

DẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



ĐỒ ÁN

THIẾT KẾ LUẬN LÝ

Đề tài 2: Phát triển Password Door

GVHD: Thầy Phan Đình Thế Duy

Nhóm thực hiện: Nhóm 10
Trần Ngọc Cát - 1912750
Diệp Trần Nam - 1914213



Mục lục

1 Giới thiệu	3
1.1 Access control là gì?	3
1.2 Cơ chế hoạt động của hệ thống access control	3
1.3 Ưu điểm của Access control	3
1.4 Hệ thống Kiểm soát Truy cập được sử dụng ở đâu?	4
2 Cơ sở lý thuyết - công nghệ sử dụng	5
2.1 Linh kiện sử dụng	5
2.2 Công cụ lập trình	5
2.3 Ma trận phím 4x4	5
2.4 Màn hình LCD 16x2 1602A	6
2.4.1 Chức năng của các chân LCD	6
2.4.2 Các vùng nhớ của LCD	6
2.4.3 Khởi tạo LCD	11
2.5 Giao tiếp nối tiếp I2C và DS1307	13
2.5.1 I2C	13
2.5.2 Real Time Clock DS1307	14
2.5.3 Kết nối I2C với mạch	16
2.5.4 Giao tiếp với DS1307 qua I2C	16
2.6 Điều khiển motor bằng L293D	18
2.6.1 L293D	18
2.6.2 Kết nối L293D và motor với mạch	19
3 Hiện thực	20
3.1 Tổng quan	20
3.1.1 Các phím chức năng	20
3.2 Cơ sở dữ liệu của hệ thống tài khoản	20
3.3 Máy trạng thái chính	21
3.4 USER DASHBOARD	22
3.5 ADMIN DASHBOARD	24
3.5.1 MEMBER MANAGE	25
3.5.2 CHECK IN	28
3.5.3 TIME	29
4 Kết quả	32
4.1 Link video demo	32
4.2 Một số hình ảnh thực tế	32
5 Quá trình phát triển	39
5.1 Tuần 1: 13/11 -> 20/11	39
5.2 Tuần 2: 20/11 -> 27/11	39
5.3 Tuần 3: 27/11 -> 4/12	40
5.4 Tuần 4: 4/11 -> 11/12	40
5.5 Tuần 5: 11/11 -> 18/12	40
5.6 Tuần 6: 18/12 -> 25/12	41
5.7 Xin cảm ơn các nhóm đồ án đã hỗ trợ nhóm 10:	41



6 Phân công	42
Tham khảo	43



1 Giới thiệu

1.1 Access control là gì?

Hệ thống kiểm soát ra vào – Access Control là một trong những hệ thống thông dụng nhất hiện nay trong việc kiểm soát ra vào hạn chế một số khu vực định. Có nhiều công nghệ được ứng dụng trong giải pháp kiểm soát ra vào nhưng phổ biến nhất vẫn là :

- Công nghệ thẻ từ (Công nghệ RFID).
- Công nghệ vân tay.
- Công nghệ nhận diện khuôn mặt.
- Công nghệ bằng mã số bàn phím.

Ngoài ra để tăng khả năng giám sát các hệ thống kiểm soát ra vào còn kết hợp liên động như camera ghi hình, hệ thống chấm công, hệ thống chuông cửa có hình (video door phone).

1.2 Cơ chế hoạt động của hệ thống access control

Hoạt động dựa trên công nghệ nhận dạng (thẻ, mật mã hoặc vân tay) và kiểm tra mật khẩu, ID (đã được cài đặt) của tất cả những cá nhân, những người ra vào qua hệ thống kiểm soát:

- Mỗi thành viên sẽ được cấp 1 số ID cụ thể bằng thẻ, bằng một mật mã hoặc bằng chính dấu vân tay của mình làm số ID. Hệ thống kiểm soát vào ra sẽ quản lý và phân quyền dựa trên số ID đã cấp.
- Tất cả những lần vào ra của mỗi thành viên sẽ được lưu lại để kiểm soát (Số ID của cá nhân, ngày giờ vào/ra, cổng, cửa vào/ra, tình trạng vào/ra...)

1.3 Ưu điểm của Access control

- Đảm bảo an ninh cho khu vực sử dụng hệ thống, ngăn ngừa kẻ xâm nhập với mục đích xấu như trộm cướp, phá hoại.
- Thiết lập quyền hạn ra vào cho từng cá nhân, nhóm người.
- Giảm thiểu công việc cho nhân sự trong việc quản lý giờ làm việc của nhân viên.
- Phát hiện những xâm nhập bất hợp pháp. Báo động trong trường hợp khẩn cấp như bị đập phá, hỏa hoạn.
- Nếu xảy ra mất mát có thể dựa vào dữ liệu ra vào để truy cứu trách nhiệm.
- Đề cao tính chuyên nghiệp của công ty. Kết hợp dữ liệu cho việc chấm công cho nhân viên.



1.4 Hệ thống Kiểm soát Truy cập được sử dụng ở đâu?

Hệ thống Kiểm soát Truy cập được sử dụng trong các ứng dụng khác nhau như:

- Khách sạn
- Không gian văn phòng
- Nhiều Đơn vị nhà ở (MDU) như căn hộ và căn hộ
- Các cơ sở thể thao và trung tâm giải trí
- Cửa hàng bán lẻ
- Trung tâm y tế và bệnh viện
- Nhà dưỡng lão
- An ninh ngôi nhà
- Nhà máy và kho bãi

2 Cơ sở lý thuyết - công nghệ sử dụng

2.1 Linh kiện sử dụng

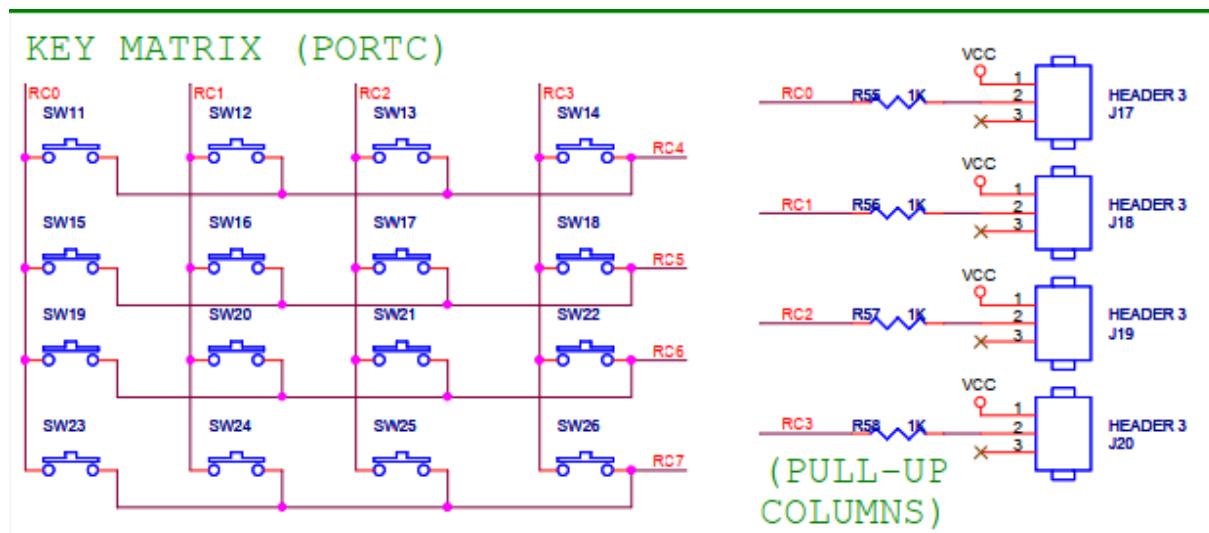
- Board thí nghiệm BKIT PIC
- Còi buzzer 5V
- 2 dây cáp - cáp

2.2 Công cụ lập trình

- MPLAB X IDE v3.26

2.3 Ma trận phím 4x4

Ma trận phím có kết nối như sau:



Hình 1: Sơ đồ kết nối ma trận phím

Như trong thiết kế, ta thấy được RC0, RC1, RC2 và RC3 được gắn với điện trở kéo lên. Đây cũng là những chân sẽ được MCU PIC đọc vào (các chân input). chân RC4, RC5, RC6, RC7 sẽ là các chân output.

Ma trận phím được điều khiển bởi PORTC của MCU PIC.

Ta tạm gọi R0, RC1, RC2, RC3 là các cột, còn RC4, RC5, RC6, RC7 là các hàng. Với thiết kế trên, khi một nút được nhấn, thì cột tương ứng với nút nhấn đó sẽ có điện thế bằng với hàng tương ứng của nút nhấn. Giả sử ban đầu RC0 và RC1 ở mức cao, RC4 ở mức thấp và RC5 ở mức cao, khi nhấn nút SW11, RC0 sẽ có điện thế bằng với RC4, tức RC0 ở mức cao bị kéo xuống mức thấp.

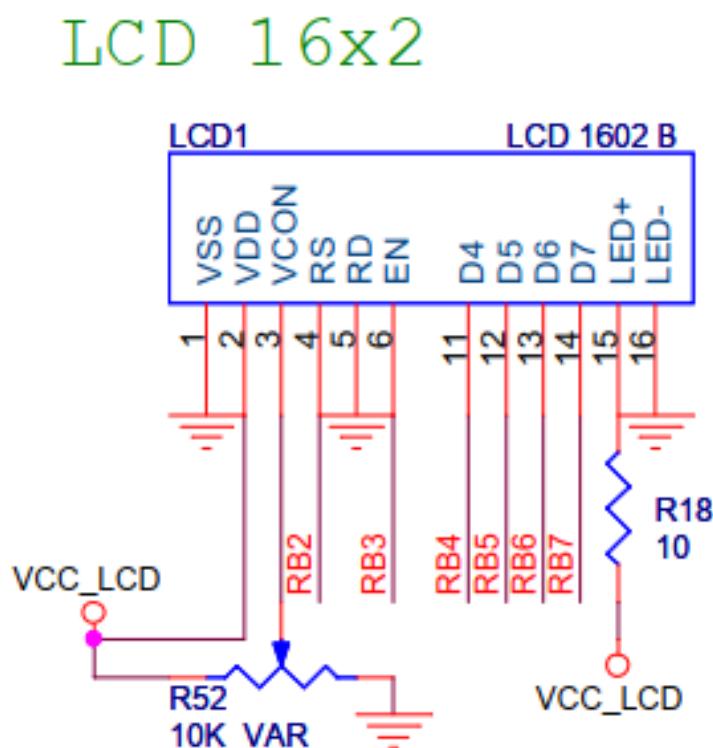
Ý tưởng quét ma trận phím như sau: Tại một thời điểm, ta chỉ xét một hàng phím. Ta điều khiển cho hàng phím đó có điện thế xuống thấp, còn các hàng phím còn lại có điện thế cao. Tại hàng phím đang xét, nếu có nút được nhấn, thì cột tương ứng với vị trí nút đó sẽ có điện thế thấp (có cùng điện thế thấp với hàng). Ta nhận diện nút nhấn bằng

cách hạ điện thế của 1 hàng cụ thể, sau đó đọc xem có cột nào có điện thế thấp không. Nếu có, tức là có nút được nhấn tại hàng đó, ở các cột có điện thế thấp.

key_code[16] là 1 mảng unsigned int có 16 phần tử, đại diện cho 16 nút. Khi có nút được nhấn, key_code tại nút tương ứng sẽ liên tục tăng giá trị lên. Giả sử nút 1 được nhấn, key_code[1] sẽ liên tục tăng giá trị của nó lên cho tới khi thả phím, thì key_code[1] sẽ được reset về 0. Hàm scan_key_matrix được dùng để đọc nút nhấn, hàm sẽ cập nhật nút được nhấn vào mảng key_code. Do đó, để kiểm tra một nút được nhấn hay không, ta chỉ cần kiểm tra key_code[] của nút đó.

2.4 Màn hình LCD 16x2 1602A

Ta sử dụng một màn hình LCD 16x2 để hiển thị. Sơ đồ kết nối với đèn LCD như sau:



Hình 2: Sơ đồ kết nối với LCD

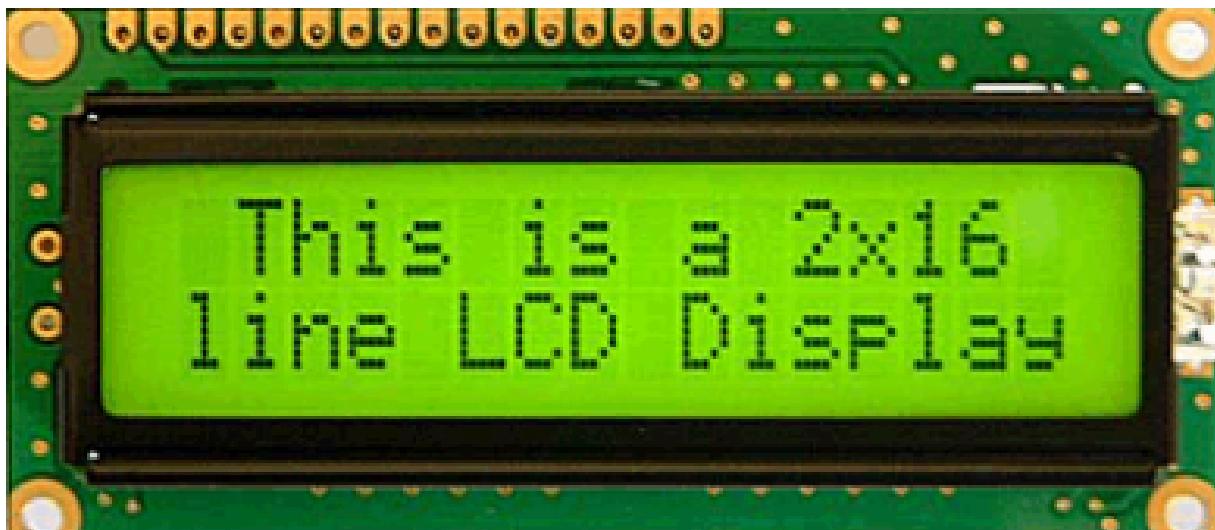
Như trong sơ đồ trên, ta sử dụng portB để điều khiển LCD. LCD hoạt động ở chế độ 4 bits, có kết nối với một biến trỏ để điều chỉnh độ sáng của LCD.

2.4.1 Chức năng của các chân LCD

LCD thường sử dụng 14 chân, chế độ 16 chân khi cần điều khiển đèn nền. Chức năng của các chân như sau:

2.4.2 Các vùng nhớ của LCD

Display Data Ram(DDRAM)



Hình 3: Một LCD 16x2

Chức năng	Thứ tự	Tên	Mức Logic	Mô tả
<i>Ground</i>	1	VSS	-	0V
<i>Power Supply</i>	2	Vdd	-	+5V
<i>Contrast</i>	3	Vee	-	0-Vdd
<i>Control Operation</i>	4	RS	0	D0-D7 là command
			1	D0-D7 là Data
	5	R/W	0	Write
			1	Read
	6	E	0	Disable
			1	Normal
			Từ 1 xuống 0	Truyền Data hoặc Command xuống LCD
<i>Data/Command</i>	7	D0	0 1	Bit 0 LSB
	8	D1	0 1	Bit 1
	9	D2	0 1	Bit 2
	10	D3	0 1	Bit 3
	11	D4	0 1	Bit 4
	12	D4	0 1	Bit 5
	13	D6	0 1	Bit 6
	14	D7	0 1	Bit 7 MSB

Lưu trữ mã ký tự hiển thị ra màn hình. Mã này giống với mã ASCII. Có tất cả 80 ô nhớ DDRAM. Vùng hiển thị tương ứng với cửa sổ gồm 16 ô nhớ hàng đầu tiên và 16 ô nhớ hàng thứ hai. Chúng ta có thể tạo hiệu ứng dịch chữ bằng cách sử dụng lệnh dịch, khi đó cửa sổ hiển thị sẽ dịch kèm theo hiệu ứng dịch chữ.

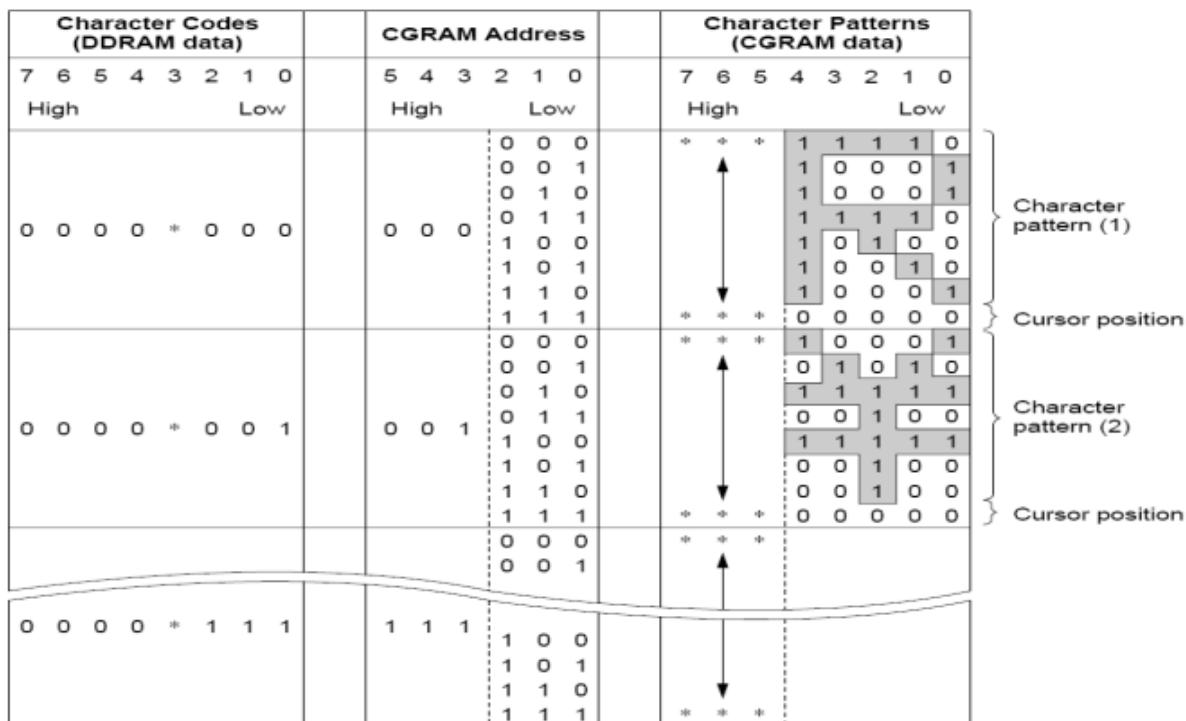
DDRAM Memory



Hình 4: DDRAM trong LCD

Character Generator Ram (CGRAM)

Lưu trữ tám mẫu ký tự do người dùng định nghĩa. Tám mẫu ký tự này tương ứng với các mã ký tự D7-D0 = 0000*D2D1D0 (* mang giá trị tùy định 0 hay 1).



