

**ĐẠI HỌC QUỐC GIA TP. HCM  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**



**BÁO CÁO  
ĐỒ ÁN TỐT NGHIỆP**

**SOFTWARE OVER-THE-AIR (OTA) UPDATE  
FOR PASSENGER CARS**

**NGÀNH: KỸ THUẬT MÁY TÍNH**

**HỘI ĐỒNG:** Hội đồng 4 Kỹ thuật Máy tính

**GVHD:** PGS. TS. Phạm Quốc Cường

**GVPB:** TS. Phạm Hoàng Anh

**SVTH1:** Nguyễn Hoàng Phúc - 1911868

**SVTH2:** Dương Thanh Thương - 1912181

**Thành phố Hồ Chí Minh - tháng 06/2023**



# LỜI CAM ĐOAN

Chúng em xin cam đoan rằng, đề tài "Software over-the-air (OTA) Update for passenger car" được trình bày trong đồ án tốt nghiệp này là của chính chúng em dưới sự hướng dẫn của Thầy/Cô hướng dẫn.

Chúng em cam đoan rằng các kết quả, số liệu và thông tin được trình bày trong luận văn này là trung thực, chính xác và được thu thập từ các nguồn đáng tin cậy.

Chúng em cũng cam đoan rằng chúng em đã không sao chép hoặc sử dụng lại bất kỳ nội dung nào từ các nguồn khác mà không được trích dẫn đầy đủ và chính xác trong đồ án tốt nghiệp này này.

Chúng em hiểu rõ rằng bất kỳ hành vi vi phạm nghiêm trọng như sao chép hay lạm dụng nội dung của người khác trong đồ án tốt nghiệp này sẽ làm mất đi tính trung thực và đạo đức khoa học của đồ án tốt nghiệp và chúng em sẽ chịu trách nhiệm trước pháp luật về hành vi vi phạm tác quyền.

Chúng em xin chân thành cảm ơn sự hỗ trợ và giúp đỡ của các Thầy/Cô, bạn bè và gia đình trong suốt quá trình nghiên cứu và hoàn thành luận văn tốt nghiệp này.



# LỜI CẢM ƠN

Đầu tiên, chúng em xin cảm ơn chân thành đến Thầy Phạm Quốc Cường đã hướng dẫn và tận tình giúp đỡ, góp những ý kiến hay, bổ ích để nhóm hoàn thành được đồ án này.

Xin gửi lời cảm ơn chân thành nhất đến toàn bộ quý thầy cô Trường Đại học Bách Khoa, Quý thầy cô khoa Khoa học và Kỹ thuật Máy tính đã truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập và rèn luyện tại trường.

Chúng em xin chân thành cảm ơn anh Trần Đắc Trịnh, anh Hồ Đức Huy, anh Bùi Khánh Văn và các anh, chị trong Công ty Bosch Global Software Technologies Vietnam đã hướng dẫn và tạo điều kiện thuận lợi cho chúng em trong suốt quá trình thực hiện đồ án. Việc được tiếp xúc thực tế, được giải đáp thắc mắc giúp chúng em có thêm hiểu biết, kiến thức thực tế để thực hiện đồ án này.

Do giới hạn kiến thức và khả năng lý luận của bản thân còn nhiều thiếu sót và hạn chế, kính mong sự chỉ dẫn và đóng góp của các Thầy, Cô để đồ án được hoàn thiện hơn. Đó sẽ là hành trang quý giá để chúng em có thể hoàn thiện mình sau này.

Chúng em xin chân thành cảm ơn!



# LỜI NÓI ĐẦU

Chúng em xin chân thành cảm ơn trường Đại học Bách Khoa- Đại học quốc gia TP Hồ Chí Minh và công ty Công ty Bosch Global Software Technologies Vietnam đã tạo điều kiện cho chúng em thực hiện đề tài nghiên cứu của mình.

Ban đầu khi nhận đề tài "Software over-the-air (OTA) Update for passenger car" và tiến hành làm đề cương thì nhóm có 4 thành viên tham gia thực hiện đề tài bao gồm:

- Nguyễn Hoàng Phúc - MSSV: 1911868
- Dương Thanh Thương - MSSV: 1912181
- Nguyễn Lê Thanh Lộc - MSSV: 1911531
- Nguyễn Lê Anh Khoa - MSSV: 1910272

Sau khi kết thúc giai đoạn 1 - Đề cương đồ án tốt nghiệp thì 2 bạn Nguyễn Lê Anh Khoa và Nguyễn Lê Thanh Lộc không đủ điều kiện để đăng ký tiếp môn học Đồ án tốt nghiệp, tuy nhiên do đây là một dự án kết hợp giữa trường Đại học Bách Khoa và công ty Bosch Global Software Technologies Vietnam nên 2 bạn không thể rút khỏi đề tài, vì vậy thầy hướng dẫn đã cho phép 2 bạn tiếp tục đề tài.

Vì những lí do trên, Đồ án lần này sẽ chỉ tập trung vào phần hiện thực kiến trúc chi tiết, quá trình flashing từ Raspberry Pi 4B đến thiết bị, hiện thực một phần GUI trên xe, mô hình KIT thực tế do bạn bạn Nguyễn Hoàng Phúc và bạn Dương Thanh Thương sẽ chịu trách nhiệm chính.

Bạn Nguyễn Lê Thanh Lộc và bạn Nguyễn Lê Anh Khoa sẽ chịu trách nhiệm chính cho phần Bosch IoT Rollouts, API tương tác với Bosch IoT Rollouts, Giao diện GUI và Mobile APP tương tác với người dùng.

Vì để cho bài báo cáo hoàn thiện, tổng quát và dễ hiểu, các phần do bạn Nguyễn Lê Thanh Lộc và Nguyễn Lê Anh Khoa sẽ được giới thiệu một cách khái quát và một phần kết quả sẽ được đưa vào báo cáo lần này.

Tuy gặp nhiều sự cố phát sinh nhưng nhóm vẫn rất cố gắng để nghiên cứu và hoàn

thành báo cáo một cách hoàn thiện nhất. Chúng em xin chân thành cảm ơn, quý Thầy/Cô đã cảm thông và tạo điều kiện thuận lợi nhất cho chúng em hoàn thành Đồ án Tốt Nghiệp.

# TÓM TẮT

Ưu điểm về tiết kiệm thời gian, công sức và chi phí khiến việc cập nhật phần mềm qua mạng ngày càng phổ biến đối với người dùng và các công ty.

Các bản cập nhật phần mềm qua mạng, hay SOTA (Software Over-the-air), là các bản nâng cấp có thể được tải xuống từ xa qua internet mà người dùng không cần phải mang thiết bị của họ trở lại nhà sản xuất hoặc cơ sở hỗ trợ. SOTA trở thành một giải pháp mạnh mẽ, bền bỉ, ổn định, tiết kiệm thời gian và công sức.

Trong lĩnh vực ô tô, một số OEM (nhà sản xuất thiết bị gốc) cung cấp dịch vụ nâng cấp chương trình cơ sở qua mạng để cập nhật phần mềm cho hàng nghìn, thậm chí hàng triệu xe cùng một lúc. Mặc dù thực tế rằng cập nhật phần mềm OTA không phải là một ý tưởng mới, nhưng mục đích của đồ án này là đưa lý thuyết OTA vào thực tế bằng cách sử dụng các thiết bị và dữ liệu thực hiện đang được sử dụng trong ngành công nghiệp ô tô. Đồ án này mô tả cách nâng cấp chương trình ECU, bao gồm tải xuống và cập nhật phần mềm mới cho ECU thông qua mạng giao tiếp trên xe như CAN và sử dụng giao thức UDS.



# MỤC LỤC

<b>Lời cam đoan</b>	<b>iii</b>
<b>Lời cảm ơn</b>	<b>v</b>
<b>Lời nói đầu</b>	<b>vii</b>
<b>Tóm tắt</b>	<b>ix</b>
<b>Danh mục từ viết tắt</b>	<b>xiii</b>
<b>Danh sách hình vẽ</b>	<b>xv</b>
<b>Danh sách bảng</b>	<b>xix</b>
<b>1 Mở đầu</b>	<b>1</b>
1.1 Giới thiệu đề tài . . . . .	1
1.2 Giới hạn của đề tài. . . . .	1
1.3 Động lực chọn đề tài. . . . .	3
<b>2 Cơ sở lý thuyết</b>	<b>5</b>
2.1 ECU . . . . .	5
2.2 Controller Area network - CAN. . . . .	6
2.2.1 Tổng quát về CAN . . . . .	6
2.2.2 CAN Frame . . . . .	7
2.2.3 CAN Transport Protocol (CAN-TP) . . . . .	9
2.3 UDS Protocol . . . . .	14
2.3.1 Tổng quan . . . . .	14
2.3.2 UDS Service cho cập nhật phần mềm . . . . .	17
2.4 Bosch IoT Rollouts . . . . .	48
2.4.1 Tổng quan . . . . .	49
2.4.2 Chức năng. . . . .	50
2.4.3 Các tính năng cơ bản [8] . . . . .	51
2.5 Central Gateway Module . . . . .	59
2.6 Raspberry Pi . . . . .	61
2.6.1 Khái niệm . . . . .	61
2.6.2 Phân loại Raspberry Pi . . . . .	61

---

2.7	Dynamic view trong hệ thống ô tô . . . . .	64
2.8	Phương án cập nhật phần mềm Over The Air . . . . .	68
2.9	Các sự cố khi cập nhật OTA. . . . .	72
<b>3</b>	<b>Đề xuất hệ thống</b>	<b>77</b>
3.1	Phân tích hệ thống . . . . .	77
3.1.1	Phân tích yêu cầu . . . . .	77
3.1.2	Phân tích rủi ro . . . . .	79
3.2	Kiến trúc tổng quan . . . . .	80
3.2.1	Kiến trúc hệ thống OTA cơ bản . . . . .	80
3.2.2	Các thành phần cơ bản trong hệ thống . . . . .	81
3.2.3	Chiến lược cập nhật OTA. . . . .	83
3.3	Kiến trúc chi tiết từng phần . . . . .	85
3.3.1	Các Khối chức năng trong hệ thống . . . . .	85
3.3.2	Update package creation: OEM web server . . . . .	85
3.3.3	Update package distribution: Bosch IoT Rollouts . . . . .	85
3.3.4	Update package installation : Raspberri Pi 4B. . . . .	90
3.3.5	Giao diện người dùng : GUI . . . . .	93
3.3.6	Bảo mật của hệ thống. . . . .	94
<b>4</b>	<b>Hiện thực và Kết quả</b>	<b>97</b>
4.1	Hiện thực. . . . .	97
4.1.1	Mô hình hệ thống . . . . .	97
4.1.2	Bosch IoT Rollouts . . . . .	100
4.1.3	Giao diện người dùng: GUI. . . . .	104
4.1.4	Quá Trình Flashing của Raspberry Pi 4B . . . . .	112
4.2	Kết quả. . . . .	115
4.3	Đánh giá hệ thống . . . . .	117
<b>5</b>	<b>Kết luận</b>	<b>119</b>
	<b>Tài liệu tham khảo</b>	<b>121</b>

# DANH MỤC TỪ VIẾT TẮT

1. API: Application Programming Interface
2. AUTOSAR: Automotive Open System Architecture
3. ASW: Application Software
4. BSW: Basic Software
5. CAN: Controller Area Network
6. CAN FD: CAN Flexible Data-rate
7. CB: CustomerBoot
8. CCU: Connectivity Control Unit
9. ECU: Engine Control Unit hay Electronic Control Unit
10. E/E: electrical-electronic
11. FBL: Flash Bootloader
12. FOTA: Firmware Over The Air
13. FW: Firmware
14. GPS: Global Positioning System
15. GUI: Graphical User Interface
16. HSM: Hardware Security Module
17. HW: Hardware
18. IoT: Internet of Things
19. LIN: Local Interconnect Network
20. MCAL: Microcontroller Abstraction Layer
21. OEM: Original Equipment Manufacturer

22. OTA: Over-the-air
23. RTE: Runtime Environment
24. SB: StartupBlock
25. SOTA: Software over the air
26. SW: Software
27. UDS: Unified Diagnostic Services
28. VIN: Vehicle Identification Number

# DANH SÁCH HÌNH VẼ

1.1	<i>Tổng quan về tài [8]</i>	1
2.1	<i>ECU</i>	6
2.2	<i>Hệ thống CAN</i>	7
2.3	<i>Khung CAN tiêu chuẩn</i>	8
2.4	<i>CAN - TP Header [2]</i>	10
2.5	<i>Example Single Frame</i>	10
2.6	<i>Flow Multi Frame</i>	11
2.7	<i>Flow Control Frame</i>	12
2.8	<i>Example Multi Frame</i>	13
2.9	<i>Example Multi Frame (cont)</i>	13
2.10	<i>UDS Request - Response</i>	14
2.11	<i>Positive Response</i>	15
2.12	<i>Negative Response</i>	16
2.13	<i>Response Suppressed</i>	16
2.14	<i>Bosch IoT Cloud</i>	48
2.15	<i>Bosch IoT Suite</i>	49
2.16	<i>Bosch IoT Rollouts</i>	50
2.17	<i>Deployment</i>	54
2.18	<i>Rollout List</i>	54
2.19	<i>Rollout Group</i>	55
2.20	<i>Rollout Creation</i>	55
2.21	<i>Target Filters</i>	56
2.22	<i>Upload</i>	56
2.23	<i>User Management</i>	57
2.24	<i>Role Management</i>	57
2.25	<i>Statistics</i>	58
2.26	<i>System Configuration</i>	58
2.27	<i>Minh họa về mô hình Central Gateway</i>	60
2.28	<i>Kiến trúc mạng ô tô ngày nay [6]</i>	61
2.29	<i>Raspberry Pi 4B</i>	63
2.30	<i>Typical FBL in ECU SW Architecture</i>	65
2.31	<i>Start up Processes</i>	66

2.32 Các phân vùng trong ECU . . . . .	67
2.33 Minh họa cơ chế OTA . . . . .	68
2.34 Phân loại OTA . . . . .	70
2.35 Một số rủi ro khi cập nhật OTA . . . . .	73
2.36 Bắt đầu quá trình flash không đúng lúc . . . . .	73
2.37 Quá trình flash chưa hoàn thành . . . . .	74
2.38 Phiên bản phần mềm không tương thích . . . . .	75
2.39 Phụ thuộc không tương thích . . . . .	75
3.1 Usecase các chức năng cơ bản của hệ thống OTA . . . . .	78
3.2 Kiến trúc hệ thống OTA cơ bản . . . . .	80
3.3 Sơ đồ kiến trúc tổng quan . . . . .	82
3.4 Các giai đoạn của Chiến lược cập nhật OTA Base . . . . .	83
3.5 Luồng hoạt động cơ bản của hệ thống . . . . .	84
3.6 Sơ đồ chức năng của các khối trong hệ thống . . . . .	85
3.7 Update package distribute function: Activity Diagram . . . . .	86
3.8 Update information for car . . . . .	87
3.9 Upload software for car . . . . .	88
3.10 Control Campaign . . . . .	89
3.11 Flowchart Raspberry Pi 4B . . . . .	90
3.12 General procedure flow[14] . . . . .	92
3.13 UDS Download Sequence . . . . .	93
3.14 Minh họa cơ chế OTA . . . . .	95
3.15 Security Flashing Flow [14] . . . . .	96
4.1 Connection between devices . . . . .	98
4.2 Mô hình hệ thống . . . . .	98
4.3 Mô hình KIT . . . . .	99
4.4 Giao diện đăng nhập . . . . .	100
4.5 Trang Deployment Management . . . . .	101
4.6 Trang Upload . . . . .	101
4.7 Trang Distribution . . . . .	102
4.8 Trang Target Filter . . . . .	102
4.9 Trang Rollout . . . . .	103
4.10 Trang Rollout - giám sát các thiết bị cập nhật . . . . .	103
4.11 Trang Statistics . . . . .	104
4.12 GUI FLow . . . . .	106
4.13 Display Pages with GUI functions . . . . .	107
4.14 GUI Functions Interaction . . . . .	107
4.15 Giao diện starting . . . . .	108

4.16	<i>Trang Home</i>	108
4.17	<i>Checking for OTA updates...</i>	108
4.18	<i>Version is up to date</i>	109
4.19	<i>New version needs to update</i>	109
4.20	<i>Software downloaded</i>	109
4.21	<i>Software detail</i>	110
4.22	<i>Downloading...</i>	110
4.23	<i>Download successfully</i>	110
4.24	<i>Updating...</i>	111
4.25	<i>Update successfully</i>	111
4.26	<i>Sơ đồ các khôi chức năng trong hàm flashing</i>	112
4.27	<i>UDS Download Sequence Processes [15]</i>	113
4.28	<i>Update software Log in reality part 1</i>	114
4.29	<i>Update software Log in reality part 2</i>	115
4.30	<i>Result Table</i>	117



# DANH SÁCH BẢNG

2.1 Request Message Structure Service \$10 . . . . .	18
2.2 Positive Response Structure Service \$10 . . . . .	18
2.3 Example Positive Response Structure Service \$10 . . . . .	19
2.4 Negative Response Structure Service \$10 . . . . .	19
2.5 Supported NRC Service \$10 . . . . .	19
2.6 Example Negative Response Structure Service \$10 . . . . .	20
2.7 Request Message Structure Service \$27 - Request Seed . . . . .	21
2.8 Request Message Structure Service \$27 - Send Key . . . . .	21
2.9 Positive Response Structure Service \$27 . . . . .	22
2.10 Example Positive Response Structure Service \$27 . . . . .	22
2.11 Negative Response Structure Service \$27 . . . . .	23
2.12 Supported NRC Service \$27 . . . . .	24
2.13 Request Message Structure Service \$31 . . . . .	25
2.14 Positive Response Structure Service \$31 . . . . .	26
2.15 Example Erase Memory (0x31 + 0xFF00) . . . . .	26
2.16 Example Verification (0x31 + 0x01FF01) . . . . .	27
2.17 Negative Response Structure Service \$31 . . . . .	27
2.18 Supported NRC Service \$31 . . . . .	28
2.19 Request Message Structure Service \$34 . . . . .	29
2.20 Positive Response Structure Service \$34 . . . . .	31
2.21 Example Positive Response Structure Service \$34 . . . . .	31
2.22 Negative Response Structure Service \$34 . . . . .	32
2.23 Supported NRC Service \$34 . . . . .	33
2.24 Request Message Structure Service \$36 . . . . .	34
2.25 Positive Response Structure Service \$36 . . . . .	34
2.26 Example Positive Response Structure Service \$36 . . . . .	35
2.27 Negative Response Structure Service \$34 . . . . .	35
2.28 Supported NRC Service \$34 . . . . .	36
2.29 Request Message Structure Service \$37 . . . . .	37
2.30 Positive Response Structure Service \$37 . . . . .	37
2.31 Example Positive Response Structure Service \$37 . . . . .	37
2.32 Negative Response Structure Service \$37 . . . . .	38
2.33 Supported NRC Service \$37 . . . . .	38

2.34 Request Message Structure Service \$11 . . . . .	39
2.35 Positive Response Structure Service \$11 . . . . .	39
2.36 Example Positive Response Structure Service \$11 . . . . .	39
2.37 Negative Response Structure Service \$11 . . . . .	40
2.38 Supported NRC Service \$11 . . . . .	40
2.39 Example Negative Response Structure Service \$10 . . . . .	40
2.40 Request Message Structure Service \$22 . . . . .	42
2.41 Positive Response Structure Service \$22 . . . . .	43
2.42 Service \$22 - Read VIN from ECU . . . . .	43
2.43 Negative Response Structure Service \$22 . . . . .	44
2.44 Supported NRC Service \$22 . . . . .	44
2.45 Request Message Structure Service \$2E . . . . .	45
2.46 Positive Response Structure Service \$2E . . . . .	45
2.47 Service \$2E - Write VIN to ECU . . . . .	46
2.48 Negative Response Structure Service \$2E . . . . .	47
2.49 Supported NRC Service \$2E . . . . .	47
2.50 Bảng các loại Raspberry Pi [12] . . . . .	64
2.51 Tổng hợp phân loại OTA . . . . .	72
4.1 Danh sách thiết bị . . . . .	97

# 1

## MỞ ĐẦU

### 1.1. GIỚI THIỆU ĐỀ TÀI



Hình 1.1: *Tổng quan đề tài* [8]

Việc cập nhật phần mềm cho phương tiện giao thông thông qua mạng kết nối không dây đang trở thành xu hướng ngày nay. Phương pháp này giúp tiết kiệm thời gian, công sức và chi phí cho doanh nghiệp và người sử dụng.

Cập nhật OTA (Over the air) hay còn được gọi là SOTA (Software over-the-air). SOTA lấy ý tưởng từ việc tải và flash phần mềm mới vào ECU nhưng không cần dùng các công cụ chuẩn đoán như cách truyền thống mà phần mềm mới sẽ được tải thông qua mạng không dây và được flash vào ECU thông qua mạng giao tiếp trên xe như CAN, CAN FD,...

### 1.2. GIỚI HẠN CỦA ĐỀ TÀI

Trong đồ án lần này, chúng em sẽ hiện thực đề tài trong phạm vi nghiên cứu và thử nghiệm quy trình cập nhật phần mềm cho xe (Passenger car) cơ bản.

**1****Các bước chính của quy trình cập nhật :**

- Một nền tảng Cloud cơ bản được xây dựng để OEM có thể tải lên phiên bản cập nhật phần mềm mới.
- Sau đó, phương pháp cập nhật phần mềm từ xa (OTA) sẽ được thể hiện qua việc người dùng được thông báo về bản cập nhật và gửi xác nhận cập nhật thông qua giao diện tương tác, từ đó Central Gateway tải về bản cập nhật mới từ Cloud và tiến hành cập nhật cho xe.
- Cuối cùng, những dữ liệu được thu thập như tình trạng phần mềm trong xe sẽ được thu thập và dùng cho việc phân tích, quản lý giúp nâng cao trải nghiệm sử dụng (lái) và độ an toàn của xe.

**Giới hạn về thiết bị :**

- Công ty sẽ hỗ trợ cung cấp 1 ECU của xe đóng vai trò là Target ECU, nhóm sẽ dùng thêm 1 Raspberry Pi 4B đóng vai trò là Central Gateway trong hệ thống.
- Do không có thiết bị mô phỏng tình trạng của động cơ xe nên không thể tiến hành mô phỏng hệ thống xe phục vụ cho việc kiểm thử hệ thống sau này một cách hoàn thiện và thực tế nhất.
- Hiện tại do các giới hạn thiết bị nên đề tài chỉ giới hạn trong việc cập nhật cho 1 ECU.
- ECU cũng không hỗ trợ các vùng nhớ dành riêng cho cập nhật OTA.

**Giới hạn về công nghệ :**

- Giao diện tương tác với người dùng sẽ là Mobile APP và GUI
- Để đảm bảo tính bảo mật dữ liệu của công ty trên Cloud được dùng để quản lý phần mềm cập nhật sẽ sử dụng Bosch IoT Rollouts do công ty Bosch phát triển.
- Các phần mềm cập nhập cho ECU có dung lượng giống nhau, nên không có nhiều kịch bản kiểm thử cho các file có dung lượng khác nhau.

**Giới hạn về thời gian :** Vì đây là đề tài kết hợp với doanh nghiệp, nên các phần chính của đề tài sẽ chỉ được thực hiện tối đa trong 6 tháng tính từ khi bắt đầu nhận đề tài.

### 1.3. ĐỘNG LỰC CHỌN ĐÈ TÀI

Phần mềm đã và đang trở thành một phần cốt yếu trong ngành công nghiệp sản xuất xe. Trong mỗi chiếc xe có một mạng kết nối các ECU rất lớn, trung bình có khoảng từ 70 đến 100 ECU trong một phương tiện hiện đại. Những ECU này giúp tính toán và điều khiển hoạt động của động cơ, điều khiển phanh, giám sát hệ thống điều hòa không khí,... Khi mà các phương tiện càng ngày càng được thêm vào những tính năng mới, điều này làm cho độ phức tạp và kích thước phần mềm trên xe ngày càng lớn, số lượng dòng lệnh có thể lên tới 100 triệu dòng. Một điều mà các lập trình viên đều đồng tình đó là không có phần mềm hữu ích bao gồm hàng ngàn dòng lệnh nào là không có lỗi. Điều bắt buộc là phần mềm dành phương tiện phải có độ tin cậy 100% vì sự cố phần mềm khi lái xe có hậu quả nghiêm trọng hơn nhiều so với sự cố xảy ra khi ai đó ngồi sau bàn làm việc. Vì vậy việc thường xuyên cập nhật phần mềm để khắc phục lỗi cũng như nâng cao hiệu suất trở nên cần thiết hơn bao giờ hết. [7]

Trong những năm gần đây, các vụ thu hồi sản phẩm (Recall) đã gia tăng trên toàn thế giới cả về tần suất và cường độ trong các ngành công nghiệp khác nhau, trong đó các vụ thu hồi xe đã thu hút sự chú ý của công chúng và thường trở thành tin tức quốc tế. Ví dụ như lỗi tăng tốc đột ngột dẫn đến 89 người chết và 57 người bị thương, khiến Toyota phải thu hồi hơn 10 triệu xe trên toàn thế giới vào năm 2010 [10]. Một ví dụ khác là Volkswagen, có cổ phiếu giảm gần 30% vì vụ thu hồi liên quan đến khí thải động cơ Diesel vào năm 2015 do có lỗi về Cabliration [11].

Gần đây hơn, theo tờ Reuters (12/11/2022) - Tesla đang thu hồi hơn 321.000 xe tại Hoa Kỳ vì đèn hậu có thể không sáng liên tục, công ty cho biết trong một hồ sơ công khai hôm thứ Bảy. Thông tin này được đưa ra sau đợt thu hồi gần 30.000 xe Model X tại Hoa Kỳ của công ty vào ngày thứ Sáu do một vấn đề có thể khiến túi khí hành khách phía trước bung không đúng cách, khiến cổ phiếu của hãng giảm gần 3% xuống mức thấp gần hai năm. Trong hồ sơ được công bố hôm thứ Bảy tới Cục Quản lý An toàn Giao thông Đường cao tốc Quốc gia (NHTSA), nhà sản xuất xe điện cho biết đợt thu hồi liên quan đến đèn hậu bao gồm một số xe Model 2023 và Model Y 2020-2023. Tesla có trụ sở tại Texas cho biết họ sẽ triển khai một bản cập nhật bằng phương thức Over the air để khắc phục sự cố đèn hậu.

Các bản cập nhật phần mềm được các nhà sản xuất phương tiện phát hành định kỳ vì nhiều lý do, bao gồm sửa lỗi, cải thiện hiệu suất của phương tiện, thêm các tính năng mới hoặc bảo vệ chống lại các lỗ hổng được phát hiện gần đây có thể cho phép tin tặc truy cập vào phần mềm phương tiện và điều khiển hệ thống phương tiện. Các bản cập nhật phần mềm này không chỉ là “có thì tốt”, mà chúng là “phải

**1**

có” nếu người lái xe muốn các chương trình hoạt theo mặc định - hoạt động bình thường và không bị bên thứ ba phá hoại.

Trước đây, các bản cập nhật phần mềm được phân phối chủ yếu theo một trong hai cách là thông qua ổ đĩa có chứa phần mềm mà hãng gửi đến chủ xe, sau đó chủ sở hữu phải cài đặt ổ đĩa này trên xe hoặc bằng cách mang xe đến cho trung tâm dịch vụ để cài phần mềm mới. Cả hai phương pháp này đều yêu cầu đầu tư thời gian hoặc tiền bạc - hai tài sản mà nhiều tài xế có thể miễn cưỡng chi trả. Tưởng tượng nếu phải cập nhật cho một ngàn chiếc xe thì nhà sản xuất sẽ phải tiêu hàng triệu đô la để xây dựng dịch vụ nhà cho xe, tiền thuê nhân công chỉ để sửa chữa, bảo trì những thứ chỉ cần cập nhật phần mềm.

Việc cập nhật phần mềm là việc cần thiết và rất hữu ích cho người sử dụng tránh được các lỗi không mong muốn, thêm nhiều tính năng , được bảo mật tốt hơn và phương pháp cập nhật phần mềm Over the air sẽ giúp cho cả người dùng lẫn doanh nghiệp tiết kiệm được nhiều thời gian, tiền bạc, công sức khi tiến hành cập nhật phần mềm mới.

