tạo điều kiện thuận lợi cho các công ty startup và các nhà phát triển có ngân sách hạn chế. Ngoài ra, mã nguồn mở còn cho phép các nhà phát triển trên toàn cầu cùng đóng góp phát triển sản phẩm.
* Đề tài tập trung vào các tính năng sau:
  + Sinh ra mã RTE (c file) dựa trên các file đầu vào (arxml file) được tạo bởi AUTOSAR Builder hoặc các công cụ tương tự.
  + Sử dụng FreeRTOS như là framework nền để tổ chức sắp xếp và lập lịch cho các process (tiến trình) được cấu hình trong các file arxml.
  + Đảm bảo tính toàn viện dữ liệu của hệ thống.
  + Hỗ trợ phiên bản AUTOSAR 4.2.2.

**3. Đối tượng và phạm vi nghiên cứu**

* Đối tượng nghiên cứu: phần mềm AUTOSAR RTE Generator mã nguồn mở
* Phạm vi nghiên cứu:
  + Phân tích các yêu cầu và tính năng của RTE thuộc chuẩn AUTOSAR 4.2.2.
  + Thiết kế và phát triển AUTOSAR RTE Generator mã nguồn mở.
  + Kiểm thử và đánh giá hiệu suất của AUTOSAR RTE Generator được phát triển.
  + Sử dụng FreeRTOS để hỗ trợ việc chạy các chương trình được sinh ra.
  + Xây dựng tài liệu hướng dẫn sử dụng và triển khai AUTOSAR RTE Generator mã nguồn mở.

**4. Cách tiếp cận và phương pháp nghiên cứu**

* Cách tiếp cận:
  + Tìm hiểu về embedded system và AUTOSAR, với trọng tâm là phân tích và nghiên cứu RTE của AUTOSAR 4.2.2.
  + Tìm hiểu về FreeRTOS và cách tích hợp nó vào hệ thống để hỗ trợ chạy các chương trình được sinh ra bởi AUTOSAR RTE Generator.
  + Phát triển và đánh giá AUTOSAR RTE Generator, bao gồm kiểm tra tính đúng đắn, hiệu suất và khả năng sử dụng trên nhiều nền tảng nhúng khác nhau.
* Phương pháp nghiên cứu:
  + Phân tích và tổng hợp các tài liệu liên quan đến công nghệ embedded system, AUTOSAR, RTE của AUTOSAR 4.2.2 và FreeRTOS.
  + Thực hiện các thí nghiệm và đánh giá hiệu suất của RTE generator trên các nền tảng nhúng khác nhau.

**5. Ý nghĩa thực tiễn của đề tài**

AUTOSAR RTE Generator sẽ đem lại nhiều ý nghĩa thực tiễn đáng kể trong lĩnh vực phát triển phần mềm nhúng cho xe hơi. Sau đây là một số ý nghĩa thực tiễn của đề tài:

* Tăng tính linh hoạt cho phát triển phần mềm nhúng: việc sử dụng mã nguồn mở và các công cụ phát triển phần mềm như RTE generator và FreeRTOS sẽ giúp cho việc phát triển phần mềm nhúng trở nên linh hoạt hơn. Nhờ có sự hỗ trợ của các công cụ này, các lập trình viên có thể tập trung vào việc triển khai các tính năng cần thiết cho phần mềm nhúng mà không cần phải lo lắng nhiều về các vấn đề bản quyền.
* Giảm thiểu thời gian và chi phí phát triển phần mềm: sử dụng RTE generator và FreeRTOS có thể giảm thiểu thời gian và chi phí phát triển phần mềm nhúng, giúp các doanh nghiệp trong lĩnh vực ô tô và các nhà sản xuất phần mềm nhúng tiết kiệm được thời gian và chi phí trong quá trình phát triển sản phẩm.
* Đóng góp vào quá trình phát triển nguồn nhân lực chất lượng trong lĩnh vực ô tô và xe máy, đặc biệt là chuẩn nguồn nhân lực thành thạo chuẩn AUTOSAR.
* Đưa công nghệ mã nguồn mở vào lĩnh vực phát triển phần mềm nhúng, giúp thúc đẩy sự phát triển và ứng dụng của công nghệ này trong ngành công nghiệp ô tô và xe máy.