Hình 5: CGRAM trong LCD

Bộ nhớ CGROM

Bộ nhớ dùng để lưu trữ các ký tự hiển thị trên LCD. Các giá trị lưu trong bộ nhớ này như sau:

Có thể thấy giá trị của các ký tự tương ứng với giá trị ASCII của chúng. Để đưa một ký tự ra màn hình LCD, ta cần ghi giá trị của ký tự đó vào ô nhớ từ 0x00 đến 0x0f của hàng thứ nhất (16 ô đầu ở hàng 1) hay ô nhớ 0x40 tới 0x4f (16 ô đầu ở hàng 2).

Các lệnh cơ bản của LCD

Để truyền lệnh cho LCD, ta cần chân RS = 0, sau đó truyền dữ liệu phù hợp vào các chân D0-D7. Bảng sau trình bày ý nghĩa của một số lệnh điều khiển:

	4 higher bits of address															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)				0@P`P						-@E&p					
xxxx0001	(2)		! 1A Qa@								■ アチルäq					
xxxx0010	(3)		" 2B Rbr								「イツ×βθ					
xxxx0011	(4)		# 3C Scs								「ウテモe					
xxxx0100	(5)		\$ 4D Tdt								、エトナ					
xxxx0101	(6)		% 5E Ueu								・オナビ					
xxxx0110	(7)		& 6F Ufv								ヲカニヨ					
xxxx0111	(8)		' 7G W9w								アキヌラ					
xxxx1000	(1)		(8H Xhx								イクネリ					
xxxx1001	(2)) 9I Yiy								レルル					
xxxx1010	(3)		* : JZjz								エコハレ					
xxxx1011	(4)		+ ; Kk (オサヒロ					
xxxx1100	(5)		, < L ¥ 1								アシフワ					
xxxx1101	(6)		- = M] m)								ユスヘン					
xxxx1110	(7)		. > N ^ n →								ミセホ					
xxxx1111	(8)		/ ? O _ o ←								レウマ					

Hình 6: CGROM trong LCD



Lệnh	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Thời gian thực thi
<i>Clear display</i>	0	0	0	0	0	0	0	0	0	1	1.52ms
<i>Return home</i>	0	0	0	0	0	0	0	0	1	*	1.52ms
<i>Entry mode set</i>	0	0	0	0	0	0	0	1	I/D	S	37μs
<i>Display on/off control</i>	0	0	0	0	0	0	1	D	U	B	37μs
<i>Cursor / Display shift</i>	0	0	0	0	0	1	D/C	R/L	*	*	37μs
<i>Function set</i>	0	0	0	0	1	DL	N	F	*	*	37μs
<i>Set CGRAM address</i>	0	0	0	1	CGRAM address						37μs
<i>Set DDRAM address</i>	0	0	1	DDRAM address						37μs	
<i>Read BUSY flag (BF)</i>	0	1	BF	DDRAM address						0μs	
<i>Write to DDRAM or CGRAM</i>	1	0	D7	D6	D5	D4	D3	D2	D1	D0	43μs
<i>Read from DDRAM or CGRAM</i>	1	1	D7	D6	D5	D4	D3	D2	D1	D0	43μs

I/D 1 = Tăng (1 đơn vị)
0 = Giảm (1 đơn vị)

R/L 1 = dịch phải
0 = dịch trái

S 1 = Hiện dịch chuyển
0 = Ẩn dịch chuyển

DL 1 = mode 8-bit
0 = mode 4-bit

D 1 = Bật display
0 = Tắt display

N 1 = Display 2 dòng
0 = Display 1 dòng

U 1 = Bật con trỏ
0 = Tắt con trỏ

F 1 = ký tự ở dạng 5x10
0 = ký tự ở dạng 5x7

B 1 = Bật nháy con trỏ
0 = Tắt nháy con trỏ

D/C 1 = Dịch màn hình display
0 = Dịch con trỏ



2.4.3 Khởi tạo LCD

Để sử dụng LCD, ở trạng thái khởi động, LCD sẽ luôn làm việc ở mode 8 bit. Ta sẽ cần làm một số khởi động trước khi có thể tùy chỉnh chế độ làm việc của LCD thành 4 bit:

- chờ 15ms để Vcc tăng lên 4.5V
- Hạ chân RS xuống mức thấp để gửi các command
- Gửi 4 bit 0011 lên LCD. Vì LCD lúc này đang làm việc ở chế độ 8 bit, lệnh trên sẽ được hiểu là 0011 0000. Để gửi 4 bit, ta đưa dữ liệu vào 4 chân D4->D7, sau đó đưa chân E lên mức cao rồi hạ xuống thấp để gửi 4 bit dữ liệu ở D4->D7 lên LCD. Đây là function set để chạy LCD ở chế độ 8 bit.
- Tiếp tục gửi 4 bit 0011 lên LCD lần thứ 2
- chờ 100 microseconds
- Tiếp tục gửi 4 bit 0011 lên LCD lần thứ 3. Sau lần gửi này, ta có thể tùy chỉnh chế độ làm việc của LCD và kiểm tra được cờ busy (Busy flag - BF)
- Gửi 4 bit 0010 lên LCD. Đây là function set để chuyển LCD sang chế độ làm việc 4 bit. Sau lệnh này, mỗi khi gửi 1 command, ta phải thực hiện gửi 2 lần: lần đầu sẽ gửi 4 bit cao, lần tiếp theo là 4 bit thấp
- Gửi lệnh 0x28: gửi 4 bit 0010 trước, sau đó là 1000, tạo thành lệnh 0010 1000. Lệnh này là 1 function set, thiết lập chế độ làm việc 4 bit, hiển thị ở 2 hàng và ký tự ở dạng 5x7 điểm ảnh (5x7 dots)
- gửi lệnh 0x0c: gửi 4 bit 0000 trước, sau đó là 1100, tạo thành lệnh 0000 1100. Đây là lệnh Display on/off control, chức năng bật display, ẩn cursor và ẩn nháy cursor

Sau khi khởi tạo, ta có thể truyền dữ liệu vào LCD. Lưu ý rằng ta cần phải set tín hiệu RS thành 0 khi muốn dữ liệu từ D0 tới D7 là command, hoặc lên 1 nếu muốn dữ liệu từ D0 tới D7 là data.

Mỗi lần truyền thành công một data 8 bit vào DDRAM để hiển thị ra màn hình LCD, con trỏ sẽ tự dịch sang phải. Để truyền kí tự, vì LCD hoạt động ở chế độ 4 bit, nên ta sẽ cần phải truyền 2 lần dữ liệu 4 bit tương ứng cho 1 kí tự. Mỗi lần truyền 4 bit dữ liệu, ta cho tín hiệu RS lên 1 để cho LCD biết ta đang truyền data, RW xuống 0 để viết dữ liệu vào DDRAM (hoặc CGRAM), đặt 4 bit vào các chân D4 -> D7, sau đó đưa chân E lên điện thế cao rồi hạ chân E xuống điện thế thấp để truyền dữ liệu đi. 4 bit cao sẽ được truyền trước, sau đó là 4 bit thấp.

Ta sẽ cần phải tự chỉnh vị trí ghi lên DDRAM của LCD để đảm bảo các ký tự cần thiết sẽ được hiển thị ra màn hình (các kí tự cần hiển thị phải được ghi vào 16 ô đầu ở hàng 1 và 16 ô đầu ở hàng 2 của LCD theo thiết lập vị trí display ban đầu)

Một số hàm cấp thấp để giao tiếp với LCD:

- **init_lcd()**: khởi động LCD
- **lcd_write_4bits(dat)**: dat là một dữ liệu 8 bit. Hàm này sẽ gửi 4 bit cao của dat lên LCD, bỏ qua 4 bit thấp



- **lcd_write_cmd(cmd)**: Sử dụng lcd_write_4bits trên để lần lượt ghi 4 bit cao và 4 bit thấp của dữ liệu 8 bit cmd lên LCD. chân RS được hạ xuống 0 cho biết đang truyền command cho LCD
- **lcd_write_data(data)**: Sử dụng lcd_write_4bits trên để lần lượt ghi 4 bit cao và 4 bit thấp của dữ liệu 8 bit data lên LCD. chân RS được nâng lên 1 cho biết đang truyền data cho LCD, data này sẽ được ghi vào DDRAM.
- **lcd_print_char(c)**: là 1 hàm wrapper, gọi lại hàm lcd_write_data và truyền ký tự c vào hàm ấy để ghi dữ liệu vào DDRAM
- **lcd_set_cursor(row, column)**: đặt con trỏ LCD ở hàng row, cột column.

Một số hàm cấp cao sau đây được sử dụng để thực hiện các thao tác liên quan tới dữ liệu lên LCD:

- **LcdClearS()** : clear màn hình LCD
- **DisplayLcdScreen()**: lấy dữ liệu từ mảng LcdScreen 2 chiều (16 cột, 2 hàng) tương ứng cho màn hình LCD, ghi vào DDRAM của LCD nhằm hiển thị ra màn hình
- **LcdPrintCharS(x,y,c)**: ghi một ký tự vào mảng LcdScreen tại hàng x, cột y. Cần sử dụng hàm DisplayLcdScreen để đọc mảng LcdScreen và ghi ra màn hình
- **LcdPrintNumS(x,y,num)**: ghi giá trị số vào mảng LcdScreen bắt đầu từ hàng x, cột y. Cần sử dụng hàm DisplayLcdScreen.
- **LcdPrintStringS(x,y,*string)**: ghi một chuỗi string vào mảng LcdScreen bắt đầu từ hàng x, cột y. Cần sử dụng hàm DisplayLcdScreen.
- **LcdPrintLineS(x, *string)**: ghi một chuỗi string vào mảng LcdScreen bắt đầu từ hàng x, cột 0. Nếu độ dài chuỗi ít hơn 16 ký tự, các ký tự sau sẽ là khoảng trắng. Mục đích của hàm là ghi một chuỗi lên cả 1 hàng LCD. Cần sử dụng hàm DisplayLcdScreen.

Ngoài ra, nhóm em còn thiết kế thêm 1 hàm để tạo ký tự custom như sau:

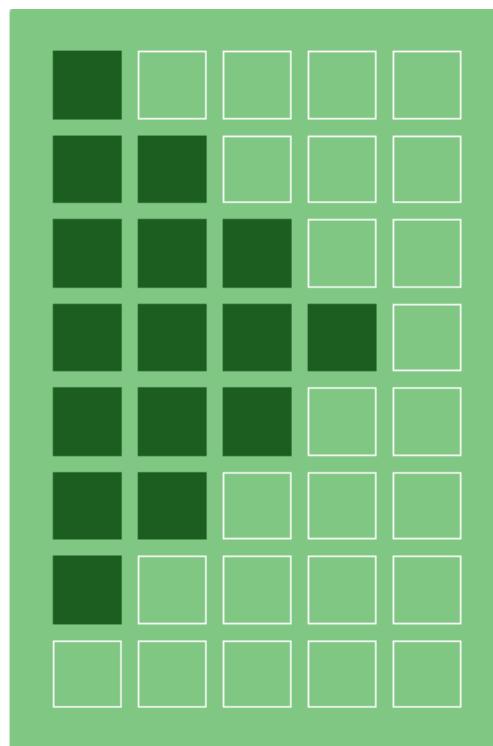
- **LcdCreateChar(unsigned char* char_code_arr, unsigned char cgram_address)**

CGRAM là nơi để ta thêm vào các ký tự custom. Trong CGRAM có 8 block, mỗi block 8 byte, điều đó có nghĩa là ta có thể tạo tối đa 8 ký tự custom. Để tạo ký tự custom, ta cần phải "vẽ" hình dáng ký tự vào một trong 8 block này. Vì LCD của ta có ký tự ở dạng 5x7, nên khi "vẽ" ký tự vào block, ta chỉ quan tâm tới 7 byte đầu của block, và mỗi byte chỉ cần chú ý tới 5 bit trọng số thấp bên phải.

Các bước để tạo 1 ký tự custom như sau:

- Gửi lệnh lên LCD, chọn 1 block trong CGRAM (lệnh Set CGRAM address)
- Lần lượt "vẽ" lên block đó bằng cách truyền từng byte vào block. Mỗi byte tương ứng với 1 hàng. Giả sử, ta cần vẽ ký tự mũi tên bên phải (hình 7).

Tương ứng với các ô in đậm là bit 1, vậy thì ta sẽ lần lượt truyền vào từng hàng của block này một giá trị trong mảng sau:



Hình 7: Kí tự mũi tên phải

```
1     unsigned char customChar[8] = {0x10, 0x18, 0x1c, 0x1e, 0x1c, 0x18, 0
2         x10, 0x00};
```

Giả sử ta chọn block 1, địa chỉ của block sẽ là **0b00|001_000**. Địa chỉ của byte đầu tiên trong block là **0b00|001_000**, ta sẽ truyền giá trị customChar[0] vào byte này. Byte tiếp theo của block là **0b00|001_001**, sẽ được truyền vào giá trị của customChar[1]... Tương tự như vậy, cho đến khi hết 8 phần tử của mảng customChar, tương ứng với 8 byte mỗi block. Vì mục đích của ta là tạo 1 kí tự 5x7, ta thường để trống byte cuối (0x00).

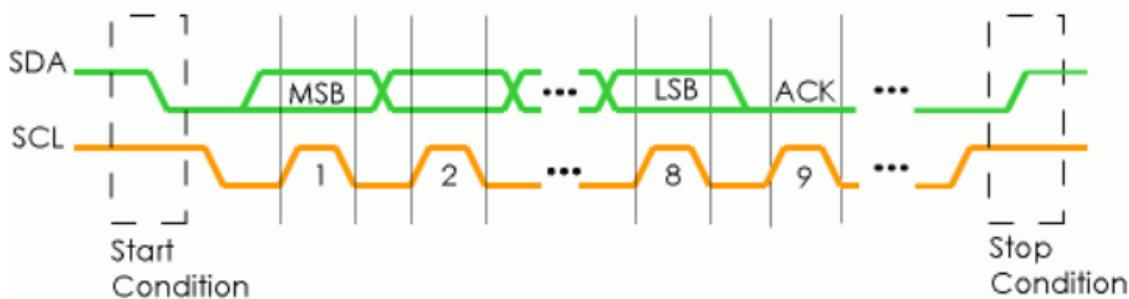
2.5 Giao tiếp nối tiếp I2C và DS1307

2.5.1 I2C

I2C là 1 chuẩn truyền nối tiếp theo mô hình Master – Slave. Một Master có thể giao tiếp với nhiều Slave. Muốn giao tiếp với slave nào, master phải gửi đúng địa chỉ để tích cực slave đó rồi mới được phép ghi hoặc đọc dữ liệu từ slave

Bus I2C gồm 2 dây tín hiệu SCL (Serial Clock Line) và SDA (Serial Data Line) đều được kéo lên nguồn. Dữ liệu được truyền từng bit ở SDA theo từng clock của SCL.

Trước khi truyền dữ liệu, ta cần khởi động I2C bằng cách kéo lần lượt SDA và SCL xuống mức thấp. Sau đó 8 bit dữ liệu sẽ được ra tuân tự theo từng cạnh xuống ở chân SCL (dữ liệu ở SDA sẽ được truyền đi khi SCL lên cao rồi xuống thấp). Clock thứ 9 sẽ dành cho bit ACK. Bit ACK này có thể là do master gửi xuống hoặc do slave gửi về. Bit ACK là 0 nếu dữ liệu thành công, ngược lại nếu là 1 (NACK), tức đã có lỗi xảy ra hoặc một vài vấn đề khác và cần được xử lý. bit ACK thường được trả về từ receiver mỗi khi



Hình 8: Giao thức I²C

nhận được 1 byte (8 bit) dữ liệu. Khi kết thúc giao tiếp I²C, ta phải stop nó bằng cách kéo 2 chân SCL và SDA lên mức cao.

2.5.2 Real Time Clock DS1307

DS1307 là IC thời gian thực (Real time clock) đếm giờ, phút, giây, tháng, ngày của tháng, ngày của tuần, năm kể cả năm nhuận (đến năm 2100).