Chủ đề Cập nhật phần mềm Over the air có nhiều đóng góp trong ngành công nghiệp sản xuất ô tô, một trong những ngành then chốt trong lĩnh vực Nhúng. Việc triển khai giải pháp cập nhật phần mềm Over the air sẽ phải giải quyết các vấn đề liên quan nhiều đến các vi điều khiển, mạng giao tiếp giữa các vi điều khiển, cách phần mềm được tải, hoạt động trong các vi điều khiển, hơn thế chúng em cũng sẽ phải tìm hiểu cách hoạt động của các kênh lưu trữ đám mây và có cơ hội hiện thực một số giao diện thân thiện với người dùng. Những công việc trên đều cần nền tảng kiến thức mà chúng em đã được học ở những năm trước như thiết kế vi mạch, vi điều khiển, hệ thống nhúng, hệ điều hành, mạng máy tính, lập trình web. Chúng em tin rằng chúng em có thể tổng quát hóa và hệ thống được các kiến thức trọng yếu mà chúng em đã học trong những năm qua và ứng dụng nó vào thực tiễn, mang lại một số lợi ích nhất định sau khi hoàn thành đề tài. Đó là những động lực để chúng em thực hiện đề tài này.

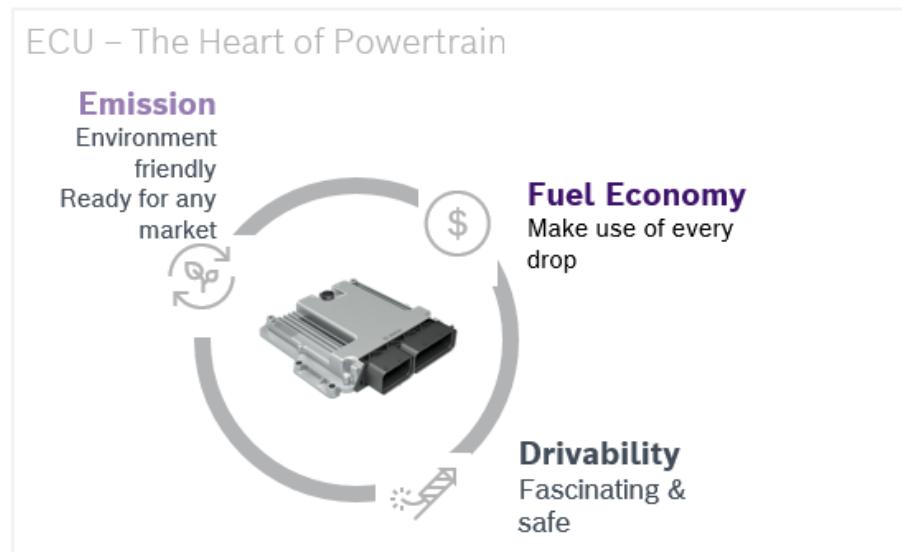
# 2

## CƠ SỞ LÝ THUYẾT

Trong chương này, chúng em sẽ tập trung trình bày cơ sở lý luận, những kiến thức cơ bản, nền tảng phục vụ cho đề tài này.

### 2.1. ECU

ECU là viết tắt của Electronic Control Unit, là một thiết bị điện tử có chức năng điều khiển các hệ thống trên xe ô tô. ECU có thể được lắp đặt trên nhiều hệ thống trên xe, bao gồm hệ thống động cơ, hệ thống phanh, hệ thống lái, hệ thống treo và hệ thống điều hòa... . ECU hoạt động dựa trên các thông số đo được từ các cảm biến trên xe và điều khiển các bộ phận hoạt động đúng theo quy trình đã được lập trình trước đó một cách vô cùng chính xác nhờ đó góp phần quan trọng trong việc cải thiện hiệu suất của các hệ thống trên xe và giảm thiểu khí thải độc hại. ECU cũng có khả năng giao tiếp với toàn bộ hệ thống trên xe đảm bảo tính tương thích và an toàn khi sử dụng xe.

Hình 2.1: *ECU*

ECU - nhận tín hiệu từ các cảm biến khác nhau, thực hiện tính toán và gửi tín hiệu đầu ra để thực hiện các chức năng và hoạt động khác nhau bên trong và xung quanh động cơ. Một số cảm biến phổ biến gửi thông tin đến hệ thống quản lý động cơ là cảm biến vị trí trực khuỷu, cảm biến vị trí trực cam và nhiều cảm biến khác. Từ các thông số hoạt động của động cơ, ECU sẽ tiến hành phân tích và đưa ra quyết định về các điều khiển cần thiết một cách nhanh chóng để đảm bảo hiệu suất tối ưu của động cơ, giảm tiêu thụ nhiên liệu và khí thải, tăng độ bền của động cơ, và đảm bảo an toàn khi sử dụng xe.[4]

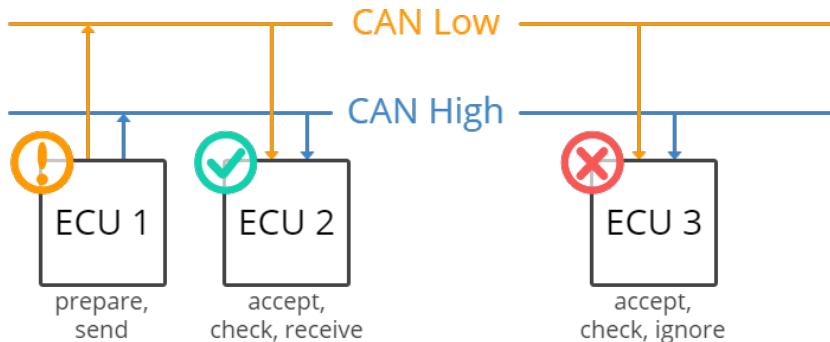
## 2.2. CONTROLLER AREA NETWORK - CAN

### 2.2.1. TỔNG QUÁT VỀ CAN

CAN là một tiêu chuẩn ISO (ISO 11898), chuẩn bus ổn định dùng cho các phương tiện giao thông, đặc biệt là xe hơi. CAN được thiết kế để các vi điều khiển và các thiết bị giao tiếp với nhau mà không cần thông qua máy tính. CAN là một giao thức hoạt động dựa trên các gói tin (message-based protocol), ban đầu được thiết kế để ghép kênh (multiplex) các dây điện trong ô tô để tiết kiệm số lượng dây dẫn, nhưng sau đó được sử dụng rộng rãi hơn trong các hệ thống tự động hóa công nghiệp hay hàng không.

Hệ thống CAN bus được phát triển bởi Bosch vào năm 1986 và nhanh chóng được

chấp nhận trong ngành công nghiệp ô tô và hàng không vũ trụ, đa số dùng trong việc giám sát và đưa ra các cảnh báo liên quan tới một hoặc nhiều chi tiết hệ thống máy móc, điện tử có sẵn trên xe đang hỏng hóc hoặc đang gặp sự cố cho người sử dụng và người sửa chữa. Ngày nay, CAN là tiêu chuẩn trong ô tô (xe hơi, xe tải, xe buýt, máy kéo,...), tàu thủy, máy bay, pin EV, máy móc và hơn thế nữa.

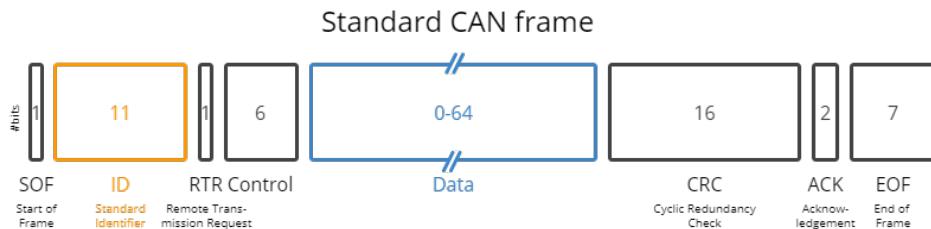


Hình 2.2: Hệ thống CAN

Các “nút” cụ thể là các ECU trên xe được kết nối với nhau bằng mạng CAN bao gồm 2 dây xoắn (dây CAN high và dây CAN low) với điện trở  $120 \Omega$  (cần ít nhất hai nút trên đường truyền CAN để có thể giao tiếp) do đó hệ thống cho phép mỗi ECU giao tiếp với tất cả các ECU khác mà không cần đi dây chuyên dụng phức tạp. Như việc một ECU có thể chuẩn bị và phát thông tin (ví dụ như dữ liệu cảm biến) thông qua CAN. Dữ liệu đã phát được chấp nhận bởi tất cả các ECU khác trên mạng CAN và mỗi ECU sau đó có thể kiểm tra dữ liệu và quyết định nhận hay bỏ qua nó. Việc ECUs giao tiếp thông qua một hệ thống CAN duy nhất thay vì thông qua các đường tín hiệu tương tự phức tạp giúp trực tiếp giảm lỗi, trọng lượng, hệ thống dây điện và chi phí. [1]

### 2.2.2. CAN FRAME

Giao tiếp qua CAN bus được thực hiện thông qua khung CAN (frame). Dưới đây là khung CAN tiêu chuẩn với định danh 11 bit (CAN 2.0A), đây là loại được sử dụng trong hầu hết các xe ô tô. Khung nhận dạng 29-bit mở rộng (CAN 2.0B) giống hệt nhau ngoại trừ ID dài hơn. Nó được sử dụng trong giao thức J1939 cho xe hạng nặng. Các khung CAN được ưu tiên theo ID để dữ liệu ưu tiên hàng đầu nhận được quyền truy cập bus ngay lập tức, mà không gây gián đoạn các khung khác. [1]



Hình 2.3: Khung CAN tiêu chuẩn

### CAN Frame bao gồm 8 trường dữ liệu :

- SOF: Start of Frame là trường đầu tiên đóng vai trò thông báo với các nút khác rằng một nút trong CAN có ý định giao tiếp.
- ID: Là mã nhận dạng của khung CAN – các giá trị thấp hơn có mức độ ưu tiên cao hơn.
- RTR: The Remote Transmission Request (Yêu cầu truyền từ xa) cho biết liệu một nút có vai trò gửi dữ liệu hay yêu cầu dữ liệu được gửi từ một nút khác.
- Control: Bộ điều khiển chứa bit định danh mở rộng (Identifier Extension Bit - IDE). Nó cũng chứa 4 bit là mã độ dài dữ liệu (Data Length Code - DLC) chỉ định độ dài của byte dữ liệu được truyền (0 đến 8 bytes).
- Data: Chứa các byte dữ liệu, bao gồm các tín hiệu CAN có thể được trích xuất và giải mã để lấy thông tin.
- CRC: Kiểm tra dự phòng theo chu kỳ được sử dụng để đảm bảo tính toàn vẹn của dữ liệu.
- ACK: Khe ACK cho biết liệu nút đã nhận được dữ liệu chính xác hay chưa
- EOF: EOF đánh dấu phần cuối của khung CAN.

Hai trường CAN ID và CAN DATA được đánh dấu tô màu trong hình là hai trường quan trọng nhất trong việc đọc ghi dữ liệu CAN Bus.

### 2.2.3. CAN TRANSPORT PROTOCOL (CAN-TP)

CAN Transport Protocol (CAN-TP) là một giao thức truyền tải dữ liệu được sử dụng trong mạng CAN (Controller Area Network). CAN-TP được thiết kế để hỗ trợ truyền tải dữ liệu trong mạng CAN với độ tin cậy cao và hiệu suất tối ưu.

2

CAN-TP cho phép truyền tải dữ liệu theo các giao thức khác nhau, bao gồm giao thức đơn giản (Simple Protocol), giao thức đa năng (Versatile Protocol) và giao thức đồng bộ (Synchronous Protocol). Các giao thức này được thiết kế để đáp ứng các yêu cầu khác nhau của các ứng dụng truyền tải dữ liệu trên mạng CAN.

CAN-TP sử dụng các khái niệm như truyền tải dữ liệu trên nhiều khung (Multi-Frame Transmission), kiểm soát luồng (Flow Control), chuyển tiếp dữ liệu (Data Forwarding) và giám sát tình trạng mạng (Network Monitoring) để đảm bảo truyền tải dữ liệu một cách hiệu quả và đáng tin cậy.

CAN-TP cung cấp các tính năng để kiểm soát lưu lượng dữ liệu và giảm độ trễ truyền tải, đồng thời tăng tính tin cậy của mạng CAN. Vì vậy, CAN-TP là một trong những giao thức phổ biến được sử dụng trong các ứng dụng truyền tải dữ liệu trên mạng CAN, đặc biệt là trong các ứng dụng công nghiệp và xe hơi.

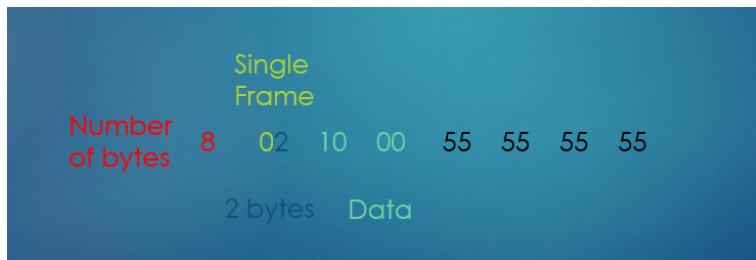
Cấu trúc của một CAN-TP Header quy định việc gửi dữ liệu lên đến 4095 byte.

2

CAN – TP Header						
Frame Type	Byte 0		Byte 1	Byte 2	....	Byte 7
Bit offset	Bit 7 to 4	Bit 3 to 0	Bit 15 to 8	Bit 23 to 16	....	Bit 63 to 56
Single Frame	Type = 0	(0 to 7) Single frame data length	Data A	Data B	....	Data G
First Frame	Type = 1	Data length of multi frame data size (8 to 4095) Byte	Data A	....	Data F	
Consecutive Frame	Type = 2	Sequence number index (0 to 15)	Data A	Data B	....	Data G
Flow Control Frame	Type = 3	Flow Control Flag (0,1,2)	Block size	Separation Time/STmin	NA	

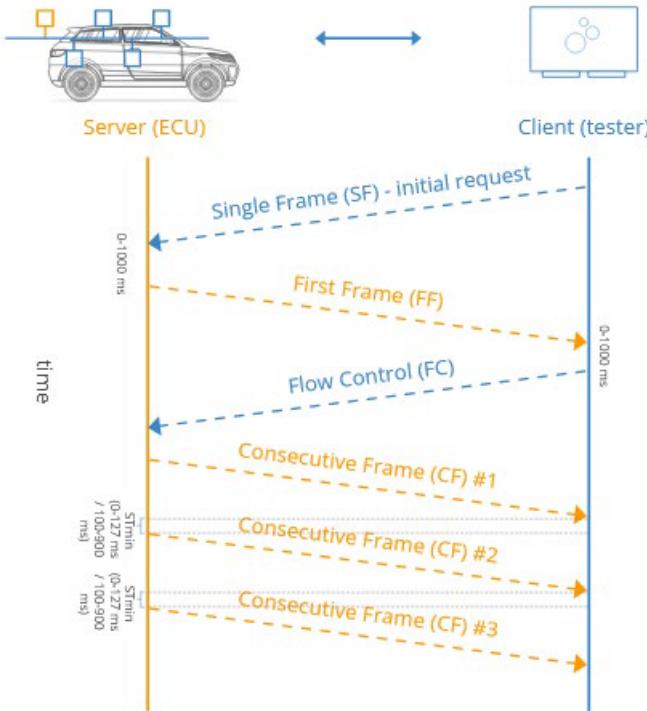
Hình 2.4: CAN - TP Header [2]

- Single Frame: kích thước data từ 0 đến 7 byte
- Example:



Hình 2.5: Example Single Frame

- Multi Frame: kích thước data từ 8 đến 4095 byte

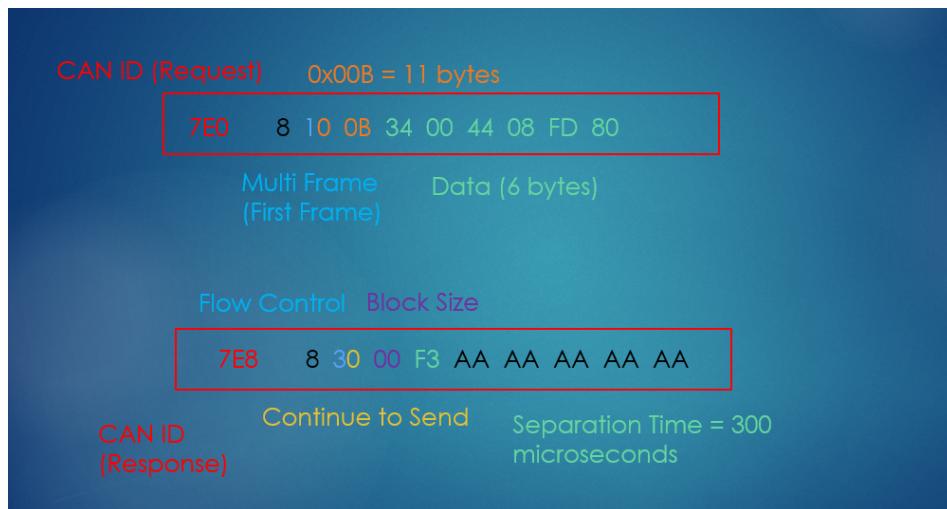
Hình 2.6: *Flow Multi Frame*

- Client gửi Single Frame đến Server (ECU)
- Trong trường hợp Server cần gửi data phản hồi từ 8 byte trở lên, Server sẽ gửi First Frame thông báo cho Client nhận Multi Frame.
- Sau khi nhận được thông báo từ Server (ECU), Client gửi phản hồi Flow Control đến Server (ECU) thông báo sẽ nhận Multi Frame.  
Flow Control Frame: Sau khi nhận được First Frame, Flow Control Frame sẽ được gửi bao gồm flow status, block size và separation time (STMIn).  
Trong trường hợp đơn giản, FC sẽ là 30 00 00 00 00 00 00 00 (tất cả các frame còn lại sẽ được gửi mà không có delay).

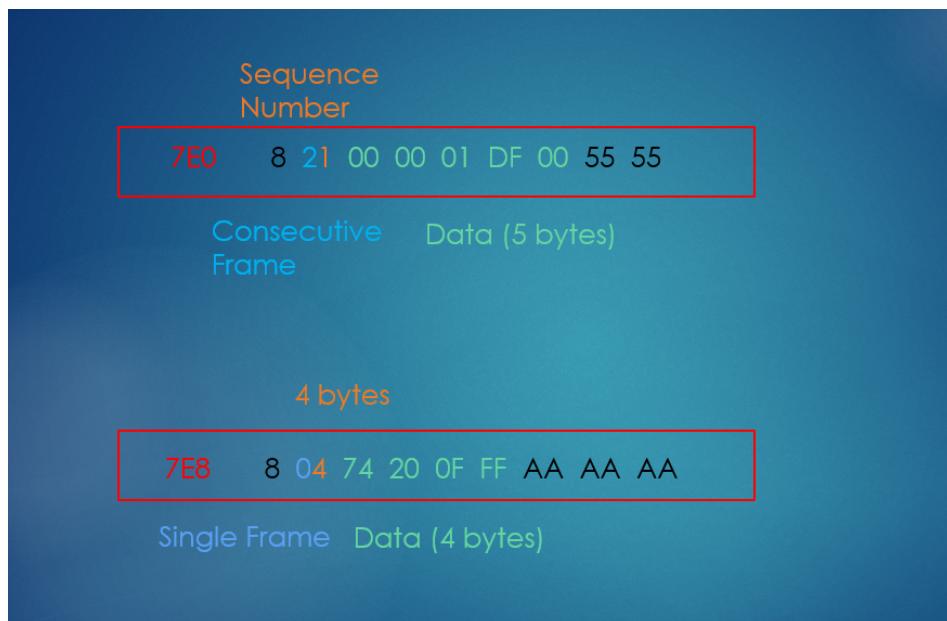
2

Flow Control				
Frame Type	Byte 0		Byte 1	Byte 2
Bit offset	Bit 7 to 4	Bit 3 to 0	Bit 15 to 8	Bit 23 to 16
Description	type	if the transfer is allowed flow status	Block Size	Separation Time (ST), minimum delay time between frames (end of one frame and the beginning of the other)
Single	type = 3	0 = Continue to send 1 = Wait for the next flow control frame 2 = Overflow/abort Over flow is detected abort transmission	0 = remaining "frames" to be sent without flow control or delay	<= 127, separation time in milliseconds.
Single	type = 3	0 = Continue to send 1 = Wait for the next flow control frame 2 = Overflow/abort Over flow is detected abort transmission	> 0 send number of "frames" before waiting for the next flow control frame	0xF1 to 0xF9 UF, 100 to 900 microseconds.

Hình 2.7: Flow Control Frame



Hình 2.8: Example Multi Frame



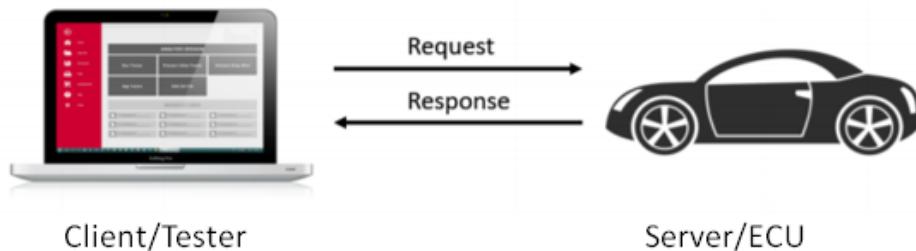
Hình 2.9: Example Multi Frame (cont)

## 2.3. UDS PROTOCOL

### 2.3.1. TỔNG QUAN

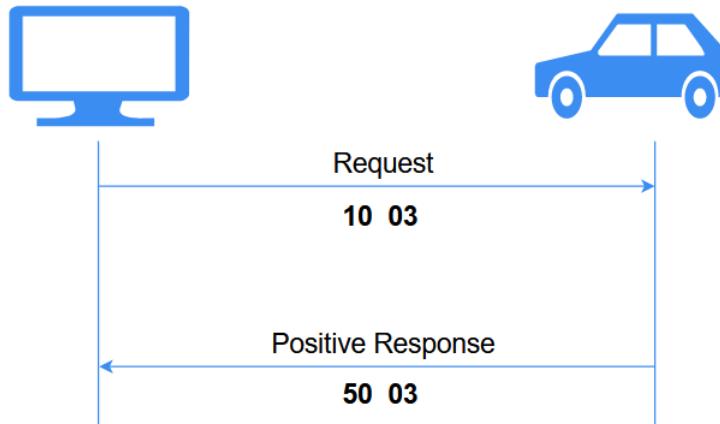
UDS là viết tắt của Unified Diagnostic Services, là một giao thức truyền thông chẩn đoán được sử dụng trong ngành công nghiệp ô tô để truyền thông với các đơn vị điều khiển điện tử (ECUs) trong xe. UDS xác định một tập hợp các dịch vụ chẩn đoán, cho phép chẩn đoán và sửa chữa các lỗi trong xe. Giao thức này được đặc tả trong tiêu chuẩn ISO 14229 và là một tiêu chuẩn quốc tế được sử dụng bởi các nhà sản xuất và nhà cung cấp ô tô. UDS cho phép chẩn đoán tiên tiến và chuẩn hóa hơn so với các giao thức trước đó, chẳng hạn như KWP2000. Nó hỗ trợ một loạt các chức năng chẩn đoán, bao gồm đọc và xóa mã lỗi chẩn đoán, đọc dữ liệu cảm biến và thực hiện các thử nghiệm bộ kích hoạt. UDS thường được sử dụng trên bus Controller Area Network (CAN), là bus truyền thông tiêu chuẩn được sử dụng trong các xe hiện đại.

UDS bao gồm Request Service and Response Service.



Hình 2.10: *UDS Request - Response*

- Service Request
  - Request without sub-function  
Request chỉ bao gồm Identifier và Data Parameter (optional)  
Example: Service \$2E + Data Parameter (Write Data By Id)
  - Request with sub-function  
Request sẽ bao gồm Service Identifier (required) + Sub-Function Byte (require) + Data Parameter (optional)  
Example: Service \$10 + 03 (Sub-function: Extended Session)]
- Service Response: Có 3 loại response phụ thuộc vào phản hồi từ Server
  - Positive Response:  
Server sẽ gửi Service requested + 0x40  
Ví dụ: Requesting service \$10, response = 0x10 + 0x40 = 0x50

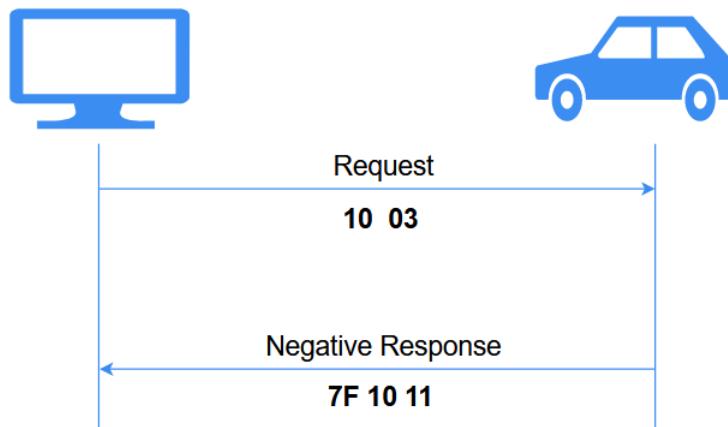


Hình 2.11: Positive Response

- Negative Response  
Server sẽ gửi 7F + Service requested + Response code  
Ví dụ: Requesting service \$10 - Negative Response 7F 10 11 (Service Not Supported)

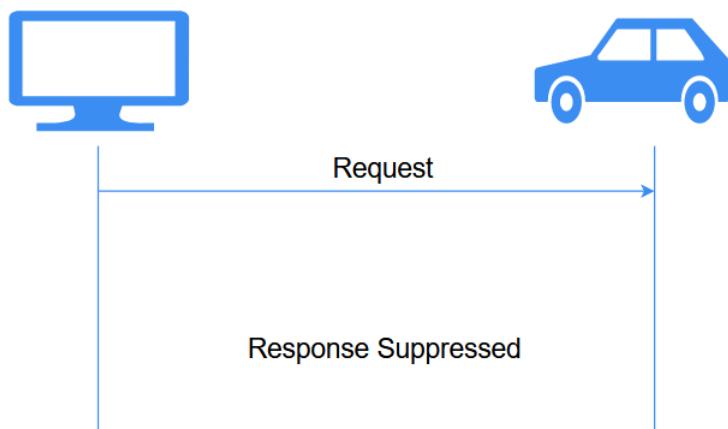
Note: Response code sẽ khác nhau tùy thuộc vào Service được gửi.

2



Hình 2.12: *Negative Response*

- Response Suppressed  
Client không yêu cầu phản hồi từ Server



Hình 2.13: *Response Suppressed*

### 2.3.2. UDS SERVICE CHO CẬP NHẬT PHẦN MỀM

#### 2.3.2.1 Các Services cơ bản cho việc cập nhật phần mềm

- Service \$10 – Diagnostic Session Control
- Service \$27 - Security Access
- Service \$31 – Routine Control, memory erase
- Transfer Data: bao gồm 3 service được thực thi nhằm gán data vào từng vùng nhớ của ECU ( ASW0, ASW1, ASW2, DS0, VDS)
  - Service \$34 - Request Download
  - Service \$36 - Transfer Data
  - Service \$37 - Request Transfer Exit
- Service \$31 - Verification Request
- Service \$11 - Reset ECU

2

#### 2.3.2.2 Service \$10 – Diagnostic Session Control

Service này cho phép phần mềm chẩn đoán trên xe hơi kiểm soát các phiên chẩn đoán trên bộ điều khiển của xe hơi.

Khi sử dụng service Diagnostic Session Control, phần mềm chẩn đoán trên xe hơi sẽ gửi một yêu cầu đến bộ điều khiển của xe hơi để chuyển đổi sang một phiên chẩn đoán khác. Có ba loại phiên chẩn đoán khác nhau:

- Default Session: Được sử dụng để truy cập các chức năng chẩn đoán cơ bản.
- Programming Session: Được sử dụng để thực hiện các chức năng lập trình và cấu hình trên bộ điều khiển của xe hơi.
- Extended Session: Được sử dụng để truy cập các chức năng chẩn đoán mở rộng và cấu hình nâng cao trên bộ điều khiển của xe hơi.

