

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**

Ngành: KỸ THUẬT ĐIỆN TỬ

Mã ngành: 8520203

LUẬN VĂN THẠC SĨ

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

Công trình được hoàn thành tại Trường Đại học Công nghiệp TP. Hồ Chí Minh.

Người hướng dẫn khoa học: PGS. TS Nguyễn Ngọc Sơn

Luận văn thạc sĩ được bảo vệ tại Hội đồng chấm bảo vệ Luận văn thạc sĩ Trường Đại học Công nghiệp thành phố Hồ Chí Minh ngày tháng năm 2023

Thành phần Hội đồng đánh giá luận văn thạc sĩ gồm:

1. - Chủ tịch Hội đồng
2. - Phản biện 1
3. - Phản biện 2
4. - Ủy viên
5. - Thư ký

(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ luận văn thạc sĩ)

CHỦ TỊCH HỘI ĐỒNG

TRƯỞNG KHOA

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên: VÕ LINH TRÚC

MSHV: 20000411

Ngày, tháng, năm sinh: 26/01/1997

Nơi sinh: Bình Thuận

Ngành: KỸ THUẬT ĐIỆN TỬ

Mã ngành: 8520203

I. TÊN ĐỀ TÀI:

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

NHIỆM VỤ VÀ NỘI DUNG:

- Nghiên cứu và thiết kế RTE Generator dựa trên tiêu chuẩn AUTOSAR 4.2.2.
- Thiết kế một phần mềm demo đơn giản với cấu hình theo chuẩn AUTOSAR.
- Triển khai mã nguồn của phần mềm sau khi được tạo ra từ RTE Generator lên hệ thống phần cứng demo có tích hợp vi điều khiển ESP32.
- Tiến hành kiểm thử kết quả hoạt động của hệ thống phần cứng demo dựa trên bản thiết kế phần mềm demo.
- Xây dựng một kho lưu trữ mã nguồn của RTE Generator và công khai trên github.

II. NGÀY GIAO NHIỆM VỤ:

Theo quyết định số 1435/QĐ-ĐHCN ngày 07/03/2023

III. NGÀY HOÀN THÀNH NHIỆM VỤ: 07/09/2023

IV. NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS. Nguyễn Ngọc Sơn

Tp. Hồ Chí Minh, ngày ... tháng ... năm 20 ...

NGƯỜI HƯỚNG DẪN

CHỦ NHIỆM BỘ MÔN ĐÀO TẠO

TRƯỞNG KHOA CÔNG NGHỆ ĐIỆN TỬ

LỜI CẢM ƠN

Để có được thành quả là bài luận văn này, không chỉ dừng lại ở sự nỗ lực học hỏi của bản thân tôi mà cùng với đó là sự giúp đỡ và hỗ trợ của gia đình, thầy cô và bạn bè.

Lời đầu tiên, tôi xin chân thành cảm ơn quý Thầy, Cô Trường Đại Học Công Nghiệp Thành Phố Hồ Chí Minh nói chung và khoa Công Nghệ Điện Tử nói riêng, đã tận tình giảng dạy cho chúng tôi trong suốt khoảng thời gian học tại trường. Đặc biệt, tôi luôn biết ơn quý Thầy, Cô trong khoa Công Nghệ Điện Tử với sự hỗ trợ và tạo mọi điều kiện tốt nhất cho tôi được sử dụng phòng thí nghiệm, thiết bị phân tích cũng như luôn ủng hộ và cho tôi những lời khuyên quý giá trong khoảng thời gian nghiên cứu đề tài này.

Tôi xin gửi lời cảm ơn đến PGS. TS Nguyễn Ngọc Sơn – người đã trực tiếp hướng dẫn, hỗ trợ, động viên tôi từ những ngày đầu tiên tiếp cận đề tài đến ngày cuối cùng hoàn thành luận văn.

Mặc dù, đã cố gắng để hoàn thành luận văn, nhưng do bản thân còn nhiều hạn chế và các yếu tố khách quan mà khóa luận không thể không có thiếu sót. Kính mong nhận được góp ý của Thầy/Cô để luận văn này hoàn thiện tốt hơn. Tôi xin trân trọng cảm ơn.

TP. Hồ Chí Minh, ngày ... tháng ... năm 2023

Học viên thực hiện

Võ Linh Trúc

TÓM TẮT LUẬN VĂN THẠC SĨ

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. Điều đó giúp cho việc phát triển phần mềm cho hệ thống điện tử trở nên dễ dàng hơn và hiệu quả hơn [1].

Nghiên cứu này hướng đến việc phát triển một công cụ AUTOSAR RTE Generator mã nguồn mở, nhằm giúp cho các nhà phát triển, sinh viên và những cá nhân có nhu cầu có thể tiếp cận với công nghệ AUTOSAR một cách dễ dàng và thuận tiện. Đồng thời, kết quả của nghiên cứu này cũng hướng tới việc thúc đẩy một nguồn nhân lực chất lượng, thành thạo trong việc áp dụng AUTOSAR, giúp thúc đẩy sự phát triển của ngành công nghiệp phần mềm nhúng ô tô.

Cụ thể, công cụ AUTOSAR RTE Generator được thiết kế trong nghiên cứu này có những nhiệm vụ sau:

- Nhận biết tệp cấu hình AUTOSAR RTE (*.arxml) trong một cây thư mục của dự án.
- Phân tích và xử lý dữ liệu trong tệp arxml sau đó chuyển thành dạng JSON để thuận tiện cho việc trích xuất dữ liệu về sau.
- Trích xuất dữ liệu từ dạng JSON, liên kết các dữ liệu liên quan để tạo ra các API (giao diện lập trình ứng dụng) thích hợp.
- Cũng từ dữ liệu ở dạng JSON, các tác vụ (task) AUTOSAR được cấu hình trước đó sẽ được chuyển thành mã nguồn C++ và các tác vụ này được quản lý bởi FreeRTOS.

- Thu thập các tệp mã nguồn ứng dụng và tổng hợp lại để nhà phát triển tiến hành compile ra ứng dụng cuối cùng.

Và để kiểm tra tính ứng dụng của công cụ AUTOSAR RTE Generator, một ứng dụng xe demo được giới thiệu bao gồm 3 ASW được cấu hình theo AUTOSAR RTE với chức năng đơn giản là điều khiển xe vào bãi đỗ qua giao diện web, xe tự động dừng nếu khoảng cách với vật cản phía sau quá gần. Những ASW trong demo cụ thể như sau:

- ASW1: Đóng vai trò là webserver nhận tín hiệu điều từ giao diện web của người dùng, gửi tín hiệu từ người dùng đến ASW để điều khiển động cơ xe.
- ASW2: Thực hiện việc nhận tín hiệu từ ASW1 để điều khiển động cơ, đồng thời kiểm tra tín hiệu từ ASW3 ra quyết định cho phép động cơ hoạt động hay không.
- ASW3: Đọc dữ liệu khoảng cách từ cảm biến, xử lý và cung cấp tín hiệu khoảng cách tới ASW2

ABSTRACT

In recent years, automotive technology has witnessed substantial advancements, marked by the emergence of numerous new technologies such as self-driving cars, intelligent connectivity, and notably advanced electronic systems. Alongside this development, the need to construct integrated software for vehicle electronic systems has grown increasingly crucial. However, developing software for vehicle electronic systems is a highly complex and time-consuming task. To address this challenge, AUTOSAR (Architecture for Automotive Software) was established to define a software architecture for vehicle electronic systems. It facilitates the development of software for electronic systems, making the process smoother and more efficient [1].

This study aims to develop an open-source AUTOSAR RTE Generator tool, intended to provide developers, students, and individuals with an easy and convenient means of accessing AUTOSAR technology. Additionally, the outcomes of this research also strive to foster a skilled and proficient workforce capable of applying AUTOSAR, thereby promoting the growth of the embedded automotive software industry.

Specifically, the AUTOSAR RTE Generator tool designed in this study has the following tasks:

- Identify AUTOSAR RTE configuration files (*.arxml) within a project directory structure.
- Analyze and process data within the arxml file, then convert it into JSON format for convenient subsequent data extraction.
- Extract data from JSON format, interlink relevant data to create appropriate Application Programming Interfaces (APIs).
- Similarly, using data in JSON format, pre-configured AUTOSAR tasks are transformed into C++ source code, and these tasks are managed by FreeRTOS.
- Collect application source codes and aggregate them for developers to compile into the final application.

And to evaluate the applicability of the AUTOSAR RTE Generator tool, a demo car application is introduced, comprising 3 ASWs (Application Software Components) configured according to AUTOSAR RTE. The primary function of this demo is to control the car's entry into a parking lot through a web interface, with the car automatically stopping if the distance to an obstacle behind it becomes too close. The ASWs in this specific demo are as follows:

- ASW1: Acts as a web server receiving control signals from the user's web interface, transmitting these signals from the user to ASWs to control the car's engine.
- ASW2: Receives signals from ASW1 to control the engine, simultaneously checks signals from ASW3 to decide whether to allow the engine to operate or not.
- ASW3: Read distance data from sensors, processes it, and provides distance signals to ASW2.

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của tôi. Các kết quả nghiên cứu và các kết luận trong luận văn này là trung thực, không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào. Việc tham khảo các nguồn tài liệu (nếu có) đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

Tôi sẽ hoàn toàn chịu trách nhiệm trước nhà trường về sự cam đoan này.

Học viên

Võ Linh Trúc

MỤC LỤC

LỜI CẢM ƠN	i
TÓM TẮT LUẬN VĂN THẠC SĨ.....	ii
ABSTRACT	iv
LỜI CAM ĐOAN	vi
MỤC LỤC.....	vii
DANH MỤC HÌNH ẢNH	xi
DANH MỤC TỪ VIẾT TẮT.....	xiv
MỞ ĐẦU.....	1
1. Đặt vấn đề	1
2. Mục tiêu, mục đích nghiên cứu	2
3. Đối tượng, phạm vi nghiên cứu	3
4. Cách tiếp cận và phương pháp nghiên cứu.....	3
5. Ý nghĩa thực tiễn của đề tài	3
CHƯƠNG 1 TỔNG QUAN.....	6
1.1 Tính cấp thiết đề tài, vấn đề nghiên cứu.....	6
1.1.1 Giới Thiệu	6
1.1.2 Nhu cầu nguồn nhân lực của doanh nghiệp trong mảng phần mềm ô tô	6
1.1.3 Các thách thức	8
1.1.4 Tối ưu điều kiện cho nhà phát triển.....	8
1.1.5 Thúc đẩy đổi mới.....	8
1.1.6 Ý nghĩa giáo dục.....	8
1.1.7 Kết luận	9
1.2 Tổng quan về tình hình nghiên cứu hiện tại	9
1.2.1 Tổng quan về phần mềm nhúng ô tô	9
1.2.2 Tổng quan về tiêu chuẩn AUTOSAR	10

1.2.3 Phân tích các nghiên cứu liên quan đến AUTOSAR RTE generator.....	12
1.3 Nội dung nghiên cứu.....	14
CHƯƠNG 2 CƠ SỞ LÝ THUYẾT.....	15
2.1 Tổng quan về AUTOSAR	15
2.1.1 AUTOSAR là gì	15
2.1.2 Kiến trúc AUTOSAR	16
2.1.3 Lớp Runtime Environment (RTE)	17
2.1.4 Khái niệm Interface trong AUTOSAR.....	18
2.1.5 Application Software (ASW) và thành phần của ASW	19
2.2 Hệ điều hành thời gian thực FreeRTOS	23
2.3 Tổng quan về dữ liệu dưới dạng XML và JSON.....	28
2.3.1 Dữ liệu dưới dạng XML	28
2.3.2 Dữ liệu dưới dạng JSON	28
2.4 Cấu trúc AUTOSAR RTE generator	30
2.4.1 Đặc tính kỹ thuật cần quan tâm khi thiết kế công cụ AUTOSAR RTE generator	31
2.4.2 Phương pháp kiểm tra tính đúng đắn trên phần cứng	33
2.5 Độ quy trong phát triển phần mềm	33
CHƯƠNG 3 TRIỂN KHAI CÔNG CỤ SINH MÃ NGUỒN RTE	36
3.1 Cấu trúc đường dẫn của các tệp.....	36
3.1.1 Thư mục "application"	36
3.1.2 Sắp xếp lại tệp	36
3.1.3 Thư mục “output”	37
3.2 Triển khai các thành phần của AUTOSAR RTE generator.....	38
3.2.1 AUTOSAR XML reader – AXR.....	38
3.2.2 AUTOSAR XML to JSON – AXJ	38

3.2.3 JSON data Extraction and Synthetization – JES	43
3.2.4 Source Code Generator – SCG.....	50
CHƯƠNG 4 KẾT QUẢ THỰC NGHIỆM.....	60
4.1 Giới thiệu cách triển khai công cụ sinh mã nguồn RTE.....	60
4.2 Cấu hình chi tiết thành phần phần mềm cho việc kiểm tra mã RTE được tạo	61
4.2.1 ASW1 - Quản lý giao diện điều khiển xe qua web	61
4.2.2 ASW3 - Thu thập tín hiệu từ cảm biến khoảng cách	63
4.2.3 ASW2 - Điều khiển động cơ	65
4.3 Cấu hình chi tiết phần cứng của demo.....	67
4.3.1 Vi điều khiển xử lý trung tâm - Wemos D1 R32	67
4.3.2 Module điều khiển động cơ - L293D shield.....	69
4.3.3 Cảm biến siêu âm - SRF05.....	70
4.3.4 Nguồn cấp năng lượng – Pin 18650.....	71
4.3.5 Động cơ	72
4.3.6 Sơ đồ kết nối phần cứng của demo	73
4.5 Thực nghiệm kiểm chứng.....	74
4.5.1 Testcase 1: Kiểm tra giá trị của tín hiệu điều khiển mà ASW2 nhận được từ ASW1	74
4.5.2 Testcase 2: Kiểm tra giá trị của tốc độ động cơ mà ASW2 nhận được từ ASW1	74
4.5.3 Testcase 3: Kiểm tra giá trị của cảm biến khoảng cách mà ASW2 nhận được từ ASW3.....	75
4.5.4 Kết quả:	75
KẾT LUẬN VÀ KIẾN NGHỊ.....	77
1. Kết luận.....	77
2. Kiến nghị.....	79
TÀI LIỆU THAM KHẢO.....	80

LÝ LỊCH TRÍCH NGANG CỦA HỌC VIÊN	81
--	----

DANH MỤC HÌNH ẢNH

Hình 1.1 Top 10 ngành nghề có nhu cầu tuyển dụng cao nhất	6
Hình 1.2 Top công ty lớn nhất Việt Nam theo số lượng nhân lực.....	7
Hình 1.3 Kiến trúc AUTOSAR.....	12
Hình 1.4 Ví dụ cấu trúc trong tệp XML AUTOSAR.....	13
Hình 1.5 Đường dẫn liên kết thông tin các đối tượng với nhau được xác định bằng nội dung của tag <SHORT-NAME>	13
Hình 2.1 Kiến trúc phần mềm AUTOSAR	17
Hình 2.2 Sự tương tác giữa các Application Software Component thông qua RTE	18
Hình 2.3 Ký hiệu các loại Interface	18
Hình 2.4 AUTOSAR Interface được biểu diễn dưới dạng XML.....	28
Hình 2.5 Dữ liệu dưới dạng JSON	29
Hình 2.6 Cấu trúc AUTOSAR RTE generator	31
Hình 3.1 Cấu trúc đường dẫn	36
Hình 3.2 Chi tiết cách hoạt động của AXJ.....	40
Hình 3.3 Cách hàm genJsonFromXml được gọi lồng nhau.....	41
Hình 3.4 Cấu trúc dữ liệu dạng XML ban đầu.....	42
Hình 3.5 Cấu trúc dữ liệu sau khi được chuyển đổi sang dạng JSON.....	42
Hình 3.6 Chi tiết cách hoạt động của JES để thu thập thông tin của các Port.....	46
Hình 3.7 Chi tiết hoạt động của JES để thu thập thông tin của các kết nối Port	47

Hình 3.8 Chi tiết cách hoạt động của JES để thu thập thông tin của các liên kết giữa sự kiện và OS_Task.....	49
Hình 3.9 Chi tiết cách hoạt động của JES để thu thập thông tin của các OS Task...	50
Hình 3.10 Quá trình SCG sinh mã cho tệp Rte_Type.h.....	52
Hình 3.11 Quá trình sinh mã khai báo dữ liệu (khai báo biến).....	53
Hình 3.12 Quá trình sinh mã khai cho các API để đọc/ghi dữ liệu	54
Hình 3.13 Quá trình sinh mã định nghĩa hành vi của API.	56
Hình 3.14 Quá trình sinh mã cho hành vi của từng OS Task	57
Hình 3.15 Quá trình tạo OS Task Executor và đăng ký Scheduler của FreeRTOS..	59
Hình 4.1 Kiến trúc ứng dụng demo.....	61
Hình 4.2 Cấu hình Autosar của ASW1	62
Hình 4.3 Tương tác giữa người dùng tới ASW1 của ứng dụng demo	63
Hình 4.4 Giao diện web điều khiển xe.....	63
Hình 4.5 Cấu hình Autosar của ASW3	64
Hình 4.6 Tương tác giữa cảm biến và ASW3	65
Hình 4.7 Cấu hình AUTOSAR của ASW2.....	66
Hình 4.8 Chi tiết hoạt động của ASW2	67
Hình 4.9 Wemos D1 R32 Pinout.....	68
Hình 4.10 L293D Shield	69
Hình 4.11 SRF05.....	70
Hình 4.12 Pin 18650	71

Hình 4.13 Động cơ	72
Hình 4.14 Sơ đồ kết nối phần cứng của demo	73

DANH MỤC TỪ VIẾT TẮT

ADT	Application Data Type
API	Application Programming Interface
ASW	Application Software
AUTOSAR	AUTomotive Open System ARchitecture
AXJ	AUTOSAR XML to JSON
AXR	Run-Time Environment
CPU	Central Processing Unit
DOM	Document Object Model
DRE	Data Received Event
DTMS	Data Type Mapping Set
ECU	Electronic Control Unit
IB	Internal Behaviour
IDT	Implementation Data Type
IF	Interface
JES	JSON data Extraction and Synthetization
JSON	JavaScript Object Notation
LE	Lifecycle Event
MCB	Mutex Control Block
MSE	Mode Switch Event
OIE	Operational Involk Event
OS	Operation System
PP	Provide Port
QCB	Queue Control Block
RE	Runnable Entity
RP	Receive Port

RTE	Runtime Enviroment
RTOS	Real Time OS
SAX	Simple API for XML
SCB	Semaphore Control Block
SCG	Source Code Generator
SSL	Secure Sockets Layer
SW	Software
SWC	Software Component
TCB	Task Control Block
TE	Timing Event
TLS	Transport Layer Security
VDP	Variable Data Point
XML	eXtensible Markup Language

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.

Từ những nội dung trên, chúng tôi đã chọn và tiến hành nghiên cứu đề tài: “: *Thiết kế và triển khai công cụ mã nguồn mở RTE generator cho phần mềm nhúng xe ô tô*”

2. Mục tiêu, mục đích nghiên cứu

Mục tiêu:

- Thiết kế công cụ AUTOSAR RTE Generator với nhiệm vụ phân tích tệp cấu hình AUTOSAR RTE và sinh mã nguồn;
- Tích hợp mã nguồn được sinh ra với hệ điều hành nhúng FreeRTOS;
- Kiểm tra tính đúng đắn của mã nguồn được sinh ra trên phần cứng demo chạy vi điều khiển ESP32;

Mục đích:

- Tạo ra công cụ AUTOSAR Generator mã nguồn mở để đáp ứng nhu cầu học tập và phát triển phần mềm nhúng theo chuẩn AUTOSAR.

3. Đối tượng, phạm vi nghiên cứu

Đối tượng: công cụ AUTOSAR RTE Generator

Phạm vi: quy mô phòng thí nghiệm

4. Cách tiếp cận và phương pháp nghiên cứu

Cách tiếp cận: dựa trên kinh nghiệm thực tế trong quá trình làm việc tại doanh nghiệp, tác giả nhận thấy nhu cầu về một công cụ AUTOSAR RTE generator mã nguồn mở thực sự rất cần thiết để góp phần phát triển ngành công nghiệp phần mềm ô tô.

Phương pháp nghiên cứu: Nghiên cứu này được thực hiện trên phương pháp phân tích, giá giá và tổng hợp các thông tin về tiêu chuẩn của AUTOSAR RTE 4.2.2, cách hoạt động của FreeRTOS để thiết kế ra một AUTOSAR RTE generator tương thích với AUTOSAR RTE 4.2.2 và FreeRTOS v202212.01. Đồng thời phương pháp thực nghiệm của được áp dụng để kiểm tra tính đúng đắn của kết quả cuối cùng mà AUTOSAR RTE generator mã nguồn tạo ra.

5. Ý nghĩa thực tiễn của đề tài

Việc phát triển trình tạo AUTOSAR RTE (Môi trường thời gian chạy) mã nguồn mở mang lại ý nghĩa và lợi ích thiết thực đáng kể cho ngành phát triển phần mềm ô tô. Dưới đây là một số ý nghĩa thực tiễn quan trọng của chủ đề này:

- Giảm chi phí và tăng khả năng tiếp cận: Các trình tạo AUTOSAR RTE độc quyền thường có chi phí cấp phép cao, khiến các công ty nhỏ hơn, công ty khởi nghiệp và nhà phát triển độc lập không thể tiếp cận được chúng. Giải pháp nguồn mở sẽ loại bỏ những rào cản tài chính này, dân chủ hóa quyền truy cập vào các công cụ phát triển phần mềm quan trọng. Đồng thời điều này cho phép nhiều nhà phát triển hơn tham gia vào ngành.
- Tăng tốc đổi mới: Bằng cách cung cấp trình tạo RTE nguồn mở, cộng đồng phát triển phần mềm ô tô có thể cộng tác hiệu quả hơn. Điều này thúc đẩy sự đổi mới khi các nhà phát triển có thể chia sẻ kiến thức, các phương pháp hay nhất và cải tiến một cách cởi mở. Do đó, tốc độ đổi mới trong lĩnh vực phần mềm ô tô có

thể tăng lên, dẫn đến sự phát triển nhanh chóng của các tính năng và công nghệ tiên tiến.

- Tùy chỉnh và khả năng thích ứng: Phần mềm nguồn mở cho phép các nhà phát triển điều chỉnh trình tạo RTE theo nhu cầu cụ thể của họ. Khả năng thích ứng này đặc biệt có giá trị trong một ngành mà các phương tiện và hệ thống khác nhau có những yêu cầu riêng. Các nhà phát triển có thể sửa đổi trình tạo mã nguồn mở để phù hợp nhu cầu dự án của họ, mang đến các giải pháp phần mềm tối ưu và hiệu quả hơn.
- Học tập và Giáo dục: Nguồn mở khuyến khích học tập và giáo dục. Sinh viên, nhà nghiên cứu và nhà giáo dục trong lĩnh vực phát triển phần mềm ô tô có thể hưởng lợi từ các công cụ nguồn mở bằng cách nghiên cứu mã, thử nghiệm và tích lũy kinh nghiệm thực hành. Việc tiếp xúc thực tế này có thể giúp chuẩn bị cho thế hệ kỹ sư và nhà nghiên cứu ô tô tiếp theo.
- Sự độc lập với nhà cung cấp: Sự phụ thuộc vào các công cụ độc quyền thường có nghĩa là bị ràng buộc với một nhà cung cấp cụ thể. Với phần mềm nguồn mở, các tổ chức có được sự độc lập với nhà cung cấp. Họ có nhiều quyền kiểm soát hơn đối với các công cụ của mình và có thể điều chỉnh chúng khi cần mà không bị ràng buộc vào hệ sinh thái của một nhà cung cấp cụ thể.
- Hợp tác cộng đồng: Nguồn mở thúc đẩy ý thức cộng đồng và cộng tác giữa các nhà phát triển. Một cộng đồng những người đóng góp có thể hình thành xung quanh trình tạo RTE nguồn mở, chia sẻ thông tin chi tiết, sửa lỗi và cải tiến. Cách tiếp cận hợp tác này có thể mang lại một công cụ mạnh mẽ và đáng tin cậy hơn theo thời gian.
- Tuân thủ và tiêu chuẩn hóa: Trình tạo AUTOSAR RTE nguồn mở có thể tuân thủ chặt chẽ hơn các tiêu chuẩn ngành và yêu cầu tuân thủ. Tính minh bạch của quá trình phát triển nguồn mở giúp dễ dàng hơn trong việc đảm bảo rằng mã được tạo tuân thủ các tiêu chuẩn liên quan, điều này rất quan trọng đối với các hệ thống ô tô quan trọng về an toàn.

- Tính bền vững lâu dài: Các công cụ độc quyền có thể đến và đi cùng với những thay đổi trên thị trường hoặc số phận của nhà cung cấp. Các dự án nguồn mở có xu hướng bền vững lâu dài hơn vì chúng được duy trì bởi cộng đồng người dùng và cộng tác viên. Điều này đảm bảo rằng trình tạo RTE luôn phù hợp và cập nhật qua nhiều năm.
- Tăng trưởng hệ sinh thái: Trình tạo AUTOSAR RTE nguồn mở có thể đóng góp vào sự phát triển của hệ sinh thái phát triển phần mềm ô tô lớn hơn. Khi ngày càng có nhiều nhà phát triển và tổ chức áp dụng và đóng góp cho công cụ nguồn mở, nó sẽ trở thành một yếu tố nền tảng của hệ sinh thái, thúc đẩy hơn nữa sự đổi mới và hợp tác.

