I. Yêu cầu

Xây dựng một sơ đồ mạch trên Proteus và lập trình, mạch gồm có:

- Sử dụng VĐK STM32

- Giảo tiếp với matrix key 4x4

- Cảm biến nhiệt độ NTC

- Màn hình LCD 20x4

- Thẻ nhớ SD card

- Giao tiếp UART với máy tính

Hoạt động:

- Chương trình sử dụng FreeRTOS

- Chương trình gồm 2 chế độ hoạt động là Running và config

- Trong chế độ Running:

+ Động cơ quay thuận với tốc độ s1 trong thời gian t1, sau đó đảo chiều quay với tốc độ s2 trong thời gian t2 (tốc độ quy đổi theo độ rộng xung PWM cũng đc ko cần đọc encoder, thời gian tính theo giây)

+ Cảm biến nhiệt độ NTC được đọc với chu kỳ 1s 1 lần

+ Ghi vào thẻ nhớ SD card thông tin dưới dạng file, mỗi lần khởi động mạch sẽ tạo 1 file mới, mỗi lần ghi các thông tin gồm có: thời gian t1, thời gian t2, tốc độ s1 tốc độ s2, nhiệt độ NTC. (sử dụng thư viện FATFS hoặt một thư viện nào đó hỗ trợ đọc ghi định dạng file cũng đc)

+ Hiển thị lên màn hình LCD các thông tin t1,t2,s1,s2,NTC

+ Gửi lên UART các thông tin tin t1,t2,s1,s2,NTC định dạng giống như ghi vào file

- Trong chế độ Config:

+ Nhấn 1 tổ hợp 2 phím (tự chọn) trên matrix key để vào màn hình nhập password

+ Nhập đúng password thì vào màn hình config

+ Màn config dạng menu có 6 mục, vào 4 mục để cài đặt các thông số t1 t2 s1 s2, 1 mục vào để thay đổi password, một mục để save các thông tin đã bị thay đổi ở các mục còn lại vào thẻ nhớ flash (thẻ nhớ flash để 1 file lưu config các thông số kia)

+ Trong màn hình config dùng 1 nút để di chuyển menu, 1 nút để enter, một nút exit, các nút số để nhập password và thông số

II. Các linh kiện sử dụng

1. Nhiệt điện trở Thermistor và cảm biến NTC NCP15XH103

Khái niệm Thermistor:

Thermistor (viết tắt của *Thermal Resistor*) là một loại điện trở đặc biệt có khả năng thay đổi giá trị điện trở của nó theo nhiệt độ môi trường xung quanh. Điểm khác biệt nổi bật giữa thermistor và các loại điện trở thông thường là độ nhạy cao với sự thay đổi nhiệt độ – một sự thay đổi nhỏ về nhiệt có thể làm thay đổi điện trở đáng kể.

Nguyên lý hoạt động:

Thermistor được chế tạo từ các vật liệu bán dẫn đặc biệt, thường là oxit kim loại như mangan, niken, cobalt… Những vật liệu này có tính chất thay đổi độ dẫn điện khi nhiệt độ thay đổi. Nhờ vậy, Thermistor hoạt động như một cảm biến nhiệt độ đơn giản nhưng hiệu quả.

Thermistor thường được tạo thành dạng hạt nhỏ, viên đĩa hoặc hình trụ, và được bọc bằng lớp vật liệu bảo vệ như epoxy hoặc thủy tinh để chống ẩm và cách điện.

Thermistor được chia thành hai loại chính dựa trên cách điện trở thay đổi theo nhiệt độ:

- NTC (Negative Temperature Coefficient): điện trở giảm khi nhiệt độ tăng. Đây là loại phổ biến dùng để đo nhiệt độ.
- PTC (Positive Temperature Coefficient): điện trở tăng khi nhiệt độ tăng. Loại này thường dùng làm thiết bị bảo vệ mạch hoặc cảm biến quá nhiệt.

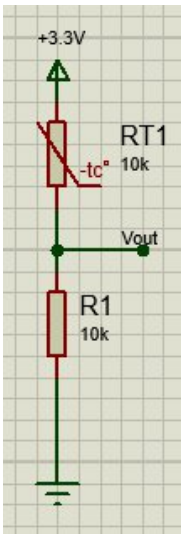
Cảm biến NTC NCP15XH103:

Cảm biến NCP15XH103 là một loại thermistor NTC do Murata sản xuất.

- Điện trở danh định tại 25°C (R25): 10kΩ (ký hiệu 103 = 10×10^3 Ω).
- Hệ số B: ~3380K (giữa 25°C và 50°C).
- Loại bọc: Epoxy resin.
- Dải nhiệt độ hoạt động: -40°C đến +125°C.
- Kích thước: rất nhỏ gọn (SMD 0402, phù hợp mạch in nhỏ).
- Đặc điểm nổi bật: độ chính xác cao, ổn định lâu dài, phù hợp cho ứng dụng đo nhiệt độ môi trường, pin, thiết bị di động, mạch bảo vệ nhiệt.

Giao tiếp NCP15XH103 với vi điều khiển:

Vì cảm biến xuất ra giá trị điện trở, nên có thể giao tiếp dễ dàng với vi điều khiển thông qua mạch chia áp. Sau đó, điện áp ngõ ra từ mạch chia áp sẽ được đưa vào chân ADC để đọc giá trị.