# TỔNG QUAN VỀ LĨNH VỰC NGHIÊN CỨU

Lĩnh vực xe hơi, tiêu chuẩn Autosar (Automotive Open System ARchitecture) đã được phát triển bởi các nhà sản xuất xe hơi lớn nhằm giảm thiểu sự phụ thuộc vào nhà sản xuất phần cứng, phần mềm, đồng thời nâng cao tính linh hoạt trong quá trình phát triển phần mềm điều khiển các chức năng trong xe hơi. Autosar đặt ra các tiêu chuẩn, giao thức, quy trình để phát triển các hệ thống điều khiển xe hơi[1].

Hiện nay, Autosar đã trở thành một tiêu chuẩn toàn cầu được nhiều nhà sản xuất ô tô áp dụng vào quá trình phát triển các hệ thống điều khiển trong xe hơi[3]. Tuy nhiên, việc phát triển phần mềm điều khiển hệ thống xe hơi sử dụng Autosar đòi hỏi một số kiến thức chuyên sâu về Autosar và các công cụ hỗ trợ phát triển phần mềm.

Các công cụ hỗ trợ phát triển phần mềm Autosar hiện nay đa số do các công ty lớn sở hữu, giá thành rất cao và không phù hợp cho các đối tượng nghiên cứu nhỏ hoặc các cá nhân mong muốn học tập và phát triển kỹ năng về Autosar[2]. Từ đó dẫn đến thực trạng nguồn nhân lực trong lĩnh vực AUTOSAR đang gặp phải nhiều khó khăn. Các công ty lớn thường sở hữu các công cụ và sản phẩm AUTOSAR, đòi hỏi phải đầu tư lớn và có kiến thức chuyên sâu mới có thể sử dụng và phát triển. Điều này khiến cho ngưỡng vào lĩnh vực này rất cao, đặc biệt là đối với các cá nhân, tổ chức nhỏ và các nhà nghiên cứu mới bắt đầu quan tâm đến AUTOSAR. Tình trạng này khiến cho lượng nhân lực có chuyên môn về AUTOSAR đang hiếm hụt và chất lượng đào tạo và bồi dưỡng cũng còn hạn chế. Do đó, việc tạo ra các công cụ, sản phẩm, hoặc giải pháp mở để hỗ trợ người dùng mới tiếp cận với AUTOSAR và cung cấp cho họ các tài liệu học tập và đào tạo chất lượng là rất cần thiết để thu hút và duy trì sự phát triển của lĩnh vực này.

# CƠ SỞ LÝ THUYẾT

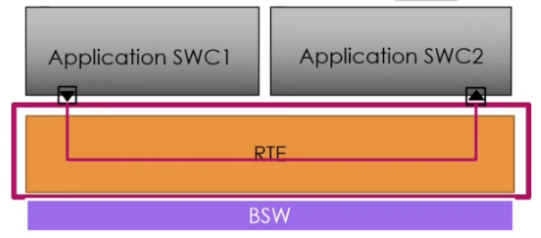
## Runtime Environment (RTE) và Software Component (SWC)

RTE là một phần mềm hỗ trợ việc triển khai ứng dụng trên các hệ thống nhúng. RTE thường được sử dụng trong các hệ thống lớn và phức tạp, nơi các Software Component (SWC) phải tương tác với nhau một cách chặt chẽ để đạt được mục tiêu chung cụ thể như cung cấp các dịch vụ như quản lý bộ nhớ, lập lịch, xử lý sự kiện, quản lý dữ liệu, và truyền thông tin giữa các module[4].

SWC là các thành phần phần mềm độc lập có thể tái sử dụng, được sử dụng để xây dựng các hệ thống phần mềm nhúng. Mỗi SWC đều có các khả năng và chức năng riêng, và được sử dụng để xử lý một phần của chức năng toàn bộ của hệ thống[4].

Trong mô hình phát triển AUTOSAR, RTE là một phần quan trọng của kiến trúc, nó có nhiệm vụ quản lý tất cả các thông điệp và sự kiện được gửi đi và nhận lại giữa các SWC. RTE hoạt động như một lớp trung gian để đảm bảo rằng các SWC có thể tương tác với nhau một cách an toàn và hiệu quả. Các SWC đồng thời cũng cung cấp các thông tin về các khả năng và chức năng của chúng đến RTE, giúp RTE quản lý và định tuyến các thông điệp và sự kiện phù hợp cho từng SWC.