Tóm lại, việc phát triển trình tạo AUTOSAR RTE nguồn mở không chỉ là nỗ lực về mặt kỹ thuật; nó có ý nghĩa thực tế giúp tiết kiệm chi phí, đổi mới, tùy chỉnh, giáo dục, tính độc lập của nhà cung cấp, tuân thủ, tính bền vững và sự tăng trưởng chung của hệ sinh thái phát triển phần mềm ô tô. Nó phù hợp với việc ngành theo đuổi các hoạt động phát triển phần mềm hợp tác, hiệu quả và dễ tiếp cận hơn.

CHƯƠNG 1 TỔNG QUAN

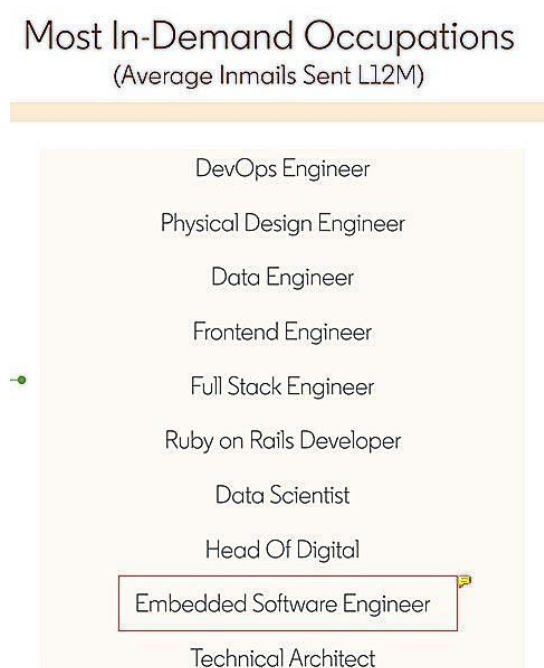
1.1 Tính cấp thiết đề tài, vấn đề nghiên cứu

1.1.1 Giới Thiệu

Lĩnh vực phát triển phần mềm ô tô đã phát triển nhanh chóng với việc tích hợp các công nghệ tiên tiến vào các phương tiện hiện đại. Tiêu chuẩn AUTOSAR (Kiến trúc hệ thống mở ô tô) đã nổi lên như một bộ khung quy tắc then chốt để tạo điều kiện thuận lợi cho việc quản lý độ phức tạp của phần mềm trong các hệ thống ô tô phức tạp này. Tuy nhiên, tính sẵn có và khả năng tiếp cận của các công cụ để tạo AUTOSAR Runtime Environment (RTE) một cách hiệu quả đã đặt ra những thách thức đáng kể cho các nhà phát triển. Phần này làm sáng tỏ tính cấp bách và sự cần thiết cho việc tạo ra trình tạo AUTOSAR RTE nguồn mở.

1.1.2 Nhu cầu nguồn nhân lực của doanh nghiệp trong mảng phần mềm ô tô

Theo báo cáo từ LinkedIn - một nền tảng truyền thông xã hội tập trung vào kinh doanh và việc làm, kỹ sư phần mềm nhúng thuộc top 10 những ngành nghề có nhu cầu tuyển dụng cao nhất.



Hình 1.1 Top 10 ngành nghề có nhu cầu tuyển dụng cao nhất

Hơn nữa, báo cáo chuyên ngành của LinkedIn về lĩnh vực phần mềm nhúng ô tô khẳng định lại nhu cầu ngày càng tăng đặc biệt là về kỹ năng AUTOSAR. Điều đáng kinh ngạc là trong báo cáo tính đến tháng 8 năm 2023 các kỹ năng liên quan đến AUTOSAR chiếm tới 23,67% tổng nhu cầu kỹ năng trong lĩnh vực này, đánh dấu mức tăng ấn tượng +113,0% so với năm trước đó.

Nhu cầu này càng được thể hiện rõ hơn khi FPT Software, top 1 công ty về số lượng nhân lực cũng đang có nhu cầu về mảng phần mềm nhúng ô tô.

“Theo ông Tạ Trần Minh, sự bùng nổ của các thiết bị điện tử trên xe hơi trong những năm gần đây đã mở ra triển vọng rất lớn cho ngành phát triển ứng dụng ô tô (Automotive) và làm tăng mạnh nhu cầu tuyển dụng kỹ sư Automotive. Tại FPT Software Global Automotive nhu cầu nhân lực tăng trung bình 30% mỗi năm, dự tính hơn 900 kỹ sư vào năm 2021.” [3]



Hình 1.2 Top công ty lớn nhất Việt Nam theo số lượng nhân lực.

1.1.3 Các thách thức

Hiện tại, ngành công nghiệp ô tô phụ thuộc rất nhiều vào các công cụ AUTOSAR RTE generator độc quyền, thường đi kèm với chi phí rất đắt đỏ. Rào cản tài chính này có thể hạn chế khả năng tiếp cận các công cụ này, cản trở tiến độ của các nhóm phát triển, công ty khởi nghiệp và tổ chức giáo dục nhỏ hơn. Hơn nữa, sự phức tạp của các giải pháp độc quyền hiện có có thể cản trở việc các nhà phát triển áp dụng chúng nếu không có chuyên môn sâu rộng về AUTOSAR [2].

1.1.4 Tối ưu điều kiện cho nhà phát triển

Sự cấp thiết phải phát triển trình tạo AUTOSAR RTE mã nguồn mở bắt nguồn từ nhu cầu truy cập vào công nghệ quan trọng này một cách tự do. Bằng cách cung cấp giải pháp miễn phí và dễ tiếp cận, các nhà phát triển trên nhiều nền tảng và tài nguyên khác nhau có thể khai thác sức mạnh của AUTOSAR mà không gặp trở ngại không cần thiết. Việc này có thể xúc tác cho sự đổi mới và thúc đẩy sự hợp tác trong cộng đồng phát triển phần mềm ô tô.

1.1.5 Thúc đẩy đổi mới

Chi phí cao và độ phức tạp liên quan đến các AUTOSAR RTE generator hiện tại đã vô tình cản trở sự đổi mới trong lĩnh vực này. Các nhà phát triển thường bị buộc phải phân bổ nguồn lực đáng kể chỉ để có được các công cụ cần thiết. Bằng cách giới thiệu một giải pháp thay thế và có mã nguồn mở, các nhà phát triển có thể chuyển hướng nỗ lực của họ sang việc khám phá những ý tưởng mới, cải tiến các quy trình hiện có và vượt qua ranh giới phát triển phần mềm ô tô.

1.1.6 Ý nghĩa giáo dục

Trong môi trường giáo dục, sự sẵn có của trình tạo AUTOSAR RTE nguồn mở có thể mang tính cải tiến đổi. Nó trao quyền cho sinh viên, nhà nghiên cứu và nhà giáo dục tham gia vào các công nghệ tiêu chuẩn ngành mà không phải chịu gánh nặng chi phí quá cao. Do đó, thế hệ kỹ sư ô tô tiếp theo có thể được chuẩn bị tốt hơn để giải quyết những thách thức của một ngành công nghiệp năng động và định hướng công nghệ.

1.1.7 Kết luận

Việc tạo và triển khai trình tạo AUTOSAR RTE nguồn mở bắt nguồn từ nhu cầu tự do truy cập và sử dụng công nghệ, thúc đẩy đổi mới và nâng cao giáo dục trong bối cảnh phát triển phần mềm ô tô. Bằng cách loại bỏ các rào cản tài chính và đơn giản hóa quy trình áp dụng, dự án này nỗ lực trao quyền cho các nhà phát triển, thúc đẩy hợp tác và mở đường cho một ngành công nghiệp ô tô toàn diện và sáng tạo hơn. Từ đó nguồn nhân lực chất lượng cao sẽ dồi dào hơn, đủ đáp ứng nhu cầu việc làm trong mảng phần mềm ô tô nói riêng và phần mềm nói chung.

1.2 Tổng quan về tình hình nghiên cứu hiện tại

1.2.1 Tổng quan về phần mềm nhúng ô tô

Ngành công nghiệp ô tô đang trải qua một sự chuyển đổi sâu sắc khi phần mềm trở thành một phần không thể thiếu đối với chức năng, sự an toàn và sự đổi mới của phương tiện. Từ hệ thống hỗ trợ người lái tiên tiến (ADAS) đến nền tảng thông tin giải trí và khả năng lái xe tự động, việc phát triển phần mềm luôn đi đầu trong việc định hình lại trải nghiệm lái xe. Phần tổng quan này đi sâu vào bối cảnh phát triển phần mềm ô tô hiện nay, nêu bật các xu hướng, thách thức và tiến bộ chính xác định ngành công nghiệp ngày nay.

Theo truyền thống được biết đến với kỹ thuật cơ khí, ngành công nghiệp ô tô đã phát triển thành một lĩnh vực định hướng công nghệ, trong đó phần mềm đóng vai trò then chốt. Các phương tiện hiện đại được trang bị một loạt bộ điều khiển điện tử (ECU) điều khiển nhiều chức năng khác nhau. Phần mềm hỗ trợ mọi thứ, từ hệ thống điều khiển động cơ và phanh cho đến các tính năng điều hướng, giải trí và liên lạc. Sự chuyển đổi này đã thúc đẩy sự phát triển phần mềm đi đầu trong đổi mới ô tô, tạo ra một mô hình mới trong đó mã cũng quan trọng như mã lực.

Phần mềm nhúng đã nổi lên như là xương sống của hệ thống ô tô, cho phép liên lạc và phối hợp liền mạch giữa các bộ phận khác nhau của xe. Các hệ thống nhúng này bao gồm nhiều chức năng khác nhau, từ hệ thống điều khiển thời gian thực quản lý hiệu suất động cơ đến các thuật toán phức tạp giúp tránh va chạm. Việc tích hợp các

hệ thống này đòi hỏi một cách tiếp cận toàn diện để phát triển phần mềm, nhấn mạnh vào độ tin cậy, bảo mật và khả năng tương tác.

Sự phức tạp ngày càng tăng của phần mềm ô tô đặt ra một thách thức đáng kể. Tính liên kết của các hệ thống, cùng với nhu cầu phản hồi theo thời gian thực, đòi hỏi các kiến trúc phần mềm phức tạp. Các kỹ sư phần mềm phải vật lộn với việc quản lý các cơ sở mã nguồn lớn, tối ưu hóa hiệu suất, đảm bảo an ninh mạng và tuân thủ các tiêu chuẩn an toàn nghiêm ngặt như ISO-14229. Việc cân bằng các nhu cầu này trong khi đáp ứng lịch trình sản xuất chặt chẽ đòi hỏi các quy trình và công cụ phát triển phức tạp.

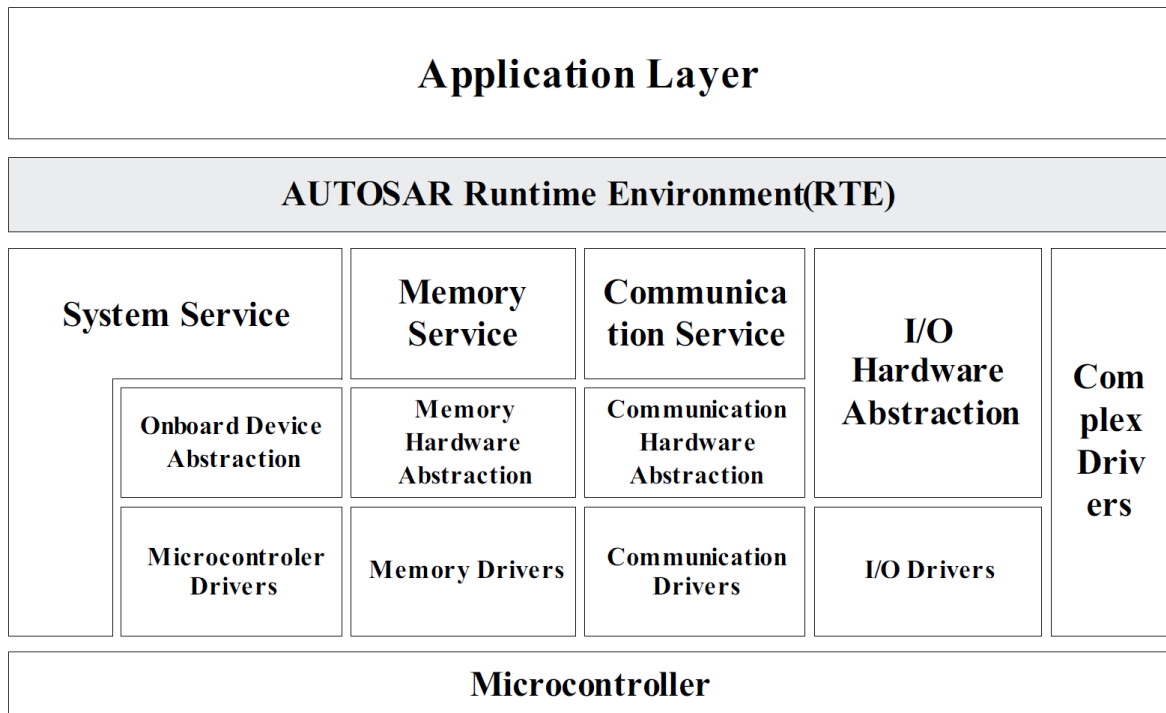
1.2.2 Tổng quan về tiêu chuẩn AUTOSAR

Để giải quyết thách thức phức tạp, ngành công nghiệp ô tô đã áp dụng các tiêu chuẩn như AUTOSAR (AUTomotive Open System ARchitecture). AUTOSAR cung cấp một khung tiêu chuẩn hóa nhằm thúc đẩy tính đóng gói, khả năng mở rộng và khả năng sử dụng lại trong phần mềm ô tô. Nó cho phép tách các thành phần phần mềm khỏi phần cứng, tạo điều kiện cộng tác hiệu quả giữa các bên liên quan khác nhau trong hệ sinh thái ô tô. Việc áp dụng AUTOSAR giúp hợp lý hóa quá trình phát triển, nâng cao khả năng tương tác và tăng tốc thời gian đưa ra thị trường các tính năng mới của xe.

Trong đó lớp RTE đóng vai trò cực kỳ quan trọng trong kiến trúc AUTOSAR. RTE có nhiệm vụ là trung tâm hỗ trợ liên lạc trong hệ thống. RTE tạo điều kiện thuận lợi cho việc trao đổi dữ liệu và tín hiệu giữa các thành phần phần mềm. RTE quản lý sự tương tác phức tạp của các chức năng phần mềm, cho phép chúng giao tiếp và hợp tác trong khi vẫn đảm bảo đáp ứng các ràng buộc về thời gian. Lớp này là minh chứng cho cam kết của AUTOSAR đối với việc phát triển phần mềm theo mô-đun, có thể mở rộng và có khả năng tương tác trong lĩnh vực ô tô.

Các chức năng chính của lớp RTE:

- Trao đổi dữ liệu: Lớp RTE điều phối luồng dữ liệu giữa các thành phần phần mềm, cung cấp cơ chế liên lạc liền mạch. Nó quản lý tính nhất quán của dữ liệu, đảm bảo thông tin được cập nhật và chính xác trên nhiều thành phần khác nhau.
- Ràng buộc về thời gian: Hệ thống thời gian thực yêu cầu thời gian chính xác. RTE thực thi các yêu cầu về thời gian bằng cách điều phối việc thực thi các chức năng phần mềm dựa trên lịch trình được xác định trước, ngăn ngừa sự chậm trễ và đảm bảo khả năng phản hồi.
- Quản lý chế độ: Hệ thống ô tô thường hoạt động ở các chế độ khác nhau, chẳng hạn như lái xe bình thường, chế độ lái tự động hoặc chế độ an toàn. Lớp RTE tạo điều kiện chuyển đổi liền mạch giữa các chế độ này bằng cách quản lý việc kích hoạt và hủy kích hoạt các thành phần phần mềm.
- Quản lý tài nguyên: Trong môi trường hạn chế về tài nguyên, RTE phân bổ và quản lý tài nguyên tính toán, ngăn ngừa xung đột và tối ưu hóa việc sử dụng sức mạnh xử lý.
- Quản lý truyền thông: Các tương tác phức tạp giữa các thành phần phần mềm đòi hỏi các cơ chế giao tiếp hiệu quả. Lớp RTE cho phép giao tiếp giữa các thành phần thông qua các giao diện được xác định rõ ràng, giảm độ phức tạp của các tương tác trực tiếp.



Hình 1.3 Kiến trúc AUTOSAR

1.2.3 Phân tích các nghiên cứu liên quan đến AUTOSAR RTE generator

1.2.3.1 Nghiên cứu trong nước

Hiện nay, trong nước vẫn chưa có bài nghiên cứu nào liên quan đến AUTOSAR RTE nói chung và AUTOSAR RTE generator nói riêng.

1.2.3.2 Nghiên cứu ngoài nước

Số lượng bài nghiên cứu ngoài nước liên quan đến AUTOSAR RTE generator hiện rất ít, trong đó nổi bật nhất là *Design and Implementation of RTE Generator for Automotive Embedded Software* được viết bởi Shiquan Piao, Hyunchul Jo, Sungho Jin và Wooyoung Jung [1]. Bài nghiên cứu này đưa ra được ý tưởng về các thành phần trong một AUTOSAR RTE generator bao gồm Generator engine, XML parser và RTE template modules, cũng như cách hoạt động của AUTOSAR RTE generator, khởi tạo các cấu hình cần thiết, phân tích tệp XML, kết hợp với mẫu RTE được định nghĩa trước đó để đưa ra kết quả cuối cùng. Nhưng về cách vận hành cụ thể chi tiết của AUTOSAR RTE generator vẫn chưa được đề cập rõ ràng. Đặc biệt là các phân tích và lấy dữ liệu từ tệp XML. Tệp mô tả AUTOSAR được cấu trúc dưới dạng

Extensible Markup Language (XML) vì vậy với các XML parse hiện tại ta chỉ có thể trích xuất dữ liệu bằng cách duyệt theo tên của tag (Hình 1.4). Trong khi đó dữ liệu của các đối tượng AUTOSAR lại được liên kết với nhau bằng nội dung của tag <SHORT-NAME> (Hình 1.5), điều này dẫn tới sự khó khăn và cồng kềnh nếu như chỉ phân tích tệp XML AUTOSAR bằng phương pháp DOM hay SAX thông thường [1].

```
<AR-PACKAGES>
  <AR-PACKAGE>
    <SHORT-NAME>RB</SHORT-NAME>
    <AR-PACKAGES>
      <AR-PACKAGE>
        <SHORT-NAME>PT</SHORT-NAME>
        <AR-PACKAGES>
          <AR-PACKAGE>
            <SHORT-NAME>PCT_DcmSample_Provider</SHORT-NAME>
            <AR-PACKAGES>
              <AR-PACKAGE>
                <SHORT-NAME>SwComponentTypes</SHORT-NAME>
                <ELEMENTS>
                  <APPLICATION-SW-COMPONENT-TYPE>
                    <SHORT-NAME>DcmSample_Provider</SHORT-NAME>
                    <PORTS>
                      <P-PORT-PROTOTYPE>
                        <SHORT-NAME>PP_DcmSample_Provider_PP1</SHORT-NAME>
                        <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">
                      </P-PORT-PROTOTYPE>
                      <R-PORT-PROTOTYPE>
                        <SHORT-NAME>RP_DcmSample_Provider_RP1</SHORT-NAME>
                        <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">
                      </R-PORT-PROTOTYPE>
                    </PORTS>
```

Hình 1.4 Ví dụ cấu trúc trong tệp XML AUTOSAR

```
<P-PORT-PROTOTYPE>
  <SHORT-NAME>PP_DcmSample_Provider_PP1</SHORT-NAME>
  <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">RB/PT/PCT_DcmSample_Provider/PortInterfaces/IF_DcmSample_Provider_IF1</PROVIDED-INTERFACE-TREF>
</P-PORT-PROTOTYPE>
<R-PORT-PROTOTYPE>
  <SHORT-NAME>RP_DcmSample_Provider_RP1</SHORT-NAME>
  <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">RB/PT/PCT_DcmSample_Provider/PortInterfaces/IF_DcmSample_Requester_IF1</REQUIRED-INTERFACE-TREF>
</R-PORT-PROTOTYPE>
```

Hình 1.5 Đường dẫn liên kết thông tin các đối tượng với nhau được xác định bằng nội dung của tag <SHORT-NAME>

Ngoài ra, bài nghiên cứu trên không cung cấp mã nguồn của AUTOSAR RTE generator, điều này làm giảm độ tin cậy của bài nghiên cứu.

Vì vậy trong nghiên cứu này - “*Thiết kế và triển khai công cụ mã nguồn mở RTE Generator cho phần mềm nhúng xe ô tô*”, một cách trích xuất dữ liệu mới sẽ được thiêu để lấy thông tin trong tệp XML AUTOSAR và liên kết chúng với nhau một

cách dễ dàng hơn, đồng thời cung cấp mã nguồn của AUTOSAR RTE generator được thiết kế và triển khai trong nghiên cứu này đồng thời giới thiệu demo ứng dụng sau khi kết quả của AUTOSAR RTE generator được triển khai trên nền tảng hệ điều hành thời gian thực FreeRTOS và vi điều khiển ESP32, nhằm tăng mức độ tin cậy và tính ứng dụng của bài nghiên cứu.

1.3 Nội dung nghiên cứu

- Nội dung 1: Cấu trúc tệp XML, cách thức trích xuất thông tin từ tệp XML.
- Nội dung 2: Cấu trúc tệp JSON, cách thức trích xuất thông tin từ tệp JSON.
- Nội dung 3: Phương pháp đệ quy và cách thức chuyển đổi cấu trúc thông tin từ tệp XML sang tệp JSON.
- Nội dung 4: Cách thức trích xuất thông tin từ tệp JSON để cung cấp cho quá trình sinh mã.
- Nội dung 5: Triển khai ứng dụng demo sau khi ứng AUTOSAR đơn giản sau khi sử dụng AUTOSAR RTE generator để sinh mã nguồn ứng dụng trên nền hệ điều hành thời gian thực FreeRTOS chạy trên vi điều khiển ESP32.

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về AUTOSAR

2.1.1 AUTOSAR là gì

AUTOSAR, viết tắt của AUTomotive Open System Architecture, là một phương pháp phát triển và kiến trúc phần mềm mở và tiêu chuẩn hóa được sử dụng trong ngành công nghiệp ô tô. Được phát triển như một nỗ lực hợp tác của các nhà sản xuất ô tô, nhà cung cấp và nhà phát triển công cụ lớn nhằm giải quyết sự phức tạp ngày càng tăng của phần mềm trên các phương tiện hiện đại.

Những điểm chính về AUTOSAR bao gồm:

Tiêu chuẩn hóa: AUTOSAR xác định một tập hợp các giao diện, giao thức và định dạng trao đổi dữ liệu được tiêu chuẩn hóa. Các tiêu chuẩn này nhằm mục đích cho phép phát triển phần mềm ô tô mang tính module hơn, có thể mở rộng và tái sử dụng.

Module hóa phần mềm: Một trong những nguyên tắc cốt lõi của AUTOSAR là module hóa phần mềm. Nó chia phần mềm ô tô thành các thành phần nhỏ hơn, dễ quản lý hơn, có thể được phát triển và thử nghiệm độc lập. Tính module này tăng cường khả năng sử dụng lại phần mềm và giảm thời gian phát triển.

Khả năng tương tác: AUTOSAR thúc đẩy khả năng tương tác giữa các thành phần phần mềm và hệ thống khác nhau trong xe. Nó đảm bảo rằng các mô-đun phần mềm từ nhiều nhà cung cấp khác nhau có thể hoạt động liền mạch với nhau, giảm thiểu các thách thức trong việc tích hợp.

Trừu tượng hóa ECU: AUTOSAR trừu tượng hóa phần cứng cơ bản (Bộ điều khiển điện tử hoặc ECU) khỏi ứng dụng phần mềm. Sự trừu tượng hóa này cho phép phần mềm được viết độc lập với phần cứng cụ thể mà nó sẽ chạy trên đó, giúp việc chuyển phần mềm giữa các mẫu xe khác nhau trở nên dễ dàng hơn.

Tiêu chuẩn truyền thông: AUTOSAR xác định các giao thức truyền thông và phần mềm trung gian cho các mạng liên lạc trên xe khác nhau. Tiêu chuẩn hóa này đơn giản hóa việc phát triển phần mềm giao tiếp qua các kiến trúc mạng khác nhau.

Hệ sinh thái công cụ: AUTOSAR đã dẫn đến sự phát triển của một hệ sinh thái quan trọng gồm các công cụ phần mềm hỗ trợ các khía cạnh khác nhau của quy trình phát triển phần mềm, bao gồm tạo mã, quản lý cấu hình và thử nghiệm.