Sử dụng dịch vụ Diagnostic Session Control là cần thiết để đảm bảo rằng phần mềm chẩn đoán trên xe hơi có thể truy cập và kiểm soát các phiên chẩn đoán trên bộ điều khiển của xe hơi.

**Request Message Structure:**

Parameter Name	Byte Value
DiagnosticSessionControl Request SID	0x10
sub-function = [ diagnosticSessionType ]	0x00 – 0xFF

2

Bảng 2.1: Request Message Structure Service \$10

Sub-functions DiagnosticSessionType:

- 0x01 – Default Session
- 0x02 – Programming Session
- 0x03 – Extended Session

### Positive Response

Parameter Name	Byte Value
DiagnosticSessionControl Response SID	0x50 (0x10 + 0x40)
sub-function = [ diagnosticSessionType ]	0x00 – 0xFF
sessionParameterRecord[] = [ data#1 : data#4 ]	0x00 – 0xFF

Bảng 2.2: Positive Response Structure Service \$10

### Example

CAN ID	Length	Data
7E0	8	02 10 02 55 55 55 55 55
7E8	8	06 50 02 00 32 01 EA AA

2

Bảng 2.3: Example Positive Response Structure Service \$10

### Negative Response

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0x10
responseCode (Negative Response Code - NRC)	0xXX

Bảng 2.4: Negative Response Structure Service \$10

### Supported Negative Response Code (NRC)

NRC	Description
0x12	<b>sub-functionNotSupported</b> Client đã gửi mã Service hợp lệ, nhưng sub-function không hợp lệ hoặc không tồn tại trong service đó.
0x13	<b>incorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x22	<b>conditionsNotCorrect</b> Trước khi tiến hành bất kỳ dịch vụ nào từ Client, Server sẽ kiểm tra các điều kiện tiên quyết có được đáp ứng hay không. Nếu không thì Server sẽ gửi NRC này.

Bảng 2.5: Supported NRC Service \$10

**Example****2**

CAN ID	Length	Data
7E0	8	02 10 <b>05</b> 55 55 55 55 55
7E8	8	03 7F 10 <b>12</b> 55 55 55 55

Bảng 2.6: Example Negative Response Structure Service \$10

### 2.3.2.3 Service \$27 - Security Access

Service này được sử dụng để cấp quyền truy cập vào các hoạt động liên quan đến bảo mật, chẳng hạn như quyền truy cập đọc hoặc ghi vào các vùng bộ nhớ nhất định hoặc khả năng thực thi các dịch vụ chẩn đoán nhất định. Service này sẽ yêu cầu SEED và KEY để trao đổi giữa Server (ECU) và Client để cấp quyền truy cập. SEED sẽ được gửi bởi Server (ECU) đến Client và Client sẽ dùng SEED này để tính toán được KEY và gửi KEY này đến Server (ECU) để được cấp quyền truy cập. Nếu KEY chính xác, Server (ECU) sẽ cấp quyền truy cập cho Client.

#### Request Message Structure - Request Seed

Parameter Name	Byte Value
SecurityAccess Request SID	0x27
sub-function = [securityAccessType = requestSeed ]	0x01, 0x03, 0x05, 0x07 – 0x7D (odd number)
securityAccessDataRecord[] = [parameter#1 : parameter#m ]	0x00 – 0xFF

Bảng 2.7: Request Message Structure Service \$27 - Request Seed

#### Request Message Structure - Send Key

Parameter Name	Byte Value
SecurityAccess Request SID	0x27
sub-function = [securityAccessType = sendKey ]	0x02, 0x04, 0x06, 0x08 – 0x7E (even number)
securityKey[] = [ key#1 (high byte):key#m (low byte) ]	0x00 – 0xFF

Bảng 2.8: Request Message Structure Service \$27 - Send Key

## Positive Response

2

Parameter Name	Byte Value
SecurityAccess Response SID	0x67
sub-function = [ securityAccessType ]	0x00 – 0x7E
securitySeed[] = [ seed#1 (high byte):seed#m (low byte) ]	0x00 – 0xFF

Bảng 2.9: Positive Response Structure Service \$27

### Example

CAN ID	Length	Data
7E0 (1)	8	02 <b>27 01</b> 55 55 55 55 55
7E8 (2)	8	06 <b>67 01 74 FF 8E AC</b> 55
7E0 (3)	8	06 <b>27 02 80 FD FE 00</b> 55
7E8 (4)	8	02 <b>67 02</b> 55 55 55 55 55

Bảng 2.10: Example Positive Response Structure Service \$27

1. Client yêu cầu “SEED” từ ECU/Server
2. Server/ECU gửi “SEED” cho Client
3. Client gửi “KEY” (tính toán dựa theo "SEED" nhận được từ ECU/Server thông qua một giải thuật của ECU/Server)
4. Server/ECU phản hồi rằng "KEY" hợp lệ và mở khóa ECU.

**Negative Response**

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0xXX
responseCode (Negative Response Code - NRC)	0xXX

2

Bảng 2.11: Negative Response Structure Service \$27

## Supported Negative Response Code (NRC)

2

NRC	Description
0x12	<b>sub-functionNotSupported</b> Client đã gửi mã Service hợp lệ, nhưng sub-function không hợp lệ hoặc không tồn tại trong service đó.
0x13	<b>incorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x22	<b>conditionsNotCorrect</b> Trước khi tiến hành bất kỳ dịch vụ nào từ Client, Server sẽ kiểm tra các điều kiện tiên quyết có được đáp ứng hay không. Nếu không thì Server sẽ gửi NRC này.
0x24	<b>requestSequenceError</b> Nếu Client gửi message không theo trình tự, thì Server/ECU sẽ gửi NRC này cho Client. Trong Service này, sub-function "send key" được nhận trước "request seed" được gửi.
0x31	<b>requestOutOfRange</b> Nếu Server/ECU nhận được yêu cầu của Client có tham số nằm ngoài phạm vi, thì Server/ECU sẽ gửi NRC này.
0x35	<b>invalidKey</b> Nếu Server/ECU nhận được "KEY" từ Client không khớp với "KEY" do Server/ECU tạo, thì NRC này sẽ được gửi.
0x36	<b>exceededNumberOfAttempts</b> Về cơ bản, ở mỗi Server/ECU sẽ có một bộ đếm truy xuất bảo mật và nó sẽ có giới hạn như 3 hoặc 5 lần. Vì vậy, nếu Client gửi sai khóa bảo mật nhiều hơn giá trị bộ đếm đã xác định, thì Server/ECU sẽ gửi NRC này cho Client.
0x37	<b>requiredTimeDelayNotExpired</b> Sau khi Client gửi sai khóa bảo mật, Client phải đợi khoảng thời gian đã xác định, sau đó sẽ gửi lại khóa. Nhưng nếu gửi khóa bảo mật trước đó thì Server sẽ gửi NRC này.

Bảng 2.12: Supported NRC Service \$27

### 2.3.2.4 Service \$31 – Routine Control

Service Routine Control là một trong những dịch vụ (service) trong giao thức UDS (Unified Diagnostic Services) được sử dụng để điều khiển và thực hiện các chức năng của các phần mềm (firmware) được lưu trữ trong các bộ điều khiển (ECU) trong ô tô.

Dịch vụ này cho phép các chuyên gia chẩn đoán và kỹ thuật viên có thể truy cập và kiểm soát các chức năng của các phần mềm trong các bộ điều khiển của xe, bao gồm cả việc khởi động lại (restart), xóa bỏ (delete) hoặc chuyển đổi (switch) giữa các chế độ hoạt động của phần mềm.

Các chức năng này được thực hiện thông qua việc sử dụng các mã lệnh (routine) được định nghĩa trong phần mềm của các bộ điều khiển. Mã lệnh này được sử dụng để kiểm soát các hoạt động của các bộ điều khiển và thực hiện các chức năng như kiểm tra lỗi, chẩn đoán, hiệu chỉnh và cập nhật phần mềm.

Service Routine Control cũng có thể được sử dụng để thực hiện các chức năng như đọc và xóa các bản ghi lỗi (trouble code) được lưu trữ trong các bộ điều khiển và kiểm tra các thông số của các cảm biến và thiết bị khác trong hệ thống của xe.

Service Routine Control là một trong số các dịch vụ được định nghĩa trong tiêu chuẩn ISO 14229-1, cho phép các chuyên gia chẩn đoán và kỹ thuật viên có thể thực hiện các chức năng kiểm soát và quản lý các phần mềm trong các bộ điều khiển của ô tô.

#### Request Message Structure:

Parameter Name	Byte Value
RoutineControl Request SID	0x31
sub-function = [ routineControlType ]	0x00 – 0xFF
routineIdentifier [] = [ byte#1 (MSB) : byte#2 (LSB) ]	0x00 – 0xFF
routineControlOptionRecord[] = [ routineControlOption#1 : routineControlOption#m ]	0x00 – 0xFF

Bảng 2.13: Request Message Structure Service \$31

Sub-functions routineControlType:

- 0x01 – Start Routine.
- 0x02 – Stop Routine.
- 0x03 – Request Routine Result.

### Positive Response

Parameter Name	Byte Value
RoutineControl ResponseSID	0x71
routineControlType	0x00 – 0xFF
routineIdentifier [] = [ byte#1 (MSB) : byte#2 (LSB) ]	0x00 – 0xFF
routineInfo	0x00 – 0xFF
routineStatusRecord[] = [routineStatus#1 : routineStatus#m ]	0x00 – 0xFF

Bảng 2.14: Positive Response Structure Service \$31

Trong cập nhật phần mềm, service Routine Control áp dụng 2 sub-function:

- Erase Memory (0x31 + 0x01FF00)

### Example

CAN ID	Length	Data
7E0	8	04 <b>31 01 FF 00</b> 55 55 55
7E8	8	04 <b>71 01 FF 00</b> AA AA AA

Bảng 2.15: Example Erase Memory (0x31 + 0xFF00)

- Verification (0x31 + 0x01FF01)

### Example

2

CAN ID	Length	Data
7E0	8	04 <b>31 01 FF 01</b> 55 55 55
7E8	8	04 <b>71 01 FF 01</b> AA AA AA

Bảng 2.16: Example Verification (0x31 + 0x01FF01)

### Negative Response

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0x31
responseCode (Negative Response Code - NRC)	0xXX

Bảng 2.17: Negative Response Structure Service \$31

### Supported Negative Response Code (NRC)

2

NRC	Description
0x13	<b>incorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x14	<b>responseTooLong</b> Nếu client đã gửi message và độ dài của thông báo phản hồi lớn hơn độ dài tối đa của giao thức Truyền tải, thì Server/ECU sẽ gửi NRC này. CAN-TP có 4096 byte có độ dài tối đa, nếu Server/ECU gửi nhiều hơn thế, thì NRC này sẽ được Server gửi đến Client.
0x22	<b>conditionsNotCorrect</b> Trước khi tiến hành bất kỳ dịch vụ nào từ Client, Server sẽ kiểm tra các điều kiện tiên quyết có được đáp ứng hay không. Nếu không thì Server sẽ gửi NRC này.
0x31	<b>requestOutOfRange</b> Nếu Server/ECU nhận được yêu cầu của Client có tham số nằm ngoài phạm vi, thì Server/ECU sẽ gửi NRC này.
0x33	<b>securityAccessDenied</b> Nếu Server/ECU bị khóa và Client yêu cầu dịch vụ 0x31, thì Server/ECU sẽ gửi NRC này. Do đó, trước khi xử lý yêu cầu 0x31, Client phải mở khóa ECU.

Bảng 2.18: Supported NRC Service \$31

### 2.3.2.5 Service \$34 - Request Download

Đây là một trong những service quan trọng trong việc cập nhật phần mềm. Khi dữ liệu cần được truyền xuống ECU, service này sẽ gửi địa chỉ bắt đầu của ECU nhận dữ liệu và kích thước của dữ liệu cần được truyền. [9]

#### Request Message Structure:

Parameter Name	Byte Value
Request Download SID	0x34
Data Format Identifier	0x00 – 0xFF
Address & Length Format Identifier	0x00 – 0xFF
Memory Address Field = [ data#1 : data#m ]	0x00 – 0xFF
Memory Size Field = [ data#1 : data#n ]	0x00 – 0xFF

Bảng 2.19: Request Message Structure Service \$34

- **Data Format Identifier:** xác định định dạng của dữ liệu được yêu cầu tải , được sử dụng để chỉ định kích thước của dữ liệu và xác định liệu nó có được nén hay không. Ví dụ, giá trị 0x00 chỉ định rằng không có phương pháp nén hoặc phương pháp mã hóa nào được sử dụng.

Hoặc với giá trị 0x10 chỉ định rằng có phương thức nén được sử dụng.

- **Address & Length Format Identifier:**

- **Low Byte (Bit: 3-0):** Length (number of bytes) of the memory address parameter.
- **High Byte (Bit: 7-4):** Length (number of bytes) of the memory size parameter.

- **Memory Address Parameter:** là địa chỉ bắt đầu của bộ nhớ ECU mà dữ liệu sẽ được ghi vào.

- **Memory Size Parameter:** Tham số này sẽ được server (ECU) sử dụng để so sánh kích thước bộ nhớ không nén với tổng lượng dữ liệu được truyền trong dịch vụ TransferData (0x36).

### Positive Response

Parameter Name	Byte Value
RequestDownload Response SID	0x74
Length Format Identifier	0x20
MaxNumberOfBlockLength = [ data#1 : data#2 ]	0x00 - 0xFF

2

Bảng 2.20: Positive Response Structure Service \$34

- Length Format Identifier = 0x20.
  - Lower Nibble from Bit 3-0:** Mặc định là 0.
  - Higher Nibble from Bit 7-4:** Giá trị là 2. Vì vậy, 2 byte dữ liệu tiếp theo chỉ định số byte tối đa có thể được chuyển từ Client sang Server (ECU).
- MaxNumberOfBlockLength: Số byte tối đa mà mỗi yêu cầu TransferData (0x34) phải được đưa vào từ Client.

### Example

CAN ID	Length	Data
7E0	8	10 0B 34 00 44 08 FD 80
7E8	8	30 00 F3 AA AA AA AA AA
7E0	8	21 00 00 01 DF 00 55 55
7E8	8	04 74 20 0F FF AA AA AA

Bảng 2.21: Example Positive Response Structure Service \$34

- 0x00B: Chiều dài của data là 10 byte

**2**

- 0x34: Service Request Download
- 0x00: Data Format Identifier
- 0x44: 4-byte memory address - 4 byte memory size
  - 0x08FD8000 (4-byte): địa chỉ bắt đầu của Server () nhận dữ liệu.
  - 0x0001DF00 (4-byte): kích thước dữ liệu (byte) từ Client cần truyền vào Server (ECU)

### Negative Response

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0x34
responseCode (Negative Response Code - NRC)	0xXX

Bảng 2.22: Negative Response Structure Service \$34

### Supported Negative Response Code (NRC)

NRC	Description
0x13	<b>incorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x22	<b>conditionsNotCorrect</b> Điều này có thể xảy ra nếu kích thước dữ liệu không khớp giữa Server/ECU và Client trong quá trình tải xuống.
0x31	<b>requestOutOfRange</b> NRC sẽ được gửi nếu: 1. DID không hợp lệ, 2. addressAndLengthFormatIdentifier không hợp lệ, hoặc 3. memory Address/memory Size không hợp lệ.
0x33	<b>securityAccessDenied</b> Nếu Server/ECU bị khóa và Client yêu cầu dịch vụ 0x34, thì Server/ECU sẽ gửi NRC này. Do đó, trước khi xử lý yêu cầu 0x34, Client phải mở khóa ECU.
0x70	<b>uploadDownloadNotAccepted</b> Đôi khi do một số tình trạng lỗi trong bộ nhớ Server/ECU, nó sẽ không chấp nhận bất kỳ yêu cầu tải lên hoặc tải xuống nào từ Client.

2

Bảng 2.23: Supported NRC Service \$34

### 2.3.2.6 Service \$36 - Transfer Data

Sau khi service Request Download được gửi đi, xác định được địa chỉ bắt đầu và kích thước data cần truyền vào Server (ECU). Server Transfer Data sẽ có nhiệm vụ gửi chính xác data cần gửi vào Server (ECU).

#### Request Message Structure:

Parameter Name	Byte Value
TransferData Request SID	0x36
Sub-function	0x00 – 0xFF
Data	0x00 – 0xFF

Bảng 2.24: Request Message Structure Service \$36

Do độ dài tối đa của khung được gửi là 0xFFFF (4095 byte) và dữ liệu cần gửi lớn hơn 4095 byte, do đó, chúng ta cần phải gửi service 0x36 nhiều lần cho đến khi hết dữ liệu. Bắt đầu của sub-function là 0x01, tiếp theo là 0x02, cho đến 0xFF rồi quay lại 0x00, 0x01,...

#### Positive Response

Parameter Name	Byte Value
TransferData Response SID	0x76
Sub-function	0x00 - 0xFF

Bảng 2.25: Positive Response Structure Service \$36

### Example

2

CAN ID	Length	Data
7E0	8	12 02 36 01 DE AD BE EF
7E8	8	30 00 F3 AA AA AA AA AA
7E0	8	21 00 00 00 00 00 00 00
7E0	8	22 00 00 00 00 00 00 00
7E0	8	...
7E0	8	2F 00 00 00 00 00 00 00
7E0	8	20 00 00 00 00 00 00 00
7E0	8	...
7E8	8	02 76 01 AA AA AA AA AA

Bảng 2.26: Example Positive Response Structure Service \$36

### Negative Response

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0x36
responseCode (Negative Response Code - NRC)	0xXX

Bảng 2.27: Negative Response Structure Service \$34

### Supported Negative Response Code (NRC)

2

NRC	Description
0x13	<b>incorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x24	<b>requestSequenceError</b> NRC này sẽ được gửi nếu: 1. Chưa kích hoạt Service Request Download (0x34) mà đã gọi Service này, hoặc 2. Đã nhận được đủ data được xác định từ Service Request Download (0x34) nhưng Service này vẫn gửi thêm data.
0x31	<b>requestOutOfRange</b> Nếu Server/ECU nhận được yêu cầu của Client có tham số nằm ngoài phạm vi, thì Server/ECU sẽ gửi NRC này.
0x71	<b>transferDataSuspended</b> NRC này sẽ được gửi nếu: 1. Hoạt động truyền dữ liệu đã bị tạm dừng do lỗi. 2. Độ dài của data tải xuống không đáp ứng các yêu cầu của tham số Kích thước bộ nhớ được gửi trong message của Service Request Download.
0x72	<b>generalProgrammingFailure</b> Nếu Server/ECU phát hiện lỗi trong quá trình xóa hoặc lập trình tại bộ nhớ vĩnh viễn (ví dụ: NVM/Flash/EEPROM), thì sẽ gửi NRC này cho Client.
0x73	<b>wrongBlockSequenceCounter</b> Về cơ bản, bất cứ khi nào một gói dữ liệu liên tiếp nhiều khung được gửi bởi Client đến Server. Giữa hai khung liên tiếp nhau, chỉ số Khung sẽ tăng lên trong mỗi khung tiếp theo. Server cũng nhận được giá trị đó và so sánh giá trị đó với giá trị trước đó phải là +1 trong mỗi khung hiện tại nhận được. Nếu có bất kỳ sự trùng khớp nào, Server sẽ gửi NRC này cho Client.

Bảng 2.28: Supported NRC Service \$34

### 2.3.2.7 Service \$37 – Request Transfer Exit

Khi quá trình truyền dữ liệu hoàn tất, Client sẽ gửi Service Request Transfer Exit, yêu cầu thoát truyền dữ liệu. Positive response sẽ được trả về nếu quá trình truyền dữ liệu hoàn tất và tất cả dữ liệu đã được nhận thành công.

2

#### Request Message Structure:

Parameter Name	Byte Value
RequestTransferExit Request SID	0x37

Bảng 2.29: Request Message Structure Service \$37

#### Positive Response

Parameter Name	Byte Value
RequestTransferExit Response SID	0x77

Bảng 2.30: Positive Response Structure Service \$37

#### Example

CAN ID	Length	Data
7E0	8	01 37 55 55 55 55 55 55
7E8	8	01 77 55 55 55 55 55 55

Bảng 2.31: Example Positive Response Structure Service \$37

#### Negative Response

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0x37
responseCode (Negative Response Code - NRC)	0xXX

Bảng 2.32: Negative Response Structure Service \$37

### Supported Negative Response Code (NRC)

NRC	Description
0x13	<b>incorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x24	<b>requestSequenceError</b> NRC này sẽ được gửi nếu: 1. Service Transfer Data (0x36) chưa hoàn thành mà đã gọi Service này, hoặc 2. Service Request Download chưa được kích hoạt.

Bảng 2.33: Supported NRC Service \$37

#### 2.3.2.8 Service \$11 – ECU Reset

Chức năng của ECU Reset Service (0x11) là reset ECU ở định dạng khác theo yêu cầu hoặc khi có sự cố xảy ra. Mục tiêu của việc reset ECU là khôi phục ECU bị trục trặc từ trạng thái không hoạt động, chẳng hạn như trạng thái treo hoặc bất kỳ trạng thái không hoạt động nào khác, nhưng nó vẫn phải có khả năng tương tác với máy tính bên ngoài. [5]

Về cơ bản, nếu có bất kỳ sự cố nào xảy ra trên xe và chúng ta không biết nguyên nhân chính xác thì chúng ta phải thử phương pháp này. Đây là một dịch vụ rất quan trọng trong giao thức UDS. Sau khi thiết lập lại thành công, ECU sẽ trở lại trạng thái mặc định.

Có nhiều loại Reset ECU, nhưng chúng ta sử dụng Hard Reset (sub-function = 0x01) là chính, nó tương đương với khởi động lại nguồn.

### Request Message Structure:

2

Parameter Name	Byte Value
ECUReset Request SID	0x11
sub-function	0x00 – 0xFF

Bảng 2.34: Request Message Structure Service \$11

### Positive Response

Parameter Name	Byte Value
ECUReset Response SID	0x51
sub-function	0x00 – 0xFF

Bảng 2.35: Positive Response Structure Service \$11

### Example

CAN ID	Length	Data
7E0	8	02 11 01 55 55 55 55 55
7E8	8	02 51 01 AA AA AA AA AA

Bảng 2.36: Example Positive Response Structure Service \$11

## Negative Response

2

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0x11
responseCode (Negative Response Code - NRC)	0xXX

Bảng 2.37: Negative Response Structure Service \$11

### Supported Negative Response Code (NRC)

NRC	Description
0x12	<b>sub-functionNotSupported</b> Client đã gửi mã Service hợp lệ, nhưng sub-function không hợp lệ hoặc không tồn tại trong service đó.
0x13	<b>incorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x22	<b>conditionsNotCorrect</b> Trước khi tiến hành bất kỳ dịch vụ nào từ Client, Server sẽ kiểm tra các điều kiện tiên quyết có được đáp ứng hay không. Nếu không thì Server sẽ gửi NRC này

Bảng 2.38: Supported NRC Service \$11

### Example

CAN ID	Length	Data
7E0	8	02 10 <b>05</b> 55 55 55 55 55
7E8	8	03 7F 10 <b>12</b> 55 55 55 55

Bảng 2.39: Example Negative Response Structure Service \$10

### 2.3.2.9 Vehicle Identification Number

Số VIN (Vehicle Identification Number) là một chuỗi gồm 17 ký tự duy nhất được gắn trên mỗi chiếc xe ô tô để xác định và phân biệt các loại xe. Số VIN cung cấp thông tin về nhà sản xuất, model, năm sản xuất, động cơ, hộp số, phân loại và một số thông tin khác về chiếc xe.

Các ký tự của số VIN được phân bố thành các nhóm như sau:

- Ký tự thứ 1-3: WMI (World Manufacturer Identifier) - xác định nhà sản xuất của xe. Đây là nhóm các ký tự đầu tiên của số VIN.
- Ký tự thứ 4-8: VDS (Vehicle Descriptor Section) - xác định thông tin về model, kiểu dáng, động cơ, hộp số và phân loại của xe. Đây là nhóm ký tự thứ hai của số VIN.
- Ký tự thứ 9: check digit (chữ số kiểm tra) - kiểm tra tính hợp lệ của số VIN. Chữ số kiểm tra được tính bằng cách sử dụng một thuật toán đơn giản dựa trên các ký tự khác trong số VIN.
- Ký tự thứ 10: thể hiện năm sản xuất của xe. Đây là nhóm ký tự thứ ba của số VIN.
- Ký tự thứ 11: thể hiện nhà sản xuất đã chọn để sử dụng cho xác nhận của họ. Nhà sản xuất có thể sử dụng ký tự này để phân biệt các loại xe được sản xuất trong cùng một năm và cùng một nhà sản xuất.
- Ký tự thứ 12-17: VIS (Vehicle Identifier Section) - xác định số thứ tự sản xuất của xe. Đây là nhóm ký tự cuối cùng của số VIN.

Số VIN cung cấp cho các cơ quan chức năng và người tiêu dùng các thông tin quan trọng về chiếc xe, bao gồm lịch sử sửa chữa, đăng ký xe, bảo hiểm, và tình trạng pháp lý. Nó cũng giúp tăng tính chính xác trong việc định giá và đánh giá xe cũ, và là một yếu tố quan trọng trong việc tìm kiếm thông tin về các chiếc xe đã qua sử dụng.

**Ví dụ:** Số VIN **1G1JC5248Y7226403** là số VIN của một chiếc xe Chevrolet Cavalier sedan được sản xuất vào năm 2000 tại Mỹ. Chi tiết về cấu trúc số VIN này như sau:

- 1: Mỹ
- G: General Motors
- 1J: Chevrolet

- C: Cavalier
- 5: Sedan 4 cửa
- 2: Động cơ xăng 4 xi-lanh 2.2L L4 OHV
- 4: Hệ thống an toàn bao gồm túi khí và dây đai an toàn
- 8: Mã kiểm soát của nhà sản xuất
- Y: Năm sản xuất 2000
- 7: Nhà máy sản xuất ở Lordstown, Ohio
- 226403: Số thứ tự sản xuất

### 2.3.2.10 UDS Service for READ VIN (0x22) and WRITE VIN (0x2E)

Service \$22 - Read Data by Identifier Service này cho phép truy vấn các giá trị dữ liệu trong ECU bằng cách sử dụng các mã định danh (Identifier) đã được định nghĩa trước đó.

Các thông số truy vấn có thể bao gồm các thông số như nhiệt độ động cơ, tốc độ xe, áp suất lốp, thông số của các cảm biến khác, v.v. Các giá trị này có thể được sử dụng để chẩn đoán các lỗi trong hệ thống, cũng như để giám sát hiệu suất của các thành phần trong xe.

#### Request Message Structure:

Parameter Name	Byte Value
ReadDataByIdentifier Request SID	0x22
dataIdentifier[]#1 = [ byte#1 (MSB) : byte#2 ]	0x00 – 0xFF

Bảng 2.40: Request Message Structure Service \$22

### Positive Response

2

Parameter Name	Byte Value
ReadDataByIdentifier Response SID	0x62
dataIdentifier[]#1 = [ byte#1 (MSB) : byte#2 ]	0x00 – 0xFF

Bảng 2.41: Positive Response Structure Service \$22

### Example Read VIN

CAN ID	Length	Data
7E0	8	03 22 F1 90 AA AA AA AA
7E8	8	10 14 62 F1 90 31 47 31
7E0	8	03 30 00 00 AA AA AA AA
7E8	8	21 4A 43 35 32 34 38 59
7E8	8	22 37 32 32 36 34 30 33

Bảng 2.42: Service \$22 - Read VIN from ECU

### Negative Response

2

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0x2E
responseCode (Negative Response Code - NRC)	0xXX

Bảng 2.43: Negative Response Structure Service \$22

### Supported Negative Response Code (NRC)

NRC	Description
0x13	<b>IncorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x14	<b>responseTooLong</b> Nếu client đã gửi message và độ dài của thông báo phản hồi lớn hơn độ dài tối đa của giao thức Truyền tải, thì Server/ECU sẽ gửi NRC này. CAN-TP có 4096 byte có độ dài tối đa, nếu Server/ECU gửi nhiều hơn thế, thì NRC này sẽ được Server gửi đến Client.
0x22	<b>conditionsNotCorrect</b> Trước khi tiến hành bất kỳ dịch vụ nào từ Client, Server sẽ kiểm tra các điều kiện tiên quyết có được đáp ứng hay không. Nếu không thì Server sẽ gửi NRC này.
0x31	<b>requestOutOfRange</b> Nếu Server/ECU nhận được yêu cầu của Client có tham số nằm ngoài phạm vi, thì Server/ECU sẽ gửi NRC này.
0x33	<b>securityAccessDenied</b> Nếu Server/ECU bị khóa và Client yêu cầu dịch vụ 0x22, thì Server/ECU sẽ gửi NRC này. Do đó, trước khi xử lý yêu cầu 0x22, Client phải mở khóa ECU.