Hình 2.1 mô tả bố trí của RTE và các SWC (Application component và Basic software component)



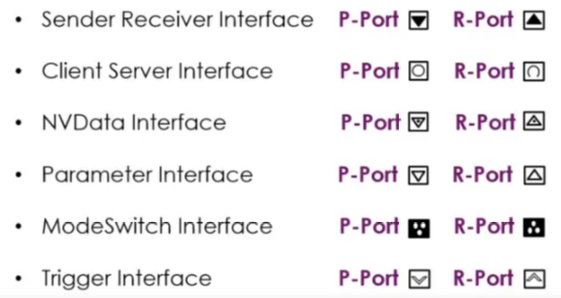
Hình 2.1 RTE và các SWC

## Khái niệm Interface trong AUTOSAR

Trong Autosar, Interface được định nghĩa là một giao diện giữa các module khác nhau trong hệ thống, cung cấp khả năng truyền thông tin giữa các module đó. Interface giúp các module tương tác với nhau một cách chặt chẽ, giảm thiểu sự phụ thuộc giữa chúng[4].

Có nhiều loại interface khác nhau được mô tả như trong Hình 2.2 trong đó có hai loại interface chính:

* Client-Server Interface: Interface này được sử dụng để truyền dữ liệu giữa các module theo kiểu Client-Server. Theo đó, module Client sẽ yêu cầu một dịch vụ từ module Server, và module Server sẽ trả về kết quả cho module Client sau khi xử lý xong yêu cầu đó.
* Sender-Receiver Interface: Interface này được sử dụng để truyền dữ liệu giữa các module theo kiểu Sender-Receiver. Theo đó, module Sender sẽ gửi dữ liệu đến module Receiver, và module Receiver sẽ nhận dữ liệu đó và xử lý theo các hành động cần thiết.



Hình 2.2 Ký hiệu các loại Interface

## Các thành phần của SWC

### Port

Trong Autosar, SWC (Software Component) có thể trao đổi thông tin với nhau thông qua các cổng (port)[4]. Các loại cổng phổ biến của SWC trong Autosar gồm:

* Sender/Provider Port (cổng gửi): SWC gửi dữ liệu đến các SWC khác thông qua sender port. Các SWC khác có thể đăng ký nhận dữ liệu từ cổng gửi này.
* Receiver Port (cổng nhận): SWC nhận dữ liệu từ các SWC khác thông qua receiver port. Các SWC khác có thể gửi dữ liệu đến cổng nhận này.
* Client Port: SWC sử dụng client port để gửi yêu cầu đến các SWC khác để yêu cầu thực hiện một chức năng nào đó. Các SWC khác phải cung cấp server port để thực hiện chức năng đó.
* Server Port: SWC cung cấp server port để phục vụ yêu cầu từ các SWC khác thông qua client port. Khi nhận được yêu cầu, SWC sẽ thực hiện chức năng tương ứng và gửi kết quả đến SWC yêu cầu thông qua server port.
* Trigger Port: SWC sử dụng trigger port để gửi tín hiệu kích hoạt (trigger signal) cho các SWC khác. Trigger signal sẽ được sử dụng để đồng bộ hóa các hoạt động giữa các SWC. Các SWC khác có thể đăng ký nhận trigger signal từ cổng này.
* Sender-Receiver Port: Sender-Receiver Port kết hợp chức năng của cổng gửi và cổng nhận. SWC có thể gửi và nhận dữ liệu thông qua cổng này. Tuy nhiên, SWC chỉ có thể gửi dữ liệu đến một SWC nhận duy nhất.
* Client-Server Port: Client-Server Port kết hợp chức năng của client port và server port. SWC có thể gửi yêu cầu và nhận kết quả thông qua cổng này. Tuy nhiên, SWC chỉ có thể gửi yêu cầu đến một SWC cung cấp server port duy nhất.

### Runnable

Trong một software component (SWC), runnable là một phần của code được thực thi và chứa logic xử lý cụ thể của SWC đó. Mỗi SWC có thể chứa nhiều runnable và mỗi runnable được định nghĩa bởi một hoặc nhiều entry point (điểm vào) để được gọi bởi RTE (Runtime Environment)[4].

Một runnable có thể được kích hoạt thông qua một event hoặc một periodic task. Event được gửi tới SWC thông qua các port hoặc interface, trong khi periodic task được lập lịch bởi RTE và kích hoạt các runnable tương ứng.