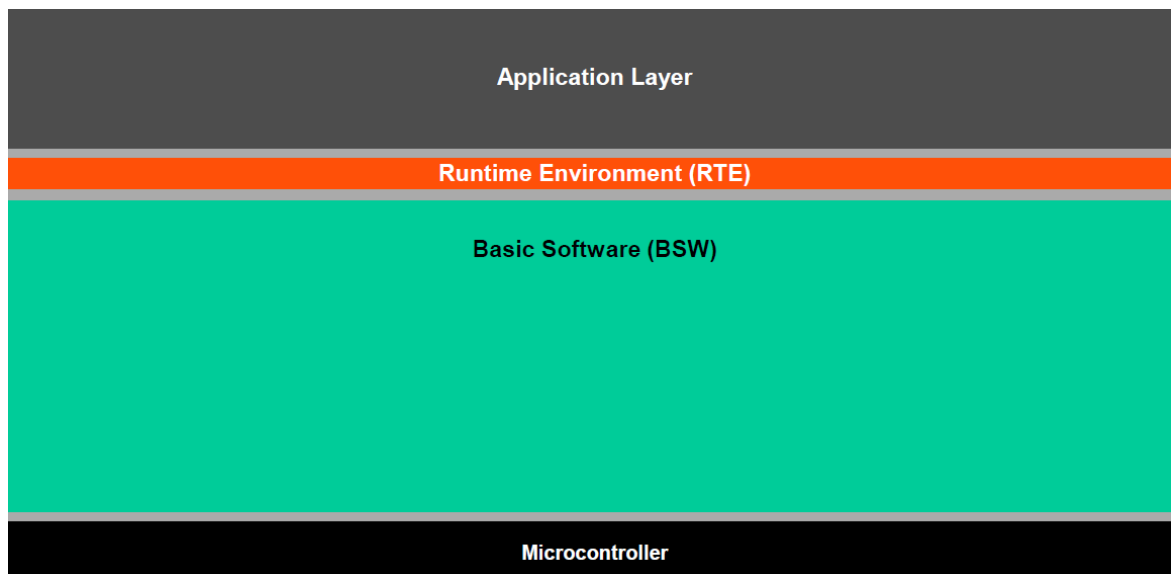
Tính linh hoạt: Trong khi cung cấp các tiêu chuẩn, AUTOSAR đủ linh hoạt để cho phép tùy chỉnh. Các nhà sản xuất và nhà cung cấp ô tô có thể điều chỉnh AUTOSAR để đáp ứng nhu cầu cụ thể của họ.

Hỗ trợ dài hạn: AUTOSAR nhằm mục đích cung cấp giải pháp lâu dài cho việc phát triển phần mềm ô tô. Điều này rất quan trọng trong một ngành mà phương tiện có vòng đời dài và phần mềm cần được bảo trì và cập nhật trong nhiều năm.

Tóm lại, AUTOSAR là một khuôn khổ tiêu chuẩn hóa nhằm cách mạng hóa việc phát triển phần mềm ô tô. Nó đơn giản hóa quá trình phát triển, tăng cường khả năng tương tác và cải thiện hiệu quả và độ an toàn của hệ thống ô tô. Nó đã trở thành công nghệ cơ bản trong các phương tiện hiện đại, giúp quản lý mức độ phức tạp ngày càng tăng của các tính năng và chức năng được điều khiển bằng phần mềm.

2.1.2 Kiến trúc AUTOSAR

Kiến trúc phần mềm được hiển thị trong Hình 2.1 bao gồm bốn lớp chính. Lớp dưới cùng là lớp Microcontroller. Hai lớp tiếp theo ở trên là lớp Basic Software và lớp RTE. Lớp trên cùng là lớp Application chứa tất cả phần mềm dành riêng cho ứng dụng (SWC) [4].



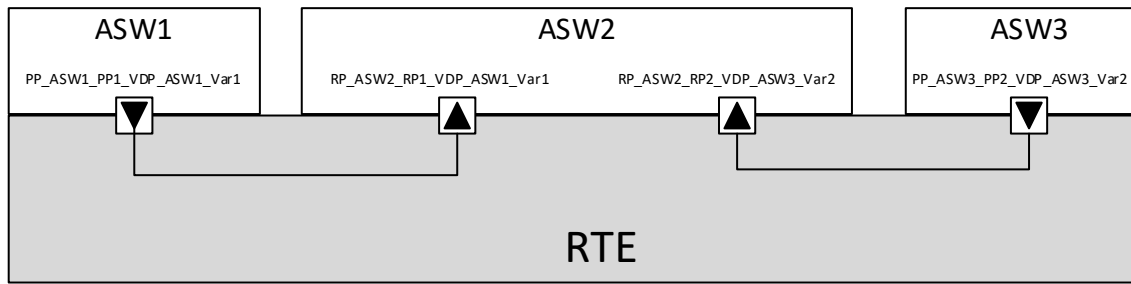
Hình 2.1 Kiến trúc phần mềm AUTOSAR

2.1.3 Lớp *Runtime Environment (RTE)*

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 [5].

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 [5].

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 2.2 Sự tương tác giữa các Application Software Component thông qua RTE

2.1.4 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 [5].

Có nhiều loại interface khác nhau được mô tả như trong Hình 2.3, 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.

• Sender Receiver Interface	P-Port	☑	R-Port	☒
• Client Server Interface	P-Port	☐	R-Port	☐
• NVData Interface	P-Port	☑	R-Port	☒
• Parameter Interface	P-Port	☑	R-Port	☒
• ModeSwitch Interface	P-Port	☒	R-Port	☒
• Trigger Interface	P-Port	☑	R-Port	☒

Hình 2.3 Ký hiệu các loại Interface

2.1.5 Application Software (ASW) và thành phần của ASW

ASW là các thành phần phần mềm ở lớp ứng dụng. Đây là các đơn vị module riêng lẻ đóng gói các chức năng hoặc dịch vụ cụ thể của xe. Ví dụ bao gồm module điều khiển động cơ, trình phát đa phương tiện hoặc hệ thống phát hiện va chạm, ...

Các thành phần phần mềm được thiết kế để có thể tái sử dụng, cho phép các nhà phát triển tạo ra các ứng dụng phức tạp bằng cách kết hợp và xác định cấu hình các thành phần hiện có.

Các thành phần cơ bản của một ASW bao gồm: Port, Runnable Entity, Event, DataTypeMapping

2.1.5.1 Port

Trong Autosar, SWC (Software Component) có thể trao đổi thông tin với nhau thông qua các cổng (port) [5]. 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.

2.1.5.2 *Runnable Entity*

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) [5].

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.

2.1.5.3 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[5].

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.

2.1.5.4 DataTypeMapping

DataTypeMapping là một cơ chế được sử dụng trong AUTOSAR để quản lý việc ánh xạ các loại dữ liệu giữa các thành phần phần mềm có thể có các cách trình bày khác nhau của cùng một dữ liệu. Nó đảm bảo rằng dữ liệu trao đổi giữa các thành phần này

được diễn giải và sử dụng chính xác, bất kể có bất kỳ sự khác biệt nào trong định nghĩa kiểu dữ liệu.

Các loại dữ liệu và các biến thể của chúng: Trong hệ thống phần mềm ô tô, các thành phần phần mềm khác nhau có thể sử dụng nhiều loại dữ liệu khác nhau để thể hiện thông tin tương tự hoặc giống hệt nhau. Ví dụ: một thành phần có thể xác định giá trị nhiệt độ dưới dạng số dấu phẩy động, trong khi thành phần khác có thể sử dụng số nguyên.

Định nghĩa `DataTypeMapping`: `DataTypeMapping` liên quan đến việc xác định ánh xạ hoặc chuyển đổi giữa các loại dữ liệu khác nhau này. Nó chỉ định cách chuyển đổi dữ liệu từ biểu diễn này sang biểu diễn khác khi nó được chia sẻ giữa các thành phần.

Tính nhất quán của dữ liệu: Bằng cách thiết lập `DataTypeMapping`, AUTOSAR đảm bảo rằng tính nhất quán của dữ liệu được duy trì trong toàn bộ hệ thống. Điều này rất quan trọng để đảm bảo rằng dữ liệu được thành phần nhận diễn giải và xử lý chính xác, bất kể cách trình bày dữ liệu của thành phần gửi.

Port Interface: `DataTypeMapping` thường được nhắc đến cùng với Port Interface trong AUTOSAR. Các Port xác định giao tiếp giữa các thành phần phần mềm và `DataTypeMapping` giúp đảm bảo rằng dữ liệu được trao đổi qua các cổng này tuân thủ cách hiểu chung về các loại dữ liệu.

Ví dụ về cách sử dụng: Hãy xem xét tình huống trong đó một thành phần tính toán nhiệt độ động cơ dưới dạng giá trị dấu phẩy động (ví dụ: 75,5°C), trong khi thành phần khác yêu cầu biểu diễn số nguyên (ví dụ: 755). `DataTypeMapping` sẽ xác định cách chuyển đổi dữ liệu giữa hai cách biểu diễn này, đảm bảo rằng dữ liệu vẫn chính xác và có ý nghĩa khi được truyền giữa các thành phần.

Lợi ích của `DataTypeMapping`:

- Khả năng tương tác: Nó tạo điều kiện cho khả năng tương tác giữa các thành phần phần mềm được phát triển bởi các nhóm hoặc nhà cung cấp khác nhau, mỗi thành phần sử dụng các loại dữ liệu ưa thích của mình.

- Tính toàn vẹn dữ liệu: DataTypeMapping đảm bảo tính toàn vẹn và nhất quán của dữ liệu bằng cách cung cấp một cách tiêu chuẩn hóa để xử lý các biến thể kiểu dữ liệu.
- Tính linh hoạt: Nó cho phép các nhà phát triển làm việc với các loại dữ liệu phù hợp nhất với chức năng cụ thể của họ trong khi vẫn cho phép giao tiếp liền mạch với các thành phần khác.
- Khả năng bảo trì: Khi cần thay đổi cách biểu diễn dữ liệu, DataTypeMapping có thể được điều chỉnh để phù hợp với những thay đổi này mà không ảnh hưởng đến tính tương thích của các thành phần hiện có.

Tóm lại, DataTypeMapping trong AUTOSAR ASW là một cơ chế quan trọng cho phép các thành phần phần mềm giao tiếp hiệu quả, ngay cả khi chúng sử dụng các cách biểu diễn kiểu dữ liệu khác nhau. Nó đảm bảo tính nhất quán, tính toàn vẹn và khả năng tương tác của dữ liệu trong các hệ thống phần mềm ô tô phức tạp, cuối cùng góp phần nâng cao độ tin cậy và chức năng của các phương tiện hiện đại.

2.2 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). 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 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 [7].

Các khái niệm cơ bản trong FreeRTOS bao gồm: Task, cấu trúc dữ liệu, scheduler.

2.2.1.1 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 [8].

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.

2.2.1.2 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 [8].
- 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 [8].
- 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 [8].
- 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 [8].
- 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 [8].

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.

2.2.1.3 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 [8].

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.

2.3 Tổng quan về dữ liệu dưới dạng XML và JSON

2.3.1 Dữ liệu dưới dạng XML

XML (Extensible Markup Language) là ngôn ngữ đánh dấu được sử dụng rộng rãi được thiết kế để lưu trữ và truyền tải dữ liệu. Nó vừa có thể đọc được bằng con người vừa có thể đọc được bằng máy, khiến nó trở thành một lựa chọn linh hoạt để biểu diễn dữ liệu có cấu trúc. XML sử dụng các tag để xác định các thành phần và mối quan hệ phân cấp của chúng, khiến nó trở nên lý tưởng cho việc tổ chức và trao đổi dữ liệu theo định dạng có cấu trúc [9].

Trong Python, thư viện lxml cung cấp một cách mạnh mẽ và hiệu quả để làm việc với các tài liệu XML. Trong đó lxml là thư viện của bên thứ ba mở rộng khả năng xử lý XML tích hợp của Python. Thư viện này được sử dụng rộng rãi trong cộng đồng Python để phân tích cú pháp, tạo và thao tác các tài liệu XML nhờ tốc độ và API giàu tính năng [10].

Phương pháp mà thư viện lxml dùng để phân tích cú pháp XML là DOM (Document Object Model).

```
<SENDER-RECEIVER-INTERFACE>
  <SHORT-NAME>IF_ASW1_IF1</SHORT-NAME>
  <IS-SERVICE>false</IS-SERVICE>
  <DATA-ELEMENTS>
    <VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>VDP_ASW1_Var1</SHORT-NAME>
      <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">../RB/PT/PCT_ASW1/ApplicationDataTypes/ADT_ASW1_uint32</TYPE-TREF>
    </VARIABLE-DATA-PROTOTYPE>
  </DATA-ELEMENTS>
</SENDER-RECEIVER-INTERFACE>
```

Hình 2.4 AUTOSAR Interface được biểu diễn dưới dạng XML

2.3.2 Dữ liệu dưới dạng JSON

JSON (JavaScript Object Notation) là một định dạng trao đổi dữ liệu nhẹ, dễ đọc và dễ viết cho con người, đồng thời dễ dàng cho máy phân tích và tạo. Nó được sử dụng

rộng rãi cho các tệp cấu hình và tuần tự hóa dữ liệu. JSON biểu thị dữ liệu dưới dạng cặp key-value (khóa-giá trị), trong đó kiểu dữ liệu của key là string và của value có thể là string, number, object, array, boolean hoặc null [11].

Ví dụ như trong Hình 2.5 Dữ liệu dưới dạng JSON, key “ASW” sẽ có value là một object, key “object_path” có value là một string.

Trong JSON recursively indexing đề cập đến quá trình truy cập các giá trị lồng nhau hoặc được nhúng sâu trong cấu trúc JSON bằng cách chỉ định một chuỗi các khóa hoặc chỉ mục dẫn đến giá trị mong muốn. JSON có bản chất phân cấp, bao gồm các đối tượng và mảng lồng nhau, đồng thời việc lập chỉ mục đệ quy cho phép bạn duyệt qua các cấu trúc lồng nhau này để truy xuất các điểm dữ liệu cụ thể.

Ví dụ như ở Hình 2.5 giá trị của key “PCT_ASW1” có thể được index như sau [“RB”][“PT”][“PCT_ASW1”]. Các index value như thế này rất phù hợp để trích xuất dữ liệu thông qua đường dẫn tham chiếu AUTOSAR dễ dàng hơn.

```
"RB": {  
  "object_path": "/RB",  
  "object_type": "AR-PACKAGE",  
  "outer_element_tag": "/AR-PACKAGES",  
  "PT": {  
    "object_path": "/RB/PT",  
    "object_type": "AR-PACKAGE",  
    "outer_element_tag": "/AR-PACKAGES",  
    "PCT_ASW1": {  
      "object_path": "/RB/PT/PCT_ASW1",  
      "object_type": "AR-PACKAGE",  
      "outer_element_tag": "/AR-PACKAGES",  
      "SwComponentTypes": {  
        "object_path": "/RB/PT/PCT_ASW1/SwComponentTypes",  
        "object_type": "AR-PACKAGE",  
        "outer_element_tag": "/AR-PACKAGES",  
        "ASW1": {  
          "object_path": "/RB/PT/PCT_ASW1/SwComponentTypes/ASW1",  
          "object_type": "APPLICATION-SW-COMPONENT-TYPE",  
          "outer_element_tag": "/ELEMENTS",  
          "PP_ASW1_PP1": {  
            "object_path": "/RB/PT/PCT_ASW1/SwComponentTypes/ASW1/PP_ASW1_PP1",  
            "object_type": "P-PORT-PROTOTYPE",  
            "outer_element_tag": "/PORTS",  

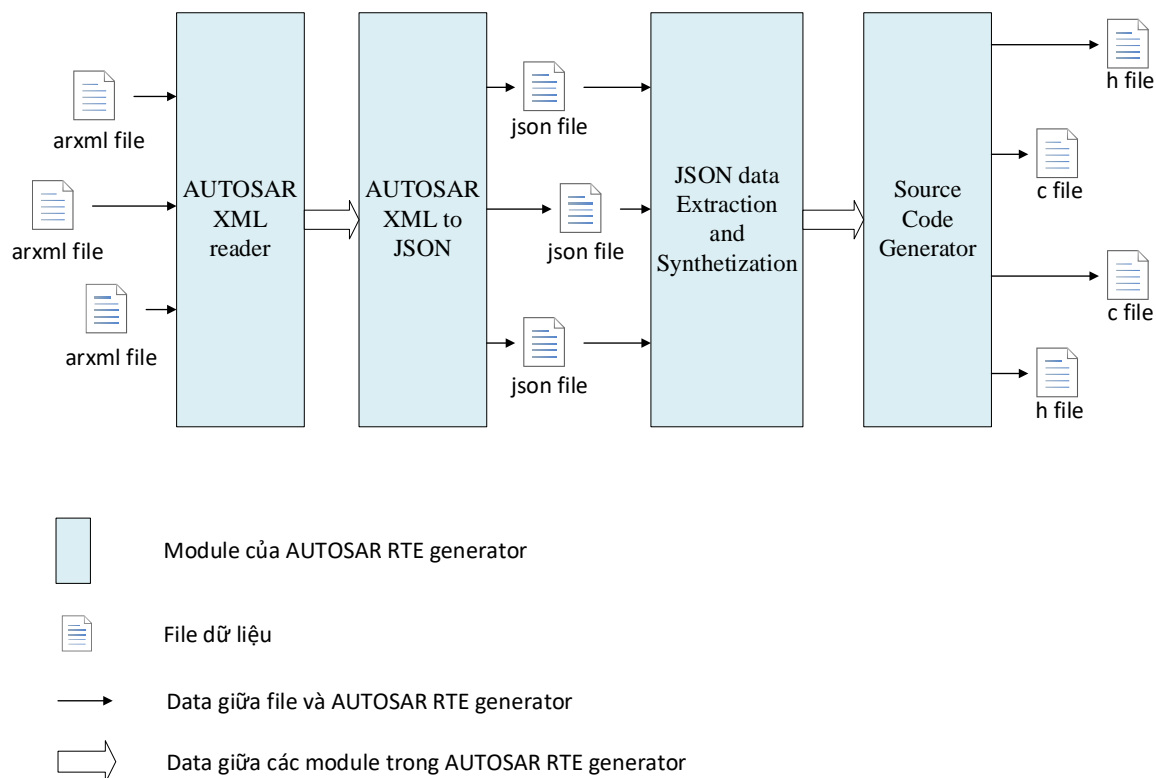
```

Hình 2.5 Dữ liệu dưới dạng JSON

2.4 Cấu trúc AUTOSAR RTE generator

Cấu trúc của AUTOSAR RTE generator được đề xuất trong nghiên cứu này bao gồm: module đọc và xử lý dữ liệu dạng AUTOSAR XML (AUTOSAR XML reader – AXR), module chuyển đổi dữ liệu AUTOSAR XML sang JSON (AUTOSAR XML to JSON - AXJ), module trích xuất và tổng hợp dữ liệu từ dạng JSON (JSON data Extraction and Synthesis - JES), module sinh mã từ dữ liệu được trích xuất (Source Code Generator - SCG).

- AXR có nhiệm vụ tìm đọc các tệp có định dạng AUTOSAR XML (*.arxml) trong folder được xác định. Đồng thời cung cấp API để trích xuất dữ liệu từ tệp arxml.
- AXJ sử dụng các API của AXR để trích xuất và phân tích dữ liệu từ tệp arxml. Chuyển đổi định dạng XML sang JSON dựa trên phương pháp để quy để đảm bảo tính phân cấp của dữ liệu, đồng thời tạo điều kiện để trích xuất dữ liệu dựa trên đường dẫn tham chiếu một cách dễ dàng hơn.
- JES đảm nhận nhiệm vụ thu thập dữ liệu dạng JSON, xử lý các đường dẫn tham chiếu AUTOSAR và tổng hợp những dữ liệu cần thiết cho việc sinh mã.
- SCG thu thập dữ liệu từ JES và tiến hành sinh mã.



Hình 2.6 Cấu trúc AUTOSAR RTE generator

2.4.1 Đặc tính kỹ thuật cần quan tâm khi thiết kế công cụ AUTOSAR RTE generator

Khi thiết kế và triển khai AUTOSAR RTE generator, có những thông tin cần quan tâm như sau:

Khả năng tương thích phiên bản AUTOSAR: AUTOSAR RTE Generator được thiết kế để tương thích với AUTOSAR phiên bản 4.2.2. Điều này đảm bảo sự phù hợp với các tiêu chuẩn ngành và hỗ trợ khả năng tương tác với nhiều hệ thống và linh kiện ô tô. Đồng thời nhà phát triển sẽ có những lựa chọn hợp lý hơn khi phát triển ứng dụng.

Môi trường phát triển: Trình tạo RTE đang được phát triển bằng Python 3.9.2. Ngôn ngữ lập trình hiện đại này cung cấp tính linh hoạt và khả năng cần thiết để phát triển phần mềm mạnh mẽ, đảm bảo việc tạo mã và tích hợp công cụ hiệu quả.

Hạn chế về tính năng: Dự án AUTOSAR RTE Generator thừa nhận phạm vi rộng lớn và độ phức tạp của tiêu chuẩn AUTOSAR. Để đạt được tính thực tế và tập trung nỗ

lực một cách hiệu quả, một số hữu hạn các tính năng được hỗ trợ trong phạm vi nghiên cứu này bao gồm:

- **Sender-Receiver Interface:** Sender-Receiver Interface là loại interface phổ biến được sử dụng nhiều cho sự tương tác giữa các ASW với nhau vì tính đơn giản và thuận tiện.
- **Data Receive Point By Values:** Để một Runnable Entity trong ASW có thể lấy dữ liệu từ bên ngoài, cần thiết phải cấu hình các điểm nhận dữ liệu cho Runnable Entity, để thuận tiện khi sử dụng và dễ tương thích với Sender-Receiver Interface được giới thiệu trước đó, Data Receive Point By Values là sự lựa chọn phù hợp để triển khai trong nghiên cứu này.
- **Data Send Point:** Ngoài việc nhận dữ liệu thì Runnable Entity cũng cần được cấu hình điểm để gửi dữ liệu ra bên ngoài, và để thuận tiện trong quá trình triển khai ứng dụng, Data Send Point được chọn để hỗ trợ triển khai nghiên cứu này.
- **Runnable Entities with Timing Events:** Trong chuẩn AUTOSAR, Runnable Entities có thể được liên kết tới nhiều kiểu sự kiện khác nhau như: Timing Event, Init Event, Background Event, Operation Invoked Event, External Trigger Occurred Event, Data Received Event, Swc Mode Switch Event,... Nhưng trong nghiên cứu này chỉ tập trung vào Timing Event để có thể triển khai được những tác vụ chạy liên tục theo chu kỳ.
- **Port Assembly Connector:** Để các ASW có thể tương tác được với nhau, cần có những cấu hình về kết nối các Port của ASW và Assembly Connector được chọn để triển khai trong nghiên cứu này.
- **Data type:** Trong một ứng dụng thực tế, có rất nhiều kiểu dữ liệu để biểu diễn giá trị của một đối tượng, các kiểu dữ liệu được phân loại thành hai nhóm chính là Application DataType đại diện cho đại lượng vật lý của các đối tượng như khối lượng (kg, g), độ dài (km, m), ... mỗi Application DataType sẽ được liên kết tới một kiểu dữ liệu có thể sử dụng trong ngôn ngữ lập trình được gọi là Implementation DataType, trong nghiên cứu này Implementation DataType

được giới hạn trong ba loại kiểu dữ liệu là uint8 (kiểu không dấu 8 bit), uint16 (kiểu không dấu 16 bit) và uint32 (kiểu không dấu 32 bit).

2.4.2 Phương pháp kiểm tra tính đúng đắn trên phần cứng

Để kiểm tra tính đúng đắn của mã nguồn sau khi được sinh ra từ AUTOSAR RTE generator, ESP32 được chọn làm vi điều khiển thí nghiệm. Một ứng dụng demo đơn giản gồm 3 ASW được tiết kế với từng chức năng khác nhau:

- ASW1: Đóng vai trò là webserver nhận tín hiệu điều từ giao diện web của người dùng, gửi tín hiệu từ người dùng đến ASW để điều khiển động cơ xe.
- ASW2: Thực hiện việc nhận tín hiệu từ ASW1 để điều khiển động cơ, đồng thời kiểm tra tín hiệu từ ASW3 ra quyết định cho phép động cơ hoạt động hay không.
- ASW3: Đọc dữ liệu khoảng cách từ cảm biến, xử lý và cung cấp tín hiệu khoảng cách tới ASW2.

Chi tiết về Testcase để kiểm chứng chức năng hoạt động của ứng dụng demo này sẽ được trình bày kỹ hơn ở chương 4.

2.5 Độ quy trong phát triển phần mềm

Độ quy trong phát triển phần mềm là một mô hình lập trình mạnh mẽ và tinh tế, trong đó một hàm tự gọi chính nó để giải quyết một vấn đề. Đó là một khái niệm có nguồn gốc sâu xa trong toán học và khoa học máy tính, đồng thời nó cung cấp một cách linh hoạt và hiệu quả để giải quyết các vấn đề phức tạp bằng cách chia chúng thành các vấn đề con đơn giản hơn, dễ quản lý hơn [12].

Đặc điểm chính của độ quy:

- Chia để trị: Độ quy tuân theo chiến lược "chia để trị", trong đó một bài toán lớn được chia thành các bài toán con nhỏ hơn, dễ giải quyết hơn. Mỗi bài toán con được giải một cách độc lập và các lời giải của chúng được kết hợp để giải quyết bài toán lớn hơn.

