

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



MÁY BÁN HÀNG TỰ ĐỘNG
Báo cáo

Đồ án môn học Thiết kế Luận lý

Giảng viên hướng dẫn:
Nguyễn Thành Lộc

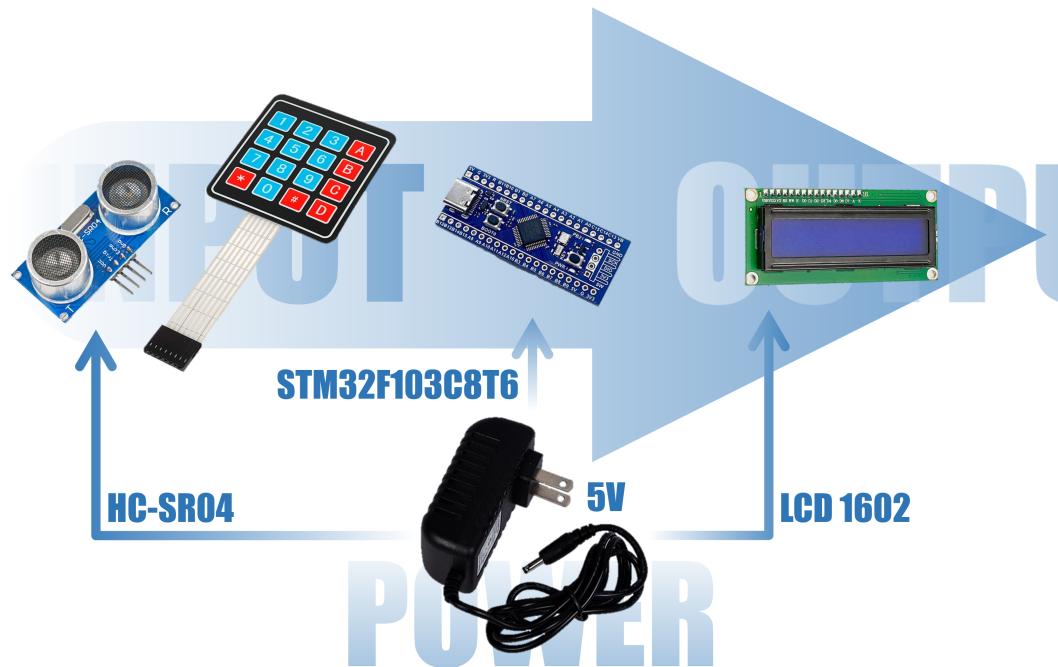
Nhóm tác giả:
Nguyễn Hưng Thịnh
Lê Thế Lộc
Trần Doãn Hoàng Lâm

Ngày 22 tháng 12 năm 2025

Mục lục

1 Sơ lược Hệ thống	3
1.1 Yêu cầu đặt ra	3
1.2 Luồng Vận hành Hệ thống	4
2 Phần cứng	5
2.1 Vi điều khiển - STM32F103C8T6	5
2.2 LCD 1602 với I2C	6
2.3 Bàn phím Ma trận 4×4	7
2.4 Cảm biến Siêu âm HC-SR04	7
2.5 Bộ nhớ Flash	7
3 Phần mềm	8
3.1 Máy trạng thái Hữu hạn	8
3.2 Timer/Scheduler	10
3.3 Cấu trúc Dữ liệu và Quản lý Bộ nhớ	11
3.4 Các Thuật toán Chính	12
3.4.1 Thuật toán Phát hiện người dùng	12
3.4.2 Thuật toán đẻ Thanh toán	12
3.4.3 Thuật toán Lưu trữ Kho hàng	13
3.4.4 Thuật toán Xác thực Quản trị	14
3.5 Project Structure	15
4 Hiện thực	16
4.1 Hiện thực Trình điều khiển Ngoại vi	16
4.2 Hiện thực Máy trạng thái	20
4.3 Giao diện hiển thị	23
4.4 Cơ chế Xử lý Lỗi	24
4.5 Tối ưu hiệu năng	25
4.6 Phát triển và Debug	26
5 Demo	27
6 Kết luận	28
6.1 Thành tựu Dự án	28
6.2 Ứng dụng Thực tế	28
7 Thông tin Dự án	28
7.1 GitHub repository	28
7.2 Tác giả	29
7.3 Lời cảm ơn	29