Bảng 2.44: Supported NRC Service \$22

Service \$2E - Write Data by Identifier Client có thể ghi thông tin vào một vị trí nội bộ cụ thể trên Server (ECU) bằng cách cung cấp data identifier.

### Request Message Structure:

Parameter Name	Byte Value
WriteDataByIdentifier Request SID	0x2E
(DID)dataIdentifier[]#1 = [ byte#1 (MSB) : byte#2 ]	0x00 – 0xFF
dataRecord[] = [ data#1 : data#m ]	0x00 – 0xFF

2

Bảng 2.45: Request Message Structure Service \$2E

### Positive Response

Parameter Name	Byte Value
WriteDataByIdentifier Response SID	0x6E
dataIdentifier[]#1 = [ byte#1 (MSB) : byte#2 ]	0x00 – 0xFF

Bảng 2.46: Positive Response Structure Service \$2E

Với WRITE VIN, cần 3 service để ghi VIN vào ECU

1. Service \$10 - Diagnostic Session Control with sub-function 0x03 (Extended Session)
2. Service \$27 - Security Access
3. Service \$2E - Write Data By Identifier with DID = 0xF190

17 ký tự trong số VIN sẽ lần lượt được chuyển đổi thành mã ASCII và sau đó được chuyển đổi thành hệ thập lục phân trước khi được gửi qua CAN. Mỗi ký tự sẽ tương ứng với một byte.

#### Example Write VIN "1G1JC5248Y7226403"to ECU

1G1JC5248Y7226403 to ASCII and hex

- 1 => 49 => 0x31

2

- G => 71 => 0x47
- 1 => 49 => 0x31
- J => 74 => 0x4A
- C => 67 => 0x43
- 5 => 53 => 0x35
- 2 => 50 => 0x32
- 4 => 52 => 0x34
- 8 => 56 => 0x38
- Y => 89 => 0x59
- 7 => 55 => 0x37
- 2 => 50 => 0x32
- 2 => 50 => 0x32
- 6 => 54 => 0x36
- 4 => 52 => 0x34
- 0 => 48 => 0x30
- 3 => 51 => 0x33

CAN ID	Length	Data
7E0	8	10 14 2E F1 90 31 47 31
7E8	8	03 30 00 00 AA AA AA AA
7E0	8	21 4A 43 35 32 34 38 59
7E0	8	22 37 32 32 36 34 30 33
7E8	8	03 6E F1 90 AA AA AA AA

Bảng 2.47: Service \$2E - Write VIN to ECU

### Negative Response

Parameter Name	Byte Value
Negative Response SID	0x7F
Request SID	0x2E
responseCode (Negative Response Code - NRC)	0xXX

2

Bảng 2.48: Negative Response Structure Service \$2E

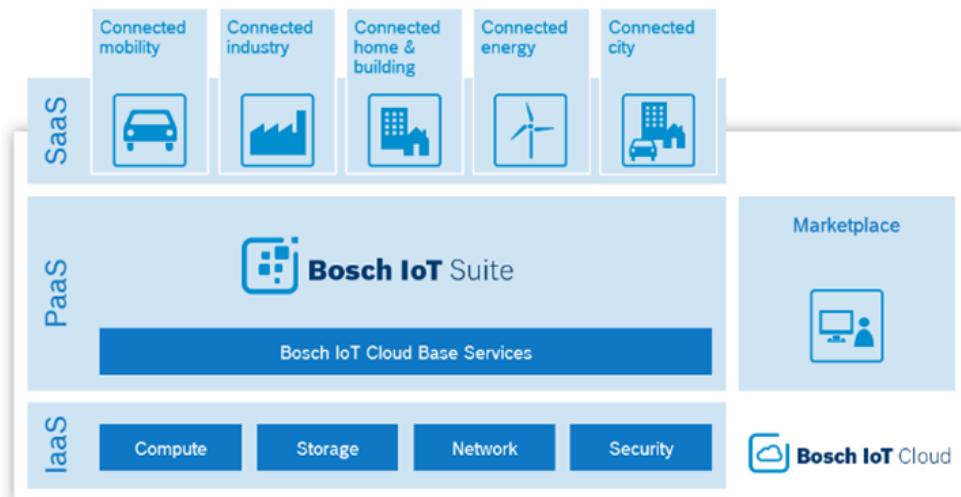
### Supported Negative Response Code (NRC)

NRC	Description
0x13	<b>incorrectMessageLengthOrInvalidFormat</b> Chiều dài của message không hợp lệ.
0x22	<b>conditionsNotCorrect</b> Trước khi tiến hành bất kỳ dịch vụ nào từ Client, Server sẽ kiểm tra các điều kiện tiên quyết có được đáp ứng hay không. Nếu không thì Server sẽ gửi NRC này.
0x31	<b>requestOutOfRange</b> Nếu Server/ECU nhận được yêu cầu của Client có tham số nằm ngoài phạm vi, thì Server/ECU sẽ gửi NRC này.
0x33	<b>serviceNotSupported</b> Nếu Server/ECU bị khóa và Client yêu cầu dịch vụ 0x31, thì Server/ECU sẽ gửi NRC này. Do đó, trước khi xử lý yêu cầu 0x31, Client phải mở khóa ECU.
0x72	<b>generalProgrammingFailure</b> Nếu Server/ECU phát hiện lỗi trong quá trình xóa hoặc lập trình tại bộ nhớ vĩnh viễn (ví dụ: NVM/Flash/EEPROM), thì sẽ gửi NRC này cho Client.

Bảng 2.49: Supported NRC Service \$2E

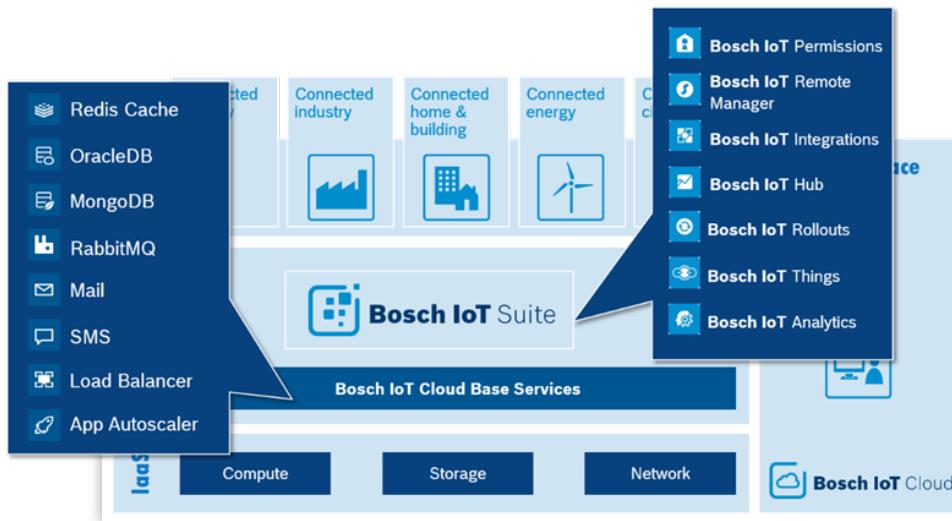
## 2.4. BOSCH IoT ROLLOUTS

Bosch IoT Cloud là một nền tảng điện toán đám mây được thiết kế dành riêng cho IoT. Nó cho phép các nhà phát triển sử dụng tài nguyên được chia sẻ như tài nguyên máy tính và mạng cũng như các dịch vụ trong khi không phải duy trì cơ sở hạ tầng cơ bản. Cũng tương tự như AWS của Amazon hay Azure của Microsoft thì Bosch IoT Cloud là sản phẩm của Bosch.



Hình 2.14: *Bosch IoT Cloud*

Cụ thể hơn, chúng em sử dụng Bosch IoT Rollouts, nó là một dịch vụ trong Bosch IoT Suite thuộc Bosch IoT Cloud.



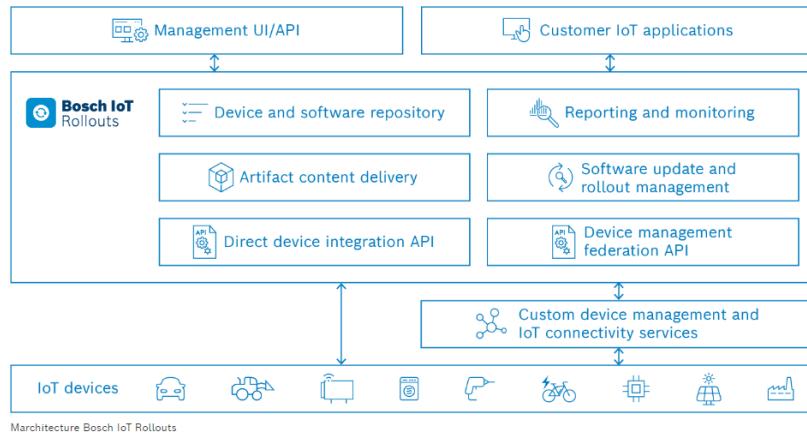
Hình 2.15: *Bosch IoT Suite*

#### 2.4.1. TỔNG QUAN

Bosch IoT Rollouts là một giải pháp phụ trợ độc lập với miền, hỗ trợ triển khai các bản cập nhật phần mềm hoặc phân phối phần mềm cho các thiết bị cạnh. Các thiết bị có thể được kết nối với máy chủ Bosch IoT Rollouts:

- Trực tiếp thông qua một giao diện được tối ưu hóa.
- Gián tiếp thông qua các máy chủ quản lý thiết bị liên kết.

2

Hình 2.16: *Bosch IoT Rollouts*

## 2.4.2. CHỨC NĂNG

### Cập nhật phần mềm cho các thiết bị trong Internet of Things

Bosch IoT Rollouts tập trung vào việc điều phối và thực hiện các bản cập nhật phần mềm cho các thiết bị IoT, có thể được sử dụng trong nhiều lĩnh vực và trường hợp khác nhau. Sử dụng Bosch IoT Rollouts độc lập thông qua giao diện người dùng hay tích hợp nó với API quản lý thì đều có thể tận dụng tối đa tính năng của nó. Điều này cho phép:

- Quản lý thiết bị trong kho thiết bị.
- Quản lý các phiên bản phần mềm trong kho phần mềm.
- Lọc thiết bị bằng ngôn ngữ truy vấn liên quan đến Bosch IoT Rollouts.
- Thực hiện các bản cập nhật phần mềm đơn lẻ hoặc quản lý các chiến dịch cập nhật phần mềm quy mô lớn.
- Sử dụng các tính năng bảo mật hiện đại.
- Theo dõi tiến độ của các quá trình cập nhật phần mềm.

### Cập nhật phần mềm thông qua các giải pháp quản lý thiết bị

Trong trường hợp cần có thêm khả năng quản lý thiết bị, Bosch IoT Rollouts cung cấp API Liên kết quản lý thiết bị. API này cho phép tích hợp với các giải pháp quản lý thiết bị.

API quản lý, trừu tượng hóa tất cả các quy trình liên quan đến cập nhật phần mềm, hoàn toàn tương thích với API của dự án mã nguồn mở Eclipse hawkBit™. [8]

2

### 2.4.3. CÁC TÍNH NĂNG CƠ BẢN [8]

#### 2.4.3.1 Kho thiết bị

- **Lưu trạng thái thiết bị:** Kho lưu trữ thiết bị nói về việc quản lý các thiết bị được kết nối (mục tiêu). Có thể truy xuất các chi tiết và thuộc tính của thiết bị cũng như duy trì các mô tả, siêu dữ liệu cũng như các thẻ. Ngoài ra có thể lấy thông tin về các phiên bản phần mềm được chỉ định và cài đặt hoặc trạng thái liên quan đến cập nhật phần mềm.

- **Gắn thẻ:** Triển khai cho phép tạo và duy trì các thẻ tùy ý. Các thẻ này có thể được gán cho các thiết bị, có thể được sử dụng như một cơ chế lọc đơn giản.

- **Lọc và Nhóm:** Có thể lọc phức tạp hơn bằng cách sử dụng cái gọi là Bộ lọc Mục tiêu. Bộ lọc mục tiêu giúp quản lý thiết bị bằng cách nhóm chúng theo truy vấn. Ngoài ra, các bộ lọc này được sử dụng để tự động gán các bản cập nhật, tức là một thiết bị mới đăng ký phù hợp với truy vấn sẽ được tự động cập nhật bằng bản cập nhật phần mềm đã chọn.

#### 2.4.3.2 Kho phần mềm

- **Bộ phân phối và module phần mềm:** Kho lưu trữ phần mềm là về cấu trúc và quản lý các bản cập nhật phần mềm. Bản cập nhật phần mềm (Bộ phân phối) bao gồm các module phần mềm, bản thân các module này chứa nhiều tệp. Mọi module phần mềm đều có một loại riêng biệt (ví dụ: Ứng dụng). Bộ phân phối cũng thuộc một loại nhất định, chỉ định loại module phần mềm nào phải hoặc có thể được bao gồm. Tất nhiên, có thể gán siêu dữ liệu cho cả hai thực thể.

- **Xác thực:** Khi xác định Bộ phân phối mới, Triển khai IoT của Bosch đảm bảo rằng các loại module phần mềm phù hợp với loại Bộ phân phối đã xác định.

- **Gắn thẻ và lọc:** Các bản cập nhật phần mềm (Bộ phân phối) có thể được gắn thẻ bằng các thẻ do người dùng xác định. Điều này một lần nữa cho phép lọc.

### 2.4.3.3 Cập nhật phần mềm và triển khai quản lý

- **Cập nhật phần mềm đơn lẻ:** Một bản cập nhật phần mềm có thể được chỉ định cho một thiết bị. Sau đó, Bosch IoT Rollouts sẽ giám sát trạng thái của quá trình cập nhật. Trạng thái của thiết bị có thể là không xác định, đã đăng ký, đang đồng bộ hóa, đang chờ xử lý hoặc có lỗi.

- **Quản lý chiến dịch:** Quản lý chiến dịch (Rollouts Management) cho phép quản lý hoạt động cập nhật phần mềm cho các kịch bản IoT quy mô lớn với hàng trăm nghìn thiết bị. Triển khai IoT của Bosch cho phép:

- Tạo và cập nhật chiến dịch:
  - Lựa chọn mục tiêu làm đầu vào cho chiến dịch dựa trên chức năng bộ lọc mục tiêu.
  - Lựa chọn một tập hợp phân phối.
  - Tự động phân tách danh sách mục tiêu đầu vào thành các nhóm triển khai được đánh số xác định (định nghĩa nhóm tự động).
  - Định nghĩa tùy chọn của các nhóm triển khai tùy chỉnh với các điều kiện riêng lẻ (định nghĩa nhóm bán tự động).
- Bắt đầu xếp tầng của các nhóm triển khai dựa trên trạng thái cài đặt của nhóm trước đó.
- Tắt khẩn cấp chiến dịch trong trường hợp một nhóm vượt quá ngưỡng lỗi xác định.
- Giám sát tiến độ chiến dịch.

- **Cấp phát thiết bị ban đầu:** Như được mô tả trong phần Kho lưu trữ thiết bị ở trên, có thể sử dụng các bộ lọc để cấp phát tự động. Bosch IoT Rollouts xác định liên tục tất cả các thiết bị phù hợp với tiêu chí bộ lọc liên quan và tự động chỉ định bản cập nhật phần mềm tương ứng.

- **Quy trình phê duyệt:** Đối với các kịch bản IoT quy mô lớn, Triển khai IoT của Bosch hỗ trợ các nguyên tắc kiểm soát kép khi bắt đầu chiến dịch. Điều thứ hai có nghĩa là một người tạo chiến dịch và yêu cầu sự chấp thuận từ người có trách nhiệm. Sau khi phê duyệt chiến dịch, có thể kích hoạt bắt đầu chiến dịch.

### 2.4.3.4 Báo cáo và giám sát

- **Lịch sử hành động:** Triển khai IoT của Bosch cung cấp phương tiện để theo dõi trạng thái của mọi thiết bị thông qua lịch sử hành động của nó. Lịch sử hành động

này minh bạch hành động nào xảy ra vào thời điểm cụ thể và kết quả của nó.

- **Thống kê:** Ngoài lịch sử hành động, còn có các chế độ xem đồ họa tổng hợp về:

- Tình trạng thiết bị.
- Hoạt động thiết bị.
- Các thiết bị được kết nối theo thời gian.
- Các hành động được tạo theo thời gian.
- Phiên bản cập nhật phần mềm đã cài đặt.

2

#### 2.4.3.5 Quản lý người dùng

Bosch IoT Rollouts cung cấp khả năng quản lý người dùng chi tiết, cho phép gán các quyền cụ thể cho người dùng.

#### 2.4.3.6 Giao diện người dùng

- Bosch IoT Rollouts cung cấp một giao diện người dùng, cho phép kiểm soát tất cả các tính năng sẵn có một cách thuận tiện. Giao diện người dùng được cấu trúc theo các dạng xem sau:

- Deployment: Tổng quan trạng thái, quản lý và triển khai thử công.
- Rollout: Triển khai phần mềm trên quy mô lớn, tổng quan về trạng thái triển khai và quản lý triển khai.

2

The screenshot shows the Deployment Management interface. On the left, there's a sidebar with navigation links: Deployment, Rollout, Target Filters, Distributions, Upload, Statistics, System Config, Documentation, User Management, and Support. The main area has tabs for Targets and Distributions. Under Targets, there's a table with columns: Name, Version, Status, and Actions. The table lists various targets like DevKit007112, DevKit00763, etc. Under Distributions, there's another table with columns: Name, Version, Status, and Actions. This table lists packages such as Package 1.0.8, Package 1.0.7, etc. To the right, there's a large table titled 'Action History' with columns: Active, Distribution set, Date and time, Status, For..., and Action. The table contains many entries for different packages and their deployment actions.

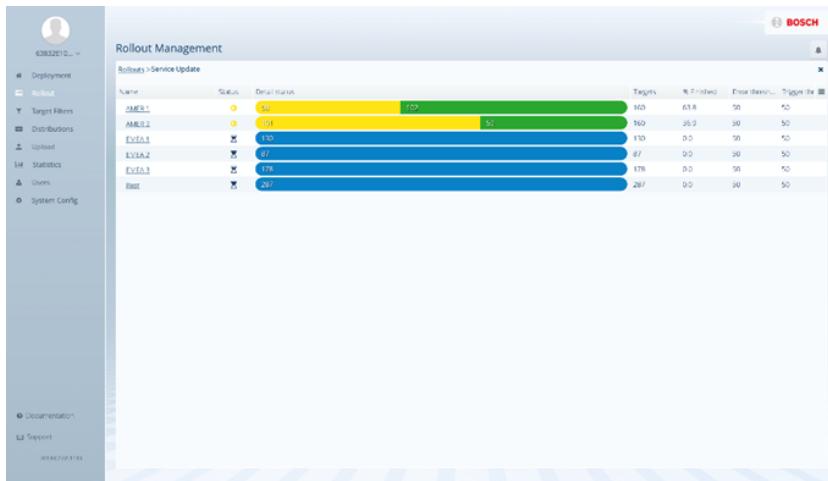
Hình 2.17: Deployment

### – Rollout List:

The screenshot shows the Rollout Management interface. On the left, there's a sidebar with navigation links: Deployment, Rollout, Target Filters, Distributions, Upload, Statistics, Users, System Config, Documentation, Support, and a user profile. The main area has a table titled 'Rollouts' with columns: Name, Distribution set, Status, Detail status, Groups, Targets, and Actions. The table lists several rollouts: Global Device Initialization, Europe OS Update 1.1, Asia OS Update 1.1, Global TS Update 1.2, and Service Locator. Each rollout entry includes a progress bar indicating its status and a set of icons for managing it.

Hình 2.18: Rollout List

- Rollout Group:

Hình 2.19: *Rollout Group*

- Rollout Creation:

**Create new Rollout**

**Distribution set \***: Package 1.0.10

**Custom Target Filter \***: All the cars

**Description**: Rollout the latest software to all cars worldwide

**Action type \***: Forced

**Start type \***: Manual

**Number of Groups**: Advanced Group definition

**Groups** (Total Targets: 5006):

- Group 1: 10 (5000) [Status: OK]
- Group 2: 10 (5000) [Status: OK]
- Group 3: 10 (5000) [Status: OK]
- Group 4: 10 (5000) [Status: OK]
- Group 5: 10 (5000) [Status: OK]
- Group 6: 10 (5000) [Status: OK]
- Group 7: 10 (5000) [Status: OK]
- Group 8: 10 (5000) [Status: OK]
- Group 9: 4 (5000) [Status: OK]
- Group 10: 4 (5000) [Status: OK]
- Group 11: 4 (5000) [Status: OK]
- Group 12: 3 (2000) [Status: OK]
- Group 13: 3 (500) [Status: OK]
- Group 14: 3 (500) [Status: OK]
- Group 15: 4 (500) [Status: OK]

Hình 2.20: *Rollout Creation*

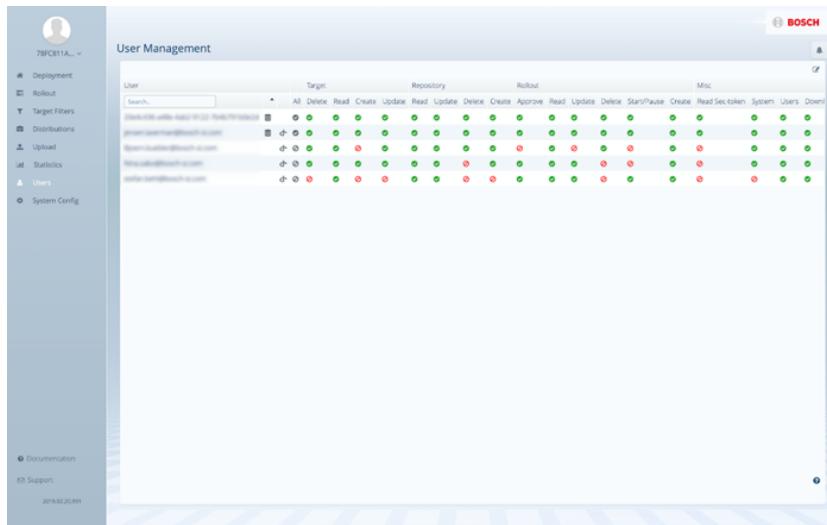
- Target Filters: Tổng quan về bộ lọc mục tiêu tùy chỉnh và quản lý bộ lọc.

Hình 2.21: Target Filters

- Upload:

Hình 2.22: Upload

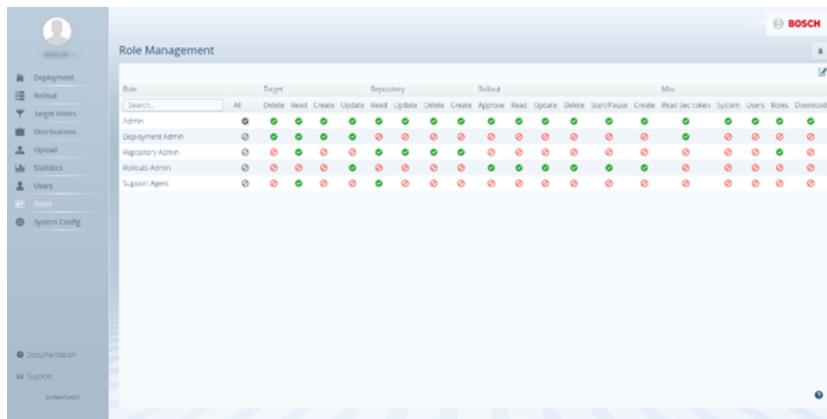
- User Management: Quản trị người dùng cho các Tài khoản Bosch đang sử dụng Bosch IoT Rollouts như một dịch vụ độc lập.



2

Hình 2.23: *User Management*

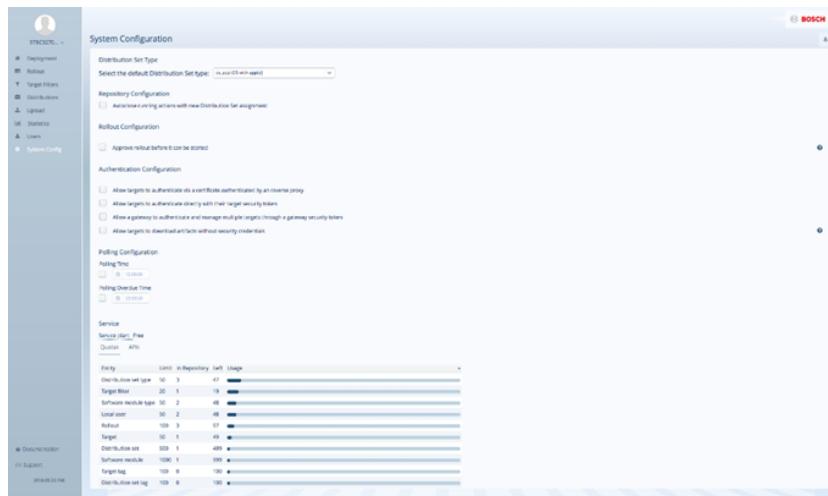
- Role Management: Nếu người dùng đã định cấu hình nhà cung cấp của riêng mình, chế độ xem Quản lý người dùng sẽ được thay thế bằng chế độ xem Quản lý vai trò. Tính năng Quản lý vai trò cho phép quản trị các vai trò và chỉ định các quyền của Triển khai IoT cho các vai trò được xác định trong nhà cung cấp danh tính bên ngoài.

Hình 2.24: *Role Management*

- Statistics: Tổng quan thống kê và nội dung lưu trữ.

Hình 2.25: *Statistics*

- System Configuration: Chế độ xem này cho phép người dùng định cấu hình cài đặt toàn cầu của người dùng cho Triển khai IoT.

Hình 2.26: *System Configuration*

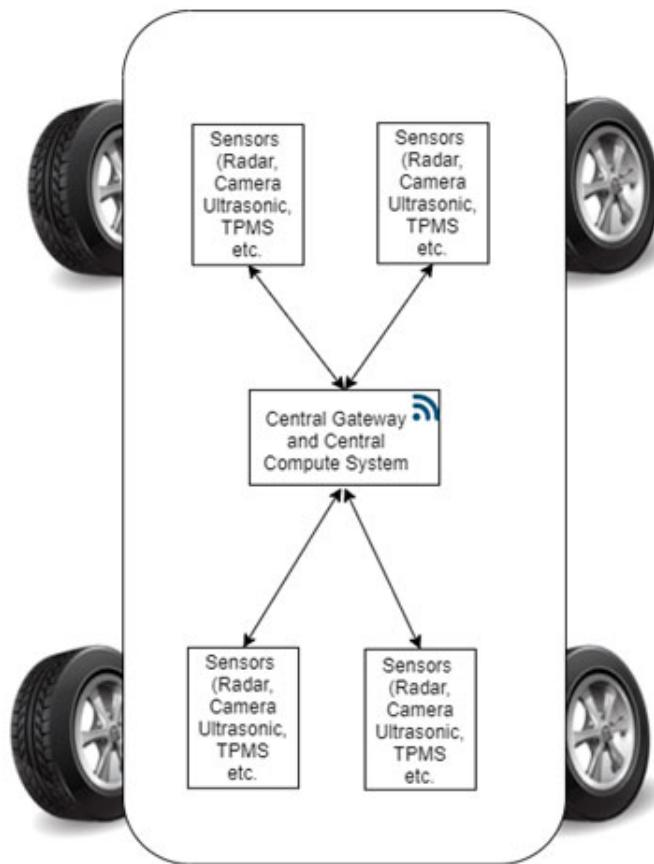
## 2.5. CENTRAL GATEWAY MODULE

Trong những ngày đầu, các nhà sản xuất phương tiện sử dụng một sợi dây để kết nối các công tắc và bộ truyền động như cần gạt nước, đồng hồ tốc độ và đèn. Trong các phương tiện ngày nay, độ phức tạp của mạng liên lạc đã tăng lên nhiều lần do nhu cầu về Chức năng hỗ trợ người lái nâng cao (Advance Driver Assistance Functions - ADAS), truyền dữ liệu, an toàn, bảo mật và các yêu cầu tuân thủ. Các giao thức/giao diện ô tô ở dạng CAN, LIN, Flex-ray và Ethernet đã được thiết kế để hỗ trợ truyền dữ liệu tốc độ cao giữa vô số cảm biến và bộ truyền động.