- Trường hợp cơ sở: Hàm đệ quy bao gồm trường hợp cơ sở, đóng vai trò là điều kiện kết thúc. Khi đáp ứng trường hợp cơ sở, hàm sẽ ngừng gọi chính nó và bắt đầu trả về giá trị. Điều này ngăn chặn đệ quy vô hạn.
- Trường hợp đệ quy: Hàm đệ quy cũng bao gồm trường hợp đệ quy, trong đó nó tự gọi chính nó với các đối số được sửa đổi để giải quyết một phiên bản nhỏ hơn hoặc đơn giản hơn của vấn đề ban đầu. Mỗi lệnh gọi đệ quy sẽ tiến gần hơn đến trường hợp cơ sở.

Ứng dụng đệ quy: Đệ quy tìm thấy các ứng dụng trong các khía cạnh khác nhau của phát triển phần mềm:

- Thiết kế thuật toán: Nhiều thuật toán cổ điển, như sắp xếp nhanh và sắp xếp hợp nhất để sắp xếp cũng như các thuật toán duyệt cây, được xác định theo cách đệ quy. Đệ quy đơn giản hóa việc thực hiện và hiểu biết về thuật toán.
- Cấu trúc dữ liệu: Đệ quy được sử dụng trong thiết kế và thao tác các cấu trúc dữ liệu như danh sách liên kết, cây nhị phân và biểu đồ. Ví dụ, các hàm đệ quy có thể duyệt cấu trúc cây một cách hiệu quả.
- Giải quyết vấn đề: Các vấn đề phức tạp, chẳng hạn như giải mê cung, che phủ bàn cờ và câu đố Tháp Hà Nội, thường được tiếp cận theo cách đệ quy, vì việc chia chúng thành các vấn đề con nhỏ hơn, tương tự sẽ đơn giản hóa lời giải.
- Lập trình hàm: Các ngôn ngữ lập trình hàm phụ thuộc rất nhiều vào đệ quy do chúng ưu tiên tính bất biến và các lệnh gọi hàm đệ quy.

Lợi ích của đệ quy:

- Rõ ràng và dễ đọc: Các giải pháp đệ quy thường phản ánh cấu trúc tự nhiên của vấn đề, làm cho mã trở nên trực quan và dễ hiểu hơn.
- Tính module: Các hàm đệ quy vốn có tính module. Chúng giải quyết các bài toán con nhỏ hơn một cách độc lập, thúc đẩy khả năng sử dụng lại mã.
- Hiệu quả: Trong một số trường hợp, đệ quy có thể mang lại giải pháp hiệu quả hơn so với phương pháp lặp. Tuy nhiên, điều cần thiết là phải tối ưu hóa hiệu quả và xử lý các vấn đề tràn ngăn xếp trong một số trường hợp nhất định.

Thách thức đệ quy:

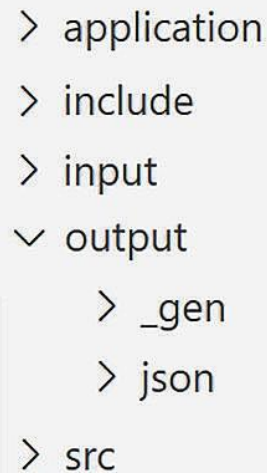
- Độ phức tạp về không gian: Lời gọi đệ quy có thể tiêu tốn bộ nhớ đáng kể.
- Gỡ lỗi: Mã đệ quy có thể khó gỡ lỗi do có nhiều lệnh gọi hàm lồng nhau. Việc theo dõi và giám sát đúng các lời gọi hàm là rất quan trọng.

Tóm lại, đệ quy là một kỹ thuật cơ bản trong phát triển phần mềm giúp trao quyền cho các nhà phát triển giải quyết các vấn đề phức tạp một cách hiệu quả và tinh tế bằng cách chia chúng thành các phần nhỏ hơn, dễ quản lý hơn. Khi được sử dụng một cách thích hợp, đệ quy có thể nâng cao tính rõ ràng của mã, tính module hóa và khả năng giải quyết vấn đề.

CHƯƠNG 3 TRIỂN KHAI CÔNG CỤ SINH MÃ NGUỒN RTE

3.1 Cấu trúc đường dẫn của các tệp

AUTOSAR RTE Generator sắp xếp hợp lý việc tổ chức các tệp đầu vào để nâng cao tính rõ ràng và hiệu quả trong quá trình phát triển phần mềm.



```
> application
> include
> input
▼ output
  > _gen
  > json
> src
```

Hình 3.1 Cấu trúc đường dẫn

3.1.1 Thư mục "*application*"

Tệp arxml: Các tệp arxml cốt lõi được đặt ở đây, đóng gói thông tin cần thiết cho ứng dụng. Những tệp này xác định kiến trúc phần mềm ô tô, bao gồm các thành phần, giao diện, kết nối giao tiếp, ...

Tệp mã nguồn: Trong thư mục này, các tệp nguồn C++ thích hợp cho ứng dụng sẽ được bao gồm. Những tệp này chứa logic phần mềm điều khiển các chức năng và tính năng khác nhau của xe.

3.1.2 Sắp xếp lại tệp

Để đảm bảo tích hợp liền mạch vào cấu trúc dự án C++, các tệp đầu vào trong thư mục "*application*" phải được sắp xếp lại một cách tỉ mỉ.

3.1.2.1 Thư mục "input"

Thư mục này đóng vai trò là kho lưu trữ cho các tệp arxml, lưu trữ các bản sao của các tệp arxml từ thư mục “application”. Những tệp này đóng vai trò là nền tảng cho quá trình tạo mã.

3.1.2.2 Thư mục "include"

Các tệp header được sao chép một cách có hệ thống vào thư mục "include". Các tệp header này đóng vai trò then chốt trong việc xác định giao diện và khai báo cho các thành phần phần mềm.

3.1.2.3 Thư mục "src"

Mã nguồn, bao gồm các tệp mã C++, có thể tìm thấy vị trí của tệp mã nguồn trong thư mục "src". Các tệp này chứa chi tiết triển khai của các thành phần phần mềm.

3.1.3 Thư mục “output”

Trong thư mục “output” bao gồm hai thư mục con “json” và “_gen” dùng để chứa các tệp thông tin xác định.

Thư mục con "json": Trong thư mục con này, các tệp được tạo để thể hiện thông tin đã xử lý từ các tệp arxml sang định dạng json. Các tệp json này cung cấp chế độ xem có cấu trúc về kiến trúc phần mềm ô tô, giúp các nhà phát triển truy cập và thao tác dữ liệu dễ dàng hơn.

Thư mục con "_gen": chứa đầu ra của quá trình tạo mã. Ở đây, các tệp mã nguồn được tạo dựa trên thông tin được trích xuất từ các tệp json. Các tệp mã nguồn và header được tạo này đã sẵn sàng để đưa vào dự án C++.

Quy trình phát triển

Cấu trúc thư mục này được thiết kế để tạo điều kiện thuận lợi cho quy trình làm việc thân thiện với nhà phát triển. Các nhà phát triển có thể phát triển ứng dụng của họ một cách thuận tiện trong thư mục "application" và bắt đầu quá trình tạo RTE. Sau

đó, các tệp nguồn và tiêu đề được tạo sẽ được sắp xếp tự động để dễ dàng tích hợp vào dự án C++.

Bằng cách áp dụng cách tiếp cận có cấu trúc này, AUTOSAR RTE generator nâng cao quá trình phát triển, hợp lý hóa việc quản lý tệp đầu vào và đầu ra đồng thời đảm bảo tích hợp liền mạch vào các dự án C++.

3.2 Triển khai các thành phần của AUTOSAR RTE generator

3.2.1 AUTOSAR XML reader – AXR

Trong AUTOSAR RTE generator, có một cách tiếp cận hiệu quả để xử lý các tệp XML AUTOSAR. Thay vì phát triển AXR tùy chỉnh ngay từ đầu, các khả năng của thư viện lxml/etree đã được tận dụng. Quyết định này bắt nguồn từ việc theo đuổi cả hiệu quả về thời gian và tài nguyên. Thư viện lxml/etree, nổi tiếng với khả năng phân tích cú pháp XML mạnh mẽ, cung cấp giải pháp đáng tin cậy và được thiết lập tốt, phù hợp liền mạch với các mục tiêu dự án, đảm bảo trích xuất dữ liệu nhanh chóng và chính xác từ các tệp XML AUTOSAR, đồng thời tiết kiệm thời gian phát triển quý giá. Cách tiếp cận thực tế này cho phép phân bổ nhiều tài nguyên hơn để nâng cao các khía cạnh quan trọng khác của AUTOSAR RTE generator, cuối cùng là hợp lý hóa quy trình phát triển phần mềm.

3.2.2 AUTOSAR XML to JSON – AXJ

Trong AUTOSAR RTE generator này, AXJ đóng vai trò là thành phần chính dành riêng cho việc chuyển đổi liền mạch dữ liệu XML AUTOSAR phức tạp thành định dạng JSON có cấu trúc dễ dàng cho việc đọc và xử lý về sau. AXJ được thiết kế dựa trên phương pháp đệ quy, cách này không chỉ đảm bảo chuyển đổi dữ liệu toàn diện mà còn giúp giảm đáng kể kích thước và độ phức tạp của chương trình AUTOSAR RTE generator.

Cụ thể AXJ sử dụng phương pháp đệ quy để điều hướng các lớp dữ liệu XML AUTOSAR với độ chính xác cao. Cách tiếp cận đệ quy này đảm bảo rằng mỗi đoạn thông tin, bất kể vị trí của nó trong hệ thống phân cấp XML, đều được chuyển đổi

một cách tỉ mỉ và toàn diện thành dạng biểu diễn JSON. Sức mạnh của đệ quy không chỉ nằm ở tính kỹ lưỡng mà còn ở khả năng tối ưu hóa đáng kể quá trình tạo.

Ưu điểm đáng chú ý của phương pháp đệ quy của AXJ là giảm đáng kể kích thước và độ phức tạp của chương trình AUTOSAR RTE generator. Việc đơn giản hóa này không chỉ nâng cao khả năng quản lý và bảo trì mã mà còn khuếch đại đáng kể hiệu suất và hiệu quả tổng thể của AUTOSAR RTE generator.

Quá trình chuyển đổi của AXJ giới thiệu một mô hình tổ chức dữ liệu khác tập trung vào SHORT-NAME của đối tượng. Từ các hệ thống phân cấp XML phức tạp và lồng nhau mang lại kết quả đầu ra JSON ưu tiên sự rõ ràng và dễ trích xuất dữ liệu bằng đường dẫn AUTOSAR. Dữ liệu được cấu trúc tỉ mỉ để trao quyền cho các nhà phát triển truy cập, thao tác và truy vấn thông tin dựa trên tên SHORT-NAME của đối tượng. Sự thay đổi quan trọng này không chỉ đơn giản hóa việc xử lý dữ liệu mà còn nâng cao tính linh hoạt của dự án.

Chi tiết các hoạt động của AXJ được mô tả ở Hình 3.2. Hàm thực hiện chức năng của AXJ được đặt tên là `genJsonFromXml` với hai đối số chính là `xml_ele` và `json_obj`.

Hàm `genJsonFromXml` có chức năng như sau:

- Chuyển đổi XML sang JSON: Chuyển đổi dữ liệu XML AUTOSAR phức tạp thành định dạng JSON có cấu trúc. Quá trình chuyển đổi bao gồm việc duyệt qua hệ thống phân cấp XML và các phần tử ánh xạ tới các phần tử JSON tương ứng của chúng.
- Tổ chức dữ liệu: Tập lệnh sắp xếp lại dữ liệu dựa trên SHORT-NAME của đối tượng, tạo cấu trúc JSON cho phép các nhà phát triển truy cập, thao tác và truy vấn thông tin một cách hiệu quả.
- Tối ưu hóa mã: Nó sử dụng đệ quy để hợp lý hóa cơ sở mã, giảm kích thước và độ phức tạp của mã để nâng cao khả năng bảo trì.

Các tham số chính:

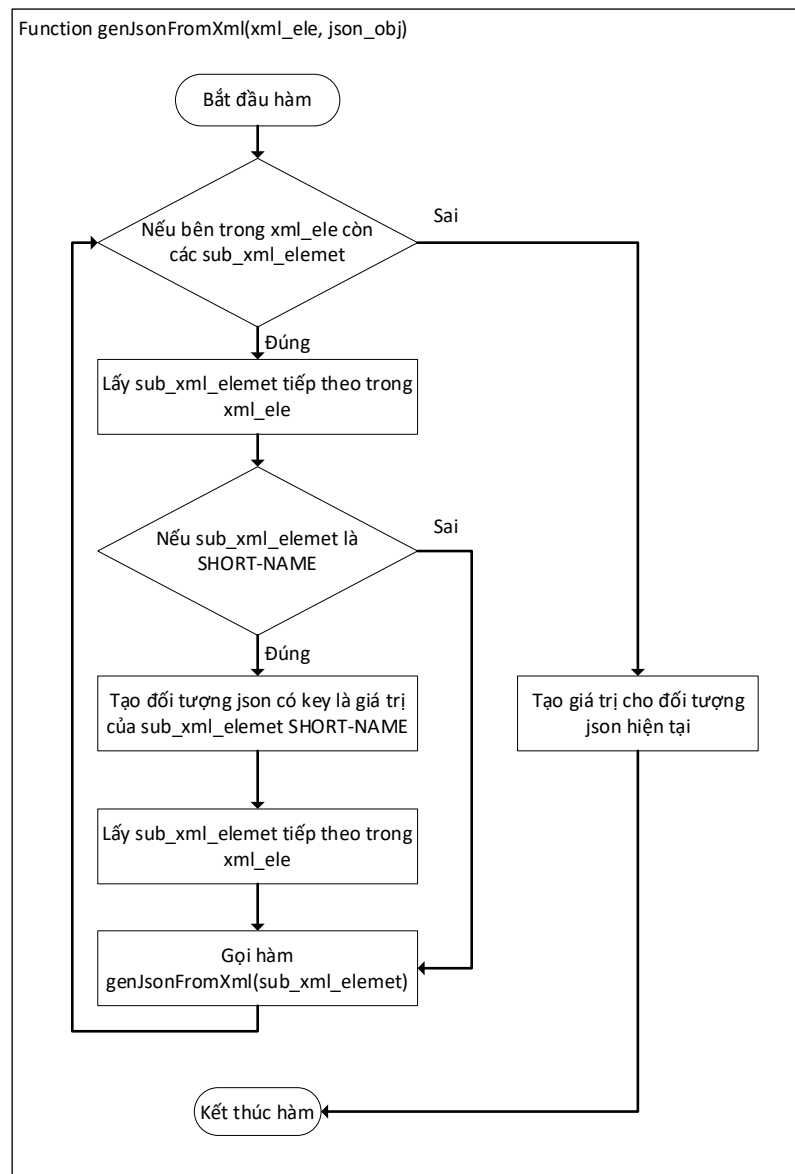
- `xml_ele`: Biểu thị phần tử XML hiện tại đang được xử lý.

- json_obj: Biểu thị đối tượng JSON đang được xây dựng.

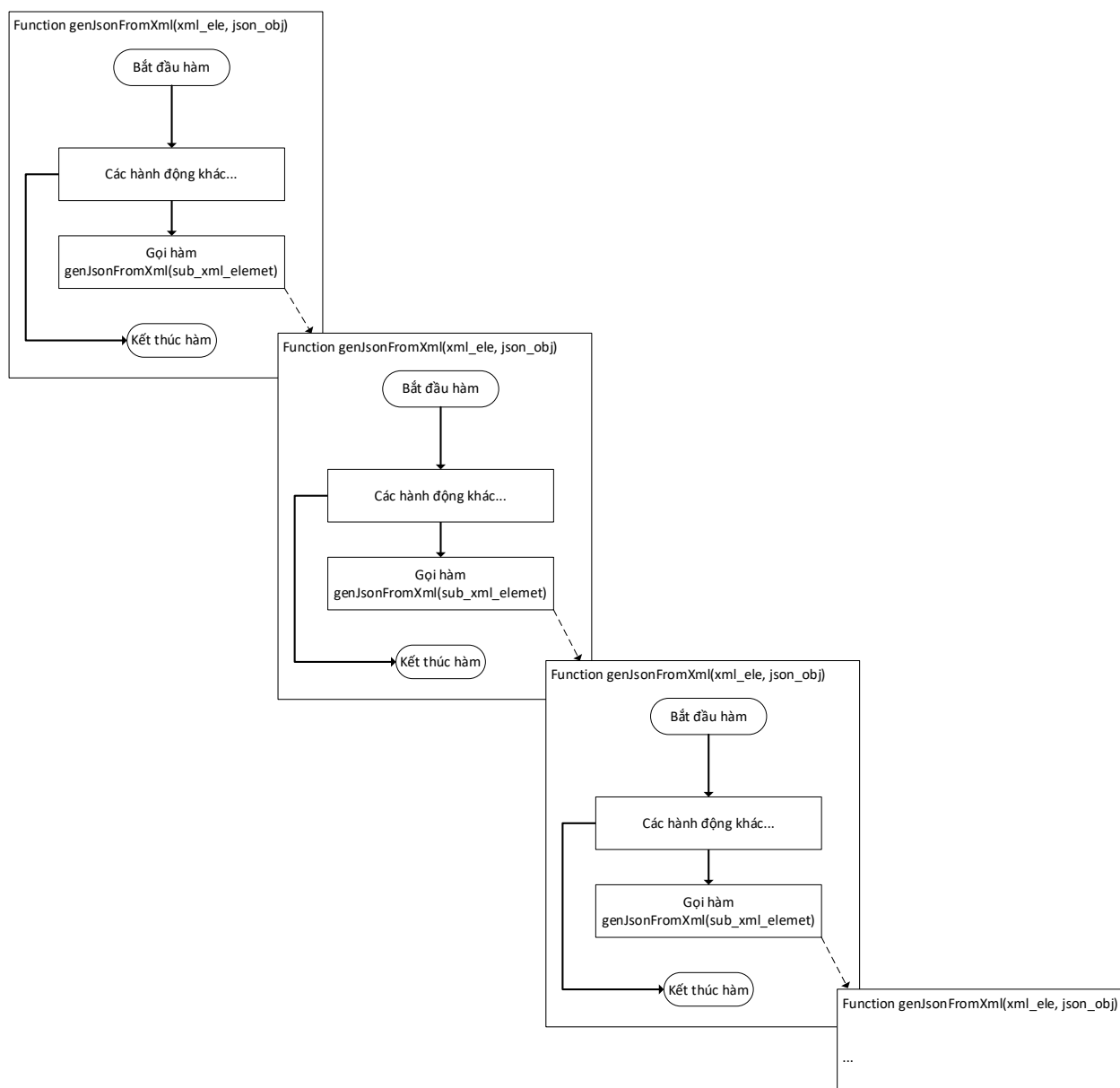
Chi tiết cách hoạt động:

- Xử lý thành phần xml: Tập lệnh xử lý các thành phần xml và sử dụng SHORT-NAME của đối tượng để sắp xếp dữ liệu JSON.
- Xử lý đệ quy: Sử dụng đệ quy để điều hướng hệ thống phân cấp XML, xử lý các phần tử con và xây dựng cấu trúc JSON lặp đi lặp lại.

Cách genJsonFromXml gọi lồng nhau được thể hiện ở Hình 3.3.



Hình 3.2 Chi tiết cách hoạt động của AXJ



Hình 3.3 Cách hàm genJsonFromXml được gọi lồng nhau

Sau quá trình chuyển đổi thì dữ liệu XML như được minh họa ở Hình 3.4 sẽ được chuyển thành dạng JSON như hình Hình 3.5.

Việc tích hợp AXJ vào AUTOSAR RTE generator có ý nghĩa sâu rộng đối với quy trình phát triển. Với cách tiếp cận đệ quy, mã nguồn của AUTOSAR RTE generator được tối ưu hóa với hiệu quả vượt trội. Điều này cho phép các nhà phát triển làm việc với dữ liệu AUTOSAR một cách trực quan và hiệu quả, cuối cùng là đẩy nhanh quá

trình phát triển phần mềm trong khi vẫn duy trì các tiêu chuẩn cao nhất về tính toàn vẹn dữ liệu.

```
<AR-PACKAGES>..
<AR-PACKAGE>..
  <SHORT-NAME>PCT_ASW1</SHORT-NAME>
  <AR-PACKAGES>..
    <AR-PACKAGE>..
      <SHORT-NAME>SwComponentTypes</SHORT-NAME>
      <ELEMENTS>..
        <APPLICATION-SW-COMPONENT-TYPE>..
          <SHORT-NAME>ASW1</SHORT-NAME>
          <PORTS>..
            <P-PORT-PROTOTYPE>..
              <SHORT-NAME>PP_ASW1_PP1</SHORT-NAME>
              <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">..RB/PT/PCT_ASW1/PortInterfaces/IF_ASW1_IF1</PROVIDE
            </P-PORT-PROTOTYPE>
            <R-PORT-PROTOTYPE>..
              <SHORT-NAME>RP_ASW1_RP1</SHORT-NAME>
              <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">..RB/PT/PCT_ASW1/PortInterfaces/IF_ASW2_IF1</REQUIRE
            </R-PORT-PROTOTYPE>
```

Hình 3.4 Cấu trúc dữ liệu dạng XML ban đầu

```
"PT": {
  "object_path": "/RB/PT",
  "object_type": "AR-PACKAGE",
  "outer_element_tag": "/AR-PACKAGES",
  "PCT_ASW1": {
    "object_path": "/RB/PT/PCT_ASW1",
    "object_type": "AR-PACKAGE",
    "outer_element_tag": "/AR-PACKAGES",
    "SwComponentTypes": {
      "object_path": "/RB/PT/PCT_ASW1/SwComponentTypes",
      "object_type": "AR-PACKAGE",
      "outer_element_tag": "/AR-PACKAGES",
      "ASW1": {
        "object_path": "/RB/PT/PCT_ASW1/SwComponentTypes/ASW1",
        "object_type": "APPLICATION-SW-COMPONENT-TYPE",
        "outer_element_tag": "/ELEMENTS",
        "PP_ASW1_PP1": { ...
      },
      "RP_ASW1_RP1": { ...
    },
  },
}
```

Hình 3.5 Cấu trúc dữ liệu sau khi được chuyển đổi sang dạng JSON

Tóm lại, AUTOSAR XML to JSON – AXJ trong AUTOSAR RTE generator trong nghiên cứu này thể hiện cam kết vững chắc về tính hiệu quả, độ chính xác và sự đơn giản. Bằng cách áp dụng phương pháp đệ quy, nghiên cứu không chỉ đạt được sự chuyển đổi dữ liệu toàn diện mà còn giảm đáng kể kích thước và độ phức tạp của mã.

Cách tiếp cận chiến lược này nâng cao năng lực khả năng cung cấp các giải pháp phần mềm ô tô đồng thời tối ưu hóa các nguồn lực phát triển ở mức tối đa.

3.2.3 JSON data Extraction and Synthetization – JES

Trong AUTOSAR RTE generator, JES đóng vai trò then chốt trong giai đoạn sau chuyển đổi. JES chịu trách nhiệm trích xuất dữ liệu từ cấu trúc JSON được tạo trong quá trình chuyển đổi XML sang JSON. Sau đó, nó liên kết liên mạch dữ liệu này để tổng hợp một bộ cấu trúc toàn diện, mỗi cấu trúc chứa thông tin liên quan.

3.2.3.1 Tổng hợp cấu trúc thông tin cho Receive/Provide Port API:

API của Receive Port được xác định như sau:

```
<return>  
Rte_[Byps_]DRead_<p>_<o>([IN Rte_Instance <instance>], [OUT  
Rte_TransformerError transformerError]) [5]
```

Trong đó <p> là tên cổng và <o> tên thành phần dữ liệu được định nghĩa trong Interface liên kết với Port. [Byps_] là một thành phần tùy chọn và sẽ không được dùng trong phạm vi nghiên cứu này. Ngoài ra để đơn giản, hai thành phần tùy chọn ([IN Rte_Instance <instance>] và [OUT Rte_TransformerError transformerError] cũng sẽ được mặc định là void.

Ví dụ một API để lấy dữ liệu từ một thành phần dữ liệu VDP_ASW1_Var1 (có Implementation DataType là IDT_ASW1_uint32) từ Receive Port RP_ASW2_RP1 của component ASW2 sẽ được xác định như sau:

```
IDT_ASW3_uint32 Rte_DRead_ASW2_RP_ASW2_RP2_VDP_ASW3_Var2(void )
```

API của Provide Port được xác định như sau:

```
Std_ReturnType Rte_[Byps_]Write_<p>_<o>([IN Rte_Instance <instance>], IN  
<data>,[OUT Rte_TransformerError transformerError])[5]
```

Trong đó <p> là tên cổng và <o> tên thành phần dữ liệu được định nghĩa trong Interface liên kết với Port. [Byps_] là một thành phần tùy chọn và sẽ không được dùng trong phạm vi nghiên cứu này. Ngoài ra để đơn giản, hai thành phần tùy chọn ([IN Rte_Instance <instance>] và [OUT Rte_TransformerError transformerError]

cũng sẽ được lược bỏ, IN xác định tham số đầu vào của API bao gồm thông tin của Implementation DataType và tên của đối số (mặc định là “data”).