Các runnable được chia thành các loại khác nhau, bao gồm:

* Runnable cập nhật trạng thái (State Update Runnable): Thực hiện cập nhật trạng thái của SWC, bao gồm việc đọc các giá trị từ các port hoặc interface và cập nhật các biến nội bộ của SWC.
* Runnable xử lý sự kiện (Event Runnable): Thực hiện xử lý sự kiện được kích hoạt thông qua các port hoặc interface.
* Runnable xử lý các luồng dữ liệu (Data Processing Runnable): Thực hiện xử lý dữ liệu được đọc từ các port hoặc interface và thực hiện các tính toán để tạo ra các giá trị mới.
* Runnable xử lý lỗi (Error Handling Runnable): Thực hiện xử lý lỗi và cập nhật trạng thái của SWC khi có lỗi xảy ra.
* Runnable xử lý các tác vụ định kỳ (Periodic Runnable): Thực hiện xử lý các tác vụ được lập lịch định kỳ bởi RTE.
* Mỗi loại runnable có một mục đích và chức năng khác nhau, và được sử dụng để xử lý các nhiệm vụ cụ thể trong SWC.

### Event

Trong AUTOSAR, SWC Event là một sự kiện mà một hoặc nhiều software component (SWC) có thể gửi hoặc nhận để tương tác với nhau. SWC Event thường được sử dụng để đồng bộ hóa việc thực thi của các SWC khác nhau trong một hệ thống AUTOSAR[4].

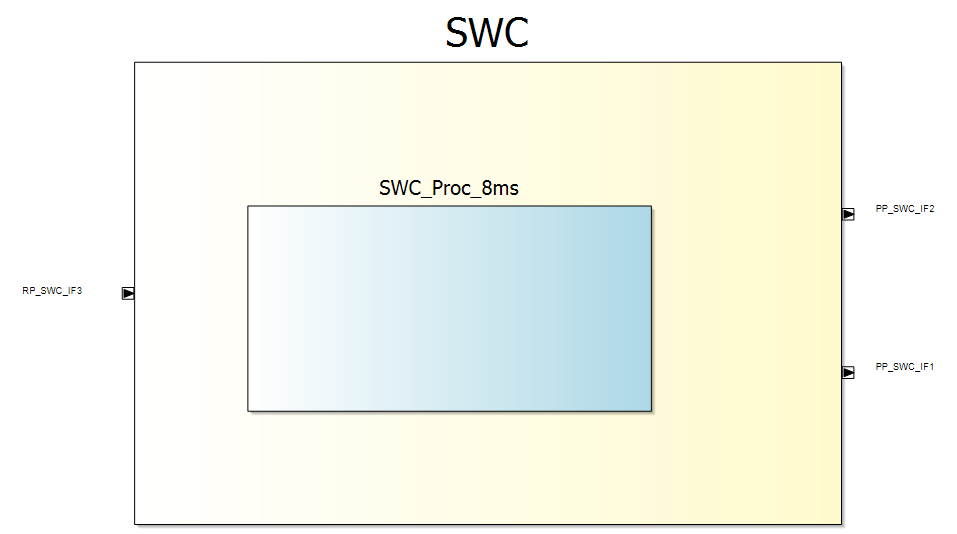
Một SWC Event được định nghĩa bởi một SWC và có thể được sử dụng bởi các SWC khác trong hệ thống. Một SWC Event có thể được kích hoạt bởi một SWC hoặc bởi một điều kiện nào đó trong hệ thống, và các SWC khác có thể đăng ký để nhận thông báo về sự kiện này.

Ví dụ, khi một SWC Event được kích hoạt, một SWC có thể gửi một tín hiệu tới các SWC khác trong hệ thống để thông báo về sự kiện này. Các SWC khác có thể sử dụng tín hiệu này để bắt đầu thực hiện các hoạt động liên quan đến sự kiện đó.

Các loại SWC Event trong AUTOSAR bao gồm:

* Data Received Event (DRE): sự kiện xảy ra khi một SWC nhận dữ liệu từ một SWC khác.
* Operation Invoked Event (OIE): sự kiện xảy ra khi một SWC gọi một hoạt động (operation) của một SWC khác.
* Timing Event (TE): sự kiện xảy ra theo thời gian định kỳ.
* Mode Switch Event (MSE): sự kiện xảy ra khi hệ thống chuyển đổi sang một chế độ hoạt động khác.
* Lifecycle Event (LE): sự kiện xảy ra khi một SWC thay đổi trạng thái.

Hình 2.3 mô tả 1 SWC có 2 Provider Port là PP\_SWC\_IF1 và PP\_SWC\_IF2, 1 Receiver Port RP\_SWC\_IF3, 1 Runnable SWC\_Proc\_8ms được cấu hình để chạy với chu kỳ 8ms.