56 byte Ram để lưu trữ dữ liệu, nhưng dữ liệu không bị mất khi tắt nguồn.

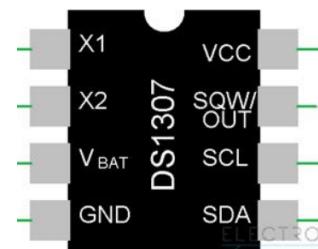
Sử dụng 2 dây tín hiệu để truyền dữ liệu theo giao thức I²C. Có thể lập trình được để xuất tín hiệu xung vuông.

Tự động phát hiện ra nguồn cung cấp bị lỗi (ngắt nguồn) và chuyển qua mạch bảo vệ sử dụng nguồn pin dự trữ.

2.5.2.1 Nguyên lý hoạt động

DS1307 hoạt động như một slaver trên bus dữ liệu nối tiếp. Để truy xuất nội dung ta phải thiết lập một điều kiện Start và cung cấp mã nhận dạng của IC (Device Identification Code) theo sau bởi thanh ghi địa chỉ. Các thanh ghi theo sau được truy xuất tuần tự cho đến khi gặp tín hiệu Stop.

Khi VCC = 1.25Vbat thì DS1307 sẽ kết thúc việc truy xuất và reset lại bộ đếm địa chỉ. Các Input sẽ không được nhận ra tại thời điểm này để ngăn ngừa một số lượng lớn dữ liệu được ghi tới DS1307 từ hệ thống bên ngoài. Khi VCC < Vbat thì ic này sẽ chuyển sang mode sử dụng pin dự trữ. Khi nguồn chính được bật lên thì IC này sẽ chuyển từ dùng nguồn pin sang dùng nguồn chính.



2.5.2.2 Các tín hiệu Input và Output

VCC, GND : Nguồn DC được cung cấp cho IC qua những chân này. Khi gắn vào nguồn 5V thì IC này có thể đọc ghi bình thường. Nhưng khi nguồn giảm xuống còn 3V thì việc đọc ghi sẽ không được phép. Tuy nhiên, các chức năng của timer vẫn tiếp tục với nguồn cung cấp thấp. Khi Vcc giảm xuống dưới VBAT thì RAM và timekeeper được chuyển qua sử dụng nguồn cung cấp tại VBAT.

VBAT : Cung cấp nguồn dữ trữ 3V. Để hoạt động ở chế độ sử dụng nguồn Vbat thì $2.0V < V_{bat} < 3.5V$. Khi VCC gần bằng 1.25VBAT thì chúng ta sẽ không được phép truy xuất vào RTC (Real time clock) và Ram bên trong của IC. **SCL (Serial Clock Input)** : SCL được dùng để đồng bộ dữ liệu trên đường truyền nối tiếp.



SDA (Serial Data Input/Output) : SDA là chân I/O. SDA là chân Open drain nên cần có điện trở kéo lên ở bên ngoài.

SQW/OUT (Square Wave/Output Driver) : Khi được bật lên, thì bit SQWE set lên 1, và chân này sẽ output ra 1 trong 4 tần số sóng vuông là 1hz, 4khz, 8khz, 32khz. Chân này cũng là chân Open drain nên cũng yêu cầu có điện trở kéo lên nguồn ở bên ngoài. SQW/OUT sẽ hoạt động khi có nguồn cung cấp vào cho dù đó là nguồn VCC hay là VBAT.

X1, X2 : Kết nối với thạch anh 32.768Khz. Mạch tạo xung bên trong được thiết kế để hoạt động với thạch anh và tụ CL = 12.5 pF.

2.5.2.3 RTC và sơ đồ địa chỉ của RAM

Sơ đồ địa chỉ của RTC và các thanh ghi Ram của DS1307 như ở hình dưới. Các thanh ghi RTC được định địa chỉ từ 00h đến 07h. Các thanh ghi Ram được định địa chỉ tiếp theo sau đó và từ 08h đến 3fh. Trong khi truy suất nhiều byte và khi con trỏ địa chỉ chỉ tới ô 3fh, vị trí cuối của vùng nhớ Ram, thì nó sẽ quay lại địa chỉ 00h để truy xuất tiếp.

2.5.2.4 Thông tin thời gian và lịch

Thông tin thời gian và lịch được chứa trong các thanh ghi tương ứng. Các thanh ghi RTC như ở hình trên. Thời gian và lịch được set hoặc khởi tạo bằng cách ghi ra các byte thanh khi tương ứng. Nội dung của các thanh ghi thời gian và lịch được định dạng theo kiểu BCD (bit từ 6 tới 4 là 3 bit BCD dành cao, còn bit từ 4 tới 0 là 4 bit dành cho BCD thấp - ví dụ, giá trị giây 23 sẽ được lưu từ bit 6 tới bit 0 là : 010 0011 trong thanh ghi 0x00h - thanh ghi chứa giá trị giây). Bit 7 của thanh ghi 0 là clock halt bit (CH). Khi bit này được set lên 1 thì mạch dao động sẽ bị ẩn không được sử dụng nữa, khi clear xuống 0 thì mạch dao động sẽ được kích hoạt trở lại.

DS1307 có thể chạy ở chế độ 12h hay 24h. Bit thứ 6 của thanh ghi hours được định nghĩa để set xem sử dụng IC này ở chế độ nào. Khi bit này bằng 1 thì chế độ 12h được chọn.

Trong chế độ 12h thì bit 5 chỉ AM/PM (PM khi bit này là 1). Trong chế độ 24h, thì bit 5 là bit thứ 2 của 10hour (20:23)

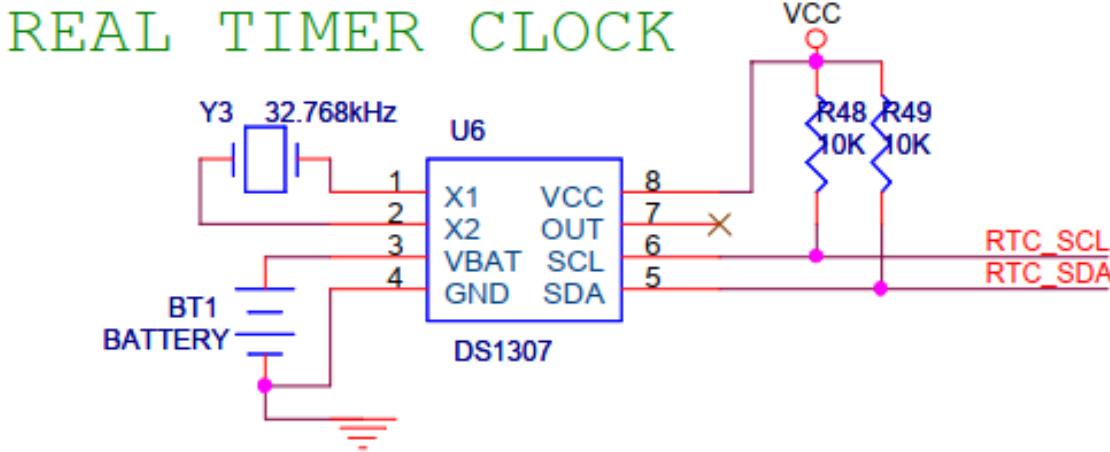
Ds1307 Register Memory Map

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE			
00h	CH		10 Seconds		Seconds				Seconds	00–59			
01h	0		10 Minutes		Minutes				Minutes	00–59			
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23			
		24	PM/AM										
03h	0	0	0	0	0	DAY			Day	01–07			
04h	0	0	10 Date		Date				Date	01–31			
05h	0	0	0	10 Month	Month				Month	01–12			
06h	10 Year				Year				Year	00–99			
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—			
08h–3Fh									RAM 56 x 8	00h–FFh			

Hình 9: Thông tin thời gian và lịch trong các thanh ghi

2.5.3 Kết nối I2C với mạch

Kết nối của I2C với mạch như sau:



Hình 10: Sơ đồ kết nối I2C

Trong sơ đồ, chân SCL sẽ được nối với RC3, và SDA được nối với chân RC4 của vi điều khiển. Ta sẽ điều khiển và giao tiếp với DS1307 thông qua 2 chân RC4 và RC3.

2.5.4 Giao tiếp với DS1307 qua I2C

DS1307 có địa chỉ là 1101000 (7 bits). Ta sẽ dùng 7 bit địa chỉ này để chọn slave và giao tiếp với DS1307 thông qua MCU

2.5.4.1 Điều kiện khởi động - start condition

Đây là điều kiện để bắt đầu giao tiếp với DS1307 thông qua I2C. Hàm **i2c_start_cond()** dùng để tạo điều kiện khởi động bằng cách **kéo chân SDA xuống 0, sau đó kéo chân SCL xuống 0**. Hàm này cũng có thể tạo 1 điều kiện khởi động lại bằng cách đưa chân SDA lên 1 rồi chân SCL lên 1 trước khi kéo 2 chân ấy xuống 0. Đồng thời cờ **isStarted** được bật lên 1 để cho biết quá trình giao tiếp đã bắt đầu.

2.5.4.2 Điều kiện kết thúc - end condition

Đây là điều kiện để kết thúc giao tiếp với DS1307 thông qua I2C. Hàm **i2c_stop_cond()** dùng để tạo điều kiện kết thúc bằng cách **kéo chân SDA xuống 0, cho chân SCL lên 1 và sau đó là chân SDA lên 1**, hay nói cách khác là **kéo chân SDA lên 1 sau khi cho chân SCL lên 1**. Đồng thời cờ **isStarted** cũng kéo xuống 0, cho biết quá trình giao tiếp đã kết thúc

2.5.4.3 Gửi một byte lên DS1307

Hàm **i2c_write_byte** dùng để gửi từng bit của byte dữ liệu lên DS1307. Ta sử dụng 1 vòng lặp, lặp 8 lần, mỗi lần lặp lấy ra từng bit của byte đưa vào chân SDA, Sau đó kéo chân SCL lên cao để dữ liệu (1 bit) được chuyển qua DS1307, rồi kéo chân SCL xuống thấp. Sau khi hoàn tất chuyển 8 bit dữ liệu, ta chỉnh lại chân SDA thành chân input và nâng SCL lên 1 để nhận lại bit ACK từ DS1307 với hàm **read_sda_pin()**. Sau khi nhận



lại bit ACK, ta chỉnh lại chân SDA thành chân output để tiếp tục điều khiển thông qua MCU.

2.5.4.4 Nhận một byte từ DS1307

Hàm `i2c_read_byte` dùng để nhận 1 byte từ DS1307, ta tiến hành chỉnh chân SDA thành chân input, rồi dùng vòng lặp 8 lần, lần lượt nâng chân SCL lên cao rồi hạ xuống để nhận từng bit từ DS1307 thông qua chân input SDA cho tới khi đủ 8 bit. Sau khi nhận xong, ta chỉnh lại chân SDA thành chân output, và gửi lại tín hiệu ACK hoặc NACK thông qua chân SDA lại cho DS1307.

2.5.4.5 Viết dữ liệu lên DS1307

Sử dụng hàm gửi một byte lên DS1307 (`i2c_write_byte`), hàm `write_ds1307` lần lượt làm các bước sau để gửi dữ liệu theo gói tin lên DS1307:

- Tạo điều kiện khởi động
- Gửi 1 byte dữ liệu lên DS1307, là địa chỉ của DS1307 (110100) kết hợp với 1 bit 0 (yêu cầu write) để DS1307 biết ta muốn ghi dữ liệu. Thao tác này giúp cho master (MCU) xác nhận slave mà nó muốn giao tiếp.
- Gửi 1 byte dữ liệu, lần này là địa chỉ vùng nhớ trong DS1307 mà ta muốn ghi vào.
- Gửi 1 byte dữ liệu, đây là dữ liệu sẽ được ghi vào vùng nhớ đã được xác nhận ở bước trước
- Tạo điều kiện kết thúc

2.5.4.6 Đọc dữ liệu từ DS1307

Thao tác đọc dữ liệu sẽ cần thêm 1 vài bước, chủ yếu để xác định vị trí dữ liệu cần phải đọc. Hàm `read_ds1307` được sử dụng để đọc dữ liệu theo các bước sau:

- Tạo điều kiện khởi động
- Gửi 1 byte dữ liệu lên DS1307, là địa chỉ của DS1307 (110100) kết hợp với 1 bit 0 (yêu cầu write) để DS1307 biết ta muốn ghi dữ liệu. Thao tác này giúp cho master (MCU) xác nhận slave mà nó muốn giao tiếp.
- Gửi 1 byte dữ liệu, lần này là địa chỉ vùng nhớ trong DS1307 mà ta muốn ghi vào. Thực chất đây là thao tác giúp DS1307 hiểu là mình đã chọn địa chỉ vùng nhớ này, nhằm phục vụ cho thao tác đọc lúc sau
- Tạo một điều kiện khởi động (tái khởi động lại)
- Gửi lại 1 byte dữ liệu bao gồm địa chỉ của DS1307 (110100) và kết hợp với 1 bit 1 (yêu cầu đọc) để đọc dữ liệu từ DS1307, đọc từ địa chỉ vùng nhớ ta đã xác nhận trước đó
- sử dụng hàm `i2c_read_byte()` để đọc byte từ DS1307
- Tạo điều kiện kết thúc

2.6 Điều khiển motor bằng L293D

2.6.1 L293D

L293D về cơ bản là một IC trình điều khiển hay bộ điều khiển động cơ. Nó có hai mạch cầu H tích hợp có thể điều khiển đồng thời hai động cơ DC theo cả chiều kim đồng hồ và ngược chiều kim đồng hồ. Nó hoạt động như một bộ khuếch đại dòng điện cao vì nó lấy tín hiệu dòng điện thấp ở đầu vào và cung cấp tín hiệu dòng điện cao hơn ở đầu ra để điều khiển các tải khác nhau, ví dụ động cơ bước và động cơ DC. Các tính năng của nó bao gồm phạm vi điện áp nguồn đầu vào lớn, tín hiệu đầu vào chống nhiễu cao dòng điện đầu ra lớn,... Các ứng dụng thực tế phổ biến của nó bao gồm trình điều khiển động cơ bước, trình điều khiển relay, trình điều khiển động cơ DC,...

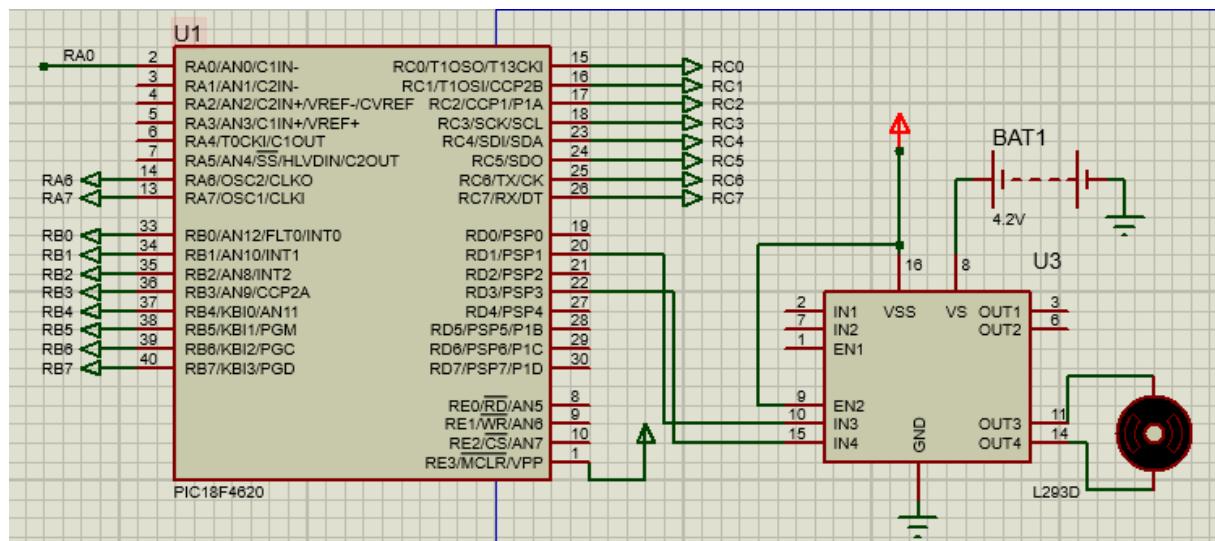


Hình 11: Sơ đồ chân của L293D

Số chân	Tên chân	Mô tả
1	Enable 1,2	Chân này bật chân đầu vào Input 1 (2) và Input 2 (7)
2	Input 1	Trực tiếp điều khiển chân Output 1. Điều khiển bằng mạch kỹ thuật số
3	Output 1	Được kết nối với một đầu của động cơ 1
4	Ground	Chân ground được nối với mass của mạch (0V)
5	Ground	Chân ground được nối với mass của mạch (0V)
6	Output 2	Được kết nối với một đầu khác của động cơ 1
7	Input 2	Điều khiển trực tiếp chân Output 2. Điều khiển bằng mạch kỹ thuật số
8	Vcc2 (Vs)	Kết nối với chân điện áp để chạy động cơ (4,5V đến 36V)
9	Enable 3,4	Chân này bật chân đầu vào Input 3 (10) và Input 4 (15)
10	Input 3	Điều khiển trực tiếp chân Output 3. Điều khiển bằng mạch kỹ thuật số
11	Output 3	Được kết nối với một đầu của động cơ 2
12	Ground	Chân ground được nối với mass của mạch (0V)
13	Ground	Chân ground được nối với mass của mạch (0V)
14	Output 4	Được kết nối với một đầu khác của động cơ 2
15	Input 4	Trực tiếp điều khiển chân Output 4. Điều khiển bằng mạch kỹ thuật số
16	Vcc2 (Vss)	Kết nối với + 5V để bật chức năng IC

2.6.2 Kết nối L293D và motor với mạch

Sử dụng L293D, nhóm em kết nối giữa motor và chip pic theo sơ đồ sau:



Hình 12: Sơ đồ kết nối L293D và motor

Chân RC1 và RC3 được nối với EN3 và EN4 của chip để điều khiển chiều quay của động cơ thông qua OUT3 và OUT4.