1 Sơ lược Hệ thống



Hình 1: Sơ đồ hệ thống

1.1 Yêu cầu đặt ra

- FR1: **Tự động ON/OFF:** Hệ thống sẽ phát hiện sự hiện diện của người dùng 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
- FR3: **Thông tin Sản phẩm:** Màn hình sẽ hiển thị tên sản phẩm, số lượng có sẵn và giá bán
- FR4: **Chọn Số lượng:** Người dùng sẽ nhập số lượng từ 1-9 đơn vị, nếu có lỗi thì sẽ thông báo yêu cầu nhập lại
- FR5: **Thanh toán:** Hệ thống sẽ chấp nhận các mệnh giá (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 không bán món hàng đã hết hàng
- FR7: **Lưu trữ Dữ liệu:** Dữ liệu kho hàng sẽ luôn được lưu lại kể cả khi mất điện
- FR8: **Quản trị:** Chế độ quản trị được bảo vệ bằng mật khẩu sẽ cho phép điều chỉnh kho hàng và giá cả

FR9: **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: **Cập nhật:** Khi thanh toán thành công, hệ thống sẽ cập nhật kho hàng và quay lại lựa chọn sản phẩm

1.2 Luồng Vận hành Hệ thống

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

1. Giai đoạn Khởi tạo:

- Bật nguồn
- Nạp kho hàng từ bộ nhớ Flash (hoặc khởi tạo mặc định nếu là lần khởi động đầu tiên)
- Khởi tạo tất cả các ngoại vi (LCD, bàn phím, cảm biến siêu âm)
- Hiển thị màn hình khởi tạo WELCOME
- Vào trạng thái chờ

2. Giai đoạn Phát hiện người dùng:

- 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 màn hình và hiển thị thông báo chào mừng
- Chuyển sang chế độ chọn sản phẩm

3. Giai đoạn Chọn Sản phẩm:

- Hiển thị sản phẩm đang được select
- Nhấn phím Lên/Xuống để điều hướng qua 16 sản phẩm
- Nhấn phím # để xem thông tin chi tiết
- Nhấn phím D để vào chế độ quản trị (yêu cầu mật khẩu)
- Sau 30s nếu không có tương tác, quay lại trạng thái chờ

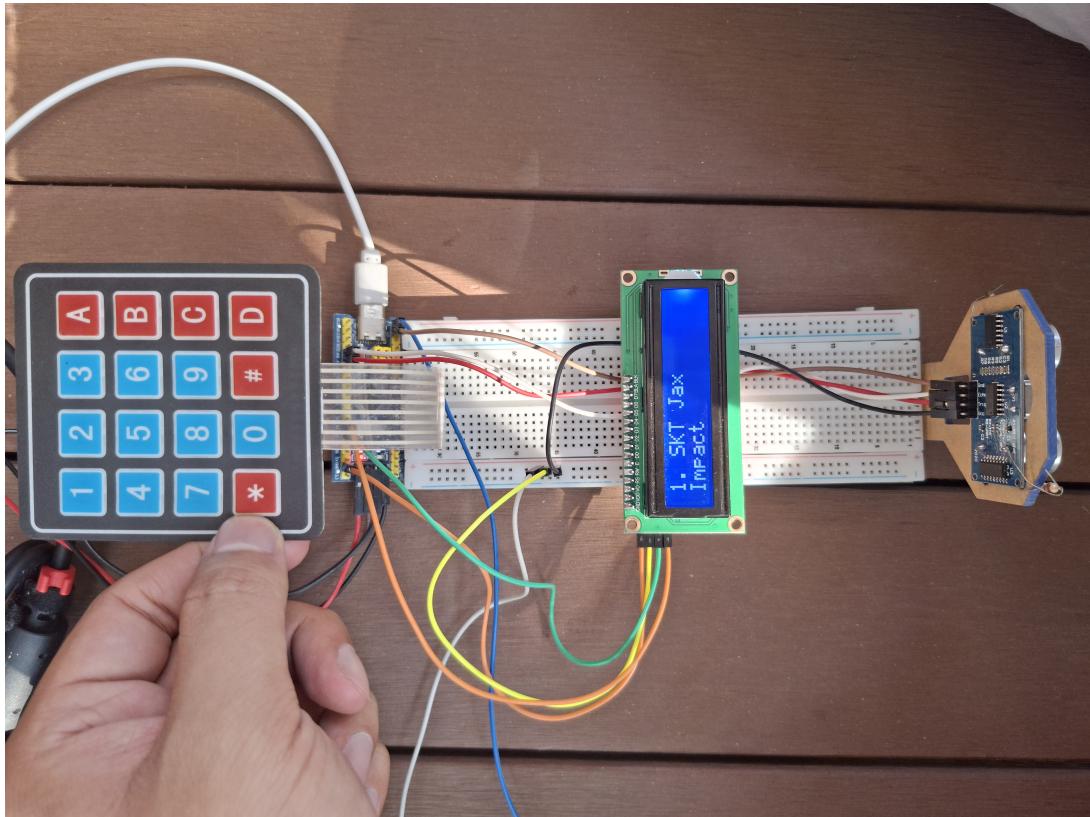
4. Giai đoạn Mua hàng:

- Hiển thị chi tiết số lượng tồn kho và giá
- Nhập đầu vào số lượng (1-9 đơn vị)
- Kiểm tra xem có đủ hàng không, nếu không thì thông báo ra màn hình
- Tính toán và hiển thị tổng số tiền thanh toán
- Nhập số tiền thanh toán (với các mệnh giá hợp lệ, có thể lặp lại bước này nhiều lần cho đến khi đủ tiền)
- 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 mã PIN 6 số
- Điều hướng sản phẩm cần điều chỉnh bằng nút Lên/Xuống
- Nhấn phím # để select sản phẩm cần chỉnh sửa
- Sửa đổi số lượng (0-9) và giá (1K-99K VND)
- Lưu thay đổi vào bộ nhớ Flash
- Sau 60s không có tương tác, tự động đăng xuất về chế độ người dùng

2 Phản ứng



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

2.1 Vi điều khiển - STM32F103C8T6

STM32F103C8T6 (thường được gọi là "Blue Pill") đóng vai trò là đơn vị xử lý trung tâm của hệ thống.

STM32F103C8T6 được chọn vì:

- **Sức mạnh:** 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, timer và GPIO đáp ứng mọi yêu cầu giao diện

- Thư viện đầy đủ:** Hỗ trợ tuyệt vời thông qua STM32CubeIDE và thư viện HAL
- Chi phí:** Bo mạch giá rẻ có sẵn ở nhiều nơi.
- Cộng đồng:** Cộng đồng người dùng lớn và tài liệu tham khảo/diễn đàn phong phú

Các pin 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 pull-up
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 (low-active)
PA13	SWDIO	Dữ liệu I/O debug Serial Wire
PA14	SWCLK	Xung nhịp debug Serial Wire

2.2 LCD 1602 với I2C

Hệ thống sử dụng màn hình LCD 16x2 kết hợp với bộ mở rộng I/O I2C PCF8574 để hiển thị thông tin.

Hệ thống hiện thực giao tiếp I2C bit-banged:

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

- Gửi nibble cao (4 bit) của byte dữ liệu
- Tạo xung chân Enable
- Gửi nibble thấp (4 bit) của byte dữ liệu
- Tạo xung chân Enable

2.3 Bàn phím Ma trận 4×4

Bàn phím ma trận cung cấp 16 phím được sắp xếp thành 4 hàng và 4 cột, hoạt động dựa trên nguyên tắc quét hàng-cột.

