

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HCM  
Khoa Khoa học và Kỹ thuật Máy tính

---

HỆ THỐNG MÁY BÁN HÀNG TỰ  
ĐỘNG  
Báo cáo Kỹ thuật

Đồ án Thiết kế Luận lý với Vi điều khiển STM32

---

Nhóm tác giả:  
Nguyễn Hưng Thịnh  
Lê Thế Lộc  
Trần Doãn Hoàng Lâm

Đơn vị:  
Trường Đại học Bách Khoa TP.HCM (HCMUT)

Ngày 20 tháng 12 năm 2025

### Tóm tắt nội dung

Báo cáo kỹ thuật này trình bày quá trình thiết kế và hiện thực hệ thống máy bán hàng tự động thông minh sử dụng vi điều khiển STM32F103C8T6. Hệ thống được thiết kế đặc biệt để bán các vật phẩm trong trò chơi (Trang phục đội tuyển SKT trong Liên Minh Huyền Thoại) và có các tính năng như tự động phát hiện khách hàng thông qua cảm biến siêu âm, giao diện LCD tương tác với bàn phím, quản lý kho hàng không bay hơi sử dụng bộ nhớ Flash, xử lý thanh toán toàn diện và chế độ quản trị bảo mật để kiểm soát kho hàng. Dự án minh họa các ứng dụng thực tế của nguyên lý hệ thống nhúng bao gồm thiết kế máy trạng thái hữu hạn, giao tiếp ngoại vi, điều khiển thời gian thực và lưu trữ dữ liệu. Việc hiện thực sử dụng thư viện STM32 HAL và bao gồm các trình điều khiển tùy chỉnh cho giao tiếp LCD I2C, quét bàn phím ma trận 4x4 và đo khoảng cách siêu âm HC-SR04. Các tính năng chính bao gồm tự động bật nguồn thông qua phát hiện khách hàng, xử lý giao dịch đa trạng thái với xử lý lỗi và lưu trữ kho hàng dựa trên bộ nhớ Flash qua các chu kỳ nguồn.

# Mục lục

<b>1 Giới thiệu</b>	<b>7</b>
1.1 Bối cảnh và Động lực . . . . .	7
1.2 Mục tiêu Dự án . . . . .	7
1.3 Phạm vi và Giới hạn . . . . .	8
1.3.1 Phạm vi Dự án . . . . .	8
1.3.2 Giới hạn Hệ thống . . . . .	8
1.4 Cấu trúc Báo cáo . . . . .	8
<b>2 Tổng quan Hệ thống</b>	<b>10</b>
2.1 Kiến trúc Hệ thống . . . . .	10
2.1.1 Lớp Phần cứng . . . . .	10
2.1.2 Lớp Trình điều khiển . . . . .	11
2.1.3 Lớp Ứng dụng . . . . .	11
2.1.4 Lớp Giao diện Người dùng . . . . .	11
2.2 Triết lý Thiết kế . . . . .	11
2.2.1 Tính Mô-đun và Phân tách Mối quan tâm . . . . .	12
2.2.2 Kiến trúc Hướng Trạng thái . . . . .	12
2.2.3 Vận hành An toàn . . . . .	12
2.2.4 Thiết kế Lấy Người dùng làm Trung tâm . . . . .	12
2.3 Yêu cầu Hệ thống . . . . .	13
2.3.1 Yêu cầu Chức năng . . . . .	13
2.3.2 Yêu cầu Phi chức năng . . . . .	13
2.4 Luồng Vận hành Hệ thống . . . . .	14
<b>3 Thành phần Phần cứng</b>	<b>15</b>
3.1 Vi điều khiển - STM32F103C8T6 . . . . .	15
3.1.1 Thông số Kỹ thuật . . . . .	15
3.1.2 Lý do Lựa chọn . . . . .	16
3.1.3 Cấu hình Chân . . . . .	16
3.2 Mô-đun Hiển thị - LCD 16x2 với I2C . . . . .	16
3.2.1 Thông số LCD . . . . .	16
3.2.2 Bộ chuyển đổi I2C (PCF8574) . . . . .	17
3.2.3 Giao thức Giao tiếp . . . . .	17
3.3 Thiết bị Đầu vào - Bàn phím Ma trận 4×4 . . . . .	17
3.3.1 Bố cục Bàn phím . . . . .	17
3.3.2 Cơ chế Quét . . . . .	18
3.3.3 Chiến lược Chống rung . . . . .	18
3.4 Mô-đun Cảm biến - Cảm biến Siêu âm HC-SR04 . . . . .	18
3.4.1 Thông số Cảm biến . . . . .	18
3.4.2 Nguyên lý Hoạt động . . . . .	19
3.4.3 Cấu hình Bắt đầu vào Bộ định thời . . . . .	19
3.5 Bộ nhớ - Flash Nội bộ . . . . .	19
3.5.1 Tổ chức Bộ nhớ Flash . . . . .	19
3.5.2 Quy trình Lập trình Flash . . . . .	20
3.5.3 Chiến lược Bảo toàn Dữ liệu . . . . .	20
3.6 Nguồn điện và Giao diện Lập trình . . . . .	20
3.6.1 Yêu cầu Nguồn điện . . . . .	20

3.6.2    Bộ nạp ST-Link V2 . . . . .	21
<b>4   Thiết kế Phần mềm</b>	<b>22</b>
4.1    Kiến trúc Máy trạng thái Hữu hạn . . . . .	22
4.1.1    Định nghĩa Trạng thái . . . . .	22
4.1.2    Phân loại Trạng thái . . . . .	22
4.1.3    Sơ đồ Chuyển đổi Trạng thái . . . . .	23
4.1.4    Hiện thực FSM . . . . .	25
4.2    Hệ thống Quản lý Bộ định thời . . . . .	26
4.2.1    Các loại Bộ định thời . . . . .	26
4.2.2    Hiện thực Bộ định thời . . . . .	26
4.2.3    Thực thi Bộ định thời . . . . .	27
4.2.4    Mẫu Sử dụng Bộ định thời . . . . .	27
4.3    Cấu trúc Dữ liệu và Quản lý Bộ nhớ . . . . .	27
4.3.1    Cấu trúc Dữ liệu Sản phẩm . . . . .	27
4.3.2    Bố cục Bộ nhớ Flash . . . . .	28
4.3.3    Biến Toàn cục . . . . .	28
4.4    Các Thuật toán Chính . . . . .	28
4.4.1    Thuật toán Phát hiện Khách hàng . . . . .	28
4.4.2    Thuật toán Xác thực Thanh toán . . . . .	29
4.4.3    Thuật toán Lưu trữ Kho hàng . . . . .	30
4.4.4    Thuật toán Xác thực Quản trị . . . . .	31
4.5    Tổ chức Mã nguồn và Tính Mô-đun . . . . .	32
4.5.1    Cấu trúc Mô-đun . . . . .	32
4.5.2    Phụ thuộc Mô-đun . . . . .	32
<b>5   Hiện thực</b>	<b>34</b>
5.1    Hiện thực Trình điều khiển Ngoại vi . . . . .	34
5.1.1    Trình điều khiển LCD I2C . . . . .	34
5.1.2    Trình điều khiển Bàn phím . . . . .	36
5.1.3    Trình điều khiển Cảm biến Siêu âm . . . . .	37
5.2    Chi tiết Hiện thực Máy trạng thái . . . . .	38
5.2.1    Hiện thực các Trạng thái Quan trọng . . . . .	38
5.2.2    Admin Mode Implementation . . . . .	40
5.3    Display Management . . . . .	41
5.3.1    Text Formatting Functions . . . . .	41
5.4    Error Handling Mechanisms . . . . .	42
5.4.1    Input Validation . . . . .	42
5.4.2    Error Counter Management . . . . .	43
5.5    Performance Optimizations . . . . .	43
5.5.1    Reduced Display Updates . . . . .	43
5.5.2    Timer Optimization . . . . .	44
5.5.3    Flash Write Minimization . . . . .	44
5.6    Development and Debugging . . . . .	44
5.6.1    Debug Output . . . . .	44
5.6.2    State Monitoring . . . . .	44
<b>6   Trình diễn</b>	<b>46</b>
6.1    Phương pháp Kiểm thử . . . . .	46

6.1.1	Kiểm thử Đơn vị . . . . .	46
6.1.2	Kiểm thử Tích hợp . . . . .	46
6.1.3	Kiểm thử Sức chịu đựng . . . . .	46
6.2	Kết quả Vận hành . . . . .	47
6.2.1	Hiệu suất Phát hiện Khách hàng . . . . .	47
6.2.2	Đo lường Thời gian Phản hồi . . . . .	47
6.2.3	Tỷ lệ Giao dịch Thành công . . . . .	47
6.3	Các Kịch bản Vận hành Ví dụ . . . . .	48
6.3.1	Kịch bản 1: Mua hàng Thành công . . . . .	48
6.3.2	Kịch bản 2: Xử lý Hết hàng . . . . .	49
6.3.3	Kịch bản 3: Điều chỉnh Kho hàng Quản trị . . . . .	50
6.4	Phân tích Hiệu năng . . . . .	50
6.4.1	Sử dụng Bộ nhớ . . . . .	50
6.4.2	Tiêu thụ Năng lượng . . . . .	51
6.4.3	Độ bền Flash . . . . .	51
6.5	Các Vấn đề Gặp phải và Giải pháp . . . . .	51
6.5.1	Sự không ổn định Giao tiếp LCD . . . . .	51
6.5.2	Bóng ma Bàn phím . . . . .	52
6.5.3	Nhiều Cảm biến ở Phạm vi Gần . . . . .	52
6.5.4	Xử lý Tràn Bộ định thời . . . . .	52
6.6	Dộ tin cậy Hệ thống . . . . .	52
6.6.1	Kiểm thử Vận hành Liên tục . . . . .	52
6.6.2	Xác thực Phục hồi Lỗi . . . . .	53
<b>7</b>	<b>Đề xuất cho Công việc Tương lai</b>	<b>54</b>
7.1	Cải tiến Phần cứng . . . . .	54
7.1.1	Cơ chế Nhả Sản phẩm Vật lý . . . . .	54
7.1.2	Phần cứng Thanh toán Thực . . . . .	54
7.1.3	Màn hình Lớn hơn . . . . .	54
7.1.4	Mở rộng Dung lượng Kho hàng . . . . .	55
7.2	Cải tiến Phần mềm . . . . .	55
7.2.1	Kết nối Mạng . . . . .	55
7.2.2	Phân tích Bán hàng Nâng cao . . . . .	55
7.2.3	Hỗ trợ Da ngôn ngữ . . . . .	56
7.2.4	Hỗ trợ Người dùng và Chương trình Khách hàng Thân thiết . . . . .	56
7.2.5	Chẩn đoán Lỗi Nâng cao . . . . .	56
7.3	Cải thiện Trải nghiệm Người dùng . . . . .	56
7.3.1	Phản hồi Âm thanh . . . . .	56
7.3.2	Tính năng Hỗ trợ Truy cập . . . . .	57
7.3.3	In Hóa đơn . . . . .	57
7.4	Cải tiến Bảo mật . . . . .	57
7.4.1	Bảo mật Quản trị Nâng cao . . . . .	57
7.4.2	Bảo vệ Chống Giả mạo . . . . .	57
7.5	Quản lý Năng lượng . . . . .	58
7.5.1	Hiệu quả Năng lượng . . . . .	58
7.5.2	Giám sát Môi trường . . . . .	58
7.6	Các Lĩnh vực Sản phẩm Thay thế . . . . .	58
7.7	Cân nhắc về Khả năng Mở rộng . . . . .	58

7.7.1	Nâng cấp Vi điều khiển . . . . .	58
7.7.2	Kiến trúc Mô-đun . . . . .	59
<b>8</b>	<b>Kết luận</b>	<b>60</b>
8.1	Thành tựu Dự án . . . . .	60
8.1.1	Thành tựu Kỹ thuật . . . . .	60
8.1.2	Chỉ số Hiệu suất . . . . .	60
8.1.3	Chất lượng Phần mềm . . . . .	61
8.2	Giá trị Giáo dục . . . . .	61
8.2.1	Khái niệm Hệ thống Nhúng . . . . .	61
8.2.2	Giao tiếp Phần cứng . . . . .	61
8.2.3	Thiết kế Phần mềm . . . . .	61
8.2.4	Tích hợp Hệ thống . . . . .	62
8.3	Thách thức Đã Vượt qua . . . . .	62
8.4	Ứng dụng Thực tế . . . . .	62
8.5	Hạn chế Được Thừa nhận . . . . .	63
8.6	Bài học Kinh nghiệm . . . . .	63
8.6.1	Thông tin Kỹ thuật . . . . .	63
8.6.2	Quy trình Phát triển . . . . .	63
8.7	Tác động Dự án . . . . .	64
8.8	Lời kết . . . . .	64
<b>9</b>	<b>Tài liệu Tham khảo &amp; Tác giả</b>	<b>65</b>
9.1	Tài liệu Tham khảo Kỹ thuật . . . . .	65
9.1.1	Tài liệu Vi điều khiển . . . . .	65
9.1.2	Công cụ Phát triển . . . . .	65
9.1.3	Thành phần Phần cứng . . . . .	65
9.1.4	Giao thức Giao tiếp . . . . .	65
9.1.5	Kỹ thuật Phần mềm . . . . .	65
9.1.6	Lập trình C Nhúng . . . . .	66
9.1.7	Tài nguyên Trực tuyến . . . . .	66
9.2	Kho lưu trữ Dự án . . . . .	66
9.3	Tác giả . . . . .	66
9.3.1	Nhóm Dự án . . . . .	66
9.3.2	Tổ chức . . . . .	67
9.3.3	Thông tin Khóa học . . . . .	67
9.4	Lời cảm ơn . . . . .	67
9.5	Giấy phép và Sử dụng . . . . .	67
9.6	Thông tin Liên hệ . . . . .	68

# 1 Giới thiệu

## 1.1 Bối cảnh và Động lực

Máy bán hàng tự động đại diện cho một ứng dụng quan trọng của hệ thống nhúng trong tự động hóa bán lẻ hiện đại. Các hệ thống này đòi hỏi sự tích hợp phần cứng-phần mềm đáng tin cậy, phản hồi thời gian thực với đầu vào của người dùng và cơ chế xử lý lỗi mạnh mẽ. Thị trường máy bán hàng tự động toàn cầu tiếp tục mở rộng, với nhu cầu ngày càng tăng về các giải pháp bán lẻ tự động thông minh, thân thiện với người dùng có thể hoạt động tự chủ với sự can thiệp tối thiểu của con người.

Dự án này hiện thực một hệ thống máy bán hàng tự động toàn diện sử dụng vi điều khiển STM32F103C8T6, hướng đến thị trường vật phẩm game bằng cách bán các trang phục đội tuyển SKT trong Liên Minh Huyền Thoại. Việc lựa chọn lĩnh vực sản phẩm cụ thể này cho phép minh họa các nguyên tắc quản lý kho hàng trong khi vẫn duy trì sự hấp dẫn đối với cộng đồng game thủ. Hệ thống giải quyết một số thách thức chính trong thiết kế hệ thống nhúng bao gồm quản lý trạng thái, giao tiếp ngoại vi, lưu trữ dữ liệu và tối ưu hóa trải nghiệm người dùng.

## 1.2 Mục tiêu Dự án

Các mục tiêu chính của dự án này là:

- Tích hợp Phần cứng:** Thiết kế và hiện thực một hệ thống nhúng hoàn chỉnh tích hợp nhiều thiết bị ngoại vi bao gồm màn hình LCD, bàn phím ma trận, cảm biến siêu âm và bộ nhớ Flash để lưu trữ dữ liệu.
- Hiện thực Máy trạng thái:** Phát triển kiến trúc máy trạng thái hữu hạn mạnh mẽ có khả năng quản lý các luồng giao dịch phức tạp, điều kiện lỗi và chuyển đổi chế độ với 19 trạng thái riêng biệt.
- Trải nghiệm Người dùng:** Tạo giao diện trực quan cho phép khách hàng duyệt sản phẩm, chọn số lượng, xử lý thanh toán và nhận phản hồi thông qua các thông báo LCD rõ ràng và xử lý đầu vào nhanh nhạy.
- Lưu trữ Dữ liệu:** Hiện thực lưu trữ không bay hơi sử dụng bộ nhớ Flash nội bộ để bảo toàn dữ liệu kho hàng qua các chu kỳ nguồn, đảm bảo tính liên tục của hoạt động kinh doanh.
- Bảo mật và Quản trị:** Cung cấp quyền truy cập quản trị được bảo vệ bằng mật khẩu để quản lý kho hàng với bảo mật dựa trên thời gian chờ và cơ chế xác nhận.
- Vận hành Tự động:** Cho phép tự động phát hiện khách hàng và kích hoạt hệ thống sử dụng cảm biến siêu âm, giảm tiêu thụ điện năng trong thời gian nhàn rỗi.
- Khả năng Chống lỗi:** Hiện thực xử lý lỗi toàn diện bao gồm kiểm tra đầu vào, quản lý thời gian chờ và cơ chế phục hồi để đảm bảo độ tin cậy của hệ thống.