Việc liên lạc/truyền dữ liệu giữa tất cả các giao thức ô tô và việc liên lạc/truyền dữ liệu giữa phương tiện và đám mây phải được quản lý bởi Central Gateway.

Central Gateway là một hub để kết nối và truyền dữ liệu qua nhiều mạng khác nhau trong các phương tiện theo cách thức an toàn và đáng tin cậy. Nó được cho là cung cấp sự cách ly vật lý giữa các ECU khác nhau (điều khiển các cảm biến và bộ truyền động khác nhau) và để định tuyến tín hiệu giữa các miền chức năng trong xe. Ví dụ: Hệ thống truyền động, khung gầm và an toàn, kiểm soát thân xe, viễn thông,... và chia sẻ dữ liệu. Nói tóm lại, Central Gateway hoạt động như một bộ định tuyến và cho phép liên lạc giữa các dịch vụ đa giao thức và kết nối các thiết bị lại với nhau. [13]

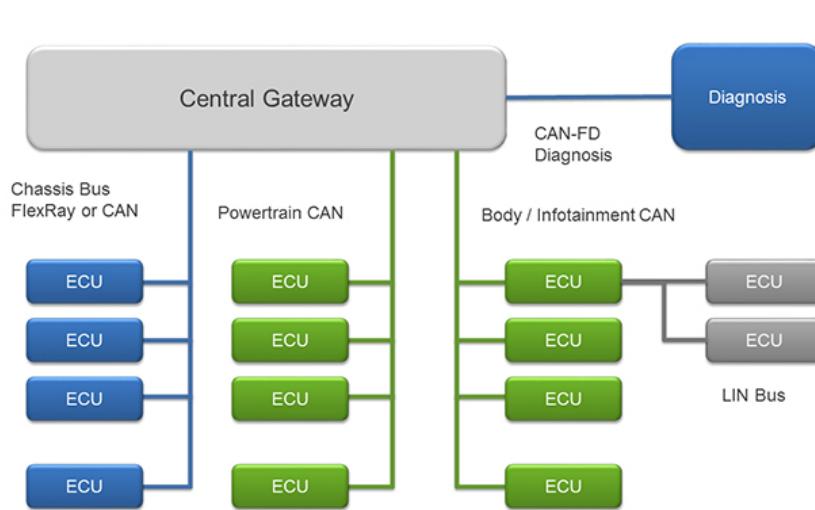
2



Hình 2.27: Minh họa về mô hình Central Gateway

Một ví dụ điển hình về Central Gateway là rào chắn đường mới được đặt (được đặt do tai nạn hoặc công trình xây dựng) trên đường cao tốc. Việc sử dụng dữ liệu GPS, được trích xuất từ thiết bị định vị và thông tin chuyển động của xe, được trích xuất từ thiết bị điều khiển Chương trình Cân bằng điện tử (Electronic Stability Program - ESP), có thể được đối chiếu với dữ liệu bộ truyền động và cảm biến khác. Dữ liệu đối chiếu này sau đó được truyền đến một dịch vụ đám mây. Sau khi thu thập vô số dữ liệu từ các phương tiện khác nhau đi trên cùng một đường cao tốc, phương tiện đó có thể xác định vị trí chính xác của rào chắn đường trên đường cao tốc. Thông báo này có thể được gửi đến các phương tiện khác đang đến gần rào chắn đường. Điều này sẽ giúp họ hoàn toàn tránh được rào cản, bằng cách thay đổi lộ trình di chuyển của phương tiện đến điểm đến hoặc ít nhất là khiến người

lái xe nhận thức được các vân đê ở phía trước.



Hình 2.28: Kiến trúc mạng ô tô ngày nay [6]

## 2.6. RASPBERRY PI

### 2.6.1. KHÁI NIỆM

Raspberry Pi là một máy tính mini có kích thước bằng thẻ tín dụng có thể tương thích với bất kỳ thiết bị phần cứng đầu vào và đầu ra nào như màn hình, TV, chuột hoặc bàn phím - chuyển đổi thiết lập thành một PC chính thức một cách hiệu quả với chi phí thấp. [3]

Người ta có thể sử dụng Raspberry Pi cho nhiều mục đích khác nhau, bao gồm học ngôn ngữ lập trình và quản lý mạng phối hợp. Nó đa chức năng và thậm chí còn trở nên phổ biến hơn trong những năm gần đây.

### 2.6.2. PHÂN LOẠI RASPBERRY PI

- **Raspberry Pi 1**

Raspberry Pi 1 Model B được phát hành vào năm 2012. Nó được dùng làm kích thước cơ sở cho các bản phát hành trong tương lai. Ban đầu, nó có 26 chân GPIO, dung lượng RAM 256 MB và một lõi CPU. Chúng ta không thể sử dụng nó cho các tác vụ nặng với nhu cầu xử lý cao. Vào năm 2014,

Raspberry Pi B+ đã được phát hành với dung lượng RAM khởi điểm là 512MB và 40 chân GPIO, trở thành tiêu chuẩn trên tất cả các kiểu Raspberry khác. Raspberry Pi Model B+ được bán với giá 25 đô la và đi kèm với bốn cổng USB và kết nối Ethernet. Pi 1 Model A+ (20\$) có thể được xem xét để có tốc độ xử lý CPU nhanh hơn, nhưng nó không có kết nối Ethernet.

### • Raspberry Pi 2 B

Vào tháng 2 năm 2015, Raspberry đã phát hành mô hình 2B. So với các bản phát hành trước, Raspberry Pi 2 B đã cải thiện đáng kể, cụ thể là về bộ nhớ và tốc độ. Dung lượng RAM được tăng lên 1GB. Pi 2B có kích thước tiêu chuẩn, với 4 cổng USB. Nó hiện có giá khoảng 35\$.

### • Raspberry Pi Zero

Đây là mẫu Raspberry rẻ nhất do hãng sản xuất. Người ta có thể mua nó với giá thấp nhất là 5\$, điều này khá ấn tượng khi xét đến mức độ chức năng của nó. Mặc dù không phải là model đầu tiên được phát hành, nhưng nó tự hào có kích thước nhỏ hơn, gọn hơn so với Raspberry Pi 1. Raspberry Pi Zero có cùng bộ xử lý và RAM (512 MB) như Pi 1 Model B+. Raspberry Pi Zero không đi kèm với Wi-Fi hoặc Bluetooth, nhưng có thể truy cập internet qua USB.

Phiên bản đắt hơn một chút, Raspberry Pi Zero W, đi kèm với Bluetooth 4.0 và kết nối Wi-Fi 802.11n tích hợp. Đối với các dự án yêu cầu chân GPIO, các phiên bản khác của Raspberry Pi có thể phù hợp hơn.

### • Raspberry Pi 3

Raspberry Pi 3 B được phát hành vào năm 2016. Phiên bản B+, ra mắt vào năm 2018, có thể tự hào về bộ xử lý, Ethernet (802.11ac) và Wi-Fi nhanh hơn phiên bản trước. Nói chung, Raspberry PI 3 cung cấp cho người dùng nhiều mục đích sử dụng. Nó đi kèm với cổng HDMI và USB tiêu chuẩn, RAM 1GB, kết nối Wi-Fi và Bluetooth bên cạnh Ethernet đã hoạt động. Một điều đáng chú ý ở model này là nó không tỏa nhiều nhiệt hay tiêu tốn quá nhiều điện năng. Điều này làm cho nó phù hợp với các dự án yêu cầu làm mát thụ động và có thể được mua với giá 35\$.

### • Raspberry Pi 4B

Được phát hành vào năm 2019, Raspberry 4B là một cải tiến vượt bậc so với các phiên bản trước đó, với dung lượng bộ nhớ thay đổi từ RAM 2GB đến RAM 8GB. Nó cũng có bộ xử lý 1.5 GHz nhanh hơn và kết hợp tốt giữa các cổng USB 2.0 và 3.0. Pi 4B là một mẫu Raspberry lý tưởng vì nó phù hợp với hầu hết mọi trường hợp sử dụng với dung lượng RAM cao hơn để

đáp ứng ngay cả những lập trình viên chuyên dụng nhất. Tùy thuộc vào bộ nhớ, giá dao động từ 35\$ đến 75\$.

Vì vậy, với giá thành phù hợp và tính năng vượt trội, chúng em quyết định sử dụng Raspberry Pi 4B làm Central Gateway cho đề tài này.

2



Hình 2.29: *Raspberry Pi 4B*

Raspberry Type	Module	Memory	Ether-net	Wire-less	GPIO	Release
Raspberry Pi 1	B	256MB 512MB	Yes	No	26 pin	2/2012 10/2012 2013
	A	256MB	No			2014
	B+ A+	512MB	Yes No		40 pin	
Raspberry Pi 2	B	1GB	Yes	No	40 pin	2015
Raspberry Pi Zero	Zero W/WH 2W	512MB	No	No	40 pin	2015
				Yes		2017 2021
Raspberry Pi 3	B	1GB	Yes	Yes	40 pin	2016
	A+	512MB	No			2018
	B+	1GB	Yes			
Raspberry Pi 4	B	1GB	Yes	Yes	40 pin	2019
		2GB				
		4GB				
		8GB				2020
	400	4GB				

Bảng 2.50: Bảng các loại Raspberry Pi [12]

## 2.7. DYNAMIC VIEW TRONG HỆ THỐNG Ô TÔ

### Flash Bootloader

Khi khởi động hệ thống sẽ có một chương trình chạy đầu tiên có trách nhiệm kiểm tra và khởi tạo các tác vụ khác bao gồm cả hệ điều hành từ đó giúp toàn bộ hệ thống trên xe đi vào hoạt động, người ta gọi bộ phận đảm nhận công việc đó là BootBlock. Nó là một phần quan trọng của các thiết bị như điện thoại thông minh, máy tính, máy tính bảng và các thiết bị nhúng.

Bootblock có thể chia thành 2 phần: Bootmanager và Bootloader[14]. Ở mỗi hệ thống khác nhau, 2 bộ phận này có thể có nhiều tên gọi khác nhau như:

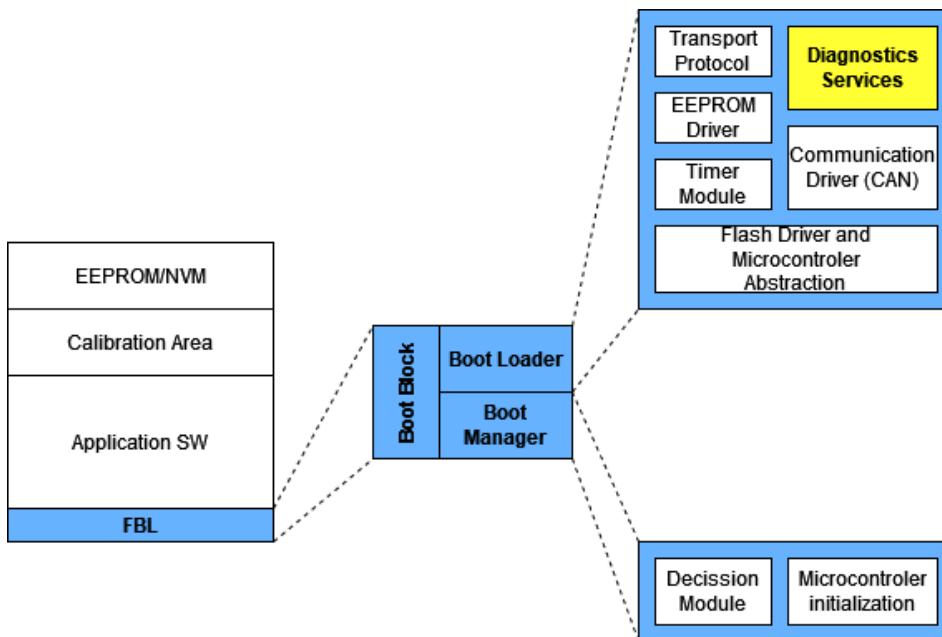
- Bootmanager có thể gọi là StartupBlock (SB) hoặc BootCtrl.  
Là một phần của Bootblock, sau khi ECU khởi động nó sẽ chạy các tác vụ như khởi tạo ECU và các mô-đun khác trên ECU, tự kiểm tra phần cứng ECU, kiểm tra tính toàn vẹn của vùng Application Software (ASW), Database... trong quá trình khởi động.

- Bootloader có thể gọi là CustomerBoot (CB), Flashloader or FBL.

Dùng trong quá trình flashing phần mềm bằng CAN, FlexRay... Các quá trình thực thi chương trình cũng như các dịch vụ liên quan đã được OEM định nghĩa.

Tóm lại, Bootblock là chương trình quản lý quá trình khởi động của hệ thống và có vai trò quan trọng trong đảm bảo tính ổn định và hiệu suất của hệ thống xe, đảm nhận các tác vụ liên quan đến cập nhật phần mềm cho xe.

2

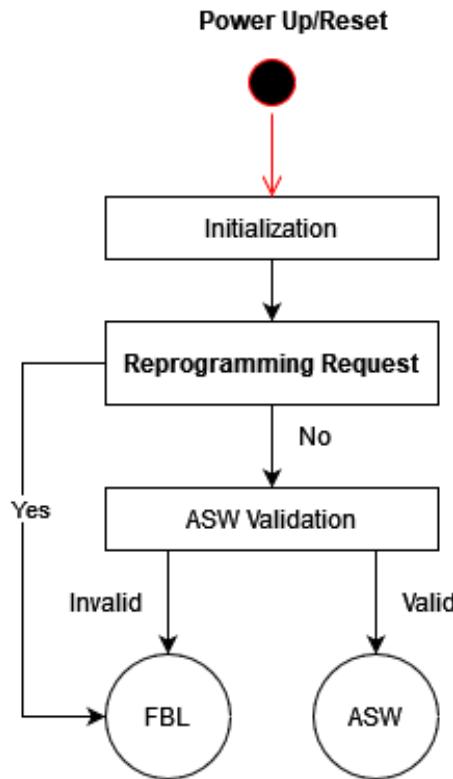


Hình 2.30: Typical FBL in ECU SW Architecture

### Quá trình khởi động

Quá trình khởi động của hệ thống được minh họa bằng hình dưới đây :

2



Hình 2.31: *Start up Processes*

### Flash Layout



Hình 2.32: Các phân vùng trong ECU

Trong đó:

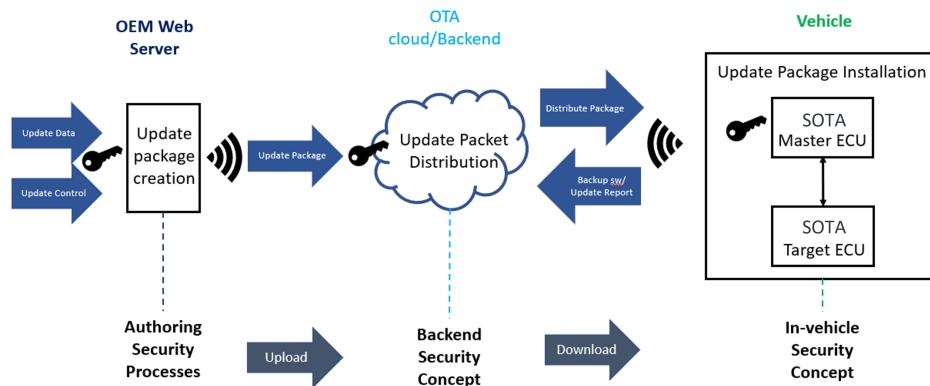
- ASW: vùng nhớ chưa dữ liệu liên quan đến code cho vùng nhớ Application
- DAT: vùng nhớ chứa dữ liệu như dataset, các biến calibration...
- EcuSecu: vùng nhớ bảo mật, chứa các dữ liệu nhạy cảm cần bảo vệ và các cơ chế bảo mật chống xâm nhập từ bên thứ 3 của ECU

## 2.8. PHƯƠNG ÁN CẬP NHẬT PHẦN MỀM OVER THE AIR

Cập nhật phần mềm Over The Air có thể chia thành 3 phần chính:[16]

2

- Update Package Creation (Authoring)
- Update Package Distribution (Backend)
- Update Package Installation/Verification/Activation (Vehicle)



Hình 2.33: Minh họa cơ chế OTA

### Update Package Creation (Authoring)

Ở bước Authoring trước tiên ta sẽ tải phần mềm mà ta muốn cập nhật (như file .hex mới) lên một nơi lưu trữ (OTA Cloud/Backend).

### Update Package Distribution (Backend)

Backend dùng cho ứng dụng OTA sẽ có 3 phần chính:

- **Vehicle/Bike Management:** Quản lý thông tin về dòng xe
- **Software Management:** Quản lý các phần mềm được cập nhật
- **Campaign Management:** Quản lý tác vụ cập nhật (lên kế hoạch cập nhật “dòng xe - phần mềm - thời gian cập nhật”).

### Update Package Installation

**“(SOTA) update”** - là toàn bộ quá trình cập nhật phần mềm cho ECU (hoặc nhiều ECU). Bao gồm toàn bộ quá trình: Download, Installation, Verification, Activation, Rollback (nếu cần thiết).

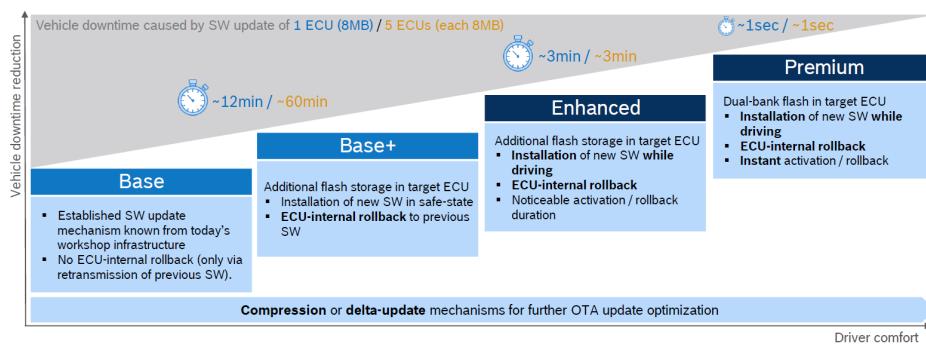
- **Download:** Tải tất cả phần mềm cần thiết để cập nhật cho target ECU, tải từ backend server tới OTA Master ECU.
  - **Installation:** Quá trình chuyển phần mềm mới cập nhật từ OTA Master ECU đến Target ECU.
  - **Verification:** Phải được hiện thực cụ thể, quá trình này sẽ đảm bảo tính đúng đắn của phần mềm mới được flash vào ECU.
  - **Activation:** Kích hoạt (activate) phần mềm mới vừa cập nhật (**n**) , đồng thời tạo ra một bản sau của phần mềm cũ (**n-1** - được dùng trong quá trình rollback).
- Lưu ý:* Quá trình Activation phải được thực hiện trong điều kiện phương tiện dừng ở một nơi an toàn (**vehicle-safe state**). Trong quá trình này Target ECU cũng sẽ không có sẵn cho các mạng giao tiếp khác.
- **Rollback:** Trong quá trình này, phần mềm trước đó (**n-1**) sẽ được phục hồi.

### Các chiến lược cập nhật cơ bản[17]

Chiến lược để triển khai FOTA rất đa dạng, nhưng đều cho phép các nhà sản xuất xe sửa chữa, bảo trì và cải thiện chất lượng sản phẩm thông qua việc điều khiển cập nhật phần mềm mới lên xe.

Các chiến lược triển khai này có thể tóm tắt thành các loại chính như sau: [8]

- OTA Base
- OTA Base+
- OTA Enhanced
- OTA Premium



Hình 2.34: Phân loại OTA

### OTA Base

OTA Base giống với việc cập nhật phần mềm bằng **diagnostic tester** ở trong trạm sửa xe. Trong kiến trúc này OTA Master sử dụng flash-bootloader của Target ECU để cập nhật phần mềm và thực thi các tác vụ chuẩn đoán (diagnostic) như bình thường. Target ECU có thể chuyển đổi việc cập nhật phần mềm từ Master ECU (kết nối với OTA cloud/Backend) hoặc từ diagnostic tester (thông qua kết nối OBD). Nếu xảy ra lỗi khi cập nhật, OTA Master có thể thực thi lại phần mềm **n-1** trước đó (phần mềm **n-1** đã được sao lưu và lưu trữ trong OTA Backend).

### OTA Base+

OTA Base+ khá giống với **OTA Base** nhưng có khả năng thực hiện ECU-internal rollback. Vì vậy bản thân Target ECU phải có đủ dung lượng để có thể chứa đồng thời bản sao lưu phần mềm hiện tại và phần mềm cập nhật mới.

Trong cả hai quá trình tải (download) và kích hoạt (activation) phần mềm đều được thực hiện bởi flash bootloader và phương tiện phải được giữ trong trạng thái an toàn.

### OTA Enhanced

OTA Enhanced ra đời nhằm mục đích rút ngắn thời gian phương tiện không được sử dụng vì cập nhật phần mềm.

Việc tải phần mềm sẽ được hoàn thành trong tác vụ nền của ECU (trong chế độ ASW), khi xe vẫn đang chạy). Quá trình kích hoạt phần mềm (activation) sẽ

được thực hiện khi xe đang ở trạng thái an toàn bởi flash bootloader.

ECU internal rollback vẫn được hỗ trợ trong kiểu triển khai này.

### OTA Premium

2

OTA Premium giống với **OTA Enhanced** nhưng có thêm một số tính năng :

- Phần mềm mới có thể được kích hoạt trực tiếp tại vị trí bộ nhớ nó được tải về (sau bước download), làm được như thế là nhờ bộ nhớ kép (dual-bank). Phương thức Dual-bank cho phép ECU chọn vùng nhớ bắt đầu thực thi khi khởi động. Khi xảy ra Rollback, ECU chỉ việc chọn vùng nhớ chưa phần mềm cũ (**n-1**) và khởi động lại. Rollback không thể thực hiện khi xe vẫn đang chạy.
- Quá trình tải phần mềm sẽ được hiện thực bởi ASW. Và xe phải ở trạng thái an toàn khi chuyển giữa hai vùng nhớ.

Concept	Download	Activation	ECU Internal RollBack	Dual Bank
OTA Base	Flash Bootloader	Flash Bootloader	No	No
OTA Base+	Flash Bootloader	Flash Bootloader	Yes	No
OTA Enhanced	ASW	Flash Bootloader	Yes	No
OTA Premium	ASW	Flash Bootloader	Yes	Yes
Note	Flash Bootloader	Vehicle must be in safe-state		

Bảng 2.51: Tổng hợp phân loại OTA

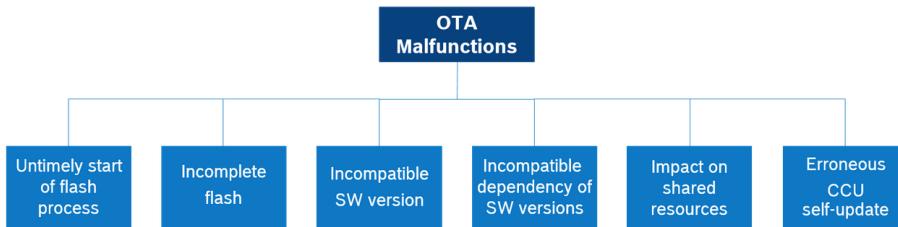
### SOTA Security Concept [8]

Kiến trúc bảo mật được thiết kế để ngăn chặn tấn công có thể ảnh hưởng nghiêm trọng đến hoạt động của ECU trong quá trình cài đặt phần mềm. Kiến trúc bảo mật này được chia thành 3 phần chính:

- Update Package Creation: Dành cho quá trình authoring, quá trình này giúp đảm bảo chỉ có người có quyền (admin right) mới có thể upload phần mềm cập nhật mới lên OTA Cloud/Backend.
- Update Package Distribution: Dành cho quá trình kiểm thử liên quan đến nội dung, phần mềm (các package), lên kế hoạch cập nhật phần mềm. Đảm bảo chỉ có những gói tin đúng là được cật nhật lên OTA cloud/ Backend.
- Update Package Installation: Dành cho những tác vụ cập nhật cho thiết bị trên xe. Đảm bảo ngăn chặn tấn công từ xa trong quá trình installing tới các thiết bị trong xe, ngăn chặn tấn công các cơ chế bảo mật của các thiết bị như ECU.

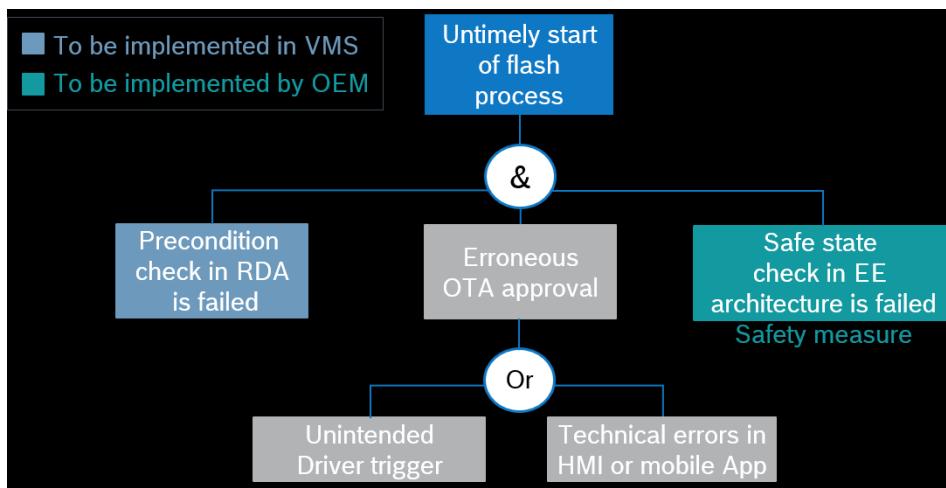
## 2.9. CÁC SỰ CỐ KHI CẬP NHẬT OTA

Cập nhật OTA cho phép flash phần mềm từ xa đến ECU của ô tô qua mạng không dây bằng cách thao tác trên giao diện web hoặc API web. Việc cập nhật OTA từ xa sẽ xuất hiện một số sự cố tiềm ẩn, các sự cố tiềm ẩn của cập nhật OTA được thể hiện trong hình bên dưới. [8]



Hình 2.35: Một số rủi ro khi cập nhật OTA

- Untimely start of flash process - **Bắt đầu quá trình flash không đúng lúc**



Hình 2.36: Bắt đầu quá trình flash không đúng lúc

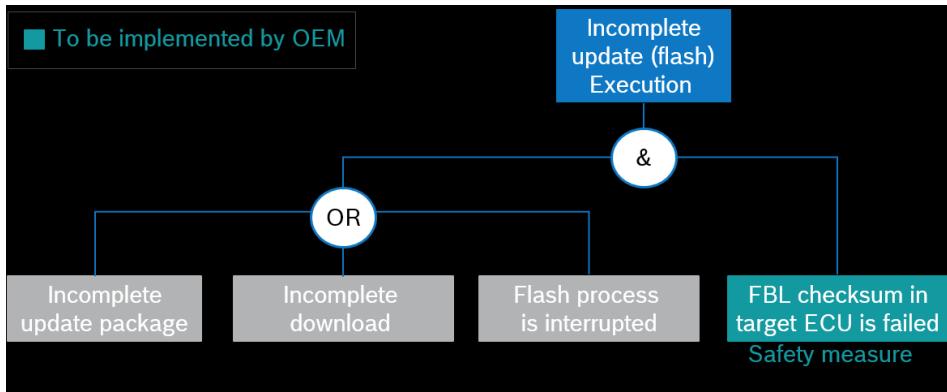
Việc flash ECU trong khi lái xe có thể dẫn đến các tình huống nguy hiểm, ảnh hưởng nghiêm trọng tới an toàn của người trên xe. Vì vậy, việc flash ECU phải được bắt đầu khi xe đang ở trạng thái an toàn.