Nếu EN3 và EN4 đều là mức cao hoặc mức thấp, động cơ sẽ dừng. Nhưng nếu 1 trong 2 chân là mức cao, output sẽ cho ra tương tự: Nếu IN3 ở mức cao, IN4 ở mức thấp thì OUT3 sẽ ở mức cao và OUT4 ở mức thấp, động cơ quay theo chiều kim đồng hồ. Ngược lại, nếu IN4 ở mức cao, IN3 ở mức thấp thì OUT4 sẽ ở mức cao và OUT3 ở mức thấp, động cơ quay ngược chiều kim đồng hồ.

VSS là mức điện thế cấp cho chip hoạt động, là 5V. VS là điện thế cần cho motor quay. Nhóm em sử dụng một cục pin 4.2V để cấp cho motor RE 260 thông qua L293D. Chân EN2 được nối với nguồn 5V nên sẽ luôn ở trạng thái on, điều này có nghĩa là động cơ sẽ luôn chạy khi 1 trong 2 chân IN3 hoặc IN4 ở mức cao theo chiều kim đồng hồ hoặc ngược kim đồng hồ. Nhưng Nếu IN3 hoặc IN4 đều ở mức thấp (hoặc mức cao), động cơ sẽ dừng.

Khi sử dụng motor này, mục đích của nhóm em là kết hợp nó với một hệ thống bánh răng nhằm giảm tốc độ quay của đầu ra, và sử dụng đầu ra đó để điều khiển một cửa cuốn mini nhằm minh họa cho hoạt động mở cửa/dóng cửa. Vì sử dụng nguồn điện cấp cho motor là 4.2V, cùng với hệ thống bánh răng được sắp xếp nhằm giảm tốc độ đầu ra, chúng em đã điều khiển được việc đóng mở cửa với tốc độ vừa phải: việc đóng cửa/mở cửa sẽ diễn ra trong vòng 2 giây cho mỗi hoạt động mà không cần sử dụng tín hiệu PWM



3 Hiện thực

3.1 Tổng quan

Để tăng tính bảo mật và dễ dàng kiểm soát thì *hệ thống kiểm soát cửa ra vào bằng mật khẩu* (Password Door) của chúng ta sẽ xác thực qua 2 lớp bảo mật, đó là: **xác thực ID** và sau đó **xác thực mật khẩu**.

Mỗi người dùng sẽ được cung cấp một ID riêng biệt để hệ thống dễ dàng quản lý. Ngoài ra, vì hệ thống có 2 loại tài khoản: **admin** và **user**, và mỗi loại tài khoản có các chức năng và các quyền hạn khác nhau nên chúng ta cần phải phân biệt 2 loại tài khoản này. Để làm được điều đó, chúng ta tận dụng ID đã được cung cấp cho người dùng và quy ước rằng: Nếu **ID bằng 0** thì đây chính là tài khoản **admin**, còn lại các **ID khác 0** sẽ là tài khoản **user**.

3.1.1 Các phím chức năng

Hệ thống sử dụng ma trận phím 4x4 để phục vụ các tính năng như sau:

- **Các phím số** được sử dụng khi nhập mật khẩu hay ID.
- **Phím * và #** thường được dùng để di chuyển qua lại giữa các trang tính năng. **Phím 0** dùng để chọn 1 tính năng của trang.
- Một số trường hợp, **phím *** giúp quay lại trạng thái trước đó, **phím #** dùng để xác nhận, chấp nhận khi nhập ID, mật khẩu, thay đổi mật khẩu, tạo thành viên mới, xóa thành viên, thay đổi mật khẩu thành viên.
- **Phím A** cho phép xóa đi dữ liệu khi nhập sai lúc nhập ID hay mật khẩu
- **Phím 1, 2, 3, 4** được sử dụng để chọn thành viên trong chức năng xóa thành viên, thay đổi mật khẩu thành viên.
- **Phím 1, 2, 3** được sử dụng trong trường hợp chỉnh sửa thời gian. Phím 3 dùng để chọn mode, trong khi phím 1 để tăng giá trị và phím 2 để giảm giá trị.

3.2 Cơ sở dữ liệu của hệ thống tài khoản

Để quản lý các tài khoản trong hệ thống, nhóm em sử dụng cơ sở dữ liệu sau:

```
1 #define MAX_ACCOUNT 200
2
3 typedef struct user_account {
4     unsigned int ID;
5     unsigned char password[PASSWORD_LENGTH];
6     enum Check_in checkin;
7 } user_account;
8
9 #pragma iodata large_idata
10 user_account accounts[MAX_ACCOUNT] = {
11     {0, {1,2,3,4,5,6}, VANG},
12     {1, {2,7,8,9,7,8}, VANG},
13     {2, {2,2,2,2,2,2}, VANG},
```

```

14     {3, {3,3,3,3,3,3}, VANG},
15     {4, {4,4,4,4,4,4}, VANG},
16     {5, {5,5,5,5,5,5}, VANG},
17     {6, {6,6,6,6,6,6}, VANG},
18     {7, {7,7,7,7,7,7}, VANG},
19     {8, {8,8,8,8,8,8}, VANG},
20     {9, {9,9,9,9,9,9}, VANG},
21     {10, {1,0,1,0,1,0}, VANG},
22     {11, {1,1,1,1,1,1}, VANG}
23 };
24 #pragma iodata
25 //doi voi object lon, phai dung con tro de dieu khien.
26 user_account * account = accounts;

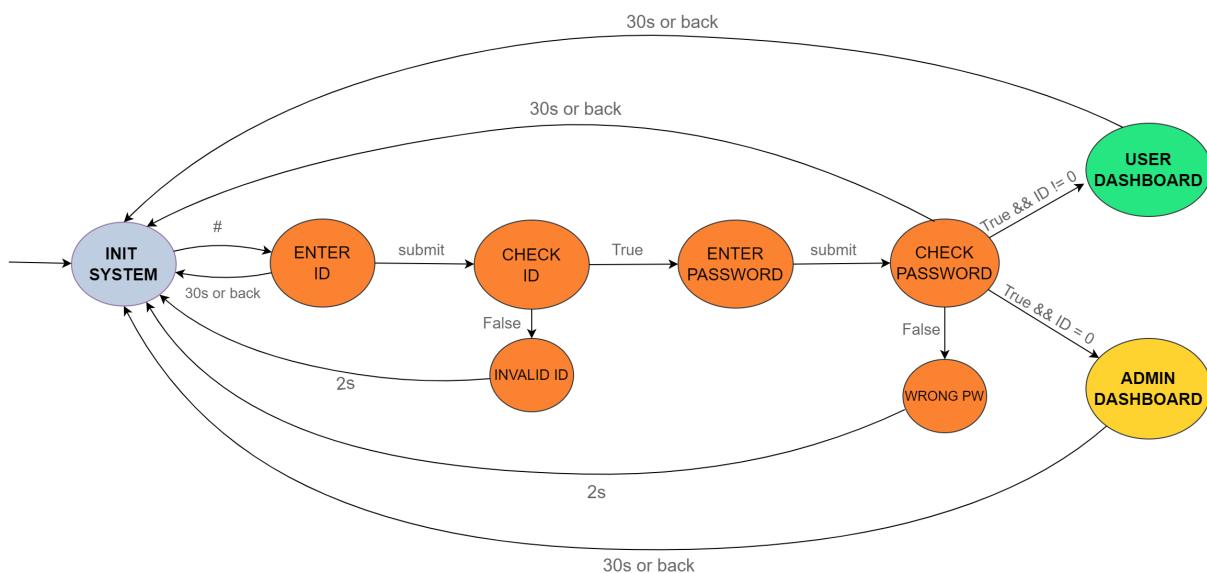
```

Các tài khoản sẽ được lưu trong 1 mảng tĩnh accounts kiểu user_account gồm 200 phần tử. Mỗi 1 phần tử chứa 3 thông tin chính của 1 tài khoản được khai báo trong **struct user_account** là **ID** của tài khoản đó, 1 mảng char **password** gồm 6 phần tử lưu trữ 6 ký số cho mật khẩu của mỗi tài khoản, và biến checkin lưu trữ thông tin hiện diện, trễ hay vắng mặt của tài khoản.

Khi khai báo mảng accounts gồm 200 phần tử, vì kích thước của mảng tĩnh này khá lớn (khoảng 2800 bytes), trong khi đó PIC18 chỉ cho phép khai báo một kiểu dữ liệu tối đa khoảng 256 byte, nên nhóm em đã thực hiện chỉnh sửa file linker, gộp nhiều bank memory lại sao cho đủ 2800 byte, đặt tên nó là **large_idata** và sử dụng **#pragma** để khởi tạo mảng accounts sử dụng vùng không gian 2800 bytes đã đặt trong file linker, vượt qua giới hạn 256 bytes của PIC18.

Nhóm em cũng tạo 1 con trỏ kiểu **user_account** tên **account** trỏ tới mảng **accounts**, và sử dụng con trỏ này để truy xuất vào mảng trong các đoạn code hiện thực tính năng phía sau.

3.3 Máy trạng thái chính



Hình 13: FSM của chính của hệ thống

Đây là máy trạng thái chung của hệ thống. Ban đầu, hệ thống sẽ ở trạng thái **INIT**



SYSTEM - cửa đang đóng, các chức năng đều bị khóa. Để tương tác với hệ thống, chúng ta nhấn phím #.

Tiếp theo, cũng là bước quan trọng nhất đó chính là xác thực người dùng:

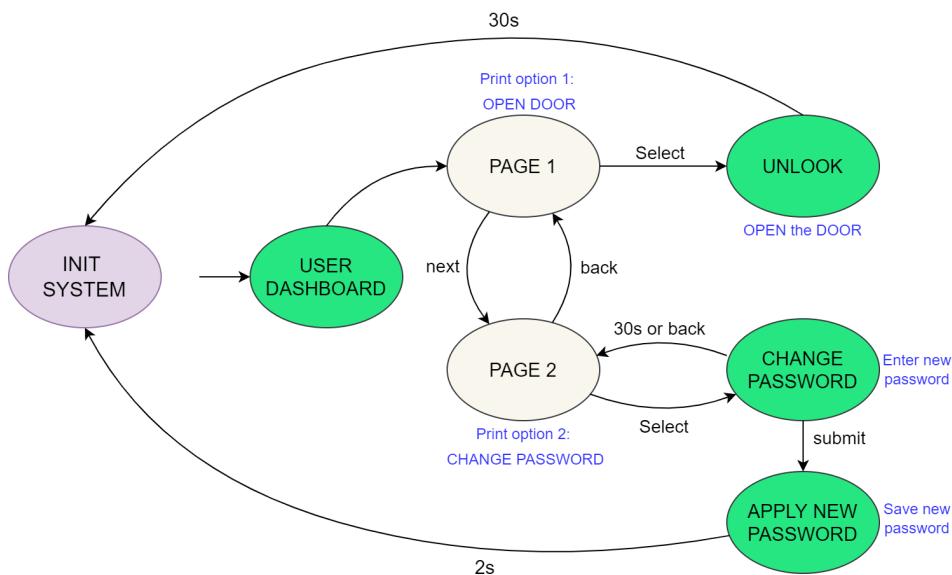
- Đầu tiên, Tại trạng thái **ENTER ID**, người dùng cần nhập vào ID của mình. ID là một số có tối đa 3 chữ số, được hệ thống cấp cho người dùng khi tạo tài khoản. Trong quá trình nhập ID, người dùng có thể tiến hành submit ID của mình để hệ thống kiểm tra khi nhập đủ 3 chữ số hay chưa đủ nhưng nhấn nút #. Từ trạng thái **ENTER ID**, sau khi submit, hệ thống sẽ tiến hành so sánh ID vừa nhập với dữ liệu có trong hệ thống, xem có người dùng nào với ID ấy tồn tại hay không. Nếu không, sẽ chuyển sang trạng thái Invalid ID, in thông báo ra màn hình kèm 2 tiếng bip báo hiệu và quay ngược lại về **INIT SYSTEM**. Nếu ID có tồn tại, tiến hành chuyển sang trạng thái tiếp theo: nhập mật khẩu (**ENTER PASSWORD**)
- Tiếp theo là bước xác thực mật khẩu, người dùng được yêu cầu nhập mật khẩu tương ứng với ID đã nhập - trạng thái **ENTER PASSWORD**, ở trạng thái này, nếu người dùng không tương tác với hệ thống quá 30 giây thì hệ thống sẽ tự động thoát ra (quay về **INIT SYSTEM**), hoặc nếu không muốn tiếp tục nữa thì người dùng có thể chủ động thoát ra bằng cách nhấn nút quay về (phím *). Mật khẩu là một chuỗi kí số gồm 6 kí số, trong quá trình nhập, nếu nhập sai, người dùng có thể dùng nút A để xóa kí số bị sai. Ngay sau khi nhập kí số thứ 6, hệ thống sẽ ngay lập tức chuyển sang trạng thái Check Password.

Sau khi nhập mật khẩu và **submit**, hệ thống sẽ kiểm tra xem mật khẩu vừa nhập có đúng hay không - trạng thái **CHECK PASSWORD**. Nếu mật khẩu không đúng thì hiển thị ra màn hình LCD "**WRONG PW**" trong 2 giây rồi quay về lại trạng thái **INIT SYSTEM**. Còn nếu mật khẩu đúng, thì chúng ta sẽ có 2 trường hợp:

- Với **ID = 0** - đây chính là tài khoản **admin**, hệ thống sẽ đưa chúng ta đến giao diện của admin - **ADMIN DASHBOARD**.
 - Ngược lại, với **ID != 0** - đây chính là tài khoản **user**, hệ thống sẽ đưa chúng ta đến giao diện của user - **USER DASHBOARD**.
- Tương tự, nếu khi đã vào được giao diện của admin hay user rồi nhưng người dùng không thực hiện bất cứ thao tác gì (không tương tác với hệ thống) trong 30 giây thì hệ thống sẽ tự khóa lại (quay về **INIT SYSTEM**) hoặc người dùng cũng có thể chủ động rời khỏi hệ thống bằng cách nhấn nút quay về (phím *).

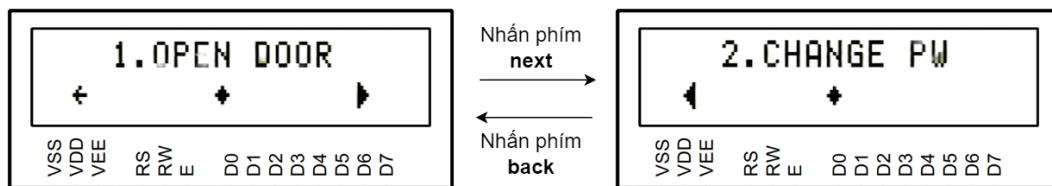
3.4 USER DASHBOARD

Người dùng (user) sẽ có 2 quyền hạn khi đã được xác thực đúng ID và mật khẩu đó là: **Vào cửa - OPEN DOOR** và **Thay đổi mật khẩu của mình - CHANGE PASSWORD**.