Ví dụ một API để ghi dữ liệu vào một thành phần dữ liệu VDP_ASW1_Var1 (có Implementation DataType là IDT_ASW1_uint32) trong Port PP_ASW1_PP1 của component ASW1 sẽ được xác định như sau:

Std_ReturnType

Rte_Write_ASW1_PP_ASW1_PP1_VDP_ASW1_Var1(IDT_ASW1_uint32 data)

Trong chương trình ứng dụng, khi có một hoặc nhiều Receive/Provide Port được cấu hình, JES có nhiệm vụ tổng hợp những thông tin liên quan cần thiết thành một danh sách có tên là *receive_data_point_by_value_api_list* /

send_data_point_by_value_api_list, danh sách này phục vụ cho việc sinh ra API để thao tác dữ liệu trên Receive/Provide Port, những thông tin đó bao gồm:

- *port_api_type*: Implementation DataType của thành phần dữ liệu trong Port.
- *port_api_header*: Tiêu đề của Port API.
- *port_api_p*: Chứa tên thành phần của Port và tên của Port.
- *port_api_o*: Tên thành phần dữ liệu bên trong Port.
- *port_api_arg_type*: Chỉ định kiểu dữ liệu của đối số trong Port API được tạo.
- *port_api_arg_value*: Xác định tên đối số trong Port API được tạo.

Để *receive_data_point_by_value_api_list* / *send_data_point_by_value_api_list* có đầy đủ thông tin, JES sẽ thực hiện các bước như sau:

- Duyệt hết tất cả ASW. Lấy được tên của từng ASW.
- Trong mỗi ASW, duyệt hết tất cả Receive Port trong cấu trúc dữ liệu JSON, lấy tên Port.
- Mỗi Receive/Provide Port sẽ có những thông tin chính như tên port, Interface mà Port sử dụng mà trong Interface đó có chứa thành phần dữ liệu mà Port đang sử dụng. Từ đây tên của thành phần dữ liệu đã được xác định.

- Trong mỗi thành phần dữ liệu có chứa thông tin về Application DataType, tên của Application DataType trong thành phần dữ liệu này đã được xác định.
- Với tên của Application DataType, JES tiến hành tìm kiếm trong đối tượng JSON có kiểu là DATA-TYPE-MAPPING-SET, từ đây Implementation DataType liên kết với Application DataType được xác định thông qua đường dẫn nằm trong đối tượng IMPLEMENTATION-DATA-TYPE-REF.
- Tổng hợp các thông tin đã lấy được vào receive_data_point_by_value_api_list / send_data_point_by_value_api_list.

3.2.3.2 Tổng hợp thông tin danh sách kết nối các Port

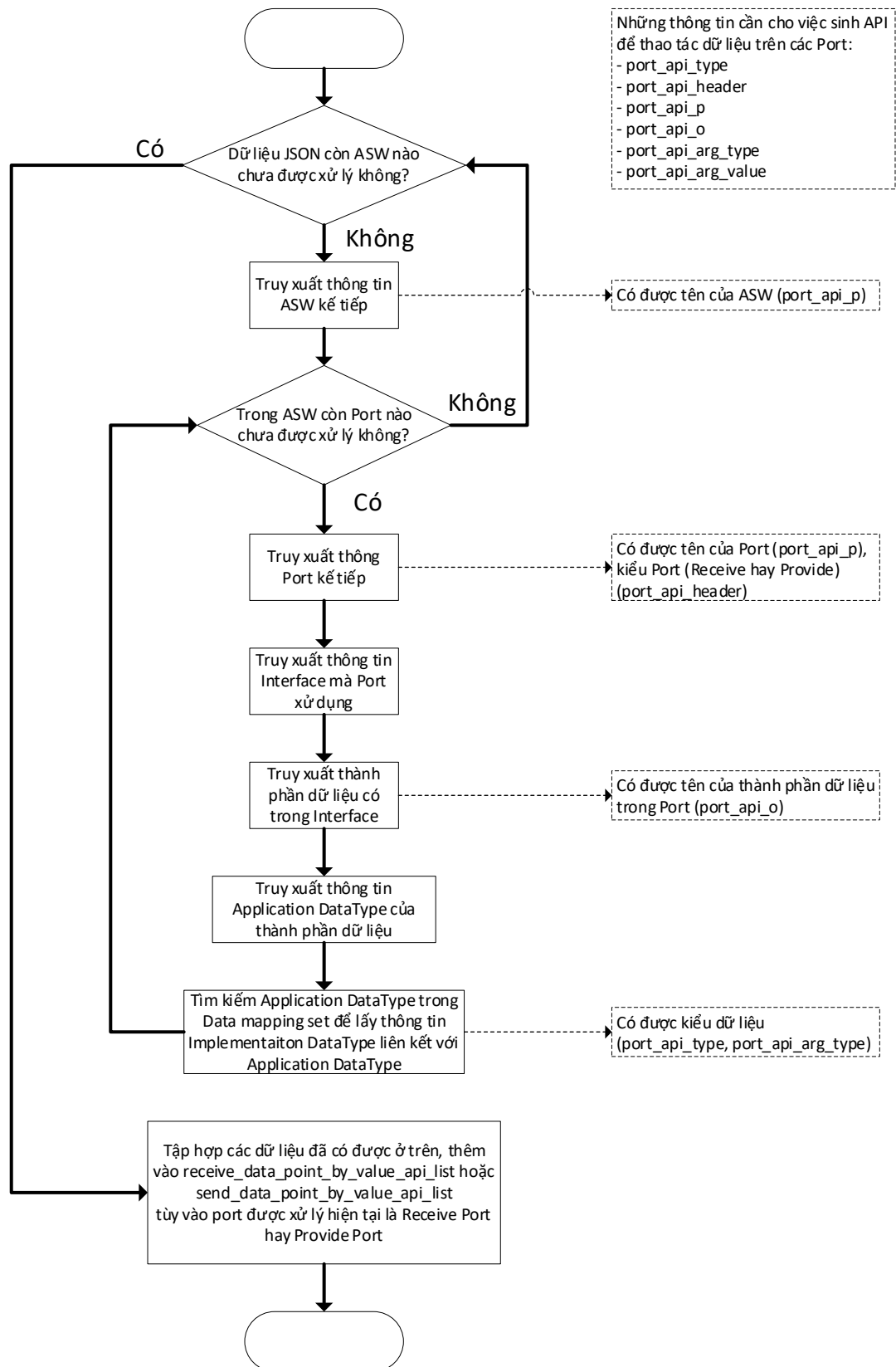
JES tập hợp các thông tin liên quan tới kết nối của các Provide Port và Receive Port vào một danh sách gọi là connector_list, danh sách này cung cấp danh mục đầy đủ các kết nối liên kết các cổng với nhau. Các đầu nối này đại diện cho mạng lưới kết nối phức tạp trong kiến trúc AUTOSAR. Từ các thông tin trong connector_list Source Code Generator – SCG sẽ nhận biết được Receive Port nào cần được cập nhật dữ liệu khi dữ liệu của Provide Port liên kết tới Receive Port đó được thay đổi.

Các thuộc tính trong một thành phần trong danh sách connector_list bao gồm:

- provider_component_name: tên của Provide Port
- provider_port_name: tên của ASW chứa Provide Port
- requester_component_name: tên của Receive Port
- requester_port_name: tên của ASW chứa Receive Port

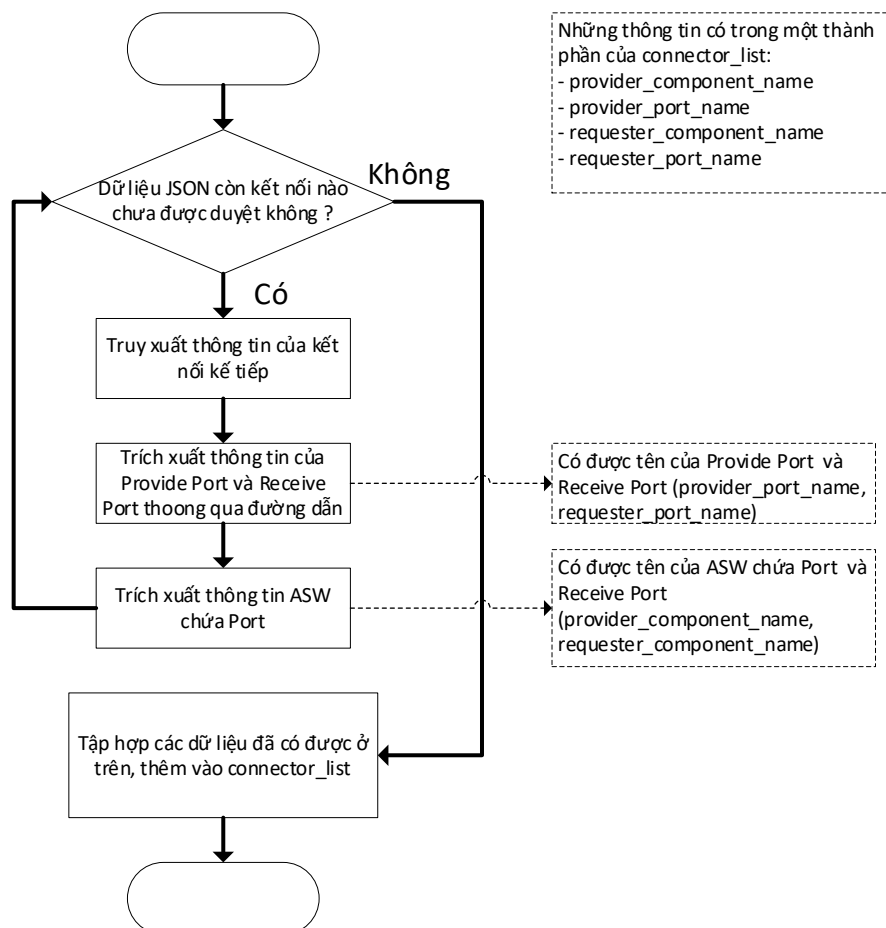
Hoạt động của việc tập hợp danh sách các kết nối liên kết Port được thực hiện theo các bước sau:

- JES duyệt trong cấu trúc JSON tất cả các kết nối chưa được xử lý – đối tượng có kiểu là ASSEMBLY-SW-CONNECTOR.
- Ứng với mỗi đối tượng kiểu ASSEMBLY-SW-CONNECTOR được duyệt, JES trích xuất thông tin đường dẫn của Provide Port và Receive Port thông qua key TARGET-P-PORT-REF và TARGET-R-PORT-REF



Hình 3.6 Chi tiết cách hoạt động của JES để thu thập thông tin của các Port

- Với các đường dẫn đã được trích xuất, JES truy cập tới Provide Port và Receive Port tương ứng để lấy thông tin về tên của Port và ASW chứa các Port đó.
- Tổng hợp các thông tin đã lấy được vào connector_list.



Hình 3.7 Chi tiết hoạt động của JES để thu thập thông tin của các kết nối Port

3.2.3.3 Tổng hợp các liên kết của sự kiện trong ASW đến OS Task

Để ASW có thể hoạt động thì các sự kiện của ASW cần được liên kết tới một OS Task nhất định (task thực hiện trong 10ms, 100ms,...), từ đó hệ điều hành có thể phân chia công việc của các ASW một cách chính xác. Các liên kết giữa các sự kiện của ASW và các OS Task được JES tổng hợp từ các thông tin trong cấu trúc dữ liệu JSON thành một danh sách được gọi là event_to_task_mapping_list.

Mỗi thành phần của danh sách event_to_task_mapping_list bao gồm những thuộc tính sau:

- event: sự kiện cần được liên kết
- runnable: tên hoạt động thực tế của ASW khi sự kiện được kích hoạt
- function_name: tên của hàm mô tả chi tiết hoạt động của runnable
- os_task: OS Task mà sự kiện sẽ được liên kết tới
- event_position_in_task: Một OS Task có thể kích hoạt một cách tuần tự nhiều sự kiện từ nhiều ASW. Tổng số này giúp xác định vị trí của sự kiện trong OS Task.

Hoạt động của việc tập hợp danh sách các liên kết sự kiện của ASW và OS Task được thực hiện theo các bước sau:

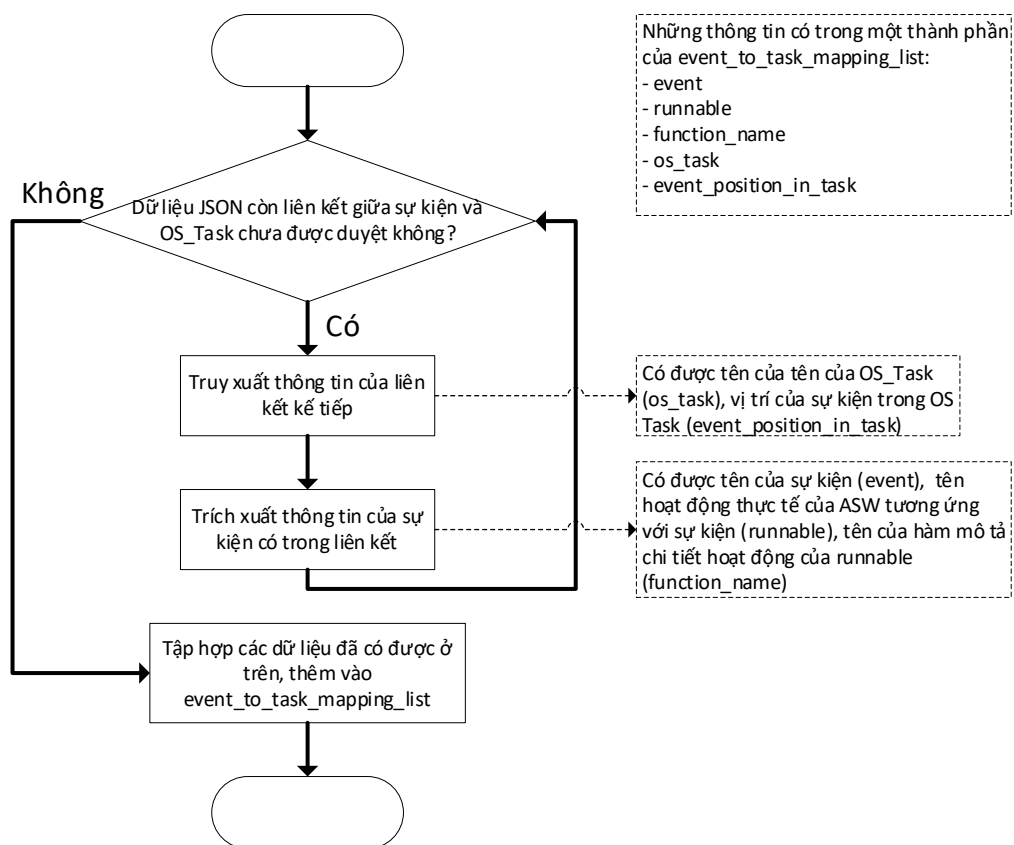
- Trong cấu trúc dữ liệu JSON, JES duyệt các đối tượng có thành phần DEFINITION-REF-RteEventToTaskMapping, những đối tượng này mang thông tin của liên kết giữa sự kiện và OS Task.
- Trích xuất thông tin của các đối tượng được duyệt ở trên để thu thập thông tin về os_task, event_position_in_task.
- Cũng trong các đối tượng được duyệt có đường dẫn tới đối tượng sự kiện. Từ đường dẫn này, JES truy cập tới đối tượng sự kiện và trích xuất các thông tin liên quan đến event, runnable và function_name.
- Tổng hợp các thông tin đã lấy được vào event_to_task_mapping_list.

3.2.3.4 Tổng hợp thông tin của OS task

Cuối cùng, JES điền vào một danh sách được gọi là task_info_list, danh sách này chứa rất nhiều thông tin liên quan đến các OS Task của hệ điều hành. Thông tin này bao gồm các thuộc tính dành riêng cho OS Task như os_task_name, task_cycle và task_priority. Những chi tiết này không thể thiếu để định cấu hình và tối ưu hóa việc thực thi các tác vụ theo thời gian thực.

Trong đó:

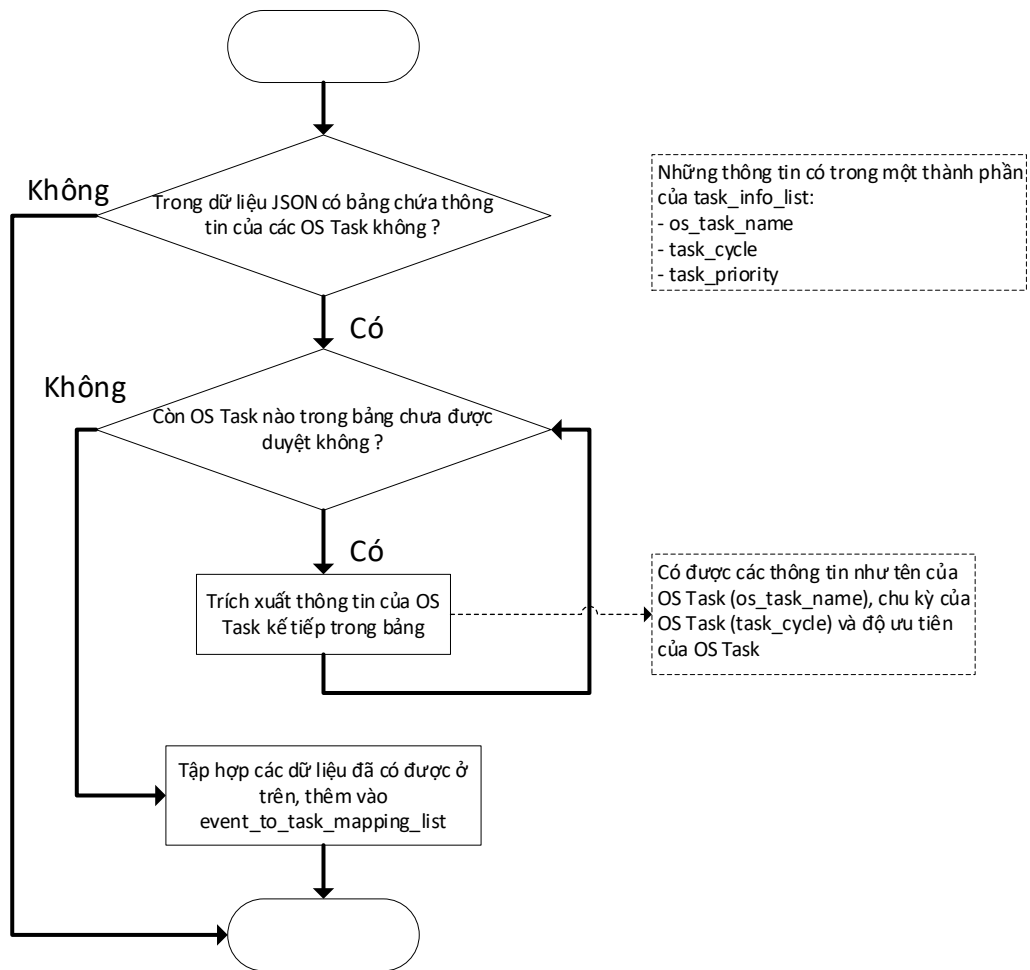
- os_task_name: tên của OS Task
- task_cycle: Chu kỳ hoạt động của OS Task (1ms, 10ms, 100ms, 1s,...)
- task_priority: độ ưu tiên của OS Task



Hình 3.8 Chi tiết cách hoạt động của JES để thu thập thông tin của các liên kết giữa sự kiện và OS_Task

Để trích xuất được những thông tin này trong cấu trúc dữ liệu JSON, JES thực hiện những công việc cụ thể sau:

- JES tìm kiếm đối tượng esc_stackDTtable trong dữ liệu JSON, đây là bảng chứa tất cả thông tin của các OS Task của ứng dụng.
- Sau khi có được dữ liệu từ esc_stackDTtable, JES tiến hành duyệt từng OS Task trong esc_stackDTtable để lấy những thông tin cần thiết (os_task_name, task_cycle, task_priority)
- Sau đó tổng hợp những thông tin này thành một thành phần của danh sách task_info_list.



Hình 3.9 Chi tiết cách hoạt động của JES để thu thập thông tin của các OS Task

3.2.4 Source Code Generator – SCG

Sau quá trình tổng hợp nhưng thông tin cần thiết hoàn tất bởi JES, tiếp theo SCG sẽ đảm nhận việc sinh mã từ những thông tin mà JES đã tổng hợp. Các tệp mã nguồn và header mà SCG sẽ sinh ra bao gồm: Rte_Type.h, Rte.h, Rte_<Component_name>.h, Rte.cpp, RTE_Event_To_Task_Mapping.cpp, main.cpp. Trong đó số lượng tệp Rte_<Component_name>.h phụ thuộc vào số lượng component có trong ứng dụng. Quá trình sinh ra từng tệp sẽ được trình bày chi tiết sau đây.

3.2.4.1 Tệp Rte_Type.h

Rte_Type.h định nghĩa lại các Implementation DataType được định nghĩa trong cấu hình AUTOSAR RTE thành các kiểu dữ liệu mà ngôn ngữ lập trình có thể hiểu được.

Ví dụ có một cấu hình Implementation DataType **IDT_ASW1_uint32** được định nghĩa là **uint32** trong ngôn ngữ lập trình, thì trong tệp Rte_Type.h sẽ có một dòng định nghĩa như sau:

typedef uint32 IDT_ASW1_uint32;

Để có được kết quả như trên SCG thực hiện các bước sau:

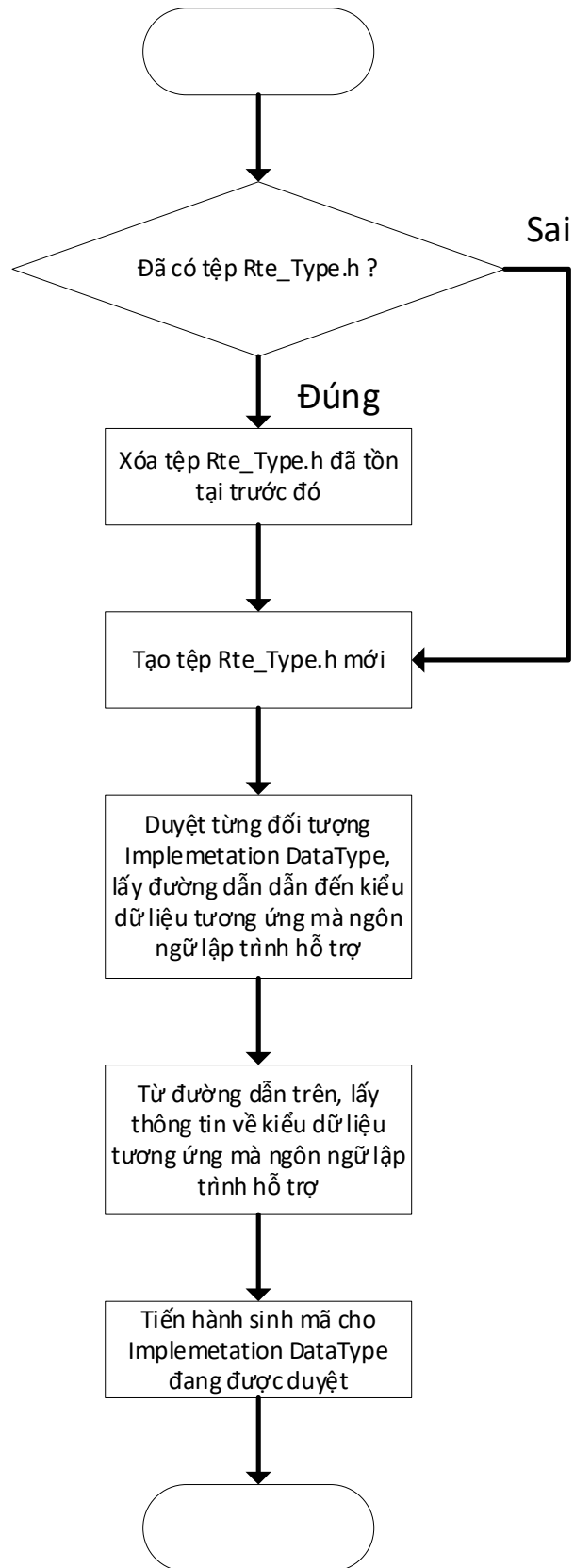
- Tạo tệp Rte_Type.h nếu như tệp chưa có sẵn, nếu như tệp đã tồn tại SCG tiến hành xóa tệp cũ và tạo lại tệp mới.
- Duyệt từng đối tượng Implementation DataType, trong mỗi đối tượng Implementation DataType có đường dẫn dẫn tới kiểu dữ liệu của ngôn ngữ
- Từ đường dẫn dẫn tới kiểu dữ liệu của ngôn ngữ SCG xác định được tên của kiểu dữ liệu mà ngôn ngữ lập trình hỗ trợ
- Tạo mã để định nghĩa Implementation DataType thành kiểu dữ liệu mà ngôn ngữ lập trình có thể hiểu được

3.2.4.2 Tệp Rte.h

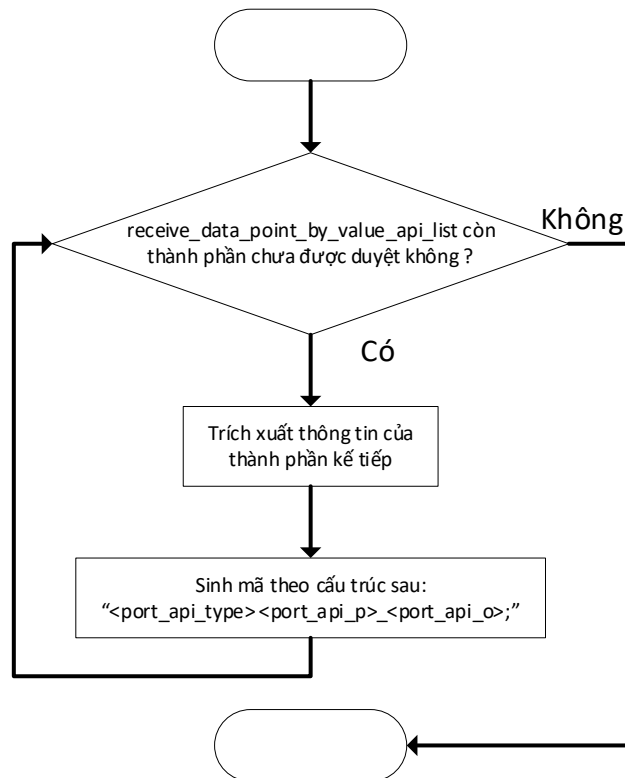
Tệp Rte.h chứa tất cả những khai báo tương ứng với các thành phần dữ liệu trong các Interface được sử dụng trong ứng dụng, ngoài ra các API cũng được khai báo ở tệp này. Quá trình sinh mã cho tệp Rte.h được chia thành hai phần, phần đầu tiên là sinh mã khai báo dữ liệu (khai báo biến), phần thứ hai là sinh mã cho các API để đọc/ghi dữ liệu.