Quá trình thực thi Cập nhật OTA có hai bước riêng biệt: Kiểm tra điều kiện tiên quyết trong thiết bị kết nối và phê duyệt OTA. Trong kiểm tra điều kiện tiên quyết, các trạng thái xe cần thiết, ví dụ: Động cơ đang chạy, tốc độ xe và trạng thái phanh phải được xác minh. Người dùng phương tiện (người lái

xe hoặc người điều khiển phương tiện) có thể thực hiện phê duyệt cập nhật thông qua GUI. Hơn nữa, việc cập nhật phải bao gồm các biện pháp để ngăn quá trình khởi động flash không đúng lúc.

## 2

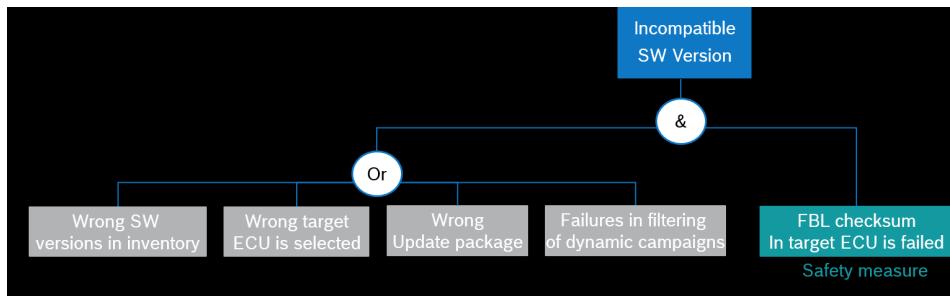
- Incomplete flash process - Quá trình flash chưa hoàn thành



Hình 2.37: Quá trình flash chưa hoàn thành

- OEM chịu trách nhiệm tạo và tải lên các gói cập nhật hoàn chỉnh.
- Việc thực hiện cập nhật OTA chỉ bắt đầu khi các gói cập nhật được tải xuống hoàn toàn.
- Phải có biện pháp lưu lại phần mềm cũ khi việc flash bị gián đoạn.

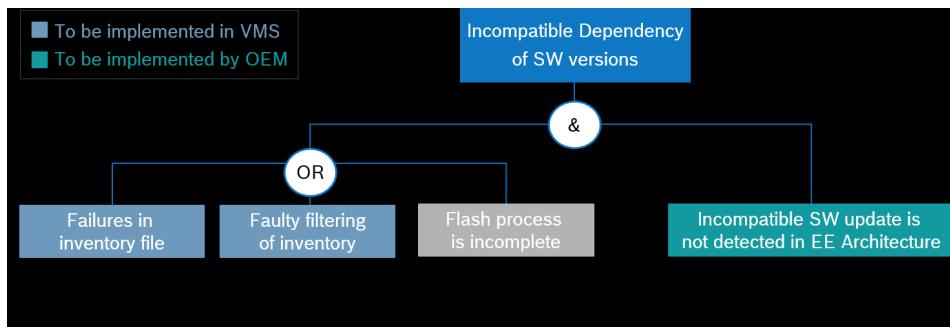
- Incompatible SW version - Phiên bản phần mềm không tương thích



Hình 2.38: Phiên bản phần mềm không tương thích

Việc gán sai các gói cập nhật cho một phương tiện hoặc một nhóm phương tiện có thể gây ra những tác động nguy hiểm đối với hoạt động của phương tiện. Để tránh hành vi không thể đoán trước của phương tiện do hậu quả của việc cập nhật sai dữ liệu, OEM cần phải đảm bảo việc tạo và tải các gói cập nhật là đúng và tương thích.

- Incompatible dependency - Phụ thuộc không tương thích



Hình 2.39: Phụ thuộc không tương thích

Để flash nhiều ECU, dữ liệu flash thích hợp và quy trình cập nhật của các Target ECU được kết hợp trong một gói cập nhật duy nhất. Sự phụ thuộc lẫn nhau của các ECU riêng lẻ phải được xem xét khi tạo gói.

Ví dụ: Động cơ và ECU phanh, có thể dẫn đến các hậu quả nghiêm trọng về an toàn. OEM phải có trách nhiệm xác minh sự phụ thuộc của các phiên bản SW.

## 2

### • Impact on shared resources - Tác động đến tài nguyên được chia sẻ

Quá trình OTA có thể có tác động đến các tài nguyên được chia sẻ. Ví dụ: Tải các gói phân phối xuống ECU có thể dẫn đến quá tải tạm thời cho các bus nội bộ. Để tránh hành vi không thể đoán trước của phương tiện:

- Phải đảm bảo rằng các bus giao tiếp trong xe (ví dụ như CAN bus) không bị quá tải trong quá trình OTA.
- Phải đảm bảo rằng có đủ điện cho quá trình OTA của các ECU mục tiêu cũng như khả năng tự cập nhật của CCU.

# 3

## ĐỀ XUẤT HỆ THỐNG

Trong chương này, chúng ta sẽ thiết kế kiến trúc hệ thống nhằm phục vụ yêu cầu của đề tài bao gồm kiến trúc tổng quan và kiến trúc của từng chi tiết trong hệ thống.

### 3.1. PHÂN TÍCH HỆ THỐNG

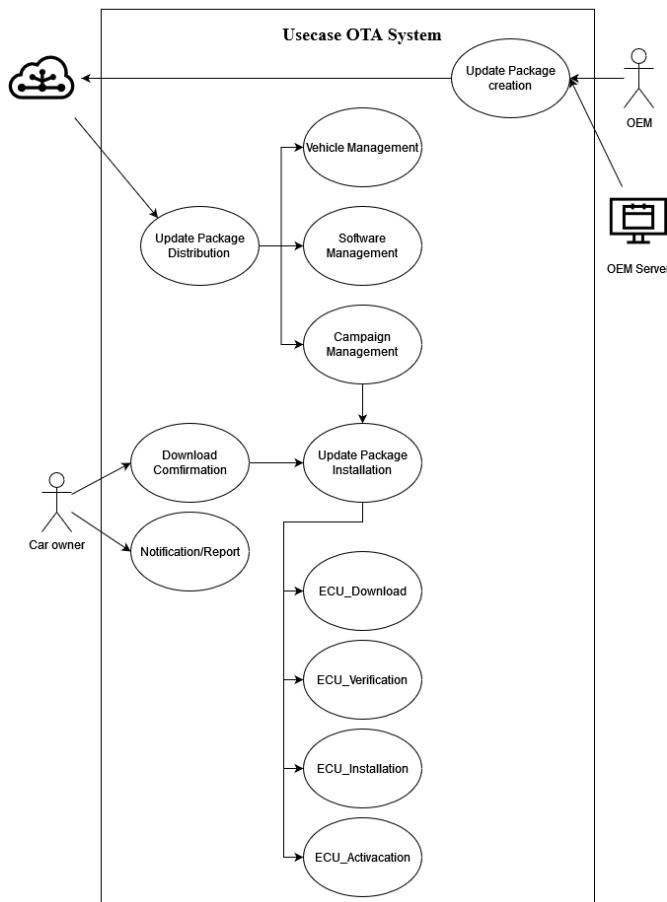
#### 3.1.1. PHÂN TÍCH YÊU CẦU

1. ECU sẽ cung cấp thông tin để nhận diện được SW và HW đã được cài đặt.
2. Việc kích hoạt SW mới sẽ được kích hoạt bởi OTA. Việc kích hoạt sẽ được chấp nhận bởi ECU và quá trình kích hoạt bắt đầu trong các điều kiện sau:
  - Xác minh tính toàn vẹn và tính xác thực của SW mới.
  - Các điều kiện tiên quyết cập nhật an toàn cụ thể của ECU được đáp ứng.
  - Các điều kiện tiên quyết cập nhật an toàn cụ thể của ECU phải phù hợp với các yêu cầu an toàn chức năng cụ thể của ECU.
3. Có thể khôi phục SW trước đó. Trong trường hợp không thể cài đặt SW mới, ECU sẽ tự quay lại SW cũ trước đó.
4. ECU sẽ chỉ ra giá trị thời gian (trung bình) dự kiến để hoàn thành việc cài đặt.
5. Việc cài đặt SW mới sẽ được bảo vệ bởi các cơ chế bảo mật tương ứng.
6. Gián đoạn trong việc cài đặt (ví dụ: ECU bị mất điện). Quá trình sẽ được khởi động lại trong các điều kiện hợp lệ.

7. Không được vi phạm các mục tiêu an toàn cụ thể của ECU tại bất kỳ thời điểm nào trong toàn bộ quá trình cập nhật OTA.

Với hai đối tượng sử dụng hệ thống là OEM và người dùng, Usecase diagram thể hiện những chức năng của hệ thống mà mỗi đối tượng sử dụng như sau:

3



Hình 3.1: Usecase các chức năng cơ bản của hệ thống OTA

### 3.1.2. PHÂN TÍCH RỦI RO

Các rủi ro khi cật nhật khi cập nhật phần mềm OTA có thể chia thành các loại như bảng dưới đây sau:

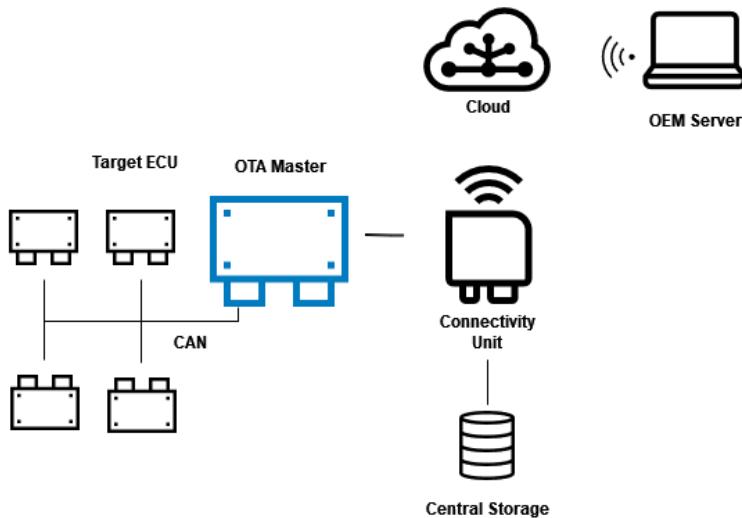
Rủi ro	Giải pháp	
Rủi ro về sự cố trong quá trình cập nhật	Bắt đầu quá trình flash không đúng lúc  Phiên bản phần mềm không tương thích  Tác động đến tài nguyên được chia sẻ	Đảm bảo xe đang trong trạng thái safe-state trước khi flash bằng cách thực hiện bước Pre-programming.  Quá trình flash chỉ bắt đầu khi có xác nhận từ chủ xe.  Trước khi flash tiến hành kiểm tra cái thông tin của xe như VIN thông qua UDS service (0x22) Các phần mềm trong chiến dịch cập nhật đã được OEM đảm bảo về tính tương thích với ECU trên xe.  Đảm bảo xe đang trong trạng thái safe-state trước khi flash.
Rủi ro về bảo mật	Tấn công trong quá trình Update Package Creation  Tấn công trong quá trình Update Package Distribution  Tấn công trong quá trình Update Package Installation	Authoring Security Processes  Backend Security concept  In-Vehicle Security concept
Rủi ro về tính toàn vẹn của dữ liệu	Có thể bị mất mát dữ liệu trong quá trình flash	Tính toàn vẹn được đảm bảo bằng cách tính toán checksum trong quá trình flashing.
Rủi ro về các tác nhân bên ngoài gây ra lỗi	Gián đoạn quá trình tải do mất điện	Quá trình flashing sẽ được khởi động lại khi các điều kiện hợp lệ.

## 3.2. KIẾN TRÚC TỔNG QUAN

### 3.2.1. KIẾN TRÚC HỆ THỐNG OTA CƠ BẢN

Kiến trúc hệ thống OTA cơ bản được triển khai cho passenger car thường được tóm tắt như hình dưới đây: [17]

3



Hình 3.2: Kiến trúc hệ thống OTA cơ bản

Chức năng chính của từng phần trong hệ thống:

**OEM Server:** Nơi phát hành phần mềm mới, đảm bảo tính toàn vẹn và an toàn của phần mềm. Bao gồm kiểm tra, phê duyệt và chứng nhận phần mềm trước khi phát hành.

**Clouds:** Nơi lưu trữ phần mềm mới; kiểm soát và triển khai các chiến lược cập nhật phần mềm cho các dòng xe.

**Connectivity unit:** Đóng vai trò là giao diện tương tác giữa Cloud và OTA Master ECU, đảm nhiệm việc tải phần mềm cập nhật mới từ Cloud về central storage để OTA Master ECU tiến hành các bước cập nhật phần mềm.

**Central storage:** Nơi lưu trữ phần mềm cập nhật mới được Connectivity Unit tải về.

**OTA Master:** chịu trách nhiệm điều phối toàn bộ quá trình cật nhật phần mềm bên trong xe. Bao gồm:

- Xác minh và xác thực phần mềm cập nhật.
- Đóng vai trò trung gian giúp các Target ECUs Installation/Verification/Activation phần mềm mới bằng cách gửi các lệnh UDS Services và dữ liệu của phần mềm mới tới Target ECUs.
- Kiểm tra xe có đang ở trong tình trạng an toàn( safe state) hay không để tiến hành cập nhật phần mềm.

Thông thường, người ta sẽ sử dụng Central Gateway làm OTA Master ECU trong hệ thống OTA.

**Target ECUs:** là những ECUs cần được cập nhật phần mềm mới, sau khi hoàn thành bước installation phần mềm mới, Target ECUs sẽ nhận được lệnh và tiến hành các bước Verification/Activation phần mềm.

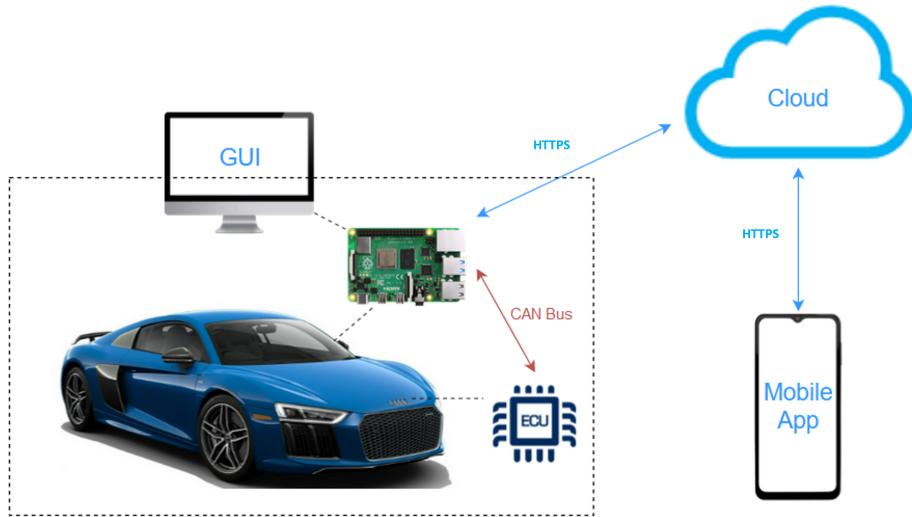
### 3.2.2. CÁC THÀNH PHẦN CƠ BẢN TRONG HỆ THỐNG

Dựa trên kiến trúc hệ thống cơ bản bên trên, nhóm đã tiến hành phân tích và đánh giá các giới hạn về kỹ thuật, thiết bị, chức năng. Từ đó, nhóm đã quyết định thực hiện sự thay đổi và triển khai một kiến trúc hệ thống mới, được thiết kế đặc biệt để đáp ứng yêu cầu và mục tiêu của đề tài này.

Nhóm đã xây dựng kiến trúc tổng quan của hệ thống và chia làm 4 khối: ECU động cơ, Central Gateway, Bosch IoT Rollouts và giao diện tương tác cho người dùng( Mobile App và GUI).

Dưới đây thể hiện vị trí, đường kết nối và cách thức kết nối của các khối:

3



Hình 3.3: Sơ đồ kiến trúc tổng quan

#### **Bosch IoT Rollouts:** Giữ vai trò của Clouds

Việc sử dụng Bosch IoT Rollouts, quá trình Authoring diễn ra dễ dàng và đáng tin cậy nhờ có cái tính năng và cơ chế bảo mật có sẵn. Đồng thời, Bosch IoT Rollout cũng cho phép người dùng hiệu chỉnh các trường trong Website một cách linh hoạt, phù hợp với yêu cầu của từng đối tượng và trường hợp cụ thể của doanh nghiệp.

#### **Mobile app:** Mobile App giao diện tương tác dành cho người dùng

Ngoài việc nhận thông báo bằng GUI, một Mobile App cũng được phát triển để người dùng nhận được thông báo cập nhật, dễ dàng theo dõi về phiên bản cập nhật của xe thông qua điện thoại di động cá nhân.

#### **GUI:** Giao diện tương tác giữa người dùng và OTA Master ECU.

GUI cho phép người dùng kiểm tra các phiên bản phần mềm hiện có, tùy chọn tải phần mềm và tiến hành cập nhật cho xe.

#### **Raspberry Pi 4B:** Giữ vai trò của OTA Master ECUs, Connectivity Unit và Central Storage. Là nơi thực hiện việc tải phần mềm mới từ cloud, lưu trữ và thực hiện các việc flash các bản cập nhật vào Target ECUs thông qua giao thức Standard CAN.

Việc Raspberry Pi 4B cùng lúc đảm nhận cùng lúc nhiều chức năng của hệ thống

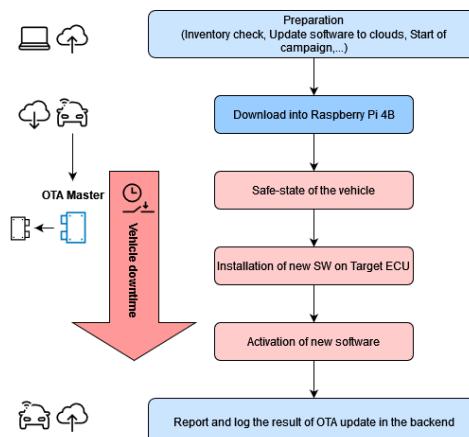
khắc phục được nhiều vấn đề và mang lại những lợi ích như:

- Giúp cho số lượng thiết bị cần thiết để hiện thực hệ thống giảm xuống mức tối thiểu nhưng vẫn đầy đủ tính năng.
- Ngoài ra nếu dùng tất cả các thiết bị như kiến trúc hệ thống cơ bản như trên (CCU, CGW) thì sẽ không thể chỉnh sửa các kết nối của phần mềm để tiến hành flash theo mong muốn vì phần mềm điều khiển bên trong ECU liên quan đến nhiều bên trong nội bộ công ty.

Khi dùng Raspberry Pi 4B thay thế sẽ giúp nhóm giải quyết vấn đề này, nhóm sẽ có thể hiện thực quy trình OTA theo mong muốn mà không cần phải chỉnh sửa các phần mềm bên trong và nhờ vậy cũng giúp nhóm hiện thực đề tài từng bước một khi trực tiếp viết phần mềm cho toàn bộ quá trình cập nhật OTA.

### 3.2.3. CHIẾN LƯỢC CẬP NHẬT OTA

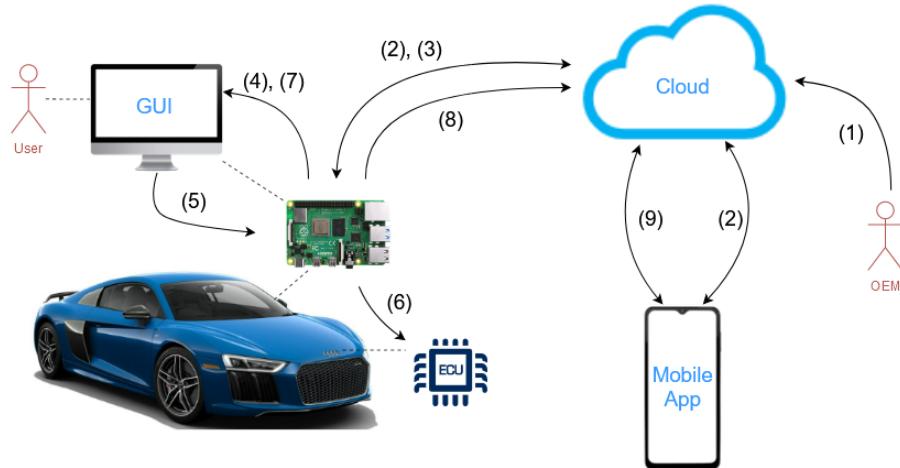
Do giới hạn phần cứng, Target ECU nhóm đang sử dụng không hỗ trợ các vùng nhớ dành riêng để OTA có thể thực hiện các tác vụ như ECU Internal Rollback hay Dual Bank, nên không thể chọn các chiến lược OTA khác như OTA Base+, OTA Enhanced, OTA Premium. Nên nhóm quyết định chọn chiến lược cập nhật **OTA Base** để hiện thực hệ thống. Các giai đoạn cập nhật của chiến lược được minh họa bằng hình dưới đây:[16]



Hình 3.4: Các giai đoạn của Chiến lược cập nhật OTA Base

Tuy hệ thống gồm 5 khối chính, nhưng công việc phát triển chỉ nằm trên 4 khối là Central Gateway, Cloud, Website và GUI. Khối ECU chỉ có vai trò được khối Central Gateway flashing phần mềm.

3



Hình 3.5: *Luồng hoạt động cơ bản của hệ thống*

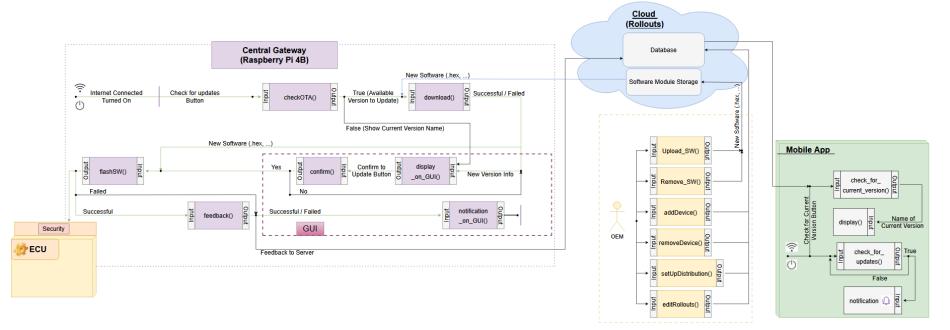
Trong đó:

- (1): OEM tải gói phần mềm mới lên Cloud.
- (2): Central Gateway và Mobile App kiểm tra phiên bản cập nhật mới hiện có và thông báo cho người dùng.
- (3): Central Gateway tải gói phần mềm mới từ Cloud.
- (4): Central Gateway hiển thị thông tin của phiên bản cập nhật mới cho người dùng qua GUI.
- (5): Người dùng xác nhận tiến hành cập nhật phần mềm.
- (6): Central Gateway thực hiện flash phần mềm mới vào ECU.
- (7): Central Gateway hiện thông báo cập nhật thành công cho người dùng.
- (8): Central Gateway gửi thông báo đã hoàn thành cập nhật của người dùng lên Cloud.
- (9): Người dùng kiểm tra lại phiên bản hiện tại của ECU (phiên bản vừa được cập nhật) qua Mobile App.

### 3.3. KIẾN TRÚC CHI TIẾT TỪNG PHẦN

#### 3.3.1. CÁC KHỐI CHỨC NĂNG TRONG HỆ THỐNG

Sơ đồ dưới đây sẽ được thể hiện lại ở các phần sau phân tích để cụ thể hơn về kiến trúc chi tiết, chức năng của mỗi khối dưới dạng function và flow-chart.



3

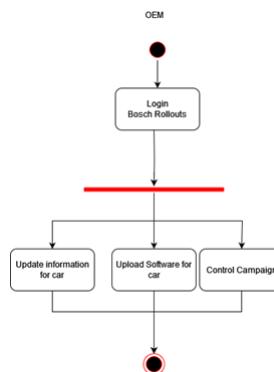
Hình 3.6: Sơ đồ chức năng của các khối trong hệ thống

#### 3.3.2. UPDATE PACKAGE CREATION: OEM WEB SERVER

Được hoàn thành hoàn toàn bởi OEM, OEM sẽ đảm bảo các phần mềm trên Bosch IoT Rollouts đều tương thích với hệ thống và có chứa dữ liệu được sử dụng cho quá trình bảo mật và xác thực phần mềm.

#### 3.3.3. UPDATE PACKAGE DISTRIBUTION: BOSCH IoT ROLLOUTS

Luồng hoạt động chính của chức năng Update package distribution được mô tả bằng hình bên dưới:

**3**

Hình 3.7: *Update package distribute function: Activity Diagram*

Update package distribution bao gồm:

- **Update information for car**

Bosch IoT Rollouts cho phép OEM có thể quản lý thông tin các dòng xe của công ty, khi muốn cập nhật phần mềm mới khách hàng có thể dựa vào những thông tin này để lọc ra những dòng xe cần cập nhật phần mềm.

Các thông tin OEM cập nhật cho xe bao gồm: Controller ID, Name, Type, Description

Trang trong Bosch IoT Rollouts thực tế.

Create new Target ✖

Controller ID \*

Name

Type  ▼

Description

\* Mandatory Field

Save  Cancel

3

Hình 3.8: *Update information for car*

- **Upload software for car**

Một gói phần mềm cập nhật mà OEM cập nhật cho xe sẽ bao gồm 1 file hex (hoặc 1 file bin).

Trang trong Bosch IoT Rollouts thực tế.

The screenshot shows the Bosch IoT Rollouts interface. On the left, there is a sidebar with the number '3'. The main area has two tabs: 'Software Module' and 'Artifact Details of testing:1'. The 'Software Module' tab shows a list of modules: 'testing' (version 1) and 'testing#2' (version 1). The 'testing' module is selected. The 'Artifact Details of testing:1' tab shows a table with one row: File name P1655CS12C4\_VK0A\_dcm.hex, Size(B) 16475657, Last modified da... Fri May 12 16:51..., and Actions. Below these tabs, there is a detailed view of the 'testing:1' module. It includes sections for 'Details', 'Description', 'Logs', and 'Metadata'. Under 'Details', it shows 'Vendor:', 'Type: Application', 'Assignment type: Software (SW)', and 'Artifact encryption: Disabled'. To the right of this detailed view, there is a section titled 'Drop Files to upload' with a large downward arrow icon and a 'Upload File' button.

Hình 3.9: *Upload software for car*

- **Control Campaign**

Sau khi đã cài nhật phần mềm lên Bosch IoT Rollouts và chọn được dòng xe muốn cập nhật, Bosch IoT Rollouts có hỗ trợ OEM tạo ra một chiến dịch cập nhật, OEM có thể cài đặt thời gian khi chiến dịch bắt đầu và kết thúc thực tế. Trang trong Bosch IoT Rollouts thực tế.

3

Create new Rollout

Name  \*

Distribution set  \*

Custom Target Filter  \*

Description

Action type  ⚡ Forced  Soft  Download Only  Time Forced 5/28/23 08:32 PM

Start type  ⏱ Manual  ➡ Auto  Scheduled 5/14/23 09:02 PM

Number of Groups

Generate the groups automatically with the specified thresholds.

Number of groups  \*

Trigger threshold  \* %

Error threshold  \*  %  Count

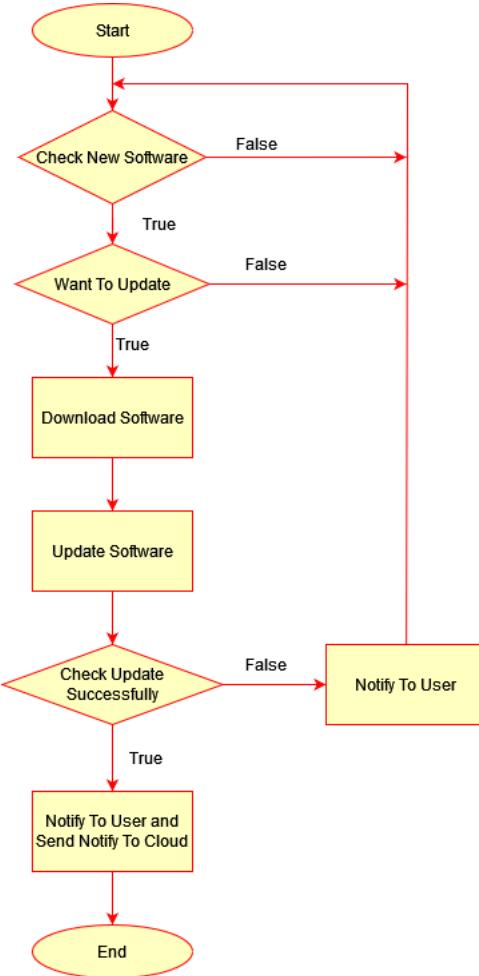
\* Mandatory Field

Hình 3.10: *Control Campaign*

### 3.3.4. UPDATE PACKAGE INSTALLATION : RASPBERRY PI 4B

#### 3.3.4.1 Các chức năng cơ bản và luồng hoạt động

3



Hình 3.11: Flowchart Raspberry Pi 4B

Raspberry Pi 4B đóng vai trò là Central Gateway, giúp kết nối các ECU và Cloud, nhằm mục đích giao tiếp giữa ECU và Cloud. Raspberry Pi 4B có các tính năng cơ bản sau:

- Kiểm tra đã có phần mềm mới cần được cập nhật hay không? Nếu có, sẽ thông báo đến người dùng thông qua GUI.
- Tải phần mềm từ Cloud.