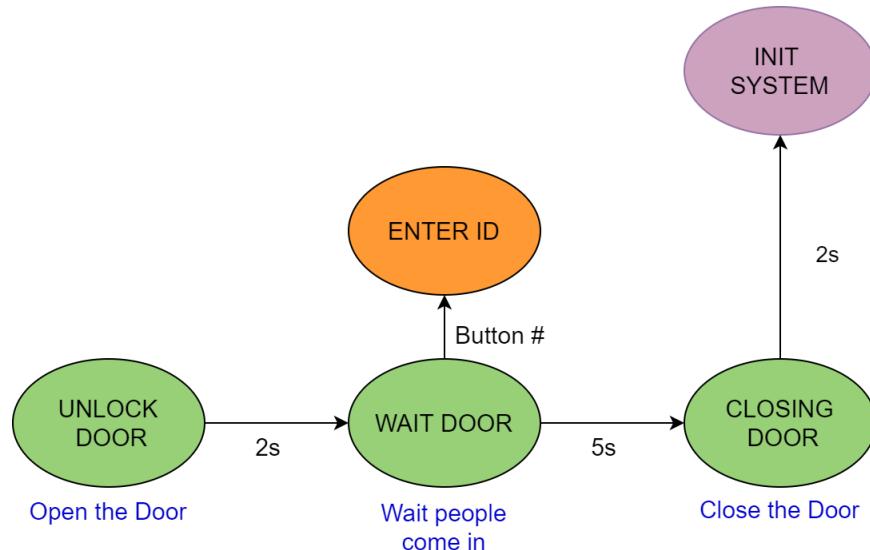
Hình 14: *FSM của USER DASHBOARD*

Để có được một giao diện dễ sử dụng cho người dùng thì mỗi chức năng sẽ được hiển thị trên 1 trang và người dùng có thể chuyển trang thông qua 2 nút là **next** (phím #) và **back** (phím *)



Hình 15: *Giao diện của USER DASHBOARD*

- Trang thứ nhất hiển thị option 1: **OPEN DOOR**. Nếu người dùng muốn thực hiện chức năng này thì nhấn nút **select** - phím số 0, hệ thống sẽ chuyển đến trạng thái **UNLOCK** (hay **UNLOCK DOOR**) - Mở cửa, cửa sẽ mở trong 2s bằng cách điều khiển motor thông qua L293D đồng thời báo hiệu với còi buzzer. Sau 2s, hệ thống chuyển qua trạng thái **WAIT DOOR**, lúc này cửa đã mở và chờ trong 5s. Trong 5s này, nếu người dùng khác muốn tiếp tục vào cửa, họ có thể nhấn nút # để đi tới trạng thái **ENTER ID** nhằm enter 1 ID mới. Nhưng nếu không có ai nhấn nút # trong 5 giây này, hệ thống sẽ chuyển sang trạng thái **CLOSING DOOR**, điều khiển motor quay ngược chiều nhằm đóng cửa trong 2s kèm còi báo hiệu. Sau khi đóng cửa xong, hệ thống trở lại trạng thái **INIT SYSTEM**. Còn nếu người dùng không muốn thực hiện chức năng này thì có thể chuyển sang trang khác bằng cách nhấn phím **next** (phím #).
- Trang thứ hai hiển thị option 2 đó là **CHANGE PASSWORD**. Nếu người dùng muốn thay đổi mật khẩu của mình thì người dùng nhấn nút chọn chức năng này (phím số 0) để thực hiện thay đổi mật khẩu. Hệ thống sẽ chuyển sang trạng thái **CHANGE PASSWORD**, ở trạng thái này người dùng sẽ nhập một mật khẩu mới cho tài khoản của mình. Sau khi nhập xong và nhấn **submit** (phím #) thì hệ thống sẽ chuyển sang trạng thái **APPLY NEW PASSWORD**, ở trạng thái



Hình 16: *FSM của OPEN DOOR*

này, mật khẩu mới sẽ được lưu lại và hiển thị thông báo ra màn hình "**CHANGE SUCCESS**" trong 2 giây rồi quay về lại trạng thái **INIT SYSTEM**.

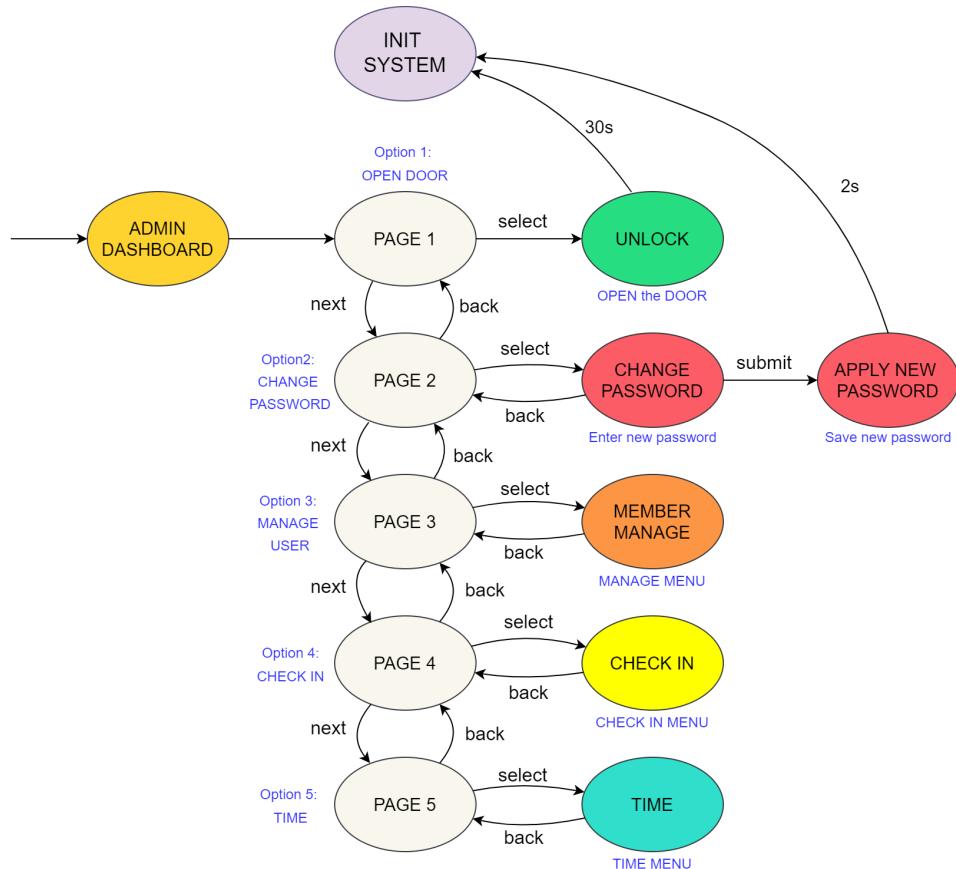
3.5 ADMIN DASHBOARD

Đối với tài khoản admin thì admin sẽ có 5 chức năng chính đó là:

- Mở cửa - **OPEN DOOR**
- Thay đổi mật khẩu của admin - **CHANGE PASSWORD**
- Quản lý thành viên - **MEMBER MANAGE**
- Quản lý điểm danh - **CHECK IN**
- Cài đặt thời gian - **TIME**

Tương tự như giao diện của user thì giao diện của admin cũng được phân ra nhiều trang để dễ sử dụng: trang thứ nhất hiển thị option 1 - **OPEN DOOR**; trang thứ 2 hiển thị option 2 - **CHANGE PASSWORD**; trang thứ 3 hiển thị option 3 - **MEMBER MANAGE**; trang thứ 4 hiển thị option 4 - **CHECK IN** và trang thứ 5 hiển thị option 5 - **TIME**. Và các trang này có thể được điều hướng qua lại bằng 2 phím **next** và **back** (phím **#** và phím *****). Khi muốn thực hiện một chức năng nào thì nhấn phím **số 0** để chọn chức năng đó.

Hai chức năng: **OPEN DOOR** và **CHANGE PASSWORD** thì hoạt động tương tự như của user nên chúng ta sẽ tìm hiểu chi tiết hơn các chức năng còn lại:



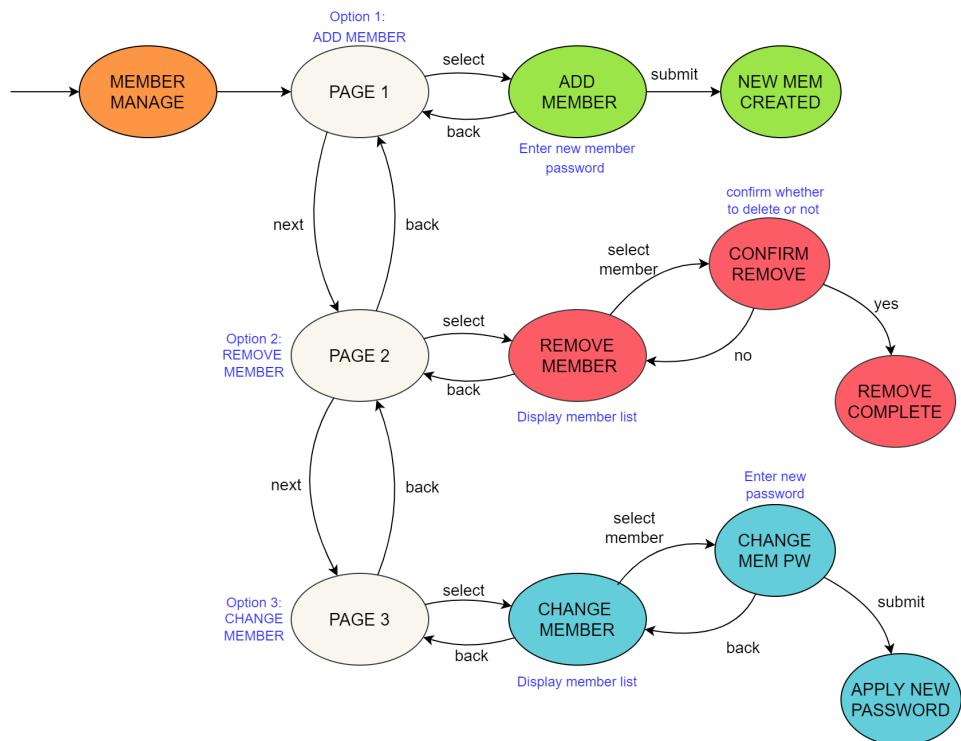
Hình 17: *FSM của ADMIN DASHBOARD*

3.5.1 MEMBER MANAGE

Với chức năng **MEMBER MANAGE**, admin sẽ có 3 quyền đó là: Thêm một thành viên mới - **ADD MEMBER**; Xóa một thành viên - **REMOVE MEMBER** và Thay đổi mật khẩu của các thành viên - **CHANGE MEMBER**.

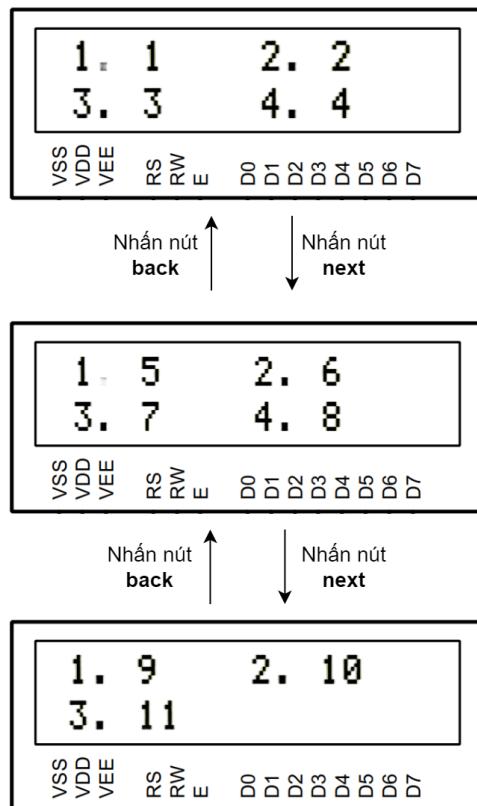
Để hiển thị các lựa chọn thì chúng ta cũng phân ra làm 3 trang và mỗi trang sẽ hiển thị một lựa chọn.

- Ở option 1 - **ADD MEMBER**: Trước tiên, chương trình sẽ kiểm tra số lượng thành viên trong hệ thống, nếu số lượng thành viên hiện tại là 200 người, tức là số lượng thành viên đã đạt tối đa, hệ thống sẽ ngăn cản việc tạo thêm tài khoản mới, in ra thông báo “**MAX ACCOUT REACHED**” và đưa admin quay lại **ADMIN DASHBOARD**. Nhưng nếu số lượng tài khoản bé hơn 200, hệ thống sẽ tiến hành cấp một tài khoản với ID tăng dần, và admin sẽ tạo một mật khẩu cho tài khoản đó. Sau khi nhập mật khẩu đủ 6 ký số, admin nhấn nút# để xác nhận tạo tài khoản với ID và mật khẩu mới, trạng thái chuyển qua **NEW MEMBER CREATED** và in ra trên màn hình LCD thông tin ID và mật khẩu của tài khoản vừa tạo.



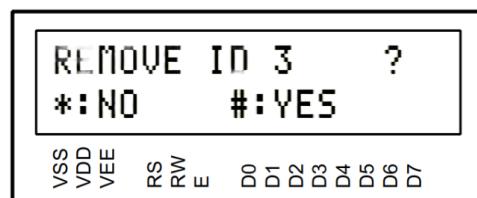
Hình 18: *FSM của MEMBER MANAGE*

- Option 2 - **REMOVE MEMBER**: Tiếp theo là chức năng remove Member. Khi chọn chức năng này, chương trình sẽ tiến hành quét mảng chứa thông tin người dùng và liệt kê chúng trên màn hình LCD, có tối đa 4 người dùng có thể hiện cùng lúc trên màn, và chúng ta có rất nhiều tài khoản, ta sẽ tiến hành phân trang, và ta có thể di chuyển qua lại giữa các trang liệt kê bằng phím# để tới trang kế tiếp hay phím * để quay lại trang trước đó. Một trang hiển thị 4 người dùng sẽ được gắn liền với số thứ tự từ 1 tới 4, và ta sẽ chọn tài khoản cần xóa bằng cách nhấn nút số 1,2,3 hay 4 tương ứng với ID của tài khoản đó.



Hình 19: Giao diện liệt kê danh sách thành viên trong REMOVE MEMBER VÀ CHANGE MEMBER

Khi chọn tài khoản cần xóa bằng các phím số từ 1 tới 4, hệ thống sẽ chuyển tới trạng thái **CONFIRM REMOVE**. Màn hình sẽ yêu cầu admin xác nhận một lần nữa rằng họ có muốn xóa tài khoản này không. Nếu không, admin sẽ nhấn phím * để quay trở lại trang liệt kê (trạng thái **REMOVE MEMBER**). Nếu có, admin nhấn phím #, hệ thống sẽ di chuyển tới trạng thái **REMOVE COMPLETE**, tiến hành xóa đi tài khoản đó khỏi mảng tài khoản bằng cách đem những tài khoản ở phía sau tài khoản đó đẩy lên trên, đè lên tài khoản cần xóa, đồng thời in ra màn hình thông tin xác nhận “**REMOVE COMPLETE**”.



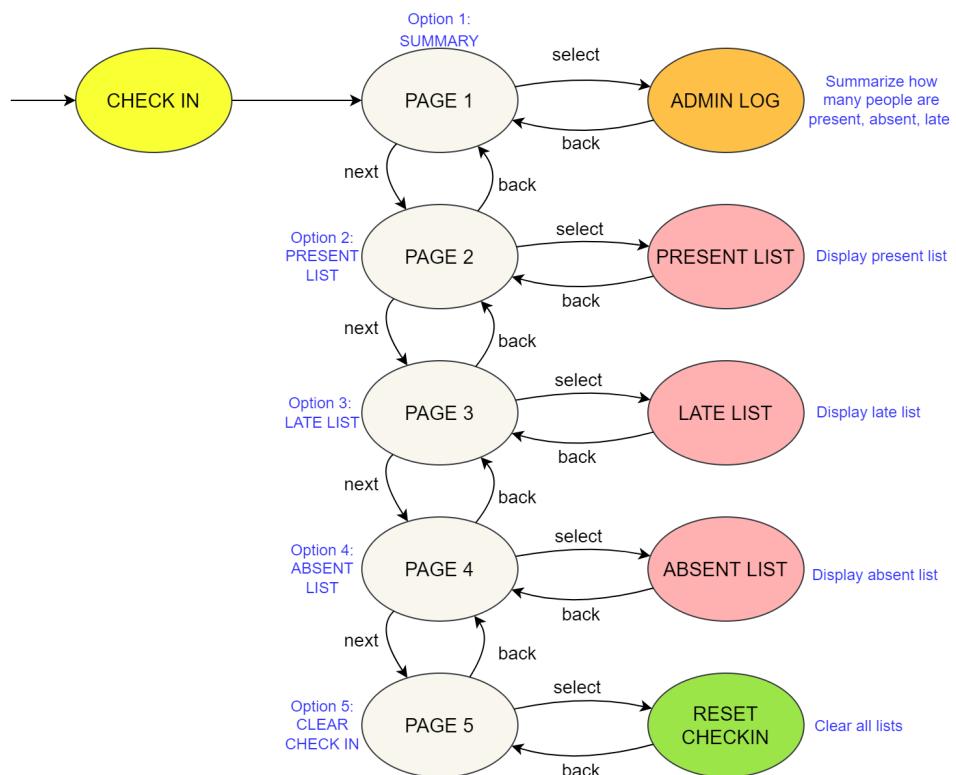
Hình 20: Giao diện CONFIRM REMOVE

- Option 3 - **CHANGE MEMBER**: tương tự, hệ thống sẽ in ra danh sách các thành viên để admin lựa chọn. Sau khi chọn được thành viên muốn thay đổi mật khẩu thì hệ thống chuyển sang trạng thái **CHANGE MEM PW**, ở trạng thái này, admin sẽ được yêu cầu nhập mật khẩu mới cho tài khoản user đã chọn. Sau khi nhấn **submit** thì mật khẩu mới sẽ được lưu lại và in ra xác nhận thông tin đã thay đổi - trạng thái **APPLY NEW PASSWORD**.

3.5.2 CHECK IN

Với chức năng Quản lý điểm danh - **CHECK IN**, admin sẽ có thể làm các công việc sau:

- Tổng kết số lượng user hiện diện, vắng mặt hay đi trễ - **SUMMARY**
- Xem danh sách các user hiện diện - **PRESENT LIST**
- Xem danh sách các user đi trễ - **LATE LIST**
- Xem danh sách các user vắng mặt - **ABSENT LIST**
- Reset lại hệ thống quản lý điểm danh - **CLEAR CHECK IN**



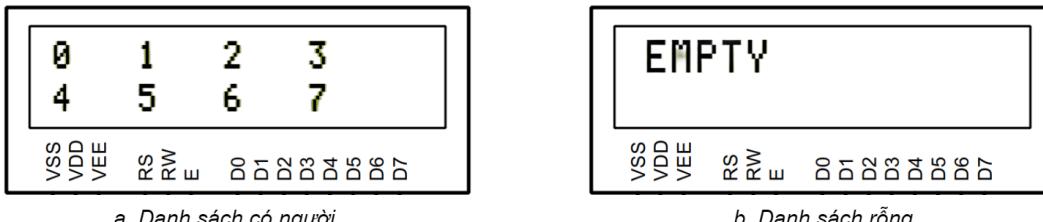
Hình 21: *FSM của CHECK IN*

- Với option 1 - **SUMMARY**: khi chọn chức năng này, hệ thống sẽ chuyển sang trạng thái **ADMIN LOG** - liệt kê ra số lượng user hiện diện, vắng mặt, đi trễ và in chúng ra màn hình.