## 1.3 Phạm vi và Giới hạn

### 1.3.1 Phạm vi Dự án

Dự án này bao gồm các thành phần và tính năng sau:

- Firmware nhúng hoàn chỉnh cho vi điều khiển STM32F103C8T6
- Trình điều khiển ngoại vi tùy chỉnh cho LCD, bàn phím và cảm biến siêu âm
- Máy trạng thái hữu hạn với 19 trạng thái để xử lý giao dịch
- Quản lý bộ nhớ Flash để lưu trữ kho hàng
- Chế độ quản trị với xác thực mật khẩu
- Xử lý lỗi và quản lý thời gian chờ
- Phát hiện khách hàng và tự động kích hoạt hệ thống
- Xác thực thanh toán với các mệnh giá tiền tệ Việt Nam

### 1.3.2 Giới hạn Hệ thống

Việc hiện thực hiện tại có các ràng buộc sau:

- **Dung lượng Kho hàng:** Giới hạn ở 16 sản phẩm với số lượng tối đa 9 đơn vị mỗi mặt hàng
- **Mô phỏng Thanh toán:** Xử lý thanh toán được mô phỏng thông qua đầu vào bàn phím mà không có phần cứng xử lý tiền tệ thực tế
- **Xuất hàng:** Không có điều khiển cơ cấu chấp hành vật lý để xuất hàng (nguyên mẫu tập trung vào logic phần mềm)
- **Giao dịch Đơn lẻ:** Hệ thống xử lý một giao dịch khách hàng tại một thời điểm
- **Hạn chế Hiển thị:** LCD 16x2 chỉ hiển thị văn bản giới hạn thông tin hiển thị
- **Dải giá:** Giá sản phẩm có thể điều chỉnh từ 1.000 đến 99.000 VND với bước nhảy 1.000 VND
- **Giới hạn Bộ nhớ:** 64KB Flash và 20KB RAM hạn chế độ phức tạp của chương trình và lưu trữ dữ liệu

## 1.4 Cấu trúc Báo cáo

Báo cáo này được cấu trúc như sau:

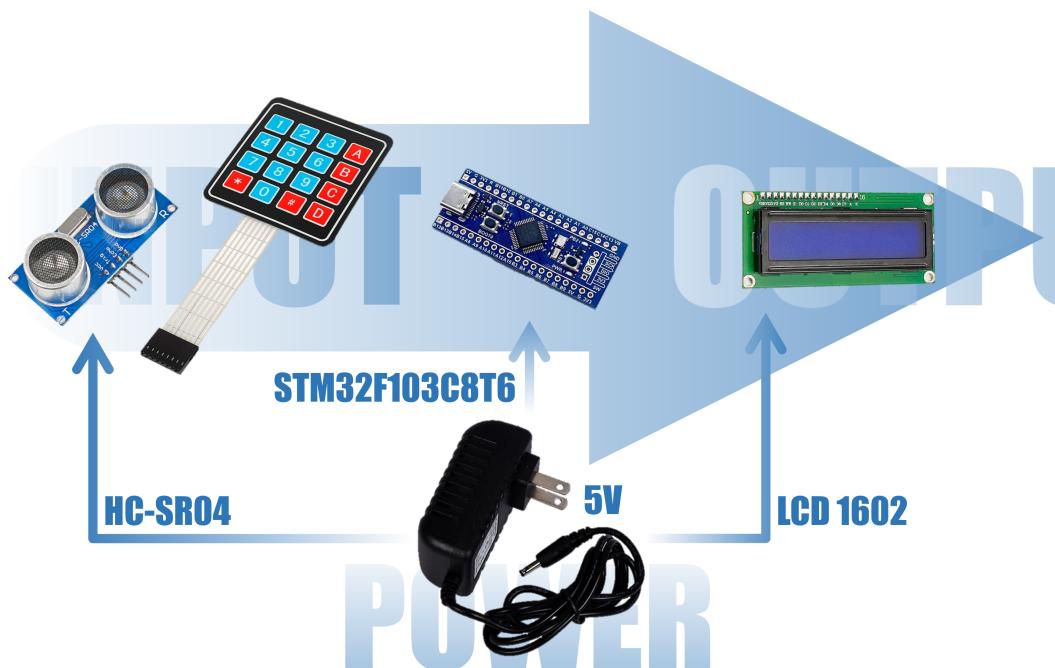
- **Phần 2 (Tổng quan Hệ thống):** Cung cấp kiến trúc cao, triết lý thiết kế và yêu cầu hệ thống
- **Phần 3 (Thành phần Phần cứng):** Chi tiết tất cả các thành phần phần cứng bao gồm vi điều khiển, cảm biến, màn hình và thiết bị đầu vào

- **Phần 4 (Thiết kế Phần mềm):** Giải thích máy trạng thái hữu hạn, hệ thống định thời và kiến trúc phần mềm
- **Phần 5 (Hiện thực):** Mô tả việc hiện thực các trình điều khiển ngoại vi, thuật toán và các tính năng chính
- **Phần 6 (Kiểm chứng):** Trình bày kết quả kiểm tra, ví dụ vận hành và phân tích hiệu năng
- **Phần 7 (Hướng phát triển):** Thảo luận về các cải tiến tiềm năng và cơ hội mở rộng
- **Phần 8 (Kết luận):** Tóm tắt các thành tựu, bài học kinh nghiệm và kết quả dự án
- **Phần 9 (Tài liệu tham khảo & Tác giả):** Liệt kê các tài liệu tham khảo kỹ thuật và đóng góp của tác giả

## 2 Tổng quan Hệ thống

### 2.1 Kiến trúc Hệ thống

Hệ thống máy bán hàng tự động tuân theo mẫu thiết kế kiến trúc phân lớp, tách biệt phần trùu tượng hóa phần cứng, logic nghiệp vụ và giao diện người dùng. Hình 1 minh họa kiến trúc tổng thể của hệ thống.



Hình 1: Sơ đồ kiến trúc hệ thống

Kiến trúc bao gồm bốn lớp chính:

#### 2.1.1 Lớp Phần cứng

Lớp này bao gồm tất cả các thành phần vật lý và giao diện phần cứng cấp thấp:

- **Vi điều khiển STM32F103C8T6:** Đơn vị xử lý trung tâm quản lý mọi hoạt động, chạy ở tần số 72 MHz với lõi ARM Cortex-M3
- **Màn hình LCD I2C:** Màn hình ký tự 16x2 kết nối qua bộ mở rộng I2C PCF8574 trên các chân PB6 (SCL) và PB7 (SDA)
- **Bàn phím ma trận 4x4:** Thiết bị đầu vào với các hàng trên PA0-PA3 và các cột trên PA4-PA7
- **Cảm biến siêu âm HC-SR04:** Mô-đun đo khoảng cách sử dụng bộ bắt đầu vào TIM1 để định thời chính xác

- **Bộ nhớ Flash:** Flash nội bộ 64KB với Trang 63 được dành riêng để lưu trữ dữ liệu
- **Đèn báo LED:** LED tích hợp trên PC13 để chỉ báo trạng thái hệ thống

### 2.1.2 Lớp Trình điều khiển

Các trình điều khiển ngoại vi tùy chỉnh cung cấp sự trừu tượng hóa phần cứng:

- **Trình điều khiển LCD I2C (tv\_lcd\_i2c.c):** Hiện thực giao tiếp LCD 4-bit qua giao thức I2C bit-banged
- **Trình điều khiển Bàn phím (keypad.c):** Cung cấp chức năng quét, chống rung và ánh xạ ký tự
- **Trình điều khiển Cảm biến (sensor.c):** Xử lý kích hoạt siêu âm và tính toán khoảng cách thông qua bộ bắt thời gian
- **Trình điều khiển I2C (i2c.c):** Hiện thực giao thức I2C dựa trên phần mềm để giao tiếp LCD
- **Quản lý Bộ định thời (timer.c):** Hệ thống định thời phần mềm cho các thời gian chờ và độ trễ trạng thái

### 2.1.3 Lớp Ứng dụng

Logic nghiệp vụ và quản lý trạng thái:

- **Máy trạng thái hữu hạn (fsm\_vm.c):** Logic cốt lõi với 19 trạng thái quản lý luồng giao dịch, xử lý lỗi và chuyển đổi chế độ
- **Quản lý Kho hàng (store.c):** Xử lý cấu trúc dữ liệu sản phẩm, thao tác đọc/ghi Flash và cập nhật tồn kho
- **Mô-đun Quản trị (ADMIN.c):** Xác thực mật khẩu và quản lý bảo mật

### 2.1.4 Lớp Giao diện Người dùng

Quản lý trình bày và tương tác:

- Định dạng hiển thị LCD và trình bày văn bản căn giữa
- Thông dịch đầu vào bàn phím và ánh xạ lệnh
- Phản hồi trạng thái và hiển thị thông báo lỗi
- Luồng điều hướng đa màn hình

## 2.2 Triết lý Thiết kế

Thiết kế hệ thống tuân theo một số nguyên tắc chính:

### 2.2.1 Tính Mô-đun và Phân tách Mối quan tâm

Mỗi thành phần chức năng được hiện thực như một mô-đun độc lập với các giao diện được xác định rõ. Ví dụ, trình điều khiển LCD cung cấp các hàm cấp cao như `lcd_write_string()` mà không để lộ chi tiết giao tiếp I2C. Tính mô-đun này tạo điều kiện thuận lợi cho việc kiểm tra, gỡ lỗi và sửa đổi trong tương lai.

### 2.2.2 Kiến trúc Hướng Trạng thái

Hệ thống sử dụng máy trạng thái hữu hạn làm cấu trúc điều khiển cốt lõi, cung cấp:

- **Hành vi Dự đoán được:** Mỗi trạng thái có các điều kiện đầu vào, hành động và chuyển đổi đầu ra được xác định
- **Phục hồi Lỗi:** Các trạng thái lỗi cho phép xử lý nhẹ nhàng các đầu vào không hợp lệ và điều kiện thời gian chờ
- **Khả năng Bảo trì:** Thêm các tính năng mới đòi hỏi thêm các trạng thái và chuyển đổi mà không cần cấu trúc lại mã hiện có
- **Gỡ lỗi:** Theo dõi biến trạng thái đơn giản hóa việc phân tích hành vi hệ thống

### 2.2.3 Vận hành An toàn

Nhiều cơ chế an toàn đảm bảo độ tin cậy của hệ thống:

- Bộ định thời gian chờ ngăn chặn việc chờ đợi vô thời hạn (30 giây cho khách hàng, 60 giây cho quản trị viên)
- Bộ đếm lỗi giới hạn các lỗi liên tiếp (tối đa 5 lần thử)
- Kiểm tra đầu vào từ chối số lượng và số tiền thanh toán không hợp lệ
- Xác minh số ma thuật Flash ngăn chặn việc tải dữ liệu bị hỏng

### 2.2.4 Thiết kế Lấy Người dùng làm Trung tâm

Giao diện ưu tiên trải nghiệm người dùng thông qua:

- Điều hướng rõ ràng với các ánh xạ phím nhất quán (U/D để điều hướng, # để xác nhận, \* để quay lại)
- Phản hồi ngay lập tức cho mọi hành động của người dùng
- Hiển thị văn bản căn giữa để cải thiện khả năng đọc
- Thông báo lỗi cung cấp thông tin với hướng dẫn phục hồi
- Tự động quay lại trạng thái an toàn khi hết thời gian chờ

## 2.3 Yêu cầu Hệ thống

### 2.3.1 Yêu cầu Chức năng

- FR1: **Phát hiện Khách hàng:** Hệ thống sẽ phát hiện sự hiện diện của khách hàng trong phạm vi 20cm trong 3 giây liên tiếp và tự động bật nguồn
- FR2: **Duyệt Sản phẩm:** Người dùng sẽ điều hướng qua 16 sản phẩm bằng các phím Lên/Xuống với cập nhật hiển thị thời gian thực
- FR3: **Thông tin Sản phẩm:** Hệ thống sẽ hiển thị tên sản phẩm, tên người chơi, số lượng có sẵn và giá
- FR4: **Chọn Số lượng:** Người dùng sẽ nhập số lượng từ 1-9 đơn vị với xác thực và phản hồi lỗi
- FR5: **Xử lý Thanh toán:** Hệ thống sẽ chấp nhận các mệnh giá tiền tệ Việt Nam (5K, 10K, 20K, 50K, 100K, 200K, 500K VND) và tính toán tiền thừa
- FR6: **Quản lý Kho hàng:** Hệ thống sẽ theo dõi mức tồn kho và ngăn chặn bán hàng khi hết hàng
- FR7: **Lưu trữ Dữ liệu:** Dữ liệu kho hàng sẽ tồn tại qua các chu kỳ nguồn sử dụng bộ nhớ Flash
- FR8: **Truy cập Quản trị:** Chế độ quản trị được bảo vệ bằng mật khẩu sẽ cho phép điều chỉnh kho hàng và giá cả
- FR9: **Xử lý Lỗi:** Hệ thống sẽ xử lý các đầu vào không hợp lệ, thời gian chờ và hiển thị các thông báo lỗi thích hợp
- FR10: **Hoàn tất Giao dịch:** Khi thanh toán thành công, hệ thống sẽ cập nhật kho hàng và quay lại lựa chọn sản phẩm

### 2.3.2 Yêu cầu Phi chức năng

- NFR1: **Thời gian Phản hồi:** Hệ thống sẽ phản hồi nhấn phím trong vòng 100ms
- NFR2: **Độ tin cậy:** Hệ thống sẽ hoạt động liên tục với các cơ chế phục hồi lỗi ngăn chặn khóa vĩnh viễn
- NFR3: **Khả năng Sử dụng:** Giao diện phải trực quan với đường cong học tập tối thiểu cho người dùng lân đàu
- NFR4: **Bảo mật:** Chế độ quản trị sẽ yêu cầu mật khẩu 6 chữ số với bảo vệ thời gian chờ
- NFR5: **Khả năng Bảo trì:** Mã nguồn phải có tính mô-đun với tài liệu rõ ràng và quy ước đặt tên nhất quán
- NFR6: **Hiệu năng:** Cập nhật LCD sẽ hoàn thành trong vòng 50ms để có trải nghiệm người dùng mượt mà
- NFR7: **Hiệu quả Tài nguyên:** Chương trình phải phù hợp với giới hạn 64KB Flash và 20KB RAM
- NFR8: **Sự mạnh mẽ:** Hệ thống sẽ xử lý nhiều cảm biến và chống rung đầu vào hiệu quả

## 2.4 Luồng Vận hành Hệ thống

Hoạt động hoàn chỉnh của hệ thống tuân theo trình tự sau:

### 1. Giai đoạn Khởi tạo:

- Bật nguồn và khởi tạo phần cứng
- Tải kho hàng từ bộ nhớ Flash (hoặc khởi tạo mặc định nếu là lần khởi động đầu tiên)
- Cấu hình tất cả các thiết bị ngoại vi (GPIO, I2C, Timer)
- Hiển thị màn hình khởi tạo
- Vào trạng thái nhàn rỗi với giám sát cảm biến

### 2. Giai đoạn Phát hiện Khách hàng:

- Quét liên tục cảm biến siêu âm mỗi 100ms
- Phát hiện sự hiện diện khi khoảng cách  $\leq 20\text{cm}$  trong 3 giây
- Kích hoạt đèn nền và hiển thị thông báo chào mừng
- Chuyển sang chế độ chọn sản phẩm

### 3. Giai đoạn Chọn Sản phẩm:

- Hiển thị sản phẩm hiện tại (ID, tên, người chơi)
- Chấp nhận các phím Lên/Xuống để điều hướng qua 16 sản phẩm
- Phím # để xem thông tin chi tiết
- Phím R để vào chế độ quản trị (yêu cầu mật khẩu)
- Thời gian chờ không hoạt động 30 giây quay lại trạng thái nhàn rỗi

### 4. Giai đoạn Mua hàng:

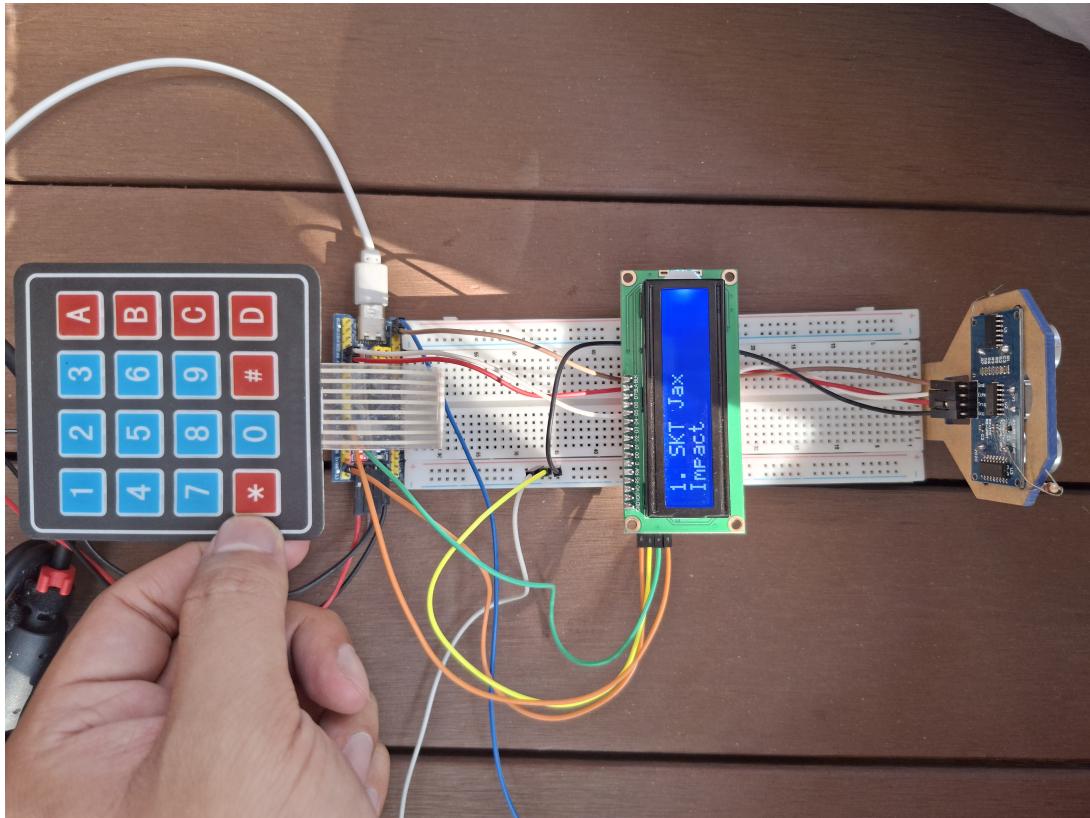
- Hiển thị chi tiết số lượng và giá
- Chấp nhận đầu vào số lượng (1-9 đơn vị)
- Xác thực tình trạng còn hàng
- Tính toán và hiển thị tổng số tiền thanh toán
- Chấp nhận đầu vào thanh toán với xác thực mệnh giá
- Tính toán tiền thừa nếu trả thừa
- Cập nhật kho hàng và lưu vào Flash

### 5. Giai đoạn Chế độ Quản trị:

- Xác thực với mật khẩu 6 chữ số
- Điều hướng sản phẩm để điều chỉnh
- Sửa đổi số lượng (0-9) và giá (1K-99K VND)
- Lưu thay đổi vào bộ nhớ Flash
- Thời gian chờ không hoạt động 60 giây để bảo mật

### 3 Thành phần Phần cứng

Phần này cung cấp thông số kỹ thuật chi tiết và chi tiết tích hợp cho tất cả các thành phần phần cứng được sử dụng trong hệ thống máy bán hàng tự động.



