

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

-----o0o-----



LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

THIẾT KẾ ỨNG DỤNG XEM ẢNH SỐ

GVHD: Ths.Bùi Quốc Bảo

SVTH: Nguyễn Vũ Hùng

MSSV: 41001338

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2014

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
TRƯỜNG ĐẠI HỌC BÁCH KHOA

Độc lập – Tự do – Hạnh phúc.

----- ☆ -----

----- ☆ -----

Số: _____ /BKĐT

Khoa: **Điện – Điện tử**

Bộ Môn: **Điện Tử**

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

1. HỌ VÀ TÊN: NGUYỄN VŨ HÙNG MSSV: 41001338

2. NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG** LỚP : DD10DV3

3. Đề tài: THIẾT KẾ ỨNG DỤNG XEM ẢNH SỐ

4. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):

.....
.....
.....
.....
.....
.....

5. Ngày giao nhiệm vụ luận văn:

6. Ngày hoàn thành nhiệm vụ:

7. Họ và tên người hướng dẫn: Phân hướng dẫn

.....
.....

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

Tp.HCM, ngày..... tháng..... năm 2014

CHỦ NHIỆM BỘ MÔN

NGƯỜI HƯỚNG DẪN CHÍNH

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):

Đơn vị:

Ngày bảo vệ:

Điểm tổng kết:

Nơi lưu trữ luận văn:

LỜI CẢM ƠN

Trước hết cảm ơn ba mẹ, gia đình là nguồn động lực to lớn, là chỗ dựa tinh thần vững chắc nhất của con trong suốt quá trình học tập.

Kính gửi đến thầy BÙI QUỐC BẢO lời cảm ơn chân thành và sâu sắc, cảm ơn thầy đã chỉ bảo, hướng dẫn tận tình trong suốt quá trình em làm đồ án cho đến khi hoàn thành luận văn tốt nghiệp.

Em xin cảm ơn tất cả quý thầy cô trường Đại học Bách Khoa TPHCM nói chung, quý thầy cô khoa Điện-Điện tử và bộ môn Điện tử nói riêng đã tận tình giảng dạy, trang bị cho em những kiến thức bổ ích trong suốt quá trình học đại học.

Tôi xin cảm ơn tất cả bạn bè đẽ động viên, giúp đỡ tôi trong suốt quá trình học tập và làm luận văn tốt nghiệp.

Tôi xin cảm ơn tất cả.

Tp. Hồ Chí Minh, ngày tháng năm .

Sinh viên

TÓM TẮT LUẬN VĂN

Nhiệm vụ chính của luận văn là thiết kế ứng dụng xem ảnh kỹ thuật số có định dạng BMP, JPEG từ thẻ nhớ ra màn hình màu TFT 7 inch có độ phân giải 800 x 480, có cảm ứng điện trở để tương tác với màn hình.

Ứng dụng được thực hiện qua 2 giai đoạn chính: thiết kế phần cứng và thiết kế phần mềm

Quá trình thiết kế phần cứng được thực hiện như sau: Sử dụng phần mềm Altium để vẽ mạch nguyên lý và layout PCB kết nối đến màn hình LCD. Phần cứng của ứng dụng sử dụng vi điều khiển ARM Coortex M4 – STM32F429IIT6 của ST tần số hoạt động tối đa 180Mhz, có embedded graphic acerlator để giao tiếp trực tiếp với LCD thông qua các chân HSYNC, VSYNC, CLK, EN, R[5:0], G[5:0], B[5:0] mà không thông qua driver ngoài. Ngoài ra PCB còn có thêm các phần khác như SDRAM 256Mbit dùng làm buffer frame cho dữ liệu hình ảnh, IC ADS7843 để điều khiển cảm ứng điện trở, khe cắm thẻ nhớ,...

Phần mềm của ứng dụng có nhúng hệ điều hành thời gian thực FreeRTOS để quản lý các tác vụ. Để thực hiện việc đọc ảnh trước tiên ta cần quét tất cả các tập tin trong thẻ nhớ để xác định những tập tin nào là ảnh nhằm thực hiện việc đọc dữ liệu và xuất ra màn hình LCD.

MỤC LỤC

1. GIỚI THIỆU.....	1
1.1 Tổng quan.....	1
1.2 Nhiệm vụ luận văn.....	1
2. ĐẶC TÀ SẢN PHẨM	2
3. ĐẶC TÀ KỸ THUẬT	2
3.1 LCD-TFT Controller.....	2
3.2 Chuẩn truyền thông SPI	4
4. THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG.....	5
4.1 Sơ đồ khái hệ thống	5
4.2 Khối vi điều khiển	7
4.2.1 Giới thiệu về vi điều khiển STM32F429IIT6	9
4.2.2 Tổng quan chức năng của STM32F429IIT6	12
4.3 Khối nguồn.....	18
4.4 Khối SDRAM.....	19
4.5 Khối LCD-TFT và resistive touch panel.....	22
4.6 Khối thẻ nhớ SDCard.....	26
4.7 Khối nạp chương SWD (Serial Wire Debug).....	27
4.8 Sơ đồ nguyên lý khôi ngoại vi I/O.....	27
5. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM	28
5.1 Cấu trúc phần mềm	28
5.2 Các middleware chính dùng trong ứng dụng	29
5.2.1 FatFs File System	29
5.2.2 STemWin	32
5.2.3 CMSIS-RTOS Module.....	39
5.3 Thiết kế các hàm chức năng	43
5.4 Các phần mềm cần dùng trong thiết kế ứng dụng	50

6.	KẾT QUẢ THỰC HIỆN.....	53
6.1	Kết quả thực hiện phần cứng.....	53
7.	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	62
7.1	Kết luận	62
7.2	Hướng phát triển.....	63
8.	TÀI LIỆU THAM KHẢO	64
9.	PHỤ LỤC	65

DANH SÁCH HÌNH MINH HỌA

Hình 3.1 Cấu trúc LCD-TFT	3
Hình 4.1 Sơ đồ nguyên lý khói toàn bộ hệ thống	6
Hình 4.2 Sơ đồ nguyên lý khói MCU	7
Hình 4.3 Ảnh minh họa các chân I/O của vi điều khiển STM32F429IIT6 được sử dụng	8
Hình 4.4 Sơ đồ khói STM32F429xx.....	11
Hình 4.5 Sơ đồ khói Multi AHB matrix.....	13
Hình 4.6 Chi tiết chân vi điều khiển	17
Hình 4.7 Sơ đồ nguyên lý khói nguồn	18
Hình 4.8 Sơ đồ nguyên lý SDRAM	19
Hình 4.9 minh họa SDRAM MT48LC16M16A2P-75 IT.....	19
Hình 4.10 Sơ đồ khói SDRAM MT48LC16M16A2P	21
Hình 4.11 Sơ đồ nguyên lý khói LCD-TFT và resistive touch panel	22
Hình 4.12 Sơ đồ chân chip ADS7846.....	24
Hình 4.13 Sơ đồ khói chip ADS7846	25
Hình 4.14 Sơ đồ nguyên lý khói SDCard.....	26
Hình 4.15 Sơ đồ nguyên lý khói SWD	27
Hình 4.16 Sơ đồ nguyên lý khói ngoại vi I/O	28
Hình 5.1 Cấu trúc Firmware STM32CubeF4.....	28
Hình 5.2 Cấu trúc module FatFs.....	30
Hình 5.3 Cấu trúc module Middleware FatFs	32
Hình 5.4 Hình minh họa STemwin	33
Hình 5.5 Cấu trúc của STemwin.....	34
Hình 5.6 STemwin Component	35
Hình 5.7 Hiệu ứng alpha blending.....	36
Hình 5.8 minh họa một Sprite	36
Hình 5.9 Minh họa một Cursor.....	37

Hình 5.10 Hiệu ứng xoay và scale dùng memory device.....	37
Hình 5.11 Một số ví dụ về Widget.....	38
Hình 5.12 Minh họa VNC server.....	39
Hình 5.13 Tô chúc mã nguồn FreeRTOS	40
Hình 5.14 Cấu trúc module CMSIS RTOS	41
Hình 5.15 Minh họa tác vụ GUIThread	44
Hình 5.16 Minh họa tác vụ Timer Callback	44
Hình 5.17 Minh họa hàm k_CalendarBkupInit().....	45
Hình 5.18 Cấu hình LCD controller	46
Hình 5.19 Cân chỉnh màn hình cảm ứng.....	47
Hình 5.20 Khởi động menu	48
Hình 5.21 Khởi động thẻ nhớ	49
Hình 5.22 Khởi động bộ nhớ pool	49
Hình 5.23 Minh họa trình biên dịch Keil ARM	50
Hình 5.24 Hình minh họa STM32 ST-LINK Utility	51
Hình 5.25 Minh họa giao diện phần mềm emWin GUIBuilder	52
Hình 5.26 Giao diện phần mềm tạo Font chữ	52
Hình 5.27 Hình minh họa visual studio	53
Hình 6.1 Hình minh họa phần mềm Altium	53
Hình 6.2 Kết quả thiết kế mạch lớp Top trên Altium	54
Hình 6.3 Kết quả thiết kế mạch lớp Bottom trên Altium.....	55
Hình 6.4 Mạch in lớp Top	56
Hình 6.5 Mạch in lớp Bottom.....	57
Hình 6.6 Kết quả board mạch mặt trên sau khi hoàn thành	58
Hình 6.7 Kết quả board mạch mặt dưới sau khi hoàn thành	59
Hình 6.8 Một số kết quả phần mềm thu được	60
Hình 6.9 Một số kết quả phần mềm thu được	61

Hình 6.10 Một số kết quả phần mềm thu được	62
Hình 9.1 Sơ đồ mạch khôi MCU	65
Hình 9.2 Sơ đồ nguyên lý khôi nguồn	66
Hình 9.3 Sơ đồ nguyên lý khôi bộ nhớ SDRAM.....	67
Hình 9.4 Sơ đồ nguyên lý khôi LCD và Touch.....	68
Hình 9.5 Sơ đồ nguyên lý khôi thẻ nhớ	69
Hình 9.6 Sơ đồ nguyên lý khôi ngoại vi I/O	69
Hình 9.7 Sơ đồ nguyên lý khôi SWD	70

DANH SÁCH BẢNG SỐ LIỆU

Bảng 5.1 Bảng các API FreeRTOS.....	40
Bảng 5.2 Bảng các API của module CMSIS RTOS	42

1. GIỚI THIỆU

1.1 Tổng quan

Ngày nay với sự phát triển vũ bão của khoa học công nghệ nói chung và lĩnh vực hệ thống nhúng nói riêng ngày càng nhiều thiết bị ra đời tích hợp rất nhiều chức năng chuyên biệt trong hầu hết các lĩnh vực từ y tế, hàng không vũ trụ đến điện tử dân dụng nhằm phục vụ đời sống con người tốt hơn như Laptop, điện thoại, Ipad, máy nghe nhạc, máy đo nhịp tim, máy đo nồng độ cồn, tủ lạnh, máy giặt, máy rửa chén, máy điều hòa nhiệt độ... Lĩnh vực hệ thống nhúng đã len lỏi trong từng ngõ ngách trong cuộc sống chúng ta, trong hầu hết các thiết bị điện tử hiện đại.

Cùng với xu hướng phát triển hệ thống nhúng đó, đề tài luận văn của tôi là tìm hiểu và phát triển ứng dụng xem ảnh kỹ thuật số nhằm thực hiện toàn bộ các khâu phát triển 1 ứng dụng nhúng từ lên ý tưởng đến thiết kế board mạch dựa trên những IC rời rạc cho đến nhúng phần mềm nhúng vào phần cứng để thực hiện được chức năng mình mong muốn.

Nhiệm vụ chính của luận văn là thiết kế phần cứng, phần mềm để phát triển ứng dụng xem ảnh số từ thẻ nhớ.

1.2 Nhiệm vụ luận văn

Đề tài có 2 nội dung chính cần thực hiện:

Nội dung 1: Thiết kế board mạch để điều khiển LCD.

Yêu cầu: PCB phải có đầy đủ chức năng để điều khiển màn hình LCD cảm ứng điện trở 7 inch, có SDRAM để làm bộ nhớ đệm cho dữ liệu, khe cắm thẻ nhớ để lưu tập tin hình ảnh.

Kết quả cần đạt: PCB phải hoạt động tốt, ổn định, hạn chế nhiễu EMI ở tần số cao và cung cấp đủ dòng, áp cho LCD hoạt động.

Nội dung 2: Thiết kế phần mềm cho ứng dụng

Yêu cầu và kết quả cần đạt được:

- Phần mềm có khả năng tìm các tập tin ảnh định dạng BMP, JPEG trong thẻ nhớ.
- Hiển thị ảnh lên LCD mượt, màu sắc ảnh chân thực, độ phân giải cao.
- Có giao diện điều khiển để người dùng chọn chế độ hiển thị.

2. ĐẶC TẢ SẢN PHẨM

- Ứng dụng được dùng để hiển thị hình ảnh từ thẻ nhớ ra màn hình LCD-TFT cảm ứng điện trở 7 inch, độ phân giải 800 x 480.
- Chức năng:
 - Có nhúng hệ điều hành thời gian thực FreeRTOS
 - Xem ảnh định dạng BMP, JPEG từ thẻ nhớ.
 - Có giao diện đồ họa cho người dùng tương tác thông qua các button để Next, Previous, Play, Stop, cài đặt thời gian slide ảnh.
 - Ngôn ngữ: Tiếng việt
 - Có hiển thị thời gian thực của hệ thống

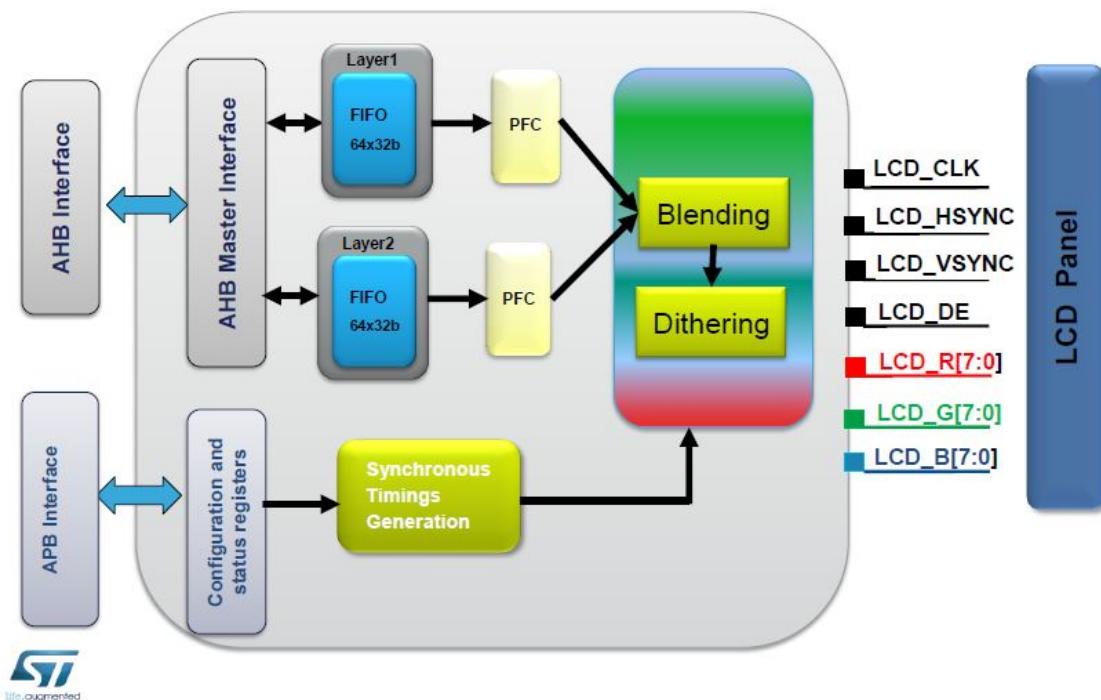
3. ĐẶC TẢ KĨ THUẬT

- Ngõ vào:
 - Giao tiếp LCD với MCU qua LCD-TFT Controller và phần cứng tăng tốc xử lý đồ họa Chrom-ART AcceleratorTM được tích hợp trong chip STM32F429IIT6.
 - Giao tiếp SDRAM data bus 16 bit bằng Flexible Memory Controller.
 - Giao tiếp thẻ nhớ SDCard qua chuẩn SDIO.
 - Giao tiếp Touch Screen Controller qua chuẩn SPI.
- Ngõ ra: Slide hình ảnh ra màn hình LCD-TFT 7 inch và các giao diện đồ họa
- Tương tác với người dùng: Qua Resistive Touch Screen Panel.

3.1 LCD-TFT Controller

- LCD-TFT Display Controller hỗ trợ giao tiếp chế độ song song đến 24 bit màu RGB và truyền tất cả các tín hiệu giao tiếp trực tiếp đến hầu hết các LCD-TFT với độ phân giải lên đến XGA (1024X768) với 1 vài đặt trưng như sau:
 - 2 Layer hiển thị với bộ đệm FIFO mỗi layer là 64x32 bit.

- Color Look-Up Table (CLUT) lên đến 256 màu (256x24 bit) mỗi layer.
- Có đến 8 ngõ vào màu định dạng có thể chọn cho mỗi layer
- Pha trộn màu linh hoạt giữa 2 layer để dùng làm giá trị alpha.
- Có các thông số lập trình linh hoạt cho mỗi layer.
- Color keying (màu trong suốt).
- Hỗ trợ đến 4 sự kiện ngắt.



Hình 3.1 Cấu trúc LCD-TFT

- LCD_CLK (LCD Pixel clock): được dùng để control panel refresh rate.
- LCD_EN: được dùng để enable LCD data.
- VSYNC (Vertical Sync for TFT): được dùng để reset hàng LCD trở đến vị trí trên cùng của LCD.
- HSYNC (Horizontal Sync for TFT): được dùng để reset cột LCD trở đến vị trí cuối cùng của LCD.
- LCD_R[7:0]: 8 bit Red data.
- LCD_G[7:0]: 8 bit Green data.
- LCD_B[7:0]: 8 bit Blue data.

3.2 Chuẩn truyền thông SPI

SPI (Serial Peripheral Bus) là một chuẩn truyền thông nối tiếp tốc độ cao do hãng Motorola đề xuất. Đây là kiểu truyền thông Master-Slave, trong đó có 1 chip Master điều phối quá trình truyền thông và các chip Slaves được điều khiển bởi Master vì thế truyền thông chỉ xảy ra giữa Master và Slave. SPI là một cách truyền song công (full duplex) nghĩa là tại cùng một thời điểm quá trình truyền và nhận có thể xảy ra đồng thời. SPI đôi khi được gọi là chuẩn truyền thông “4 dây” vì có 4 đường giao tiếp trong chuẩn này đó là SCK (Serial Clock), MISO (Master Input Slave Output), MOSI (Master Ouput Slave Input) và SS (Slave Select)

SCK: Xung giữ nhịp cho giao tiếp SPI, vì SPI là chuẩn truyền đồng bộ nên cần 1 đường giữ nhịp, mỗi nhịp trên chân SCK báo 1 bit dữ liệu đến hoặc đi. Sự tồn tại của chân SCK giúp quá trình truyền ít bị lỗi và vì thế tốc độ truyền của SPI có thể đạt rất cao. Xung nhịp chỉ được tạo ra bởi chip Master.

MISO – Master Input / Slave Output: nếu là chip Master thì đây là đường Input còn nếu là chip Slave thì MISO lại là Output. MISO của Master và các Slaves được nối trực tiếp với nhau..

MOSI – Master Output / Slave Input: nếu là chip Master thì đây là đường Output còn nếu là chip Slave thì MOSI là Input. MOSI của Master và các Slaves được nối trực tiếp với nhau.

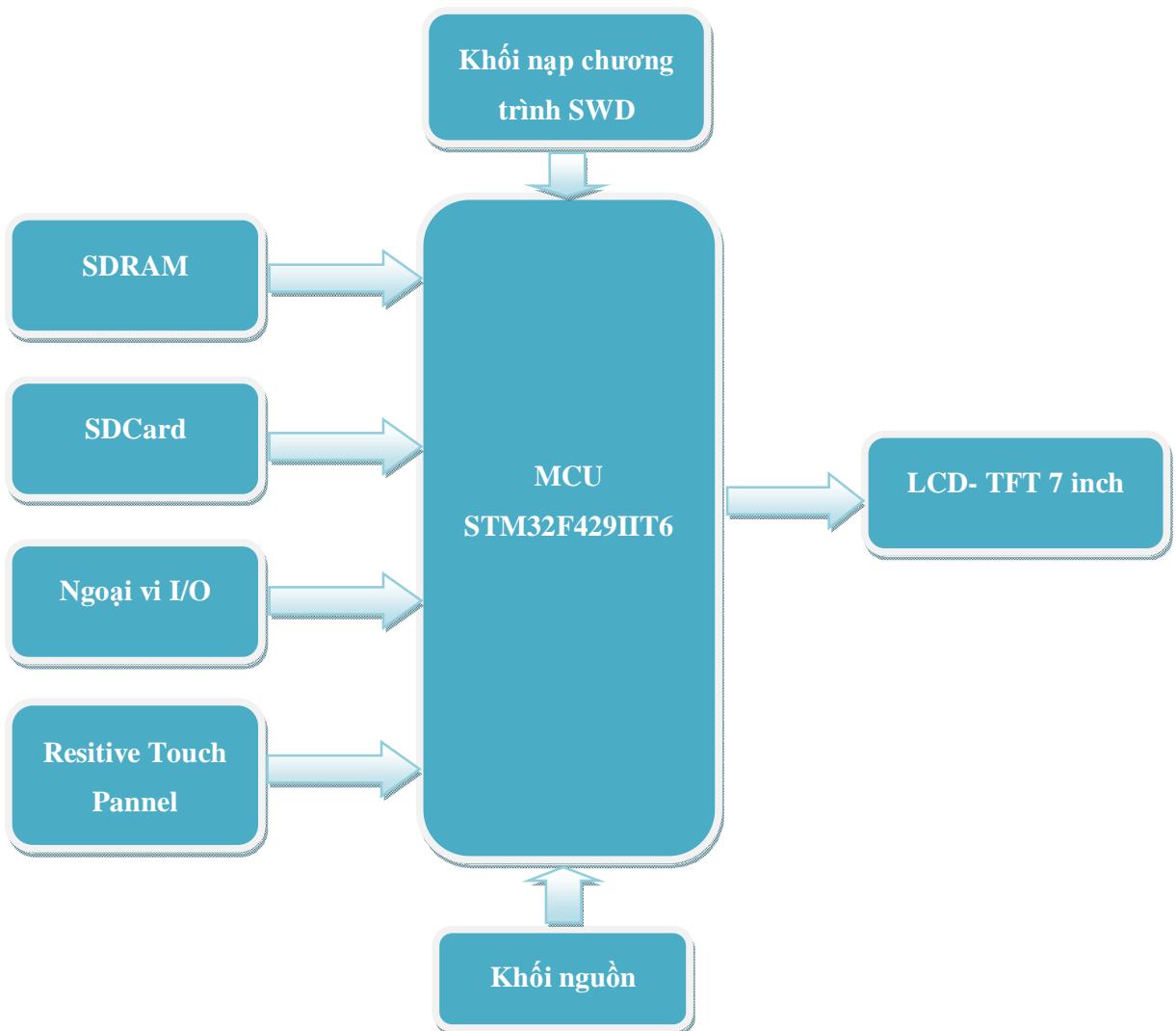
SS – Slave Select: SS là đường chọn Slave cần giao tiếp, trên các chip Slave đường SS sẽ ở mức cao khi không làm việc. Nếu chip Master kéo đường SS của một Slave nào đó xuống mức thấp thì việc giao tiếp sẽ xảy ra giữa Master và Slave đó. Chỉ có 1 đường SS trên mỗi Slave nhưng có thể có nhiều đường điều khiển SS trên Master, tùy thuộc vào thiết kế của người dùng.