- Nhận tín hiệu xác nhận cập nhật phần mềm từ người dùng, sau đó tiến hành cập nhật.
- Flash phần mềm vào ECU.

### 3.3.4.2 Quá trình cập nhật phần mềm mới cho Target ECU

Quá trình cập nhật phần mềm mới từ Raspberry Pi 4B vào Target ECU được miêu tả chi tiết thông qua bảng dưới đây:

Usecase	Flashing software for Target ECU
Actor	Chủ xe
Description	Raspberry Pi 4B tiến hành cập nhật phần mềm mới cho Target ECU khi có xác nhận của chủ xe thông qua GUI.
Pre-Condition	Hệ thống xe phải đang trong trạng thái an toàn (safe-state).
Postcondition	Target ECU cập nhật thành công phần mềm. Hệ thống hoạt động bình thường.
Basic flow	<ol style="list-style-type: none"> <li>1. Chủ xe xác nhận cập nhật phần mềm thông qua GUI.</li> <li>2. Raspberry xác nhận tình trạng hiện tại của xe.</li> <li>3. Raspberry tiến hành các bước cập nhật bằng cách gửi các yêu cầu UDS cho Target ECU.</li> <li>4. Reset Target ECU</li> </ol>
Alternative flow	Alternative 1 : Ở bước 1, chủ xe có thể chọn “Download and Update” hoặc “Download only” để bắt đầu quá trình flashing phần mềm cho xe.

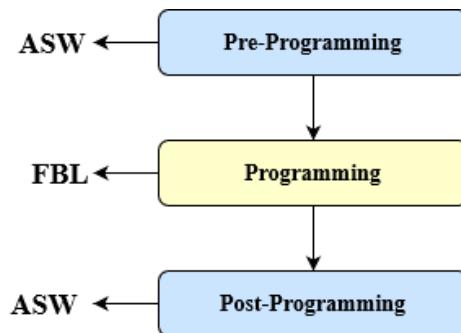
**3**

Exception flow

Exception 1 : Ở bước 2, Nếu hệ thống xe đang chạy, hệ thống xe từ chối cập nhật và đưa ra cảnh báo và hướng dẫn cho người dùng

Exception 2 : Ở bước 3, Nếu quá trình cập nhật diễn ra không thành công, hệ thống sẽ gửi thông báo cho người dùng để bắt đầu lại.

Quá trình cập nhật phần mềm mới cho Target ECU có thể được hoàn thành bằng cách gửi các yêu cầu cho Target ECU bằng dịch vụ UDS từ Raspberry Pi 4.



Hình 3.12: General procedure flow[14]

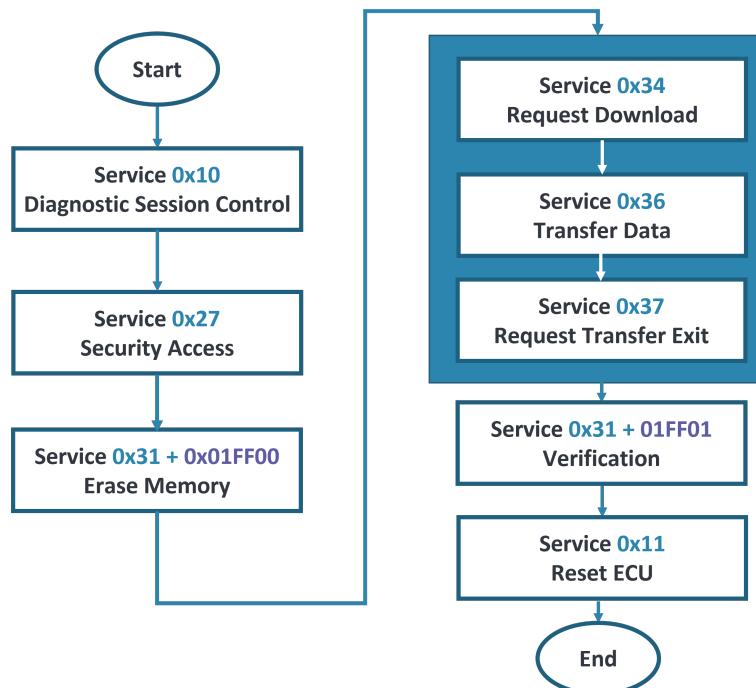
## 1. Re-programming

Trước khi chuyển quyền kiểm soát từ ASW cho Bootloader để tiến hành cập nhật thì các tiền điều kiện sẽ được thực thi để kiểm tra xe trong trạng thái an toàn để tiến hành cập nhật hay chưa. Hệ thống sẽ tiến hành kiểm tra Engine/Vehicle speed, turn off DTC, turn off Inter ECU communication. Nếu quyền điều khiển đã ở Bootloader ( xe vừa khởi động và chương trình chưa vào vùng ASW) thì quá trình này không cần thiết.

## 2. Programming

Ở bước này, Raspberry Pi 4B sẽ tiến hành gửi các lệnh dịch vụ UDS theo quy trình để thực hiện quá trình Installation phần mềm mới cho Target ECU. Raspberry Pi 4B sẽ đọc dữ liệu từ file .hex sau đó sẽ ghi dữ liệu đó vào các phân vùng trong Target ECU. Ở đây, chúng ta sẽ ghi dữ liệu vào các phân vùng CB, ASW (ASW0, ASW1, ASW2), DAT (DS0, VDS).

Quy trình cập nhật phần mềm có các bước như hình bên dưới :



3

Hình 3.13: *UDS Download Sequence*

### 3. Post-programming

Sau khi ECU Reset, hệ thống sẽ tiến hành kiểm tra Engine/Vehicle speed, cho phép DTC, cho phép Inter ECU communication, trở về default session để đảm bảo hệ thống hoạt động bình thường.

#### 3.3.5. GIAO DIỆN NGƯỜI DÙNG : GUI

GUI là màn hình trong xe kết nối trực tiếp với Raspberry Pi 4B với mục đích hiển thị giao diện cho người dùng với các tính năng cơ bản như sau:

- Thông báo có phần mềm mới từ OEM cần được cập nhật - Người dùng xác nhận việc cập nhật bằng nút nhấn Yes/No.
- Sau khi người dùng xác nhận cập nhật, GUI sẽ gửi yêu cầu cập nhật New Software đến Raspberry Pi 4B.
- Hiển thị cập nhật phần mềm thành công hoặc thất bại.
- Ngoài ra, GUI còn có chức năng hiển thị phiên bản phần mềm hiện tại.

### 3.3.6. BẢO MẬT CỦA HỆ THỐNG

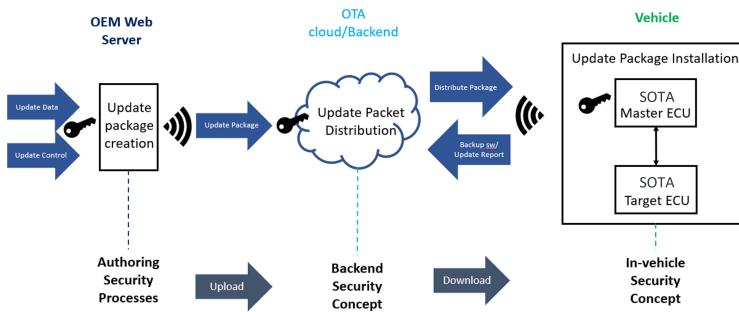
Bảo mật trong cập nhật phần mềm Over-the-air (OTA) rất quan trọng vì OTA cho phép cập nhật phần mềm từ xa, mà không cần đến trung tâm bảo trì của nhà sản xuất. Tuy nhiên, nó cũng mở ra một cửa ngõ cho các tấn công mạng và vi phạm bảo mật.

3

Nếu không có bảo mật đầy đủ trong quá trình cập nhật OTA, kẻ tấn công có thể tận dụng các lỗ hổng trong mã OTA để truy cập và kiểm soát thiết bị của người dùng, đánh cắp thông tin cá nhân hoặc tấn công mạng khác.

Do đó, việc đảm bảo an toàn và bảo mật trong quá trình OTA là rất quan trọng và cần được xem như một yếu tố cốt lõi của thiết kế hệ thống OTA.

Trong đồ án lần này, do đặc thù chuyên môn của ngành kỹ thuật máy tính nhóm đặt trọng tâm vào quá trình cập nhật phần mềm cho Target ECU, nhưng cũng đã xem xét đến các rủi ro về bảo mật và đưa ra một quy trình cụ thể như hình bên dưới



3

Hình 3.14: Minh họa cơ chế OTA

Hệ thống bảo mật ở 3 bước chính của quá trình cập nhật phần mềm :

- **Authoring Security Processes**

Người cập nhật các thông tin và dữ liệu lên Bosch IoT Rollouts sẽ được xác thực bằng việc đăng nhập. Nhờ đó tính chính xác của các thông tin trên Bosch IoT Rollouts được đảm bảo.

- **Backend Security concept**

Khi muốn tải dữ liệu từ Bosch IoT Rollouts, thì phải cung cấp được các thông tin bảo mật như (username, password, tenant, gateway token). Phần mềm được Raspberry Pi 4B tải từ Bosch Rollouts về bằng giao thức HTTPS giúp đảm bảo tính an toàn của dữ liệu.

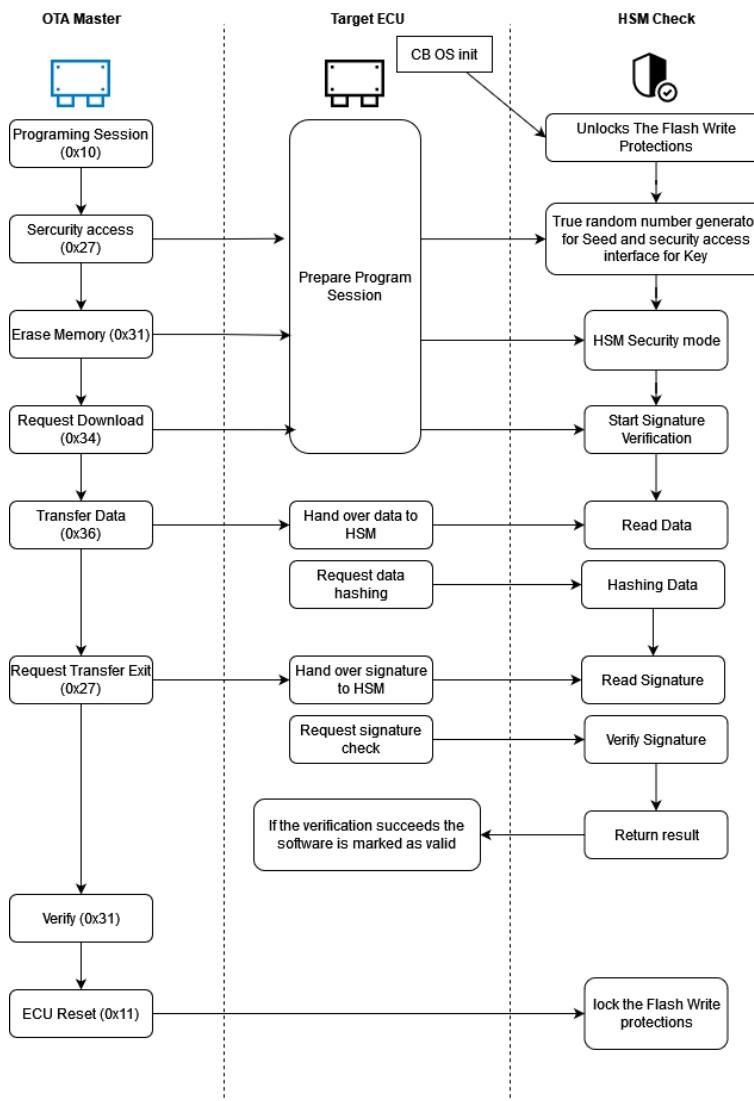
- **In-Vehicle Sercurity concept**

Sau khi vào được vùng programing, trong ECU có một vùng nhớ dành cho Hardware Security Module- một thành phần phần cứng được tích hợp vào các thiết bị điện tử của xe nhằm đảm bảo tính bảo mật của các dữ liệu nhạy cảm được lưu trữ và xử lý trong hệ thống.

Phần mềm cập nhật mới trước khi được upload lên Bosch IoT Rollouts sẽ được duy trì tính xác thực và toàn vẹn bằng cách thêm vào các chữ ký số. Trước khi quá trình flashing phần mềm mới bắt đầu, HSM sẽ tiến hành kiểm tra tính bảo mật thông qua các chữ ký số này.

Sơ đồ dưới đây miêu tả quá trình cập nhật đảm bảo tính toàn vẹn và xác thực của phần mềm:

3

Hình 3.15: *Security Flashing Flow* [14]

# 4

## HIỆN THỰC VÀ KẾT QUẢ

Trong chương này, chúng ta sẽ hiện thực hệ thống đã thiết kế, đồng thời đánh giá kết quả thực hiện được

### 4.1. HIỆN THỰC

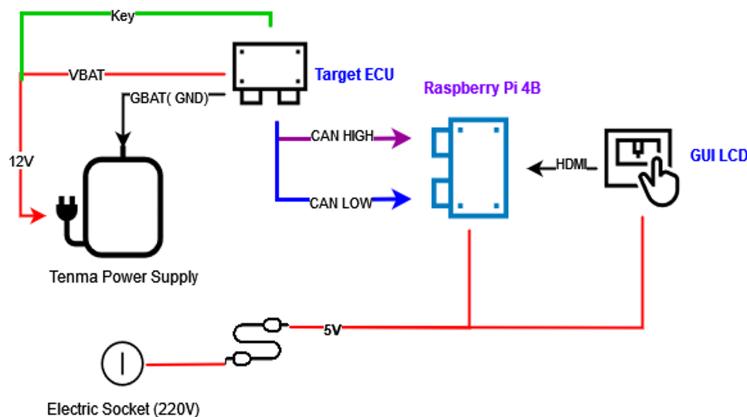
#### 4.1.1. MÔ HÌNH HỆ THỐNG

Dựa theo kiến trúc hệ thống đã được thiết kế ở các phần trước thì hệ thống trong thực tế sẽ gồm các phần cứng : Nguồn, target ECU, OTA Master, GUI LCD kết nối với nhau. Danh sách thiết bị trong hệ thống được mô tả trong bảng sau :

STT	Thiết bị	Thông tin thiết bị
1	Nguồn	Bech Power supply 72-2690 Teanma
2	Target ECU	Bosch ECU
3	GUI LCD	Waveshare 7 inch HDMI Touch screen LCD
4	OTA Master	Raspberry Pi 4B

Bảng 4.1: Danh sách thiết bị

Cách kết nối thiết bị trong hệ thống được mô tả bằng hình bên dưới:



4

Hình 4.1: Connection between devices

Hệ thống trong thực tế được mô tả như hình bên dưới:



Hình 4.2: Mô hình hệ thống

Để thuận tiện cho việc demo kết quả và sử dụng sau này, nhóm đã thiết kế một bộ KIT cho hệ thống trên.



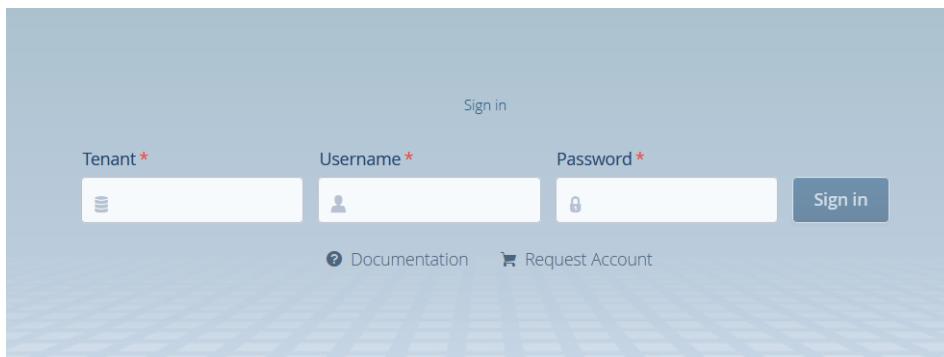
4



Hình 4.3: Mô hình KIT

#### 4.1.2. BOSCH IoT ROLLOUTS

Bosch IoT Rollouts là nơi để OEM quản lý thông tin phương tiện, quản lý phần mềm cập nhật và thực hiện các chiến dịch phần mềm như đã nói ở các phần trước. Đầu tiên OEM phải tiến hành đăng nhập vào Bosch IoT Rollouts để xác thực danh tính.



Hình 4.4: Giao diện đăng nhập

Các trang chức năng trong Bosch IoT Rollouts dành cho OEM thực hiện quy trình cập nhật phần mềm bao gồm :

- Deployment Management : sẽ quản lý tất cả các thiết bị xe, thông tin chi tiết của từng chiếc xe, chiến lược cập nhật, lịch sử cập nhật,...
- Upload: tải các gói phần mềm cần cập nhật lên và đóng gói thành các Software Module.
- Distribution: từ các Software Module sẽ đóng gói thành các Distribution.
- Target Filter: lọc ra các thiết bị cần cập nhật bằng các câu lệnh query.
- Rollout: gán target filter với distribution, dùng để cập nhật cho hàng loạt thiết bị.
- Statistics: thống kê việc cập nhật phần mềm, các thiết bị trên rollouts.

Giao diện các trang chức năng trong Bosch IoT Rollouts:

The screenshot shows the Deployment Management section of a software interface. On the left, a sidebar includes links for Deployment, Rollout, Target Filters, Distributions, Upload, Statistics, Users, System Config, Documentation, and Support, along with a timestamp (2023.04.25.1363). The main area has tabs for Targets and Distributions. Under Targets, there are two tables: 'Targets' (listing items like MG1CU023N1, Hyundai Santa Fe\_0, etc.) and 'Distributions' (listing items like testing, testing#2). A central panel displays details for a selected target ('Hyundai Santa Fe\_0') including Controller Id, Type, and Last poll. To the right is an 'Action history for Hyundai Santa Fe\_0' table with columns: Active, Distribution, Date, Status, Type, and Actions.

Hình 4.5: Trang Deployment Management

4

This screenshot shows the Upload section of the same software interface. The layout is identical to the Deployment Management screen, with a sidebar, Targets and Distributions tables, a central target details panel, and an Action history table on the right. The Action history table shows a series of actions for the 'Hyundai Santa Fe\_0' target, with the last entry being 'testing1' on May 12, 2023.

Hình 4.6: Trang Upload

The screenshot shows the 'Distributions Management' section of a software application. On the left, a sidebar lists navigation options: Deployment, Rollout, Target Filters, Distributions, Upload, Statistics, Users, System Config, Documentation, and Support. The date '2023.04.25.1303' is also shown.

**Distributions**

Name	Version	Delete
testing	1	trash
testing#2	1	trash

**Software Module**

Name	Version	Delete
testing	1	trash
testing#2	1	trash

**Distribution set: testing:1**

- Details Description Modules Tags Logs Metadata

Type: App(s) only  
Required Migration Step: No

**Software Module: testing:1**

- Details Description Logs Metadata

Vendor:  
Type: Application  
Assignment type: Software (SW)  
Artifact encryption: Disabled

Hình 4.7: Trang Distribution

The screenshot shows the 'Target Filter Management' section. The sidebar includes the same navigation options as the previous screenshot, along with 'Total Filtered Targets: 7'.

**Custom Filters > Hyundai**

**Query:** name==Hyundai

Controller ID	Name	Description	Status	Created By	Created Date	Modified By	Modified Date
HYUNDAI_SANTA_FE_0_0	Hyundai Santa Fe_0	1GTW7AFG7L1176090	●	52493b6e-ab5c-4917-9...	Mon May 8 12:23:08 ICT...	52493b6e-ab5c-4917-9...	Mon May 8 12:23:08 ICT...
HYUNDAI_SANTA_FE_0_0	Hyundai Santa Fe_0	1GTW7AFG7L1176090	●	52493b6e-ab5c-4917-9...	Mon May 8 12:22:56 ICT...	52493b6e-ab5c-4917-9...	Fri May 12 17:27:15 ICT...
HYUNDAI_SANTA_FE_1_0	Hyundai Santa Fe_1	1GTW7AFG7L1176090	●	52493b6e-ab5c-4917-9...	Mon May 8 12:23:15 ICT...	52493b6e-ab5c-4917-9...	Mon May 8 12:23:15 ICT...
HYUNDAI_SANTA_FE_1_1	Hyundai Santa Fe_1	1GTW7AFG7L1176090	●	52493b6e-ab5c-4917-9...	Mon May 8 12:23:22 ICT...	52493b6e-ab5c-4917-9...	Mon May 8 12:23:22 ICT...
HYUNDAI SONATA_0_1	Hyundai Sonata_0	1GTW7AFG7L1176090	●	52493b6e-ab5c-4917-9...	Mon May 8 12:23:36 ICT...	52493b6e-ab5c-4917-9...	Mon May 8 12:23:36 ICT...
HYUNDAI SONATA_0_2	Hyundai Sonata_0	1GTW7AFG7L1176090	●	52493b6e-ab5c-4917-9...	Mon May 8 12:23:43 ICT...	52493b6e-ab5c-4917-9...	Mon May 8 12:23:43 ICT...
HYUNDAI SONATA_0_0	Hyundai Sonata_0	1GTW7AFG7L1176090	●	52493b6e-ab5c-4917-9...	Mon May 8 12:23:29 ICT...	52493b6e-ab5c-4917-9...	Fri May 12 16:52:01 ICT...

Hình 4.8: Trang Target Filter

4

Create new Rollout

Name	<input type="text" value="Hyundai"/> *
Distribution set	<input type="text" value="testing:1"/> *
Custom Target Filter	<input type="text" value="Hyundai"/> *
Description	<input type="text" value="Description"/>
Action type	<input type="radio"/> ⚡ Forced <input checked="" type="radio"/> ⚡ Soft <input type="radio"/> ⚡ Download Only <input type="radio"/> ⚡ Time Forced
Start type	<input checked="" type="radio"/> ⏲ Manual <input type="radio"/> ➡ Auto <input type="radio"/> ⏱ Scheduled <span style="border: 1px solid #ccc; padding: 2px;">5/13/23 01:28 PM</span>
Number of Groups	<input type="text" value="Advanced Group definition"/>
Generate the groups automatically with the specified thresholds.	
Number of groups	<input type="text" value="2"/> * Targets per group:4
Trigger threshold	<input type="text" value="50"/> *
Error threshold	<input type="text" value="50"/> * <input checked="" type="radio"/> % <input type="radio"/> Count

\* Mandatory Field

Hình 4.9: Trang Rollout

The screenshot shows the Bosch Rollout Management interface. On the left is a sidebar with navigation links: Deployment, Rollout (selected), Target Filters, Distributions, Upload, Statistics, Users, and System Config. The main area is titled "Rollout Management" and contains a table titled "Rollouts". The table has columns: Name, Distribution set, Status, Detail status, Groups, Targets, Actions, and a search/filter icon. One row is visible, showing "Hyundai" as the name, "testing:1" as the distribution set, and "0" as the detail status. The "Groups" column shows "2", "Targets" shows "7", and the "Actions" column has several icons.

Hình 4.10: Trang Rollout - giám sát các thiết bị cập nhật



Hình 4.11: Trang Statistics

### 4.1.3. GIAO DIỆN NGƯỜI DÙNG: GUI

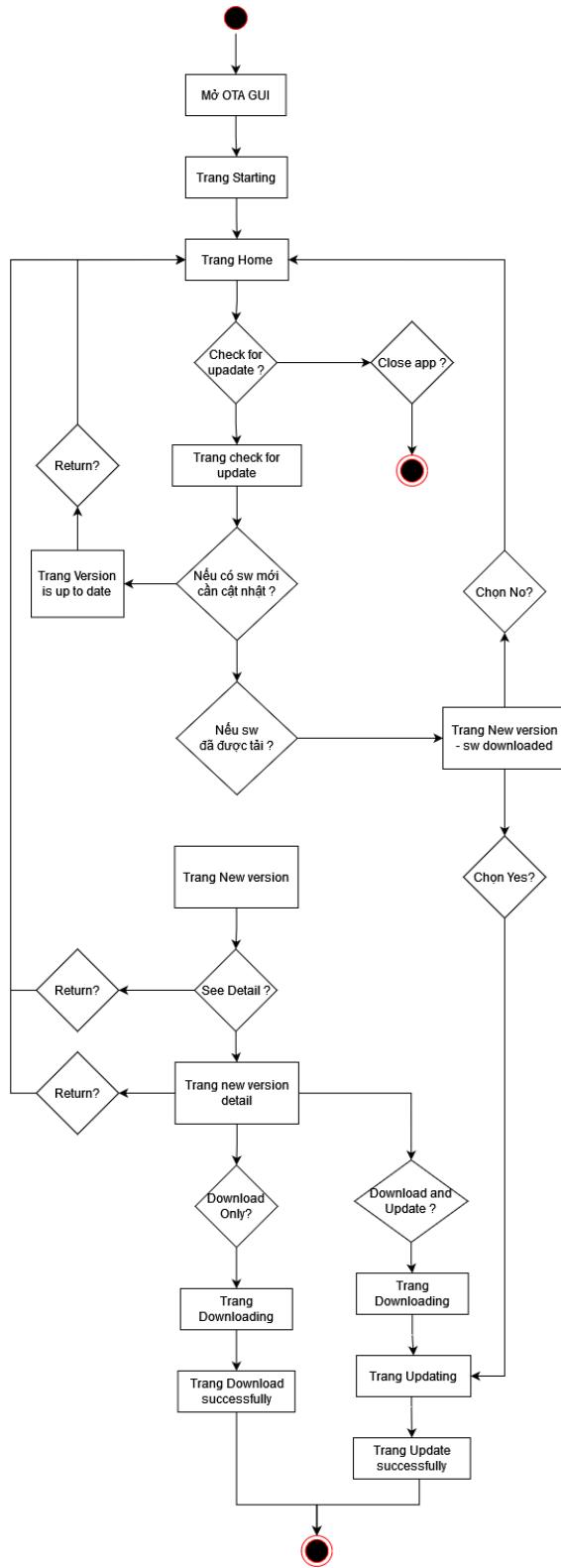
Để hiện thực giao diện tương tác với người dùng, nhóm đã sử dụng các ngôn ngữ và thư viện sau:

- Ngôn ngữ lập trình: Python
- Thư viện Python: tkinter, Pillow, requests, json, time, threading, PIL, time
  - Giao diện đồ họa: Tkinter (một thư viện phổ biến trong Python cho phát triển giao diện đồ họa). Customtkinter: một giao diện người dùng tùy chỉnh được tạo ra bằng Tkinter.
  - Thư viện Pillow được sử dụng để load và hiển thị hình ảnh gif (GIF image format)
  - Thư viện requests và json được sử dụng để gửi và nhận phản hồi từ REST API.
  - Thư viện threading được sử dụng để tạo luồng đồng thời (multi-threading).
  - Công nghệ: REST API (phương thức truyền tải dữ liệu độc lập)
  - PIL (Python Imaging Library): thư viện được sử dụng để xử lý hình ảnh trong Python.
  - thư viện time được sử dụng để delay ở một vài chỗ trong chương trình. Ví dụ, time.sleep(0.065) được sử dụng để tạo một khoảng thời gian trễ giữa các lần lặp của vòng lặp chạy animation gif để cho chuyển động mượt hơn.