Công thức tính điện trở cảm biến:

$$R_{th} = \left(\frac{(2^N - 1) * R}{ADC\_Output}\right) - R$$

Với N là độ phân giải của bộ ADC, R là điện trở nối tiếp với Rth

Tính toán nhiệt độ từ giá trị Rth:

Từ phương trình:

$$R = R_0 \exp\left(B\left(\frac{1}{T} - \frac{1}{T_0}\right)\right)$$

R: Điện trở Rth tại nhiệt độ môi trường T (tính theo Kelvin-K)
T: Nhiệt độ tuyệt đối (K)
R0: Điện trở tại nhiệt độ môi trường T0(K)
B: Hằng số B của nhiệt điện trở (=3380)

2. Key Pad 4x4

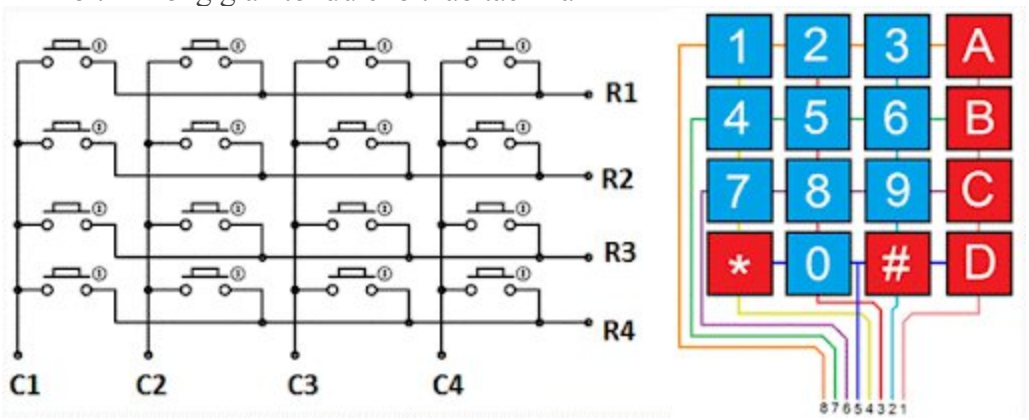
Keypad là thiết bị đầu vào chứa tập hợp các phím bấm được thiết kế để nhập các giá trị:

- Chữ số (0-9)
- Ký tự chữ cái cơ bản
- Các lệnh điều khiển đơn giản

Thiết kế:

- Không tích hợp đầy đủ bộ mã ASCII như bàn phím máy tính
- Thiết kế gọn nhẹ với số lượng phím hạn chế

- Cấu trúc ma trận 4 hàng × 4 cột
- Tổng cộng 16 phím bấm độc lập
- Bố trí không gian tối ưu cho thao tác nhanh



Cách sử dụng:

Để giao tiếp với KeyPad 4×4, người lập trình thường áp dụng phương pháp quét phím (key scanning). Đây là một kỹ thuật cho phép vi điều khiển (VĐK) kiểm tra liên tục trạng thái của từng phím bấm trên bàn phím.

Cách hoạt động của giải thuật này là: vi điều khiển lần lượt xuất tín hiệu trên một nhóm chân (ví dụ: các hàng) và đọc tín hiệu phản hồi từ nhóm chân còn lại (các cột). Khi một phím được nhấn, nó tạo thành một cầu nối giữa một hàng và một cột. Điều này giúp xác định chính xác vị trí phím đã bấm dựa vào tín hiệu được truyền và nhận.

Có hai cách quét phổ biến là quét theo hàng hoặc theo cột, tuỳ theo việc bạn chọn xuất tín hiệu trên hàng hay trên cột. Trong báo cáo này, phương pháp được sử dụng là xuất tín hiệu trên các hàng và đọc tín hiệu ở các cột.

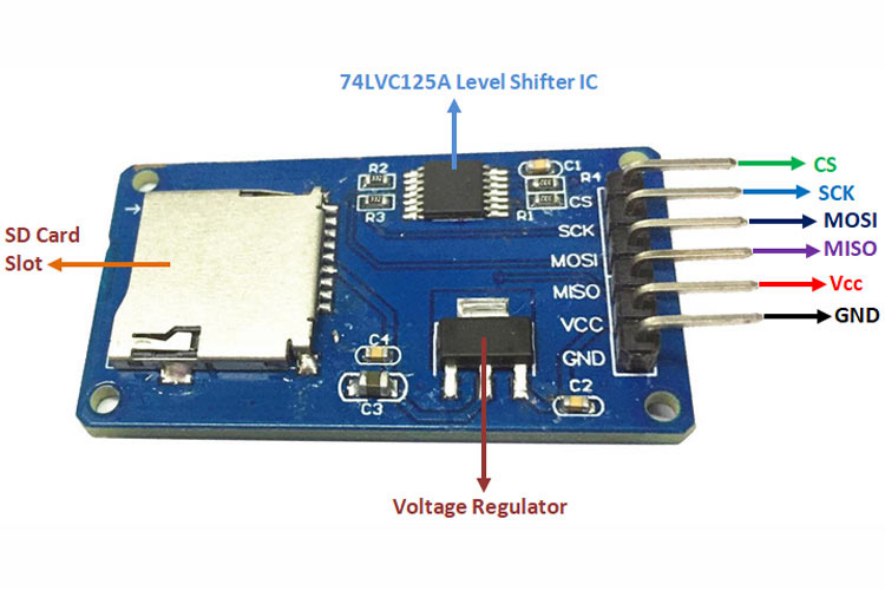
Ví dụ: nếu người dùng nhấn phím ‘2’, khi đó, hàng C và cột 2 sẽ được kết nối với nhau thông qua phím bấm. Nếu vi điều khiển xuất mức thấp (GND) tại hàng C, thì mức điện áp ở chân cột 2 sẽ bị kéo xuống 0V. Nhờ đó, vi điều khiển nhận ra có sự thay đổi và xác định được chính xác phím đang được nhấn. Tương tự, các phím nằm trên cùng hàng C sẽ phản hồi theo cách tương tự khi được quét.