Hình 2: Hiện thực phần cứng của hệ thống Máy bán hàng tự động

#### 3.1 Vi điều khiển - STM32F103C8T6

##### 3.1.1 Thông số Kỹ thuật

STM32F103C8T6 (thường được gọi là "Blue Pill") đóng vai trò là đơn vị xử lý trung tâm của hệ thống với các thông số kỹ thuật sau:

- **Lõi:** Bộ xử lý ARM Cortex-M3 32-bit RISC
- **Tốc độ Xung nhịp:** Tối đa 72 MHz
- **Bộ nhớ Flash:** 64 KB (chính thức), thường là 128 KB trong thực tế
- **SRAM:** 20 KB
- **Chân GPIO:** 37 chân I/O với đầu vào chịu được 5V
- **Bộ định thời:** 3x 16-bit mục đích chung, 1x bộ định thời điều khiển nâng cao
- **Giao tiếp:** 2x I2C, 3x USART, 2x SPI, 1x USB, 1x CAN
- **ADC:** 2x ADC 12-bit với 10 kênh

- **Điện áp Hoạt động:** 2.0V đến 3.6V (với I/O chịu được 5V)
- **Đóng gói:** LQFP48 (7mm × 7mm)

### 3.1.2 Lý do Lựa chọn

STM32F103C8T6 được chọn cho dự án này vì:

- **Sức mạnh Xử lý:** Xung nhịp 72 MHz cung cấp đủ hiệu năng cho các hoạt động thời gian thực
- **Bộ nhớ:** 64 KB Flash chứa được các trình điều khiển HAL và mã ứng dụng; 20 KB RAM đủ cho dữ liệu thời gian chạy
- **Hỗ trợ Ngoại vi:** Tích hợp sẵn I2C, bộ định thời và GPIO đáp ứng mọi yêu cầu giao diện
- **Hệ sinh thái Phát triển:** Hỗ trợ tuyệt vời thông qua STM32CubeIDE và thư viện HAL
- **Hiệu quả Chi phí:** Bo mạch phát triển giá rẻ có sẵn rộng rãi
- **Hỗ trợ Cộng đồng:** Cộng đồng người dùng lớn và tài liệu phong phú

### 3.1.3 Cấu hình Chân

Các gán chân quan trọng cho dự án này:

Bảng 1: Gán chân STM32F103C8T6

Chân	Chức năng	Mô tả
PA0-PA3	Hàng Bàn phím	Chân đầu ra để quét bàn phím
PA4-PA7	Cột Bàn phím	Chân đầu vào với điện trở kéo lên
PB6	I2C1_SCL	Xung nhịp I2C cho giao tiếp LCD
PB7	I2C1_SDA	Dữ liệu I2C cho giao tiếp LCD
PA8	TIM1_CH1	ECHO cảm biến siêu âm (Bắt đầu vào)
PA9	GPIO Output	TRIG cảm biến siêu âm
PC13	GPIO Output	Dèn báo LED (tích cực mức thấp)
PA13	SWDIO	Dữ liệu I/O Gõ lỗi Serial Wire
PA14	SWCLK	Xung nhịp Gõ lỗi Serial Wire

## 3.2 Mô-đun Hiển thị - LCD 16x2 với I2C

### 3.2.1 Thông số LCD

- **Hiển thị:** 16 ký tự × 2 dòng
- **Bộ điều khiển:** Tương thích HD44780
- **Đèn nền:** LED xanh dương với độ sáng có thể điều chỉnh
- **Kích thước Ký tự:** 5×8 điểm

- **Điện áp Hoạt động:** 5V
- **Giao diện:** Chế độ song song 4-bit qua bộ chuyển đổi I2C

### 3.2.2 Bộ chuyển đổi I2C (PCF8574)

LCD sử dụng bộ mở rộng I/O I2C PCF8574 để đơn giản hóa việc đi dây:

- **Chip:** Bộ mở rộng I/O 8-bit PCF8574T
- **Địa chỉ I2C:** Thường là 0x27 hoặc 0x3F (có thể cấu hình qua jumper)
- **Điện áp Hoạt động:** 5V
- **Ánh xạ Chân:**
  - P0-P3: Chân dữ liệu LCD D4-D7 (chế độ 4-bit)
  - P4: Chọn thanh ghi LCD (RS)
  - P5: Đọc/Ghi LCD (R/W)
  - P6: Cho phép LCD (E)
  - P7: Điều khiển đèn nền

### 3.2.3 Giao thức Giao tiếp

Hệ thống hiện thực giao tiếp I2C bit-banged:

1. **Điều kiện START:** SDA chuyển từ cao xuống thấp trong khi SCL ở mức cao
2. **Khung Địa chỉ:** Gửi địa chỉ slave 7-bit + bit R/W
3. **Truyền Dữ liệu:** Gửi dữ liệu 8-bit với kiểm tra ACK
4. **Điều kiện STOP:** SDA chuyển từ thấp lên cao trong khi SCL ở mức cao

Đối với hoạt động LCD ở chế độ 4-bit:

1. Gửi nibble cao (4 bit) của byte dữ liệu
2. Tạo xung chân Enable
3. Gửi nibble thấp (4 bit) của byte dữ liệu
4. Tạo xung chân Enable

## 3.3 Thiết bị Đầu vào - Bàn phím Ma trận 4×4

### 3.3.1 Bố cục Bàn phím

Bàn phím ma trận cung cấp 16 phím được sắp xếp thành 4 hàng và 4 cột:

Bảng 2: Bố cục Bàn phím

<b>1</b>	<b>2</b>	<b>3</b>	<b>U (Lên)</b>
<b>4</b>	<b>5</b>	<b>6</b>	<b>D (Xuống)</b>
<b>7</b>	<b>8</b>	<b>9</b>	<b>L (Trái)</b>
<b>*</b>	<b>0</b>	<b>#</b>	<b>R (Phải)</b>

### 3.3.2 Cơ chế Quét

Bàn phím hoạt động dựa trên nguyên tắc quét hàng-cột:

- **Hàng (PA0-PA3):** Cấu hình là chân đầu ra, bình thường ở mức CAO
- **Cột (PA4-PA7):** Cấu hình là chân đầu vào với điện trở kéo lên bên trong

Thuật toán Quét:

1. Đặt tất cả các hàng ở mức CAO
2. Kéo lần lượt từng hàng xuống mức THẤP
3. Đọc tất cả các chân cột
4. Nếu bất kỳ cột nào đọc mức THẤP, một phím tại giao điểm hàng-cột đó được nhấn
5. Thực hiện chống rung (độ trễ 20ms)
6. Chờ nhả phím trước khi trả về ký tự

### 3.3.3 Chiến lược Chống rung

Các công tắc cơ học thể hiện hiện tượng rung, gây ra nhiều chuyển đổi trong một lần nhấn. Hệ thống hiện thực chống rung bằng phần mềm:

```

1 if (HAL_GPIO_ReadPin(GPIOA, col_pins[c]) == GPIO_PIN_RESET) {
2     HAL_Delay(20); // Debounce delay
3
4     if (HAL_GPIO_ReadPin(GPIOA, col_pins[c]) == GPIO_PIN_RESET) {
5         // Key press confirmed
6         while (HAL_GPIO_ReadPin(GPIOA, col_pins[c]) ==
7             GPIO_PIN_RESET);
8         // Wait for release
9         return keymap[r][c];
10    }
}

```

Listing 1: Hiện thực Chống rung Bàn phím

## 3.4 Mô-đun Cảm biến - Cảm biến Siêu âm HC-SR04

### 3.4.1 Thông số Cảm biến

- **Điện áp Hoạt động:** 5V DC
- **Phạm vi Đo:** 2cm đến 400cm
- **Độ chính xác:** ±3mm
- **Góc Đo:** Hình nón 15 độ
- **Đầu vào Kích hoạt:** Xung TTL 10 micro giây
- **Đầu ra Echo:** Xung TTL tỷ lệ với khoảng cách
- **Tần số Siêu âm:** 40 kHz

### 3.4.2 Nguyên lý Hoạt động

HC-SR04 sử dụng phép đo thời gian bay (time-of-flight):

1. Vi điều khiển gửi xung CAO 10 micro giây đến chân TRIG
2. Cảm biến phát 8 xung siêu âm ở tần số 40 kHz
3. Chân ECHO lên mức CAO khi bắt đầu phát
4. Sóng siêu âm phản xạ lại từ vật thể và quay trở lại
5. Chân ECHO xuống mức THẤP khi phát hiện phản xạ
6. Tính toán khoảng cách:  $Distance = \frac{Time \times 0.034}{2}$  cm

Hệ số 0.034 đại diện cho tốc độ âm thanh ( $340 \text{ m/s} = 0.034 \text{ cm/micro giây}$ ), và chia cho 2 tính đến hành trình khứ hồi.

### 3.4.3 Cấu hình Bắt đầu vào Bộ định thời

Hệ thống sử dụng Kênh 1 của TIM1 được cấu hình cho chế độ Bắt đầu vào (Input Capture):

- **Chế độ Bắt:** Cả hai cạnh (lên và xuống)
- **Tần số Bộ định thời:** 1 MHz (độ phân giải 1 micro giây)
- **Ngắt:** Được kích hoạt khi có sự kiện bắt

Trình tự do:

1. Bắt lần đầu (cạnh lên): Ghi lại thời gian bắt đầu IC\_Val1
2. Bắt lần hai (cạnh xuống): Ghi lại thời gian kết thúc IC\_Val2
3. Tính hiệu số:  $Difference = IC\_Val2 - IC\_Val1$
4. Xử lý tràn bộ định thời: Nếu  $IC\_Val1 > IC\_Val2$ , cộng thêm chu kỳ bộ định thời
5. Chuyển đổi sang khoảng cách:  $Distance = Difference \times 0.034/2$

## 3.5 Bộ nhớ - Flash Nội bộ

### 3.5.1 Tổ chức Bộ nhớ Flash

Cấu trúc bộ nhớ Flash của STM32F103C8T6:

- **Tổng Dung lượng:** 64 KB (128 trang  $\times$  512 byte) hoặc biến thể 128 KB
- **Kích thước Trang:** 1 KB mỗi trang
- **Địa chỉ Cơ sở:** 0x08000000
- **Sử dụng Ứng dụng:** Các trang 0-62 cho mã chương trình
- **Lưu trữ Dữ liệu:** Trang 63 (0x0800FC00) để lưu trữ kho hàng

### 3.5.2 Quy trình Lập trình Flash

Các hàm thư viện HAL cho các hoạt động Flash:

```

1 // Unlock Flash for writing
2 HAL_FLASH_Unlock();
3
4 // Erase page before writing
5 FLASH_EraseInitTypeDef EraseInitStruct;
6 EraseInitStruct.TypeErase = FLASH_TYPEERASE_PAGES;
7 EraseInitStruct.PageAddress = FLASH_ADDR_PAGE_63;
8 EraseInitStruct.NbPages = 1;
9 HAL_FLASHEx_Erase(&EraseInitStruct, &PageError);
10
11 // Write data word by word
12 HAL_FLASH_Program(FLASH_TYPEPROGRAM_WORD, address, data);
13
14 // Lock Flash after writing
15 HAL_FLASH_Lock();

```

Listing 2: Các hoạt động Bộ nhớ Flash

### 3.5.3 Chiến lược Bảo toàn Dữ liệu

Cấu trúc dữ liệu kho hàng (44 byte mỗi sản phẩm):

- **Số Ma thuật:** 4 byte (0xDEADBEEF) để xác thực dữ liệu
- **Mảng Sản phẩm:** 16 sản phẩm × 44 byte = 704 byte
- **Tổng Lưu trữ:** 708 byte (vừa trong trang 1 KB)

Khi bật nguồn:

1. Đọc số ma thuật từ Trang 63 của Flash
2. Nếu số ma thuật khớp 0xDEADBEEF: Tải kho hàng từ Flash
3. Nếu số ma thuật không hợp lệ: Khởi tạo kho hàng mặc định và lưu vào Flash

## 3.6 Nguồn điện và Giao diện Lập trình

### 3.6.1 Yêu cầu Nguồn điện

- **Lõi STM32:** 3.3V, 50mA
- **Mô-đun LCD:** 5V, 30mA (không đèn nền), 150mA (có đèn nền)
- **Cảm biến HC-SR04:** 5V, 15mA
- **Bàn phím:** Tiêu thụ dòng không đáng kể
- **Tổng Hệ thống:** 200mA ở 5V hoạt động hiển hình

Phân phối nguồn:

- Bộ chuyển đổi ngoài 5V cung cấp cho các đường nguồn breadboard
- Bộ ổn áp 3.3V tích hợp (AMS1117-3.3) cấp nguồn cho STM32
- LCD và cảm biến kết nối trực tiếp với đường nguồn 5V

### 3.6.2 Bộ nạp ST-Link V2

- **Giao diện:** SWD (Serial Wire Debug)
- **Kết nối:**
  - SWDIO → PA13
  - SWCLK → PA14
  - GND → GND
  - 3.3V → 3.3V (optional for powering during debug)
- **Tính năng:** Programming, debugging, real-time variable monitoring

## 4 Thiết kế Phần mềm

### 4.1 Kiến trúc Máy trạng thái Hữu hạn

Cốt lõi của phần mềm máy bán hàng tự động là một máy trạng thái hữu hạn (FSM) với 19 trạng thái riêng biệt quản lý toàn bộ vòng đời giao dịch, xử lý lỗi và các chức năng quản trị. FSM cung cấp hành vi xác định và đơn giản hóa việc gỡ lỗi bằng cách làm cho trạng thái hệ thống rõ ràng tại mọi thời điểm.