Hoạt động: mỗi chip Master hay Slave có một thanh ghi dữ liệu 8 bits. Cứ mỗi xung nhịp do Master tạo ra trên đường giữ nhịp SCK, một bit trong thanh ghi dữ liệu của Master được truyền qua Slave trên đường MOSI, đồng thời một bit trong thanh ghi dữ liệu của chip Slave cũng được truyền qua Master trên đường MISO. Do 2 gói dữ liệu trên 2 chip được gửi qua lại đồng thời nên quá trình truyền dữ liệu này được gọi là “song công”.

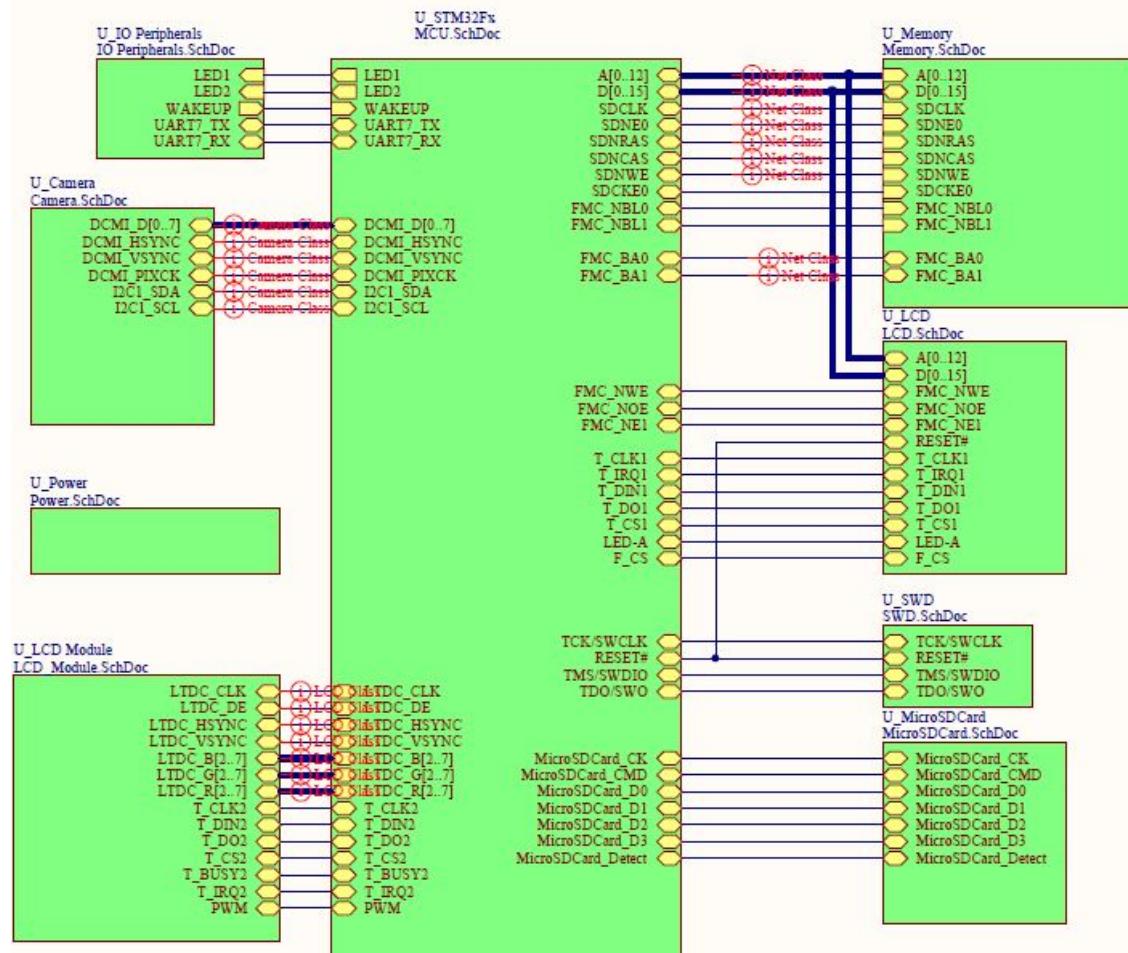
4. THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG

- Yêu cầu phần cứng:
 - Sử dụng vi điều khiển phù hợp để ứng dụng hoạt động có hiệu quả nhất
 - Có RAM ngoài dung lượng đủ lớn để làm bộ nhớ đệm
 - Có Resitive Touch Screen
 - Có khe cắm thẻ nhớ SDCard
- Phần mềm thiết kế mạch: Altium 14

4.1 Sơ đồ khối hệ thống



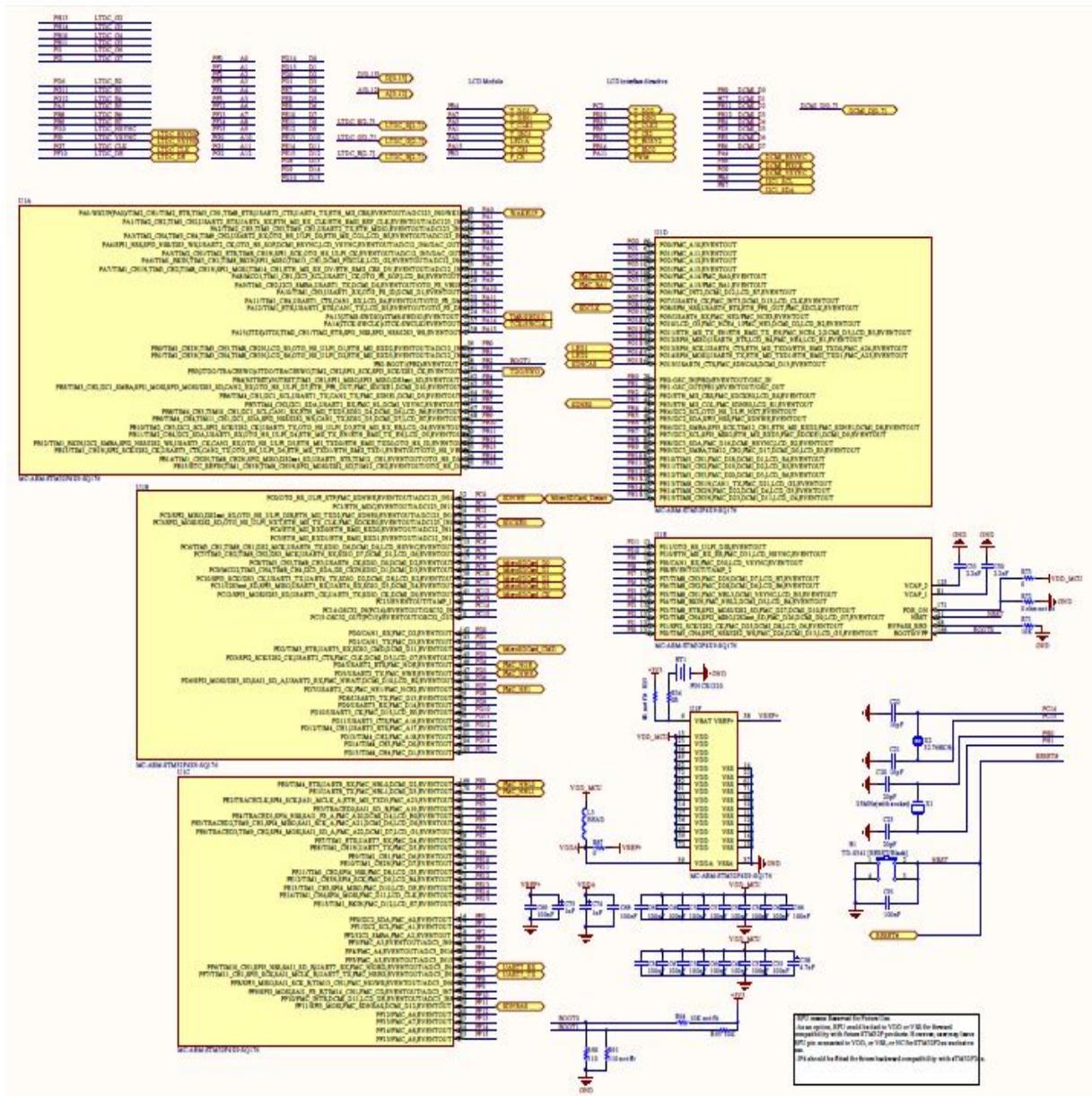
Ứng dụng có 8 khối chính: Khối cấp nguồn, khói xử lý trung tâm, khói LCD-TFT, khói bộ nhớ SDRAM, khói thẻ nhớ SDCard, khói cảm ứng điện trở, khói nạp chương trình và khói ngoại vi I/O.



Hình 4.1 Sơ đồ nguyên lý khối toàn bộ hệ thống

4.2 Khối vi điều khiển

Khối vi điều khiển trung tâm có chức năng nhận và xử lý tất cả các tín hiệu, dữ liệu để điều khiển LCD-TFT, resistive touch panel, SDRAM, SDCard.



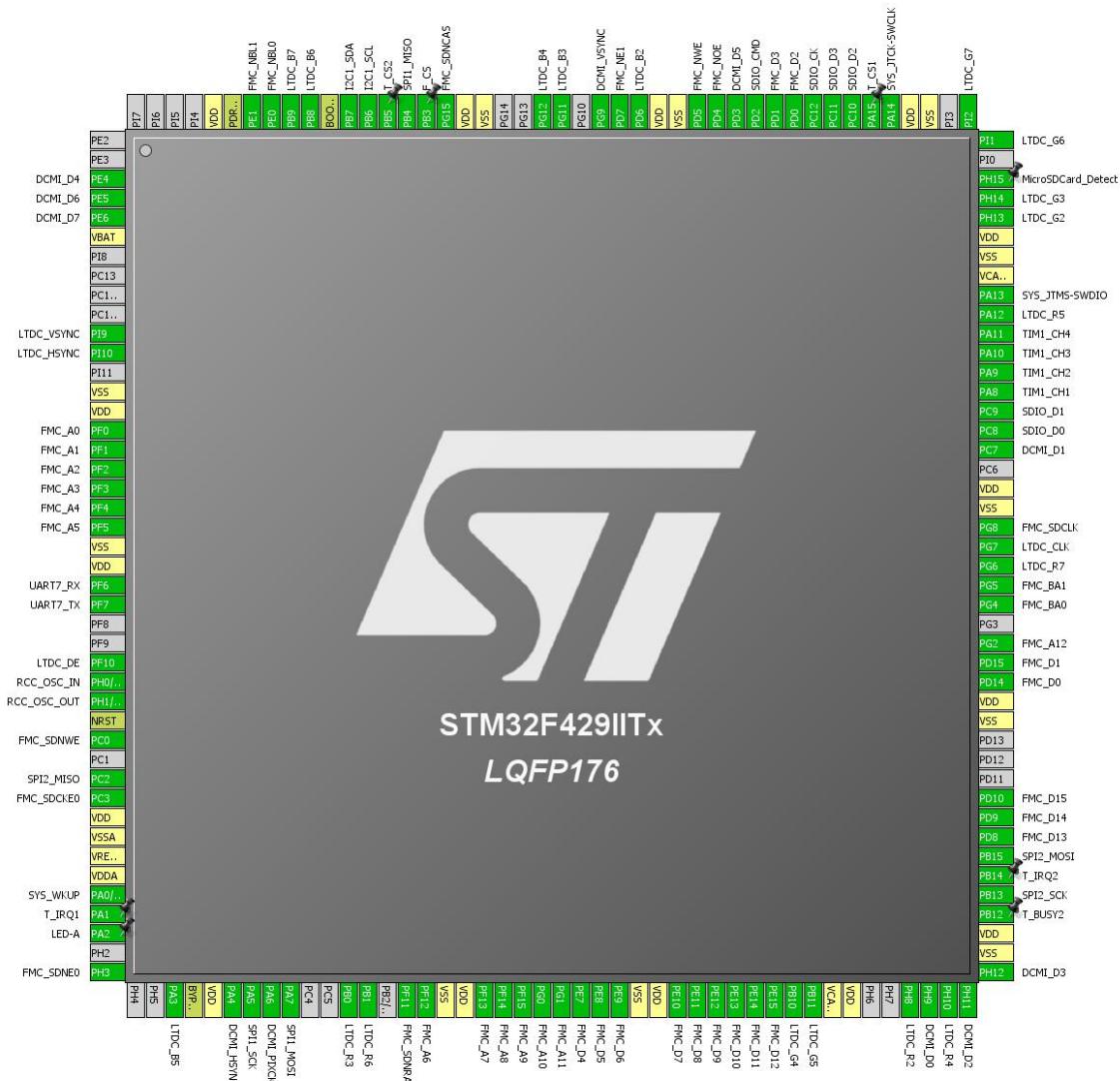
Hình 4.2 Sơ đồ nguyên lý khối MCU

Các thành phần chính trong khối MCU:

- Vi điều khiển STM32F429IIT6
- Thạch anh 25 Mhz và thạch anh dao động 32.678 Khz
- Nút nhấn Reset MCU

- Pin CR1220 3V để nuôi RTC nhằm lưu thời gian hệ thống
- Ferarit Bead để lọc nhiễu tần số cao
- Các tụ điện decoupling, tụ Vcap
- Các đường tín hiệu LCD, SDIO, SPI, SDRAM, SWD để nối đến các khối khác.

Vì điều khiển được sử dụng cho ứng dụng là STM32F429IIT6 loại LQFP 176 pin



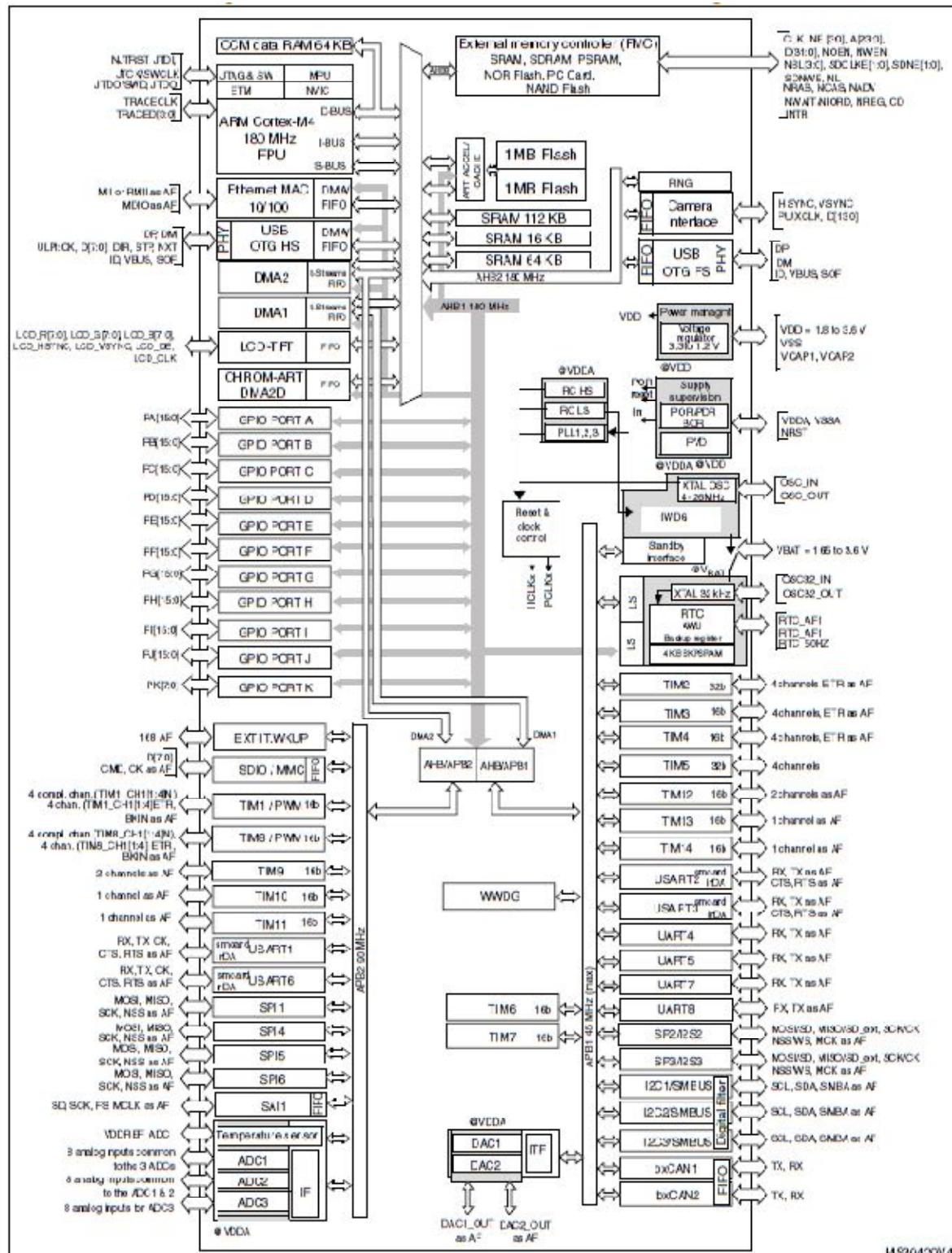
Hình 4.3 Ảnh minh họa các chân I/O của vi điều khiển STM32F429IIT6 được sử dụng

- STM32F429IIT6 là dòng vi điều khiển ARM Cotex M4 32 bit do STMicroelectronics sản xuất. Với nhiều tính năng nổi trội được tích hợp như:
 - Hiệu suất MCU cao được thực thi từ bộ nhớ Flash nhúng 2MB, SRAM256Kb, tích hợp core Cortex M4 cùng với DSP và FPU xung clock hoạt động lên đến 180Mhz / 225DMIPS.
 - Giao tiếp SDRAM tối đa 512MB với data lên đến 32 bit.
 - Nhúng LCD-TFT Controller hỗ trợ độ phân giải cao lên đến SVGA (800X600).
 - Nhúng phần cứng Chrom-ART Accelerator™ (DMA2D) giúp gia tốc xử lý dữ liệu đồ họa gấp 2 lần so với CPU đơn và giảm tải dữ liệu đồ họa cho CPU như Raw data copy, Pixel format conversion, Image blending.
 - Hiệu suất sử dụng năng lượng cao.
- ⇒ Với những tính năng nổi trội nêu trên STM32F429IIT6 là dòng vi điều khiển phù hợp cho việc phát triển các ứng dụng nhúng trong nhiều lĩnh vực công nghiệp, y tế, an ninh nói chung và ứng dụng xem ảnh kỹ thuật số nói riêng.

4.2.1 Giới thiệu về vi điều khiển STM32F429IIT6

- Chức năng:
 - Core: ARM 32 bit Cortex-M4 CPU với FPU, ART Accelerator cho phép thực thi chương trình từ bộ nhớ Flash với độ trễ bằng 0, tần số hoạt động lên đến 180MHz
 - Bộ nhớ:
 - Bộ nhớ Flash có dung lượng 2MB được tổ chức thành 2 bank cho phép vừa đọc vừa ghi dữ liệu
 - SRAM 256+4 KB bao gồm 64 KB CCM (Core coupled memory)
 - Flexible external memory controller lên đến 32 bit data bus: SRAM, PSRAM, SDRAM/LPSDR SDRAM, Compact Flash/NOR/NAND memories.
 - LCD parallel interface: chế độ 8080/6800.
 - LCD-TFT controller hỗ trợ độ phân giải lên đến SVGA (800x600) với phần cứng Chrom-ART Accelerator™ (DMA2D) giúp nâng cao chất lượng đồ họa.
 - 3x12 bit, 2.4 MPS ADC: 24 kênh
 - 2 bộ chuyển đổi D/A 12 bit
 - 16 –stream DMA controller với hỗ trợ FIFO và burst.

- 17 bộ timer: 12 bộ 16 bit và 2 bộ 32 bit hoạt động lên đến 180MHz với 4 ngõ vào IC/ OC/ PWM /pulse counter.
- Có khả năng ngắt tới 168 I/O port.
- 21 communication interface: 3xI²C + 4 USART / UART + 6 SPI + 1 SAI + 2 CAN +SDIO.
- Audio PLL+Serial Audio I/F
- Chế độ Debug:
 - SWD và JTAG
 - Cortex-M4 Trace MacrocellTM.
- Hash: Hỗ trợ SHA-2 và GCM.
- Nguồn cấp : 1.71 - 3.6 V
- Kiểu chân :LQFP 176 pin



1. The timers connected to APB2 are clocked from TIM_xCLK up to 180 MHz, while the timers connected to APB1 are clocked from TIM_xCLK either up to 90 MHz or 180 MHz depending on TIMPRE bit configuration in the RCC_DCKCFGR register.
2. The LCD-TFT is available only on STM32F429xx devices.