Phương pháp này được áp dụng cho mô hình KeyPad được trình bày dưới đây để nhận diện chính xác thao tác người dùng.

3. Module SD Card với SPI và thư viện FatFS

Một số dòng vi điều khiển STM32 được tích hợp sẵn phần cứng SDIO (hoặc SDMMC) – đây là một giao tiếp tốc độ cao được thiết kế chuyên biệt để làm việc với thẻ nhớ SD. Nhờ vào phần cứng SDIO, tốc độ truyền dữ liệu giữa vi điều khiển và thẻ SD có thể đạt mức tối đa mà chuẩn SD hỗ trợ.

Tuy nhiên, trong trường hợp vi điều khiển không có SDIO, hoặc đối với các dòng STM32 cấp thấp, người dùng vẫn có thể giao tiếp với thẻ SD thông qua giao tiếp SPI – một giao thức phổ biến có mặt trên hầu hết các vi điều khiển, bao gồm tất cả dòng STM32. Mặc dù SPI có tốc độ thấp hơn so với SDIO, nhưng nó đơn giản hơn nhiều trong việc cấu hình và lập trình.

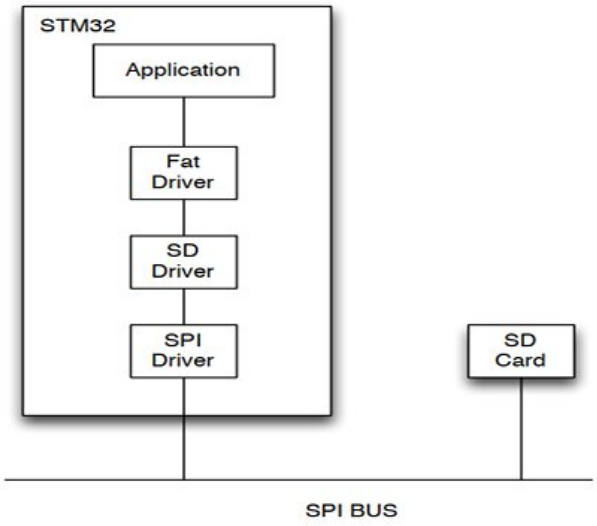


FatFS là một thư viện hệ thống tệp (*file system*) dựa trên chuẩn FAT (File Allocation Table), được phát triển dành riêng cho các hệ thống nhúng. Điểm mạnh của FatFS là được viết hoàn toàn bằng ANSI C và không phụ thuộc vào phần cứng, vì vậy nó có thể hoạt động trên rất nhiều nền tảng và vi điều khiển khác nhau.

FatFS đóng vai trò như một lớp trung gian (Middleware) nằm giữa tầng phần cứng (Hardware) và tầng ứng dụng (Application), giúp lập trình viên dễ dàng thao tác với dữ liệu lưu trữ mà không cần quan tâm đến chi tiết phần cứng bên dưới. FatFS hỗ trợ nhiều dòng vi điều khiển khác nhau như: 8051, PIC, AVR, ARM, Z80, RX, và nhiều loại hệ thống nhúng có tài nguyên giới hạn khác.

Một số hàm thường dùng trong FatFS:

- f_mount(): Gắn hoặc gỡ vùng làm việc với hệ thống tệp.
- f_open(): Mở hoặc tạo mới một tập tin.
- f_close(): Đóng tập tin sau khi sử dụng.
- f_read(): Đọc dữ liệu từ tập tin.
- f_write(): Ghi dữ liệu vào tập tin.
- f_lseek(): Di chuyển con trỏ đọc/ghi đến vị trí chỉ định hoặc mở rộng tập tin.
- f_truncate(): Cắt ngắn độ dài của tập tin.
- f_sync(): Ghi toàn bộ dữ liệu tạm lên thiết bị lưu trữ.
- f_opendir(): Mở thư mục để truy cập các tệp con bên trong.