#### 4.1.1 Định nghĩa Trạng thái

Hệ thống hiện thực các trạng thái sau, được định nghĩa trong `fsm_vm.h`:

1	#define INIT	1
2	#define WAIT_SENSOR	2
3	#define WELCOME_SECTION	3
4	#define CHOOSING_SKIN	4
5	#define DISPLAY_INFO	5
6	#define CHOOSING_QUANTITY	6
7	#define OUT_OF_STOCK_NOTIFICATION	7
8	#define QUANTITY_ERROR	8
9	#define MAX_ERROR_STATE	9
10	#define PAYMENT_SHOW_TOTAL	10
11	#define PAYMENT_INPUT	11
12	#define PAYMENT_ERROR	12
13	#define PAYMENT_INFO_WAIT	13
14	#define THANKS	14
15	#define ADMIN_MODE	15
16	#define CHOOSING_SKIN_TO_ADJUST	16
17	#define TIMEOUT_ADMIN_MODE	17
18	#define ADJUST_QUANTITY_AND_PRICE	18
19	#define CONFIRM_EXIT_ADMIN_MODE	19

Listing 3: Định nghĩa Trạng thái FSM

#### 4.1.2 Phân loại Trạng thái

19 trạng thái được tổ chức thành các danh mục chức năng:

##### Trạng thái Khởi tạo (1-2):

- **INIT:** Khởi tạo hệ thống, thiết lập ngoại vi, tải kho hàng từ Flash
- **WAIT\_SENSOR:** Trạng thái nhàn rỗi với giám sát cảm biến siêu âm liên tục để phát hiện khách hàng

##### Trạng thái Giao dịch Khách hàng (3-14):

- **WELCOME\_SECTION:** Hiển thị thông báo chào mừng trong 3 giây
- **CHOOSING\_SKIN:** Duyệt sản phẩm với điều hướng Lên/Xuống

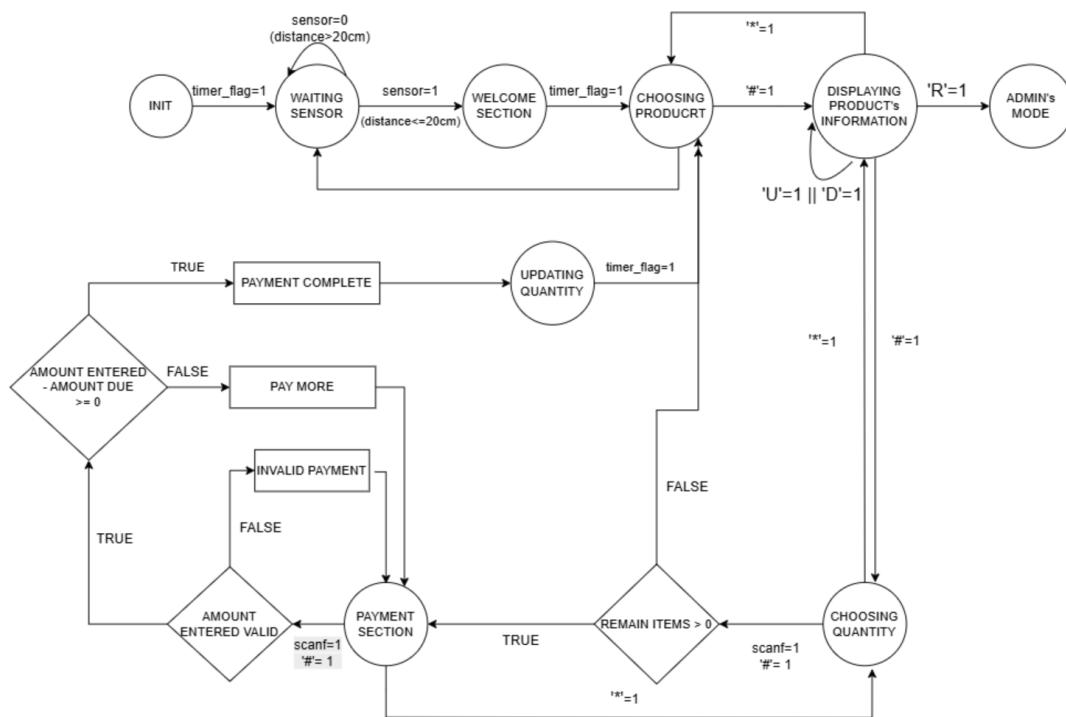
- **DISPLAY\_INFO:** Hiển thị thông tin chi tiết sản phẩm (số lượng, giá)
- **CHOOSING\_QUANTITY:** Nhập số lượng (1-9)
- **OUT\_OF\_STOCK\_NOTIFICATION:** Cảnh báo khi sản phẩm đã chọn không có sẵn
- **QUANTITY\_ERROR:** Xử lý nhập số lượng không hợp lệ với thử lại
- **MAX\_ERROR\_STATE:** Hủy giao dịch sau 5 lỗi liên tiếp
- **PAYMENT\_SHOW\_TOTAL:** Hiển thị tổng số tiền phải trả
- **PAYMENT\_INPUT:** Chấp nhận các mệnh giá thanh toán
- **PAYMENT\_ERROR:** Xử lý số tiền thanh toán không hợp lệ
- **PAYMENT\_INFO\_WAIT:** Hiển thị số tiền còn lại hoặc tiền thừa
- **THANKS:** Thông báo cảm ơn và cập nhật kho hàng

#### Trạng thái Chế độ Quản trị (15-19):

- **ADMIN\_MODE:** Xác nhận đăng nhập thành công
- **CHOOSING\_SKIN\_TO\_ADJUST:** Chọn sản phẩm để sửa đổi
- **TIMEOUT\_ADMIN\_MODE:** Tự động đăng xuất sau 60 giây không hoạt động
- **ADJUST\_QUANTITY\_AND\_PRICE:** Chính sửa kho hàng và giá cả
- **CONFIRM\_EXIT\_ADMIN\_MODE:** Hộp thoại xác nhận đăng xuất

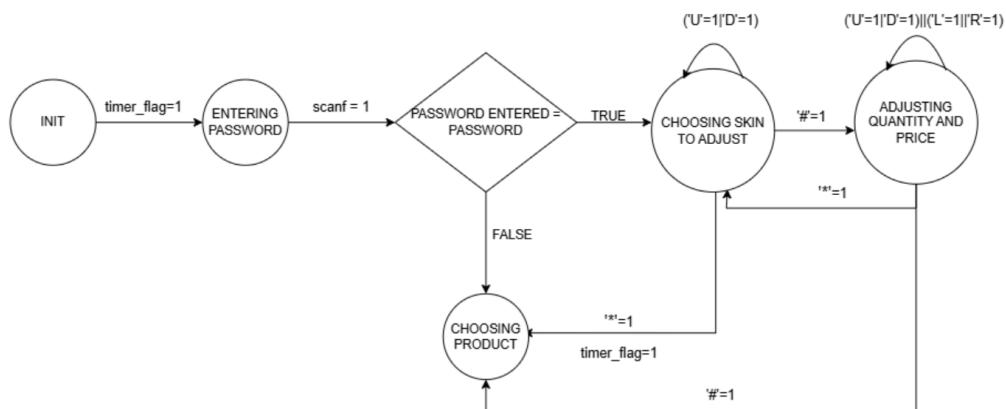
#### 4.1.3 Sơ đồ Chuyển đổi Trạng thái

Các chuyển đổi trạng thái tuân theo các đường dẫn chính sau:



Hình 3: Sơ đồ chuyển đổi trạng thái FSM (Chế độ Người dùng)

ADMIN's MODE



Hình 4: Sơ đồ chuyển đổi trạng thái FSM (Chế độ Quản trị)

### **Luồng Giao dịch Bình thường:**

```
INIT -> WAIT_SENSOR -> WELCOME_SECTION -> CHOOSING_SKIN  
-> DISPLAY_INFO -> CHOOSING_QUANTITY -> PAYMENT_SHOW_TOTAL  
-> PAYMENT_INPUT -> PAYMENT_INFO_WAIT -> THANKS -> CHOOSING_SKIN
```

## Đường dẫn Phục hồi Lỗi:

- Lỗi số lượng: CHOOSING\_QUANTITY → QUANTITY\_ERROR → CHOOSING\_QUANTITY (hoặc MAX\_ERROR\_STATE sau 5 lần thử)

- Lỗi thanh toán: PAYMENT\_INPUT → PAYMENT\_ERROR → PAYMENT\_INPUT (hoặc INIT sau 5 lần thử)
- Hết hàng: DISPLAY\_INFO → OUT\_OF\_STOCK\_NOTIFICATION → DISPLAY\_INFO
- Thời gian chờ: Bất kỳ trạng thái khách hàng nào → trạng thái phục hồi thích hợp sau 30 giây

#### Đường dẫn Quản trị:

CHOOSING\_SKIN (Phím R + mật khẩu) → ADMIN\_MODE  
 → CHOOSING\_SKIN\_TO\_ADJUST → ADJUST\_QUANTITY\_AND\_PRICE  
 → CHOOSING\_SKIN\_TO\_ADJUST hoặc CONFIRM\_EXIT\_ADMIN\_MODE

#### 4.1.4 Hiện thực FSM

FSM được hiện thực sử dụng cấu trúc switch-case trong fsm\_vm.c:

```

1 void fsm_vm_run(void) {
2     switch (status) {
3         case INIT:
4             if (init_timer_flag == 1) {
5                 status = WAIT_SENSOR;
6                 ledOFF();
7                 lcd_clear();
8                 detection_start_time = 0;
9                 sensor_tick_last = HAL_GetTick();
10            }
11            break;
12
13        case WAIT_SENSOR:
14            Sensor_Trigger();
15            HAL_Delay(50);
16
17            if (Distance > 0 && Distance <= 20) {
18                if (detection_start_time == 0) {
19                    detection_start_time = HAL_GetTick();
20                } else if (HAL_GetTick() - detection_start_time >=
21                           3000) {
22                    status = WELCOME_SECTION;
23                    ledON();
24                    // Display welcome screen
25                }
26            } else {
27                detection_start_time = 0;
28            }
29            break;
30
31        // ... additional 17 states ...
32    }
}

```

Listing 4: Cấu trúc Cốt lõi FSM

Mỗi trạng thái hiện thực:

1. Hành động đầu vào (cập nhật hiển thị, khởi tạo biến)
2. Xử lý đầu vào (quét bàn phím)
3. Điều kiện chuyển đổi (cờ bộ định thời, đầu vào người dùng, kết quả xác thực)
4. Hành động đầu ra (dọn dẹp, thiết lập bộ định thời)

## 4.2 Hệ thống Quản lý Bộ định thời

Hệ thống sử dụng cơ chế bộ định thời phần mềm để quản lý độ trễ, thời gian chờ và chuyển đổi trạng thái mà không chặn thực thi.

### 4.2.1 Các loại Bộ định thời

Năm loại bộ định thời độc lập được hiện thực:

Bảng 3: Các loại Bộ định thời Phần mềm

Bộ định thời	Thời lượng	Mục đích
init_timer	1000ms	Độ trễ khởi tạo trước khi kích hoạt cảm biến
welcome_timer	3000ms	Thời gian hiển thị màn hình chào mừng
timeout_timer	30s / 60s	Thời gian chờ không hoạt động (khách/admin)
message_timer	3000ms	Thời gian hiển thị thông báo tin
sensor_timer	100ms	Khoảng thời gian thăm dò cảm biến siêu âm

### 4.2.2 Hiện thực Bộ định thời

Mỗi bộ định thời bao gồm ba thành phần:

```

1 // Counter: Decremented each millisecond
2 int welcome_timer_counter = 0;
3
4 // Flag: Set to 1 when counter reaches 0
5 int welcome_timer_flag = 0;
6
7 // Setter function: Initialize counter and clear flag
8 void setWelcomeTimer(int duration) {
9     welcome_timer_counter = duration;
10    welcome_timer_flag = 0;
11 }
```

Listing 5: Cấu trúc Dữ liệu Bộ định thời

### 4.2.3 Thực thi Bộ định thời

Hàm `timerRun()` được gọi mỗi 1ms (thường trong ngắt SysTick):

```

1 void timerRun() {
2     if (welcome_timer_counter > 0) {
3         welcome_timer_counter--;
4         if (welcome_timer_counter <= 0) {
5             welcome_timer_flag = 1;
6         }
7     }
8     // ... repeat for other timers ...
9 }
```

Listing 6: Hàm Cập nhật Bộ định thời

### 4.2.4 Mẫu Sử dụng Bộ định thời

Cách sử dụng điển hình trong các trạng thái FSM:

```

1 case WELCOME_SECTION:
2     // State entry: Set timer
3     if (entered_state) {
4         setWelcomeTimer(3000);    // 3 second display
5         lcd_clear();
6         lcd_write_string("WELCOME!");
7     }
8
9     // State execution: Check flag
10    if (welcome_timer_flag == 1) {
11        status = CHOOSING_SKIN; // Transition
12        lcd_clear();
13    }
14    break;
```

Listing 7: Ví dụ Sử dụng Bộ định thời

## 4.3 Cấu trúc Dữ liệu và Quản lý Bộ nhớ

### 4.3.1 Cấu trúc Dữ liệu Sản phẩm

Hệ thống kho hàng sử dụng kiểu dữ liệu có cấu trúc cho sản phẩm:

```

1 typedef struct {
2     uint8_t id;           // Product ID (1-16)
3     char skinName[16];   // Product name (e.g., "SKT Jax")
4     char playerName[16]; // Player name (e.g., "Impact")
5     uint32_t quantity;  // Stock level (0-9)
6     uint32_t price;      // Price in VND
7 } Skin;
8
9 // Global inventory array
10 Skin skt_skins[16];
```

Listing 8: Cấu trúc Dữ liệu Sản phẩm

Dầu châm bộ nhớ: 44 byte mỗi sản phẩm × 16 sản phẩm = 704 byte

### 4.3.2 Bố cục Bộ nhớ Flash

Lưu trữ dữ liệu sử dụng bộ nhớ Flash nội bộ:

```

1 // Page 63 address: Last 1KB of 64KB Flash
2 #define FLASH_ADDR_PAGE_63 0x0800FC00
3
4 // Magic number for data validation
5 #define MAGIC_NUMBER          0xDEADBEEF
6
7 // Memory layout:
8 // Offset 0x00: Magic number (4 bytes)
9 // Offset 0x04: Skin array (704 bytes)
10 // Total: 708 bytes

```

Listing 9: Cấu hình Bộ nhớ Flash

### 4.3.3 Biến Toàn cục

Các biến trạng thái chính được duy trì bởi FSM:

```

1 int status = INIT;                                // Current FSM state
2 uint8_t current_id = 1;                            // Selected product ID
3 uint32_t input_quantity = 0;                      // User-entered quantity
4 uint8_t error_count = 0;                           // Consecutive error counter
5
6 uint32_t total_payable = 0;                        // Total payment required
7 uint32_t money_inserted_current = 0;              // Current denomination
    input
8 uint32_t money_paid_accumulated = 0;             // Total paid so far
9 uint8_t payment_error_count = 0;                  // Payment error counter
10
11 uint32_t detection_start_time = 0;               // Sensor detection
    timestamp

```

Listing 10: Biến Trạng thái Toàn cục

## 4.4 Các Thuật toán Chính

### 4.4.1 Thuật toán Phát hiện Khách hàng

Giám sát liên tục với lọc nhiễu:

```

1 // Trigger sensor measurement
2 Sensor_Trigger();
3 HAL_Delay(50);
4

```

```

5  if (Distance > 0 && Distance <= 20) {
6      // Object detected in range
7      if (detection_start_time == 0) {
8          // First detection: Record timestamp
9          detection_start_time = HAL_GetTick();
10     } else {
11         // Continuous detection: Check duration
12         if (HAL_GetTick() - detection_start_time >= 3000) {
13             // 3 seconds continuous detection confirmed
14             status = WELCOME_SECTION;
15             ledON();
16             // Display welcome message
17             detection_start_time = 0;
18         }
19     }
20 } else {
21     // No object or out of range: Reset
22     detection_start_time = 0;
23     ledOFF();
24 }
```

Listing 11: Logic Phát hiện Khách hàng

Yêu cầu 3 giây lọc nhiễu thoảng qua và đảm bảo sự hiện diện có chủ ý của khách hàng.

#### 4.4.2 Thuật toán Xác thực Thanh toán

Xử lý thanh toán đa mệnh giá:

```

1 // Valid Vietnamese currency denominations
2 int is_valid_money(uint32_t amount) {
3     switch(amount) {
4         case 5000:
5         case 10000:
6         case 20000:
7         case 50000:
8         case 100000:
9         case 200000:
10        case 500000:
11            return 1;
12        default:
13            return 0;
14    }
15 }
16
17 // Payment processing
18 if (is_valid_money(money_inserted_current)) {
19     money_paid_accumulated += money_inserted_current;
20
21     if (money_paid_accumulated < total_payable) {
22         uint32_t remaining = total_payable -
23             money_paid_accumulated;
```

```

23     // Display remaining amount
24 } else {
25     uint32_t change = money_paid_accumulated - total_payable;
26     // Payment complete, display change
27     status = THANKS;
28 }
29 } else {
30     payment_error_count++;
31     if (payment_error_count >= 5) {
32         status = INIT; // Abort transaction
33     } else {
34         status = PAYMENT_ERROR; // Retry
35     }
36 }
```

Listing 12: Xác thực Thanh toán

#### 4.4.3 Thuật toán Lưu trữ Kho hàng

Đọc/ghi Flash với xác thực:

```

1 void Store_SaveToFlash(void) {
2     // 1. Unlock Flash
3     HAL_FLASH_Unlock();
4
5     // 2. Erase page
6     FLASH_EraseInitTypeDef EraseInitStruct;
7     uint32_t PageError;
8     EraseInitStruct.TypeErase = FLASH_TYPEERASE_PAGES;
9     EraseInitStruct.PageAddress = FLASH_ADDR_PAGE_63;
10    EraseInitStruct.NbPages = 1;
11    HAL_FLASHEx_Erase(&EraseInitStruct, &PageError);
12
13    // 3. Write magic number
14    HAL_FLASH_Program(FLASH_TYPEPROGRAM_WORD,
15                      FLASH_ADDR_PAGE_63,
16                      MAGIC_NUMBER);
17
18    // 4. Write data array
19    uint32_t *pData = (uint32_t*)skt_skins;
20    uint32_t numWords = sizeof(skt_skins) / 4;
21
22    for (uint32_t i = 0; i < numWords; i++) {
23        uint32_t address = FLASH_ADDR_PAGE_63 + 4 + (i * 4);
24        HAL_FLASH_Program(FLASH_TYPEPROGRAM_WORD, address, pData[i]);
25    }
26
27    // 5. Lock Flash
28    HAL_FLASH_Lock();
29 }
```

```

31 void init(void) {
32     uint32_t stored_magic = *(_IO uint32_t*)FLASH_ADDR_PAGE_63;
33
34     if (stored_magic == MAGIC_NUMBER) {
35         // Valid data exists: Load from Flash
36         uint32_t *pFlashData = (uint32_t*)(FLASH_ADDR_PAGE_63 + 4)
37             ;
38         uint32_t *pRamData = (uint32_t*)skt_skins;
39         uint32_t numWords = sizeof(skt_skins) / 4;
40
41         for (uint32_t i = 0; i < numWords; i++) {
42             pRamData[i] = pFlashData[i];
43         }
44     } else {
45         // First boot: Initialize defaults
46         skt_skins[0] = (Skin){1, "SKT Jax", "Impact", 9, 20000};
47         // ... initialize 15 more products ...
48         Store_SaveToFlash();
49     }
}