Sinh mã khai báo dữ liệu (khai báo biến): quá trình này được thực hiện chi tiết như sau. SCG duyệt từng thành phần trong receive_data_point_by_value_api_list, ứng với mỗi thành phần trong receive_data_point_by_value_api_list, SCG sẽ dùng những thông tin port_api_type, port_api_p và port_api_o, nối những dữ liệu này lại với nhau theo mẫu "<port_api_type> <port_api_p>_<port_api_o>;", ví dụ thành phần dữ liệu có port_api_type: *IDT_ASW2_uint32*, port_api_p: *ASW1_RP_ASW1_RP1* và port_api_o: *VDP_ASW2_Var1* thì mã được sinh ra sẽ như sau:

IDT_ASW2_uint32 ASW1_RP_ASW1_RP1_VDP_ASW2_Var1;



Hình 3.10 Quá trình SCG sinh mã cho tệp Rte_Type.h



Hình 3.11 Quá trình sinh mã khai báo dữ liệu (khai báo biến)

Sinh mã cho các API để đọc/ghi dữ liệu: quá trình này được thực hiện chi tiết như sau, với dữ liệu mà JES đã tổng hợp trong `receive_data_point_by_value_api_list` và `send_data_point_by_value_api_list`, SCG xử dụng thông tin từ hai danh sách này để sinh ra từng API tương ứng.

API: `<port_api_type>`

`<port_api_header>_<port_api_p>_<port_api_o>(<port_api_arg_type>
<port_api_arg_value>);`

Ví dụ với một phần tử trong `receive_data_point_by_value_api_list` có các thông tin `port_api_type`: “`IDT_ASW2_uint32`”, `port_api_header`: “`Rte_DRead`”, `port_api_p`: “`ASW1_RP_ASW1_RP1`”, `port_api_o`: “`VDP_ASW2_Var1`”, `port_api_arg_type`: “`void`”, `port_api_arg_value`: “” thì API sinh ra sẽ như sau:

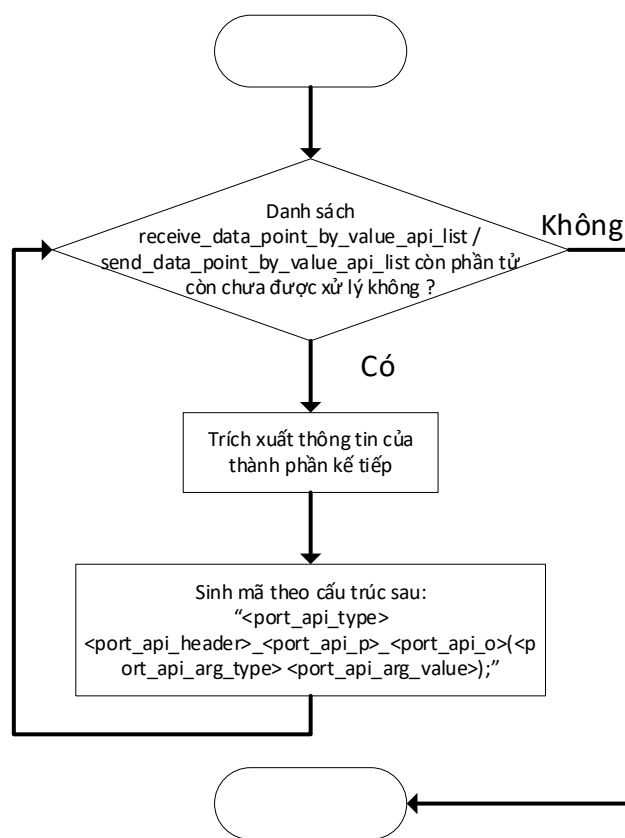
`IDT_ASW2_uint32 Rte_DRead_ASW1_RP_ASW1_RP1_VDP_ASW2_Var1(void);`

Còn với phần tử trong `send_data_point_by_value_api_list` ví dụ có các thông tin cụ thể như sau:

`port_api_type`: “*Std_ReturnType*”, `port_api_header`: “*Rte_Write*”, `port_api_p`: “*ASW1_PP_ASW1_PPI*”, `port_api_o`: “*VDP_ASW1_Var1*”, `port_api_arg_type`: “*IDT_ASW1_uint32*”, `port_api_arg_value`: “*data*” thì API được sinh ra sẽ là:

Std_ReturnType

Rte_Write_ASW1_PP_ASW1_PPI_VDP_ASW1_Var1(IDT_ASW1_uint32 data);



Hình 3.12 Quá trình sinh mã khai cho các API để đọc/ghi dữ liệu

3.2.4.3 Các tệp *Rte_<Component_name>.h*

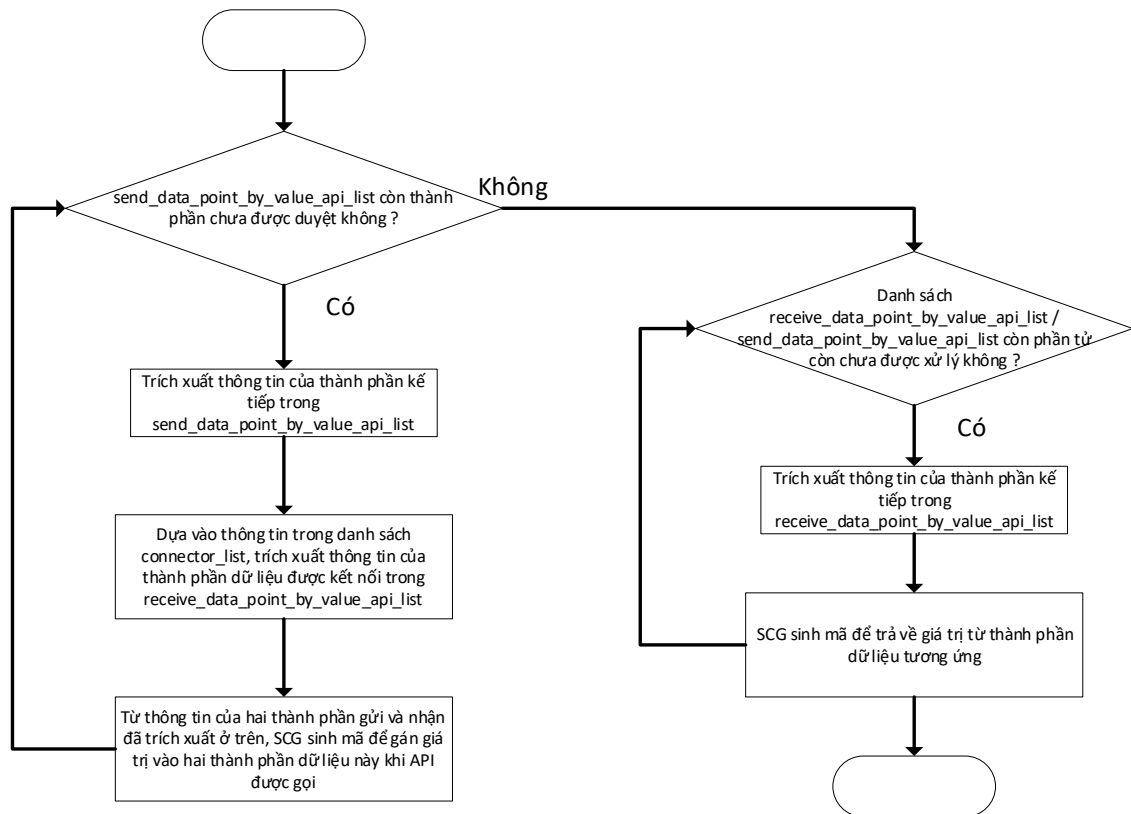
Các tệp *Rte_<Component_name>.h* trong đó *<Component_name>* là tên của component trong ứng dụng, ứng dụng có bao nhiêu component thì SCG sẽ sinh ra bấy nhiêu tệp *Rte_<Component_name>.h*, tệp này đơn giản chỉ chứa những API rút gọn của các API trong tệp *Rte.h*, thành phần được rút gọn là tên của chính component này trong API, ví dụ nếu tệp *Rte_ASW1.h* sử dụng API

Rte_DRead_ASW1_RP_ASW1_RP2_VDP_ASW3_Var1 thì sẽ có một API rút gọn đi tên của chính component này và API rút gọn sẽ là Rte_DRead_RP_ASW1_RP2_VDP_ASW3_Var1.

3.2.4.4 Tập Rte.cpp

Tập Rte.cpp sẽ là tập định nghĩa hành vi của từng API trong ứng dụng. quá trình sinh mã cho phần định nghĩa này được thực hiện theo trình tự như sau:

- SCG duyệt từng phần tử trong send_data_point_by_value_api_list, đồng thời kiểm tra thông tin về kết nối trong connector_list để sinh ra mã định nghĩa hoạt động cho từng API ghi dữ liệu tương ứng. Ví dụ một thành phần trong send_data_point_by_value_api_list có port_api_p: “ASW1_PP_ASW1_PPI”, port_api_o: “VDP_ASW1_Var1” được kết nối tới một thành phần trong receive_data_point_by_value_api_list có port_api_p: “ASW2_RP_ASW2_RPI”, port_api_o: “VDP_ASW1_Var1” thì API được sinh ra là *Std_ReturnType Rte_Write_ASW1_PP_ASW1_PPI_VDP_ASW1_Var1(IDT_ASW1_uint32 data)* và trong phần định nghĩa của API này sẽ thực hiện hai phép gán, phép gán thứ nhất là gán dữ liệu cần ghi vào thành phần dữ liệu của Provide Port, phép gán còn lại sẽ gán dữ liệu cần ghi cho thành phần dữ liệu của Receive Port. Với những dữ liệu từ ví dụ trên thì mã được sinh ra cho hai phép gán này là:
ASW1_PP_ASW1_PPI_VDP_ASW1_Var1 = data; và
ASW2_RP_ASW2_RPI_VDP_ASW1_Var1 =
ASW1_PP_ASW1_PPI_VDP_ASW1_Var1;
- SCG duyệt từng phần tử trong receive_data_point_by_value_api_list, để sinh ra mã thực hiện trả về giá trị của thành phần dữ liệu tương ứng: Ví dụ một thành phần trong receive_data_point_by_value_api_list có thông tin port_api_p: “ASW1_RP_ASW1_RPI”, port_api_o: “VDP_ASW2_Var1” thì mã sinh ra có việc trả về giá trị này là:
return ASW1_RP_ASW1_RPI_VDP_ASW2_Var1;



Hình 3.13 Quá trình sinh mã định nghĩa hành vi của API.

3.2.4.5 Tập *RTE_Event_To_Task_Mapping.cpp*

Từ những thông tin mà JES tổng hợp trong danh sách *event_to_task_mapping_list*, SCG tiến hành việc sinh mã cho hành vi cụ thể của từng OS Task như sau:

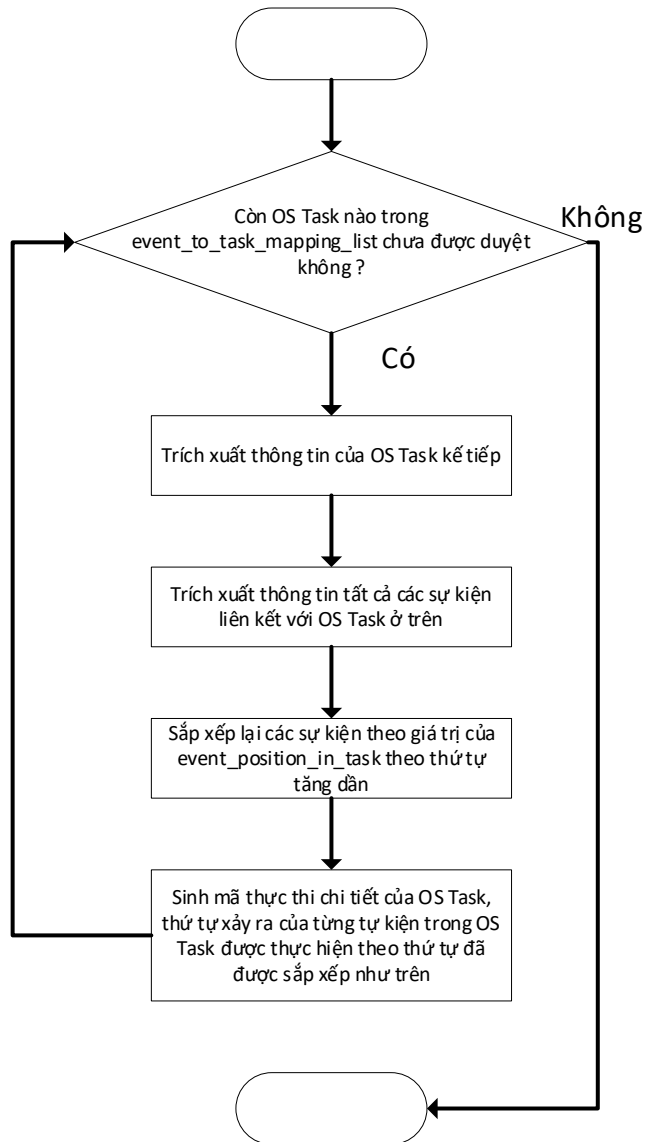
- Ứng với mỗi OS Task, SCG lấy trích xuất thông tin của tất cả các sự kiện liên kết tới OS Task này.
- SCG thực hiện sắp xếp lại các sự kiện theo thứ tự thực thi từ thấp đến cao dựa theo thông tin *event_position_in_task*
- Cuối cùng là sinh mã thực thi chi tiết cho từng OS Task

Ví dụ một OS Task có tên là *OS_1Q1_10ms_Task* được liên kết với các sự kiện *ASW1_10ms*, *ASW3_10ms* và *ASW2_10ms* với giá trị *event_position_in_task* tương ứng lần lượt là 1, 3, 2 thì phần mã thực thi chi tiết được sinh ra là:

```

void OS_1Q1_10ms_Task(){
    (void) ASW1_10ms();
    (void) ASW2_10ms();
    (void) ASW3_10ms();
}

```



Hình 3.14 Quá trình sinh mã cho hành vi của từng OS Task

3.2.4.6 Tập main.cpp

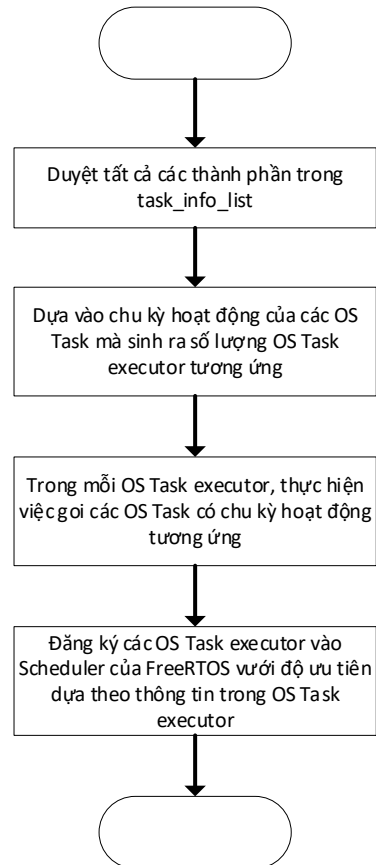
Đây là tệp chính của toàn ứng dụng, tất cả những hoạt động của các component trong ứng dụng sẽ được quản lý bởi tệp này.

Cụ thể thì tệp này sẽ định nghĩa những hàm (OS Task executor) thực thi việc gọi từng OS Task của ứng dụng. Những hàm này sẽ quyết định chu kỳ thực thi của OS Task (1ms, 10ms, 1000ms,...) và độ ưu tiên của OS Task bằng cách đăng ký OS Task executor với FreeRTOS thông qua API `xTaskCreate` được cung cấp bởi FreeRTOS. Các thông tin về chu kỳ và độ ưu tiên được trích xuất từ danh sách `task_info_list`. Cụ thể quá trình sinh mã cho tệp `main.cpp` được thực hiện theo các bước sau:

- Duyệt từng thành phần của `task_info_list` để biết được có bao nhiêu OS Task executor sẽ được tạo. Ví dụ trong chương trình có 5 OS Task, trong đó có 2 OS Task thực thi với chu kỳ 1000ms và 3 OS Task thực thi với chu kỳ 10ms thì số lượng OS Task executor được tạo là 2, một cho chu kỳ 10ms và một cho chu kỳ 1000ms.
- Trong mỗi OS Task executor, các OS Task với chu kỳ thực thi tương ứng sẽ được gọi.
- Để đảm bảo mỗi OS Task executor thực thi đúng thời gian được quy định thì API `vTaskDelayUntil` được hỗ trợ bởi FreeRTOS sẽ được sử dụng.
- Cuối cùng là các OS Task executor được đăng ký vào Scheduler của FreeRTOS với độ ưu tiên được các định trong thành phần của `task_info_list`.

Ví dụ một OS Task được thực thi với chu kỳ 10ms thì OS Task Executor sẽ như sau:

```
void esc_stack_EventList_OS_10ms_Task( void *pvParameters ){
    TickType_t xLastWakeTime;
    const TickType_t xFrequency = 10;
    for(;;){
        (void) OS_1Q1_10ms_Task();
        vTaskDelayUntil( &xLastWakeTime, xFrequency );
    }
}
```



Hình 3.15 Quá trình tạo OS Task Executor và đăng ký Scheduler của FreeRTOS

Tóm lại, Source Code Generator – SCG đóng vai trò cốt lõi trong AUTOSAR RTE generator, nó thực đại diện cho chức năng chính của AUTOSAR RTE generator là sinh mã. Với sự hỗ trợ từ JES thì công việc của SCG đã được tối ưu tối đa, nhằm giảm mức độ phức tạp của trong AUTOSAR RTE generator bằng chia nhỏ công việc và xử lý những thông tin thực sự cần thiết.

CHƯƠNG 4 KẾT QUẢ THỰC NGHIỆM

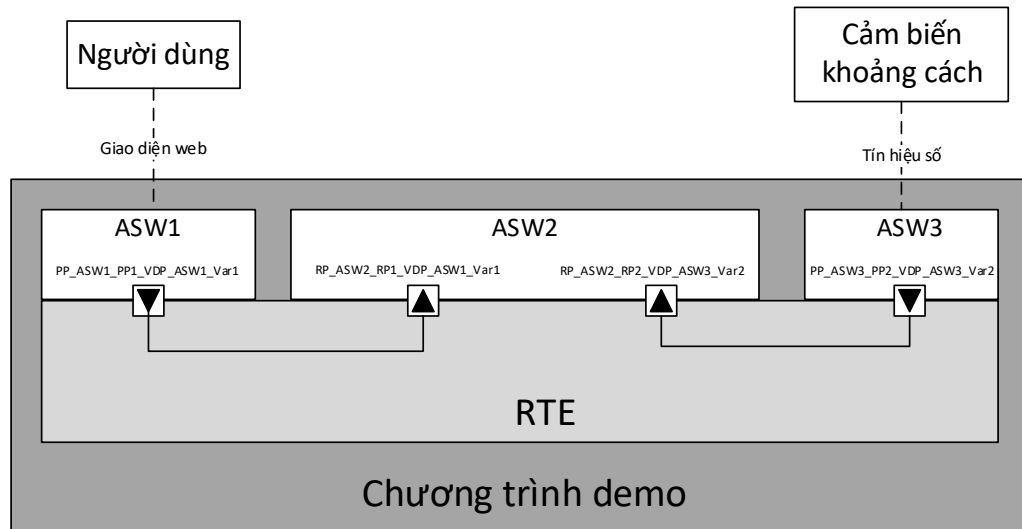
4.1 Giới thiệu cách triển khai công cụ sinh mã nguồn RTE

Để đánh giá hiệu quả và khả năng ứng dụng thực tế của công cụ AUTOSAR RTE generator, một ứng dụng xe demo được thiết kế để vi điều khiển ESP32 có thể thực thi những mã nguồn mà AUTOSAR RTE generator đã tạo ra. Ứng dụng này thể hiện các nguyên tắc của AUTOSAR RTE và thể hiện sự tích hợp liền mạch của các thành phần phần mềm ô tô khác nhau. Mục tiêu chính của bản demo này là kiểm soát chuyển động của phương tiện vào bãi đỗ xe thông qua giao diện web, kết hợp tính năng an toàn quan trọng nhằm đảm bảo phương tiện dừng lại khi khoảng cách tới chướng ngại vật phía sau quá gần.

Bản demo bao gồm ba thành phần phần mềm ô tô (ASW) riêng biệt được đặt tên tương ứng là ASW1 (quản lý giao diện điều khiển xe qua web), ASW2 (điều khiển động cơ) và ASW3 (thu thập tín hiệu từ cảm biến khoảng cách), mỗi thành phần được cấu hình tỉ mỉ theo nguyên tắc AUTOSAR RTE.

Trong đó:

- ASW1 có 1 Provide Port PP_ASW1_PP1_VDP_ASW1_Var1 mang giá trị của lệnh điều khiển đọc được từ người dùng sau khi đã được xử lý.
- ASW3 có 1 Provide Port PP_ASW3_PP2_VDP_ASW3_Var2 mang giá trị của tín hiệu cảm biến khoảng cách sau khi đã được xử lý.
- ASW2 có 2 Receive Port:
 - RP_ASW2_RP1_VDP_ASW1_Var1: nhận tín hiệu điều khiển của người dùng từ Provide Port PP_ASW1_PP1_VDP_ASW1_Var1 của ASW1.
 - RP_ASW2_RP2_VDP_ASW3_Var2: nhận giá trị khoảng cách của cảm biến từ Provide Port PP_ASW3_PP2_VDP_ASW3_Var2 của ASW3.