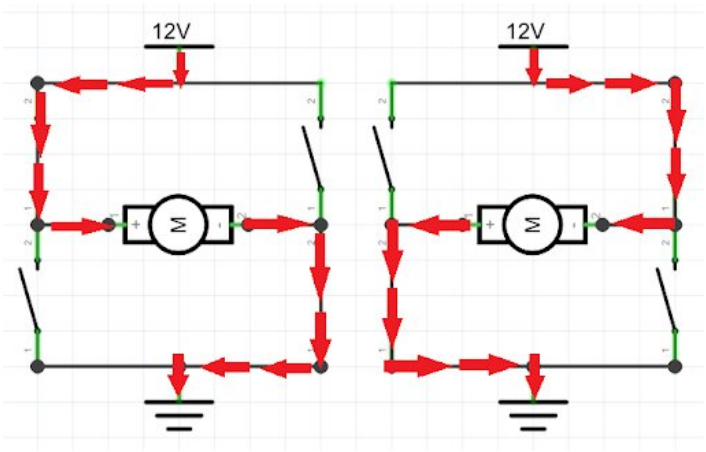


4. LD298N và Motor DC 5V

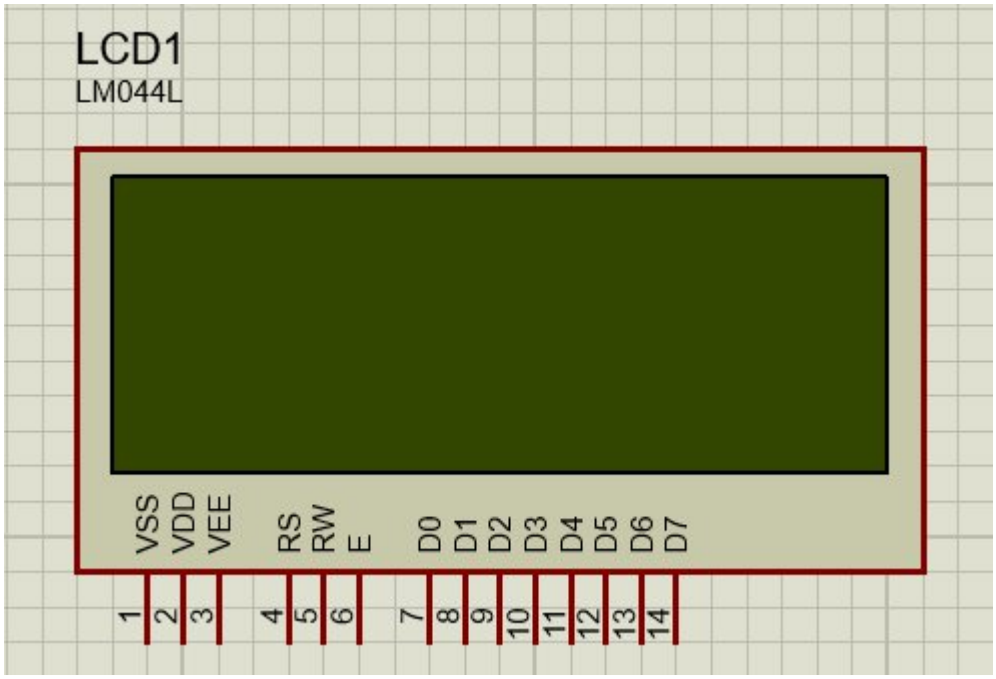
Module L298 là một mạch điều khiển động cơ một chiều DC cùng lúc. L298 là IC điều khiển cầu kép toàn kỳ có dải điện áp hoạt động rộng, xử lý dòng tải có mức tối đa 3A. Bao gồm điện áp bão hòa thấp và bảo vệ quá nhiệt. Có cấu tạo từ hai mạch cầu H transistor.

Nguyên lý hoạt động của mạch cầu H:

- Động cơ DC hòn đảo chiều quay khi thay đổi chiều dòng điện chạy vào động cơ. Do đó ta hoàn toàn có thể đổi chiều cấp điện cho động cơ để làm thay đổi chiều quay. Hình bên dưới là sơ đồ mạch cầu H đơn thuần sử dụng 4 công tắc nguồn. Các công tắc nguồn hoàn toàn có thể sửa chữa thay thế bằng relay hoặc những khóa bán dẫn công suất



5. LCD 20x4



Loại LCD mà em sử dụng là LCD 2004, tức là có 4 hàng và mỗi hang hiển thị được 20 ký tự.

Sơ đồ chức năng các chân LCD:

Thứ tự chân	Tên chân	Chức năng
1	VSS	GND của LCD
2	VCC	Nguồn cấp cho LCD
3	VEE	Điều chỉnh độ tương phản (cần được gắn với biến trở)
4	RS	Chọn thanh ghi RS=0: Đưa LCD vào chế độ ghi lệnh RS=1: Đưa LCD vào chế độ ghi dữ liệu (dữ liệu xuất lên màn hình)
5	RW	Chọn chế độ đọc/ghi LCD RW=0: Vi điều khiển truyền dữ liệu vào LCD RW=1: Vi điều khiển đọc dữ liệu từ LCD
6	E	E=0: Vô hiệu hóa đọc/ghi E=1: Cho phép LCD đọc/ghi E chuyển từ mức 1 về 0: bắt đầu đọc/ghi LCD
7	D0	Dữ liệu bit thứ 0
8	D1	Dữ liệu bit thứ 1
9	D2	Dữ liệu bit thứ 2
10	D3	Dữ liệu bit thứ 3
11	D4	Dữ liệu bit thứ 4
12	D5	Dữ liệu bit thứ 5
13	D6	Dữ liệu bit thứ 6
14	D7	Dữ liệu bit thứ 7
15	LED+	Nguồn dương cấp cho LED nền
16	LED-	Nguồn âm cấp cho LED nền

Tập lệnh giao tiếp LCD:

Khi chân RS = 1 => chế độ đọc/ghi lệnh.

Chân RW = 0 => chế độ ghi lệnh/dữ liệu.

- Chế độ ghi lệnh. Các lệnh sẽ được vi điều khiển truyền cho LCD để thực hiện các lệnh như: khởi tạo LCD, xóa nội dung LCD, chọn chế độ hoạt động, điều khiển con trỏ,…

Mã lệnh	Chức năng	T_{exe}
0x01	Xoá toàn bộ nội dung đang hiển thị trên màn hình.	1.52ms
0x02	Di chuyển con trỏ về vị trí đầu màn hình.	1.52ms
0x06	Tự động di chuyển con trỏ đến vị trí tiếp theo mỗi khi xuất ra LCD 1 ký tự.	37us
0x0C	Bật hiển thị và tắt con trỏ	37us
0x0E	Bật hiển thị và bật con trỏ	37us
0x80	Di chuyển con trỏ về đầu dòng 1	37us
0xC0	Di chuyển con trỏ về đầu dòng 2	37us
0x38	Giao tiếp 8 bit, hiển thị 2 dòng, kích thước font 5x7	37us
0x28	Giao tiếp 4 bit, hiển thị 2 dòng, kích thước font 5x7	37us

Địa chỉ DDRAM và vị trí hiển thị :

Character No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1st line	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
2nd line	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
3rd line	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
4th line	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

Đề con trỏ di chuyển đến các vị trí trong màn hình thì ta sẽ gửi lệnh là địa chỉ của vị trí đó.