Hình 2.3 Ví dụ về SWC

## AUTOSAR RTE Generator

AUTOSAR RTE (Run-Time Environment) Generator là công cụ tự động tạo ra mã nguồn phần mềm dựa trên mô hình phần mềm AUTOSAR được cấu hình trong file arxml, giúp giảm thiểu công sức của lập trình viên trong việc triển khai phần mềm trên các thiết bị nhúng. Nó cung cấp các API (Application Programming Interface) cho các Component trong phần mềm để giao tiếp và trao đổi dữ liệu với nhau thông qua các Port và Interface. AUTOSAR RTE Generator được sử dụng trong các dự án phát triển phần mềm nhúng của các nhà sản xuất ô tô và công nghiệp xe hơi, giúp họ nhanh chóng tạo ra các ứng dụng chất lượng cao và đáp ứng các tiêu chuẩn an toàn[5].

## Hệ điều hành thời gian thực FreeRTOS

FreeRTOS (Free Real-time Operating System) là một hệ điều hành thời gian thực (RTOS) mã nguồn mở được thiết kế để sử dụng trên các thiết bị nhúng, bao gồm các vi điều khiển, máy tính nhúng, cảm biến, thiết bị y tế, thiết bị giao tiếp mạng, các thiết bị IoT và nhiều thiết bị khác[6].

FreeRTOS được phát triển bởi Richard Barry, và hiện nay được bảo trợ bởi AWS (Amazon Web Services)[7]. FreeRTOS có kích thước nhỏ, linh hoạt, cung cấp một bộ lập lịch đa nhiệm, hỗ trợ cho nhiều tác vụ, semaphores, mutexes, queues, timers và hỗ trợ cho các phương pháp quản lý bộ nhớ như Heap Memory Management hoặc Static Memory Allocation.

FreeRTOS được thiết kế để đáp ứng yêu cầu của các ứng dụng thời gian thực, nghĩa là các ứng dụng phải thực hiện các tác vụ trong thời gian quy định và đáp ứng các yêu cầu thời gian thực. Nó cung cấp một bộ lập lịch đa nhiệm tiên tiến để quản lý tài nguyên và điều phối các tác vụ, cho phép các tác vụ được thực hiện đồng thời mà không phụ thuộc vào nhau. Điều này cung cấp hiệu quả và độ tin cậy cao cho các ứng dụng thời gian thực.

FreeRTOS cũng hỗ trợ các tính năng bảo mật để bảo vệ hệ thống nhúng khỏi các cuộc tấn công và các lỗ hổng bảo mật. Nó có các tính năng như bảo vệ khu vực bộ nhớ và kiểm soát truy cập tài nguyên, cũng như hỗ trợ các giao thức mạng an toàn như SSL / TLS.

FreeRTOS cũng được hỗ trợ bởi một cộng đồng lớn và nhiều nhà phát triển trên toàn thế giới, vì vậy việc tìm kiếm tài liệu, hỗ trợ và giải đáp thắc mắc trên các diễn đàn thường rất dễ dàng.

Với các tính năng linh hoạt, hiệu suất và độ tin cậy cao, FreeRTOS là một lựa chọn phổ biến cho các ứng dụng nhúng thời gian thực[8].

## Các khái niệm trong FreeRTOS:

### Task

Trong FreeRTOS, một task là một phần của ứng dụng được thực thi bởi hệ điều hành như một luồng độc lập. Mỗi task có một số thuộc tính, bao gồm tên, ưu tiên, kích thước của stack, địa chỉ của hàm task, và một con trỏ để lưu trữ dữ liệu của task[9].

Các task trong FreeRTOS được quản lý bởi kernel của hệ điều hành và được thực thi bởi một lập lịch đa nhiệm. Khi một task được tạo, nó được thêm vào một danh sách các task sẵn sàng để được thực thi. Kernel sẽ sau đó chọn task tiếp theo để thực thi dựa trên mức ưu tiên và trạng thái của các task khác.

Mỗi task trong FreeRTOS sẽ chạy trong một vòng lặp vô hạn. Trong vòng lặp này, task sẽ thực hiện các hoạt động và xử lý dữ liệu. Khi task đã thực hiện xong tác vụ của nó, nó sẽ chuyển sang trạng thái Blocked (chặn) hoặc Suspended (tạm dừng) cho đến khi nó được kích hoạt lại bởi một sự kiện nào đó.