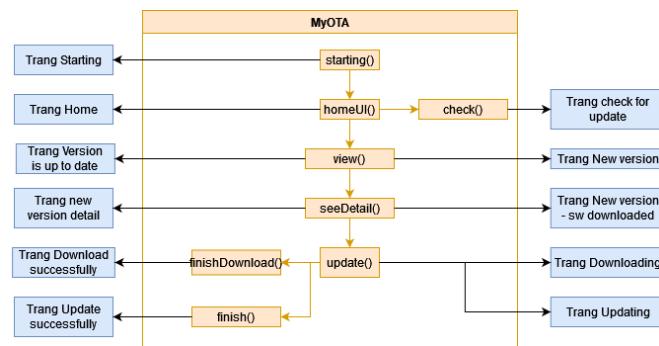
- Ngoài ra một hàm scheduler() được sử dụng để đưa tác vụ vào hàng đợi và lên lịch để thực thi trong tương lai, giúp tối ưu hiệu suất và tránh xảy ra sự cố trong quá trình thực thi.

Các chức năng mà GUI cung cấp có thể được tóm tắt bằng flowchart dưới đây:

4

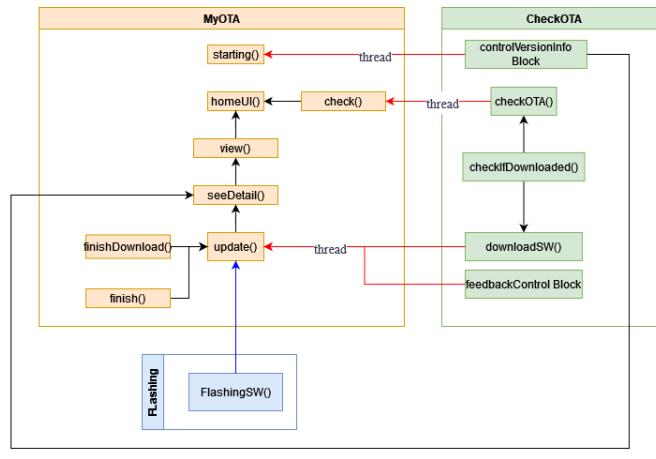


Sơ đồ khôi phục chức năng trong GUI:



4

Hình 4.13: *Display Pages with GUI functions*

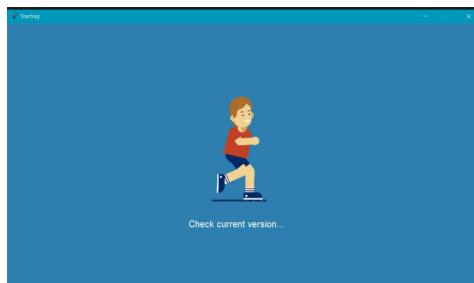


Note : : A call B

Hình 4.14: *GUI Functions Interaction*

Các trang giao diện người dùng trong thực tế:

- Trang Starting



4

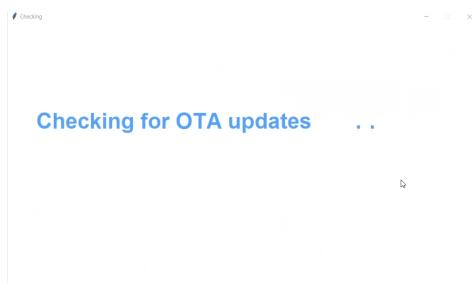
Hình 4.15: *Giao diện starting*

- Trang Home



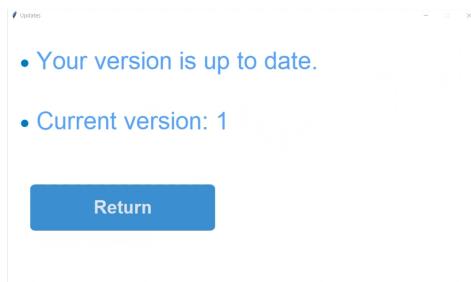
Hình 4.16: *Trang Home*

- Trang Check for Update



Hình 4.17: *Checking for OTA updates...*

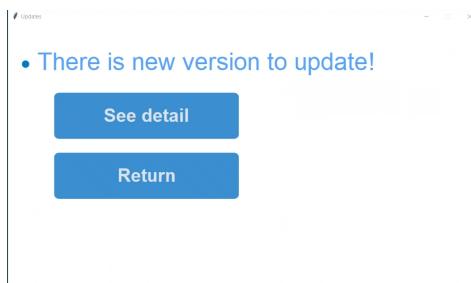
- Trang Version is up to date



Hình 4.18: *Version is up to date*

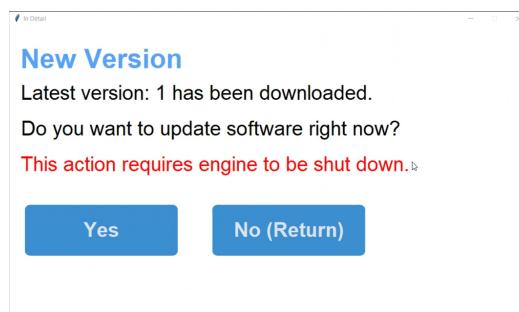
4

- Trang New version



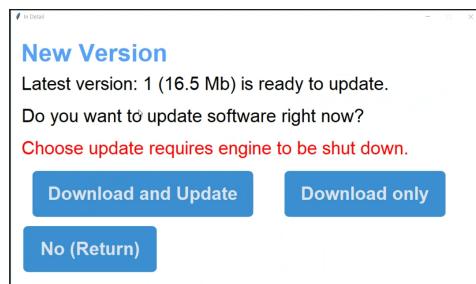
Hình 4.19: *New version needs to update*

- Trang New version software downloaded



Hình 4.20: *Software downloaded*

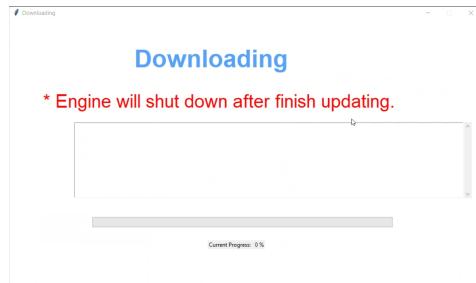
- Trang New version detail



Hình 4.21: *Software detail*

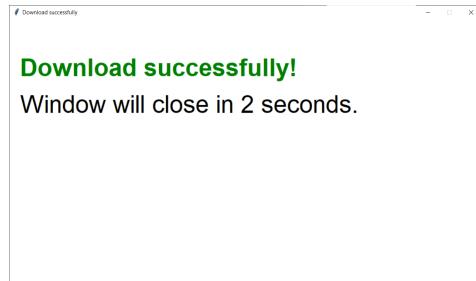
4

- Trang Downloading



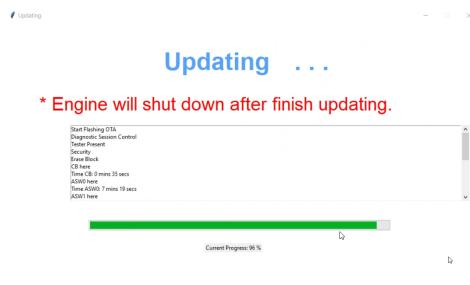
Hình 4.22: *Downloading...*

- Trang Download successfully



Hình 4.23: *Download successfully*

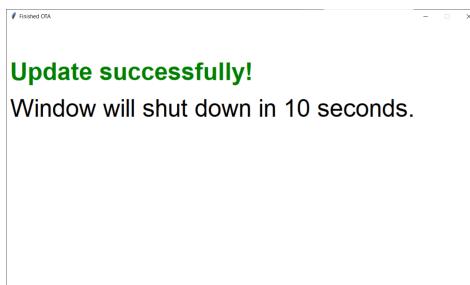
- Trang Updating



Hình 4.24: *Updating...*

4

- Trang Update Successfully



Hình 4.25: *Update successfully*

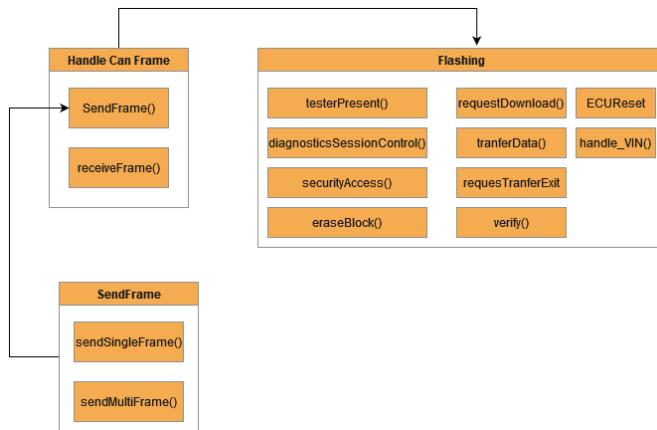
#### 4.1.4. QUÁ TRÌNH FLASHING CỦA RASPBERRY PI 4B

Để hiện thực quá trình flashing nhóm đã sử dụng các ngôn ngữ và thư viện sau:

- Ngôn ngữ lập trình: Python.
- Công Nghệ: Can Bus.
- Thư viện: Can.

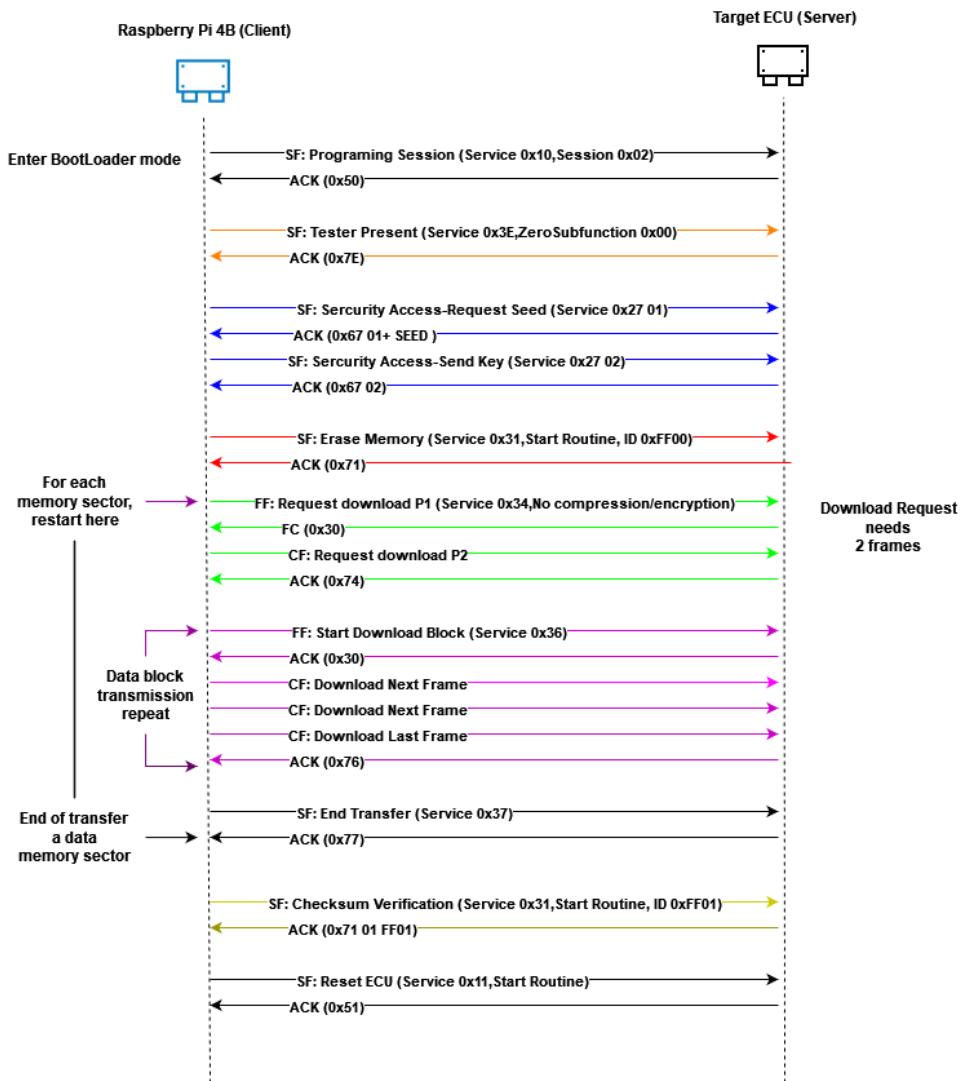
Sơ đồ Các hàm con trong hàm flashing

4



Hình 4.26: Sơ đồ các khối chức năng trong hàm flashing

Sau khi phần mềm mới được Raspberry Pi 4B tải về, việc flashing phần mềm này sẽ được Raspberry Pi 4B tải về từng bước một theo chu trình sau:



Hình 4.27: UDS Download Sequence Processes [15]

Quá trình flashing trong thực tế được ghi lại dưới đây:

```

1   ID      Data
2 0x7e0    02 10 02 00 00 00 00 00
3 0x7e8    06 50 02 00 32 01 ea aa
4
5
6
7
8 0x7e0    02 27 01 00 00 00 00 00
9 0x7e8    06 67 01 74 ff 8e ac 55
10 0x7e0   06 27 02 80 fd fe 00 55
11 0x7e8    02 67 02 55 55 55 55 55
12
13 0x7e0   04 31 01 ff 00 00 00 00
14 0x7e8    04 71 01 ff 00 aa aa aa
15
16 0x7e0   10 0b 34 00 44 80 01 c0
17 0x7e8    30 00 f3 aa aa aa aa aa
18
19 0x7e0   21 00 00 01 c0 00 00 00
20 0x7e8    04 74 20 0f ff aa aa aa
21
22 0x7e0   1f e2 36 01 ef be ad de
23 0x7e8    30 00 f3 aa aa aa aa aa
24
25 0x7e0   21 00 00 00 00 00 00 00
26 0x7e0   22 00 00 00 00 00 00 00
27 0x7e0   23 00 00 00 00 00 00 00
28 0x7e0   24 00 00 00 00 00 00 00
29 0x7e0   25 50 00 02 c0 52 03
30 0x7e0   26 80 00 80 42 80 68 c1
31 0x7e0   27 01 80 01 01 06 01 fa
32 0x7e0   28 fa fa fa fa fa fa
33 0x7e0   29 fa fa fa fa 00 00 00
34 0x7e0   2a 00 00 00 00 00 00 00
35 0x7e0   2b 00 00 00 00 00 00 00
36 0x7e0   2c 00 00 00 00 00 00 00
37 0x7e0   2d 00 00 00 00 00 00 00
38 0x7e0   2e 00 c8 52 03 80 30 30
39 0x7e0   2f 43 42 4e 4f 54 53 45
40 0x7e0   20 54 04 00 00 c1 01 80
41 0x7e0   21 d7 57 03 80 09 00 00
42 0x7e0   22 02 d8 57 03 80 4c a4
43 0x7e0   23 84 e9 b3 5b 7b 16 8a
44 0x7e0   24 00 04 00 05 00 00 00

```

Programming session (Bootloader Mode)

Security Access - Request Seed

Security Access - Send Key

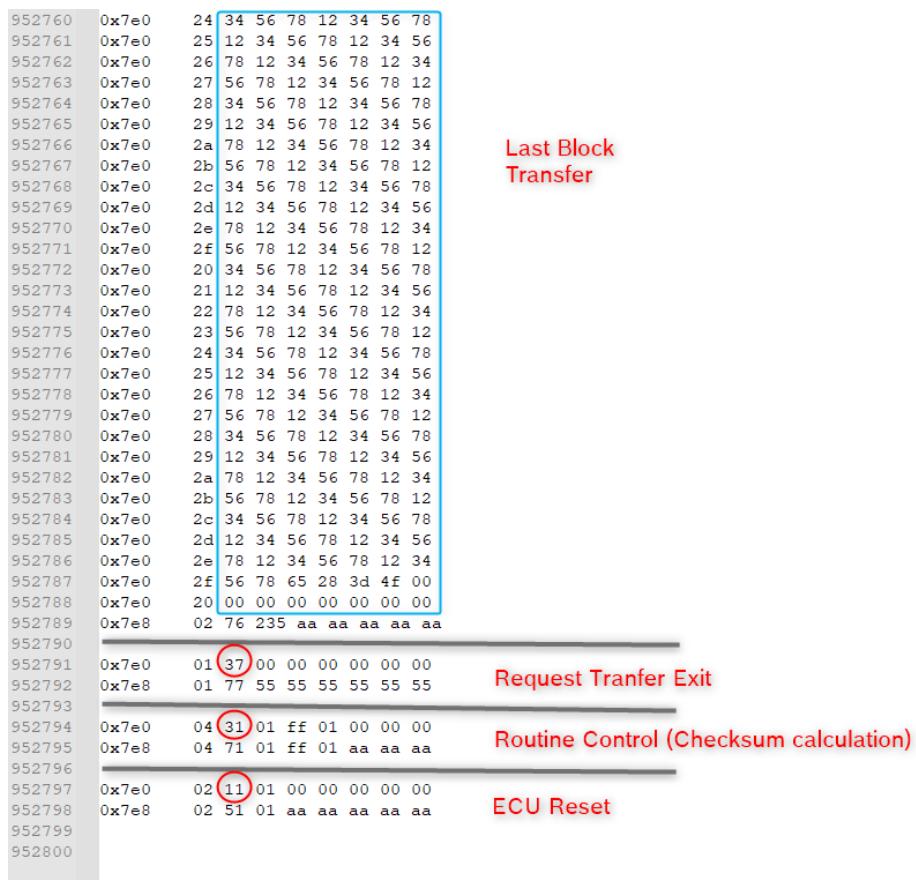
Routine Control (Erase)

Request Download ( Need 2 frame)

Transfer Block 1

4

Hình 4.28: Update software Log in reality part 1



The screenshot shows a software log window with memory dump and command history. The memory dump is a grid of hex values. A blue box highlights the last few lines of the dump, labeled 'Last Block Transfer'. Below the dump, several command entries are shown, each with a red circle around the first byte of the command code:

- 0x7e0 01 37 00 00 00 00 00 00 Request Tranfer Exit
- 0x7e0 04 31 01 ff 01 00 00 00 Routine Control (Checksum calculation)
- 0x7e0 02 11 01 00 00 00 00 00 ECU Reset

A large red number '4' is in the top right corner of the screenshot.

Hình 4.29: Update software Log in reality part 2

## 4.2. KẾT QUẢ

Sau quá trình hiện thực chúng em đã hoàn thành được các tính năng chính đã đề ra bao gồm:

- Các chức năng quản lí trên Cloud: Tải phần mềm lên Bosch IoT Rollouts, quản lí phần mềm và thiết bị
- Các chức năng tương tác với người dùng GUI:
  - Kiểm tra xe có phần mềm cập nhật hay không
  - Tải phần mềm từ Bosch IoT Rollouts về Raspberry Pi 4B của phương tiện
  - Phần mềm của target ECU được cập nhật thông qua Raspberry Pi 4B

Ngoài ra các luồng thực thi phụ cũng được hiện thực và có các kết quả như hình ảnh như trong phần Hiện thực.

Sau khi hoàn thiện hệ thống chúng em đã tiến hành thử nghiệm.

### Mô tả thí nghiệm

- Mục đích thí nghiệm: Kiểm tra toàn bộ hệ thống "Software Over-the-air update for Passengers car".
- Các giả định:

**4**

- Do không có các thiết bị để mô phỏng quá trình xe đang chạy, nên giả định khi tiến hành cập nhật hệ thống xe đang ở trạng thái an toàn.
- Do mỗi thiết bị trong chiến dịch cập nhật sẽ tự truy cập vào Bosch IoT Rollouts và tải phần mềm về một cách độc lập, nên trong thí nghiệm sẽ chỉ tiến hành cập nhật cho một thiết bị (1 Target ECU).
- Giả định tốc độ Internet là bình thường tốc độ tải xuống khoảng 50Mbps và tốc độ tải lên trung bình là 50Mbps. Các điều kiện về nguồn điện trong xe được đảm bảo.
- Thiết bị: Bộ OTA KIT thí nghiệm và một LED 5V, phần mềm cập nhật ( hex, bin )
- Các bước tiến hành thí nghiệm:
  - Bước 1: Thêm device cho Bosch IoT Rollouts
  - Bước 2: Thêm phần mềm và tạo chiến dịch cập nhật cho 1 thiết bị
  - Bước 3: Mở MyOTA GUI trên xe lên
  - Bước 4: Chọn "Check for updates" trên màn hình
  - Bước 5: Chọn "See detail"
  - Bước 6: Chọn "Yes" ở cửa sổ New Version
  - Bước 7: Chọn "Download only"
  - Bước 8: Chọn "Yes" ở cửa sổ New version software downloaded
  - Bước 9: Chờ đến khi quá trình cập nhật hoàn thành
- Kết quả:
  - Phần mềm được cập nhật thành công, đèn LED nhấp nháy chứng minh phần mềm mới đã cập nhật thành công.

- Tốc độ của quá trình flashing được tóm tắt trong bảng sau :

	Dung lượng	Số lần	Thời gian Download	Thời gian Flashing
Trường hợp 1	16 KB	10	5s	34m
Trường hợp 2	2 KB	10	2s	5m40s

Hình 4.30: Result Table

4

### Video demo:

<https://drive.google.com/file/d/1mSwIrREKltIWmdoultx35bDyyeUjK3oq/view?usp=sharing>

## 4.3. ĐÁNH GIÁ HỆ THỐNG

### 1. Tính ổn định và đáng tin cậy

Hệ thống OTA cần đảm bảo tính ổn định và đáng tin cậy trong việc cập nhật phần mềm. Khi tiến hành kiểm thử, nhóm đã tiến hành flash 10 lần liên tiếp và hệ thống vẫn có thể tiến hành flashing. Thời gian flashing giữa các lần xấp xỉ nhau cho thấy tính ổn định của hệ thống.

### 2. Tốc độ truyền dữ liệu

Có một số yếu tố ảnh hưởng đến tốc độ flashing của ECU trong ứng dụng OTA. Dưới đây là một số yếu tố quan trọng:

- Kích thước và số lượng dữ liệu cần flash

Kích thước và số lượng dữ liệu cần flash vào ECU sẽ ảnh hưởng đến thời gian cần thiết để hoàn tất quá trình flashing. Khi kích thước dữ liệu lớn, thời gian flashing sẽ tăng lên. Khi kích thước file cập nhật mới giảm 8 lần ( 16KB -> 2 KB) tốc độ flashing cũng tăng lên xấp xỉ 6 lần. (34 phút -> 5 phút 40 giây).

- Dung lượng của Tx Buffer của CAN HAT gắn trên Raspberry có ảnh hưởng đến quá trình flash khi tiến hành truyền hàng loạt dữ liệu lên đường CAN bus trong giai đoạn Transfer Data của quá trình flashing.

### 3. Bảo mật

Toàn bộ quá trình hoạt động được đảm bảo bằng 3 giai đoạn bảo mật: Authoring Security Processes, Backend Security concept, In-Vehicle Security concept.

**4. Quản lý lỗi**

Hệ thống OTA cần có khả năng quản lý lỗi và khôi phục sau lỗi trong quá trình cập nhật. Nếu xảy ra lỗi trong quá trình cập nhật OTA, hệ thống sẽ đưa ra thông báo và yêu cầu người dùng tiến hành cập nhật lại phần mềm.

**5. Tính linh hoạt**

Hệ thống OTA nên linh hoạt để cho phép cập nhật phần mềm cho các ECU mục tiêu khác, KIT OTA có tính linh hoạt giúp dễ dàng thay thế Target ECU một cách nhanh chóng, dễ dàng di chuyển và tiện dụng.

# 5

## KẾT LUẬN

Sau khi tiến hành nghiên cứu và hiện thực thành công toàn bộ quá trình cập nhật phần mềm over-the-air, nhóm đã nắm rõ các luồng hoạt động trên của ECU, mạng kết nối ECU trên xe trong thực tế, đồng thời hiểu được quy trình cập nhật phần mềm trong xe bằng cách sử dụng chuỗi các lệnh dịch vụ UDS.

Để hoàn thành đồ án lần này, đòi hỏi nhóm vận dụng kiến thức của tất cả các môn học trong ngành kỹ thuật máy tính như hệ thống nhúng, hệ thống số, hệ điều hành, quản lý dự án,... giúp nhóm tổng hợp lại các kiến thức đã được học và vận dụng thành công vào thực tế.

Sau khi đánh giá lại hệ thống, nhóm đề xuất một số hướng phát triển trong tương lai giúp hệ thống ngày càng hoàn thiện và mang lại nhiều lợi ích hơn như:

- Tăng khả năng tương tác với người dùng:
  - Tạo thêm một Mobile APP giúp người dùng dễ dàng thực hiện cập nhật phần mềm ngay cả khi không ngồi trên xe.
  - Cải thiện giao diện GUI trên xe, thêm các tính năng thông báo và cảnh báo khi có sự cố cập nhật.
- Phát triển thêm các công nghệ mới: Nếu trong tương lai có thể sử dụng ECU có bộ nhớ hỗ trợ cho ứng dụng OTA thì có thể phát triển thêm các tính năng Rollbacks như ở chiến lược Base+, Enhanced, Preminum.
- Tích hợp với phần mềm của ECU: Hiện tại không thể tích hợp ứng dụng OTA như một phần của Application software trên ECU vì muốn làm thế

cần liên quan đến nhiều bên, nhưng trong tương lai nếu có điều kiện thì có thể thực hiện.

- Việc tạo ra KIT tiện dụng, trong tương lai có thể thay đổi Target ECU một cách linh hoạt nhờ vậy bộ KIT này sẽ hỗ trợ việc kiểm thử chức năng cập nhật phần mềm cho nhiều loại ECU, hỗ trợ kiểm thử các chức năng của Bosch IoT Rollouts.

Nhóm hy vọng rằng các đóng góp của đồ án này sẽ góp phần đưa công nghệ update OTA phát triển hơn nữa và đem lại lợi ích thực tiễn cho người dùng. Đồng thời những kiến thức và kinh nghiệm trong quá trình hiện thực hệ thống cập nhật phần mềm over-the-air đã trở thành hành trang quý báu cho nhóm trong tương lai.

# TÀI LIỆU THAM KHẢO

- [1] Can bus explained. In <https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial>.
- [2] Can tp (iso 15765 - 2). In <https://www.linkedin.com/pulse/can-tp-iso-15765-2-vivek-maurya>.
- [3] The different versions of the raspberry pi. In <https://pimylifeup.com/raspberry-pi-versions/>.
- [4] Ecu explained. In <https://www.ecutesting.com/categories/ecu-explained/>.
- [5] Ecu reset service identifier (0x11): Uds protocol. In <https://piembstech.com/ecu-reset-service-identifier-0x11-uds-protocol/>.
- [6] The evolution of gateway processors in the auto market. In <https://www.electronicspecifier.com/products/artificial-intelligence/the-evolution-of-gateway-processors-in-the-auto-market>.
- [7] A modern car runs on 100 million lines of code — but who will write them in the future? In <https://medium.com/next-level-german-engineering/porsche-future-of-code-526eb3de3bbe>.
- [8] Nội dung tham khảo thuộc về tài liệu của bosch.
- [9] Request download (0x34) service: Uds protocol. In <https://piembstech.com/request-download-0x34-service-uds-protocol/>.
- [10] Toyota to recall 6.4 million vehicles. In <https://www.nytimes.com/2014/04/10/business/international/toyota-to-recall-vehicles.html>.
- [11] Volkswagen stock drops 20% on us diesel recall probe. In <https://www.cnbc.com/2015/09/21/volkswagen-stock-drops-20-on-us-diesel-recall-probe.html>.
- [12] What is a raspberry pi? In <https://opensource.com/resources/raspberry-pi>.

- [13] What is central gateway module and its significance in automotive world. In <https://www.einfochips.com/blog/what-is-central-gateway-module-and-its-significance-in-automotive-world/>.
- [14] Haefner Christian. Fbl-generic information. Docupedia, 2022. Tài liệu lưu hành nội bộ của RBVH.
- [15] Silva Milton. Uds application layer. Docupedia, 2023. Tài liệu lưu hành nội bộ của RBVH.
- [16] Sandeep Sathyanarayana. Ota strategy. In *Future Portfolio view - India*. Docupedia, 2022. Tài liệu lưu hành nội bộ của RBVH.
- [17] Kriston Tibor. Overtheair flashing. In *FOTA*. Docupedia, 2020. Tài liệu lưu hành nội bộ của RBVH.