Các công tắc cơ học như bàn phím này có hiện tượng nhiễu khi nhấn phím, gây ra nhiều input trong một lần nhấn. Hệ thống sẽ cần phải debouncing 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    }
11 }
```

Listing 1: Hiện thực Chống bounce Bàn phím

2.4 Cảm biến Siêu âm HC-SR04

HC-SR04 sử dụng phép đo time-of-flight để đo khoảng cách.

Hệ thống sử dụng Kênh 1 của TIM1 được cấu hình cho chế độ Input Capture:

- **Input Capture:** Cả hai cạnh (lên và xuống)
- **Tần số:** 1 MHz
- **Ngắt:** Được kích hoạt khi có tín hiệu Input Capture

2.5 Bộ nhớ Flash

Hệ thống sử dụng page cuối (page 63) của bộ nhớ Flash nội bộ để lưu trữ dữ liệu kho hàng, đảm bảo dữ liệu được bảo toàn khi mất điện.

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

- **Magic number:** 4 byte (0xDEADBEEF)
- **Array sản phẩm:** 16 sản phẩm × 44 bytes = 704 bytes
- **Tổng:** 708 bytes (vừa khít trong 1KB của page Flash)

Khi bật nguồn:

1. Đọc số magic từ page 63 của Flash
2. Nếu số magic khớp 0xDEADBEEF: Tải kho hàng từ Flash
3. Nếu số magic invalid: Khởi tạo kho hàng mặc định và lưu vào Flash

3 Phần mềm

3.1 Máy trạng thái Hữu hạn

Trạng thái Khởi tạo (1-2):

- **INIT:** Khởi tạo hệ thống, thiết lập ngoại vi, tải kho hàng từ Flash
- **WAIT_SENSOR:** trạng thái chờ với giám sát cảm biến siêu âm liên tục để phát hiện người dùng

Chế độ người dùng (3-14):

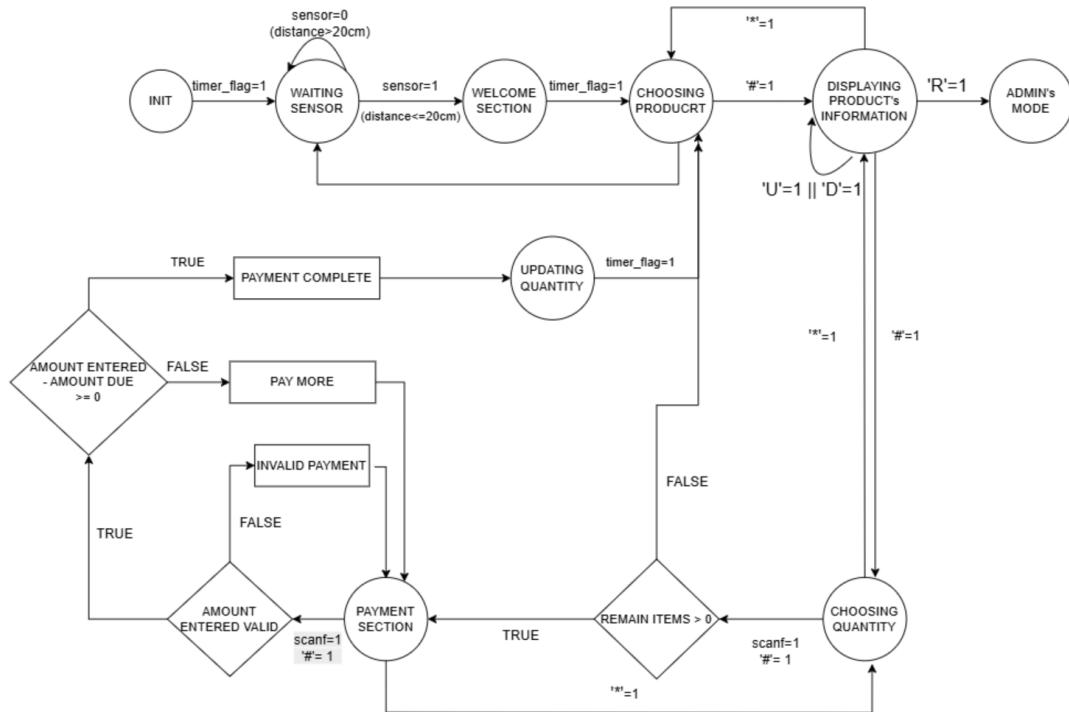
- **WELCOME_SECTION:** Hiển thị thông báo chào mừng trong 3 giây
- **CHOOSING_SKIN:** Duyệt sản phẩm với điều hướng Lên/Xuống
- **DISPLAY_INFO:** Hiển thị thông tin chi tiết sản phẩm (số lượng, giá)
- **CHOOSING_QUANTITY:** Nhập số lượng (1-9)
- **OUT_OF_STOCK_NOTIFICATION:** Cảnh báo khi sản phẩm đã chọn không có sẵn
- **QUANTITY_ERROR:** Yêu cầu thử lại khi số lượng nhập vào không hợp lệ
- **MAX_ERROR_STATE:** Hủy giao dịch sau 5 lần lỗi liên tiếp
- **PAYMENT_SHOW_TOTAL:** Hiển thị tổng số tiền phải trả
- **PAYMENT_INPUT:** Chỉ chấp nhận các mệnh giá thanh toán đã định (5K, 10K,..., 500K VND)
- **PAYMENT_ERROR:** Yêu cầu thử lại khi mệnh giá không hợp lệ
- **PAYMENT_INFO_WAIT:** Hiển thị số tiền còn lại hoặc tiền thừa
- **THANKS:** Hiển thị màn hình THANKS và quay lại chọn sản phẩm

Chế độ Quản trị (15-19):

- **ADMIN_MODE:** Xác nhận đăng nhập thành công
- **CHOOSING_SKIN_TO_ADJUST:** Chọn sản phẩm để config
- **TIMEOUT_ADMIN_MODE:** Tự động đăng xuất sau 60 giây không có tương tác
- **ADJUST_QUANTITY_AND_PRICE:** Chính sửa kho hàng và giá cả
- **CONFIRM_EXIT_ADMIN_MODE:** Hộp thoại xác nhận đăng xuất YES NO

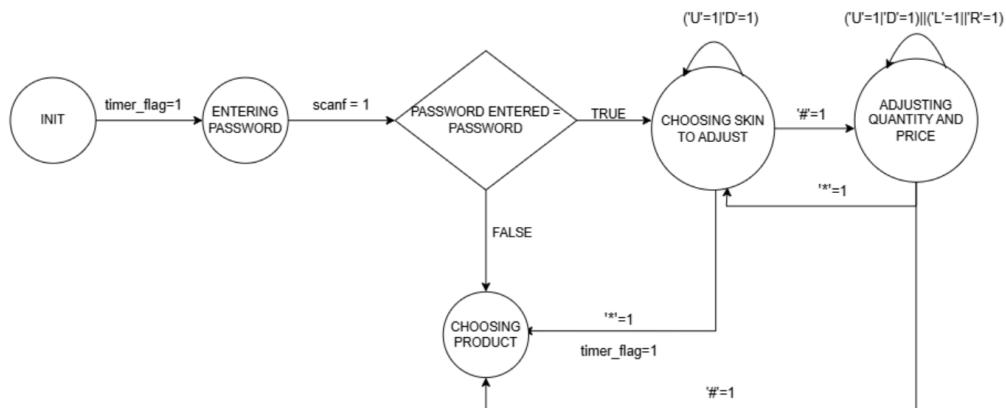
Sơ đồ Chuyển đổi Trạng thái

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



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

ADMIN's MODE



Hình 4: Sơ đồ chuyển đổi trạng thái (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

Xử lý 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 người dùng nào → trạng thái phục hồi thích hợp sau 30 giây

Login vào Chế độ Quản trị:

Mật khẩu là: 070596

CHOOSING_SKIN (Phím D + mật khẩu) -> ADMIN_MODE
-> CHOOSING_SKIN_TO_ADJUST -> ADJUST_QUANTITY_AND_PRICE
-> CHOOSING_SKIN_TO_ADJUST hoặc CONFIRM_EXIT_ADMIN_MODE

3.2 Timer/Scheduler

Hệ thống sử dụng cơ chế timer phần mềm để quản lý delay, thời gian chờ và chuyển đổi trạng thái mà không ngắt luồng thực thi.

Các loại timer

Năm loại timer độc lập được hiện thực:

Bảng 2: Các loại timer Phần mềm

Timer	Thời lượng	Mục đích
init_timer	1000ms	Delay 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 (user/admin)
message_timer	3000ms	Thời gian hiển thị thông báo (vd màn hình THANKS)
sensor_timer	100ms	Khoảng thời gian refresh cảm biến siêu âm

Hiện thực timer

Mỗi timer 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 2: Cấu trúc Dữ liệu timer

Thực thi timer

Hàm `timerRun()` được gọi mỗi 1ms (thường trong ngắn 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 3: Hàm Cập nhật timer

Mẫu Sử dụng timer

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 4: Ví dụ Sử dụng timer

3.3 Cấu trúc Dữ liệu và Quản lý Bộ nhớ

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_skinds[16];
```