Hình 4.4 Sơ đồ khái STM32F429xx

- Các chuẩn kết nối nâng cao:

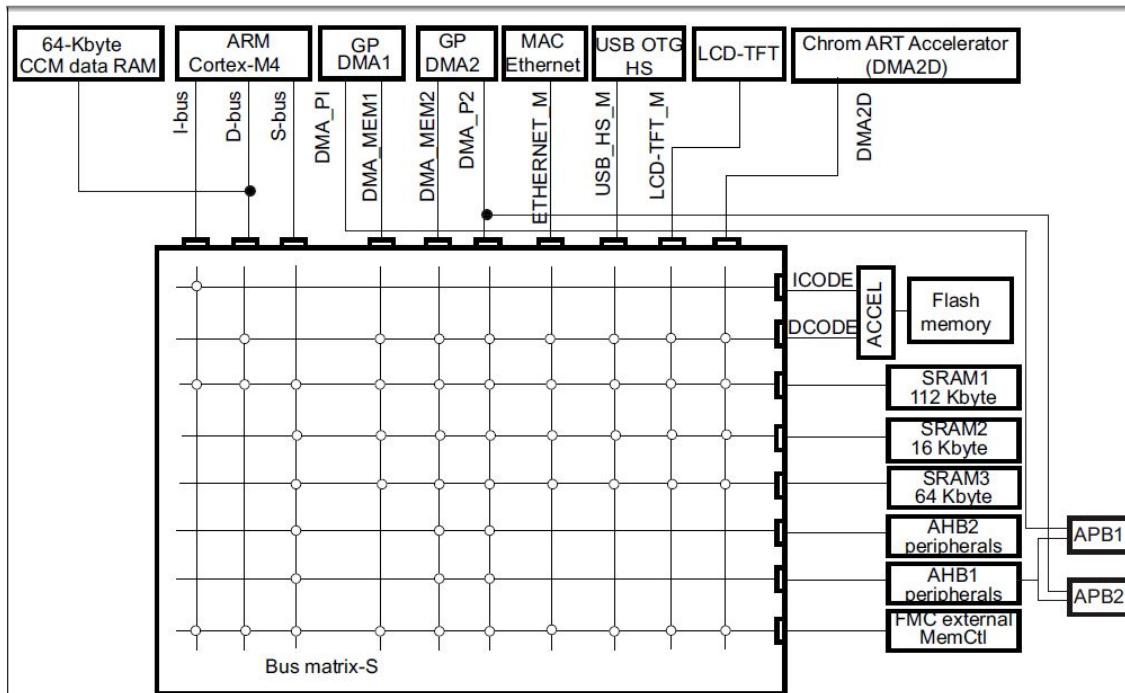
Ngoại vi	Performance
USB FS/HS	12 MBit/s / 480 MBit/s
USART	Lên đến 11.25 MBit/s
SPI	Lên đến 45 MBit/s
I ² C	1 Mbit/s
GPIO toggling	Lên đến 90MHz
3 phase MC timer	180 MHz PWM timer clock input
SDIO	Lên đến 48 MHz
I ² S và SAI	Tần số lấy mẫu từ 8 kHz đến 192 kHz
Camera interface	Lên đến 54 MByte/s tại tần số 54 MHz (giao tiếp từ 8-14 bit) song song
Crypto/ hash processor	AES-256 lên đến 149.33 MBytes
FMC	Lên đến 90 MHz (Data bus 8/16/32 , hỗ trợ SRAM, PSRAM, NAND and NOR Flash, SDRAM, Parallel Graphic LCD)
12 bit ADC / 12 bit DAC	0.41 us (2.4 MSPS, 7.2MSPS trong chế độ Interleaved) / 1 MSPS dual DAC
CAN 2.0B	Lên đến 2 CAN độc lập
Ethernet	10/100 MBit/s MAC với phần cứng IEEE 1588
LCD TFT Controller	Độ phân giải: QVGA, QWVGA, VGA, SVGA với phôi và trộn màu 2 layer

4.2.2 Tổng quan chức năng của STM32F429IIT6

- ARM Cortex-M4 với FPU, Flash nhúng và SRAM.
 - ARM Cortex-M4 với bộ xử lý FPU là thế hệ mới nhất của bộ vi xử lý ARM cho các hệ thống nhúng. Nó được phát triển để cung cấp 1 nền tảng chi phí thấp nhưng đáp ứng được nhu cầu sử dụng, công suất tiêu thụ thấp trong khi hiệu năng tính toán nhanh và khả năng đáp ứng interrupt nhanh.
 - ARM Cortex – M4 với lõi FPU là bộ vi xử lý RISC 32 bit có tính năng nổi bật về code-efficiency, cung cấp hiệu suất cao thường được kết hợp với các thiết bị 8/16 bit.
 - Bộ xử lý hỗ trợ các tập lệnh DSP cho phép quá trình xử lý tín hiệu hiệu quả và thực thi các giải thuật phức tạp.
 - Bộ FPU tăng tốc phát triển phần mềm bằng cách dùng các công cụ phát triển siêu ngôn ngữ.
- Adaptive real-time memory accelerator (ART AcceleratorTM)
 - ART AcceleratorTM là bộ tăng tốc bộ nhớ được tối ưu cho chuẩn ARM Cortex-M4 với bộ xử lý FPU. Nó cân bằng hiệu suất vốn là lợi thế của ARM Cortex -M4

với bộ FPU trên công nghệ bộ nhớ Flash thường đòi hỏi bộ xử lý chờ bộ nhớ flash ở tần số cao.

- Bộ nhớ Embedded Flash
 - Thiết bị được nhúng bộ nhớ Flash lên đến 2Mbytes có thể lưu cả chương trình và dữ liệu
- Embedded SRAM
 - Lên đến 256 Kbyte SRAM bao gồm 64Kbyte CCM (core coupled memory) data RAM.
 - Bộ nhớ RAM được truy cập tại tốc độ clock CPU với trạng thái chờ là 0.
 - 4 Kbyte backup SRAM. Đây là chỉ được truy cập bởi CPU .
- Multi-AHB bus matrix
 - Đây là ma trận đa bus AHB liên kết tất cả các master (CPU, DMAs, Ethernet, USB HS, LCD-TFT và DMA2D) với slave (Flash memory, RAM, FMC, AHB và APB) và đảm bảo chúng hoạt động liên tục và hiệu quả ngay cả khi một số thiết bị ngoại vi tốc độ cao làm việc cùng một lúc.



Hình 4.5 Sơ đồ khái niệm Multi AHB matrix

- DMA Controller

- Thiết bị có khả năng tạo ra dual port DMA(DMA1 và DMA2) với 8 stream mỗi loại. Chúng có khả năng quản lý truyền dữ liệu từ bộ nhớ -> bộ nhớ, từ ngoài vi -> bộ nhớ, từ bộ nhớ -> ngoại vi.
- DMA có thể được dùng cho các ngoại vi sau:
 - o SPI và I²S
 - o I²C
 - o USART
 - o TIM
 - o DAC
 - o SDIO
 - o Camera interface (DCMI)
 - o ADC
 - o SAI1
- Flexible memory controller (FMC)
 - FMC có 4 ngõ ra chọn chip hỗ trợ các chế độ: PCCard/Compact Flash, SDRAM/PLSDR SDRAM, SRAM, PSRAM, NOR Flash và NAND Flash.
 - FMC có 1 vài chức năng như:
 - o Hỗ trợ data bus 8/16/32 bit
 - o Đọc FIFO cho SDRAM controller
 - o Ghi FIFO

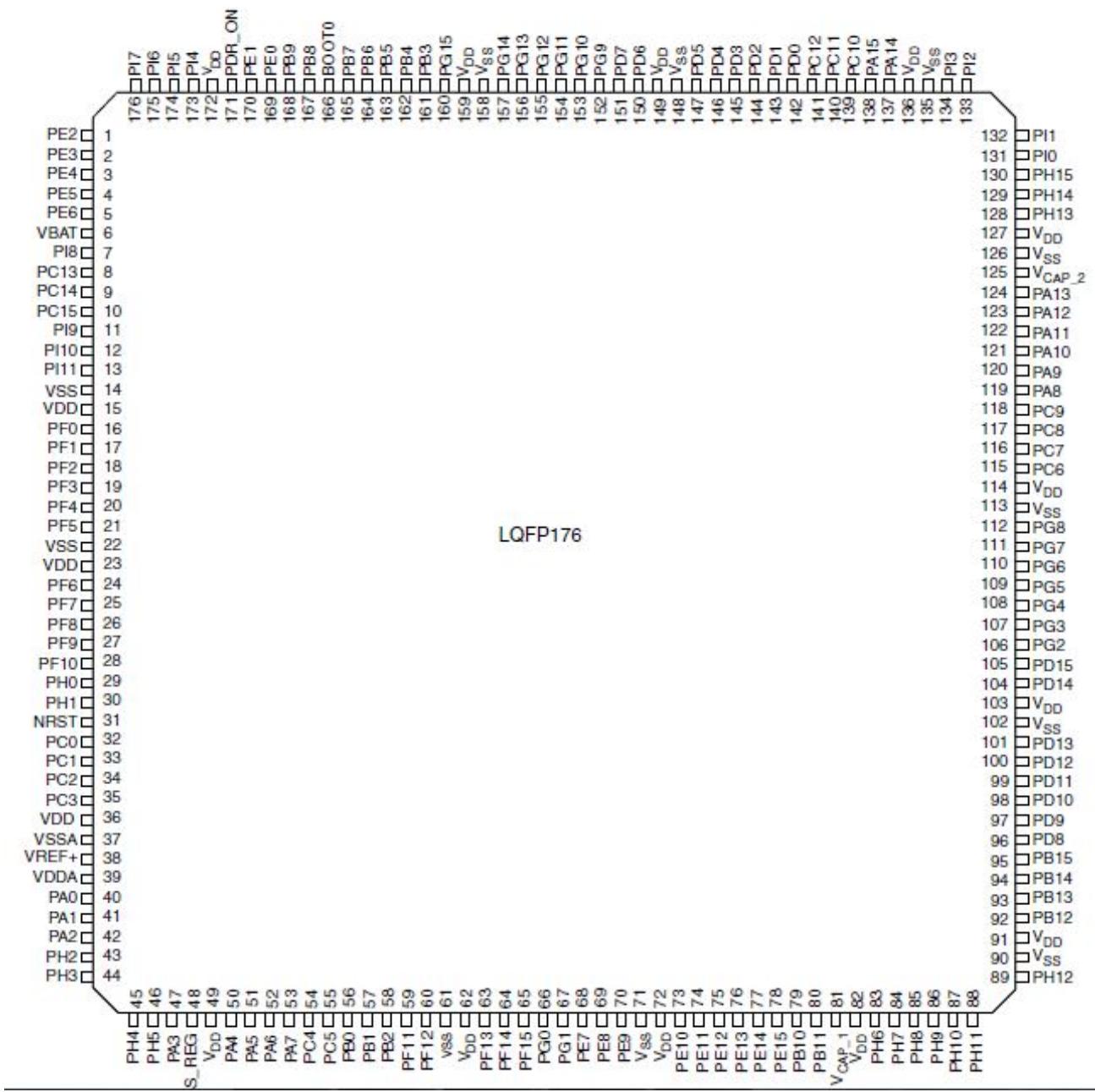
LCD parallel interface

- FMC có thể được cấu hình phù hợp để giao tiếp với hầu hết các LCD controller . Nó hỗ trợ chét độ 8080 của Intel và 6800 của Motorola, và có khả năng tùy biến cao để tương thích với các chuẩn giao tiếp LCD cụ thể. Chuẩn giao tiếp song song giúp ta có thể tạo nên các ứng dụng đồ họa với chi phí thấp mà hiệu quả bằng cách dùng các modun LCD có nhúng controller
- LCD-TFT Controller
 - LCD-TFT Controller hỗ trợ giao tiếp song song với 24 bit màu RGB và tất cả các tín hiệu được giao tiếp trực tiếp đến LCD-TFT panel với độ phân giải lên đến XGA (1024x768).
- Chrom-ART AcceleratorTM (DMA2D)
 - Chrom-ART AcceleratorTM (DMA2D) là bộ tăng tốc xử lý đồ họa giúp giảm tải cho CPU khi xử lý các dữ liệu đồ họa

- Chrom-ART kết hợp cả DMA2D và các chức năng định hướng đồ họa để phối, trộn ảnh và chuyển đổi định dạng pixel.
- Nested vectored interrupt controller (NVIC)
 - Chip được nhúng 1 trình điều khiển các vector ngắn giúp quản lý 16 mức độ ưu tiên và xử lý 91 kênh ngắn mở rộng.
 - Khối phần cứng cung cấp chức năng quản lý ngắn linh hoạt với độ trễ ngắn là tối thiểu.
- External interrupt/event controller (EXTI)
 - Trình điều khiển ngắn ngoài/sự kiện bao gồm 23 edge-detector line được dùng để tạo ra các ngắn ngoài/sự kiện. Mỗi line có thể được cấu hình độc lập để lựa chọn sự kiện kích hoạt ngắn (ngắn cạnh lên, ngắn cạnh xuống hoặc cả 2)
 - 168 GPIO được kết nối đến 16 đường ngắn ngoài.
- Inter-integrated sound (I²S):
 - Chuẩn giao tiếp I²S được hình thành bằng cách dồn kênh SPI2 và SPI. Chúng có thể hoạt động trong chế độ master hoặc slave, chế độ giao tiếp đơn hoặc song công và có thể được cấu hình hoạt động với độ phân giải 16/32 bit. Tần số lấy mẫu âm thanh hỗ trợ từ 8 kHz đến 192 kHz.
- Secure digital input/output interface (SDIO)
 - SDIO hỗ trợ giao tiếp SD/SDIO/MMC chế độ 1/4/8 bit
 - Chế độ này cho phép truyền data với tốc độ lên tới 48 Mhz
- Chết độ Boot:
 - Tại thời điểm bắt đầu khởi động, chân boot được dùng để chọn một trong 3 chế độ sau:
 - o Boot từ user Flash
 - o Boot từ system memory
 - o Boot từ embedded SRAM
- Bootloader nằm trong bộ nhớ hệ thống. Nó có thể được dùng để tái lập trình Flash memory thông qua serial interface.
- Real-time clock (RTC), backup SRAM và backup registers
 - Back up domain bao gồm:
 - o Real-time clock (RTC).
 - o 4 Kbytes backup SRAM.
 - o 20 thanh ghi backup

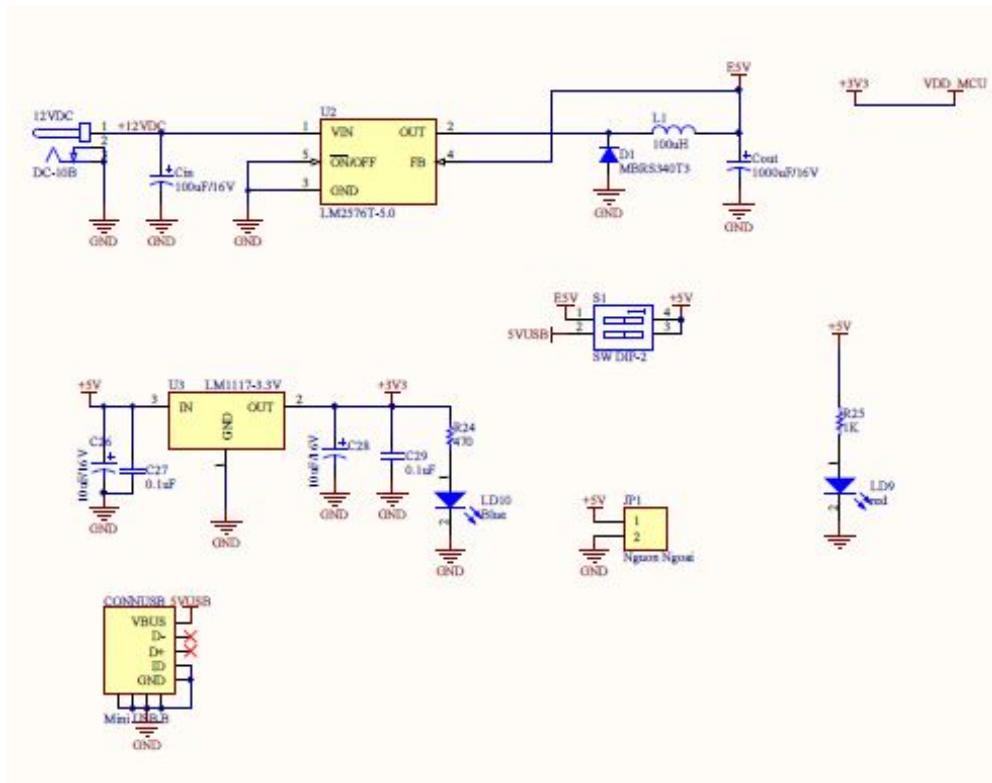
- Đồng hồ thời gian thực (RTC) là một bộ đếm BCD độc lập. Các thanh ghi bao gồm second, minute, hour (chế độ 12/24), week day, date, month, year lưu giá trị định dạng số BCD. Các ngày 28,29 (năm thiếu), 30, 31 được tự động hiệu chỉnh. RTC cung cấp khả năng lập trình báo động và ngắt theo chu kỳ
- RTC định thời gian bằng thạch anh ngoài 32.768 Khz
- Hai thanh ghi alarm được dùng để tạo ra 1 báo động tại thời gian cụ thể.
- Một bộ chia tần 20 bit được dùng như một đồng hồ nền. Nó được cấu hình mặc định để tạo ra thời gian nên 1s tại tần số 32.768 Khz.
- 4 Kbytes backup SRAM giống như một vùng bộ nhớ EEPROM. Nó được dùng để lưu dữ liệu cần được giữ lại trong chế độ VBAT và standby. Vùng bộ nhớ này mặc định bị tắt để tối thiểu hóa năng lượng tiêu thụ. Nó có thể được kích hoạt bằng phần mềm.

- Chi tiết các chân vi điều khiển



Hình 4.6 Chi tiết chân vi điều khiển

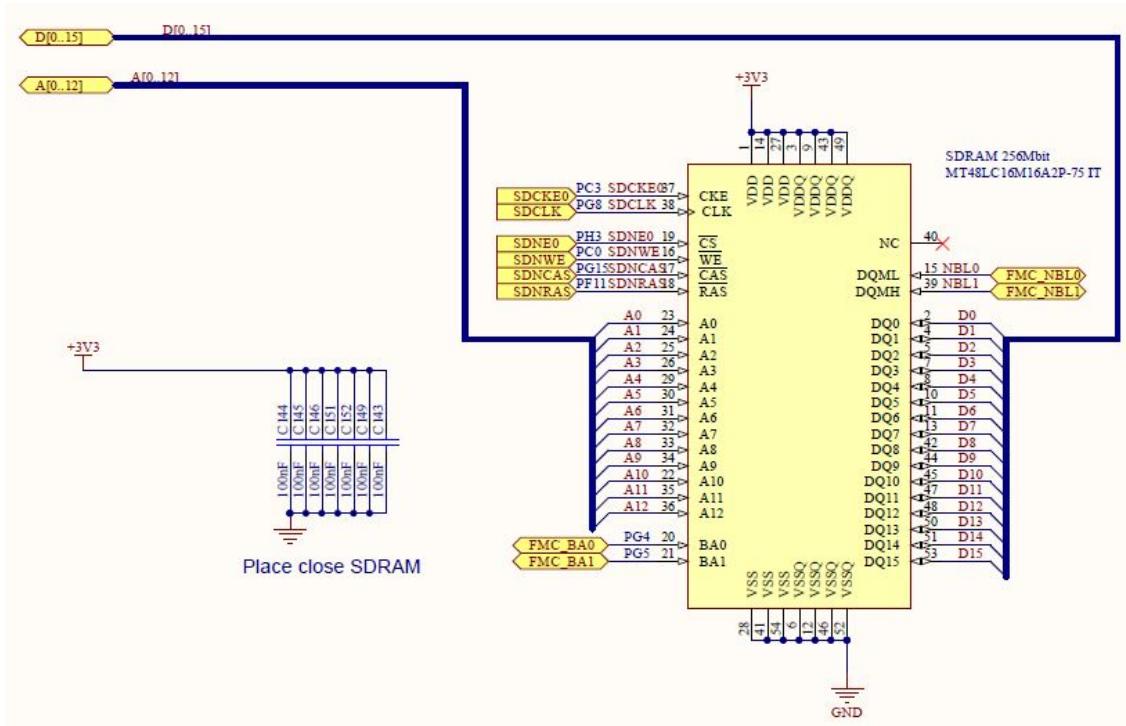
4.3 Khối nguồn



Hình 4.7 Sơ đồ nguyên lý khói nguồn

- Khối nguồn dùng để cấp nguồn 3.3V cho vi điều khiển và 5V cho LCD.
- Ứng dụng sử dụng 2 nguồn chính:
 - Lấy nguồn 5V trực tiếp từ USB.
 - Lấy nguồn 12V convert xuống 5V và 3.3V.
- Một số linh kiện chính dùng trong khối nguồn:
 - LM2576: IC ổn áp 5V đầu ra từ 12V đầu vào.
 - LM1117: IC ổn áp 3.3V đầu ra từ 5V đầu vào.
 - MBRS340T3: Diode Schottky Rectifier.
 - Các đèn led báo nguồn 5V, 3.3V.
 - DC_IN: Socket lấy nguồn 12V.
 - Công tắc chọn nguồn USB hay nguồn ngoài.

4.4 Khối SDRAM



Hình 4.8 Sơ đồ nguyên lý SDRAM

- SDRAM sử dụng là MT48LC16M16A2P-75 IT loại 4 Meg x 16 x 4 banks.
- Dung lượng bộ nhớ: 256 Mbit.
- Loại footprint: 54 pin TSOP II OCPL.

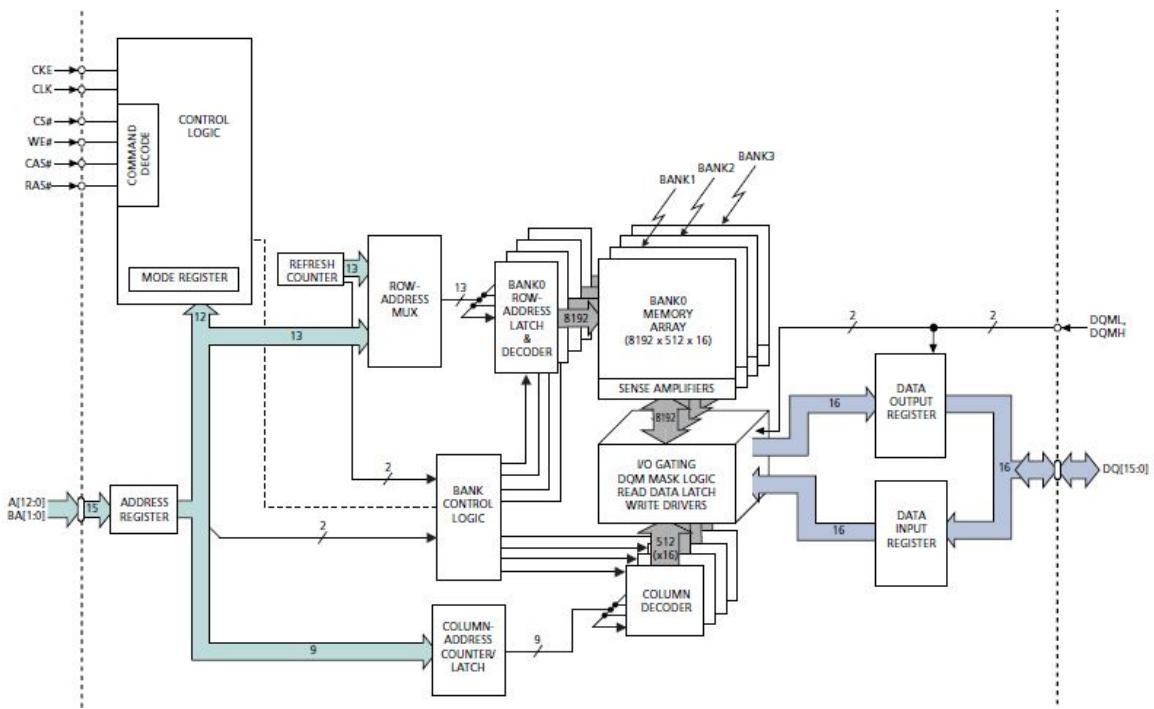


Hình 4.9 minh họa SDRAM MT48LC16M16A2P-75 IT

- Bảng chi tiết chân SDRAM:

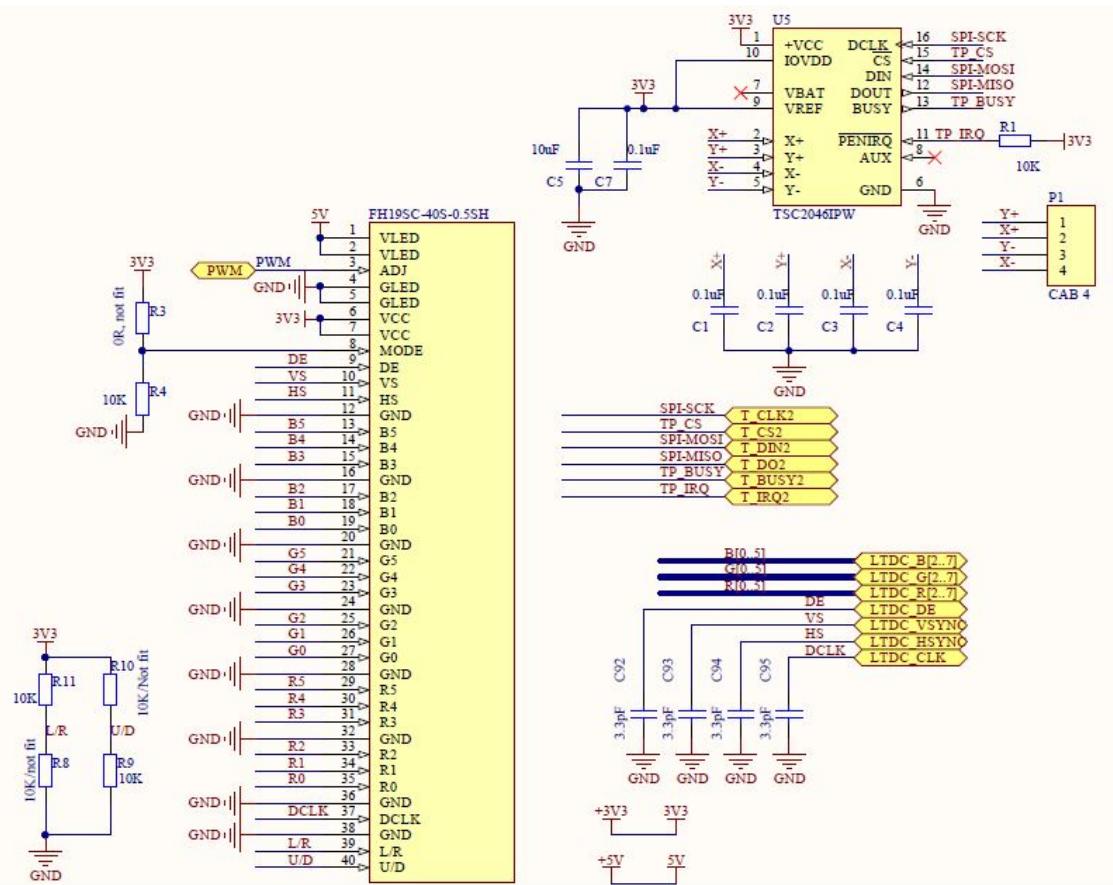
Tên	Loại	Mô tả
CLK	Input	Clock: CLK được lái bởi clock hệ thống. Tất cả các tín hiệu đầu vào của SDRAM được lấy mẫu theo cạnh tích cực của CLK. CLK cũng tăng bộ đếm nội và điều khiển các thanh ghi ngoại
CKE	Input	Clock enable: CKE tích cực mức cao và tích cực mức thấp tín hiệu CLK. Clock tích cực mức thấp cung cấp hoạt động precharge power-down và SELF REFRESH, kích hoạt hoạt động power-down hoặc CLOCK SUSPEND. CKE được đồng bộ ngoại trừ sau khi thiết bị vào chế độ power-down và self refresh, nơi CKE sẽ bắt đồng bộ cho tới sau khi tồn tại cùng 1 chế độ. Bộ đệm ngõ vào bao gồm CLK được tắt trong suốt chế độ power-down và self-refresh.
CS#	Input	Chip select: bộ giải mã lệnh CS# enable ở mức thấp và disable ở mức cao. Tất cả các lệnh được mask khi CS# tích cực mức cao, nhưng READ/WRITE sẵn sang tiếp tục và sự hoạt động của DQM sẽ giữ lại khả năng mặt nạ DQ trong khi CS# tích cực cao. CS# cung cấp bank ngoài sự lựa chọn trên hệ thống đa bank.
CAS#, RAS#, WE#	Input	Lệnh ngõ vào: RAS#, CAS#, và WE# định nghĩa lệnh được vào
DQML, DQML	Input	Input/ Output mask: DQM được lấy mẫu mức cao, là mặt nạ tín hiệu vào cho truy cập ghi và tín hiệu cho phép ngõ ra truy cập đọc. Dữ liệu ngõ vào được mask trong suốt trong chu kỳ WRITE. Bộ đệm ngõ ra ở mức cao trong suốt chu kỳ READ.
BA[1:0]	Input	Địa chỉ bank ngõ vào: BA[1:0] định nghĩa bank nào được áp dụng các lệnh ACTIVE, READ, WRITE hay PRECHARGE
A[12:0]	Input	Địa chỉ ngõ vào: A[12:0] được lấy mẫu khi có lệnh ACTIVE , lệnh READ hay WRITE để lựa chọn vị trí dãy bộ nhớ trong từng bank. Địa chỉ ngõ vào cũng cung cấp op-code khi có lệnh LOAD MODE REGISTER
DQ[15:0]	I/O	Data input/output: Ngõ vào/ngõ ra dữ liệu
V _{DDQ}	Supply	Nguồn cấp DQ để triệt nhiễu
V _{SSQ}	Supply	Chặn nồi GND DQ để triệt nhiễu
V _{DD}	Supply	Nguồn cấp cho SDRAM 3.3V
V _{SS}	Supply	Chân GND của SDRAM

- Sơ đồ khối SDRAM:



Hình 4.10 Sơ đồ khối SDRAM MT48LC16M16A2P

4.5 Khối LCD-TFT và resistive touch panel



Hình 4.11 Sơ đồ nguyên lý khối LCD-TFT và resistive touch panel

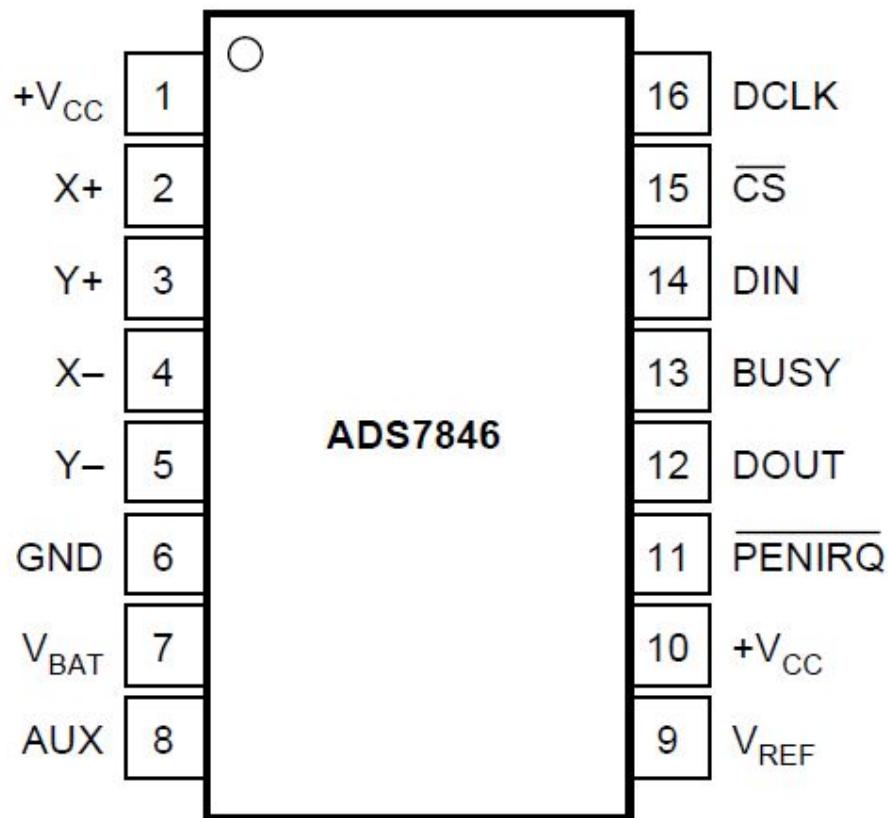
- Khối LCD-TFT:

- Kích thước LCD: 7 inch
- Độ phân giải: 800x480
- Chi tiết chân LCD:

Chân số	Tên	Chức năng
1	V _{LED}	Nguồn 5V cho LED Driver
2	V _{LED}	Nguồn 5V cho LED Driver
3	ADJ	Chỉnh độ tương phản màn hình
4	G _{LED}	Chân GND cho mạch LED
5	G _{LED}	Chân GND cho mạch LED
6	V _{CC}	Nguồn cấp 3.3V cho LCD
7	V _{CC}	Nguồn cấp 3.3V cho LCD
8	MODE	Chọn chế độ HV hay DE
9	DE	Data enable
10	VS	Ngõ vào tín hiệu VSYNC
11	HS	Ngõ vào tín hiệu HSYNC
12	GND	Ground

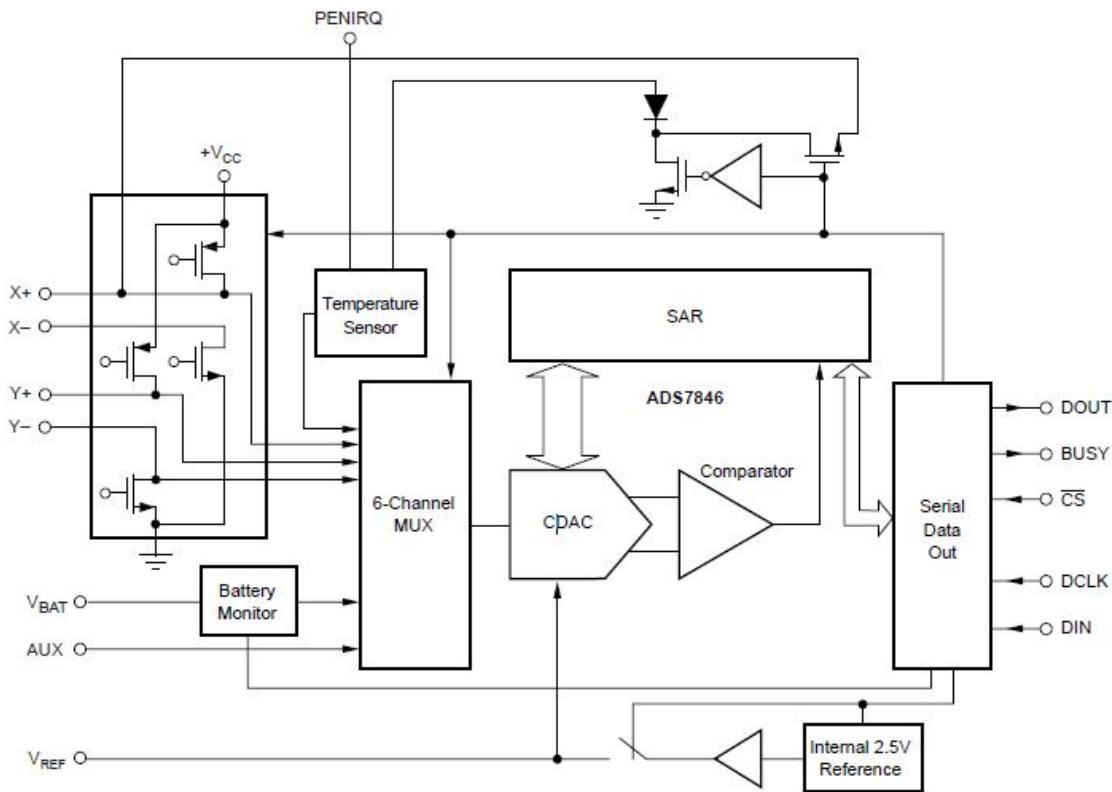
13	B5	Blue data input (MSB)
14	B4	Blue data input
15	B3	Blue data input
16	GND	Ground
17	B2	Blue data input
18	B1	Blue data input
19	B0	Blue data input
20	GND	Ground
21	G5	Green data input
22	G4	Green data input
23	G3	Green data input
24	GND	Ground
25	G2	Green data input
26	G1	Green data input
27	G0	Green data input
28	GND	Ground
29	R5	Red data input
30	R4	Red data input
31	R3	Red data input
32	GND	Ground
33	R2	Red data input
34	R1	Red data input
35	R0	Red data input
36	GND	Ground
37	DCLK	Sample clock
38	GND	Ground
39	L/R	Chọn quét màn hình trái hoặc phải
40	U/D	Chọn quét màn hình Up hoặc Down

- Khối resistive touch panel:
 - Touch Screen Controller sử dụng là chip ADS7846:



Hình 4.12 Sơ đồ chân chip ADS7846

- Sơ đồ khối chip ADS7846:



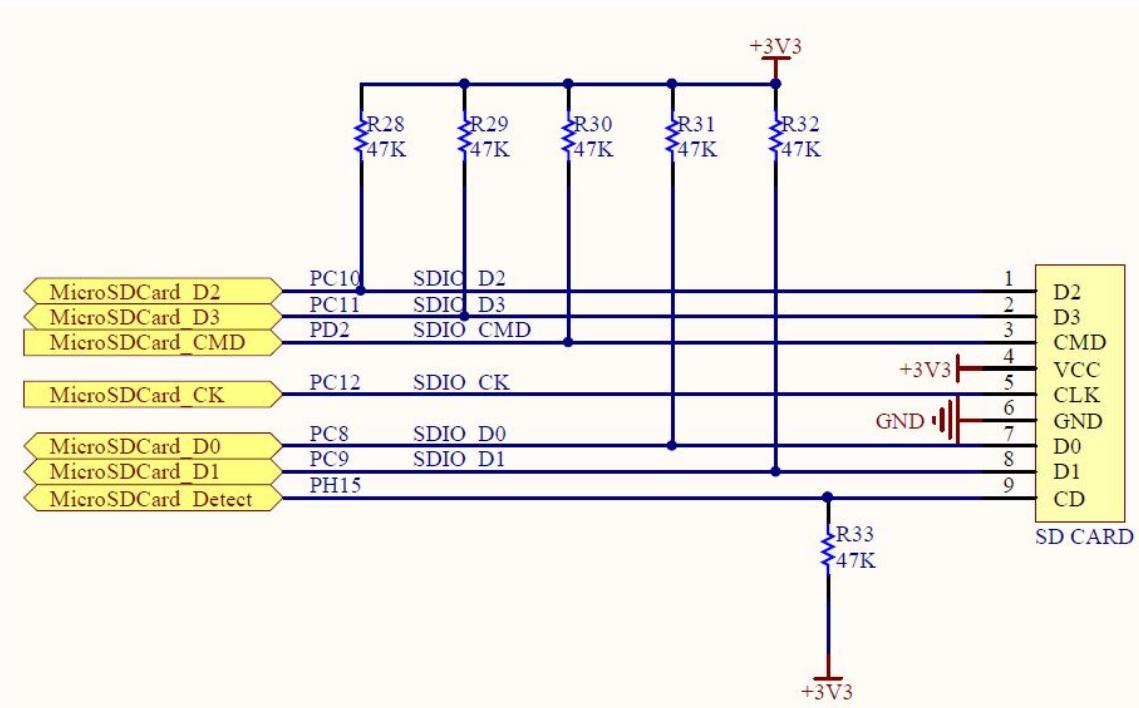
Hình 4.13 Sơ đồ khối chip ADS7846

- Chi tiết chân chip ADS7846:

Chân số	Tên	Mô tả
1	+V _{CC}	Nguồn cấp cho IC hoạt động
2	X+	Ngõ vào vị trí X+
3	Y+	Ngõ vào vị trí Y+
4	X-	Ngõ vào vị trí X-
5	Y-	Ngõ vào vị trí Y-
6	GND	Ground
7	V _{BAT}	Ngõ vào giám sát pin
8	AUX	Ngõ vào hỗ trợ ADC
9	V _{REF}	Ngõ ra/ ngõ vào điện áp tham chiếu
10	+V _{CC}	Nguồn cấp cho các chân I/O số
11	PENIRQ	Pen Interrupt
12	DOUT	Ngõ ra dữ liệu nối tiếp. Dữ liệu được dịch khi gấp DCLK cạnh xuống. Ngõ ra ở trạng thái trở kháng cao khi CS ở mức cao
13	BUSY	Busy output. Ngõ ra này ở trạng thái trở kháng cao khi CS ở mức cao.
14	DIN	Ngõ vào dữ liệu nối tiếp. Nếu CS xuống mức thấp thì dữ liệu được chốt khi gấp cạnh lên của DCLK.

15	CS	Ngõ vào chọn chip.
16	DCLK	Xung nhịp

4.6 Khối thẻ nhớ SDCard

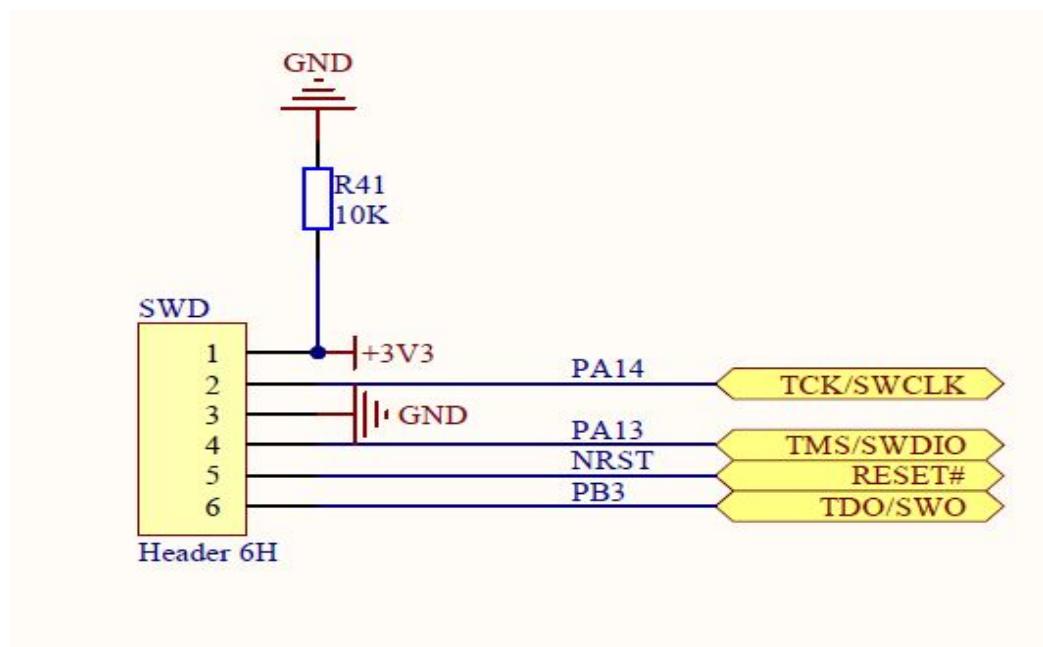


Hình 4.14 Sơ đồ nguyên lý khói SDCard

- Giao tiếp với thẻ nhớ qua chuẩn giao tiếp SDIO 4 bit
- D0: SD Serial Data 0
- D1: SD Serial Data 1
- D2: SD Serial Data 2
- D3: SD Serial Data 3
- CMD: Command, Response
- CLK: Serial Clock

4.7 Khối nạp chương SWD (Serial Wire Debug)

Khối SWD được dùng để nạp chương trình vào chip



Hình 4.15 Sơ đồ nguyên lý khói SWD

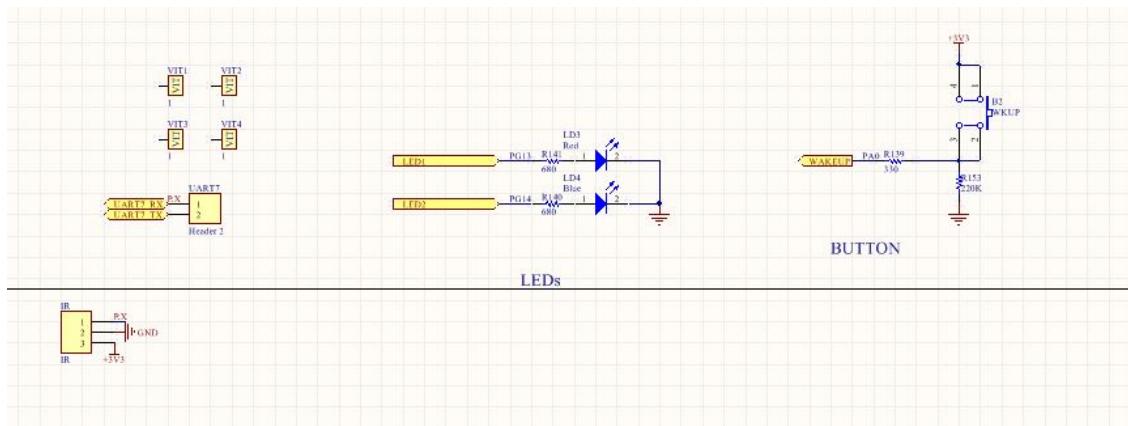
SWCLK: Serial Wire Debug Clock

SWDIO: Serial Wire Debug Data Input/Output

RESET: Chân RESET MCU

SWO: Chân reserved

4.8 Sơ đồ nguyên lý khói ngoại vi I/O



Hình 4.16 Sơ đồ nguyên lý khôi ngoại vi I/O

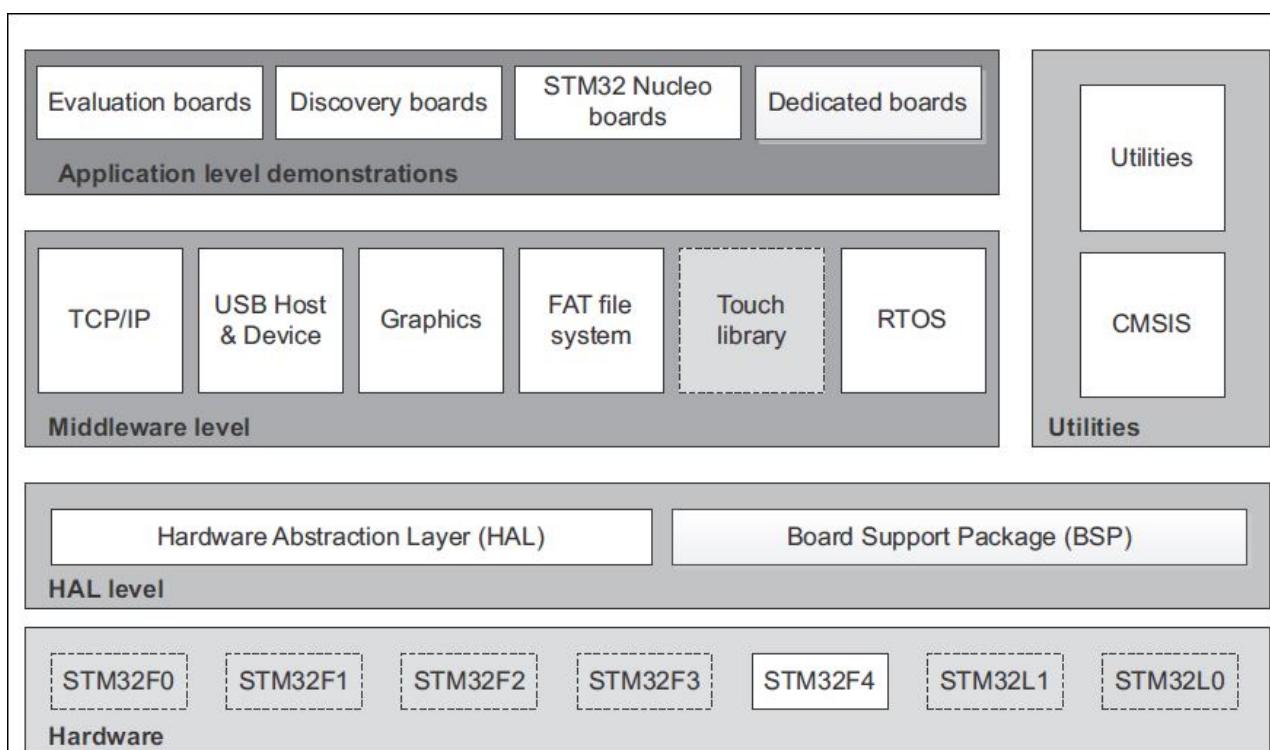
- Khối này gồm một linh kiện như:
 - Led xanh dương, đỏ.
 - Nút nhấn.

5. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

- Yêu cầu đặt ra cho phần mềm:
 - o Phần mềm hoạt động ổn định, chính xác.
 - o Đọc được hình ảnh trong thẻ nhớ và xuất ra màn hình LCD-TFT.
 - o Màu sắc hình ảnh trung thực.

5.1 Cấu trúc phần mềm

- Phần mềm ứng dụng được phát triển theo firmware STM32CubeF4 mà ST cung cấp.



Hình 5.1 Cấu trúc Firmware STM32CubeF4

- STM32CubeF4 là một gói các component phần mềm nhúng dùng để phát triển các ứng dụng trên vi điều khiển STM32F4.
- Gói STM32CubeF4 bao gồm các component Middleware:
 - Full USB host và device hỗ trợ nhiều lớp:

- Host class: HID, MSC, CDC, Audio, MTP
 - Device class: HID, MSC, CDC, Audio, DFU
 - Graphic:
 - STemWin là giải pháp đồ họa chuyên nghiệp dựa trên emWin từ đối tác SEGGER của ST.
 - LibJPEG: một mã nguồn mở để thực hiện mã hóa và giải nén ảnh JPEG trên STM32.
 - CMSIS-RTOS với giải pháp mã nguồn mở FreeRTOS.
 - FAT File system dựa trên giải pháp mã nguồn mở FatFs.
 - TCP/IP stack dựa trên giải pháp mã nguồn mở LwIP.
 - SSL/TLS secure layer dựa trên mã nguồn mở Polar SSL.
- ⇒ Ứng dụng đê tài sử dụng các middleware chính: Graphic, FAT File system, CMSIS RTOS.

5.2 Các middleware chính dùng trong ứng dụng

Ứng dụng dùng 3 middleware chính là: FatFs File system để giao tiếp với MMC/SD Card nhằm truy cập tập tin thẻ nhớ, STemwin của ST tạo giao diện đồ họa trên LCD, hệ điều hành FreeRTOS để quản lí các tác vụ.

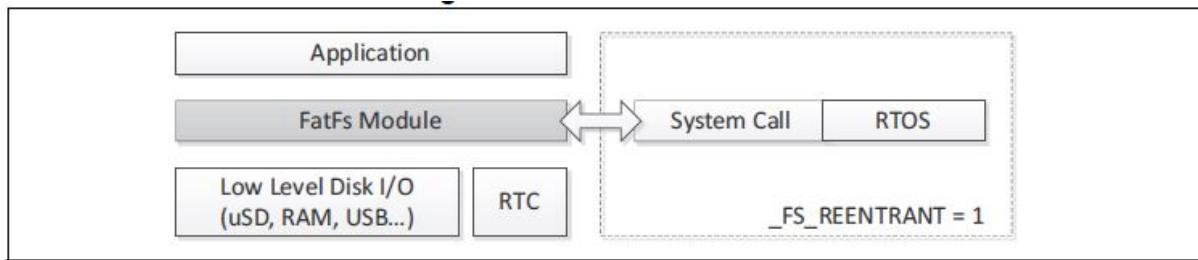
5.2.1 FatFs File System

FatFs là bộ mã nguồn mở miễn phí hỗ trợ định dạng FAT cho các hệ thống nhúng nhỏ. Module này được viết tương thích với ANSI C và hoàn toàn riêng rẽ với disk I/O layer nên nó độc lập với kiến trúc phần cứng.

- Chức năng của module FatFs:
 - Tương thích với hệ thống tập tin FAT của window.
 - Có footprint code và work area rất nhỏ.
 - Hỗ trợ trên tập tin dài với ANSI/OEM hay Unicode.
 - Có hỗ trợ RTOS.
 - Hỗ trợ đa kích thước sector.
 - Hỗ trợ các loại định dạng FAT như FAT12, FAT16, FAT32.
 - Số lượng tập tin mở không giới hạn chỉ phụ thuộc vào bộ nhớ.
 - Số ô đĩa lên tới 10.
 - Có platform độc lập và dễ port.

- Cấu trúc FatFs File System:

- Module FatFs là một middleware hỗ trợ nhiều hàm để truy cập đến FAT volume như : f_open(), f_close(), f_read(), f_write(), f_mount(), ...
- Một module low level disk I/O được dùng để read/write ổ đĩa vật lý.
- Một module RTC được dùng để lấy thời gian hiện tại.
- Module low level disk I/O và RTC hoàn toàn riêng rẽ với module FatFs



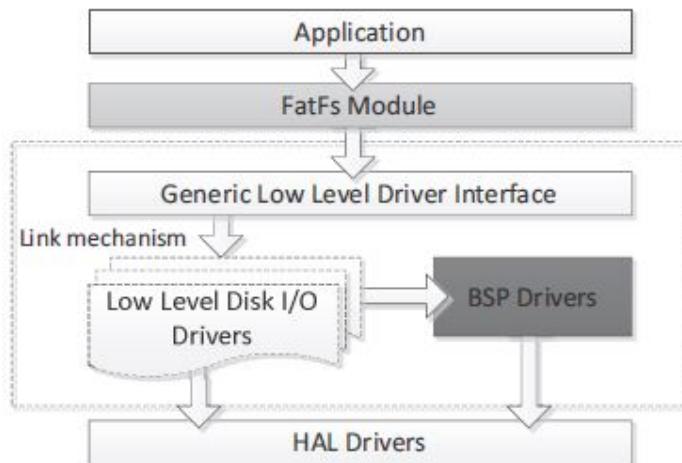
Hình 5.2 Cấu trúc module FatFs

- FatFs APIs:

- Các lớp FatFs API thực thi các API hệ thống file. Nó dùng disk I/O interface để giao tiếp với ổ đĩa vật lý phù hợp. Các API được chia thành 4 nhóm:
 - o Nhóm các API vận hành logical volume hoặc partition.
 - o Nhóm các API vận hành thư mục.
 - o Nhóm các API vận hành tập tin.
 - o Nhóm các API vận hành cả tập tin và thư mục.
- Một số hàm để FatFs có thể truy cập vào FAT volume.
 - o f_mount(): Đăng ký/ hủy đăng ký vùng làm việc.
 - o f_open(): Mở/ them tập tin.
 - o f_close(): Đóng tập tin.
 - o f_read(): Đọc tập tin.
 - o f_write(): Ghi vào một tập tin.
 - o f_lseek(): Di chuyển pointer đọc/ghi. Mở rộng kích thước tập tin.
 - o f_truncate(): Cắt kích thước 1 tập tin.
 - o f_sync(): Flush cached data.
 - o f_opendir(): Mở một thư mục.
 - o f_readdir(): Đọc một danh mục thư mục.
 - o f_getfree(): Get free clusters.

- f_stat(): Kiểm tra trạng thái của đối tượng.
 - f_mkdir(): Thêm một thư mục.
 - f_unlink(): Xóa một tập tin hoặc thư mục.
 - f_chmod(): Thay đổi một thuộc tính.
 - f_utime(): Change timestamp.
 - f_rename(): Đổi tên một tập tin hoặc thư mục.
 - f_chdir(): Thay đổi thư mục hiện tại.
 - f_chdrive(): Thay đổi ổ đĩa hiện tại.
 - f_getcwd(): Lấy thư mục hiện tại.
 - f_getlabel(): Lấy tên nhãn ổ đĩa.
 - f_setlabel(): Thiết lập tên nhãn ổ đĩa.
 - f_forward(): Forward file data to the stream directly.
 - f_mkfs(): Tạo một tập tin hệ thống trên ổ đĩa.
 - f_fdisk(): Devide a physical drive.
 - f_gets(): Đọc một chuỗi kí tự.
 - f_putc(): Ghi một chữ.
 - f_puts(): Ghi một chuỗi kí tự.
 - f_printf(): Ghi một chuỗi kí tự được định dạng.
 - f_tell(): Lấy pointer ghi/đọc hiện tại.
 - f_eof(): Kiểm tra vị trí cuối của tập tin.
 - f_size(): Lấy kích thước tập tin.
 - f_error(): Kiểm tra lỗi của một tập tin.
- Các low level API FatFs:
 - Bởi module FatFs hoàn toàn riêng rẽ giữa module disk I/O và RTC nên module FatFs yêu cầu một vài hàm mức thấp để vận hành ổ đĩa vật lý: read/write và lấy thời gian hiện tại. Bởi vì các function low level disk I/O và module RTC không là thành phần của module FatFs nên chúng cần được tạo bởi người dùng.
 - Giải pháp FatFs Middleware cung cấp driver low level disk I/O hỗ trợ một số ổ đĩa như RAMDisk , uSD, USBDisk.
 - Một số hàm I/O low level:
 - disk_initialize(): Khởi động ổ đĩa.
 - disk_status(): Trả về trạng thái ổ đĩa được chọn.
 - disk_read(): Đọc các sector từ ổ đĩa.
 - disk_write(): Ghi các sector đến ổ đĩa.

- disk_ioctl(): Controls device-specified features.
 - get_fattime(): Trả về thời gian hiện tại.
- ⇒ Những hàm này chỉ được gọi bởi các hàm FatFs file system như f_mount(), f_read(), f_write(),.. chứ không phải bởi chương trình ứng dụng.
- Tích hợp FatFs vào STM32CubeF4:
 - Để liên kết module FatFs với một low level disk driver , ta có thể dùng hàm FATFS_LinkDriver() và FATFS_UnLinkDriver() để thêm hoặc xóa một disk I/O driver. Để biết được số disk I/O driver được sử dụng ta có thể dùng hàm FATFS_GetAttachedDriverNbr().



Hình 5.3 Cấu trúc module Middleware FatFs

5.2.2 STemWin

Giới thiệu: STemWin dùng để thiết kế giao diện đồ họa độc lập với phần cứng cho các ứng dụng dùng graphic LCD.Nó cũng tương thích với môi trường đơn hay đa tác vụ cũng như là có hệ điều hành hay không có.



Hình 5.4 Hình minh họa STemwin

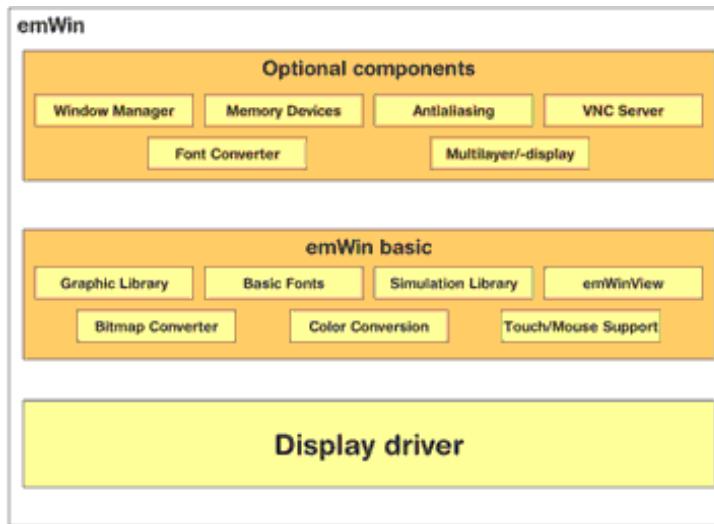
- **Chức năng:**
 - Chỉ với trình biên dịch ANSI C, ta có thể dùng với bất kì CPU nào 8/16/32 bit.
 - Thích hợp với bất kì LCD nào (monochrome, grayscale hay color) với bất kì controller nào (nếu driver có sẵn).
 - Có thể hoạt động mà không có LCD controller trên màn hình nhỏ.
 - Các hàm được thiết kế tối ưu cả kích thước lẫn tốc độ.
 - Có hỗ trợ touch screen, mouse và keypad,...
 - Mô phỏng giao diện đồ họa bằng Visual studio mà không cần phần cứng.
- **Cấu trúc của STemwin:**



Hình 5.5 Cấu trúc của STemwin

- STemWin bao gồm 2 driver được tối ưu:
 - Driver truy cập trực tiếp tuyền tính (LIN) cho STM32F429 TFT-LCD controller với kỹ thuật tăng tốc đồ họa DMA2D ChromART.
 - Driver truy cập gián tiếp (FlexColor) cho các LCD Controller có bus song song và nối tiếp.
- Để dùng thư viện STemWin, trong chương trình chính ta cần hiệu chỉnh 2 file quan trọng sau:
 - File cấu hình LCD:
 - o Khởi động và cấu hình LCD Display.
 - o Tùy chỉnh và liên kết driver LCD Display.
 - File cấu hình GUI:
 - o Lựa chọn các module: memory device, window manager.
 - o GUI memory và heap management.

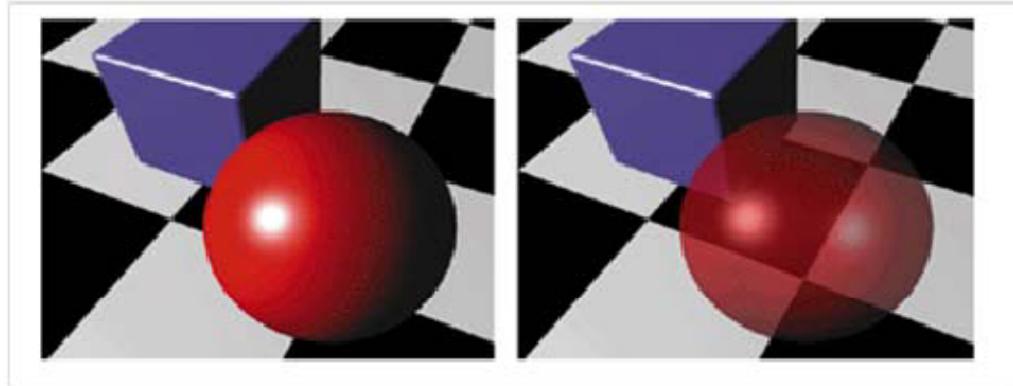
- Các thành phần của STemwin:



Hình 5.6 STemwin Component

- Cáu hình để dùng thư viện STemWin:
 - Cáu hình: Gồm 2 phần cấu hình GUI và cấu hình LCD
 - Cấu hình GUI: Để cấu hình bộ nhớ, màu, font chữ
 - Cáu hình GUI được cấu hình trong file GUIConf.c: Trong file này ta cần thực thi hàm GUI_X_Config() lần đầu tiên trong suốt quá trình khởi động. Nhiệm vụ chính của hàm này là thiết lập bộ nhớ có sẵn cho GUI và sau đó đăng ký nó đến hệ thống quản lý bộ nhớ động thông qua hàm GUI_ALLOC_AssignMemory(): Hàm này thông qua 1 pointer trả đến khôi bộ nhớ và kích thước bộ nhớ đến memory manager.
 - Cấu hình LCD: Cấu hình này phụ thuộc vào phần cứng và cho phép người dùng định nghĩa kích thước hiển thị, Driver Display, và các hàm color conversion được dùng.
 - LCD được cấu hình trong file LCDConf.c. Trong file này hàm chính được dùng là LCD_X_Config(), được gọi ngay khi hàm GUI_X_Config() được thực thi. Hàm LCD_X_Config() cho phép thêm và cấu hình display driver cho mỗi layer bằng cách gọi các hàm sau:
 - GUI_DEVICE_CreateAndLink(): Thêm driver device và liên kết nó đến thiết bị.
 - LCD_SetSizeEx() và LCD_SetVSizeEx(): để thiết lập cấu hình display size.

- Một số function trong STemwin:
 - Các hàm chính của thư viện STemWin:
 - o Hiển thị file hình ảnh: STemWin hỗ trợ hiển thị các file hình ảnh định dạng BMP, JPEG, PNG, GIF.
 - o Alpha blending: đây là phương pháp kết hợp alpha channel với layer khác để tạo hiệu ứng transparency.

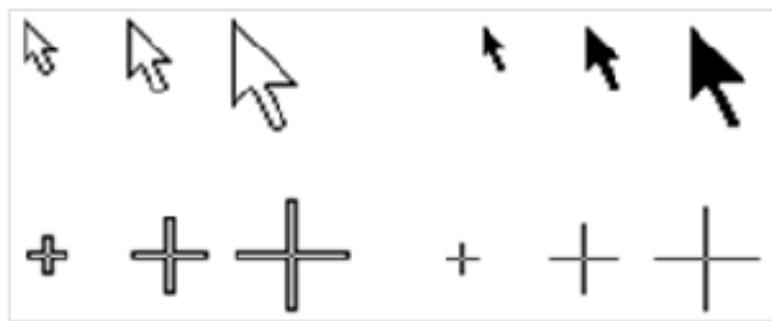


Hình 5.7 Hiệu ứng alpha blending

- Sprites và cursors:
 - o Một Sprite là một hình ảnh có thể hiển thị trên tất cả các hình khác trên màn hình.

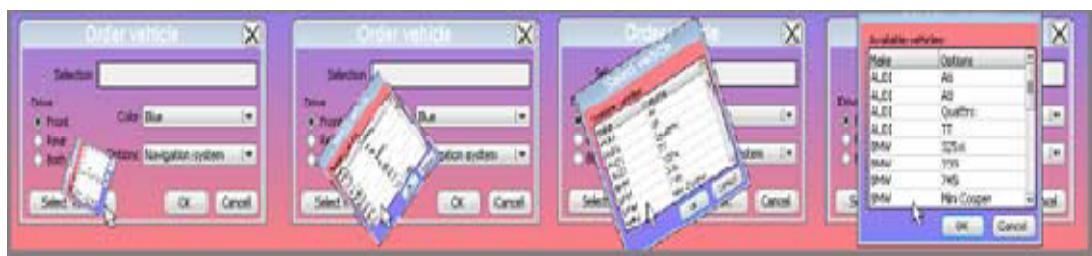


Hình 5.8 minh họa một Sprite



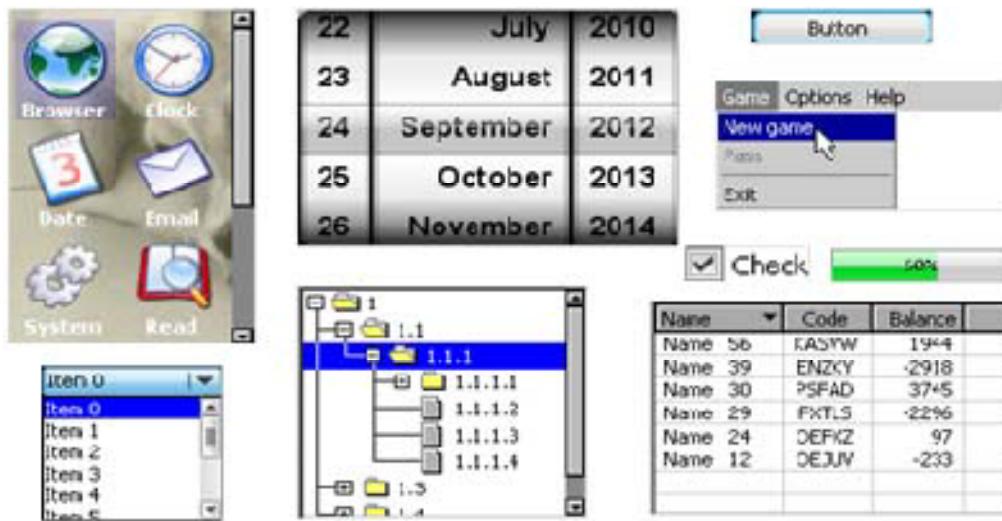
Hình 5.9 Minh họa một Cursor

- Bộ nhớ thiết bị:Bộ nhớ thiết bị có thể được dùng:
 - o Tránh hiệu ứng flickering (bởi hiện tượng vẽ trực tiếp lên màn hình).
 - o Dùng cho các hoạt động xoay hay scale đối tượng.
 - o Cho hoạt động fading.
 - o Tạo hiệu ứng transparency.



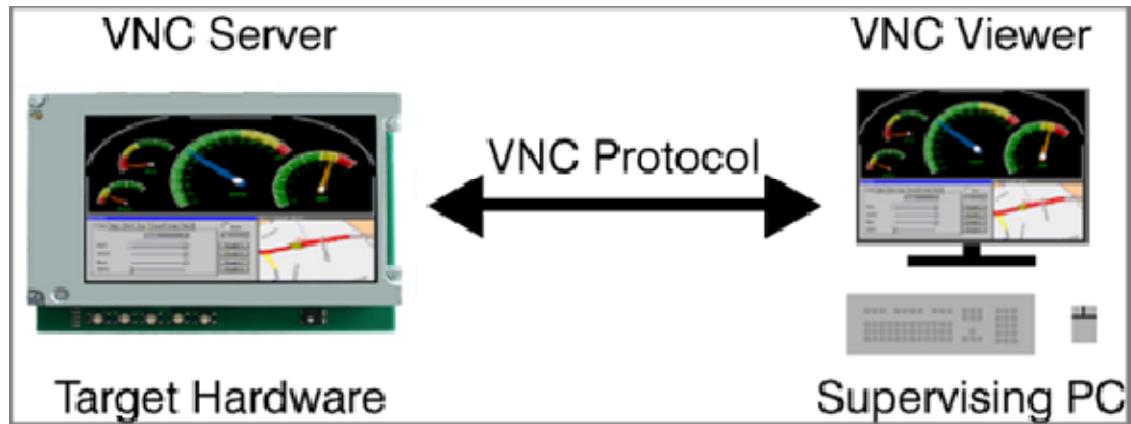
Hình 5.10 Hiệu ứng xoay và scale dùng memory device

- Window Manager:
 - o Là một hệ thống quản lý có cấu trúc thứ bậc: Mỗi layer sở hữu một desktop window. Mỗi desktop window có thể có các window con.
 - o Là một hệ thống dựa trên cơ chế gọi lại:Sự giao tiếp dựa trên cơ sở sự kiện gọi lại. Tất cả các hoạt động draw được thực hiện với sự kiện WM_PAINT.
 - o Là nền tảng cho thư viện widget: Tất cả các widget dựa trên các hàm của window manager.
- Thư viện Widget
 - o Widget là sự kết hợp giữa window với các đối tượng. Chúng yêu cầu một trình quản lý window.
 - o Một vài ví dụ về widget như hình bên dưới.