Để gửi lệnh ra LCD, ta phải làm theo trình tự sau:

B1: Kéo chân RW xuống mức thấp để chọn chế độ là ghi.

B2: Kéo chân RS xuống mức thấp để cho LCD hiểu là chúng ta muốn ghi lệnh không phải dữ liệu.

B3: gửi byte lệnh ra các chân D7…D0

B4: Tạo xung trên chân E bằng cách cho E xuống mức 0 rồi lên lại mức 1 hoặc ngược lại để cho phép lệnh được ghi vào LCD.

B5: Delay 1 khoảng thời gian để LCD thực hiện xong lệnh.

Để màn hình hiển thị được các ký tự thì ta sẽ gửi dữ liệu theo trình tự sau:

B1: Kéo chân RW xuống mức thấp để chọn chế độ là ghi.

B2: kéo chân RS lên mức cao để LCD hiểu là chúng ta muốn ghi dữ liệu

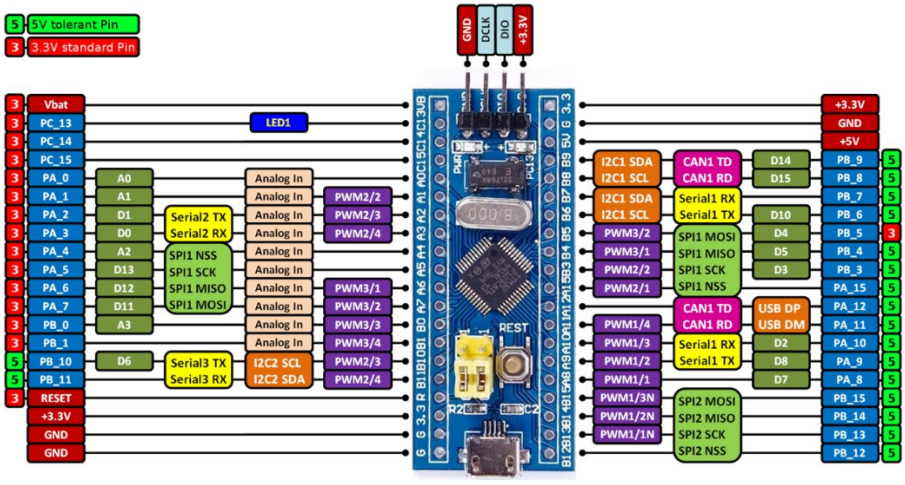
B3: Gửi mã của ký tự (ASCII) ra các chân D7…D0.

B4: Tạo xung trên chân E bằng cách cho E xuống mức 0 rồi lên lại mức 1 hoặc ngược lại để cho phép lệnh được ghi vào LCD.

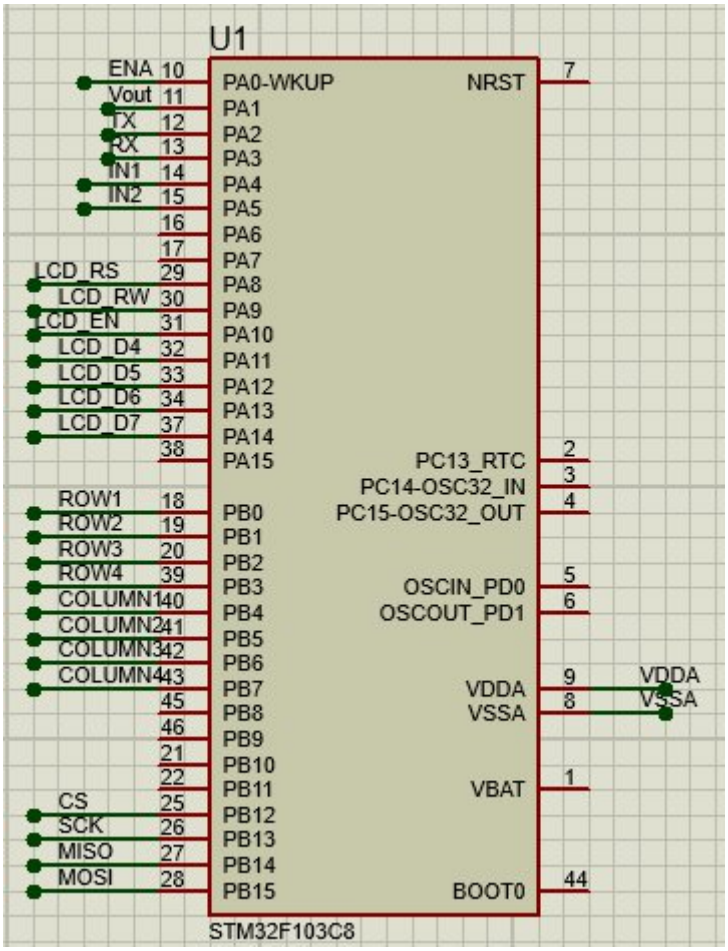
B5: Delay 1 khoảng thời gian để LCD hiển thị ký tự.