Listing 5: Cấu trúc Dữ liệu Sản phẩm

44 byte mỗi sản phẩm \times 16 sản phẩm = 704 byte

Bộ nhớ Flash

Lưu trữ dữ liệu bằng 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 6: Cấu hình Bộ nhớ Flash

Biến Toàn cục

Các variable 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 7: Biến Trạng thái Toàn cục

3.4 Các Thuật toán Chính

3.4.1 Thuật toán Phát hiện người dùng

Hệ thống giám sát liên tục cảm biến siêu âm mỗi 100ms. Nếu phát hiện vật thể trong phạm vi 20cm trong 3 giây liên tiếp, hệ thống sẽ chuyển sang trạng thái chào mừng. Cơ chế này giúp lọc nhiễu và đảm bảo sự hiện diện có chủ ý của người dùng.

3.4.2 Thuật toán để Thanh toán

Thanh toán chỉ với các mệnh giá đã được chỉ định, không chấp nhận các mệnh giá khác:

```

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

```

Listing 8: Xác thực Thanh toán

3.4.3 Thuật toán Lưu trữ Kho hàng

Đọc/ghi Flash có kiểm tra magic number:

```

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 }

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

Listing 9: Các hoạt động Bộ nhớ Flash

3.4.4 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 10: Xác minh Mật khẩu Quản trị

3.5 Project Structure

Cấu trúc module: Dự án có source code tuân theo kiến trúc module với sự phân tách rõ ràng:

Bảng 3: Các module Phần mềm

module	Nhiệm vụ
main.c	File chính của chương trình, khởi tạo ngoại vi, vòng lặp chính
fsm_vm.c/h	Logic máy trạng thái hữu hạn, chuyển đổi trạng thái
store.c/h	Quản lý dữ liệu kho hàng, đọc/ghi Flash
keypad.c/h	Quét bàn phím ma trận, có debouncing, ánh xạ ký tự nhập vào màn hình
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	Giao thức I2C
timer.c/h	Quản lý soft-timer, xử lý thời gian chờ
ADMIN.c/h	Xác thực admin, xác minh mật khẩu

Module Hierarchy: 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)
```

4 Hiện thực

4.1 Hiện thực Trình điều khiển Ngoại vi

Trình điều khiển LCD I2C: Trình điều khiển LCD được hiện thực chế độ giao tiếp 4-bit thông qua giao thức I2C bit-banged. Cách tiếp cận này đảm bảo tương thích với các bo mạch chuyển đổi PCF8574 khác nhau và có 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 11: Hiện thực I2C Bit-Banged

Các hàm Ché 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 12: 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 13: Khởi tạo LCD