Hình 22: Giao diện chức năng SUMMARY

- Option 2, 3 và 4 lần lượt in ra danh sách các user hiện diện - **PRESENT LIST**, đi trễ - **LATE LIST** và vắng mặt - **ABSENT LIST**. Tính năng **PRESENT LIST**, liệt kê những tài khoản hiện diện/tới sớm. Khi này, 1 danh sách gồm tối đa 8 tài khoản sẽ được hiển thị trên LCD, kết hợp với khả năng phân trang, di chuyển giữa các trang bằng nút # để đi tới trang tiếp theo hay * để quay lại trang trước đó. Còn nếu danh sách rỗng thì màn hình sẽ hiển thị "EMPTY". Để rà soát và liệt kê những tài khoản hiện diện, nhóm sử dụng 1 hàm để quét toàn bộ mảng tài khoản và tìm kiếm những tài khoản có thuộc tính checkin tương ứng.



a. Danh sách có người

b. Danh sách rỗng

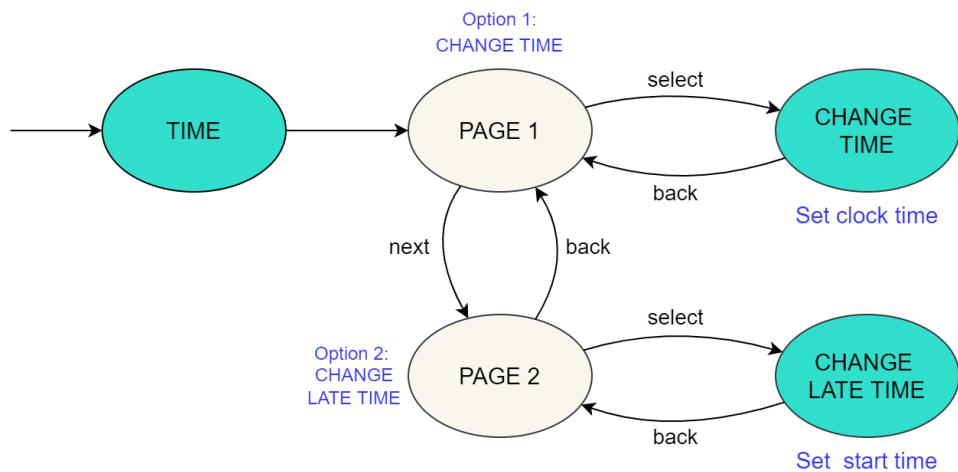
Hình 23: Giao diện liệt kê danh sách thành viên trong PRESENT LIST, ABSENT LIST và LATE LIST

Tương tự như **PRESENT LIST**, **LATE LIST** và **ABSENT LIST** rà soát và liệt kê trên màn hình LCD những tài khoản đi trễ và vắng.

- Option 5 - **CLEAR CHECK IN**: khi admin chọn chức năng này hệ thống sẽ chuyển sang trạng thái **RESET CHECKIN** - xóa tất cả các lịch sử ra vào của user (reset lại các danh sách hiện diện, trễ và vắng).

3.5.3 TIME

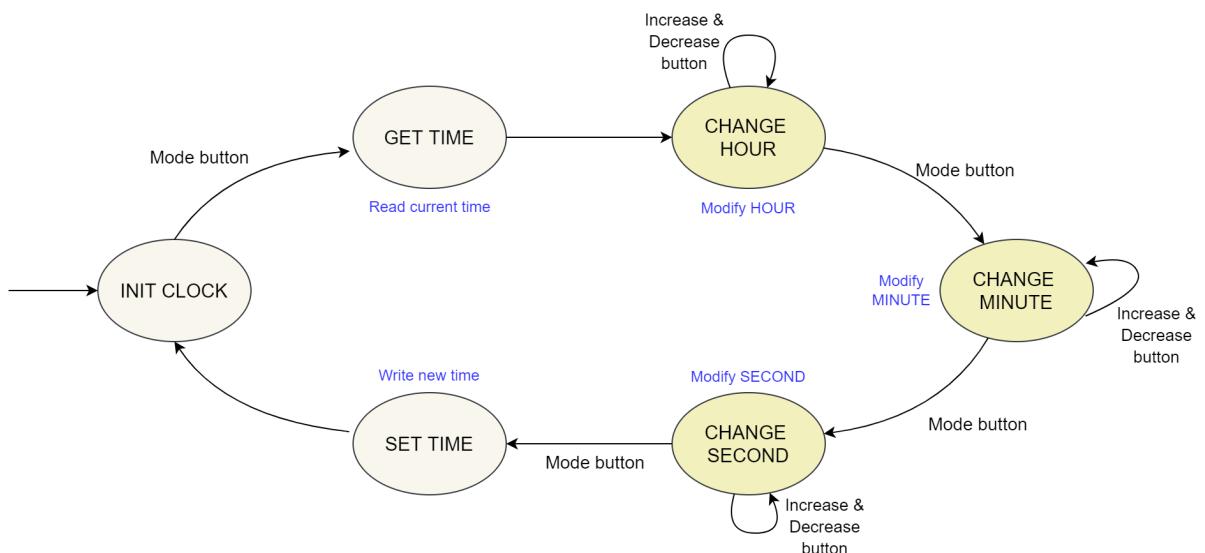
Với chức năng Cài đặt thời gian - **TIME**, admin có 2 quyền hạn đó là: Điều chỉnh thời gian thực - **CHANGE TIME** và Điều chỉnh thời điểm bắt đầu (sau thời điểm này, mọi user đăng nhập vào hệ thống sẽ bị xem là trễ) - **CHANGE LATE TIME**.



Hình 24: *FSM của TIME*

Việc phân trang và điều hướng cũng tương tự như các phần trên. Tiếp theo, chúng ta sẽ tìm hiểu chi tiết hơn về 2 chế độ **CHANGE TIME** và **CHANGE LATE TIME**:

3.5.3.1 CHANGE TIME



Hình 25: *FSM của CHANGE TIME*

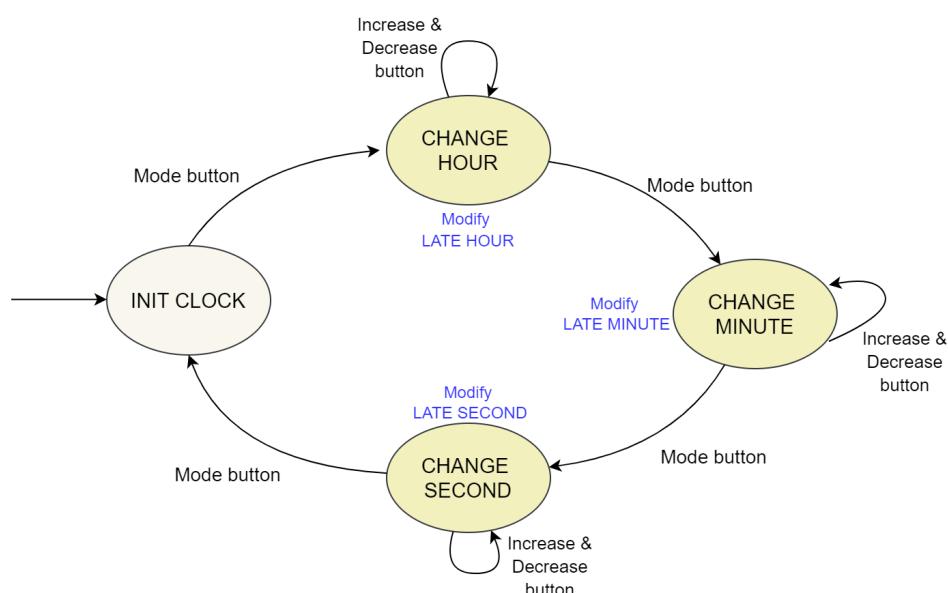
Đây là máy trạng thái của **CHANGE TIME**, mục đích để thay đổi thời gian của đồng hồ qua việc sử dụng DS1307. Khi mới bắt đầu, ta sẽ ở trạng thái **INIT CLOCK**. Để tiến hành chỉnh thời gian, ta nhấn phím Mode (phím 3), hệ thống sẽ đọc các giá trị giờ, phút, giây từ DS1307 (trạng thái **GET TIME**) đưa vào các biến tạm - các biến tạm này sẽ được chỉnh sửa trong các mode chỉnh giờ/phút/giây rồi tiến hành lưu lại vào DS1307 khi xác nhận chỉnh sửa. Sau khi **GET TIME**, hệ thống sẽ tự động vào chế độ chỉnh giờ (**CHANGE HOUR**). Khi ở chế độ chỉnh giờ, ta có thể dùng phím 1 để tăng giá trị, phím 2 để giảm giá trị, hoặc nhấn giữ phím để tăng nhanh/giảm nhanh. Khi vào chế độ chỉnh giờ, giá trị giờ sẽ nhấp nháy trên màn hình LCD để báo hiệu cho ta biết.

Tương tự như vậy, nhấn phím Mode để chuyển từ chế độ chỉnh giờ sang chỉnh phút (**CHANGE MINUTE**), và từ chỉnh phút sang chỉnh giây (**CHANGE SECOND**).

Nếu từ chế độ chỉnh giây, ta nhấn nút Mode thêm lần nữa, thì thời gian ta đã chỉnh (các biến tạm) sẽ được ghi vào DS1307 ở trạng thái **SET TIME**. Điều đó có nghĩa là chỉ sau khi nhấn nút Mode sau trạng thái chỉnh giây, thì thời gian của đồng hồ mới bị thay đổi. Sau khi lưu thời gian vào DS1307, DS1307 sẽ chạy tiếp dựa trên giá trị mới, đồng thời máy trạng thái quay về **INIT CLOCK**.

Nếu ta nhấn nút trở về (nút *), ta sẽ trở về màn hình chọn chức năng của **TIME**. Vì thời gian chỉ bị thay đổi sau lần nhấn nút mode sau trạng thái chỉnh giây, nếu ta không muốn thay đổi thời gian nữa, có thể nhấn nút * khi đang ở trạng thái chỉnh giờ/chính phút/chỉnh giây. Khi đó, các giá trị sẽ không được lưu vào DS1307, do đó không làm thay đổi thời gian gốc trước khi chỉnh.

3.5.3.2 CHANGE LATE TIME



Hình 26: *FSM của CHANGE LATE TIME*

Đây là máy FSM dùng để chỉnh mốc thời gian trễ. Mốc thời gian trễ này được dùng để phân biệt giữa người đi sớm và người đi trễ: nếu người vào cửa đi lúc thời gian trễ hơn mốc này, sẽ được xem là đi trễ. Máy trạng thái này cũng có khả năng chỉnh giờ, phút, giây của mốc thời gian trễ, tuy nhiên chỉ có 4 trạng thái vì khi chỉnh mốc thời gian này, ta thay đổi trực tiếp các biến liên quan tới mốc thời gian trễ - thay vì xử lý biến tạm, như ở máy trạng thái trước. Do đó, mọi sự thay đổi về giờ trong trạng thái chỉnh giờ, về phút trong trạng thái chỉnh phút và tương tự với giây sẽ được lưu lại mà không nhất thiết phải bấm nút mode sau trạng thái chỉnh giây.

Tương tự như trạng thái chỉnh giờ, nếu ta nhấn nút trở về (nút *), ta sẽ trở về màn hình chọn chức năng của **TIME**

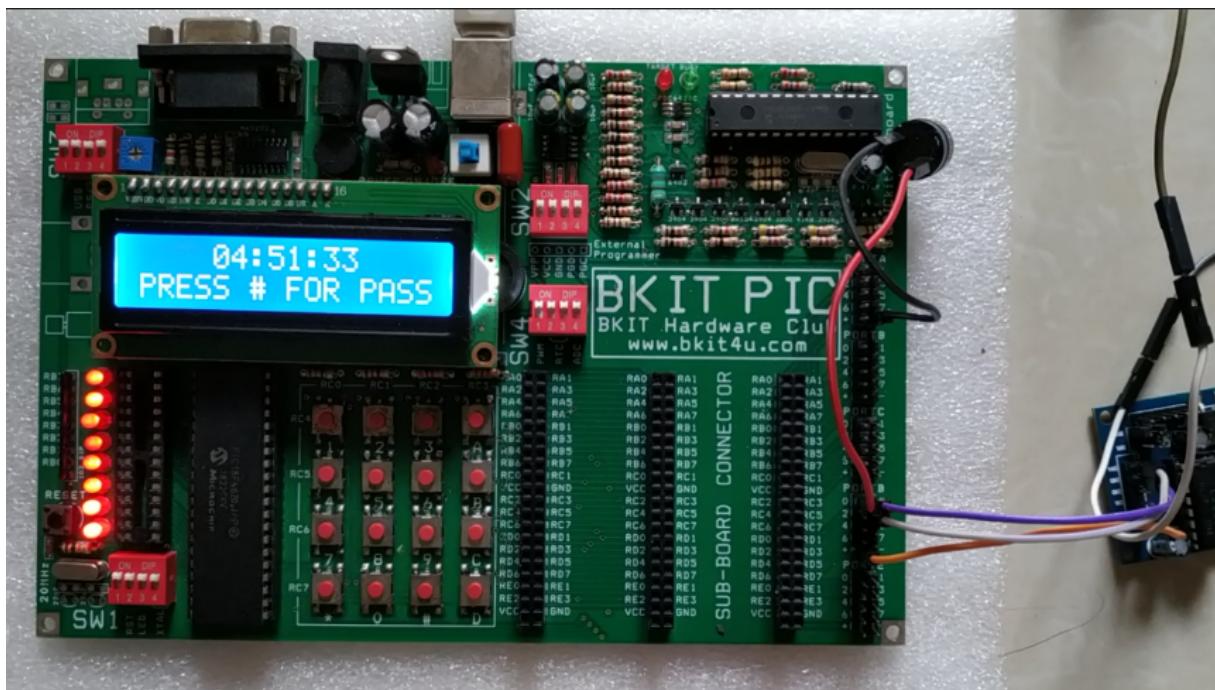
4 Kết quả

4.1 Link video demo

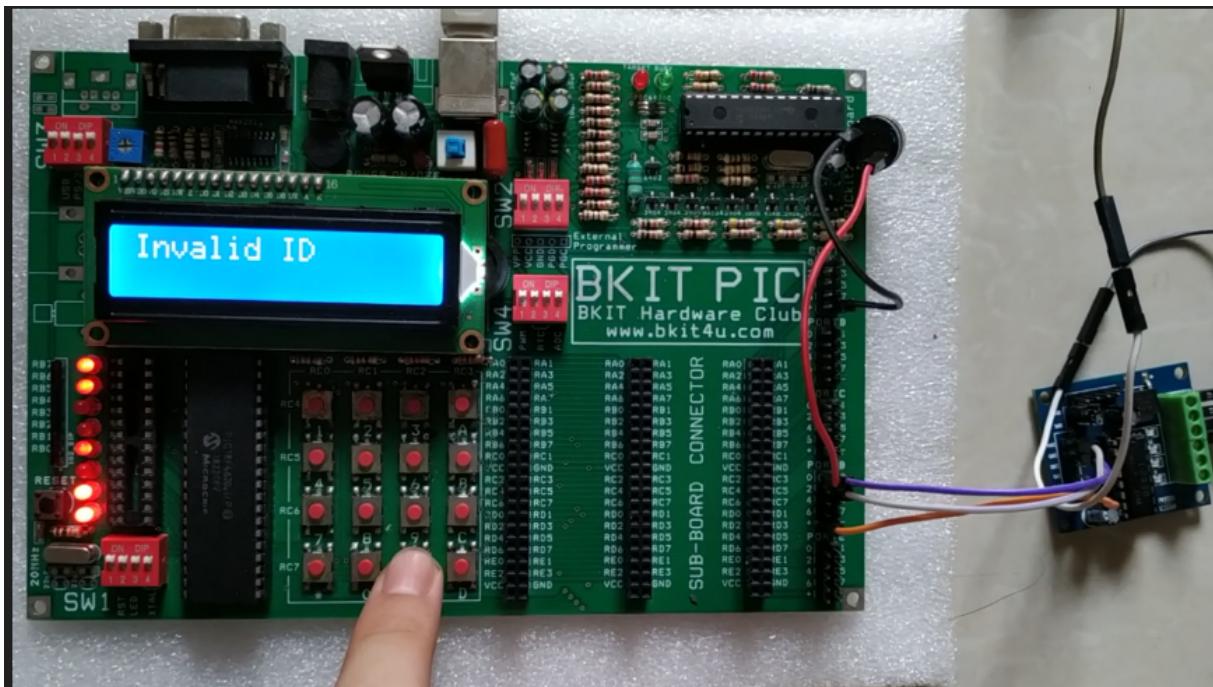
VIDEO DEMO KẾT QUẢ - ĐỒ ÁN THIẾT KẾ LUẬN LÝ

4.2 Một số hình ảnh thực tế

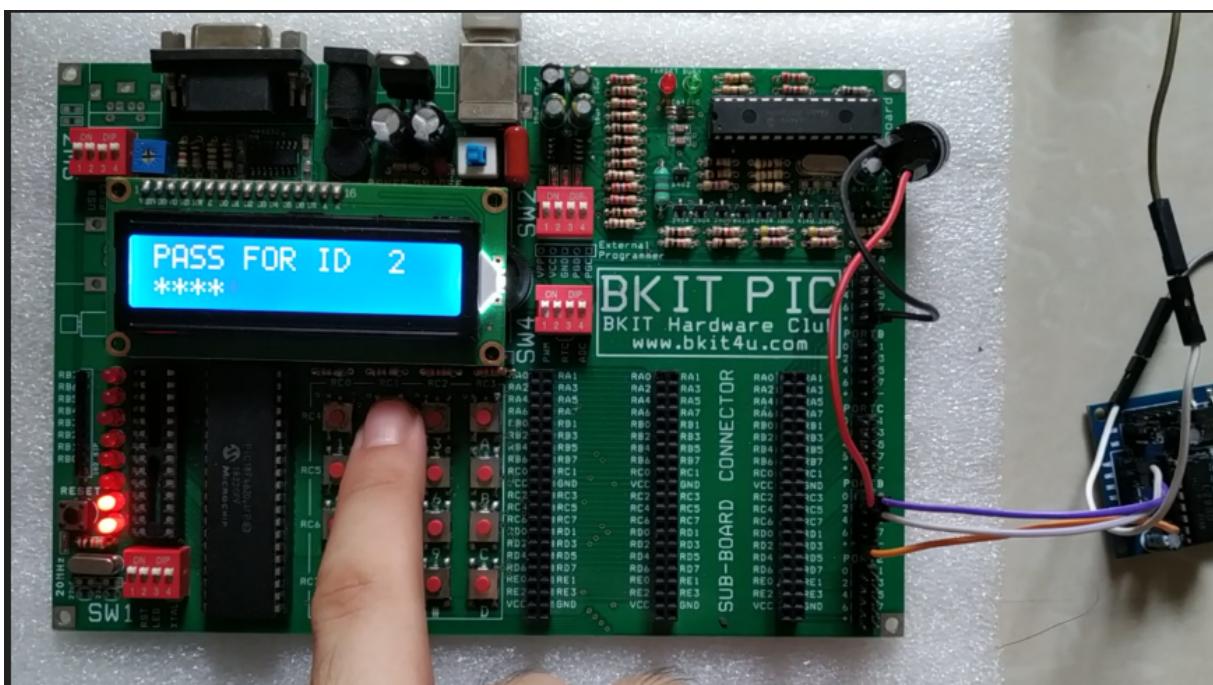
Dưới đây là một số hình ảnh thực tế chạy trên bo mạch