6. Vi điều khiển STM32f103c8t6

- STM32F103C8T6 – Tổng quan
  - Dòng chip: STM32F1 (ARM Cortex-M3)
  - Hãng sản xuất: STMicroelectronics
  - Kiến trúc: ARM Cortex-M3, 32-bit
  - Tốc độ xung nhịp: Lên đến 72 MHz
  - Bộ nhớ Flash: 64 KB hoặc 128 KB
  - RAM (SRAM): 20 KB
- Ngoại vi và Giao tiếp
  - GPIO: 37 chân I/O (trong tổng số 48 chân)
  - ADC: 10 kênh 12-bit
  - USART: 2 cổng (USART1, USART2)
  - SPI: 2 cổng SPI
  - I2C: 2 cổng I2C
  - Timers: 3 bộ định thời (16-bit), 1 timer hệ thống SysTick
  - PWM: Hỗ trợ trên các timer
  - USB: Có tích hợp USB 2.0 FS (thiết bị)
  - DMA: 7 kênh DMA hỗ trợ truyền dữ liệu nhanh
- Thông số kỹ thuật khác
  - Điện áp hoạt động: 2.0V – 3.6V
  - Nhiệt độ hoạt động: –40°C đến +85°C
  - Package: LQFP-48



III. Kết nối phần cứng



- PA0: là chân xung PWM nối với L289
- PA1: là chân ADC đọc dữ liệu từ cảm biến
- PA2, PA3: là các chân TX và RX của UART
- PA4, PA5: nối IN1, IN2 của L289 để đổi chiều động cơ
- PA8, PA9, PA10, PA11, PA12, PA13, PA14: lần lượt nối các chân RS, RW, EN, D4, D5, D6, D7 của LCD
- PB0, PB1, PB2, PB3: là các hàng của Keypad
- PB4, PB5, PB6, PB7: là các cột của keypad
- PB12, PB13, PB14, PB15: lần lượt nối CS, SCK, MISO, MOSI của SD card adapter.

IV. Thiết kế chương trình

Trong project này em lập trình vi điều khiển bằng thanh ghi (can thiệp trực tiếp vào tầng thanh ghi của STM32 để thiết lập xung clock và các ngoại vi cho vi điều khiển). Em sử dụng file config.h để khai báo địa chỉ của các thanh ghi được sử dụng.

Do phần mềm mô phỏng proteus không hỗ trợ mô phỏng hệ điều hành thời gian thực RTOS nên em sẽ làm thêm một chương trình khác sử dụng vòng lặp while và ngắt để mô phỏng proteus

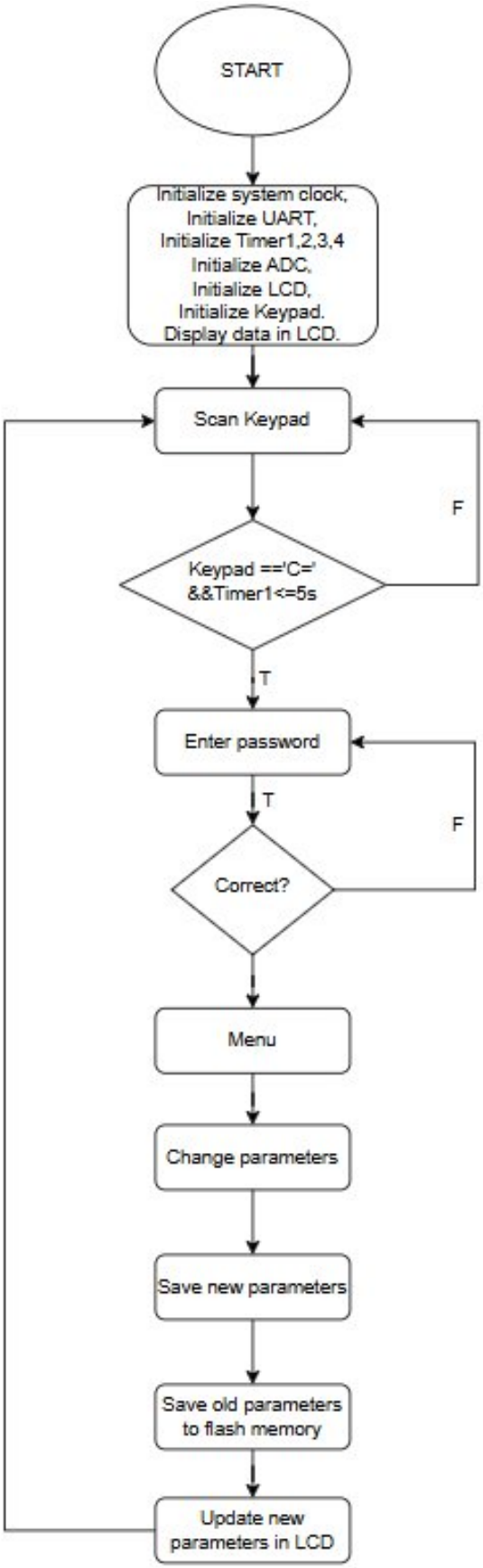
Ngoài ra em cũng được giao thêm một bài tập khác là ghi dữ liệu vào thanh ghi theo phương thức xoay vòng, nghĩa là khi dữ liệu trong thẻ nhớ đã đầy thì sẽ xóa file lưu dữ liệu cũ nhất đi và lưu dữ liệu mới vào file đó.