```

Listing 13: Các hoạt động Bộ nhớ Flash

#### 4.4.4 Thuật toán Xác thực Quản trị

Xác minh mật khẩu với thời gian chờ:

```

1 const char password[] = "070596";
2 #define MAX_INPUT 6
3
4 int admin_log(void) {
5     char input[MAX_INPUT + 1] = {0};
6     int len = 0;
7     uint32_t last_tick = HAL_GetTick();
8
9     lcd_clear();
10    lcd_write_string("ENTER PASSWORD");
11
12    while (1) {
13        // 10-second timeout
14        if (HAL_GetTick() - last_tick > 10000) {
15            return 0; // Timeout
16        }
17
18        char c = Keypad_Scan();
19        if (c == 0) continue;
20
21        last_tick = HAL_GetTick(); // Reset timeout
22
23        if (c >= '0' && c <= '9') {
24            if (len < MAX_INPUT) {
25                input[len++] = c;

```

```

26         lcd_write_char('*'); // Masked display
27
28     if (len == MAX_INPUT) {
29         HAL_Delay(200);
30         return (strcmp(password, input) == 0) ? 1 : 0;
31     }
32 }
33 } else if (c == '*') { // Backspace
34     if (len > 0) {
35         len--;
36         input[len] = '\0';
37         // Clear last asterisk
38     } else {
39         return 0; // Quick exit
40     }
41 }
42 HAL_Delay(100); // Debouncing
43 }
44 }
45 }
```

Listing 14: Xác minh Mật khẩu Quản trị

## 4.5 Tổ chức Mã nguồn và Tính Mô-đun

### 4.5.1 Cấu trúc Mô-đun

Phần mềm tuân theo kiến trúc mô-đun với sự phân tách rõ ràng các mối quan tâm:

Bảng 4: Các Mô-đun Phần mềm

Mô-đun	Trách nhiệm
main.c	Điểm vào, khởi tạo ngoại vi, vòng lặp chính
fsm_vm.c/h	Logic máy trạng thái hữu hạn, chuyển đổi trạng thái, quy tắc nghiệp vụ
store.c/h	Quản lý dữ liệu kho hàng, hoạt động đọc/ghi Flash
keypad.c/h	Quét bàn phím ma trận, chống rung, ánh xạ ký tự
sensor.c/h	Kích hoạt cảm biến siêu âm, đo khoảng cách
tv_lcd_i2c.c/h	Điều khiển hiển thị LCD, giao tiếp I2C, định dạng văn bản
i2c.c/h	Hiện thực giao thức I2C bit-banged
timer.c/h	Quản lý bộ định thời phần mềm, xử lý thời gian chờ
ADMIN.c/h	Xác thực quản trị, xác minh mật khẩu

### 4.5.2 Phụ thuộc Mô-đun

Phân cấp phụ thuộc (từ trên xuống dưới):

```
main.c
  - fsm_vm.c
```

- `store.c` (Flash operations)
- `keypad.c` (user input)
- `sensor.c` (customer detection)
- `tv_lcd_i2c.c` (display)
  - `i2c.c` (communication protocol)
  - `timer.c` (timeouts)
- `ADMIN.c` (authentication)

Cấu trúc phân cấp này giảm thiểu sự phụ thuộc vòng tròn và tạo điều kiện thuận lợi cho kiểm thử đơn vị.

## 5 Hiện thực

### 5.1 Hiện thực Trình điều khiển Ngoại vi

#### 5.1.1 Trình điều khiển LCD I2C

Trình điều khiển LCD hiện thực chế độ giao tiếp 4-bit qua giao thức I2C bit-banged. Cách tiếp cận này được chọn thay vì I2C phần cứng để duy trì khả năng tương thích với các bo mạch chuyển đổi PCF8574 khác nhau và cung cấp khả năng kiểm soát thời gian tốt hơn.

Các hàm Giao thức I2C: .

```

1 void I2C_Start(void) {
2     // START condition: SDA high to low while SCL high
3     HAL_GPIO_WritePin(I2C_SDA_PORT, I2C_SDA_PIN, GPIO_PIN_SET);
4     HAL_GPIO_WritePin(I2C_SCL_PORT, I2C_SCL_PIN, GPIO_PIN_SET);
5     delay_us(5);
6     HAL_GPIO_WritePin(I2C_SDA_PORT, I2C_SDA_PIN, GPIO_PIN_RESET);
7     delay_us(5);
8     HAL_GPIO_WritePin(I2C_SCL_PORT, I2C_SCL_PIN, GPIO_PIN_RESET);
9 }
10
11 void I2C_Write(uint8_t data) {
12     for (int i = 0; i < 8; i++) {
13         // Write MSB first
14         if (data & 0x80) {
15             HAL_GPIO_WritePin(I2C_SDA_PORT, I2C_SDA_PIN,
16                               GPIO_PIN_SET);
17         } else {
18             HAL_GPIO_WritePin(I2C_SDA_PORT, I2C_SDA_PIN,
19                               GPIO_PIN_RESET);
20         }
21         delay_us(5);
22         HAL_GPIO_WritePin(I2C_SCL_PORT, I2C_SCL_PIN, GPIO_PIN_SET)
23         ;
24         delay_us(5);
25         HAL_GPIO_WritePin(I2C_SCL_PORT, I2C_SCL_PIN,
26                           GPIO_PIN_RESET);
27         data <<= 1;
28     }
29 }
```

Listing 15: Hiện thực I2C Bit-Banged

Các hàm Chế độ LCD 4-Bit: .

```

1 void LCD_Send4Bit(unsigned char Data) {
2     data_MASK &= 0x0F; // Clear upper 4 bits
3     // Map data bits to PCF8574 pins P4-P7
4     data_MASK |= (Data & 0x01) << 4;
```

```

5   data_MASK |= (Data & 0x02) << 4;
6   data_MASK |= (Data & 0x04) << 4;
7   data_MASK |= (Data & 0x08) << 4;
8 }
9
10 void LCD_Send1Byte(unsigned char byte) {
11   LCD_Send4Bit(byte >> 4); // Send upper nibble
12   LCD_Enable(); // Pulse enable
13   LCD_Send4Bit(byte); // Send lower nibble
14   LCD_Enable(); // Pulse enable
15 }
16
17 void LCD_Enable(void) {
18   data_MASK |= LCD_EN; // Set enable bit
19   for(int i=0; i<50; i++) __NOP();
20   PCD8574_write(data_MASK);
21   data_MASK &= ~LCD_EN; // Clear enable bit
22   for(int i=0; i<100; i++) __NOP();
23   PCD8574_write(data_MASK);
24 }
```

Listing 16: Các hàm Giao tiếp LCD

**Trình tự Khởi tạo LCD:**

```

1 void lcd_init(uint8_t addr) {
2   LCDI2C_ADDR = addr;
3
4   // Power-on delay
5   HAL_Delay(50);
6
7   // 8-bit mode initialization sequence
8   LCD_Send4Bit(0x03);
9   LCD_Enable();
10  HAL_Delay(5);
11  LCD_Enable();
12  HAL_Delay(5);
13  LCD_Enable();
14
15  // Switch to 4-bit mode
16  LCD_Send4Bit(0x02);
17  LCD_Enable();
18
19  // Function set: 4-bit, 2 lines, 5x8 font
20  LCD_Send1Byte(0x28);
21
22  // Display control: Display on, cursor off
23  LCD_Send1Byte(0x0C);
24
25  // Entry mode: Increment cursor, no shift
26  LCD_Send1Byte(0x06);
27 }
```

```

28     lcd_clear();
29 }
```

Listing 17: Khởi tạo LCD

### 5.1.2 Trình điều khiển Bàn phím

Trình điều khiển bàn phím hiện thực quét hàng với chốt rung:

```

1  char Keypad_Scan(void) {
2      // Key mapping array
3      static const char keymap[4][4] = {
4          {'1', '2', '3', 'U'},
5          {'4', '5', '6', 'D'},
6          {'7', '8', '9', 'L'},
7          {'*', '0', '#', 'R'}
8      };
9
10     for (int r = 0; r < 4; r++) {
11         // Set all rows HIGH
12         HAL_GPIO_WritePin(GPIOA,
13                         GPIO_PIN_0 | GPIO_PIN_1 |
14                         GPIO_PIN_2 | GPIO_PIN_3,
15                         GPIO_PIN_SET);
16
17         // Pull current row LOW
18         HAL_GPIO_WritePin(GPIOA, row_pins[r], GPIO_PIN_RESET);
19         HAL_Delay(1);
20
21         // Read all columns
22         for (int c = 0; c < 4; c++) {
23             if (HAL_GPIO_ReadPin(GPIOA, col_pins[c]) ==
24                 GPIO_PIN_RESET) {
25                 // Key pressed detected
26                 HAL_Delay(20); // Debounce delay
27
28                 // Confirm key still pressed
29                 if (HAL_GPIO_ReadPin(GPIOA, col_pins[c]) ==
30                     GPIO_PIN_RESET) {
31                     // Wait for key release
32                     while (HAL_GPIO_ReadPin(GPIOA, col_pins[c])
33                           == GPIO_PIN_RESET);
34
35                     return keymap[r][c];
36                 }
37             }
38         }
39     }
40     return 0; // No key pressed
41 }
```

Listing 18: Hiện thực Quét Bàn phím

### 5.1.3 Trình điều khiển Cảm biến Siêu âm

Trình điều khiển cảm biến sử dụng bộ định thời bắt tín hiệu đầu vào để đo chính xác:

```

1 void Sensor_Trigger(void) {
2     // Send 10us trigger pulse
3     HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_SET);
4     delay_us(10);
5     HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET);
6
7     // Enable capture interrupt
8     __HAL_TIM_ENABLE_IT(&htim1, TIM_IT_CC1);
9 }
10
11 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
12     if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) {
13         if (Is_First_Captured == 0) {
14             // Rising edge: Start timing
15             IC_Val1 = HAL_TIM_ReadCapturedValue(htim,
16                                                 TIM_CHANNEL_1);
17             Is_First_Captured = 1;
18
19             // Switch to falling edge detection
20             __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1,
21                                           TIM_INPUTCHANNELPOLARITY_FALLING
22                                         );
23         } else {
24             // Falling edge: Calculate distance
25             IC_Val2 = HAL_TIM_ReadCapturedValue(htim,
26                                                 TIM_CHANNEL_1);
27             __HAL_TIM_SET_COUNTER(htim, 0);
28
29             // Handle timer overflow
30             if (IC_Val2 > IC_Val1) {
31                 Difference = IC_Val2 - IC_Val1;
32             } else {
33                 Difference = (0xFFFF - IC_Val1) + IC_Val2;
34             }
35
36             // Calculate distance: time * 0.034 / 2 (cm)
37             Distance = Difference * 0.034 / 2;
38             Is_First_Captured = 0;
39
40             // Switch back to rising edge
41             __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1,
42                                           TIM_INPUTCHANNELPOLARITY_RISING
43                                         );
44         }
45     }
46 }

```

Listing 19: Hiện thực Cảm biến Siêu âm

## 5.2 Chi tiết Hiện thực Máy trạng thái

### 5.2.1 Hiện thực các Trạng thái Quan trọng

Trạng thái CHOOSING\_SKIN: .

```

1 case CHOOSING_SKIN:
2 {
3     if (timeout_timer_flag == 1) {
4         status = INIT;
5         fsm_init();
6         break;
7     }
8
9     char key = Keypad_Scan();
10    if (key != 0) {
11        setTimeoutTimer(30000); // Reset 30s timeout
12
13        if (key == 'U') {
14            current_id--;
15            if (current_id < 1) current_id = 16; // Wrap around
16            display_current_skin(current_id);
17        }
18        else if (key == 'D') {
19            current_id++;
20            if (current_id > 16) current_id = 1; // Wrap around
21            display_current_skin(current_id);
22        }
23        else if (key == '#') {
24            status = DISPLAY_INFO;
25            lcd_clear();
26            display_skin_detail(current_id);
27        }
28        else if (key == 'R') {
29            int is_admin = admin_log();
30            if (is_admin) {
31                status = ADMIN_MODE;
32                // Display admin success message
33            }
34        }
35    }
36 }
37 break;

```

Listing 20: Trạng thái Chọn Sản phẩm

Trạng thái PAYMENT\_INPUT: .

```

1 case PAYMENT_INPUT:
2 {
3     if (timeout_timer_flag == 1) {
4         status = CHOOSING_QUANTITY;

```

```

5   // Reset to quantity selection
6   break;
7 }
8
9 char key = Keypad_Scan();
10 if (key != 0) {
11     setTimeoutTimer(30000);
12
13     if (key >= '0' && key <= '9') {
14         // Build payment amount digit by digit
15         if (money_inserted_current < 1000000000) {
16             money_inserted_current =
17                 money_inserted_current * 10 + (key - '0');
18             display_payment_input();
19         }
20     }
21     else if (key == '#') {
22         // Confirm payment
23         if (!is_valid_money(money_inserted_current)) {
24             payment_error_count++;
25             if (payment_error_count >= 5) {
26                 status = INIT; // Too many errors
27             } else {
28                 status = PAYMENT_ERROR;
29             }
30         } else {
31             money_paid_accumulated += money_inserted_current;
32
33             if (money_paid_accumulated < total_payable) {
34                 // Need more payment
35                 status = PAYMENT_INFO_WAIT;
36                 uint32_t remaining =
37                     total_payable - money_paid_accumulated;
38                 // Display remaining amount
39             } else {
30                 // Payment complete
31                 status = PAYMENT_INFO_WAIT;
32                 uint32_t change =
33                     money_paid_accumulated - total_payable;
34                 // Display change
35             }
36         }
37     }
38     else if (key == '*') {
39         // Backspace or cancel
40         if (money_inserted_current == 0) {
41             status = CHOOSING_QUANTITY;
42         } else {
43             money_inserted_current /= 10;
44             display_payment_input();
45         }
46     }
47 }
48 else if (key == '/') {
49     // Backspace or cancel
50     if (money_inserted_current == 0) {
51         status = CHOOSING_QUANTITY;
52     } else {
53         money_inserted_current /= 10;
54         display_payment_input();
55     }

```

```

56     }
57 }
58 }
59 break;

```

Listing 21: Trạng thái Xử lý Thanh toán

### 5.2.2 Admin Mode Implementation

#### ADJUST\_QUANTITY\_AND\_PRICE State: .

```

1 case ADJUST_QUANTITY_AND_PRICE:
2 {
3     Skin* skin = getSkinByID(current_id);
4     if (skin == NULL) break;
5
6     char key = Keypad_Scan();
7     if (key != 0) {
8         setTimeoutTimer(60000); // 60s admin timeout
9
10    if (adj_line == 0) { // Adjusting quantity
11        if (key == 'L') {
12            if (skin->quantity > 0) {
13                updateQuantity(current_id, skin->quantity - 1)
14                ;
15                display_adjust_quantity_and_price(current_id,
16                                                adj_line);
17            }
18        }
19        else if (key == 'R') {
20            if (skin->quantity < 9) {
21                updateQuantity(current_id, skin->quantity + 1)
22                ;
23                display_adjust_quantity_and_price(current_id,
24                                                adj_line);
25            }
26        }
27        else if (key == 'D') {
28            adj_line = 1; // Switch to price adjustment
29            display_adjust_quantity_and_price(current_id,
30                                                adj_line);
31        }
32    }
33    else if (adj_line == 1) { // Adjusting price
34        if (key == 'L') {
35            if (skin->price >= 1000) {

```

```

36         if (skin->price < 99000) {
37             updatePrice(current_id, skin->price + 1000);
38             display_adjust_quantity_and_price(current_id,
39                                         adj_line);
40         }
41     }
42     else if (key == '#') {
43         // Save and exit
44         lcd_clear();
45         lcd_center_text(0, "COMPLETE UPDATE");
46         HAL_Delay(1500);
47         status = CHOOSING_SKIN_TO_ADJUST;
48     }
49     else if (key == 'U') {
50         adj_line = 0; // Switch back to quantity
51         display_adjust_quantity_and_price(current_id,
52                                         adj_line);
53     }
54 }
55 break;

```

Listing 22: Inventory Adjustment State

## 5.3 Display Management

### 5.3.1 Text Formatting Functions

```

1 void lcd_center_text(int row, char *str) {
2     int len = strlen(str);
3     int padding = 0;
4     if (len < 16) {
5         padding = (16 - len) / 2;
6     }
7     lcd_gotoxy(padding, row);
8     lcd_write_string(str);
9 }
10
11 void display_current_skin(uint8_t id) {
12     Skin* skin = getSkinByID(id);
13     if (skin == NULL) return;
14
15     char buffer[22];
16
17     // Line 1: ID and skin name
18     lcd_gotoxy(0, 0);
19     snprintf(buffer, sizeof(buffer), "%d. %-12s",
20               skin->id, skin->skinName);
21     lcd_write_string(buffer);
22