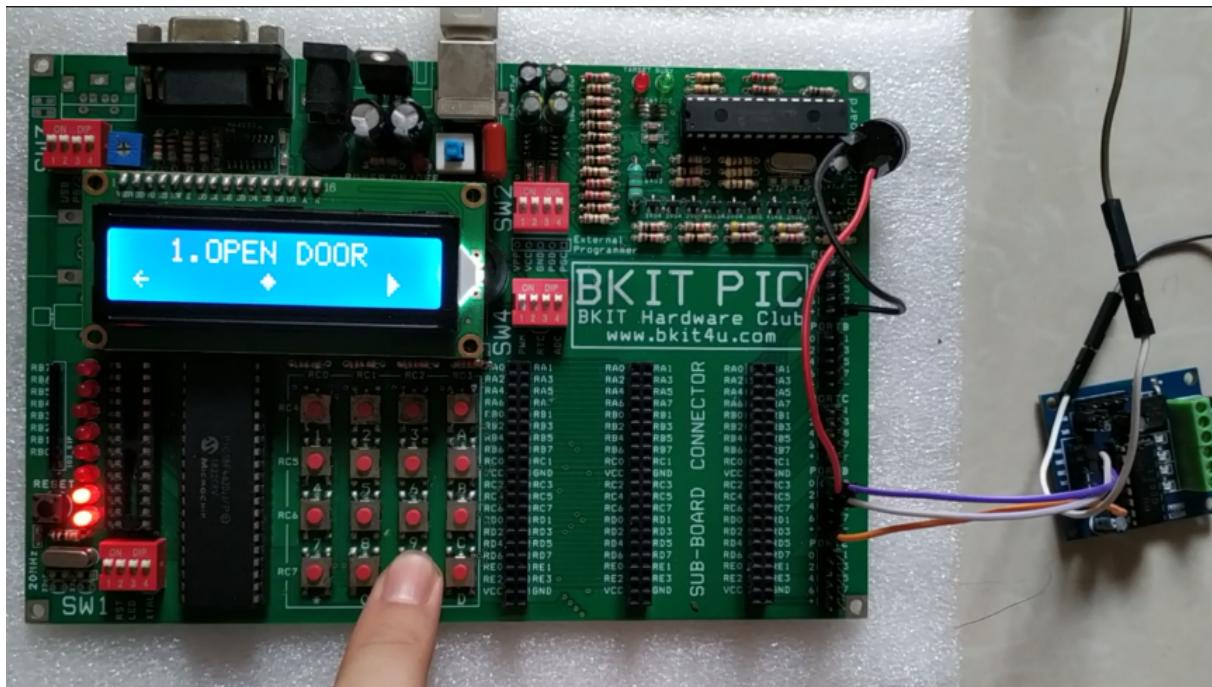
Hình 27: Màn hình khởi động của hệ thống



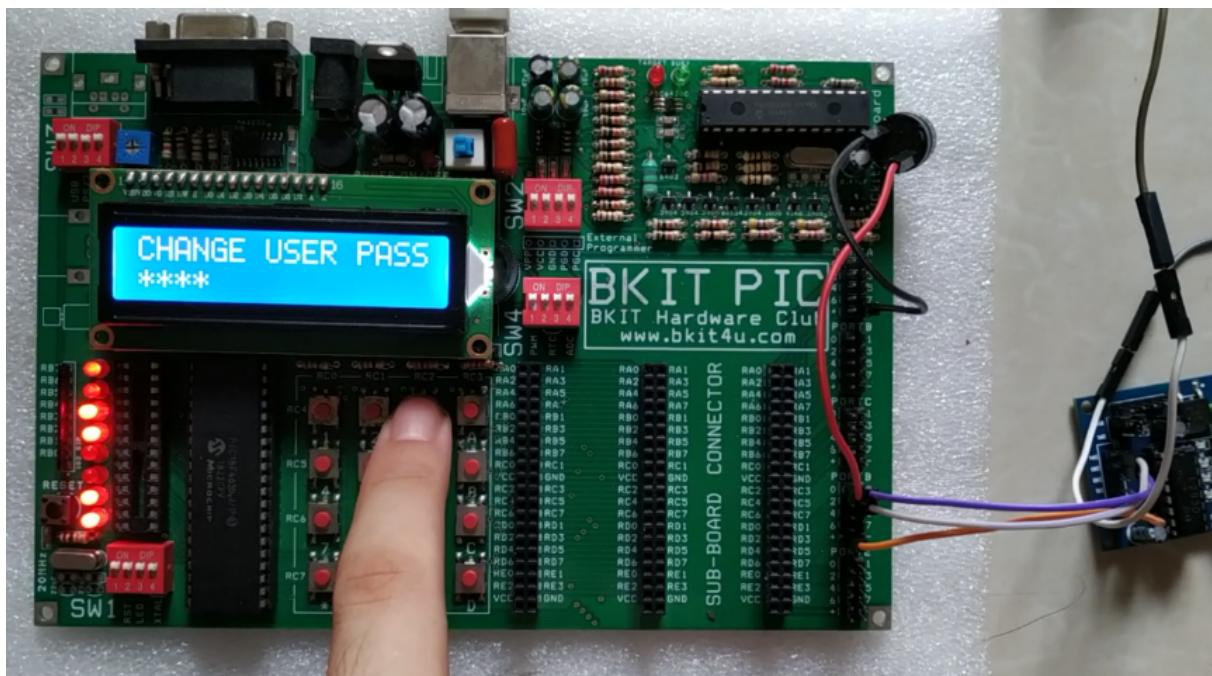
Hình 28: Màn hình thông báo khi ID nhập vào không hợp lệ/không tồn tại



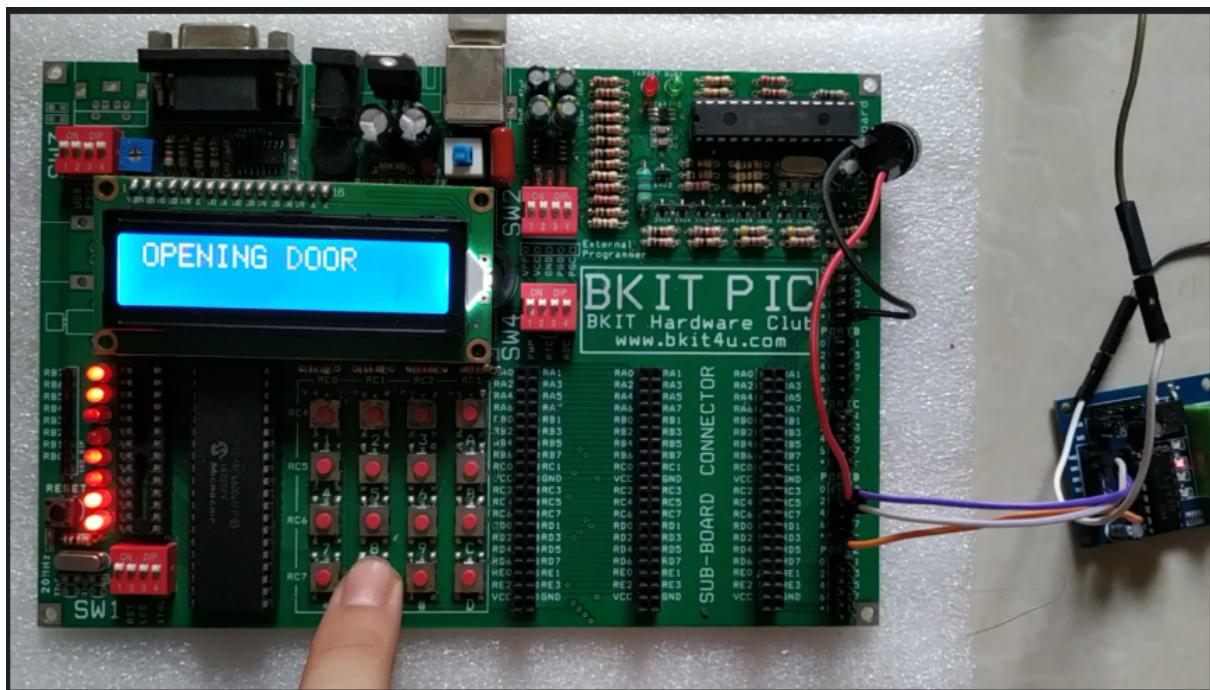
Hình 29: Màn hình nhập mật khẩu



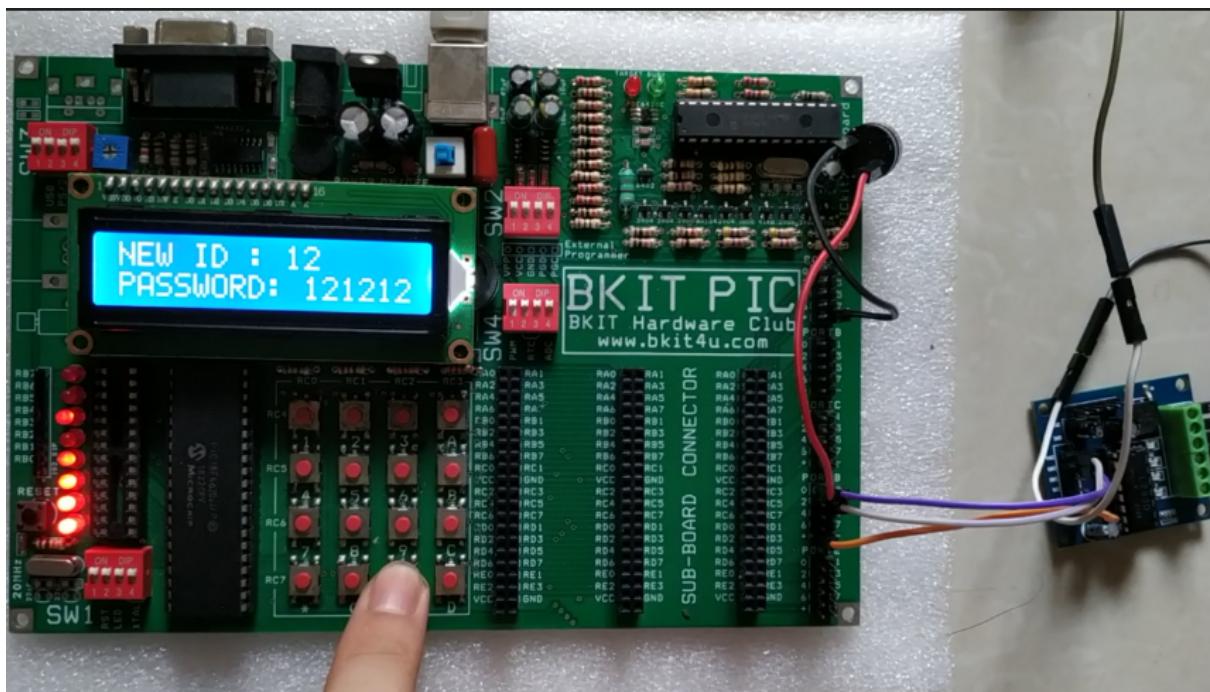
Hình 30: Giao diện phân trang cho các tính năng



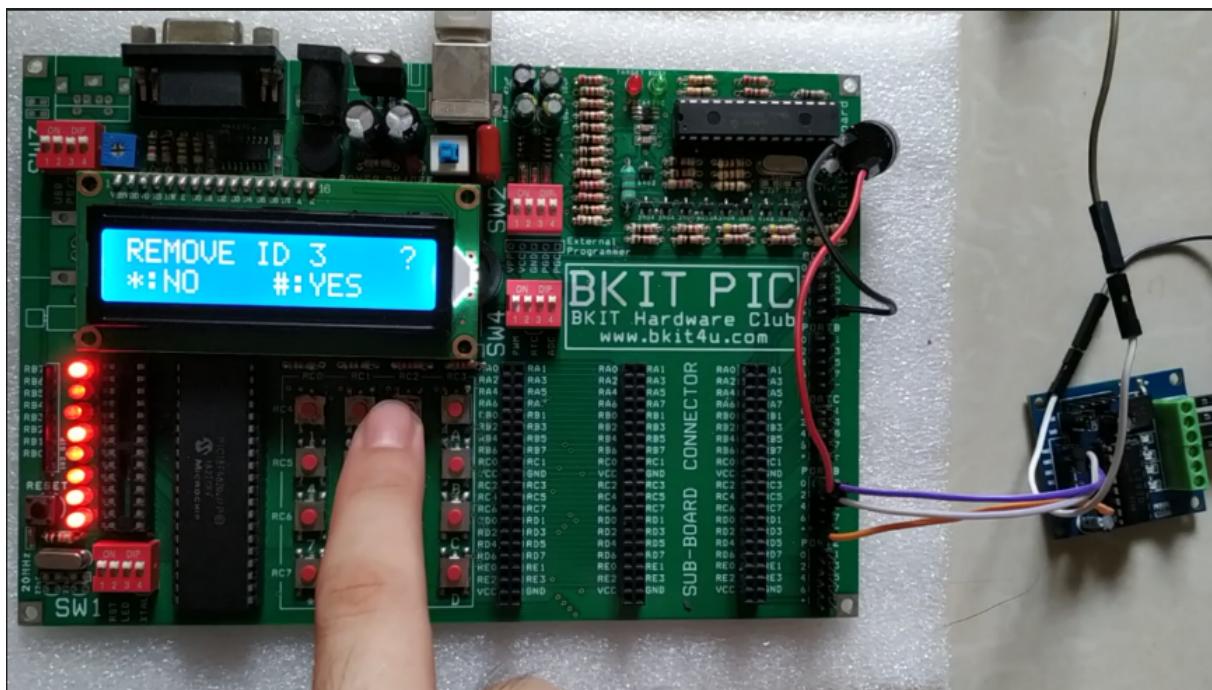
Hình 31: Màn hình người dùng đổi mật khẩu