Các task trong FreeRTOS được sử dụng để phân chia ứng dụng thành các phần độc lập và có thể thực thi đồng thời. Việc sử dụng task giúp tăng khả năng mở rộng của hệ thống, cho phép các tác vụ được chia sẻ và thực hiện đồng thời, tăng hiệu suất và hiệu quả của hệ thống.

Các task cũng được sử dụng để tương tác với các tài nguyên khác trong hệ thống như queues, semaphores và mutexes. Task có thể đọc hoặc ghi dữ liệu vào queue hoặc tăng / giảm giá trị của semaphore hoặc mutex để đồng bộ hóa truy cập tài nguyên giữa các task khác trong hệ thống.

Trong FreeRTOS, mỗi task được xác định bởi các đặc tính sau:

* Tên Task: Là tên duy nhất được gán cho task để dễ dàng xác định nó trong hệ thống.
* Ưu tiên (Priority): Là mức độ ưu tiên của task. Trong FreeRTOS, các task có độ ưu tiên cao hơn sẽ được lập lịch thực thi trước các task có độ ưu tiên thấp hơn. Các task có thể có cùng mức độ ưu tiên hoặc khác nhau.
* Stack Size: Là kích thước của stack được cấp phát cho task. Stack sử dụng để lưu trữ các biến cục bộ và các biến trung gian của task. Việc cấp phát kích thước stack phù hợp cho task sẽ giúp tránh tình trạng stack overflow (tràn stack).
* Con trỏ hàm Task: Là địa chỉ của hàm sẽ được thực thi khi task được kích hoạt. Hàm này được gọi là task function hoặc entry point của task.
* Tham số của Task: Là các giá trị được truyền vào task function để thực hiện các tác vụ cụ thể. Tham số này có thể là con trỏ đến các cấu trúc dữ liệu cần thiết hoặc là giá trị nguyên.
* Con trỏ Task Handle: Là con trỏ đến task handle. Task handle được sử dụng để thực hiện các thao tác liên quan đến task như kiểm tra trạng thái của task, xóa task, hoặc thay đổi ưu tiên của task.
* Trạng thái Task: Là trạng thái hiện tại của task. Task có thể ở trạng thái Ready (sẵn sàng), Running (đang chạy), Blocked (bị chặn), hoặc Suspended (tạm dừng).

Các đặc tính này được sử dụng để tạo và quản lý các task trong hệ thống FreeRTOS. Chúng giúp cho các task thực hiện các tác vụ của mình một cách hiệu quả và an toàn, đồng thời đảm bảo tính tin cậy và đáp ứng các yêu cầu thời gian thực của hệ thống.

### Cấu trúc dữ liệu

FreeRTOS sử dụng một số cấu trúc dữ liệu quan trọng để quản lý các tài nguyên trong hệ thống. Dưới đây là một số cấu trúc dữ liệu quan trọng trong FreeRTOS:

* Task Control Block (TCB): TCB là cấu trúc dữ liệu được sử dụng để lưu trữ thông tin về một task cụ thể trong hệ thống. TCB bao gồm các thông tin như địa chỉ của task function, độ ưu tiên của task, trạng thái của task, stack pointer, và nhiều thông tin khác. Mỗi task trong hệ thống có một TCB tương ứng để quản lý các thông tin liên quan đến task đó[9].
* Semaphore Control Block (SCB): SCB là cấu trúc dữ liệu được sử dụng để lưu trữ thông tin về một semaphore. SCB bao gồm các thông tin như giá trị của semaphore, danh sách các task đang chờ semaphore, và các thông tin khác liên quan đến semaphore đó. Mỗi semaphore trong hệ thống có một SCB tương ứng để quản lý các thông tin liên quan đến semaphore đó[9].
* Mutex Control Block (MCB): MCB là cấu trúc dữ liệu được sử dụng để lưu trữ thông tin về một mutex. MCB bao gồm các thông tin như trạng thái của mutex, danh sách các task đang chờ mutex, và các thông tin khác liên quan đến mutex đó. Mỗi mutex trong hệ thống có một MCB tương ứng để quản lý các thông tin liên quan đến mutex đó[9].
* Queue Control Block (QCB): QCB là cấu trúc dữ liệu được sử dụng để lưu trữ thông tin về một queue. QCB bao gồm các thông tin như độ dài của queue, danh sách các phần tử trong queue, và các thông tin khác liên quan đến queue đó. Mỗi queue trong hệ thống có một QCB tương ứng để quản lý các thông tin liên quan đến queue đó[9].
* Timer Control Block (TCB): TCB là cấu trúc dữ liệu được sử dụng để lưu trữ thông tin về một timer. TCB bao gồm các thông tin như khoảng thời gian của timer, task được gọi khi timer hết hạn, và các thông tin khác liên quan đến timer đó. Mỗi timer trong hệ thống có một TCB tương ứng để quản lý các thông tin liên quan đến timer đó[9].