Hình 5.11 Một số ví dụ về Widget

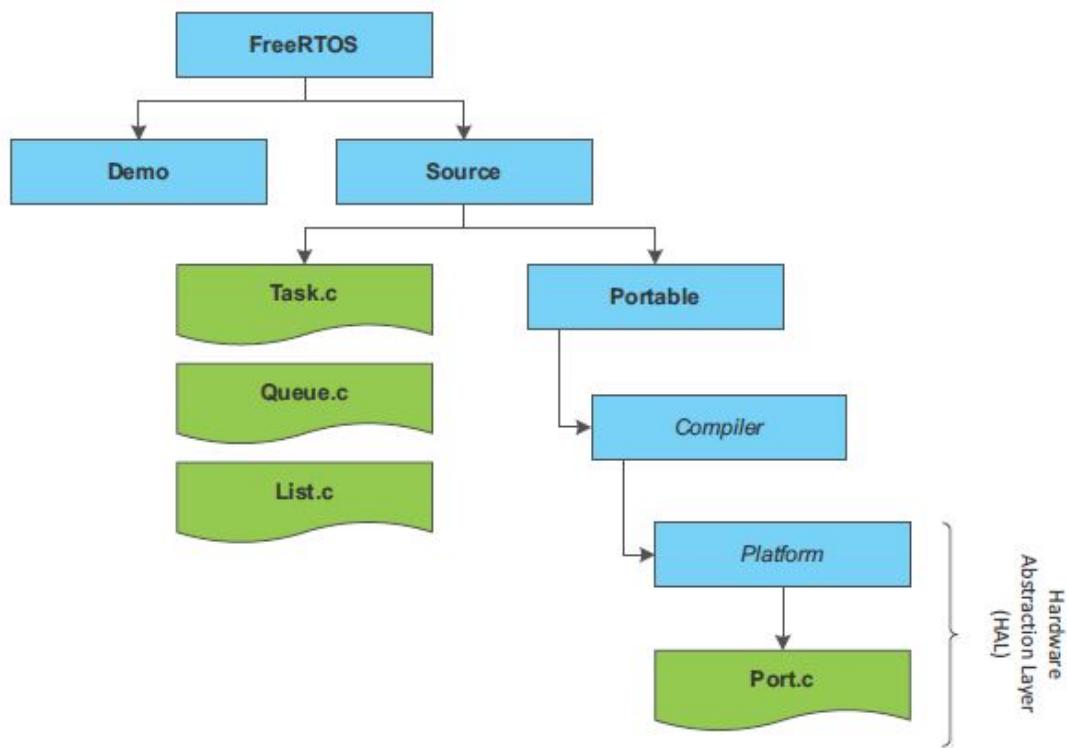
- o Các cách để thêm một widget: Ta có thể thêm một widget bằng một trong các cách sau:
 - ✓ Tạo trực tiếp: Thêm các hàm tòm tại cho mỗi widget
 - <WIDGET>_CreateEx(): Thêm mà không dùng đến user data.
 - <WIDGET>_CreateUser(): Thêm bằng cách dùng đến user data.
 - ✓ Tạo gián tiếp: Dùng các hàm thêm dialog box và một structure GUI_WIDGET_CREATE_INFO bao gồm một pointer trả đến hàm con: <WIDGET>_CreateIndirect(): thêm bằng hàm thêm dialog box
- VNC server
 - o VNC server được dùng để kết nối đến phần cứng nhúng đến mạng PC thông qua TCP/IP để có thể:
 - ✓ Xem nội dung LCD thông qua màn hình giám sát PC
 - ✓ Điều khiển môi trường nhúng thông qua chuột



Hình 5.12 Minh họa VNC server

5.2.3 CMSIS-RTOS Module

- Tổng quan về FreeRTOS:
 - FreeRTOS là một lớp của RTOS được thiết kế đủ nhỏ để hoạt động trên một vi điều khiển.
 - FreeRTOS là hệ điều hành thời gian thực được thiết kế cho các hệ thống nhúng nhỏ.
 - Một vài điểm nổi bật của FreeRTOS:
 - o FreeRTOS core có thể hoạt động với các cấu hình tùy chọn preemptive, cooperative và hybrid.
 - o Hỗ trợ đến 27 kiến trúc như ARM 7, ARM Cortex M3, M4,...
 - o FreeRTOS-MPU hỗ trợ Cortex M3 Memory Protection Unit (MPU).
 - o Được thiết kế nhỏ, đơn giản và dễ sử dụng.
 - o Cấu trúc code dễ portable bởi phần lớn được viết bằng ngôn ngữ C.
 - o Queue, binary semaphore, counting semaphore, recursive semaphore và mutex cho việc giao tiếp và đồng bộ giữa các tác vụ hoặc giữa các tác vụ và ngắt.
 - o Hỗ trợ mutex với độ ưu tiên kế thừa.
- Tổ chức source của FreeRTOS:
 - Cấu trúc thư mục FreeRTOS sau khi tải về rất đơn giản, và FreeRTOS kernel bao gồm 4 file (Một số file được thêm vào nếu như có yêu cầu dùng software timer hoặc co-routine).



Hình 5.13 Tô chúc mã nguồn FreeRTOS

- Lõi RTOS bao gồm 3 file Task.c, Queue.c, List.c trong thư mục FreeRTOS/Source. Ở cùng thư mục còn có 2 file tùy chọn là timers.c và croutine.c để thực thi chức năng software timer và co-routine.
- Bảng API FreeRTOS:

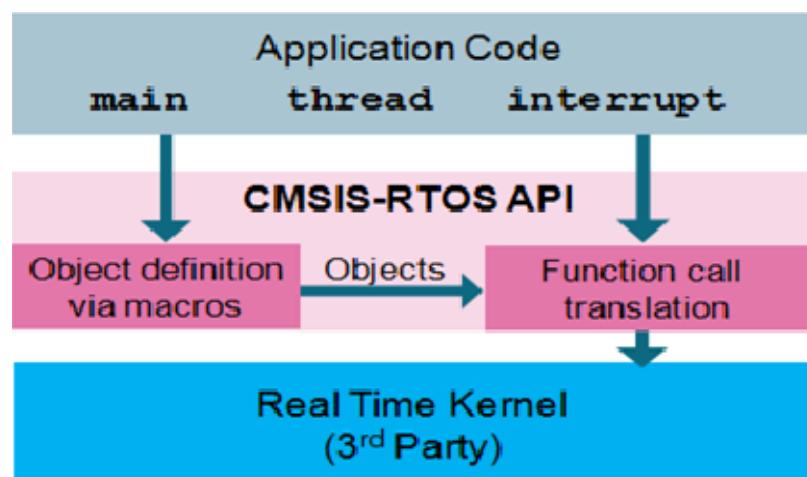
Bảng 5.1 Bảng các API FreeRTOS

Loại API	API
Task Creation	-xTaskCreate -xTaskDelete
Task Control	-vTaskDelay -vTaskDelayUntil -uxTaskPriorityGet -vTaskPrioritySet -vTaskSuspend -vTaskResume - xTaskResumeFromISR - vTaskSetApplicationTag - xTaskCallApplicationTaskHook
Task Utilities	-xTaskGetCurrentTaskHandle -xTaskGetSchedulerState -uxTaskGetNumberOfTasks -vTaskList -vTaskStartTrace

	-ulTaskEndTrace -vTaskGetRunTimeStats
Kernal Control	-vTaskStartScheduler -vTaskEndScheduler -vTaskSuspendAll -xTaskResumeAll
Queue Management	- xQueueCreate -xQueueSend - xQueueReceive -xQueuePeek -xQueueSendFromISR -xQueueSendToBackFromISR -xQueueSendToFrontFromISR -xQueueReceiveFromISR -vQueueAddToRegistry -vQueueUnregisterQueue
Semaphores	-vSemaphoreCreateBinary -vSemaphoreCreateCounting -xSemaphoreCreateMutex -xSemaphoreTake -xSemaphoreGive -xSemaphoreGiveFromISR

- Module CMSIS-RTOS:

- Tổng quan:
 - CMSIS-RTOS là một API cho hệ điều hành thời gian thực. Nó cung cấp các giao tiếp lập trình được chuẩn hóa để dễ portable đến nhiều RTOS và do đó cho phép các software template, middleware, library và những component khác có thể làm việc thông qua sự hỗ trợ của một hệ điều hành.
 - Module này được đại diện bởi file cmsis_os.c/h nằm ở trong thư mục “Middlewares\Third_Party\FreeRTOS\CMSIS_RTOS”



Hình 5.14 Cấu trúc module CMSIS RTOS

- Các API CMSIS-RTOS cung cấp các thuộc tính và chức năng sau:
 - ✓ Function name , identifier và parameter được mô tả dễ hiểu.
 - ✓ Quản lý luồng cho phép định nghĩa, thêm và điều khiển luồng.
 - ✓ Dịch vụ ngắn (ISR) có thể gọi nhiều hàm CMSIS-RTOS. Khi các hàm CMSIS-RTOS không thể gọi từ ISR thì nó sẽ từ chối yêu cầu.
 - ✓ Có ba sự kiện luồng khác nhau hỗ trợ giao tiếp giữa các luồng hoặc ISR
 - Signal: Là cờ có thể được dùng như một điều kiện cụ thể đến một luồng.
 - Message: Là một giá trị 32 bit có thể được gửi đến một luồng hoặc một ISR. Message là bộ đệm trong một queue. Loại message và kích thước queue được định nghĩa trong một descriptor.
 - Mail: Là một khối bộ nhớ cố định có thể được gửi đến một luồng hoặc một ISR. Mail được làm bộ trong một queue và phân bổ bộ nhớ được cung cấp. Loại của mail và kích thước queue được định nghĩa trong một descriptor.
 - Quản lý mutex và semaphore được kết hợp.
 - Thời gian CPU có thể được lên lịch.
- Bảng API CMSIS-RTOS

Bảng 5.2 Bảng các API của module CMSIS RTOS

Module	API	Mô tả
Kernel Information và Control	osKernelInitialize	Khởi động kernel RTOS
	osKernelStart	Bắt đầu kernel RTOS
	osKernelRunning	Chất vấn nếu như kernel đang chạy
	osKernelSys Tick	Lây thời gian của hệ thống
	osKernelSys TickFrequency	Tần số của hệ thống (Hz)
	osKernelSys TickMicroSec	Chuyển đổi giá trị us sang giá trị timer của hệ thống RTOS
Quản lý Thread: Định nghĩa, thêm và điều khiển các hàm Thread	osThreadCreate	Bắt đầu thực thi một Thread function
	osThreadTerminate	Dừng thực thi một Thread function
	osThreadYield	Thực thi Thread function đã sẵn sàng tiếp theo
	osThreadGetId	Lấy ID của thread tham khảo đến thread này
	osThreadSetPriority	Thay đổi độ ưu tiên của một thread function
	osThreadGetPriority	Lấy độ ưu tiên của một thread hiện tại
Generic Wait Functions: Đợi một chu kỳ hoặc sự kiện không rõ ràng	osDelay	Đợi một thời gian cụ thể
	osWait	Đợi bắt đầu kì sự kiện nào loại Signal, Mail hay Message.

Timer Management: Thêm và điều khiển timer và timer callback	osTimerCreate	Định nghĩa thuộc tính của timer callback function
	osTimerStart	Khởi động hoặc tái khởi động với giá trị thời gian cụ thể
Signal Management: Điều khiển hoặc đợi cờ tín hiệu	osSignalSet	Set cờ tín hiệu của một thread
	osSignalClear	Reset cờ tín hiệu của một thread
	osSignalClear	Định chỉ thực thi cho đến khi cờ tín hiệu cụ thể được set
Mutex Management: Synchronize thread execution with a Mutex.	osMutexCreate	Định nghĩa và khởi động một mutex
	osMutexWait	Thu nhận một mutex hoặc đợi cho đến khi nó trở nên có sẵn
	osMutexRelease	Phát hành một mutex
	osMutexDelete	Xóa một mutex
Semaphore Management: Giảm sát truy cập đến tài nguyên được chia sẻ	osSemaphoreCreate	Định nghĩa và khởi động một semaphore
	osSemaphoreWait	Thu nhận một token semaphore hoặc đợi cho đến khi nó có sẵn
	osSemaphoreRelease	Phát hành một token semaphore
	osSemaphoreDelete	Xóa một semaphore
Memory Pool Management: Định nghĩa và quản lý bộ nhớ	osPoolCreate	Định nghĩa và khởi động bộ nhớ pool
	osPoolAlloc	Phân bổ một khối bộ nhớ
	osPoolCAlloc	Phân bổ một khối bộ nhớ và thiết lập giá trị 0 cho bộ nhớ này
	osPoolFree	Giải phóng bộ nhớ Pool
Message Queue Management: Điều khiển, gửi, nhận hoặc đợi các message.	osMessageCreate	Định nghĩa và khởi động một message queue
	osMessagePut	Gửi một message đến một message queue
	osMessageGet	Lấy một message hoặc định chỉ thực thi luồng cho đến khi nhận được message
Mail Queue Management: Điều khiển, gửi, nhận hoặc đợi các mail.	osMailCreate	Định nghĩa và khởi động một mail queue với kích thước bộ nhớ cố định
	osMailAlloc	Phân bổ một khối bộ nhớ
	osMailCAlloc	Phân bổ một khối bộ nhớ và thiết lập zero cho khối này
	osMailPut	Gửi một khối bộ nhớ đến một mail queue
	osMailGet	Lấy một mail hoặc trì hoãn thread cho đến khi nhận được mail
	osMailFree	Giải phóng bộ nhớ cho mail queue

5.3 Thiết kế các hàm chức năng

- Chương trình có 3 tác vụ chính được quản lý bởi FreeRTOS thông qua gói CMSIS-OS layer.
 - Start Thread: Đây là tác vụ hạt nhân của chương trình chính, có độ ưu tiên bình thường và quản lý các module và một vài dịch vụ tạo độ trễ.

- GUI Thread: Đây là tác vụ khởi động menu chính và sau đó xử lý tác vụ giao diện đồ họa bởi STemWin.

```

234  /**
235   * @brief Start task
236   * @param argument: pointer that is passed to the thread function as start argument.
237   * @retval None
238   */
239 static void GUIThread(void const * argument)
240 {
241     (...)

243     /* Show the main menu */
244     k_InitMenu();

246     /* Gui background Task */
247     while(1)
248     {
249         GUI_Exec();
250         osDelay(30);
251     }
252 }
```

Hình 5.15 Minh họa tác vụ GUIThread

- Timer Callback: Đây là tác vụ quản lý timer để cập nhật trạng thái, tác vụ này được gọi mỗi 40ms.

```

262 /**
263  * @brief Timer callback (40 ms)
264  * @param n: Timer index
265  * @retval None
266  */
267 static void TimerCallback(void const *n)
268 {
269     k_TouchUpdate();
270 }
```

Hình 5.16 Minh họa tác vụ Timer Callback

- Một vài function chính được thiết kế :
- SystemClock_Config(): Cấu hình clock cho hệ thống 180 Mhz.

```

static void SystemClock_Config(void)
{
    RCC_ClkInitTypeDef RCC_ClkInitStruct;
    RCC_OscInitTypeDef RCC_OscInitStruct;

    __PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
```

```

RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLM = 25;
RCC_OscInitStruct.PLL.PLLN = 360;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
RCC_OscInitStruct.PLL.PLLQ = 7;
HAL_RCC_OscConfig(&RCC_OscInitStruct);
HAL_PWREx_ActivateOverDrive();
RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK / RCC_CLOCKTYPE_HCLK /
RCC_CLOCKTYPE_PCLK1 / RCC_CLOCKTYPE_PCLK2);
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);
}

```

- k_CalendarBkupInit(): Khởi động thanh ghi RTC

```

void k_CalendarBkupInit(void)
{
    /*##-1- Configure the RTC peripheral #####
    /* Configure RTC prescaler and RTC data registers */
    /* RTC configured as follow:
    - Hour Format      = Format 24
    - Asynch Prediv   = Value according to source clock
    - Synch Prediv    = Value according to source clock
    - OutPut          = Output Disable
    - OutPutPolarity  = High Polarity
    - OutPutType       = Open Drain */
    RtcHandle.Instance = RTC;
    RtcHandle.Init.HourFormat = RTC_HOURFORMAT_24;
    RtcHandle.Init.AsynchPrediv = RTC_ASYNCH_PREDIV;
    RtcHandle.Init.SynchPrediv = RTC_SYNCH_PREDIV;
    RtcHandle.Init.OutPut = RTC_OUTPUT_DISABLE;
    RtcHandle.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
    RtcHandle.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;

    if(HAL_RTC_Init(&RtcHandle) != HAL_OK)
    {
    }
}

```

Hình 5.17 Minh họa hàm k_CalendarBkupInit()

- LCD_LL_Init(): Cấu hình LCD controller

```
/* DeInit */
HAL_LTDC_DeInit(&hltdc);

/* Set LCD Timings */
hltdc.Init.HorizontalSync = 48;
hltdc.Init.VerticalSync = 3;
hltdc.Init.AccumulatedHBP = 88;
hltdc.Init.AccumulatedVBP = 32;
hltdc.Init.AccumulatedActiveH = 512;
hltdc.Init.AccumulatedActiveW = 888;
hltdc.Init.TotalHeigh = 525;
hltdc.Init.TotalWidth = 928;

/* background value */
hltdc.Init.Backcolor.Blue = 0;
hltdc.Init.Backcolor.Green = 0;
hltdc.Init.Backcolor.Red = 0;

/* LCD clock configuration */
/* PLLSAI_VCO Input = HSE_VALUE/PLL_M = 1 Mhz */
/* PLLSAI_VCO Output = PLLSAI_VCO Input * PLLSAIN = 192 Mhz */
/* PLLLCDCLK = PLLSAI_VCO Output/PLLSAIR = 192/3 = 64 Mhz */
/* LTDC clock frequency = PLLLCDCLK / LTDC_PLLSAI_DIVR_4 = 64/4 = 16 Mhz */
PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_LTDC;
PeriphClkInitStruct.PLLSAI.PLLSAIN = 256;
PeriphClkInitStruct.PLLSAI.PLLSAIR = 2;
PeriphClkInitStruct.PLLSAIDivR = RCC_PLLSAIDIVR_4;
HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct);
```

Hình 5.18 Câu lệnh LCD controller

- k_CalibrationInit(): Cân chỉnh màn hình cảm ứng.

```

void k_CalibrationInit(void)
{
    int aPhysX[2], aPhysY[2], aLogX[2], aLogY[2], i;

    data1.d32 = k_BkupRestoreParameter RTC_BKP_DR0;
    data2.d32 = k_BkupRestoreParameter RTC_BKP_DR1;

    A2 = data2.b.A2 ;
    B2 = data2.b.B2 ;
    A1 = data1.b.A1 ;
    B1 = data1.b.B1 ;

    if(data2.b.IsCalibrated == 0)
    {
        GUI_SetBkColor(GUI_WHITE);
        GUI_Clear();
        GUI_SetColor(GUI_BLACK);
        GUI_UC_SetEncodeUTF8();
        // GUI_SetFont(&GUI_Font13B_ASCII);
        GUI_SetFont(&GUI_FontTimesNewRoman15_B);
        Explain();
        /* Set the logical values */
        aLogX[0] = 15;
        aLogY[0] = 15;
        aLogX[1] = LCD_GetXSize() - 15;
        aLogY[1] = LCD_GetYSize() - 15;
        /* Get the physical values of the AD converter for 2 positions */
        for (i = 0; i < 2; i++) {
            _GetPhysValues(aLogX[i], aLogY[i], &aPhysX[i], &aPhysY[i], _acPos[i]);
        }
    }
}

```

Hình 5.19 Cân chỉnh màn hình cảm ứng

- k_InitMenu(): Khởi tạo màn hình menu chức năng.

```
/*
void k_InitMenu(void)
{
    WM_HWIN hItem;
    uint8_t i = 0;

    settings.d32 = k_BkupRestoreParameter(CALIBRATION_GENERAL_SETTINGS_BKP);

    WM_SetCallback(WM_HBKWIN, _cbBk);

    WM_CreateWindowAsChild(0,
                           0,
                           LCD_GetXSize(),
                           25,
                           WM_HBKWIN,
                           WM_CF_SHOW | WM_CF_HASTRANS ,
                           _cbStatus,
                           0);

    hIcon = ICONVIEW_CreateEx(0,
                             26,
                             LCD_GetXSize(),
                             LCD_GetYSize() - 26,
                             WM_HBKWIN,
                             WM_CF_SHOW | WM_CF_HASTRANS | WM_CF_BGND ,
                             ICONVIEW_CF_AUTOSCROLLBAR_V,
                             ID_ICONVIEW_MENU,
                             112,
                             105);
}
```

Hình 5.20 Khởi động menu

- k_StorageInit(): Khởi động thẻ nhớ.

```

void k_StorageInit(void)
{
    /* Link the micro SD disk I/O driver */
    FATFS_LinkDriver(&SD_Driver, mSDDISK_Drive);

    /* Create USB background task */
    osThreadDef(STORAGE_Thread, StorageThread, osPriorityBelowNormal, 0, 2 * configMINIMAL_STACK_SIZE);
    osThreadCreate (osThread(STORAGE_Thread), NULL);

    /* Create Storage Message Queue */
    osMessageQDef(osqueue, 10, uint16_t);
    StorageEvent = osMessageCreate (osMessageQ(osqueue), NULL);

    /* Enable SD Interrupt mode */
    BSP_SD_Init();
    BSP_SD_ITConfig();

    if(BSP_SD_IsDetected())
    {
        osMessagePut (StorageEvent, MSDDISK_CONNECTION_EVENT, 0);
    }
}

```

Hình 5.21 Khởi động thẻ nhớ

- k_MemInit(): Khởi động bộ nhớ pool.

```

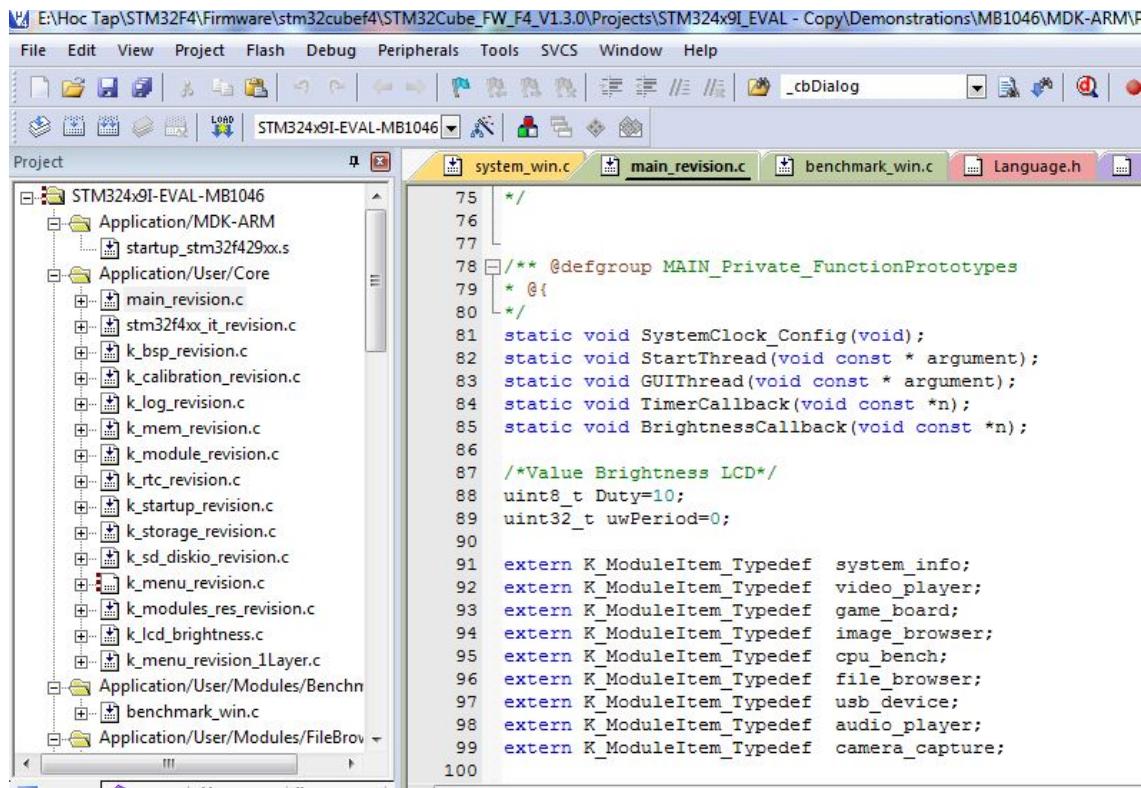
/**
 * @brief This function Initialize a memory pool for private allocator.
 * @param None
 * @retval None
 */
void k_MemInit(void)
{
    memset(&memory_pool, 0, sizeof(mem_TypeDef));
    memory_pool.mallocBase = MEM_BASE + sizeof(mem_TypeDef);
}

```

Hình 5.22 Khởi động bộ nhớ pool

5.4 Các phần mềm cần dùng trong thiết kế ứng dụng

- Trình biên dịch Keil ARM: Để biên dịch chương trình.