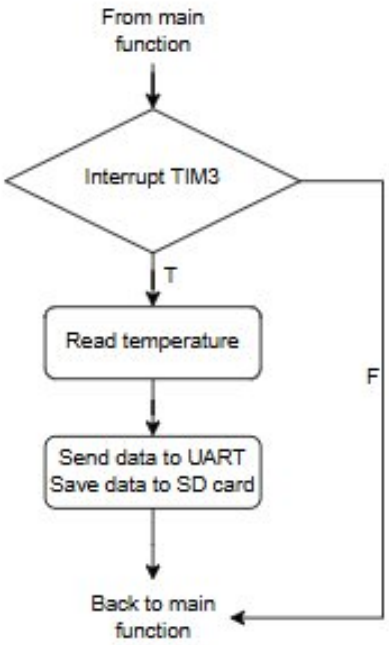
A. Chương trình sử dụng vòng lặp While

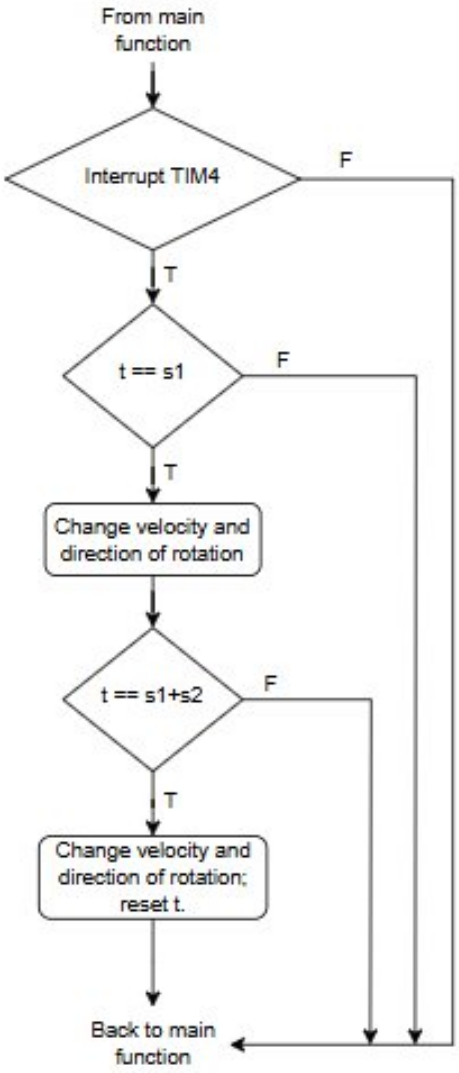
Trong chương trình này để có thể thực hiện đầy đủ được các yêu cầu của đề bài, em sẽ sử dụng 2 ngắt Timer: 1 cái để gửi dữ liệu mỗi 1s qua UART và lưu vào SD card, 1 cái để điều khiển động cơ thay đổi tốc độ theo xung PWM.

Vòng while chính trong hàm main sẽ được sử dụng để quét keypad phục vụ cho chế độ config.

Lưu đồ thuật toán:


START
Initialize system clock,
Initialize UART,
Initialize Timer1,2,3,4
Initialize ADC,
Initialize LCD,
Initialize Keypad.
Display data in LCD.
Scan Keypad
F
Keypad =='C='
&&Timer1<=5s
T
Enter password
T
F
Correct?
Menu
Change parameters
Save new parameters
Save old parameters
to flash memory
Update new
parameters in LCD


From main
function
Interrupt TIM3
T
F
Read temperature
Send data to UART
Save data to SD card
Back to main
function


From main
function
Interrupt TIM4
F
T
t == s1
F
T
Change velocity and
direction of rotation
t == s1+s2
F
T
Change velocity and
direction of rotation;
reset t.
Back to main
function

B. Chương trình sử dụng hệ điều hành RTOS

Trong chương trình này em sẽ sử dụng FreeRTOS.

Chương trình gồm có 2 task:

DefaultTask: quét keypad liên tục phục vụ cho chế độ config

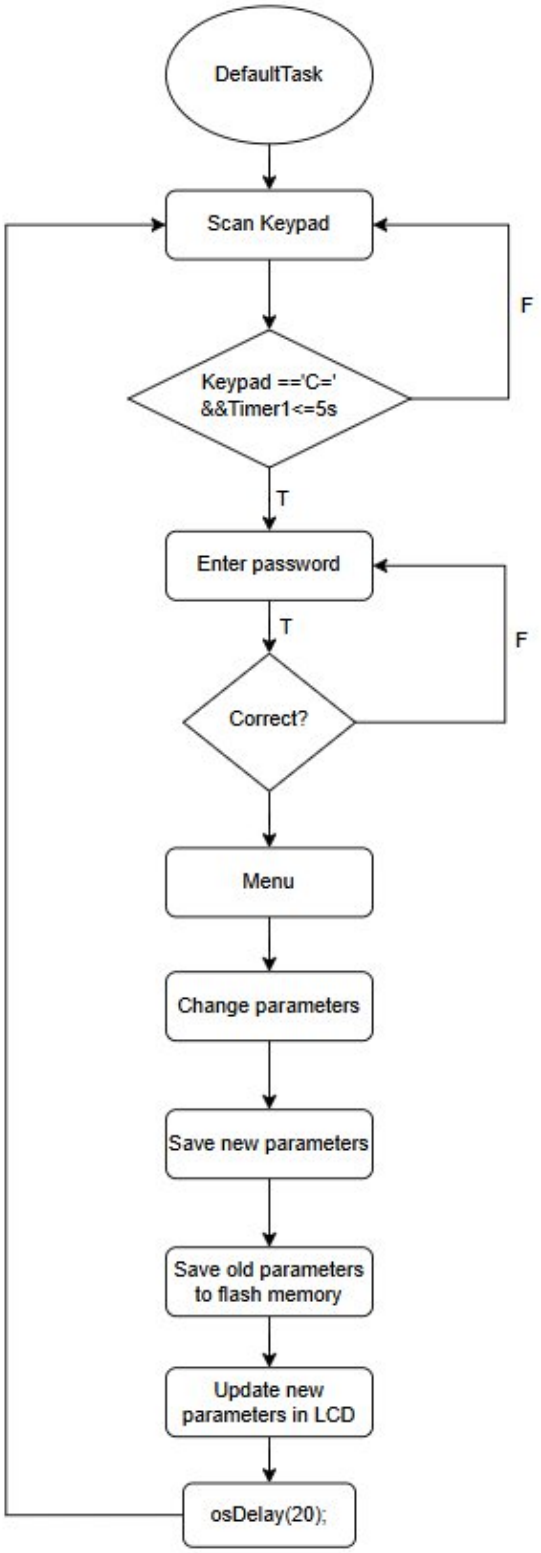
Task2: điều khiển động cơ, gửi dữ liệu qua UART và lưu dữ liệu vào SD card.