Các cấu trúc dữ liệu này được sử dụng để quản lý các tài nguyên trong hệ thống FreeRTOS. Chúng đóng vai trò quan trọng trong việc đảm bảo tính tin cậy và toàn vẹn của dữ liệu.

### Scheduler

Scheduler trong FreeRTOS là một phần mềm quản lý tài nguyên của hệ thống như bộ nhớ, CPU và thời gian. Scheduler đảm bảo rằng các task được sắp xếp và thực thi theo đúng độ ưu tiên và thời gian tương ứng với mỗi task[9].

Trong FreeRTOS, có hai loại scheduler: pre-emptive và co-operative. Scheduler pre-emptive là loại scheduler mà nó có thể ngắt bất kỳ task nào để chạy task khác có độ ưu tiên cao hơn. Scheduler co-operative chỉ chuyển quyền kiểm soát từ một task sang task khác khi task đang thực thi hoàn tất hoặc gọi một hàm chuyển giao (yield).

Trong FreeRTOS, scheduler được thiết kế để có thể hoạt động trên các hệ thống đa nhiệm và đa luồng. Scheduler có thể quản lý đến hàng trăm hoặc thậm chí hàng ngàn task một cách hiệu quả.

Scheduler được triển khai bằng cách sử dụng một hàm tick (hay tick interrupt) được gọi định kỳ theo một khoảng thời gian nhất định. Khi hàm tick được gọi, scheduler sẽ kiểm tra xem các task hiện tại có cần được chuyển đổi hay không, nếu cần thì scheduler sẽ chuyển quyền kiểm soát sang task khác có độ ưu tiên cao hơn.

Việc chuyển quyền kiểm soát giữa các task được thực hiện bằng cách lưu trữ các task vào một danh sách sắp xếp theo độ ưu tiên, được gọi là ready list. Scheduler sẽ luôn chọn task đầu tiên trong danh sách này để thực thi.

Tùy thuộc vào loại scheduler được sử dụng, các task có thể được chuyển đổi trong trường hợp có một task khác có độ ưu tiên cao hơn được yêu cầu thực thi ngay lập tức (pre-emptive scheduler), hoặc chỉ được chuyển đổi khi task hiện tại thực thi xong hoặc yêu cầu chuyển giao (co-operative scheduler).

Trong tổng quan, scheduler là một phần quan trọng trong FreeRTOS để quản lý các task trong hệ thống một cách hiệu quả và đảm bảo tính đáp ứng và ổn định của hệ thống.

# ÁP DỤNG KẾT QUẢ NGHIÊN CỨU

## Kết quả nghiên cứu ở phạm vi bài nghiên cứu

Ở phạm vi đề tài, kết quả nghiên cứu – AUTOSAR RTE generator có ứng dụng để xuất mã nguồn của RTE dựa trên những cấu hình có tronng các file đầu vào arxml. Từ đó, kiến trúc phần mềm được thiết kế dựa theo chuẩn AUTOSAR có thể được triển khai trên các thiết bị nhúng.

## Kết quả nghiên cứu ở phạm vi rộng

Công cụ mã nguồn mở mà chúng tôi xây dựng có thể giúp cộng đồng Autosar dễ dàng tiếp cận và sử dụng công nghệ này một cách hiệu quả hơn. Nó không chỉ giúp cho các công ty hoặc tổ chức nghiên cứu tiết kiệm được chi phí sở hữu công cụ, mà còn giúp cho các cá nhân hay nhóm nghiên cứu độc lập có thể tiếp cận và sử dụng công nghệ này. Điều này đặc biệt quan trọng đối với những quốc gia đang phát triển kinh tế, khi ngân sách cho nghiên cứu và phát triển không được cao.

Việc tạo ra công cụ mã nguồn mở có thể giúp mở rộng phạm vi áp dụng của Autosar trong các lĩnh vực khác nhau, bao gồm các lĩnh vực phát triển ô tô, xe đạp điện, thiết bị y tế và thiết bị thông minh. Nó cũng có thể giúp đẩy nhanh quá trình chuyển đổi sang xe tự lái thông qua việc cung cấp các công cụ hỗ trợ cho các nhà phát triển phần mềm để xây dựng các hệ thống điều khiển xe tự động.