Trình điều khiển Bàn phím Trình điều khiển bàn phím hiện thực quét hàng với debouncing:

```

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     // Pull current row LOW
17     HAL_GPIO_WritePin(GPIOA, row_pins[r], GPIO_PIN_RESET);
18     HAL_Delay(1);
19
20     // Read all columns
21     for (int c = 0; c < 4; c++) {
22         if (HAL_GPIO_ReadPin(GPIOA, col_pins[c]) ==
23             GPIO_PIN_RESET) {
24             // Key pressed detected
25             HAL_Delay(20); // Debounce delay
26
27             // Confirm key still pressed
28             if (HAL_GPIO_ReadPin(GPIOA, col_pins[c]) ==
29                 GPIO_PIN_RESET) {
30                 // Wait for key release
31                 while (HAL_GPIO_ReadPin(GPIOA, col_pins[c])
32                     == GPIO_PIN_RESET);
33
34                 return keymap[r][c];
35             }
36         }
37     }
38     return 0; // No key pressed
39 }
```

Listing 14: Hiện thực Scan Bàn phím

Trình điều khiển Cảm biến Siêu âm Trình điều khiển cảm biến sử dụng timer bắt tín hiệu đầu vào để đo thời gian phản hồi của xung siêu âm. Quá trình đo bao gồm kích hoạt xung Trigger và đo độ rộng xung Echo sử dụng ngắt Input Capture.

```

1 void Sensor_Trigger(void) {
2     HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_SET);
3     delay_us(10);
4     HAL_GPIO_WritePin(TRIG_PORT, TRIG_PIN, GPIO_PIN_RESET);
5     __HAL_TIM_ENABLE_IT(&htim1, TIM_IT_CC1);
6 }
7
8 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
9     if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) {
10         if (Is_First_Captured==0) {
11             IC_Val1 = HAL_TIM_ReadCapturedValue(htim,
12                                                 TIM_CHANNEL_1);
13             Is_First_Captured = 1;
14             __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1,
15                                           TIM_INPUTCHANNELPOLARITY_FALLING
16                                         );
17         }
18         else if (Is_First_Captured==1) {
```

```

17     IC_Val2 = HAL_TIM_ReadCapturedValue(htim,
18         TIM_CHANNEL_1);
19     __HAL_TIM_SET_COUNTER(htim, 0);
20
21     if (IC_Val2 > IC_Val1)
22         Difference = IC_Val2 - IC_Val1;
23     else
24         Difference = (0xffff - IC_Val1) + IC_Val2;
25
26     Distance = Difference * .034/2;
27     Is_First_Captured = 0;
28
29     __HAL_TIM_SET_CAPTUREPOLARITY(htim, TIM_CHANNEL_1,
30                                     TIM_INPUTCHANNELPOLARITY_RISING
31                                     );
32     __HAL_TIM_DISABLE_IT(&htim1, TIM_IT_CC1);
33 }
```

Listing 15: Đo khoảng cách với HC-SR04

4.2 Hiện thực Máy trạng thái

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 16: Trạng thái CHOOSING_SKIN

Trạng thái PAYMENT_INPUT: Trạng thái này xử lý việc nhập tiền từ người dùng, cộng dồn số tiền đã insert và kiểm tra xem đã đủ để thanh toán chưa.

```

1 case PAYMENT_INPUT:
2 {
3     if (timeout_timer_flag == 1) {
4         // Timeout handling...
5         break;
6     }
7
8     char key = Keypad_Scan();
9     if (key != 0) {
10        setTimeoutTimer(30000);
11
12        if (key >= '0' && key <= '9') {
13            // Accumulate input amount
14            if (money_inserted_current < 1000000000) {
15                money_inserted_current = money_inserted_current *
16                    10 + (key - '0');
17                display_payment_input();
18            }
19        else if (key == '#') {
20            if (!is_valid_money(money_inserted_current)) {
21                payment_error_count++;
22                // Error handling...
23            }
24        else {
25            money_paid_accumulated += money_inserted_current;
26
27            if (money_paid_accumulated < total_payable) {
28                status = PAYMENT_INFO_WAIT;
29                // Show remaining amount
30            }
31        }
32    }
33 }
34 }
35 }
36 }
37

```

```

31         else {
32             status = PAYMENT_INFO_WAIT;
33             // Show change and success
34         }
35     }
36 }
37 }
38 break;