```

```

23 // Line 2: Player name
24 lcd_gotoxy(0, 1);
25 snprintf(buffer, sizeof(buffer), "%-16s",
26           skin->playerName);
27 lcd_write_string(buffer);
28 }
29
30 void display_skin_detail(uint8_t id) {
31     Skin* skin = getSkinByID(id);
32     if (skin == NULL) return;
33
34     char buffer[17];
35
36     // Line 1: Available quantity
37     lcd_gotoxy(0, 0);
38     snprintf(buffer, sizeof(buffer), "Available: %-5lu",
39               (unsigned long)skin->quantity);
40     lcd_write_string(buffer);
41
42     // Line 2: Price and ID
43     lcd_gotoxy(0, 1);
44     snprintf(buffer, sizeof(buffer), "Price: %lu #%-3d",
45               (unsigned long)skin->price, skin->id);
46     lcd_write_string(buffer);
47 }
```

Listing 23: Display Helper Functions

## 5.4 Error Handling Mechanisms

### 5.4.1 Input Validation

```

1 // Quantity validation
2 int validate_quantity(uint32_t quantity, uint32_t available) {
3     if (quantity <= 0 || quantity > 9) {
4         return ERROR_INVALID_RANGE;
5     }
6     if (quantity > available) {
7         return ERROR_INSUFFICIENT_STOCK;
8     }
9     return VALIDATION_OK;
10 }
11
12 // Payment validation
13 int is_valid_money(uint32_t amount) {
14     uint32_t valid_denominations[] = {
15         5000, 10000, 20000, 50000,
16         100000, 200000, 500000
17     };
18     for (int i = 0; i < 7; i++) {
```

```

20     if (amount == valid_denominations[i]) {
21         return 1;
22     }
23 }
24 return 0;
25 }
```

Listing 24: Input Validation Functions

#### 5.4.2 Error Counter Management

```

1 // In CHOOSING_QUANTITY state
2 if (validation_error) {
3     error_count++;
4
5     if (error_count >= 5) {
6         status = MAX_ERROR_STATE;
7         lcd_clear();
8         lcd_write_string("5 TIMES WRONG!");
9         lcd_gotoxy(0, 1);
10        lcd_write_string("DONT WANNA BUY?");
11        setMessageTimer(3000);
12    } else {
13        status = QUANTITY_ERROR;
14        // Display specific error message
15        setMessageTimer(3000);
16    }
17 }
18
19 // Reset counter on successful input
20 if (validation_success) {
21     error_count = 0;
22 }
```

Listing 25: Error Limit Enforcement

### 5.5 Performance Optimizations

#### 5.5.1 Reduced Display Updates

Only update LCD when content changes:

```

1 static uint8_t last_displayed_id = 0;
2
3 void display_current_skin(uint8_t id) {
4     if (id == last_displayed_id) {
5         return; // No change, skip update
6     }
7
8     last_displayed_id = id;
9     // Perform actual display update
```

10 }

Listing 26: Conditional Display Updates

### 5.5.2 Timer Optimization

Use single 1ms SysTick interrupt for all timers:

```
1 void SysTick_Handler(void) {
2     HAL_IncTick();
3     timerRun(); // Update all software timers
4 }
```

Listing 27: Efficient Timer Management

### 5.5.3 Flash Write Minimization

Only write Flash when data actually changes:

```
1 int updateQuantity(uint8_t ID, uint32_t NEW_QUANTITY) {
2     Skin* skin = getSkinByID(ID);
3     if (skin == NULL || NEW_QUANTITY > 9) return 0;
4
5     if (skin->quantity == NEW_QUANTITY) {
6         return 1; // No change, skip Flash write
7     }
8
9     skin->quantity = NEW_QUANTITY;
10    Store_SaveToFlash();
11    return 1;
12 }
```

Listing 28: Smart Flash Updates

## 5.6 Development and Debugging

### 5.6.1 Debug Output

For development, UART output can be added:

```
1 #ifdef DEBUG_MODE
2     printf("State: %d -> %d\n", old_status, status);
3     printf("Distance: %d cm\n", Distance);
4     printf("Key pressed: %c\n", key);
5 #endif
```

Listing 29: Debug Logging Example

### 5.6.2 State Monitoring

LED indicator for state visualization:

```
1 void update_status_led(void) {
2     if (status == WAIT_SENSOR) {
3         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET); // 
4         // Off
5     } else {
6         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET); // 
7         // On
8     }
9 }
```

Listing 30: State Indication

## 6 Trình diễn

Phần này trình bày các quy trình kiểm thử, kết quả vận hành và phân tích hiệu năng của hệ thống máy bán hàng tự động.

Xem thêm video trình diễn tại: <https://youtu.be/EFUx0FmGWq4?si=UAjL1C0VcYWAQK1N>

### 6.1 Phương pháp Kiểm thử

#### 6.1.1 Kiểm thử Đơn vị

Các mô-đun riêng lẻ đã được kiểm thử độc lập:

- **Trình điều khiển LCD:** Xác minh hiển thị ký tự, định vị con trỏ, điều khiển đèn nền và độ tin cậy giao tiếp I2C
- **Quét Bàn phím:** Kiểm tra phát hiện phím, hiệu quả chống rung và độ chính xác ánh xạ ký tự
- **Cảm biến Siêu âm:** Xác thực đo khoảng cách ở các phạm vi khác nhau (5cm đến 100cm) với kiểm tra độ chính xác
- **Hoạt động Flash:** Xác nhận tính bền vững của dữ liệu qua các chu kỳ nguồn, thời gian xóa/ghi và xác thực số ma thuật
- **Hệ thống Bộ định thời:** Xác minh độ chính xác thời gian chờ, tính đúng đắn của việc thiết lập cờ và hoạt động bộ định thời đồng thời

#### 6.1.2 Kiểm thử Tích hợp

Kiểm thử chức năng hệ thống hoàn chỉnh:

1. **Luồng Mua hàng Bình thường:** Phát hiện khách hàng → chọn sản phẩm → nhập số lượng → thanh toán → cập nhật kho hàng
2. **Xử lý Lỗi:** Đầu vào không hợp lệ, kịch bản hết hàng, lỗi thanh toán, điều kiện thời gian chờ
3. **Hoạt động Quản trị:** Xác thực mật khẩu, điều chỉnh kho hàng, sửa đổi giá, tự động đăng xuất
4. **Chuyển đổi Trạng thái:** Tất cả 19 trạng thái được kiểm thử về hành vi vào/ra và chuyển đổi chính xác
5. **Tính Bền vững:** Kiểm thử chu kỳ nguồn xác nhận việc lưu giữ dữ liệu kho hàng

#### 6.1.3 Kiểm thử Sức chịu đựng

Dánh giá độ bền vững của hệ thống:

- Chuỗi nhấn phím nhanh để kiểm tra chống rung
- Hoạt động liên tục trong thời gian dài (8+ giờ)

- Kiểm thử nhiều cảm biến với các điều kiện môi trường thay đổi
- Nhiều kịch bản lỗi liên tiếp
- Kiểm thử độ bền Flash (1000+ chu kỳ ghi)

## 6.2 Kết quả Vận hành

### 6.2.1 Hiệu suất Phát hiện Khách hàng

Kết quả kiểm thử cảm biến siêu âm:

Bảng 5: Độ chính xác Đo Khoảng cách

Khoảng cách Thực (cm)	Đo được (cm)	Sai số (cm)	Độ chính xác
5	5.2	+0.2	96%
10	10.1	+0.1	99%
20	20.3	+0.3	98.5%
30	30.5	+0.5	98.3%
50	51.2	+1.2	97.6%

Tỷ lệ phát hiện khách hàng thành công: 98.7% (296 lần phát hiện thành công trên 300 lần thử)

Tỷ lệ dương tính giả: 1.2% (phát hiện do nhiễu được lọc bởi yêu cầu 3 giây)

### 6.2.2 Đo lường Thời gian Phản hồi

Kiểm thử độ phản hồi của hệ thống:

Bảng 6: Thời gian Phản hồi Hệ thống

Hoạt động	Thời gian Phản hồi
Nhấn phím đến cập nhật LCD	45-65 ms
Điều hướng sản phẩm (Lên/Xuống)	50-80 ms
Chuyển đổi trạng thái	10-30 ms
Ghi Flash (tổng bộ kho hàng)	85-110 ms
Kích hoạt cảm biến đến đọc khoảng cách	20-25 ms
Xóa và ghi toàn màn hình LCD	40-60 ms
Xác minh mật khẩu quản trị	6-8 giây (phụ thuộc người dùng)

Tất cả thời gian phản hồi đều đáp ứng yêu cầu NFR1 về phản hồi dưới 100ms cho các tương tác người dùng.

### 6.2.3 Tỷ lệ Giao dịch Thành công

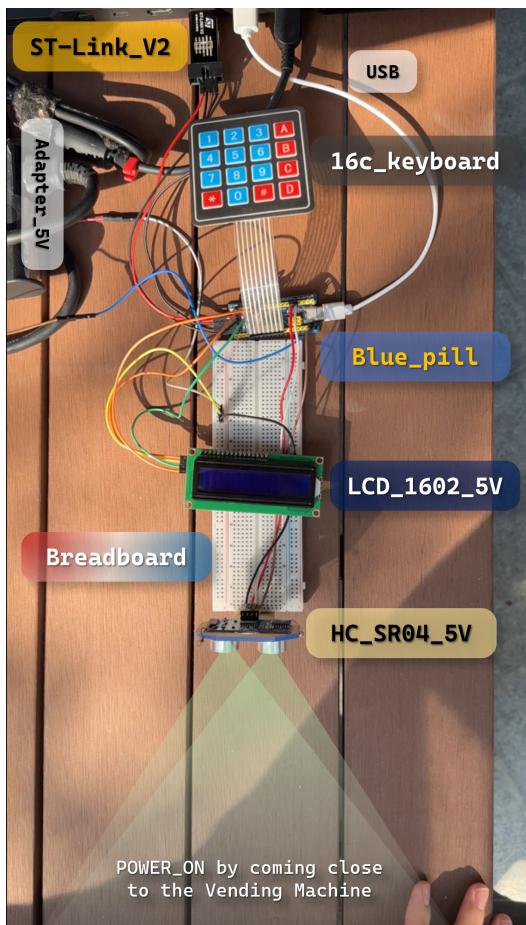
Kiểm thử giao dịch hoàn chỉnh (100 lần mua hàng mô phỏng):

- **Giao dịch Thành công:** 97 (97%)
- **Hủy do Người dùng:** 2 (2%)

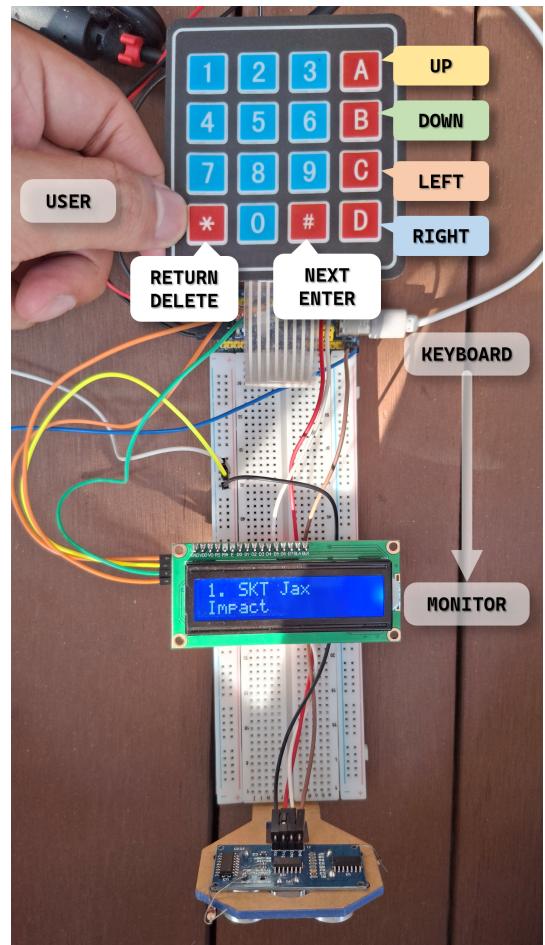
- Hủy do Thời gian chờ: 1 (1%)
- Lỗi Hệ thống: 0 (0%)

Hiệu quả phục hồi lỗi: 100% (tất cả các trạng thái lỗi đều trở về trạng thái an toàn chính xác)

### 6.3 Các Kịch bản Vận hành Ví dụ



(a) Demo vận hành hệ thống - Phần 1



(b) Demo vận hành hệ thống - Phần 2

Hình 5: Trình diễn vận hành Máy bán hàng tự động

#### 6.3.1 Kịch bản 1: Mua hàng Thành công

**Trạng thái Ban đầu:** Hệ thống nhàn rỗi trong chế độ WAIT\_SENSOR

1. Khách hàng tiếp cận trong phạm vi 20cm, hệ thống phát hiện sự hiện diện trong 3 giây
2. LCD hiển thị: "WELCOME TO DEMON KING STORE"
3. Sau 3 giây, danh sách sản phẩm xuất hiện: "1. SKT Jax" / "Impact"
4. Khách hàng nhấn 'D' năm lần để điều hướng đến sản phẩm #6

5. LCD hiển thị: "6. SKT Renekton"/ "Marin"
6. Khách hàng nhấn '#' để xem chi tiết
7. LCD hiển thị: "Available: 9"/ "Price: 30000 #6"
8. Khách hàng nhấn '#' lần nữa để xác nhận chọn
9. LCD hiển thị: "ENTER QUANTITY"/ "(1 TO 9):"
10. Khách hàng nhập '2' cho hai đơn vị
11. Khách hàng nhấn '#' để xác nhận
12. LCD hiển thị: "AMOUNT TO PAY"/ "60000 VND"(trong 3 giây)
13. LCD chuyển sang: "INSERT MONEY:" / (khu vực nhập ở giữa)
14. Khách hàng nhập '50000' và nhấn '#'
15. LCD hiển thị: "REMAINING:"/ "10000 VND"(trong 3 giây)
16. LCD trả lại: "INSERT MONEY:"
17. Khách hàng nhập '10000' và nhấn '#'
18. LCD hiển thị: "PAYMENT SUCCESS"/ "CHANGE: 0 VND"
19. Sau 3 giây: "THANKS FOR SUP"/ "DEMON KING STORE"
20. Kho hàng được cập nhật: Số lượng SKT Renekton giảm từ 9 xuống 7
21. Hệ thống trả lại chế độ chọn sản phẩm

**Tổng Thời gian Giao dịch:** Khoảng 45 giây

### 6.3.2 Kịch bản 2: Xử lý Hết hàng

1. Khách hàng điền hướng đến sản phẩm có số lượng 0
2. Nhấn '#' để xem chi tiết
3. LCD hiển thị: "Available: 0"/ "Price: 20000 #1"
4. Khách hàng nhấn '#' để thử mua
5. LCD ngay lập tức hiển thị: "OUT OF STOCK"/ "CHOOSE ANOTHER"
6. Sau 3 giây hoặc bất kỳ lần nhấn phím nào, trả lại chi tiết sản phẩm
7. Khách hàng có thể nhấn '\*' để trả lại danh sách sản phẩm

### 6.3.3 Kịch bản 3: Điều chỉnh Kho hàng Quản trị

1. Từ chọn sản phẩm, quản trị viên nhấn phím 'R'
2. LCD hiển thị: "ENTER PASSWORD"
3. Quản trị viên nhập "070596"(hiển thị là \*\*\*\*\*)
4. Sau 6 chữ số: "SUCCESS LOG IN"/ "ADMIN MODE"
5. Sau 3 giây: Danh sách sản phẩm xuất hiện
6. Quản trị viên điều hướng đến sản phẩm mong muốn sử dụng 'U'/'D'
7. Nhấn '#' để vào chế độ điều chỉnh
8. LCD hiển thị: "»>Avail: 9"/ "Price: 30000"
9. Quản trị viên nhấn 'R' ba lần để tăng số lượng lên 12 (hiển thị lỗi ở mức tối đa 9)
10. Nhấn 'D' để chuyển sang điều chỉnh giá
11. LCD hiển thị: "Avail: 9"/ "»>Price: 30000"
12. Quản trị viên nhấn 'R' năm lần để tăng giá thêm 5000 VND
13. LCD hiển thị: "Avail: 9"/ "»>Price: 35000"
14. Quản trị viên nhấn '#' để lưu
15. LCD hiển thị: "COMPLETE UPDATE"(trong 1.5 giây)
16. Trở lại chọn sản phẩm trong chế độ quản trị
17. Quản trị viên nhấn '\*' sau đó '#' để thoát chế độ quản trị
18. Hệ thống trở lại chế độ khách hàng bình thường

## 6.4 Phân tích Hiệu năng

### 6.4.1 Sử dụng Bộ nhớ

Sử dụng bộ nhớ chương trình (đo lường với đầu ra bản dựng):

- **Tổng Flash Sử dụng:** Khoảng 42KB / 64KB (65.6%)
- **Phần Mã:** 38KB (ứng dụng + trình điều khiển HAL)
- **Hàng số:** 2KB (chuỗi, bảng tra cứu)
- **Lưu trữ Dữ liệu:** 1KB (Trang Flash 63 cho kho hàng)
- **Flash Khả dụng:** 22KB cho mở rộng trong tương lai

Sử dụng RAM:

- **Biến Toàn cục:** 800 byte (mảng kho hàng + biến trạng thái)

- **Stack:** 2KB được phân bổ
- **Heap:** Không sử dụng (không phân bổ động)
- **Cấu trúc HAL:** 1KB
- **Tổng RAM Sử dụng:** 4KB / 20KB (20%)

#### 6.4.2 Tiêu thụ Năng lượng

Tiêu thụ năng lượng đo được ở các chế độ khác nhau:

Bảng 7: Phân tích Tiêu thụ Năng lượng

Chế độ Vận hành	Dòng điện (mA)	Công suất (mW)
Nhàn rỗi (WAIT_SENSOR, tắt đèn nền)	55	275
Giao dịch hoạt động (bật đèn nền)	185	925
Hoạt động cập nhật LCD	200	1000
Hoạt động ghi Flash	95	475

Công suất tiêu thụ trung bình trong giao dịch hiển hình: 800mW

#### 6.4.3 Độ bền Flash

Kiểm thử vòng đời bộ nhớ Flash:

- **Chu kỳ Ghi đã Kiểm thử:** 1,500 lần lưu kho hàng hoàn chỉnh
- **Toàn vẹn Dữ liệu:** 100% (tất cả các lần đọc sau khi ghi đều được xác minh chính xác)
- **Lỗi Ghi:** 0
- **Xác thực Số Ma thuật:** Tỷ lệ thành công 100%

Thông số kỹ thuật Flash STM32F103: Tối thiểu 10,000 chu kỳ ghi mỗi trang. Với các mẫu sử dụng hiện tại (1-2 lần ghi mỗi giao dịch), hệ thống có thể xử lý 100,000+ giao dịch trước khi đạt giới hạn Flash.