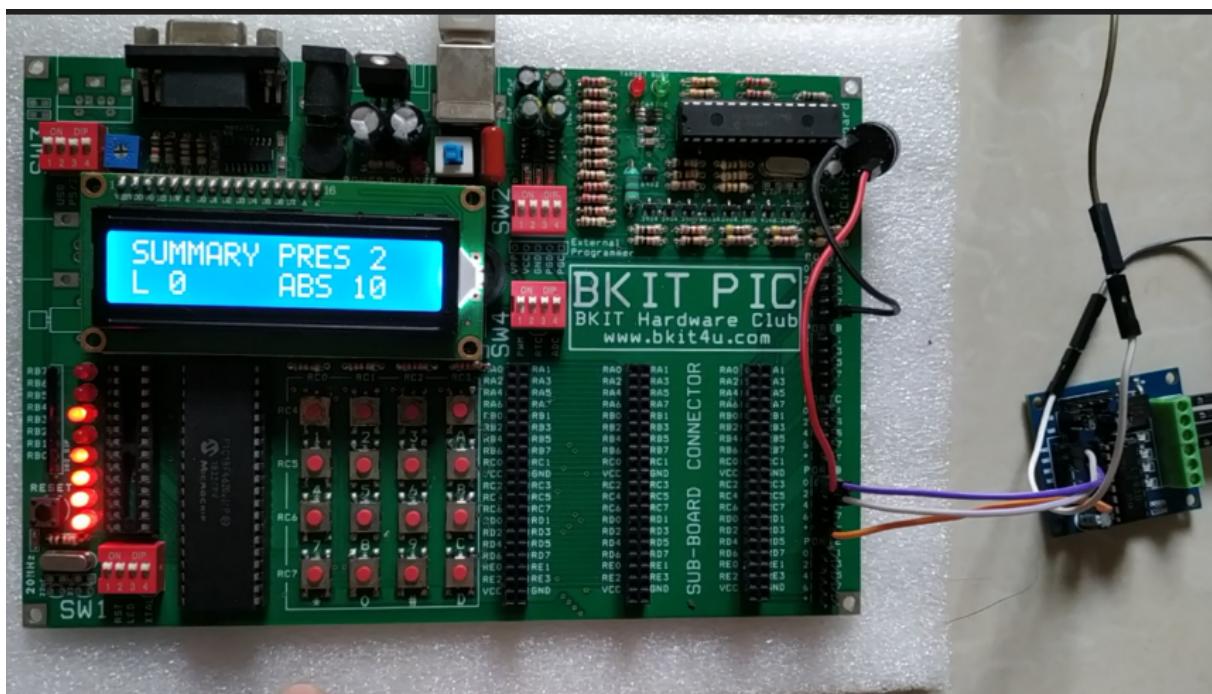
Hình 32: Thông báo mở cửa



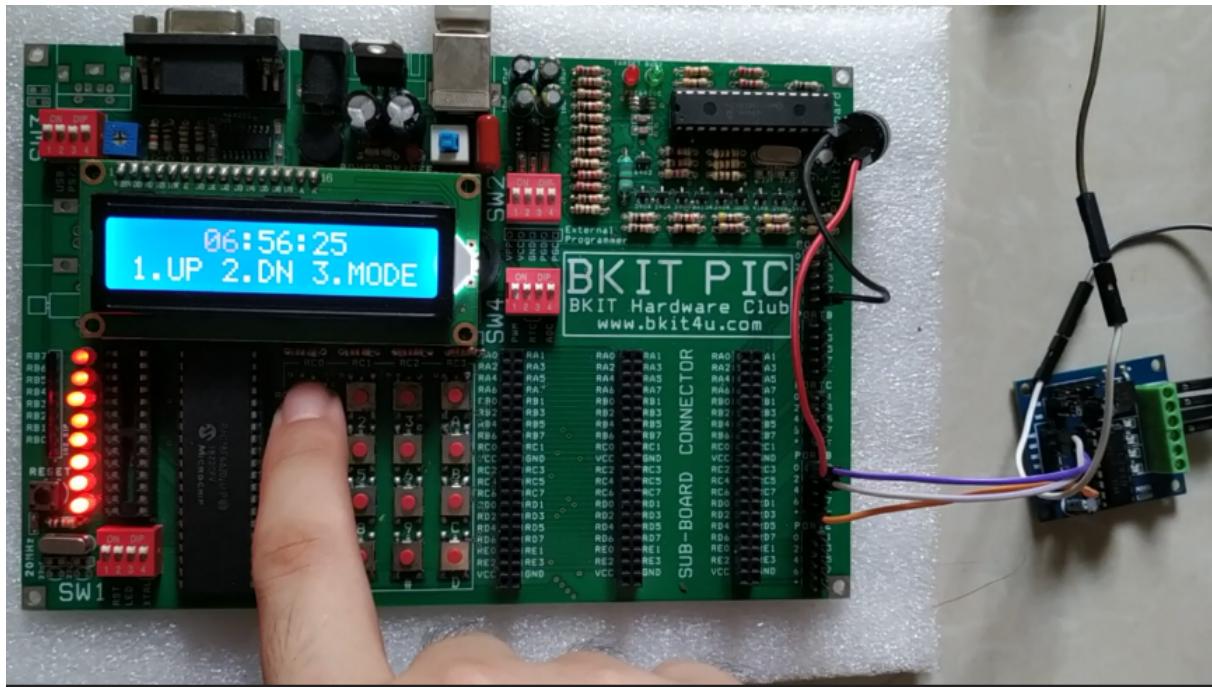
Hình 33: Màn hình khi admin tạo tài khoản mới



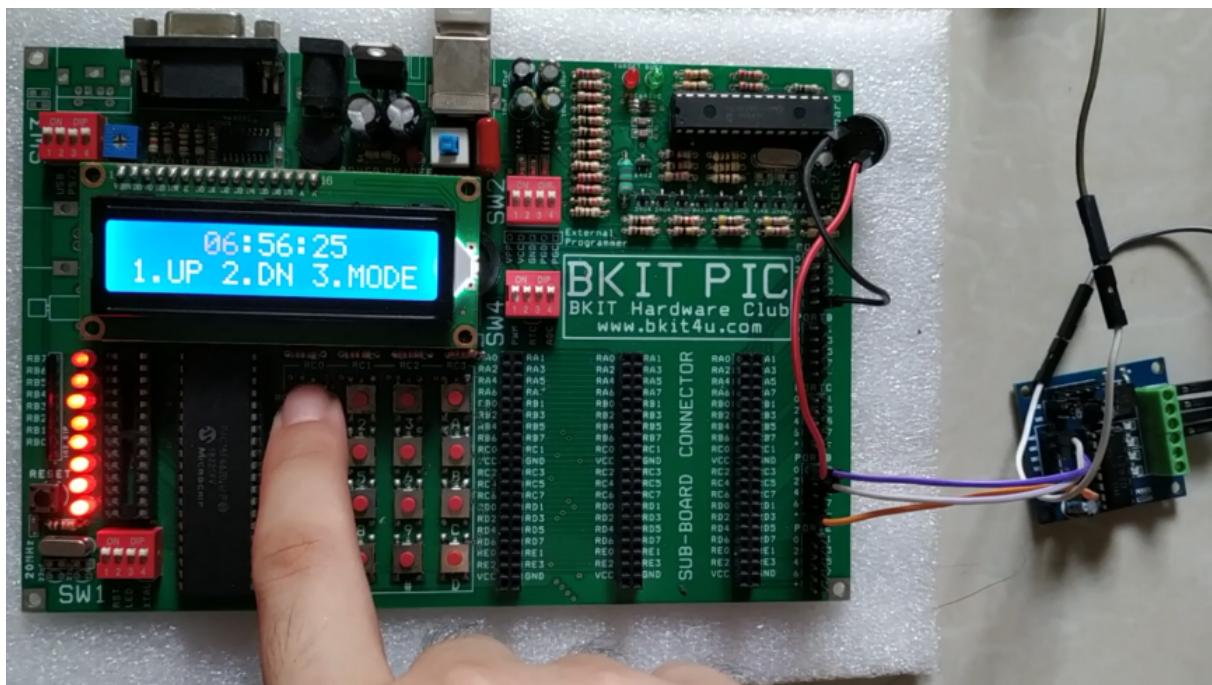
Hình 34: Màn hình xác nhận xóa tài khoản



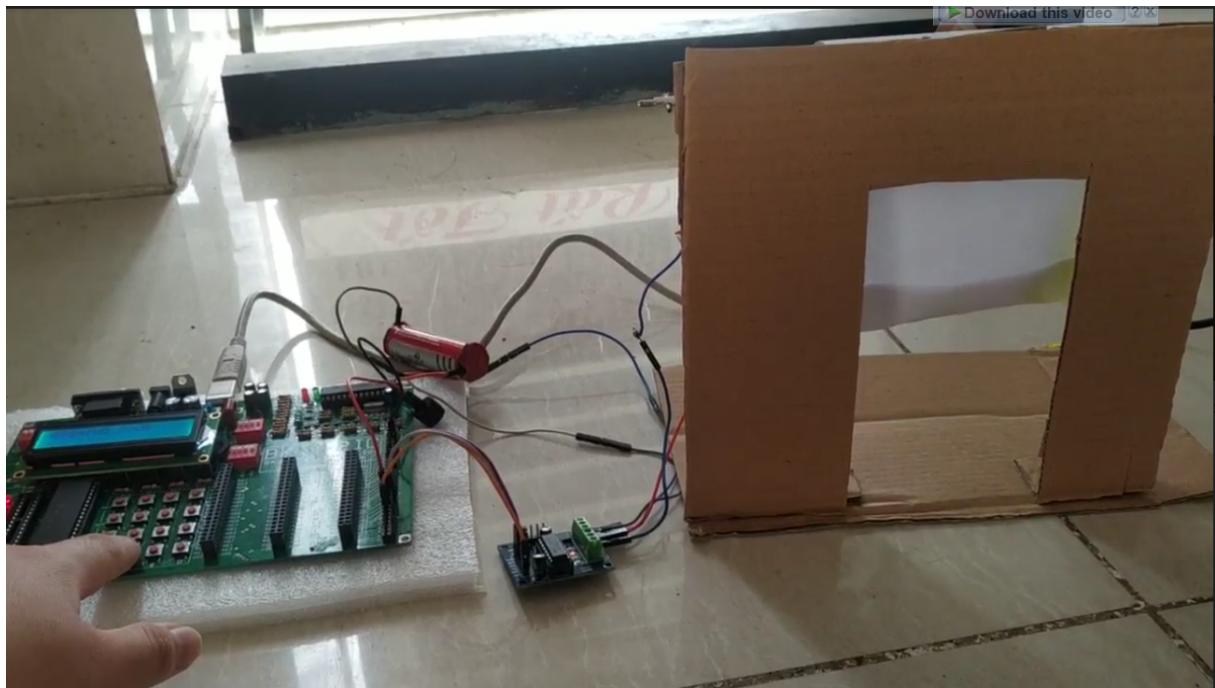
Hình 35: Màn hình tổng hợp summary



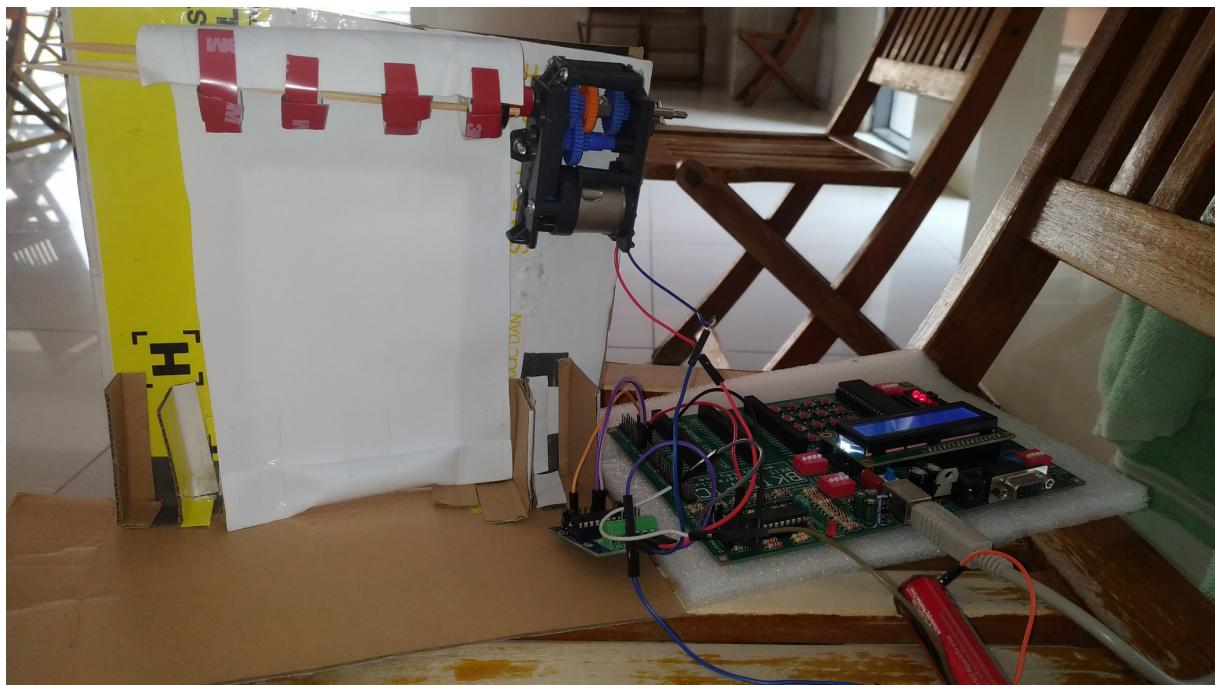
Hình 36: Màn hình khởi động của hệ thống



Hình 37: Màn hình chỉnh sửa thời gian



Hình 38: Hình ảnh của cửa cuốn mini minh họa cho hoạt động mở cửa



Hình 39: Ảnh chụp kết động cơ điều khiển cửa cuốn giấy và kết nối với L293D cùng bo mạch



5 Quá trình phát triển

5.1 Tuần 1: 13/11 -> 20/11

Hoàn thành

- Viết tính năng đăng nhập bằng mật khẩu
- Đổi thành mật khẩu 6 số
- Thêm ID cho người dùng, đăng nhập bằng ID và mật khẩu
- Tạo giao diện cho người dùng và admin
- Tính năng phân trang đối với nội dung quá dài

Dự kiến

- Thêm tính năng đổi mật khẩu
- Thêm các tính năng quản lý thành viên cho admin
- Áp dụng UART để ghi và log thông tin đăng nhập
- Áp dụng giao tiếp I2C với DS1307 để ghi lại thời gian đăng nhập của từng người và gửi qua UART

5.2 Tuần 2: 20/11 -> 27/11

Hoàn thành

- Thêm tính năng đổi mật khẩu (thành viên và admin)
- Tạo chức năng thêm thành viên (admin)
- Tạo chức năng xóa thành viên (admin)
- Tạo chức năng đổi mật khẩu thành viên (admin)
- Khó khăn khi áp dụng UART: xung đột với nút nhấn, quyết định hủy tính năng UART
- Cho phép xóa dữ liệu nếu nhập sai
- Khả năng liệt kê danh sách thành viên hiện có trong chức năng xóa và đổi mật khẩu thành viên (admin)

Dự kiến

- Tạo chức năng log lại số lượng người hiện diện và số lượng người vắng
- Hàn bo mạch BKIT PIC



5.3 Tuần 3: 27/11 -> 4/12

Hoàn thành

- Tạo chức năng log lại số lượng người hiện diện và số lượng người vắng
- Hàn bo mạch BKIT PIC hoàn thành, chưa kiểm thử

Dự kiến

- Kiểm thử bo mạch BKIT PIC, chạy thử code và sửa lỗi
- Thêm phát âm thanh loa khi mở cửa thông qua còi buzzer

5.4 Tuần 4: 4/11 -> 11/12

Hoàn thành

- Chạy thành công code trên bo mạch BKIT PIC
- Thêm âm thanh loa khi mở cửa

Dự kiến

- Áp dụng I2C, giao tiếp với DS1307 để kiểm soát thời gian, nhằm xác định thêm người đi trễ trong tính năng log
- Khả năng chỉnh sửa thời gian trong DS1307
- Khả năng chỉnh sửa, đặt thời gian mốc để xét trường hợp đi trễ (người vào cửa trễ hơn thời gian mốc sẽ được coi là đi trễ)
- Khả năng reset các trạng thái vắng, đi trễ, hiện diện
- Sắp xếp lại code thành các module
- Hoàn thiện báo cáo

5.5 Tuần 5: 11/11 -> 18/12

Hoàn thành

- Áp dụng I2C, giao tiếp với DS1307 để kiểm soát thời gian, nhằm xác định thêm người đi trễ trong tính năng log
- Khả năng chỉnh sửa thời gian trong DS1307
- Khả năng chỉnh sửa, đặt thời gian mốc để xét trường hợp đi trễ (người vào cửa trễ hơn thời gian mốc sẽ được coi là đi trễ)
- Khả năng reset các trạng thái vắng, đi trễ, hiện diện

Dự kiến

- Thay đổi lại giao diện cho hợp mắt hơn
- Tìm hiểu cách tạo ký tự custom trên LCD



5.6 Tuần 6: 18/12 -> 25/12

Hoàn thành

- Thay đổi lại giao diện
- Tạo và sử dụng thành công các kí tự custom trên LCD
- Thay đổi file linker để tạo mảng lớn (khoảng 2800 bytes) chứa thông tin người dùng, cho phép lên tới tối đa 200 người sử dụng.
- Điều chỉnh lại quá trình điều hướng giữa các trang chức năng.
- Làm một cửa cuốn mini sử dụng motor DC và L293D để mô phỏng chức năng mở cửa/đóng cửa

5.7 Xin cảm ơn các nhóm đồ án đã hỗ trợ nhóm 10:

- Nhóm bạn Kim Hưng đã giúp đỡ nhóm trong việc sửa lỗi các vấn đề về bo mạch, hỗ trợ tạo phần mềm mô phỏng và hướng dẫn sử dụng IDE để lập trình.
- Nhóm bạn Quang Hiệu hướng dẫn nhóm sử dụng còi buzzer.



6 Phân công

Trần Ngọc Cát	Diệp Trần Nam
<ul style="list-style-type: none">Viết code các chức năng dành cho adminThêm đồng hồ thời gian thực từ DS1307Thêm điều khiển motor bằng L293D và làm cửa cuốn miniViết báo cáo	<ul style="list-style-type: none">Viết code các chức năng của userChỉnh lại giao diện cho hệ thốngLàm video demo hệ thống và slideVẽ sơ đồ máy trạng tháiViết báo cáo



Tham khảo

- [1] Last Minute Engineers. *Last Minute Engineers.* URL: <https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/>.