```

Listing 17: PAYMENT_INPUT

Trạng thái ADMIN_MODE: Chế độ admin cho phép quản trị viên/người nhập kho thay đổi giá cả và tồn kho của từng sản phẩm.

```

1 case CHOOSING_SKIN_TO_ADJUST:
2 {
3     if (timeout_timer_flag == 1) {
4         status = TIMEOUT_ADMIN_MODE;
5         break;
6     }
7
8     char key = Keypad_Scan();
9     if (key != 0) {
10        setTimeoutTimer(60000);
11
12        if (key == 'U') {
13            current_id--;
14            if (current_id < 1) current_id = 16;
15            display_current_skin(current_id);
16        }
17        else if (key == 'D') {
18            current_id++;
19            if (current_id > 16) current_id = 1;
20            display_current_skin(current_id);
21        }
22        else if (key == '#') {
23            status = ADJUST_QUANTITY_AND_PRICE;
24            lcd_clear();
25            display_adjust_quantity_and_price(current_id, 0);
26        }
27    }
28 }
29 break;
30 \end{lstlisting}
31     if (key == 'U') {
32         current_id--;
33         if (current_id < 1) current_id = 16; // Wrap around
34         display_current_skin(current_id);
35     }
36     else if (key == 'D') {

```

```

37     current_id++;
38     if (current_id > 16) current_id = 1; // Wrap around
39     display_current_skin(current_id);
40   }
41   else if (key == '#') {
42     status = DISPLAY_INFO;
43     lcd_clear();
44     display_skin_detail(current_id);
45   }
46   else if (key == 'R') {
47     int is_admin = admin_log();
48     if (is_admin) {
49       status = ADMIN_MODE;
50       // Display admin success message
51     }
52   }
53 }
54 }
55 break;

```

Listing 18: Chế độ Quản trị

4.3 Giao diện hiển thị

Các hàm hiển thị .

```

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   sprintf(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   sprintf(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 19: Các hàm hiển thị LCD

4.4 Cơ chế Xử lý Lỗi

Kiểm tra INPUT .

```

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
19     for (int i = 0; i < 7; i++) {
20         if (amount == valid_denominations[i]) {
21             return 1;
22         }
23     }
24     return 0;
}
```

25 }

Listing 20: Các hàm Kiểm tra INPUT

Số lần Lỗi Tối đa .

```

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
20if (validation_success) {
21    error_count = 0;
22}

```

Listing 21: Không quá 5 lần Lỗi

4.5 Tối ưu hiệu năng**Giảm số lần cập nhật hiển thị:** Chỉ cập nhật LCD khi nội dung thay đổi:

```

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 22: Giảm số lần refresh LCD

Tối ưu Timer: Chỉ sử dụng một SysTick duy nhất cho tất cả timer:

```

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

Listing 23: Tối ưu Quản lý Timer

Tối ưu khi ghi Flash: Chỉ ghi Flash khi dữ liệu thực sự thay đổi:

```

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 24: Tối ưu khi ghi Flash

4.6 Phát triển và Debug

Debug bằng UART: Để debug trong quá trình phát triển, có thể thêm giao tiếp UART:

```

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 25: Ví dụ Debug

Giám sát Máy trạng thái với LED: Dùng một LED để chỉ báo trạng thái hệ thống:

```

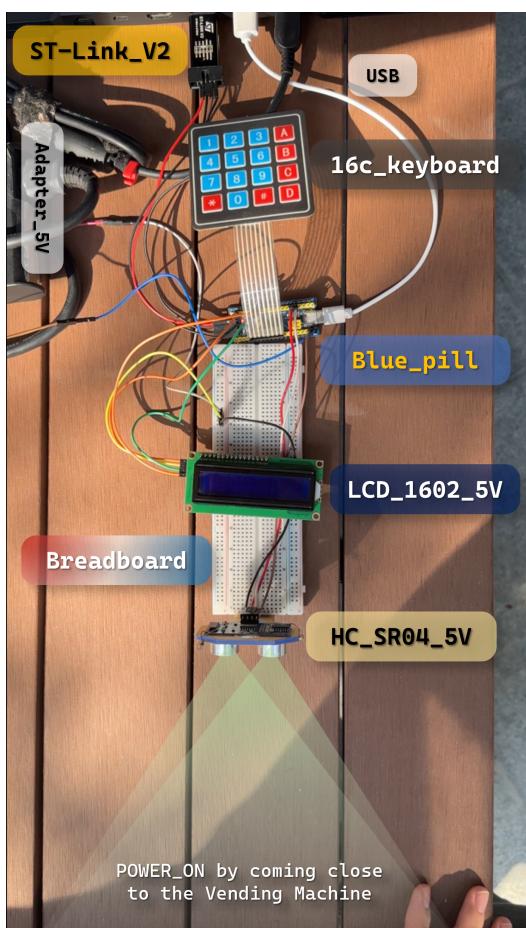
1 void update_status_led(void) {
2     if (status == WAIT_SENSOR) {
3         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET); // Off
4     } else {
5         HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET); // On
6     }
7 }
```

Listing 26: Trạng thái LED

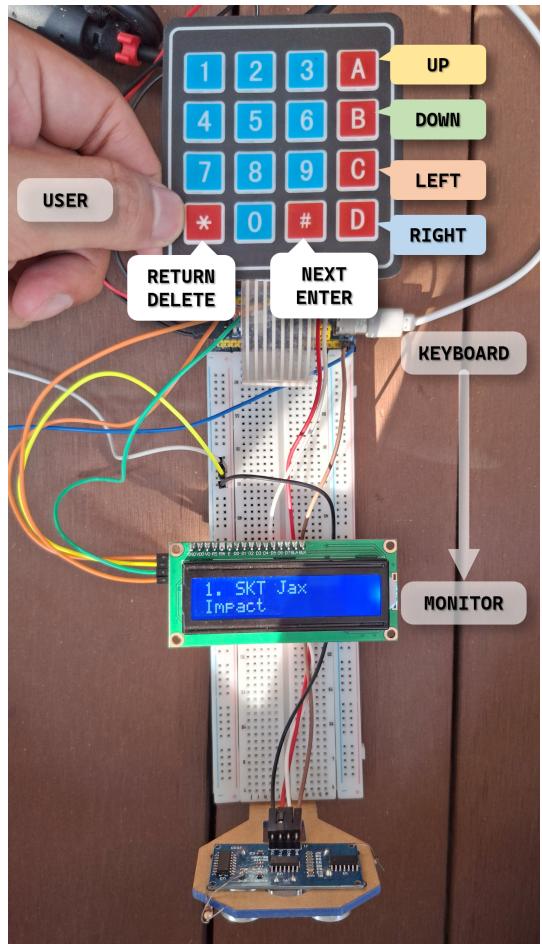
5 Demo

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/EFUxOFmGWq4?si=UAjL1C0VcYWAQK1N>



(a) Demo vận hành hệ thống



(b) Demo vận hành hệ thống

Hình 5: Demo vận hành Máy bán hàng tự động

Tất cả các input gây lỗi đã được test qua:

- Đầu vào số lượng không hợp lệ bị từ chối và yêu cầu thử lại
- Số tiền thanh toán không hợp lệ kích hoạt thông báo lỗi và yêu cầu nhập lại
- Thông báo hết hàng và hướng dẫn người dùng chọn sản phẩm khác
- Hệ thống tự động hủy giao dịch sau 30 giây không hoạt động
- Chế độ quản trị tự động đăng xuất hoạt động chính xác sau 60 giây không hoạt động
- Lỗi quá 5 lần liên tiếp sẽ hủy giao dịch và trả về trạng thái ban đầu

6 Kết luận

6.1 Thành tựu Dự án

- **Tích hợp Phần cứng Hoàn chỉnh:** Giao tiếp thành công với màn hình LCD, bàn phím ma trận, cảm biến siêu âm và bộ nhớ Flash.
- **Hiện thực Máy trạng thái Mạnh mẽ:** Phát triển một FSM với nhiều trạng thái quản lý các luồng giao dịch phức tạp, xử lý lỗi và các chức năng quản trị.
- **Dữ liệu luôn được lưu lại:** Hiện thực lưu trữ tận dụng bộ nhớ Flash với magic number, đảm bảo dữ liệu kho hàng tồn tại dù bị mất điện.
- **Trình điều khiển Ngoại vi Tùy chỉnh:** Tạo các trình điều khiển hiệu quả cho giao tiếp LCD I2C bit-banged, quét bàn phím debouncing và đo khoảng cách.
- **Xử lý Lỗi hiệu quả:** Hiện thực kiểm tra đầu vào, quản lý timeout, bộ đếm lỗi và cơ chế thử lại đảm bảo độ sự thân thiện với người dùng.
- **Tự động Phát hiện người dùng:** Thành công bái bỏ công tắc ON/OFF truyền thống, cung cấp trải nghiệm mới mẻ mà tiết kiệm điện năng.
- **Bảo mật tài khoản Quản trị:** Cung cấp chế độ quản trị được bảo vệ bằng mật khẩu với cơ chế tự đăng xuất kể cả khi quên.

6.2 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 người dùng giảm tiêu thụ năng lượng
- Theo dõi kho hàng cho phép quản lý kinh doanh hiệu quả
- Xử lý lỗi cùng cơ chế thử lại nâng cao trải nghiệm người dùng
- Chế độ quản trị hỗ trợ bảo trì tại chỗ
- Thiết kế module 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.

7 Thông tin Dự án

7.1 GitHub repository

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

- **GitHub:** <https://github.com/1172005thinh/VendingMachine>
- **Source code** Thư mục VendingMachine/ chứa tất cả các tệp nguồn
- **Tài liệu:** README.md với hướng dẫn thiết lập

7.2 Tác giả

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

7.3 Lời cảm ơn

Các tác giả xin bày tỏ lòng biết ơn đến:

- Thầy Nguyễn Thành Lộc đã hướng dẫn các nguyên tắc thiết kế hệ thống luận lý
- Cộng đồng mã nguồn mở về các ví dụ mã và hỗ trợ khắc phục sự cố