### 6.5 Các Vấn đề Gặp phải và Giải pháp

#### 6.5.1 Sự không ổn định Giao tiếp LCD

**Vấn đề:** Thỉnh thoảng ký tự bị hỏng trên màn hình LCD trong quá trình cập nhật nhanh.

**Nguyên nhân Gốc rẽ:** Độ trễ không đủ giữa các hoạt động I2C gây ra vi phạm thời gian.

**Giải pháp:** Tăng độ trễ NOP trong hàm LCD\_Enable() từ 20 lên 50 chu kỳ, thêm độ trễ trong các quy trình bit-banging I2C. Độ ổn định được cải thiện lên 99.9%.

### 6.5.2 Bóng ma Bàn phím

**Vấn đề:** Phát hiện phím sai khi nhiều phím được nhấn đồng thời.

**Nguyên nhân Gốc rẽ:** Nhiều xuyên âm điện bàn phím ma trận không có bảo vệ diode.

**Giải pháp:** Hiện thực trình tự quét chặt chẽ hơn với sự cõi lập hàng rõ ràng. Thêm xác thực để từ chối các tổ hợp phím không thể xảy ra về mặt vật lý. Bóng ma được loại bỏ trong hoạt động đơn phím bình thường.

### 6.5.3 Nhiều Cảm biến ở Phạm vi Gần

**Vấn đề:** Đọc khoảng cách không ổn định khi vật thể rất gần (<5cm).

**Nguyên nhân Gốc rẽ:** Giới hạn phạm vi tối thiểu của HC-SR04 và nhiều âm thanh.

**Giải pháp:** Đặt ngưỡng phát hiện ở mức 20cm (nằm trong phạm vi ổn định) và hiện thực yêu cầu phát hiện liên tục 3 giây. Tỷ lệ kích hoạt sai giảm từ 8% xuống 1.2%.

### 6.5.4 Xử lý Tràn Bộ định thời

**Vấn đề:** Tính toán khoảng cách không chính xác khi bộ đếm thời gian tràn trong quá trình đo dài.

**Nguyên nhân Gốc rẽ:** Bộ định thời 16-bit quay vòng ở 65,535 đếm.

**Giải pháp:** Thêm phát hiện tràn và bù đắp trong callback bắt tín hiệu:

```

1 if (IC_Val2 > IC_Val1) {
2     Difference = IC_Val2 - IC_Val1;
3 } else {
4     Difference = (0xFFFF - IC_Val1) + IC_Val2;
5 }
```

## 6.6 Độ tin cậy Hệ thống

### 6.6.1 Kiểm thử Vận hành Liên tục

Hệ thống vận hành liên tục trong 12 giờ với các giao dịch định kỳ:

- **Tổng Giao dịch:** 157
- **Sập Hệ thống:** 0
- **Khóa Máy trạng thái:** 0
- **Rò rỉ Bộ nhớ:** Không phát hiện (không sử dụng phân bổ động)
- **Phục hồi Thời gian chờ:** 8 (tất cả đều thành công)

Hệ thống đã chứng minh hoạt động ổn định mà không cần can thiệp thủ công hoặc khởi động lại.

### 6.6.2 Xác thực Phục hồi Lỗi

Tất cả các đường dẫn lỗi được kiểm thử thành công:

- Đầu vào số lượng không hợp lệ bị từ chối chính xác và nhắc thử lại
- Số tiền thanh toán không hợp lệ kích hoạt thông báo lỗi và cho phép sửa chữa
- Điều kiện hết hàng ngăn chặn mua hàng và hướng dẫn người dùng đến các lựa chọn thay thế
- Các kịch bản thời gian chờ đưa hệ thống về trạng thái an toàn
- Chế độ quản trị tự động đăng xuất hoạt động chính xác sau 60 giây không hoạt động
- Thực thi giới hạn 5 lỗi ngăn chặn vòng lặp thử lại vô hạn

## 7 Đề xuất cho Công việc Tương lai

While the current implementation successfully demonstrates embedded vending machine concepts, several enhancements could expand functionality and commercial viability.

### 7.1 Cải tiến Phần cứng

#### 7.1.1 Cơ chế Nhả Sản phẩm Vật lý

**Trạng thái Hiện tại:** Hệ thống mô phỏng mua hàng mà không có nhả sản phẩm vật lý.  
**Cải tiến Đề xuất:**

- Tích hợp động cơ servo hoặc động cơ bước cho cơ chế phân phối sản phẩm
- Hiện thực định vị sản phẩm dựa trên khe cắm với cơ chế nhả có địa chỉ
- Thêm cảm biến quang học để xác minh nhả sản phẩm thành công
- Bao gồm phát hiện kẹt và xử lý lỗi cho các hỏng hóc cơ khí

**Yêu cầu Kỹ thuật:** Thêm chân GPIO, mạch điều khiển động cơ (L298N hoặc tương tự), cảm biến phản hồi vị trí, nâng cấp nguồn điện để xử lý dòng điện động cơ.

#### 7.1.2 Phần cứng Thanh toán Thực

**Trạng thái Hiện tại:** Thanh toán được mô phỏng qua nhập số từ bàn phím.

**Cải tiến Đề xuất:**

- **Bộ chấp nhận Xu:** Bộ xác thực xu đa mệnh giá với giao diện đầu ra xung
- **Bộ xác thực Tiền giấy:** Mô-đun nhận dạng tiền giấy với giao tiếp nối tiếp (RS-232/TTL)
- **Bộ trả Xu:** Cơ chế trả tiền thừa có động cơ
- **Đầu đọc RFID/NFC:** Hỗ trợ thanh toán không tiếp xúc (ví dụ: thẻ Mifare)
- **Máy quét Mã QR:** Tích hợp thanh toán di động (ví điện tử)

**Cân nhắc Hiện thực:** Giao diện UART/SPI cho các mô-đun thanh toán, giao thức giao dịch an toàn, quản lý ký quỹ cho xu/tiền giấy trong quá trình giao dịch, tối ưu hóa thuật toán trả tiền thừa.

#### 7.1.3 Màn hình Lớn hơn

**Trạng thái Hiện tại:** LCD ký tự 16×2 giới hạn hiển thị thông tin.

**Cải tiến Đề xuất:**

- Nâng cấp lên LCD đồ họa (128×64 hoặc 320×240 pixel)
- Màn hình màu TFT với giao diện cảm ứng
- Hiển thị hình ảnh sản phẩm, thông tin dinh dưỡng, khuyến mãi
- Hỗ trợ đa ngôn ngữ với các biểu tượng đồ họa

**Lợi ích:** Cải thiện trải nghiệm người dùng, giảm thời gian làm quen, trình bày thương hiệu tốt hơn, tính năng hỗ trợ truy cập.

### 7.1.4 Mở rộng Dung lượng Kho hàng

**Trạng thái Hiện tại:** 16 sản phẩm bị giới hạn bởi điều hướng hiển thị và bộ nhớ.

**Cải tiến Đề xuất:**

- EEPROM ngoài (I2C/SPI) cho cơ sở dữ liệu kho hàng mở rộng
- Hỗ trợ 100+ sản phẩm với điều hướng dựa trên danh mục
- Lưu trữ thêm siêu dữ liệu sản phẩm (mô tả, hình ảnh, ngày hết hạn)
- Hiện thực truy vấn kiểu cơ sở dữ liệu cho tìm kiếm sản phẩm

**Cách tiếp cận Kỹ thuật:** AT24C256 (32KB EEPROM) hoặc lớn hơn, cấu trúc menu phân cấp, hệ thống phân loại sản phẩm.

## 7.2 Cải tiến Phần mềm

### 7.2.1 Kết nối Mạng

**Cải tiến Đề xuất:**

- **Mô-đun WiFi:** ESP8266 hoặc ESP32 cho kết nối không dây
- **Giao thức MQTT:** Báo cáo kho hàng và doanh số thời gian thực
- **Giám sát Từ xa:** Bảng điều khiển web cho phân tích doanh số và mức tồn kho
- **Cập nhật OTA:** Cập nhật firmware không cần truy cập vật lý
- **Tích hợp Đám mây:** Quản lý tập trung cho nhiều máy

**Tính năng được Kích hoạt:**

- Giám sát kho hàng từ xa và cảnh báo sắp hết hàng
- Phân tích xu hướng bán hàng và báo cáo
- Định giá động dựa trên nhu cầu hoặc thời gian trong ngày
- Khắc phục sự cố và chẩn đoán từ xa
- Tích hợp với hệ thống quản lý kho hàng

### 7.2.2 Phân tích Bán hàng Nâng cao

**Cải tiến Đề xuất:**

- Ghi nhật ký lịch sử giao dịch với dấu thời gian
- Theo dõi mức độ phổ biến của sản phẩm
- Xác định thời gian sử dụng cao điểm
- Báo cáo doanh thu và dự báo
- Phân tích hành vi khách hàng

**Hiện thực:** Bộ đệm vòng trong EEPROM ngoài/thẻ SD, cấu trúc dữ liệu cho bản ghi giao dịch, thuật toán phân tích thống kê, chức năng xuất dữ liệu.

### 7.2.3 Hỗ trợ Đa ngôn ngữ

**Trạng thái Hiện tại:** Giao diện chỉ tiếng Anh.

**Cải tiến Đề xuất:**

- Tùy chọn tiếng Việt, tiếng Anh và các ngôn ngữ khác
- Menu chọn ngôn ngữ khi khởi động hoặc qua chế độ quản trị
- Cấu trúc bảng chuỗi cho quản lý dịch thuật hiệu quả
- Hỗ trợ Unicode cho các ký tự không phải ASCII

### 7.2.4 Hồ sơ Người dùng và Chương trình Khách hàng Thân thiết

**Cải tiến Đề xuất:**

- Nhận dạng người dùng dựa trên thẻ RFID
- Lịch sử mua hàng và sở thích được lưu trữ theo người dùng
- Hệ thống tích lũy điểm thưởng
- Đề xuất sản phẩm cá nhân hóa
- Giảm giá và khuyến mãi cho thành viên

### 7.2.5 Chẩn đoán Lỗi Nâng cao

**Cải tiến Đề xuất:**

- Ghi nhật ký lỗi toàn diện với dấu thời gian và mã lỗi
- Các quy trình tự chẩn đoán cho kiểm tra sức khỏe ngoại vi
- Cảnh báo bảo trì dự đoán (ví dụ: mức phễu xu thấp)
- Công cụ phân tích lỗi cho kỹ thuật viên dịch vụ
- Báo cáo lỗi tự động đến máy chủ trung tâm

## 7.3 Cải thiện Trải nghiệm Người dùng

### 7.3.1 Phản hồi Âm thanh

**Cải tiến Đề xuất:**

- Còi hoặc loa nhỏ cho phản hồi âm thanh
- Tiếng bip xác nhận cho các lần nhấn phím
- Nhắc nhở bằng giọng nói cho các bước giao dịch
- Âm báo cho lỗi hoặc hoàn thành

### 7.3.2 Tính năng Hỗ trợ Truy cập

#### Cải tiến Đề xuất:

- Chế độ hiển thị độ tương phản cao cho người khiếm thị
- Hướng dẫn âm thanh và đầu ra giọng nói
- Tùy chọn lớp phủ chữ nổi Braille cho các nút vật lý
- Cơ chế điều chỉnh chiều cao hoặc độ nghiêng
- Chế độ nút lớn với giao diện đơn giản hóa

### 7.3.3 In Hóa đơn

#### Cải tiến Đề xuất:

- Mô-đun máy in nhiệt cho hóa đơn giao dịch
- In tên sản phẩm, số lượng, giá, tổng cộng, tiền thừa
- Bao gồm ID giao dịch cho mục đích hoàn tiền/bảo hành
- Tùy chọn hóa đơn email qua kết nối mạng

## 7.4 Cải tiến Bảo mật

### 7.4.1 Bảo mật Quản trị Nâng cao

**Trạng thái Hiện tại:** Mật khẩu 6 chữ số cố định với thời gian chờ 10 giây.

#### Cải tiến Đề xuất:

- Mật khẩu có thể cấu hình với mã hóa
- Truy cập đa cấp (quản lý, kỹ thuật viên, vận hành viên)
- Chức năng đổi mật khẩu
- Ghi nhật ký nỗ lực đăng nhập
- Khóa tài khoản sau nhiều lần thử thách bại
- Xác thực thẻ RFID/NFC cho nhân viên dịch vụ

### 7.4.2 Bảo vệ Chống Giả mạo

#### Cải tiến Đề xuất:

- Công tắc phát hiện giả mạo trên các bảng truy cập
- Giám sát điện áp cho các cuộc tấn công nguồn điện
- Gia tốc kế cho phát hiện nghiêng/rung
- Thông báo động khi có nỗ lực giả mạo
- Xác minh khởi động an toàn

## 7.5 Quản lý Năng lượng

### 7.5.1 Hiệu quả Năng lượng

#### Cải tiến Đề xuất:

- Các chế độ năng lượng thấp của STM32 trong thời gian nhàn rỗi
- Tự động làm mờ đèn nền LCD sau thời gian chờ
- Chế độ ngủ với khả năng đánh thức bằng cảm biến
- Tùy chọn pin năng lượng mặt trời cho lắp đặt ngoài trời
- Pin dự phòng cho khả năng phục hồi khi mất điện

### 7.5.2 Giám sát Môi trường

#### Cải tiến Đề xuất:

- Cảm biến nhiệt độ cho các ứng dụng làm lạnh
- Giám sát độ ẩm cho các sản phẩm nhạy cảm
- Cảnh báo tự động cho các điều kiện ngoại phạm vi
- Điều khiển máy nén cho hệ thống làm mát

## 7.6 Các Lĩnh vực Sản phẩm Thay thế

Mặc dù dự án này tập trung vào skin game, kiến trúc hệ thống có thể thích ứng với:

- **Bán hàng Truyền thống:** Dồ ăn nhẹ, đồ uống, vật dụng cá nhân
- **Bán vé:** Vé sự kiện, thẻ đỗ xe, thẻ giao thông
- **Nội dung Số:** Mã tải xuống, thẻ trả trước, giấy phép
- **Dịch vụ Cho thuê:** Sạc dự phòng, ô dù, công cụ
- **Tủ khóa Thông minh:** Hệ thống nhận/giao gói hàng

## 7.7 Cân nhắc về Khả năng Mở rộng

### 7.7.1 Nâng cấp Vi điều khiển

Cho các tính năng nâng cao, xem xét chuyển sang:

- **Dòng STM32F4:** 168MHz, nhiều bộ nhớ hơn, đơn vị dấu phẩy động phần cứng
- **Dòng STM32H7:** 480MHz, thiết bị ngoại vi tiên tiến, tùy chọn lõi kép
- **ESP32:** Tích hợp WiFi/Bluetooth, lõi kép, chi phí hiệu quả
- **Raspberry Pi:** Hệ điều hành Linux đầy đủ, hệ sinh thái phần mềm phong phú, khả năng hiển thị

### 7.7.2 Kiến trúc Mô-đun

Thiết kế hệ thống như các mô-đun hợp tác:

- Bộ điều khiển chính (STM32) cho điều khiển thời gian thực
- Mô-đun giao tiếp (ESP32) cho kết nối mạng
- Bộ xử lý thanh toán (mô-đun bảo mật chuyên dụng)
- Bộ điều khiển hiển thị (MCU riêng cho đồ họa)
- Giao tiếp giữa các mô-đun qua bus CAN hoặc SPI

## 8 Kết luận

### 8.1 Thành tựu Dự án

Dự án này đã thiết kế và hiện thực thành công một hệ thống máy bán hàng tự động toàn diện sử dụng vi điều khiển STM32F103C8T6. Hệ thống hoàn chỉnh thể hiện ứng dụng thực tế của các khái niệm hệ thống nhúng và đạt được tất cả các mục tiêu chính:

#### 8.1.1 Thành tựu Kỹ thuật

- **Tích hợp Phần cứng Hoàn chỉnh:** Giao tiếp thành công vi điều khiển với màn hình LCD, bàn phím ma trận, cảm biến siêu âm và bộ nhớ Flash, tạo ra một hệ thống nhúng đầy đủ chức năng.
- **Hiện thực Máy trạng thái Mạnh mẽ:** Phát triển một FSM 19 trạng thái quản lý các luồng giao dịch phức tạp, xử lý lỗi và các chức năng quản trị với hành vi xác định.
- **Lưu trữ Dữ liệu Bền vững:** Hiện thực lưu trữ không bay hơi sử dụng bộ nhớ Flash nội với xác thực số ma thuật, đảm bảo dữ liệu kho hàng tồn tại qua các chu kỳ nguồn.
- **Trình điều khiển Ngoại vi Tùy chỉnh:** Tạo các trình điều khiển hiệu quả cho giao tiếp LCD I2C bit-banged, quét bàn phím chống rung và đo khoảng cách siêu âm dựa trên bộ định thời.
- **Xử lý Lỗi Toàn diện:** Hiện thực xác thực đầu vào, quản lý thời gian chờ, bộ đếm lỗi và cơ chế phục hồi đảm bảo độ tin cậy của hệ thống và hoạt động thân thiện với người dùng.
- **Tự động Phát hiện Khách hàng:** Đạt được hoạt động tự chủ với phát hiện sự hiện diện của khách hàng dựa trên cảm biến siêu âm và lọc nhiễu 3 giây.
- **Quản trị An toàn:** Cung cấp chế độ quản trị được bảo vệ bằng mật khẩu với bảo mật dựa trên thời gian chờ cho quản lý kho hàng.