Hình 4.1 Kiến trúc ứng dụng demo

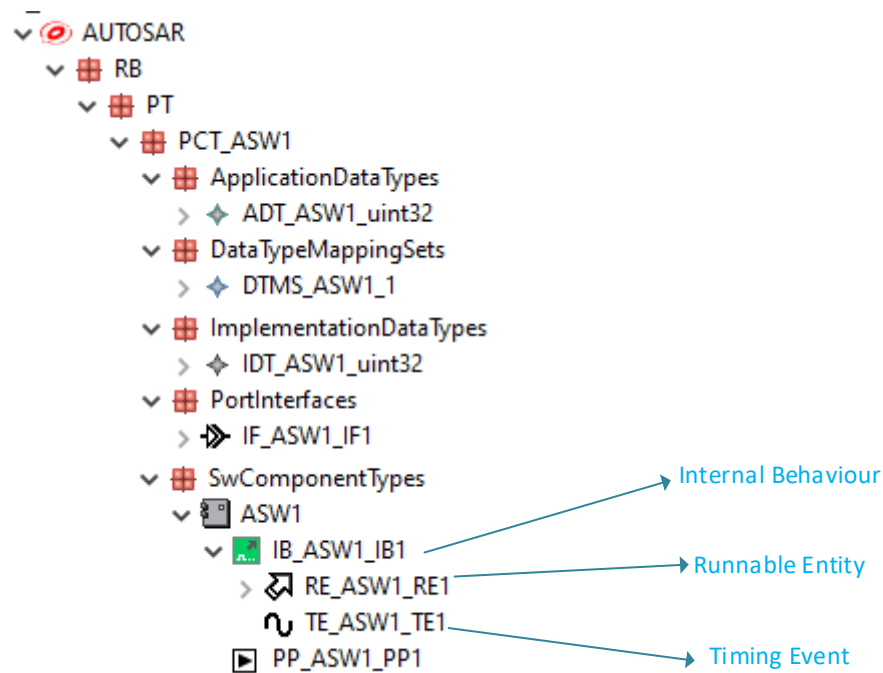
4.2 Cấu hình chi tiết thành phần phần mềm cho việc kiểm tra mã RTE được tạo

4.2.1 ASW1 - Quản lý giao diện điều khiển xe qua web

4.2.1.1 Cấu hình AUTOSAR của ASW1

Cấu hình AUTOSAR của ASW1 bao gồm:

- 1 Application DataType: ADT_ASW1_uint32
- 1 Implementation DataType: IDT_ASW1_uint32
- 1 DataType Mapping Set để liên kết ADT_ASW1_uint32 và IDT_ASW1_uint32 lại với nhau: DTMS_ASW1_1
- 1 Port Interface: IF_ASW1_IF1
- 1 SW Component: ASW1, trong ASW1 có:
- 1 Internal Behaviour: IB_ASW1_IB1, trong Internal Behaviour này có:
- 1 Runnable Entity: RE_ASW1_RE1
- 1 Timing Event: TE_ASW1_TE1
- 1 Provide Port: PP_ASW1_PP1



Hình 4.2 Cấu hình Autosar của ASW1

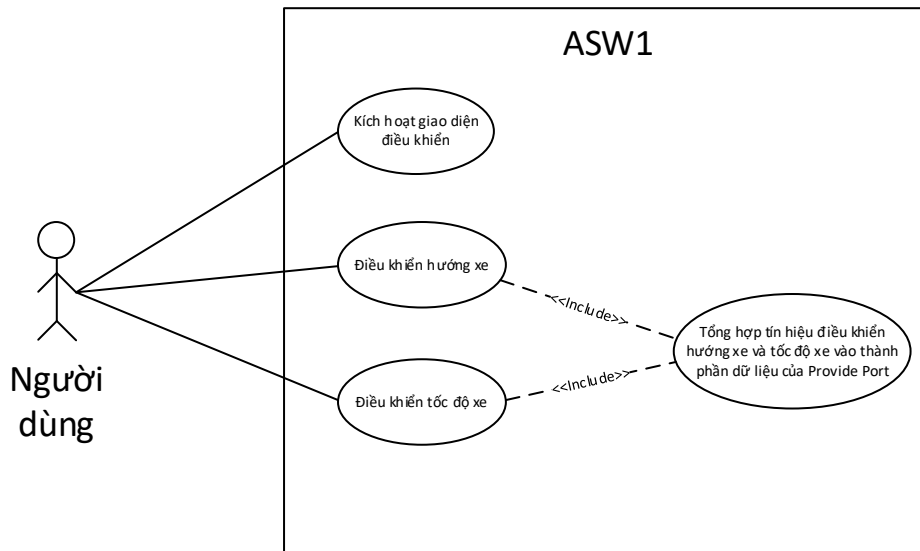
4.2.1.2 Hoạt động của ASW1

ASW1 đóng vai trò như một máy chủ web, đóng vai trò là cổng cho các tín hiệu điều khiển nhận được từ giao diện web của người dùng. Sau đó, nó chuyển tiếp các tín hiệu điều khiển này đến ASW2 chịu trách nhiệm điều khiển động cơ.

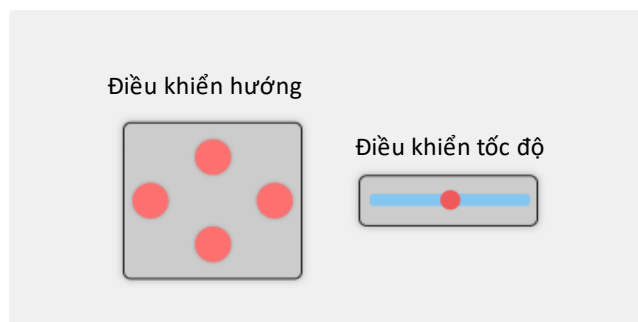
ASW1 cho phép người dùng kiểm soát động cơ của xe thông qua giao diện web.

ASW1 thực hiện các công việc chi tiết như sau:

- Mở WebServer cho phép người dùng có thể truy cập để yêu cầu giao diện điều khiển
- Tiếp nhận tín hiệu của người dùng và xử lý trước khi ghi dữ liệu vào PP_ASW1_PP1_VDP_ASW1_Var1.
- Từ bit0 đến bit7 của VDP_ASW1_Var1 lưu trữ giá trị tốc độ động cơ
- Từ bit8 đến bit15 của VDP_ASW1_Var1 lưu trữ giá trị của hướng điều khiển của xe: với các giá trị tương ứng 0: tiến thẳng, 1: quay phải, 2: lùi, 3: quay trái.



Hình 4.3 Tương tác giữa người dùng tới ASW1 của ứng dụng demo



Hình 4.4 Giao diện web điều khiển xe

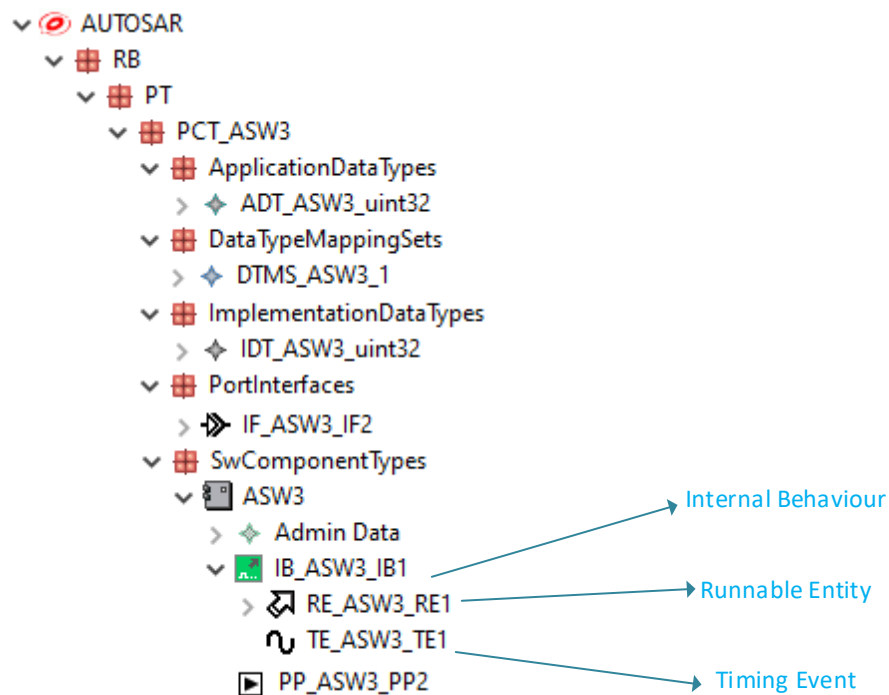
4.2.2 ASW3 - Thu thập tín hiệu từ cảm biến khoảng cách

4.2.2.1 Cấu hình AUTOSAR của ASW3

Cấu hình AUTOSAR của ASW3 bao gồm:

- 1 Application DataType: ADT_ASW3_uint32
- 1 Implementation DataType: IDT_ASW3_uint32
- 1 DataType Mapping Set để liên kết ADT_ASW3_uint32 và IDT_ASW3_uint32 lại với nhau: DTMS_ASW3_1
- 1 Port Interface: IF_ASW3_IF2
- 1 SW Component: ASW3, trong ASW3 có:

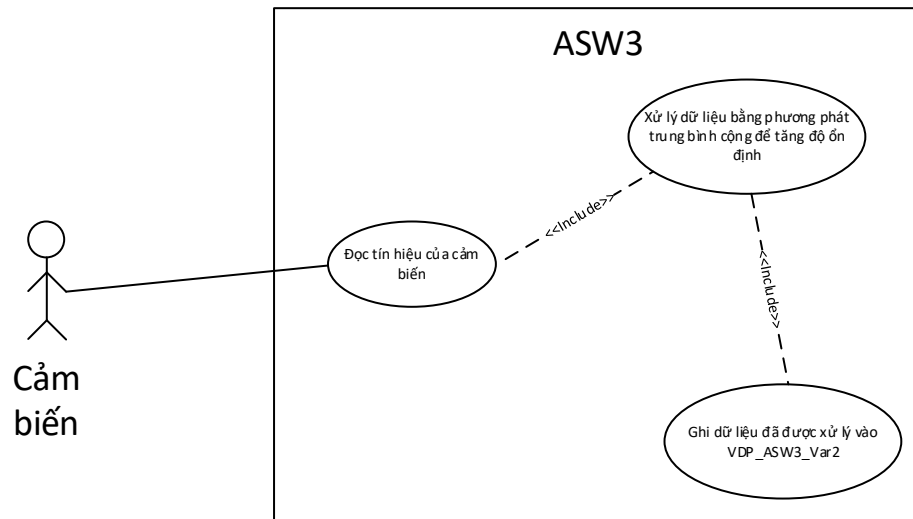
- 1 Internal Behaviour: IB_ASW3_IB1, trong Internal Behaviour này có:
- 1 Runnable Entity: RE_ASW3_RE1
- 1 Timing Event: TE_ASW3_TE1
- 1 Provide Port: PP_ASW3_PP2



Hình 4.5 Cấu hình Autosar của ASW3

4.2.2.2 Hoạt động của ASW3

ASW3 có nhiệm vụ thu thập khoảng cách từ đuôi xe tới vật cản phía sau xe sử dụng cảm biến siêu âm. Trước khi ghi dữ liệu vào VDP_ASW3_Var2 thì giá trị tín hiệu từ cảm biến được xử lý qua bộ lọc trung bình cộng để có kết quả ổn định hơn.



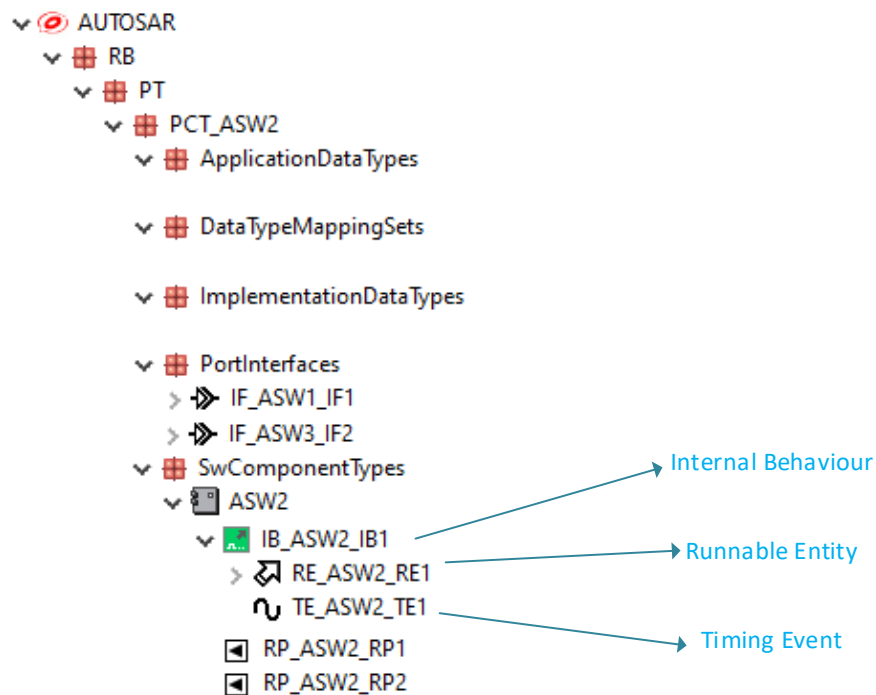
Hình 4.6 Tương tác giữa cảm biến và ASW3

4.2.3 ASW2 - Điều khiển động cơ

4.2.3.1 Cấu hình AUTOSAR của ASW2

Cấu hình AUTOSAR của ASW2 bao gồm:

- 2 Receive Interface: IF_ASW1_IF1 và IF_ASW3_IF2
- 1 SW Component: ASW2, trong ASW2 có:
- 1 Internal Behaviour: IB_ASW2_IB1, trong Internal Behaviour này có:
- 1 Runnable Entity: RE_ASW2_RE1
- 1 Timing Event: TE_ASW2_TE1
- 2 Provide Port: PP_ASW2_RP1 và PP_ASW2_RP2



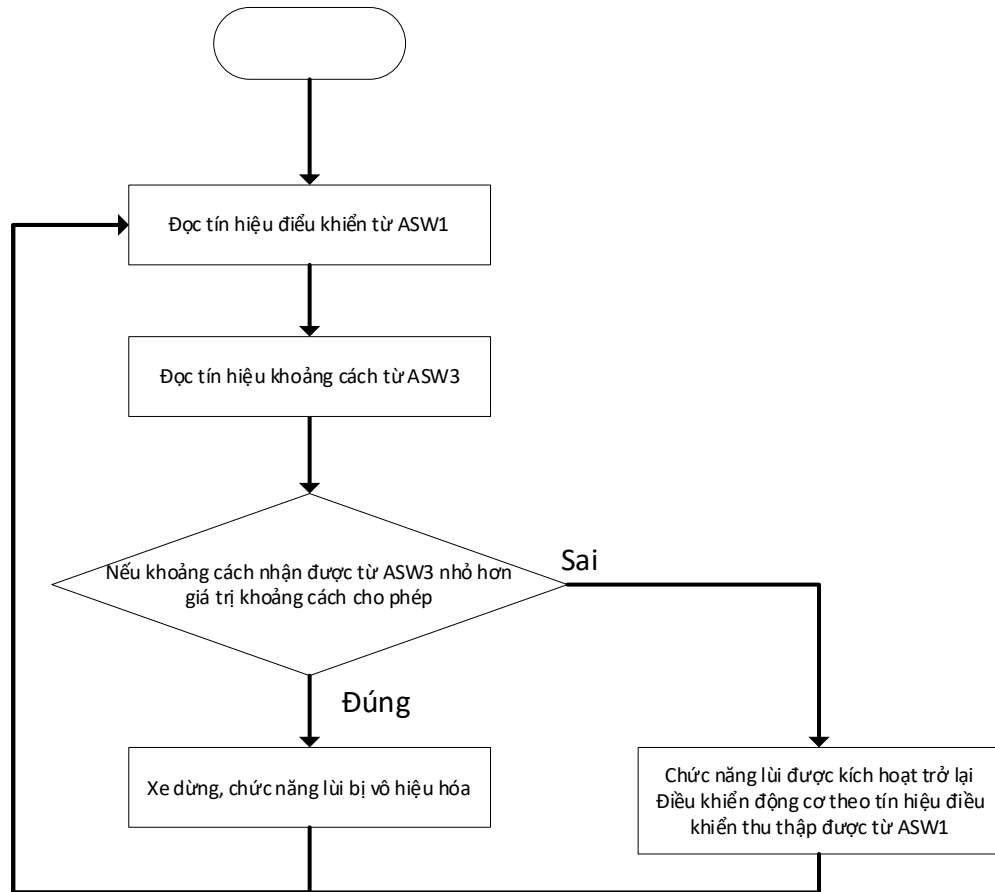
Hình 4.7 Cấu hình AUTOSAR của ASW2

4.2.3.2 Hoạt động của ASW2

ASW2 phụ trách việc kết hợp tín hiệu điều khiển của người dùng và giá trị khoảng cách được lấy từ cảm biến đã được xử lý bởi ASW3 để cho ra tín hiệu điều khiển động cơ phù hợp nhất.

Chi tiết cách hoạt động của ASW2 như sau:

- Đọc tín hiệu điều khiển từ ASW1
- Đọc tín hiệu khoảng cách từ ASW3
- Nếu giá trị khoảng cách nhận được từ ASW3 vượt quá giá trị cho phép thì tín hiệu điều khiển xe lùi xe bị vô hiệu hóa và xe sẽ dừng, nếu khoảng cách lớn hơn lớn giá trị khoảng cách tối thiểu cho phép thì chức năng lùi sẽ được kích hoạt trở lại.



Hình 4.8 Chi tiết hoạt động của ASW2

4.3 Cấu hình chi tiết phần cứng của demo

4.3.1 Vi điều khiển xử lý trung tâm - Wemos D1 R32

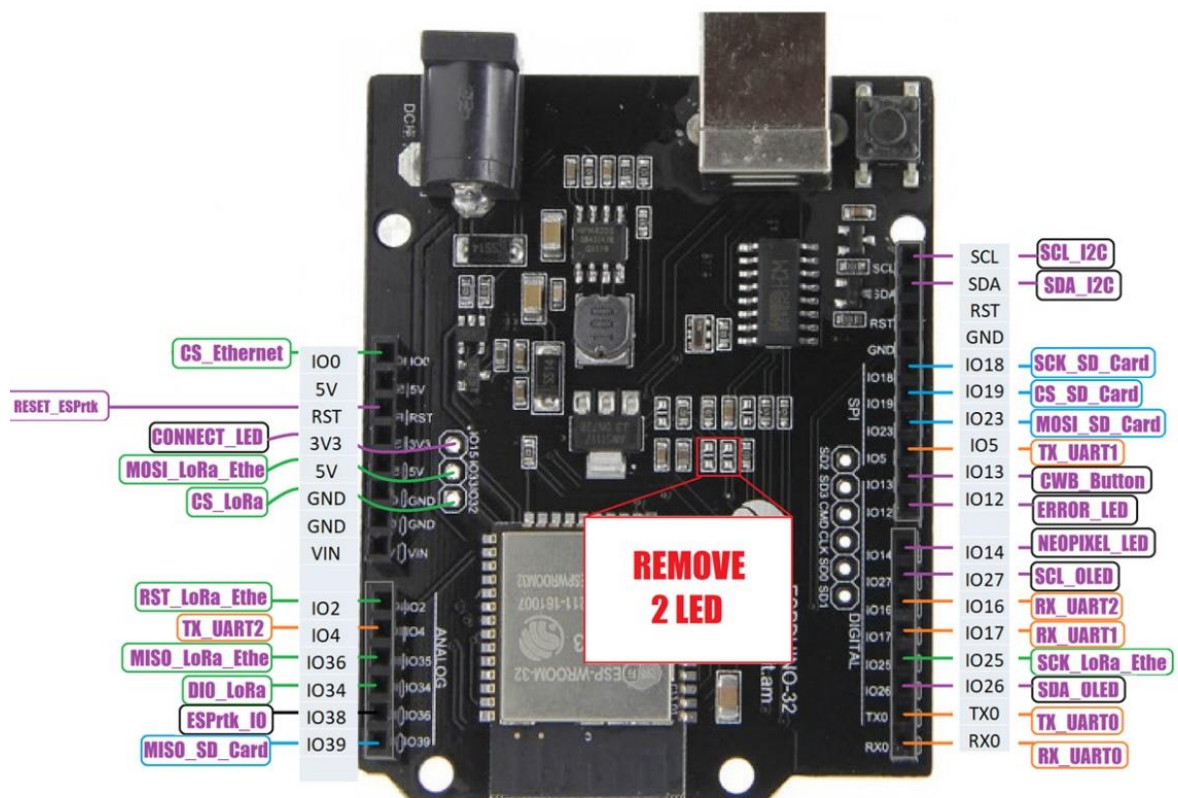
Wemos D1 R32 là một trong những bo mạch phổ biến trong cộng đồng IoT (Internet of Things). Được phát triển bởi Wemos, một công ty chuyên về các sản phẩm và dịch vụ IoT, bo mạch này dựa trên vi điều khiển ESP32 của Espressif Systems. ESP32 là một vi điều khiển mạnh mẽ và linh hoạt, kết hợp nhiều tính năng như Wi-Fi, Bluetooth Low Energy (BLE), GPIOs đa năng và nhiều giao tiếp khác.

Wemos D1 R32 được thiết kế nhỏ gọn, dễ sử dụng và có khả năng kết nối với Internet thông qua Wi-Fi, điều này giúp nó trở thành một lựa chọn phổ biến cho các ứng dụng IoT như cảm biến thông minh, hệ thống giám sát, điều khiển từ xa và nhiều ứng dụng khác. Bên cạnh đó, việc hỗ trợ nhiều thư viện và công cụ phát triển phong phú cũng

làm cho việc lập trình và phát triển các dự án IoT trở nên dễ dàng hơn với Wemos D1 R32.

WeMos D1 R32 có một số tính năng nổi bật sau:

- Vi điều khiển ESP32 với hai lõi 32-bit Xtensa LX6, tốc độ xung nhịp lên đến 240 MHz
- Bộ nhớ flash 4 MB và SRAM 520 KB
- Wi-Fi 802.11 b/g/n/e/i
- Bluetooth 4.2 LE
- 32 chân GPIO
- Hỗ trợ các giao diện I²C, SPI và UART
- Pinout dạng Arduino UNO R3



Hình 4.9 Wemos D1 R32 Pinout

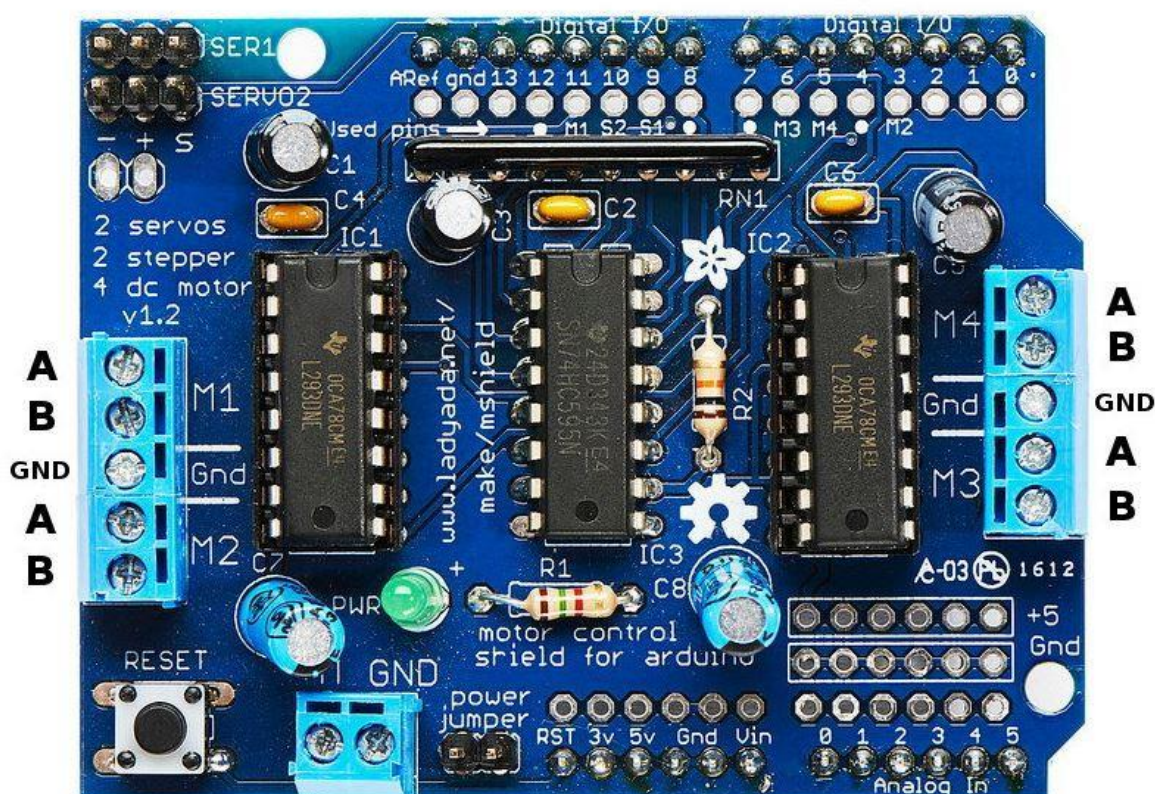
4.3.2 Module điều khiển động cơ - L293D shield

L293D shield là một bo mạch mở rộng cho Arduino, cho phép điều khiển tối đa bốn động cơ DC hoặc hai động cơ bước. Bo mạch này dựa trên IC L293D, một bộ điều khiển động cơ cầu H kép.

L293D shield có các tính năng sau:

- Hai kênh điều khiển động cơ DC
- Mỗi kênh có thể cung cấp dòng điện tối đa 1,2 A
- Hỗ trợ điện áp động cơ từ 4,5 V đến 25 V
- Có thể sử dụng nguồn điện riêng biệt cho động cơ

L293D shield là một lựa chọn tuyệt vời cho các dự án cần điều khiển động cơ DC hoặc động cơ bước. Nó dễ sử dụng và có thể được lập trình bằng Arduino.



Hình 4.10 L293D Shield

4.3.3 Cảm biến siêu âm - SRF05

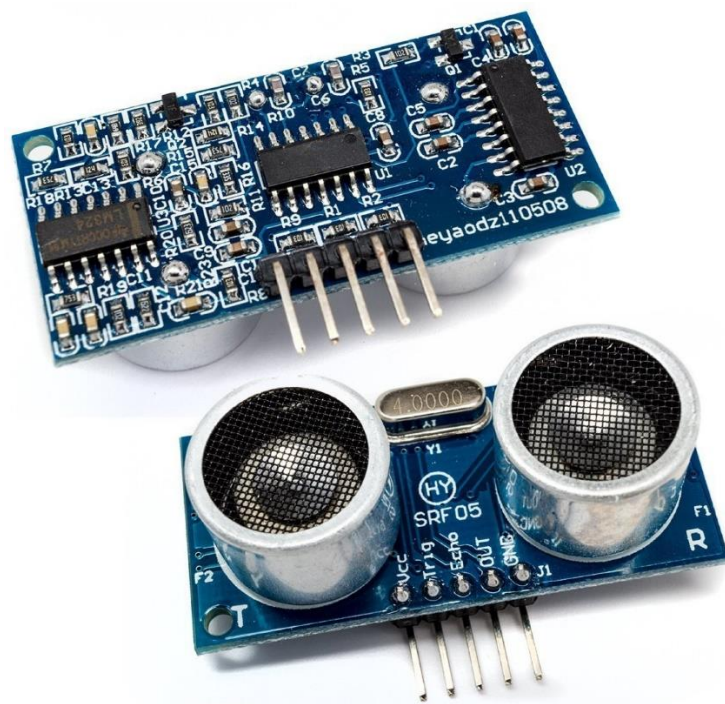
SRF05 là một cảm biến siêu âm được sử dụng để đo khoảng cách. Nó hoạt động bằng cách phát ra một xung âm thanh và sau đó đo thời gian cần thiết để âm thanh phản hồi trở lại.