Ngoài ra, việc sử dụng công cụ mã nguồn mở có thể tạo ra sự tương tác giữa các nhà nghiên cứu trên toàn cầu, giúp tăng cường sự đóng góp và phát triển của cộng đồng Autosar. Nó cũng có thể giúp các nhà nghiên cứu phát hiện các lỗi và vấn đề tiềm ẩn trong công nghệ và cải tiến nó để phù hợp hơn với nhu cầu thực tế.

Trong tương lai, chúng tôi hy vọng rằng công cụ mã nguồn mở sẽ được sử dụng rộng rãi trong cộng đồng Autosar, giúp tăng tốc quá trình phát triển và cải tiến công nghệ điều khiển xe ô tô.

KẾT LUẬN VÀ KIẾN NGHỊ

**Kết luận:**

Dựa trên các nội dung đã trình bày trong đề cương luận văn, có thể kết luận rằng việc sử dụng AUTOSAR RTE Generator trong các dự án phát triển phần mềm nhúng cho xe hơi đang trở nên ngày càng phổ biến. Tuy nhiên, việc tiếp cận với AUTOSAR RTE Generator hiện tại đòi hỏi người dùng phải có kiến thức và kinh nghiệm chuyên sâu trong lĩnh vực này và đang làm việc ở một doanh nghiệp lớn đủ khả năng sở hữu một RTE Generator. Vì vậy, việc phát triển phiên bản mã nguồn mở của AUTOSAR RTE Generator là rất cần thiết để giúp đẩy mạnh sự phát triển của cộng đồng lập trình nhúng và giảm thiểu khó khăn trong việc sử dụng và áp dụng chuẩn AUTOSAR trong các dự án. Bên cạnh đó, việc phát triển phiên bản mã nguồn mở cũng sẽ góp phần đưa AUTOSAR RTE Generator tiếp cận được nhiều hơn với các nhà phát triển và cộng đồng lập trình viên trên toàn thế giới thúc đẩy tốc độ phát triển của ngành công nghiệp phần mềm xe hơi nói riêng và các lĩnh vực khác nói chung.

**Kiến nghị:**

* Tăng cường tính năng tương thích và hỗ trợ nhiều phiên bản AUTOSAR khác nhau.
* Phát triển các plugin hoặc addons để hỗ trợ tích hợp và tương thích với các công cụ phát triển phần mềm khác.
* Cung cấp tài liệu hướng dẫn chi tiết và các ví dụ minh họa để giúp người dùng có thể sử dụng và tùy chỉnh mã nguồn mở theo nhu cầu của mình.

TÀI LIỆU THAM KHẢO

[1] Wikipedia, “AUTOSAR - Wikipedia.” https://en.wikipedia.org/wiki/AUTOSAR

[2] Prakash Kumar, “Autosar Architecture (Learn from Scratch with Demo) - Lesson 22 RTE Generator,” 2023. https://www.udemy.com/course/autosar-architecture/ (accessed Mar. 06, 2023).

[3] D. Wang, S. Liu, B. Huang, G. Zhao, and J. Zheng, “Communication mechanisms on the virtual functional bus of AUTOSAR,” *2010 Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2010*, vol. 1, pp. 982–985, 2010, doi: 10.1109/ICICTA.2010.531.

[4] AUTOSAR, “Specification of RTE 4.2.2,” no. January, pp. 1–16, 2015.

[5] S. Piao, H. Jo, S. Jin, and W. Jung, “Design and implementation of RTE generator for automotive embedded software,” *Proc. - 7th ACIS Int. Conf. Softw. Eng. Res. Manag. Appl. SERA09*, pp. 159–165, 2009, doi: 10.1109/SERA.2009.35.

[6] W. Nakano, Y. Shinohara, and N. Ishiura, “Full Hardware Implementation of FreeRTOS-Based Real-Time Systems,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2021-December, pp. 435–440, 2021, doi: 10.1109/TENCON54134.2021.9707328.

[7] Wikipedia, “FreeRTOS - Wikipedia.” https://nl.wikipedia.org/wiki/FreeRTOS (accessed Mar. 06, 2023).

[8] I. Fergusen, “What Are the Most Popular Real-Time Operating Systems?,” *Lynx Software Technologies*, 2019. https://www.lynx.com/embedded-systems-learning-center/most-popular-real-time-operating-systems-rtos

[9] R. Barry, “Mastering the FreeRTOS TM Real Time Kernel,” pp. 1–399, 2016.