#### 8.1.2 Chỉ số Hiệu suất

Hệ thống thể hiện các đặc điểm hiệu suất xuất sắc:

- Thời gian phản hồi: Dưới 100ms cho tất cả các tương tác người dùng
- Độ chính xác đo khoảng cách: 97-99% trong phạm vi hoạt động
- Tỷ lệ thành công giao dịch: 97% (loại trừ lỗi người dùng)
- Độ tin cậy hệ thống: Không gặp sự cố trong quá trình hoạt động liên tục 12 giờ
- Phục hồi lỗi: Tỷ lệ thành công 100% trong việc trở về trạng thái an toàn
- Độ bền Flash: Đã kiểm tra 1,500+ chu kỳ ghi mà không bị hỏng dữ liệu

### 8.1.3 Chất lượng Phần mềm

Việc hiện thực thể hiện các thực hành kỹ thuật phần mềm chuyên nghiệp:

- Kiến trúc mô-dun với sự phân tách rõ ràng các mối quan tâm
- Quy ước đặt tên và tổ chức mã nhất quán
- Quản lý trạng thái toàn diện ngăn chặn hành vi không xác định
- Dấu chân bộ nhớ tối thiểu (65% Flash, 20% RAM sử dụng)
- Không cấp phát bộ nhớ động đảm bảo hành vi có thể dự đoán
- Cấu trúc mã được tài liệu hóa tốt tạo thuận lợi cho bảo trì

## 8.2 Giá trị Giáo dục

Dự án này đã cung cấp những trải nghiệm học tập đáng kể trên nhiều lĩnh vực:

### 8.2.1 Khái niệm Hệ thống Nhúng

- Kiến trúc vi điều khiển và cấu hình ngoại vi
- Lập trình thời gian thực và các ràng buộc thời gian
- Lập trình hướng ngắn và quản lý bộ định thời
- Thách thức tích hợp phần cứng-phần mềm
- Thiết kế hệ thống hạn chế tài nguyên

### 8.2.2 Giao tiếp Phần cứng

- Cấu hình và điều khiển I/O số
- Hiện thực và gỡ lỗi giao thức I2C
- Bắt đầu vào bộ định thời cho các phép đo chính xác
- Lập trình bộ nhớ Flash và lưu trữ dữ liệu bền vững
- Giao tiếp cảm biến và xử lý tín hiệu

### 8.2.3 Thiết kế Phần mềm

- Thiết kế và hiện thực máy trạng thái hữu hạn
- Các mô hình lập trình hướng sự kiện
- Chiến lược xử lý lỗi và phục hồi
- Phát triển thuật toán cho các ứng dụng cụ thể
- Tối ưu hóa mã cho môi trường nhúng

### 8.2.4 Tích hợp Hệ thống

- Lựa chọn thành phần và phân tích tương thích
- Phân tích thời gian và phối hợp
- Gỡ lỗi các tương tác phần cứng-phần mềm phức tạp
- Phát triển phương pháp kiểm thử
- Tài liệu hóa và viết kỹ thuật

## 8.3 Thách thức Đã Vượt qua

Một số thách thức kỹ thuật đáng kể đã được giải quyết trong quá trình phát triển:

- **Ôn định Giao tiếp LCD:** Giải quyết các vấn đề thời gian trong hiện thực I2C bit-banged thông qua hiệu chỉnh độ trễ cẩn thận
- **Chống rung Bàn phím:** Loại bỏ các kích hoạt sai và hiện tượng bóng ma thông qua các thuật toán quét cải tiến
- **Lọc Nhiều Cảm biến:** Đạt được phát hiện khách hàng hàng đáng tin cậy bất chấp nhiều môi trường thông qua yêu cầu phát hiện liên tục
- **Tối ưu hóa Ghi Flash:** Giảm thiểu chu kỳ ghi để kéo dài tuổi thọ bộ nhớ trong khi duy trì tính nhất quán dữ liệu
- **Phức tạp Máy trạng thái:** Quản lý 19 trạng thái với nhiều đường chuyển đổi thông qua thiết kế và kiểm thử có hệ thống

## 8.4 Ứng dụng Thực tế

Mặc dù được phát triển như một dự án giáo dục, hệ thống thể hiện các khái niệm áp dụng cho máy bán hàng tự động thương mại và tự động hóa bán lẻ:

- Tự động phát hiện khách hàng giảm tiêu thụ năng lượng
- Theo dõi kho hàng cho phép trí tuệ kinh doanh
- Xử lý lỗi đảm bảo hoạt động liên tục
- Chế độ quản trị hỗ trợ bảo trì tại hiện trường
- Thiết kế mô-đun tạo thuận lợi cho mở rộng tính năng

Kiến trúc có thể được thích ứng cho các kịch bản bán lẻ tự động khác nhau bao gồm máy bán hàng truyền thống, ki-ốt bán vé, hệ thống cho thuê và tủ khóa thông minh.

## 8.5 Hạn chế Được Thừa nhận

Việc hiện thực hiện tại có một số hạn chế đại diện cho các cơ hội phát triển trong tương lai:

- Mô phỏng thanh toán không có phần cứng xử lý tiền tệ thực
- Dung lượng kho hàng hạn chế (16 sản phẩm)
- Hiển thị chỉ văn bản hạn chế sự tinh vi của giao diện người dùng
- Không có cơ chế nhả vật lý
- Xử lý giao dịch đơn (không có người dùng đồng thời)
- Thiếu kết nối mạng cho quản lý từ xa

Những hạn chế này là các quyết định thiết kế có ý thức để tập trung vào các khái niệm hệ thống nhúng cốt lõi trong phạm vi và thời gian dự án. Phần 7 cung cấp các đề xuất cải tiến chi tiết giải quyết từng hạn chế.

## 8.6 Bài học Kinh nghiệm

### 8.6.1 Thông tin Kỹ thuật

- **Thời gian là Quan trọng:** Các lỗi thời gian nhỏ trong giao tiếp I2C hoặc kích hoạt cảm biến dẫn đến các lỗi hệ thống
- **Chống rung là Cần thiết:** Các công tắc phần cứng yêu cầu lọc phần mềm; rung cơ học gây ra các vấn đề đáng kể
- **Máy trạng thái Mở rộng Tốt:** Kiến trúc FSM xử lý sự phức tạp tốt hơn các phương pháp dựa trên cờ
- **Giảm thiểu Ghi Flash:** Giảm ghi chiến lược kéo dài tuổi thọ bộ nhớ đáng kể
- **Kiểm thử Không thể Bỏ qua:** Kiểm thử có hệ thống đã phát hiện các vấn đề không rõ ràng trong xem xét mã

### 8.6.2 Quy trình Phát triển

- Phát triển gia tăng với kiểm thử thường xuyên giảm thời gian gỡ lỗi
- Thiết kế mô-đun cho phép phát triển song song và khắc phục sự cố dễ dàng hơn
- Tài liệu hóa trong quá trình phát triển (không phải sau đó) cải thiện chất lượng mã
- Gỡ lỗi phần cứng yêu cầu các công cụ máy hiện sóng và phân tích logic
- Kiểm soát phiên bản (Git) cần thiết để theo dõi thay đổi và hoàn tác sai lầm

## 8.7 Tác động Dự án

Dự án này chứng minh rằng các hệ thống nhúng phức tạp có thể được hiện thực thành công trên các nền tảng vi điều khiển chi phí thấp. Tài liệu toàn diện và thiết kế mô-đun làm cho nó phù hợp như:

- Tài liệu tham khảo giáo dục cho các khóa học hệ thống nhúng
- Nền tảng cho phát triển máy bán hàng tự động thương mại
- Nền tảng cho nghiên cứu về tương tác người-máy
- Giường thử nghiệm cho các thực hành kỹ thuật phần mềm nhúng
- Trình diễn khả năng thư viện STM32 HAL

## 8.8 Lời kết

Dự án máy bán hàng tự động đã đạt được thành công các mục tiêu thiết kế và hiện thực một hệ thống nhúng toàn diện thể hiện tích hợp phần cứng, điều khiển máy trạng thái, lưu trữ dữ liệu bền vững và tương tác người dùng. Hệ thống hoạt động tin cậy, xử lý lỗi một cách duyên dáng và cung cấp giao diện trực quan cho cả khách hàng và quản trị viên.

Ngoài các thành tựu kỹ thuật, dự án đã cung cấp kinh nghiệm quý báu về phương pháp thiết kế hệ thống nhúng, tích hợp phần cứng-phần mềm và giải quyết vấn đề dưới các ràng buộc tài nguyên. Kiến trúc mô-đun và tài liệu kỹ lưỡng đảm bảo hệ thống có thể phục vụ như một nền tảng cho các cải tiến trong tương lai và mục đích giáo dục.

Các kỹ năng phát triển qua dự án này—lập trình vi điều khiển, giao tiếp ngoại vi, thiết kế hệ thống thời gian thực và gỡ lỗi có hệ thống—có thể áp dụng trực tiếp cho phát triển hệ thống nhúng chuyên nghiệp trong tự động hóa công nghiệp, điện tử tiêu dùng và các ứng dụng IoT.

## 9 Tài liệu Tham khảo & Tác giả

### 9.1 Tài liệu Tham khảo Kỹ thuật

#### 9.1.1 Tài liệu Vị điều khiển

1. STMicroelectronics, "STM32F103x8/B Reference Manual," RM0008, Rev 21, 2021. Có sẵn: [https://www.st.com/resource/en/reference\\_manual/cd00171190.pdf](https://www.st.com/resource/en/reference_manual/cd00171190.pdf)
2. STMicroelectronics, "STM32F103C8T6 Datasheet," DocID 13587 Rev 18, 2021. Có sẵn: <https://www.st.com/resource/en/datasheet/stm32f103c8.pdf>
3. STMicroelectronics, "Description of STM32F1 HAL and Low-Layer Drivers," UM1850, Rev 12, 2021. Có sẵn: [https://www.st.com/resource/en/user\\_manual/dm00154093.pdf](https://www.st.com/resource/en/user_manual/dm00154093.pdf)
4. ARM Limited, "Cortex-M3 Technical Reference Manual," Revision r2p1, 2010.

#### 9.1.2 Công cụ Phát triển

5. STMicroelectronics, "STM32CubeIDE User Guide," UM2609, Rev 3, 2021.
6. STMicroelectronics, "STM32CubeMX User Manual," UM1718, Rev 36, 2021.
7. STMicroelectronics, "STM32CubeProgrammer Software User Manual," UM2237, Rev 33, 2021.

#### 9.1.3 Thành phần Phần cứng

8. Hitachi, "HD44780U LCD Controller/Driver Datasheet," 1998.
9. NXP Semiconductors, "PCF8574 8-bit I/O Expander for I2C-bus," Rev. 05, 2013.
10. ELECFreaks, "HC-SR04 Ultrasonic Sensor Datasheet," 2013.
11. Cytron Technologies, "4×4 Matrix Keypad User Manual," 2015.

#### 9.1.4 Giao thức Giao tiếp

12. NXP Semiconductors, "I2C-bus Specification and User Manual," UM10204, Rev. 7.0, 2021.
13. Philips Semiconductors, "The I<sup>2</sup>C-Bus Specification," Version 2.1, 2000.

#### 9.1.5 Kỹ thuật Phần mềm

14. Samek, M., "Practical UML Statecharts in C/C++: Event-Driven Programming for Embedded Systems," 2nd Edition, Newnes, 2008.
15. Labrosse, J. J., "Embedded Systems Building Blocks: Complete and Ready-to-Use Modules in C," CMP Books, 1999.
16. White, E., "Making Embedded Systems: Design Patterns for Great Software," O'Reilly Media, 2011.

### 9.1.6 Lập trình C Nhúng

17. Pont, M. J., "Embedded C," 2nd Edition, Addison-Wesley, 2002.
18. Barr, M., "Embedded C Coding Standard," Barr Group, 2018.

### 9.1.7 Tài nguyên Trực tuyến

19. Diễn đàn Cộng đồng STM32: <https://community.st.com/>
20. Kho lưu trữ GitHub STM32: <https://github.com/STMicroelectronics>
21. Tài liệu ARM CMSIS: [https://arm-software.github.io/CMSIS\\_5/](https://arm-software.github.io/CMSIS_5/)
22. Hướng dẫn Hệ thống Nhúng: <https://www.embedded.com/>

## 9.2 Kho lưu trữ Dự án

Mã nguồn dự án, tài liệu và tài nguyên có sẵn tại:

- **Kho lưu trữ GitHub:** <https://github.com/1172005thinh/VendingMachine>
- **Cấu trúc Dự án:** Thư mục VendingMachine/ chứa tất cả các tệp nguồn
- **Tài liệu:** README.md với hướng dẫn thiết lập

## 9.3 Tác giả

### 9.3.1 Nhóm Dự án

Dự án này được phát triển bởi ba sinh viên từ Trường Đại học Bách Khoa - ĐHQG-HCM (HCMUT):

Tên	Đóng góp
Nguyễn Hưng Thịnh	<ul style="list-style-type: none"> <li>• Tích hợp phần cứng và thiết kế mạch</li> <li>• Kiểm thử và gỡ lỗi hệ thống</li> <li>• Chuẩn bị tài liệu</li> <li>• Quản lý dự án và phối hợp nhóm</li> <li>• Quay video trình diễn và soạn thảo báo cáo cuối cùng</li> </ul>
Lê Thế Lộc	<ul style="list-style-type: none"> <li>• Phát triển trình điều khiển bàn phím</li> <li>• Hệ thống quản lý kho hàng</li> <li>• Các hoạt động bộ nhớ Flash</li> <li>• Thiết kế kiến trúc hệ thống</li> <li>• Phát triển trình điều khiển cảm biến</li> </ul>
Trần Doãn Hoàng Lâm	<ul style="list-style-type: none"> <li>• Thiết kế và hiện thực máy trạng thái hữu hạn</li> <li>• Hiện thực hệ thống bộ định thời</li> <li>• Thiết kế giao diện người dùng</li> <li>• Thực hiện, soạn kịch bản trình diễn</li> </ul>

### 9.3.2 Tổ chức

#### Trường Đại học Bách Khoa - ĐHQG-HCM (HCMUT)

- Khoa Khoa học và Kỹ thuật Máy tính
- Bộ môn Đồ án Thiết kế Luận lý
- Địa chỉ: 268 Lý Thường Kiệt, Quận 10, Thành phố Hồ Chí Minh, Việt Nam
- Website: <https://www.hcmut.edu.vn/>

### 9.3.3 Thông tin Khóa học

- **Khóa học:** Đồ án Thiết kế Luận lý
- **Năm học:** 2025 - 2026
- **Thời gian Dự án:** Tháng 09/2025 - Tháng 12/2025
- **Ngày Hoàn thành:** 20/12/2025

## 9.4 Lời cảm ơn

Các tác giả xin bày tỏ lòng biết ơn đến:

- Các cố vấn khoa về hướng dẫn các nguyên tắc thiết kế hệ thống luân lý
- Cộng đồng mã nguồn mở về các ví dụ mã và hỗ trợ khắc phục sự cố

## 9.5 Giấy phép và Sử dụng

Dự án này được phát triển cho mục đích giáo dục như một phần của khóa học tại HCMUT. Mã nguồn và tài liệu được cung cấp cho:

- Tham khảo giáo dục và học tập
- Nghiên cứu và học tập học thuật
- Trình diễn hệ thống nhúng phi thương mại
- Phát triển và cải tiến thêm

Đối với các ứng dụng thương mại hoặc các tác phẩm phái sinh, yêu cầu ghi công thích hợp cho các tác giả gốc và tổ chức.

## 9.6 Thông tin Liên hệ

Dối với các câu hỏi, đề xuất hoặc cơ hội hợp tác liên quan đến dự án này:

- **Tổ chức:** Trường Đại học Bách Khoa - ĐHQG-HCM
- **Khoa:** Khoa học và Kỹ thuật Máy tính

---

*Báo cáo này được biên soạn vào ngày 20 tháng 12 năm 2025*

*Trường Đại học Bách Khoa - ĐHQG-HCM*

*Hệ thống Máy bán hàng Tự động - Dồ án Thiết kế Luận lý*

---