Cảm biến SRF05 có các thông số kỹ thuật sau:

- Khoảng cách đo: 1 cm đến 4 m
- Độ chính xác: ± 3 cm
- Tần số âm thanh: 40 kHz
- Điện áp hoạt động: 5V

Cảm biến SRF05 có thể được sử dụng cho nhiều ứng dụng khác nhau, bao gồm:

- Tránh va chạm
- Đo khoảng cách
- Theo dõi chuyển động
- Kiểm soát robot



Hình 4.11 SRF05

4.3.4 Nguồn cấp năng lượng – Pin 18650

Pin 18650 là loại pin lithium-ion có đường kính 18 mm và chiều cao 65 mm. Nó là một loại pin phổ biến được sử dụng trong nhiều thiết bị điện tử, bao gồm đèn pin, máy ảnh, máy tính xách tay,...

Pin 18650 có dung lượng cao, có thể cung cấp năng lượng cho các thiết bị điện tử trong thời gian dài. Nó cũng có trọng lượng nhẹ và kích thước nhỏ gọn, khiến nó trở thành một lựa chọn lý tưởng cho các thiết bị di động.

Pin 18650 là một loại pin linh hoạt và hiệu quả, có thể được sử dụng cho nhiều ứng dụng khác nhau. Khi sử dụng pin 18650 đúng cách, bạn có thể kéo dài tuổi thọ của pin và đảm bảo an toàn cho bản thân.

Dưới đây là một số thông số kỹ thuật cơ bản của pin 18650:

- Dung lượng: 2000 mAh đến 5000 mAh
- Điện áp: 3,7 V
- Sức mạnh: 10 A đến 30 A
- Nhiệt độ hoạt động: -20 °C đến 60 °C



Hình 4.12 Pin 18650

4.3.5 Động cơ

Động cơ được sử dụng trong demo này được sử dụng phổ biến trong các ứng dụng như robot, mô hình, và thiết bị gia dụng.

Động cơ DC trong demo này có các thông số kỹ thuật sau:

- Điện áp hoạt động: 3 V đến 9 V
- Tốc độ không tải: 125 vòng/phút (3 V) đến 208 vòng/phút (9 V)
- Tỷ số truyền: 1:48
- Mô-men xoắn: 800 gf.cm (3 V)

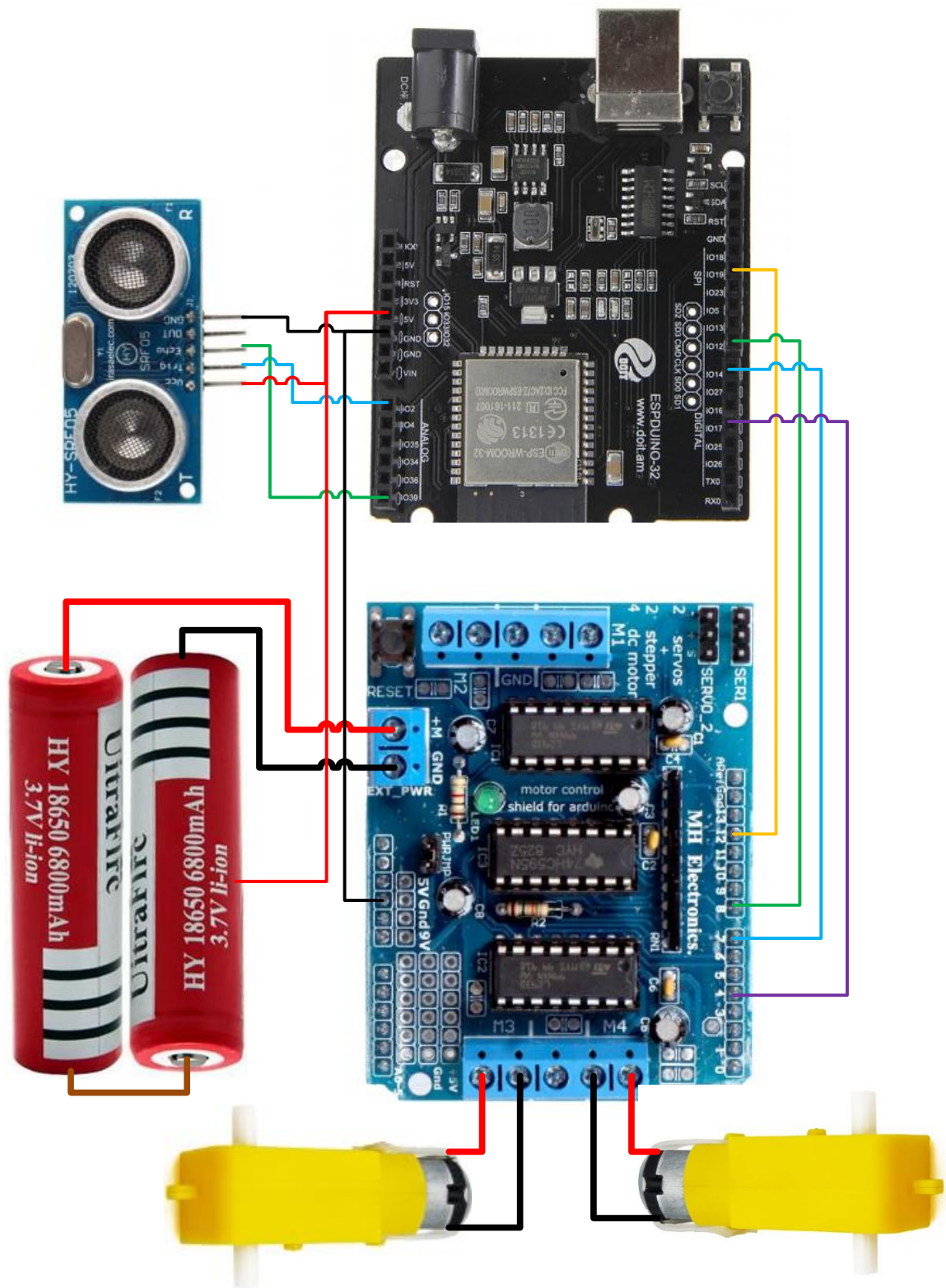
Ưu điểm của động cơ:

- Giá thành rẻ
- Dễ sử dụng
- Có sẵn nhiều kích cỡ và công suất



Hình 4.13 Động cơ

4.3.6 Sơ đồ kết nối phần cứng của demo



Hình 4.14 Sơ đồ kết nối phần cứng của demo

4.5 Thực nghiệm kiểm chứng

Để kiểm chứng tính đúng đắn của chương trình ứng dụng demo có sử dụng mã nguồn được tạo từ AUTOSAR RTE generator, các testcase phục vụ cho việc kiểm tra được thiết kế như sau:

4.5.1 Testcase 1: Kiểm tra giá trị của tín hiệu điều khiển mà ASW2 nhận được từ ASW1

- Hành động 1: Trên giao diện web nhấn nút tiến thẳng.
 - Kết quả mong muốn 1: 8bit giá trị từ bit8 đến bit15 của ASW2_RP1_VDP_ASW1_Var1 nhận được là 0.
- Hành động 2: Trên giao diện web nhấn nút quay phải
 - Kết quả mong muốn 2: 8bit giá trị từ bit8 đến bit15 của ASW2_RP1_VDP_ASW1_Var1 nhận được là 1.
- Hành động 3: Trên giao diện web nhấn nút lùi
 - Kết quả mong muốn 3: 8bit giá trị từ bit8 đến bit15 của ASW2_RP1_VDP_ASW1_Var1 nhận được là 2.
- Hành động 4: Trên giao diện web nhấn nút quay trái
 - Kết quả mong muốn 4: 8bit giá trị từ bit8 đến bit15 của ASW2_RP1_VDP_ASW1_Var1 nhận được là 3.

4.5.2 Testcase 2: Kiểm tra giá trị của tốc độ động cơ mà ASW2 nhận được từ ASW1

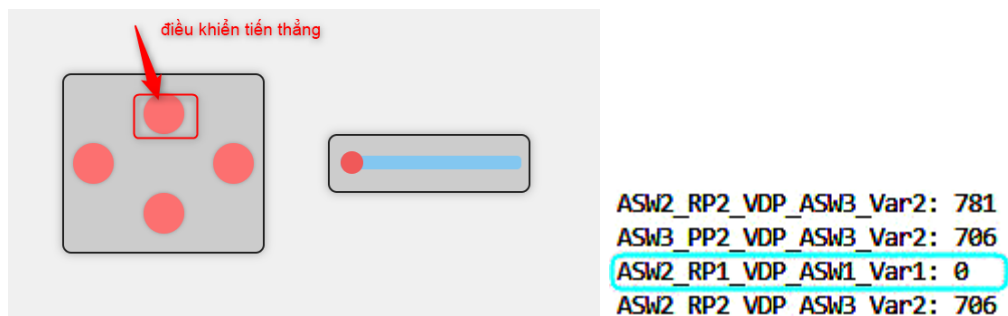
- Hành động 1: Trên giao diện web cài đặt tốc độ là 0.
 - Kết quả mong muốn 1: 8bit giá trị từ bit0 đến bit7 của ASW2_RP1_VDP_ASW1_Var1 nhận được là 0.
- Hành động 2: Trên giao diện web cài đặt tốc độ là 100.
 - Kết quả mong muốn 2: 8bit giá trị từ bit0 đến bit7 của ASW2_RP1_VDP_ASW1_Var1 nhận được là 100.

4.5.3 Testcase 3: Kiểm tra giá trị của cảm biến khoảng cách mà ASW2 nhận được từ ASW3

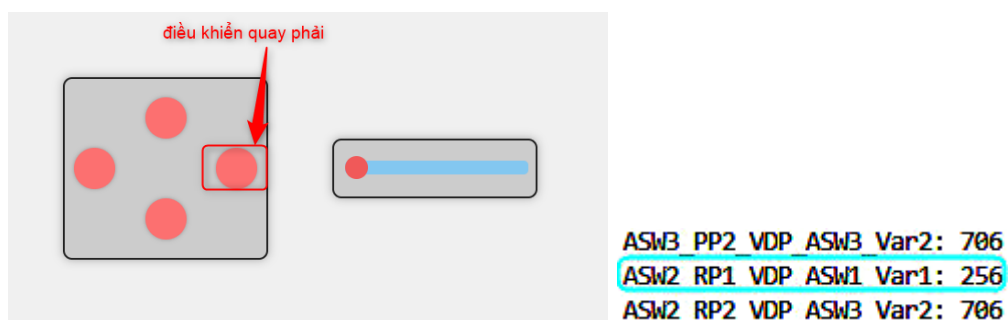
- Hành động 1: Đặt đuôi xe cách vật cản phía sau xe 200mm.
 - o Kết quả mong muốn 1: Giá trị của ASW2_RP2_VDP_ASW3_Var2 nhận được là 200.
- Hành động 2: Đặt đuôi xe cách vật cản phía sau xe 500mm.
 - o Kết quả mong muốn 2: Giá trị của ASW2_RP2_VDP_ASW3_Var2 nhận được là 500.

4.5.4 Kết quả:

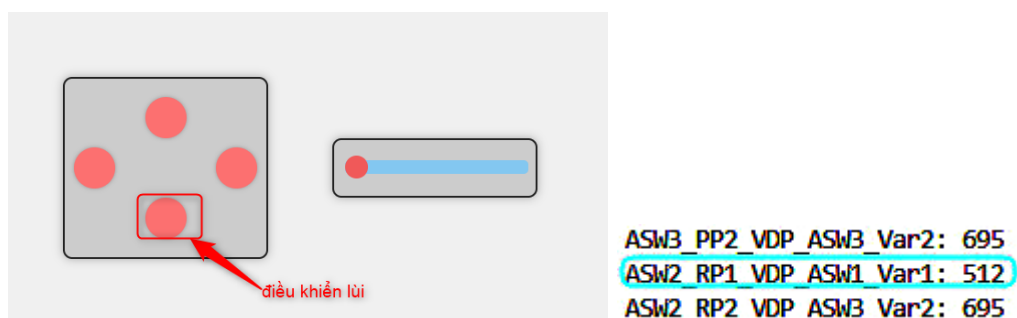
- Testcase1 – hành động 1: Đạt



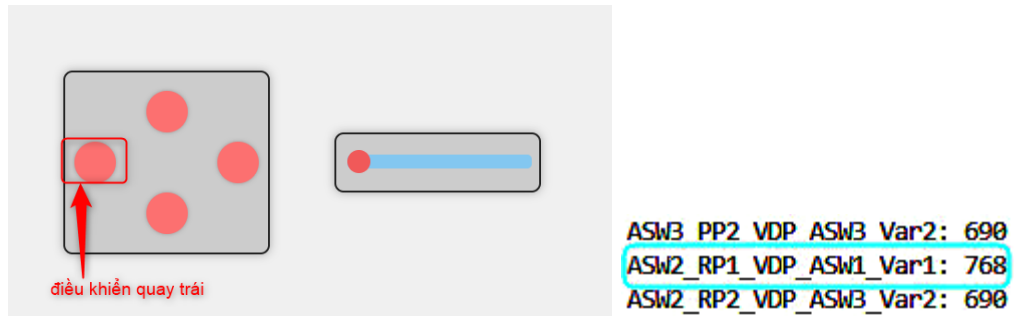
- Testcase1 – hành động 2: Đạt



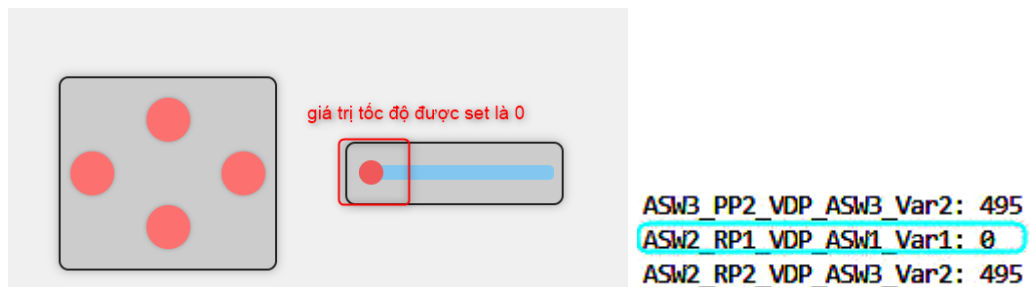
- Testcase1 – hành động 3: Đạt



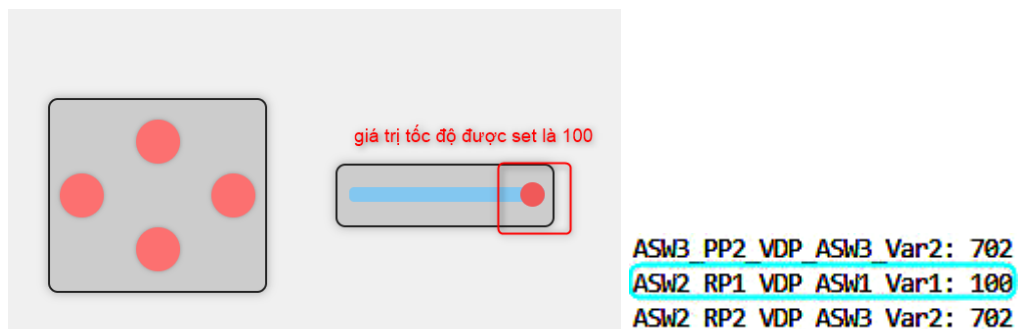
- Testcase1 – hành động 4: Đạt



- Testcase 2 – hành động 1: Đạt



- Testcase 2 – hành động 2: Đạt



- Testcase 3 – hành động 1: Đạt

ASw3_PP2_VDP_ASw3_Var2: 200
ASw2_RP1_VDP_ASw1_Var1: 0
ASw2_RP2_VDP_ASw3_Var2: 200

- Testcase 2 – hành động 2: Đạt

ASw3_PP2_VDP_ASw3_Var2: 500
ASw2_RP1_VDP_ASw1_Var1: 0
ASw2_RP2_VDP_ASw3_Var2: 500

KẾT LUẬN VÀ KIẾN NGHỊ

1. Kết luận

Việc thiết kế và triển khai AUTOSAR RTE Generator đánh dấu một cột mốc quan trọng trong lĩnh vực phát triển phần mềm nhúng ô tô. Nghiên cứu này bắt tay vào việc phát triển mã nguồn mở cho các công cụ phần mềm AUTOSAR RTE generator, giúp tất cả các nhà phát triển dễ tiếp cận hơn và có giá cả phải chăng hơn. Những nỗ lực toàn diện của nghiên cứu đã đạt được kết quả nhất định trong việc hiện thực hóa một công cụ vượt qua các rào cản, mở ra tiềm năng phát triển phần mềm ô tô hiệu quả và tiết kiệm chi phí hơn.

Đáp ứng tính cấp bách của phần mềm ô tô:

Không hề phóng đại tính cấp thiết của việc giải quyết nhu cầu về Kỹ sư phần mềm nhúng có tay nghề cao trong lĩnh vực ô tô. Như được chứng minh trong các báo cáo từ LinkedIn, vai trò của Kỹ sư phần mềm nhúng luôn được xếp hạng trong số 10 ngành nghề có nhu cầu cao nhất. Hơn nữa, sự gia tăng nhu cầu về các kỹ năng liên quan đến AUTOSAR, thể hiện rõ ở mức tăng +113,0% về nhu cầu kỹ năng trong lĩnh vực ô tô nhúng, nhấn mạnh sự phụ thuộc ngày càng tăng của ngành vào các tiêu chuẩn AUTOSAR.

Thu hẹp khoảng cách kỹ năng:

Động lực đằng sau việc bắt tay vào nghiên cứu này bắt nguồn từ mong muốn thu hẹp khoảng cách kỹ năng hiện đang cản trở khả năng tiếp cận các công cụ tạo AUTOSAR RTE. Các giải pháp hiện tại thường cực kỳ tốn kém và phức tạp, cản trở nguyện vọng của nhiều nhà phát triển tài năng tham gia phát triển phần mềm ô tô. Với AUTOSAR RTE generator, nghiên cứu đặt mục tiêu giải quyết sự chênh lệch này, cung cấp giải pháp thay thế thân thiện với người dùng và tiết kiệm chi phí.

Thành tựu chính:

AUTOSAR RTE generator mã nguồn mở: Một tính năng xác định của AUTOSAR RTE generator của nghiên cứu này là tính chất nguồn mở. Bằng cách cung cấp miễn phí mã nguồn cho cộng đồng nhà phát triển, nghiên cứu này đã trao quyền cho các kỹ sư phần mềm ô tô đầy tham vọng nắm quyền kiểm soát các dự án của họ. Sự cởi mở này thúc đẩy văn hóa hợp tác, đổi mới và hỗ trợ ngang hàng hứa hẹn sẽ thúc đẩy lĩnh vực này tiến lên. Cụ thể trong phạm vi nghiên cứu này thì công cụ sẽ hỗ trợ sinh mã cụ thể chi tiết sau:

- API để đọc/ghi dữ liệu cho từng Port của Software Component. Trong nghiên cứu này, loại Port được hỗ trợ cho việc sinh mã là Sender-Receiver Port. Loại kết nối Port được sử dụng là kết nối dạng Assembly. Kiểu truy cập dữ liệu là ghi trực tiếp và đọc trực tiếp không thông qua vùng nhớ tạm, với API đọc dữ liệu thì giá trị sẽ trả về trực tiếp.
- Kiểu dữ liệu dạng ImplementDatatype hiện tại được hỗ trợ trong quá trình sinh mã bao gồm: uint8, uint16 và uint32.
- Hỗ trợ sinh mã cho các Internal Behavior theo Timing Event, các Event có thể được đăng ký vào nhiều OS Task khác nhau và có thứ tự được xác định.
- Mã code được sinh ra và chương trình ứng dụng trên Application Software Component có thể chạy trên vi điều khiển ESP32.
- Mã nguồn của công cụ RTE Generator được công khai trên github:
https://github.com/VoLinhTruc/open_rte_generator

Với những kết quả như trên, sản phẩm của nghiên cứu này – công cụ RTE Generator phù hợp cho mục đích học tập, huấn luyện cũng như chia sẻ kiến thức về các Software component trong kiến trúc phần mềm AUTOSAR. Tác giả không khuyến khích sử dụng công cụ trong giai đoạn vừa kết thúc nghiên cứu cho các mục đích thương mại, các ứng dụng có yêu cầu tính an toàn vì tính tới thời điểm báo cáo đề tài, công cụ vẫn chưa thể đáp ứng được những nhu cầu này. Quá trình cập nhật và nâng cấp sẽ được diễn ra liên tục tùy vào tình hình thực tế. Mọi thông tin về công cụ sẽ được cập nhật liên tục ở trang github của dự án: https://github.com/VoLinhTruc/open_rte_generator

2. Kiến nghị

Với tính năng được hỗ trợ và hạn chế được nêu ra ở mục 2.4.1, các nhà phát triển sử dụng AUTOSAR RTE generator của nghiên cứu này cần lưu ý đến các tính năng có thể sử dụng, ngoài ra có thể dựa vào mã nguồn mở hiện tại của nghiên cứu để phát triển thêm các tính năng phục vụ cho nhu cầu cụ thể của nhà phát triển, tài liệu mô tả chi tiết về AUTOSAR RTE được autosar.org phát hành công khai trên mạng internet [5], các nhà phát triển có thể tham khảo thêm các tính năng và mô tả cụ thể các tính năng đó của AUTOSAR RTE trong tài liệu này.

TÀI LIỆU THAM KHẢO

- [1] S. Piao *et al.* “Design and implementation of RTE generator for automotive embedded software.” present at *Conf. Softw. Eng. Res. Manag. Appl. SERA09, proc. - 7th ACIS Int*, pp. 1-2, 2009, doi: 10.1109/SERA.2009.35.
- [2] Prakash Kumar. “Autosar Architecture (Learn from Scratch with Demo) - Lesson 22 RTE Generator,” 2023. Internet: <https://www.udemy.com/course/autosar-architecture/> (accessed Mar. 06, 2023).
- [3] vnexpress.net. “Các hãng xe lớn cần nhân sự ngành Automotive.” *Vnexpress.net*, 2020. Internet: <https://vnexpress.net/cac-hang-xe-lon-can-nhan-su-nganh-automotive-4180735.html>
- [4] AUTOSAR. “AUTOSAR Layered Software Architecture,” *Autosar*. Vol. 4.3.1, pp. 163, 2017.
- [5] AUTOSAR. “Specification of RTE 4.2.2,” no. January, pp. 1–16, 2015.
- [6] W. Nakano *et al.* “Full Hardware Implementation of FreeRTOS-Based Real-Time Systems,” present at the *Conf. TENCON*. Vol. 2021-Decem, pp. 435–440, 2021, doi: 10.1109/TENCON54134.2021.9707328.
- [7] 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>
- [8] R. Barry. “Mastering the FreeRTOS™ Real Time Kernel.” Vol. 161204 Edition, no. Real Time Engineers Ltd, p. 44, 2016.
- [9] M. Language *et al.* “A Technical Introduction to XML.” Vol. 1, no. Norman Walsh, pp. 1-2, 1997.
- [10] E. Uzun *et al.* “Comparison of Python Libraries used for Web Data Extraction,” *J. Tech. Univ. - Sofia Plovdiv branch, Bulg.*, vol. 24, no. May, pp. 87–92, 2018, [Online]. Available: https://erdincuzun.com/wp-content/uploads/download/plovdiv_journal_2018_01.pdf
- [11] N. Nurseitov *et al.* “Comparison of JSON and XML data interchange formats: A case study,” present at the *Conf. Comput. Appl. 22nd Int. Ind. Eng. 2009, CAINE 2009*, pp. 157–162, 2009.
- [12] C. Moore. “Recursion theory on the reals and continuous-time computation,” *Theor. Comput. Sci.* Vol. 162, no. 1, pp. 23–44, 1996, doi: 10.1016/0304-3975(95)00248-0.

LÝ LỊCH TRÍCH NGANG CỦA HỌC VIÊN

I. LÝ LỊCH SƠ LƯỢC:

Họ và tên: Võ Linh Trúc

Giới tính: Nam

Ngày, tháng, năm sinh: 26/01/1997

Nơi sinh: Bình Thuận

Email: vo.linh.truc@gmail.com

Điện thoại: 0388488608

II. QUÁ TRÌNH ĐÀO TẠO:

- 08/2015 – 06/2019: Học đại học tại trường Đại Học Công Nghiệp TPHCM
- 08/2020 – nay: Học cao học tại trường Đại Học Công Nghiệp TPHCM

III. QUÁ TRÌNH CÔNG TÁC CHUYÊN MÔN:

Thời gian	Nơi công tác	Công việc đảm nhiệm
09/2019 – 05/2021	CÔNG TY TNHH PERITEC	Kỹ thuật viên
05/2021 - nay	Bosch Global Software Technologies Company Limited	Kỹ sư lập trình nhúng

Tp. HCM, ngày tháng Năm 20...

Người khai

Võ Linh Trúc