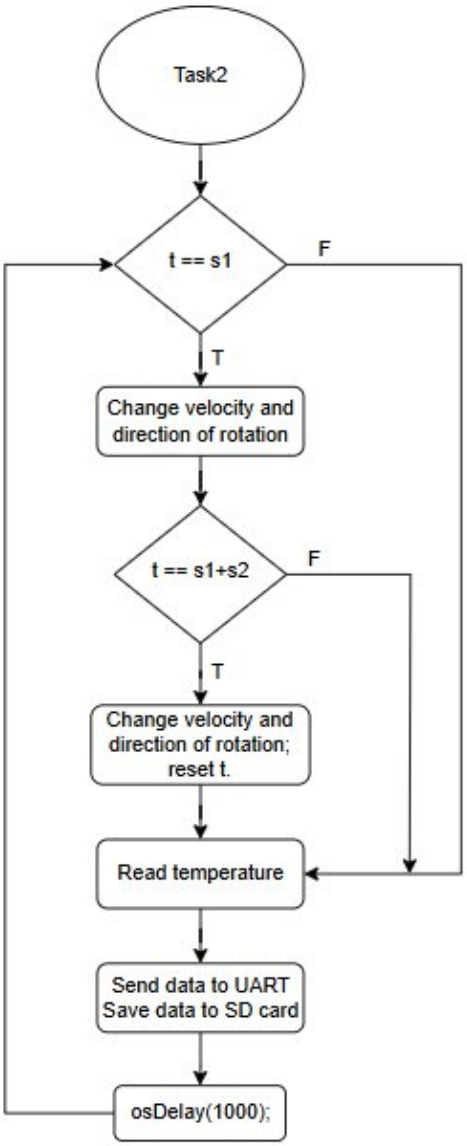
Mức ưu tiên cho các task:

DefaultTask: mức ưu tiên bình thường và thời gian delay là 20ms.

Task2: mức ưu tiên cao và có thời gian delay là 1s.

Lưu đồ thuật toán:

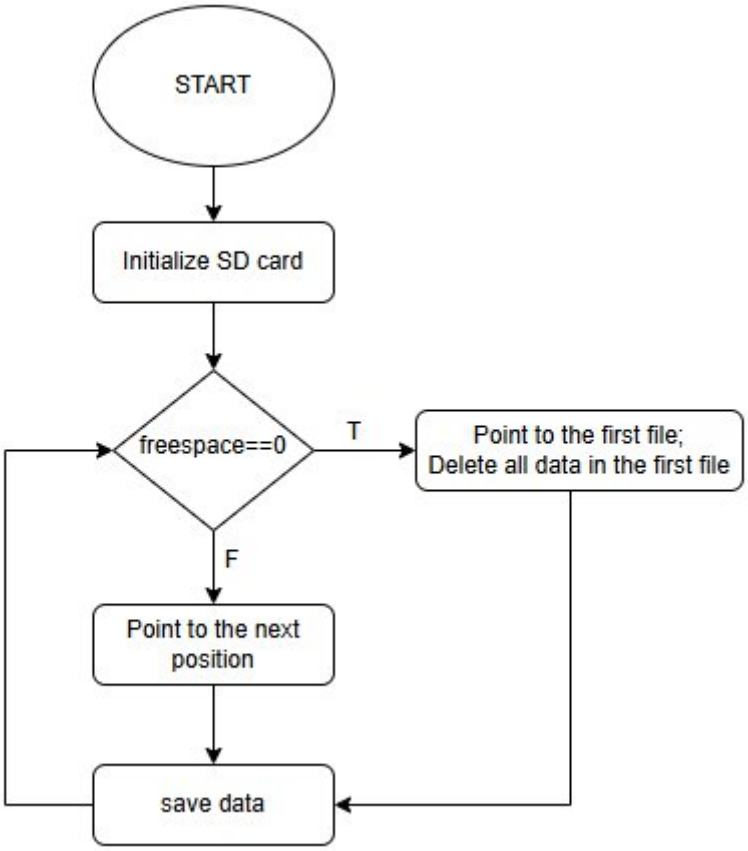

DefaultTask
Scan Keypad
F
Keypad =='C='
&&Timer1<=5s
T
Enter password
T
F
Correct?
Menu
Change parameters
Save new parameters
Save old parameters
to flash memory
Update new
parameters in LCD
osDelay(20);


Task2
t == s1
F
T
Change velocity and
direction of rotation
t == s1+s2
F
T
Change velocity and
direction of rotation;
reset t.
Read temperature
Send data to UART
Save data to SD card
osDelay(1000);

C. Chương trình lưu dữ liệu vào thẻ nhớ theo cơ chế xoay vòng

Mỗi khi khởi động MCU sẽ tạo một file mới và lưu dữ liệu, tuy nhiên dung lượng của thẻ nhớ có hạn nên khi thẻ nhớ đã đầy, dữ liệu sẽ được ghi đè lên file đầu tiên.

Lưu đồ thuật toán:



V. Kết quả

- Code: [hieuxuanhn2703/Viettel_Test]
- Video mô phỏng: [https://youtu.be/6mRbfSQJ0OA]