The screenshot shows the Keil MDK-ARM IDE interface. The title bar indicates the project is 'STM32x9I-EVAL-MB1046'. The menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Build. The project tree on the left shows the project structure under 'STM32x9I-EVAL-MB1046' with subfolders Application/MDK-ARM, Application/User/Core, Application/User/Modules/Benchmarks, and Application/User/Modules/FileBrowsing. The main code editor window displays the 'main_revision.c' file. The code starts with a comment block defining a group of private function prototypes:

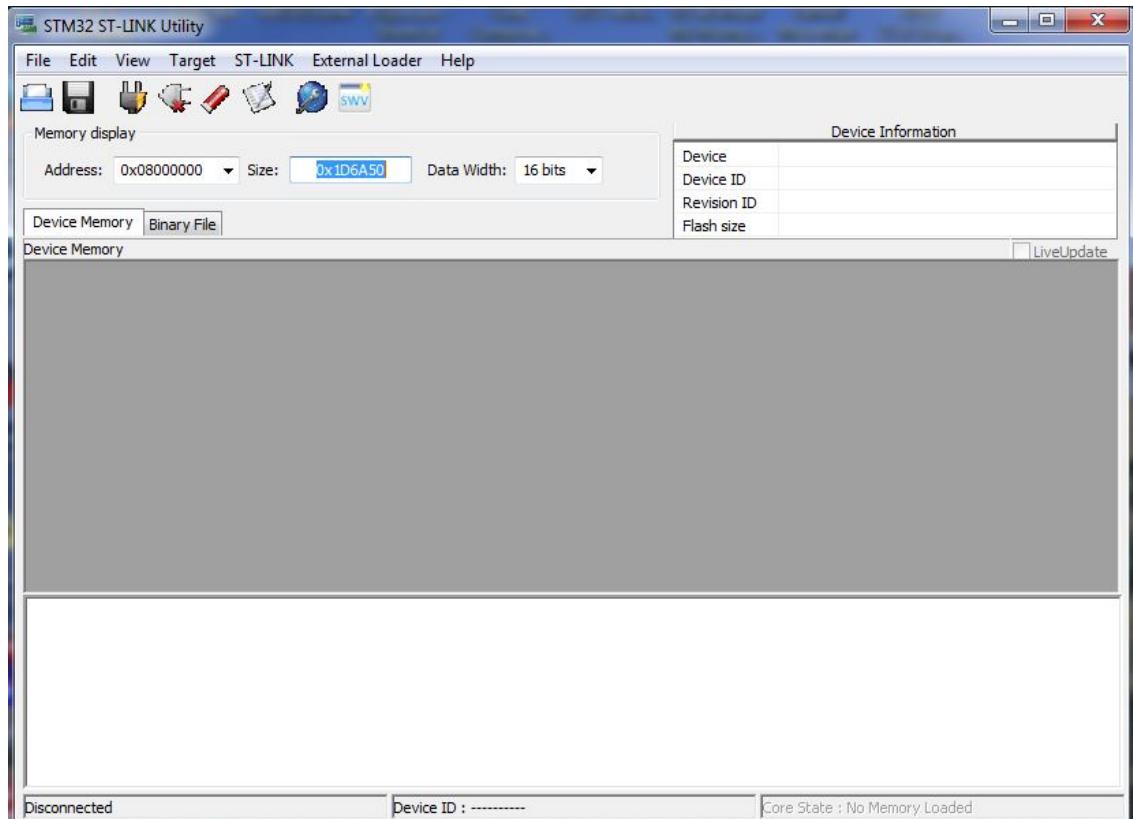
```

75 */
76
77
78 /**
79 * @{
80 */
81 static void SystemClock_Config(void);
82 static void StartThread(void const * argument);
83 static void GUIThread(void const * argument);
84 static void TimerCallback(void const *n);
85 static void BrightnessCallback(void const *n);
86
87 /*Value Brightness LCD*/
88 uint8_t Duty=10;
89 uint32_t uwPeriod=0;
90
91 extern K_ModuleItem_Typedef system_info;
92 extern K_ModuleItem_Typedef video_player;
93 extern K_ModuleItem_Typedef game_board;
94 extern K_ModuleItem_Typedef image_browser;
95 extern K_ModuleItem_Typedef cpu_bench;
96 extern K_ModuleItem_Typedef file_browser;
97 extern K_ModuleItem_Typedef usb_device;
98 extern K_ModuleItem_Typedef audio_player;
99 extern K_ModuleItem_Typedef camera_capture;
100

```

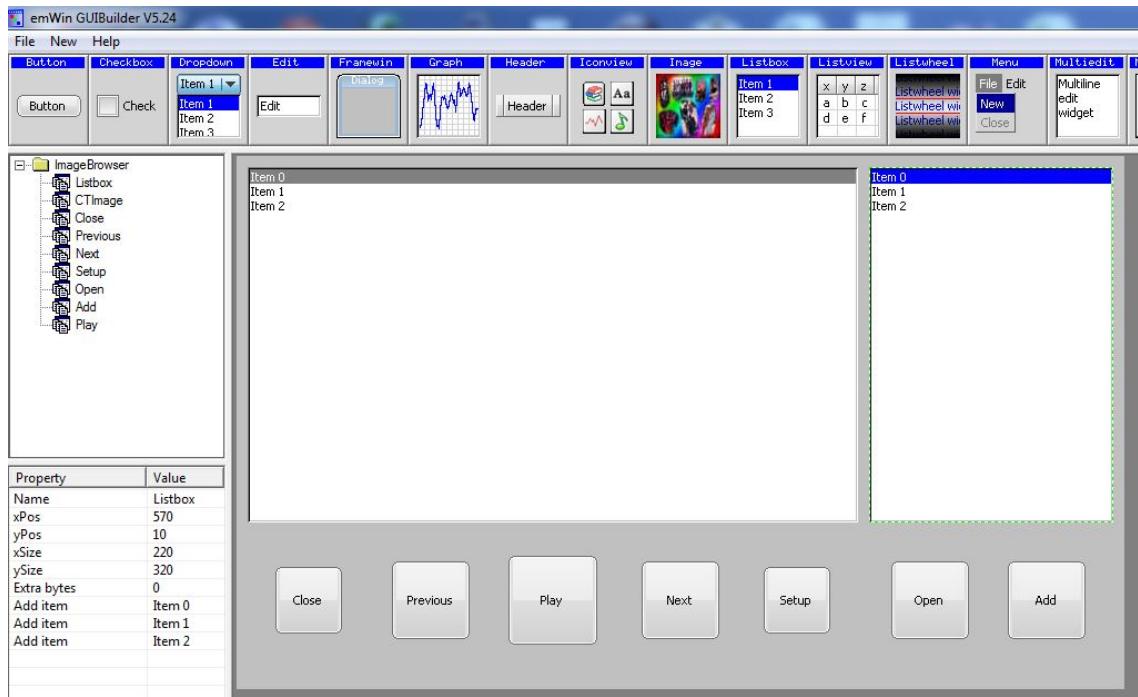
Hình 5.23 Minh họa trình biên dịch Keil ARM

- STM32 ST-LINK Utility: Dùng để download code vào phần cứng



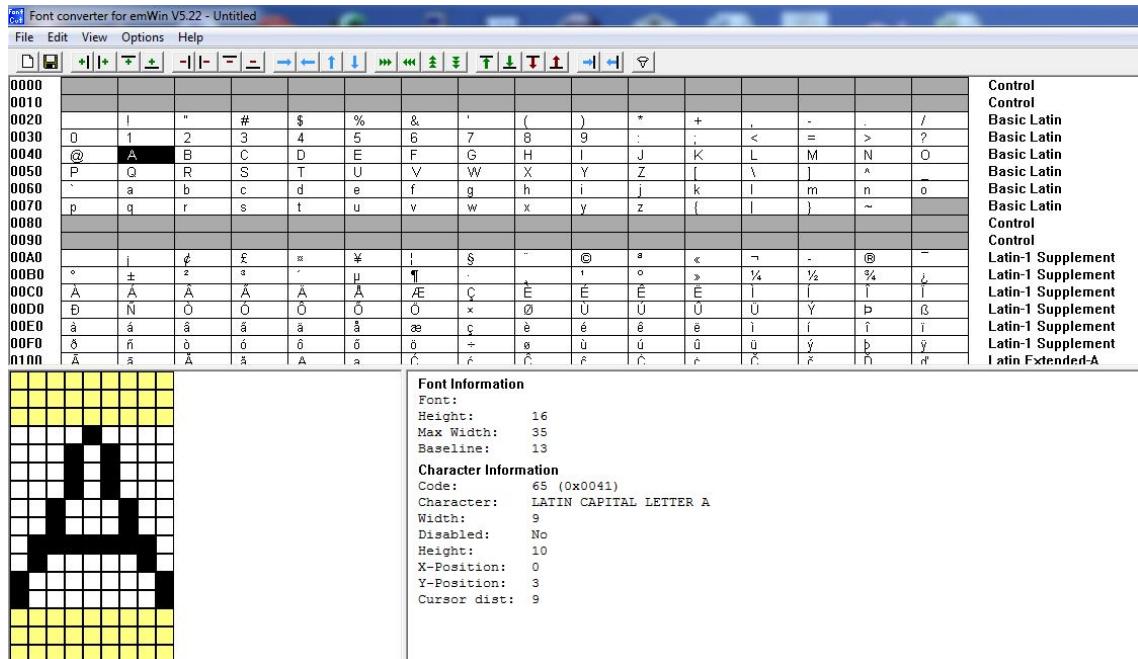
Hình 5.24 Hình minh họa STM32 ST-LINK Utility

- emWin GUIBuilder: Dùng để thiết kế giao diện đồ họa



Hình 5.25 Minh họa giao diện phần mềm emWin GUIBuilder

- Bitmap Converter: Dùng để lấy mã số Hex của hình ảnh
- Font Converter: Dùng để tạo Font cho ứng dụng



Hình 5.26 Giao diện phần mềm tạo Font chữ

- Visual Studio: Dùng để mô phỏng giao diện đồ họa.

The screenshot shows the Microsoft Visual Studio interface. In the center, there is a code editor window displaying C++ code for a GUI system. The code includes declarations for various GUI widgets like windows, text boxes, checkboxes, spin boxes, and buttons, along with their properties such as ID, position, and size. To the right of the code editor is the Solution Explorer, which lists several projects and files under a solution named 'SystemWin'. Projects include 'DEMO_SEGER', 'DemoVD', 'LUAN VAN', 'FileBrowser', 'ImageBrowser', 'K_Init', 'NewFilter2', 'StartUp', and 'SystemWin'. Under 'SystemWin', there are files like 'Draw_PNG.cpp', 'MEMDEV_AutoDev.c', 'FileBrowser_main.c', and 'SystemWin.c'. At the bottom of the screen, the status bar shows '100 %'.

Hình 5.27 Hình minh họa visual studio

6. KẾT QUẢ THỰC HIỆN

6.1 Kết quả thực hiện phần cứng

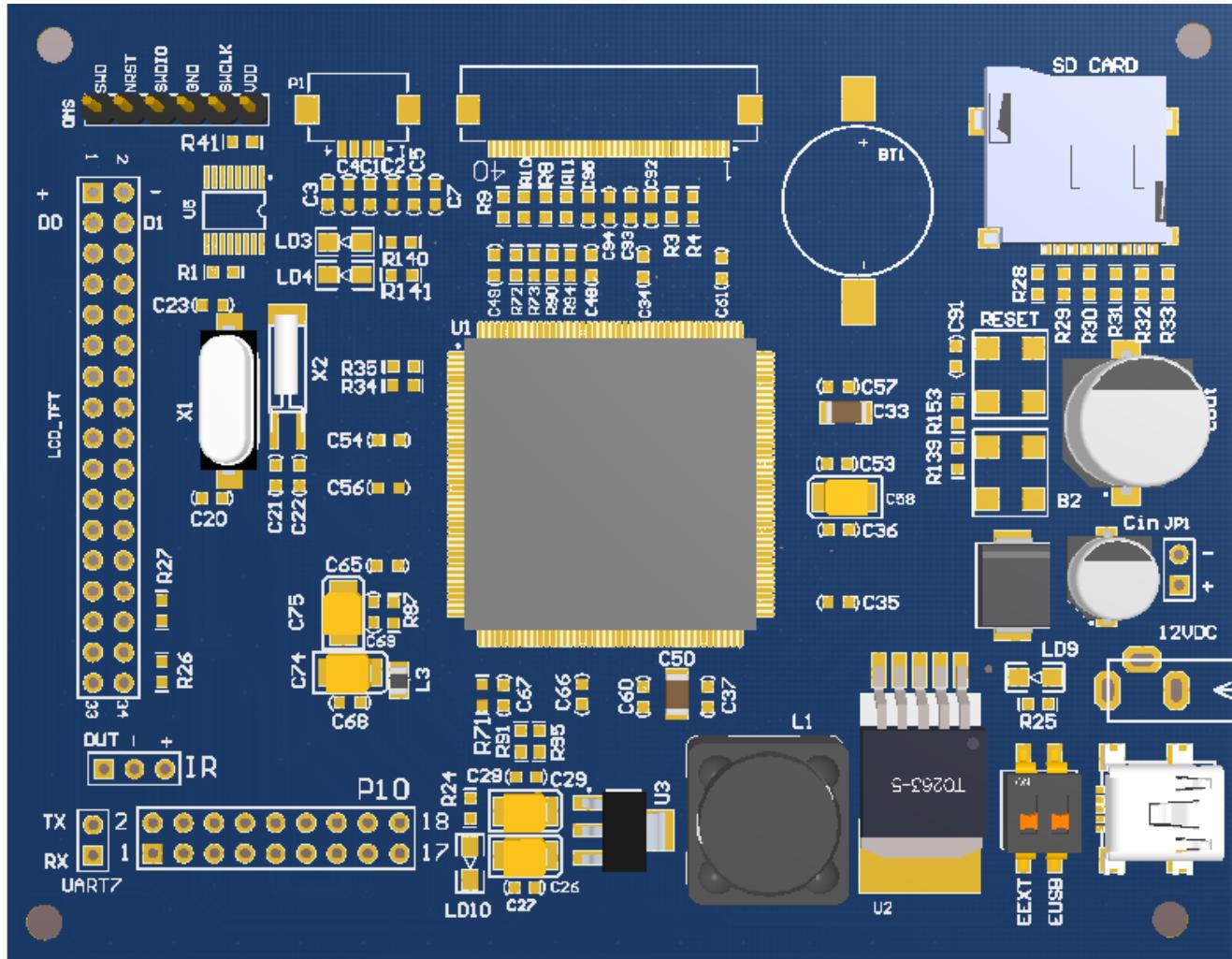
- Phần mềm sử dụng thiết kế mạch: Altium 14



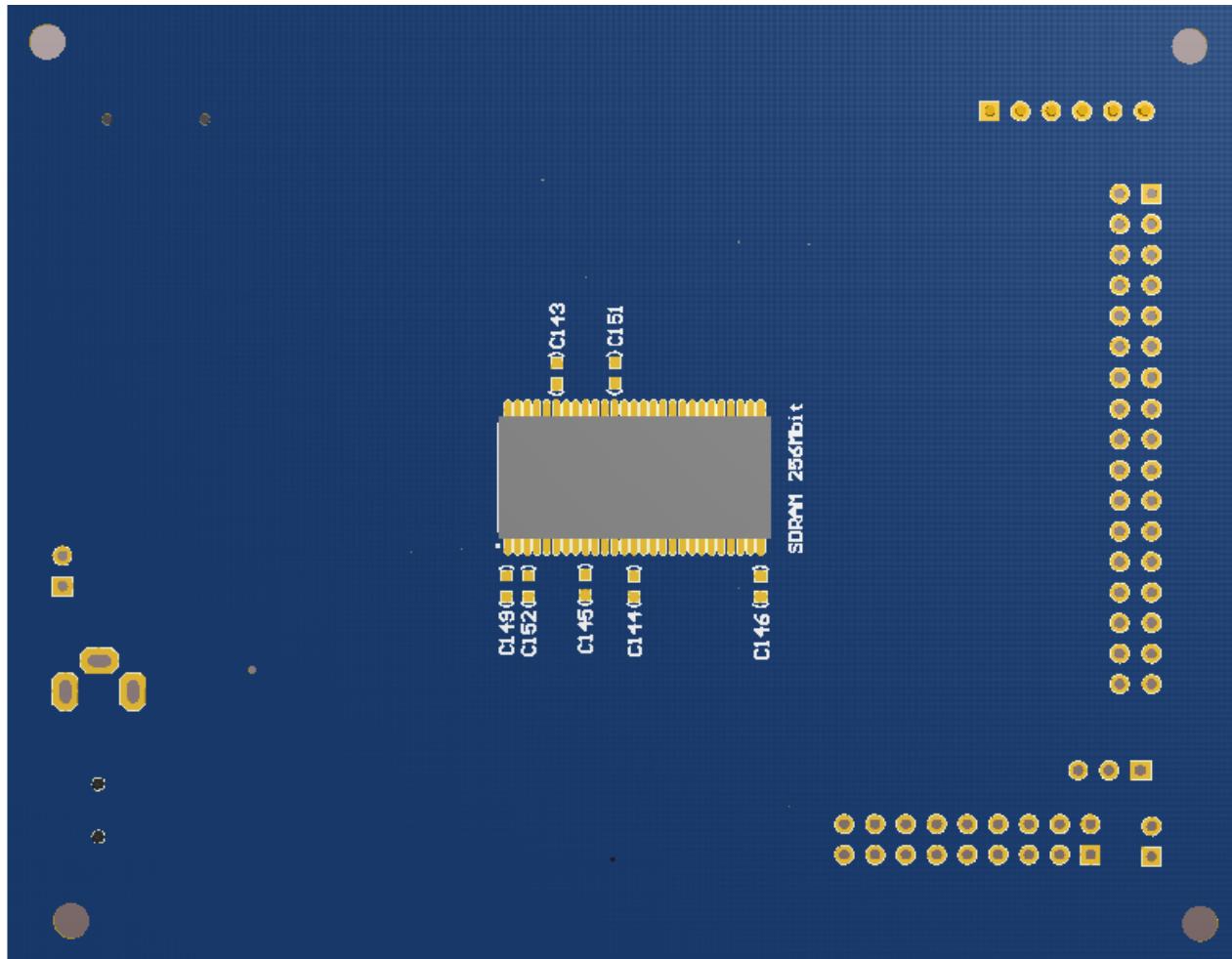
Hình 6.1 Hình minh họa phần mềm Altium

- Dụng cụ kiểm tra mạch: VOM
- Các bước test mạch:

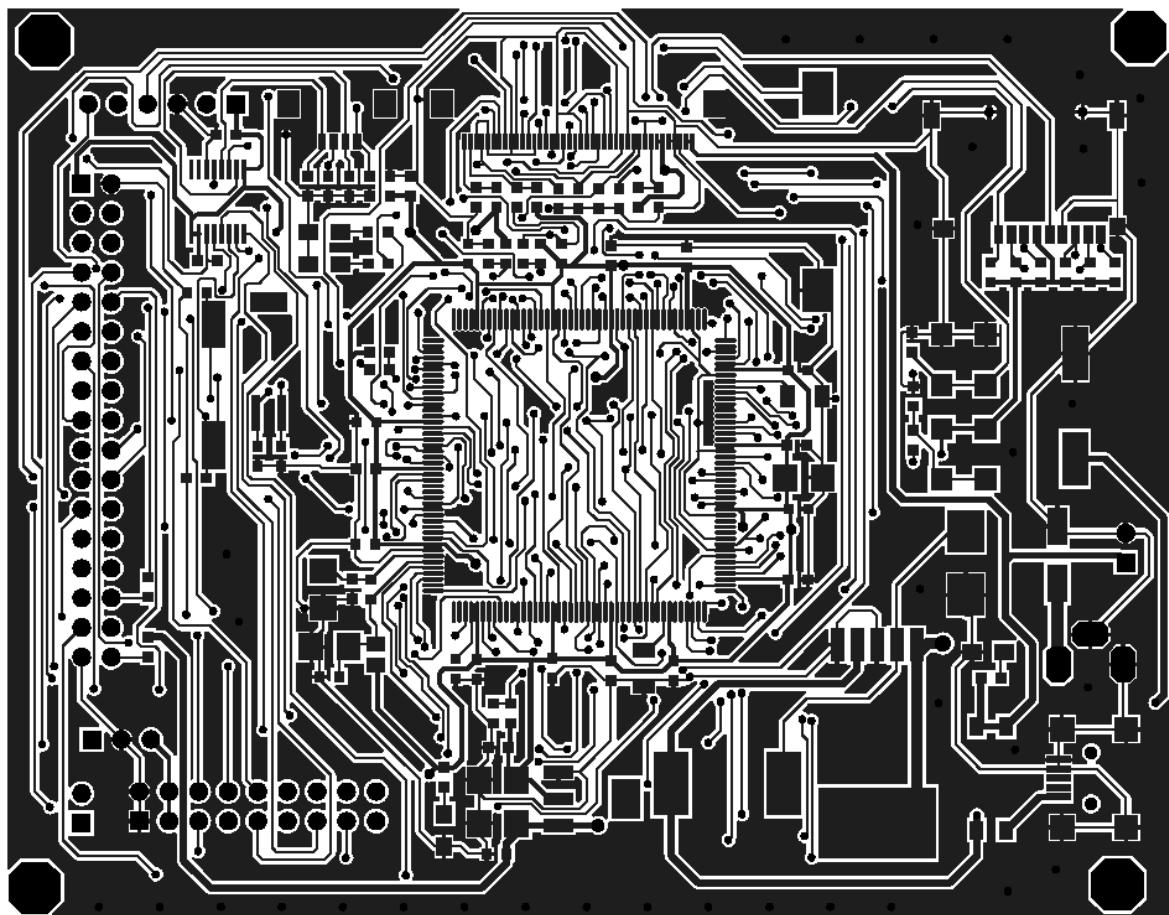
- Kiểm tra nguồn cấp 5V đầu vào.
 - Kiểm tra nguồn cấp 3.3V cho vi điều khiển.
 - Kiểm tra ngắn mạch, hở mạch.
- Kết quả thiết kế thu được:



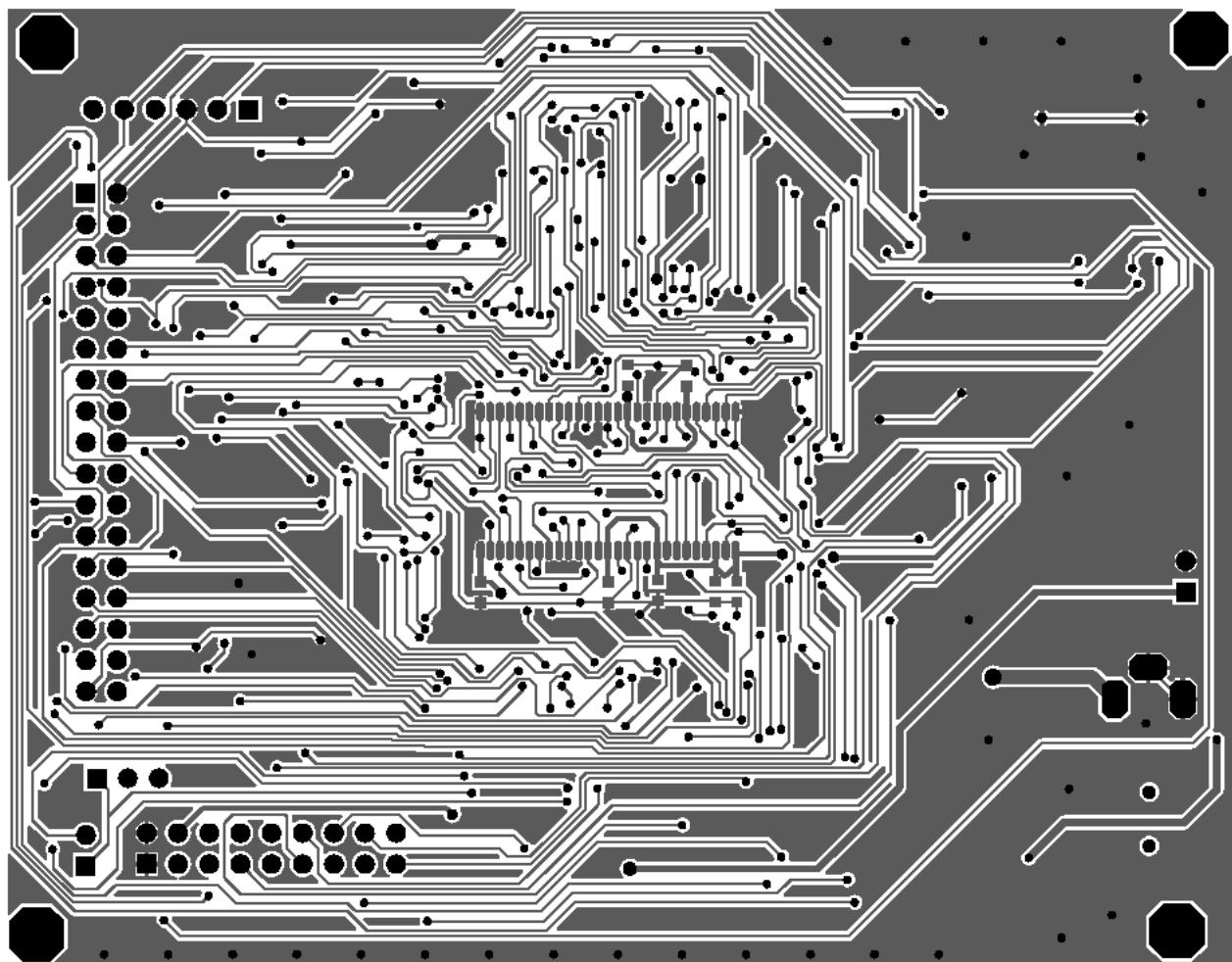
Hình 6.2 Kết quả thiết kế mạch lớp Top trên Altium



Hình 6.3 Kết quả thiết kế mạch lớp Bottom trên Alitum

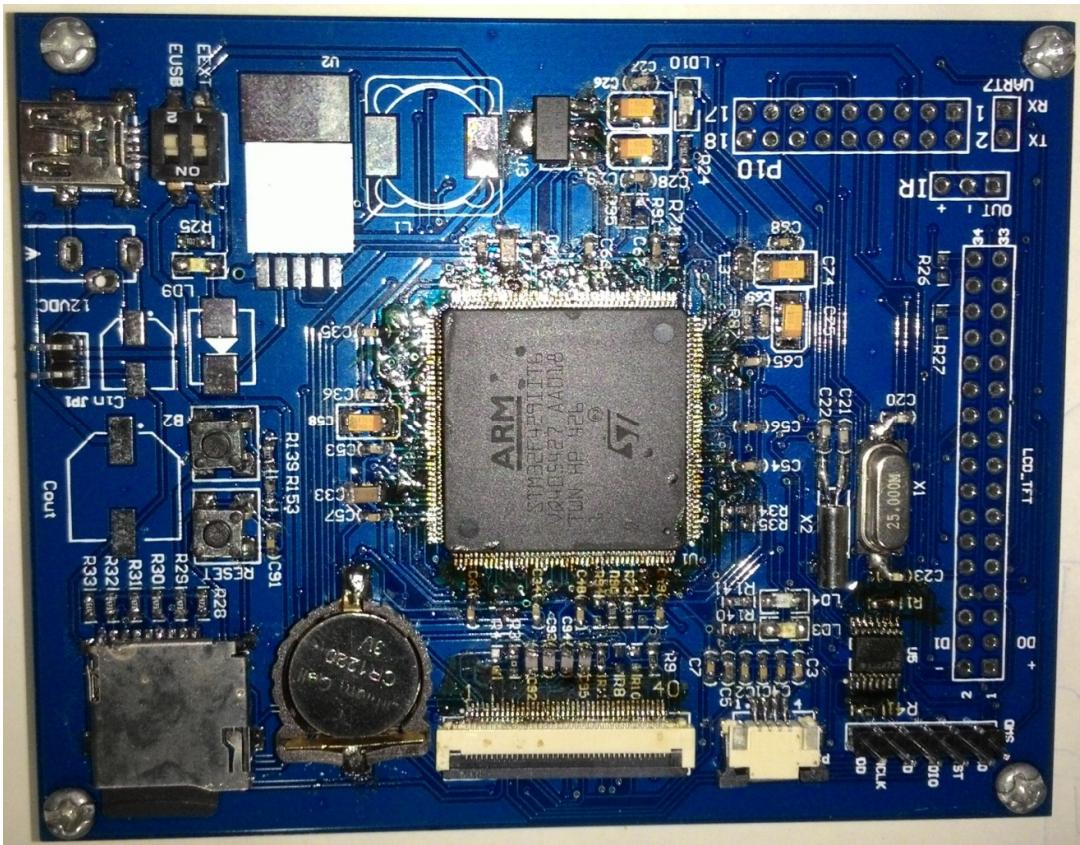


Hình 6.4 Mạch in lớp Top



Hình 6.5 Mạch in lớp Bottom

- Kết quả phần cứng thực tế:

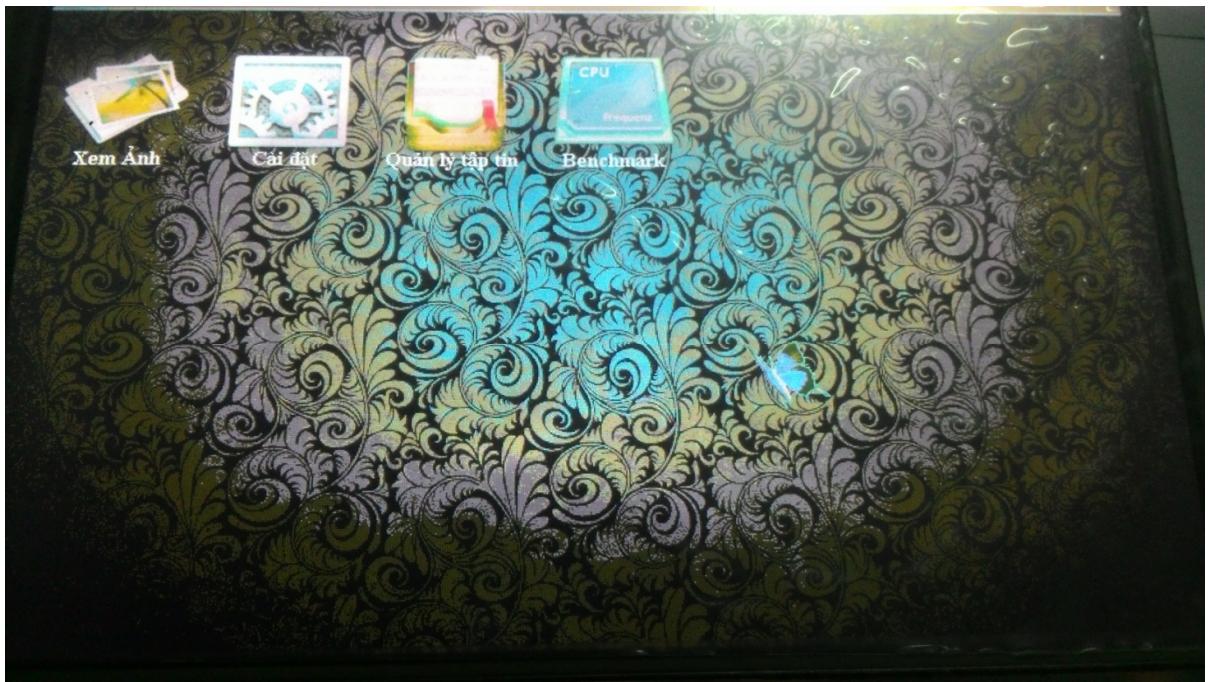


Hình 6.6 Kết quả board mạch mặt trên sau khi hoàn thành

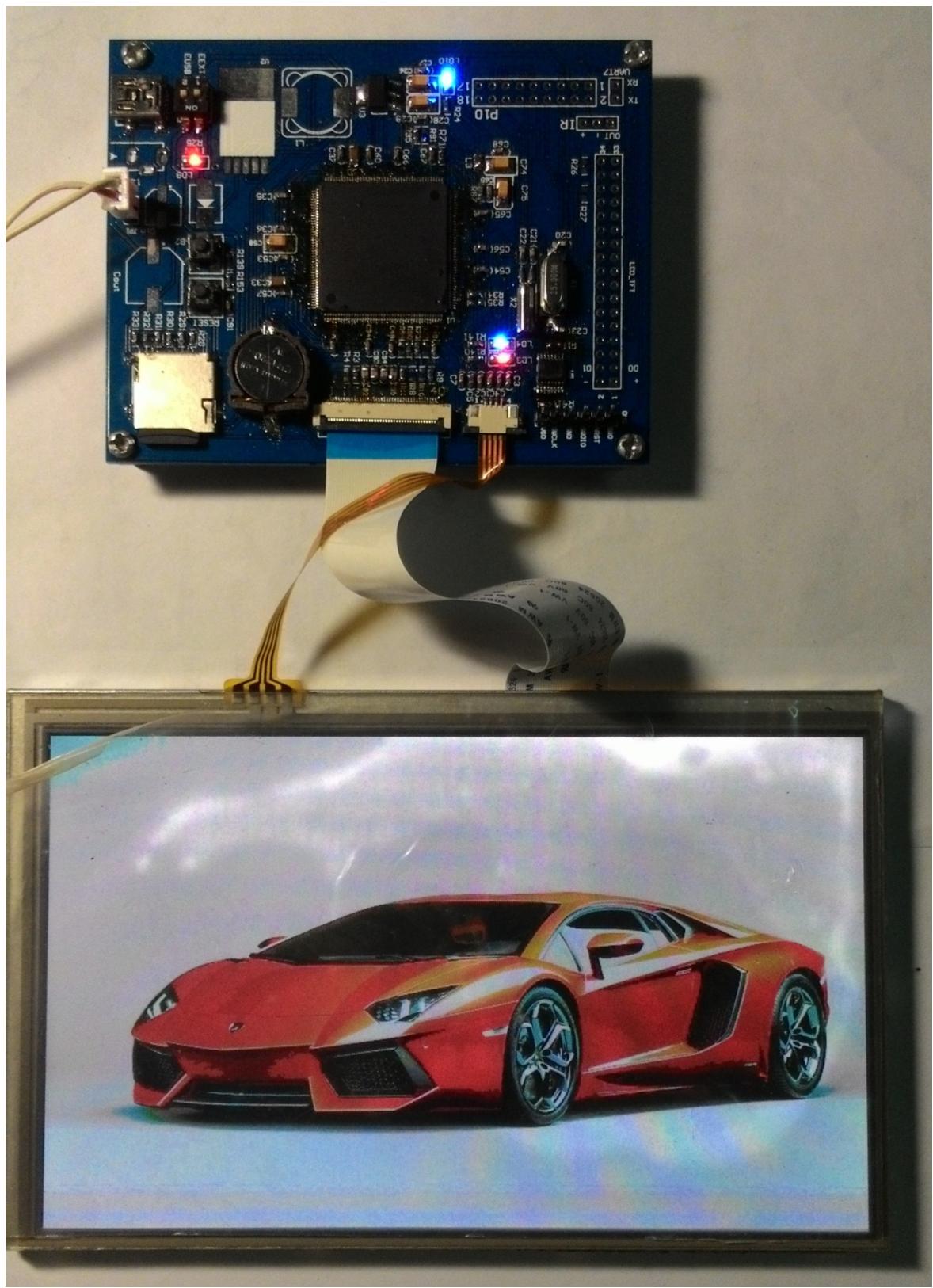


Hình 6.7 Kết quả board mạch mặt dưới sau khi hoàn thành

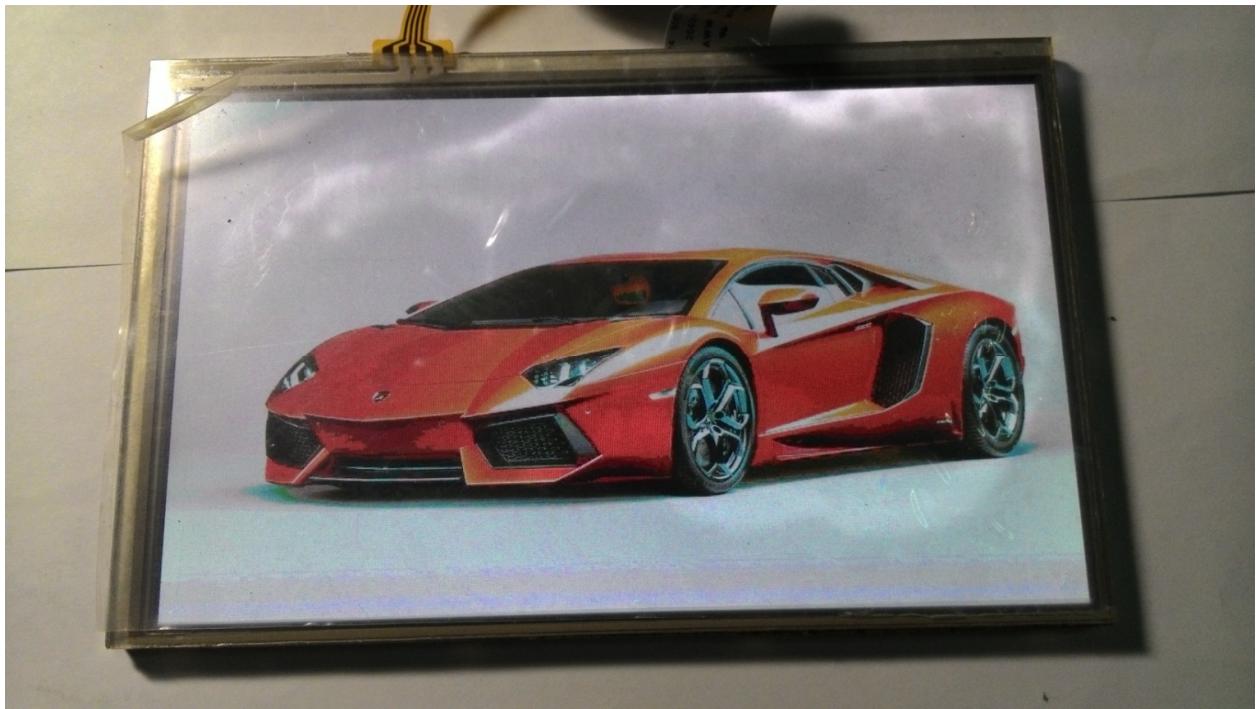
- Kết quả phần mềm thu được:



Hình 6.8 Một số kết quả phần mềm thu được



Hình 6.9 Một số kết quả phần mềm thu được



Hình 6.10 Một số kết quả phần mềm thu được

7. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1 Kết luận

- Quá trình thực hiện đề tài đã giúp em có thêm một số kinh nghiệm bổ ích:
 - Có được cái nhìn tổng quan về hệ thống nhúng cũng như nắm vững được quá trình thiết kế một hệ thống nhúng từ cơ bản đến phức tạp.
 - Nắm được quy trình thiết kế một board mạch từ khâu vẽ mạch nguyên lý đến layout bố trí linh kiện cho phù hợp. Các kỹ thuật đi dây sao cho mạch hoạt động ổn định, đặc biệt để giảm nhiễu EMI khi dùng các vi điều khiển hoạt động ở tần số cao.
 - Có được cái nhìn tổng quan về cách dùng các công cụ vẽ mạch để phục vụ việc thiết kế phần cứng cho một hệ thống nhúng bất kì.
 - Có cái nhìn tổng quan về phần mềm nhúng, nắm được cách kết hợp các driver, middleware, các component có trước thành một khối thống nhất phục vụ cho ứng dụng nhúng của mình hoạt động với một số chức năng chuyên biệt nào đó.
 - Có cái nhìn tổng quan về sự kết hợp giữa phần cứng và phần mềm trong một hệ thống nhúng.

- Ưu và khuyết điểm của kết quả nghiên cứu đề tài:
 - Ưu điểm:
 - o Tốc độ quét ảnh nhanh, độ trễ thấp
 - o Hình ảnh hiển thị rõ ràng, màu sắc trung thực
 - o Board mạch nhỏ gọn, phù hợp với ứng dụng
 - o Ứng dụng chạy ổn định.
 - Khuyết điểm:
 - o Tốc độ quét ảnh tuy nhanh nhưng so với ứng dụng thực tế như điện thoại, laptop thì vẫn còn một khoảng cách khá lớn.

7.2 Hướng phát triển

- Hướng phát triển đề tài: Có thể tích hợp thêm nhiều chức năng đa phương tiện vào ứng dụng như: camera chụp ảnh, quay phim, xem video hay thêm các module GPS để định vị,...
- Khả năng ứng dụng của đề tài: Ứng dụng đề tài để tạo các ứng dụng như làm khung ảnh số treo tường hay làm bối cảnh ứng dụng nào có liên quan đến tương tác với người dùng qua màn hình.

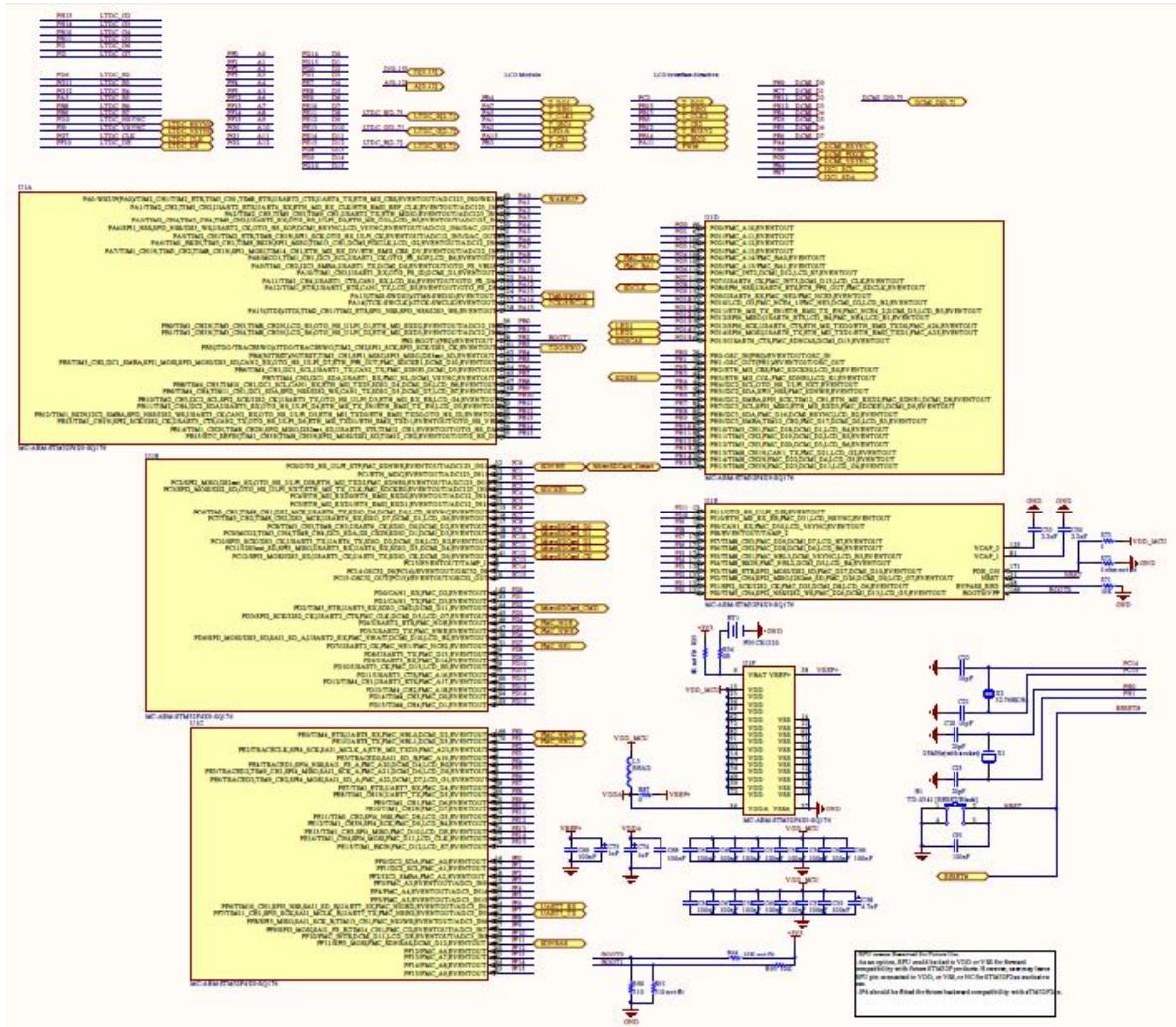
8. TÀI LIỆU THAM KHẢO

- [1] Đặng Thành Tín, “Hệ thống máy tính và ngôn ngữ C”, Nhà xuất bản Đại học Quốc Gia TPHCM.
- [2] Richard Barry. “Using the FreeRTOS™ Real Time Kernel”, ARM Cortex-M3 Edition
- [3] “STM32F4xx hardware development”, <http://www.st.com>
- [4] “Datasheet STM32F429”, <http://www.st.com>
- [5] “Description of STM32F4xx HAL drivers”, <http://www.st.com>
- [6] “Getting started with STM32CubeF4 firmware package for STM32F4xx series”,
<http://www.st.com>
- [7] “Getting started with STemWin Library”, <http://www.st.com>
- [8] “Developing Applications on STM32Cube with FatFs”, <http://www.st.com>
- [9] “Developing Applications on STM32Cube with RTOS”, <http://www.st.com>
- [10] “emwin Graphic Library with Graphical User Interface”,
<http://www.segger.com/emwin.html>

9. PHỤ LỤC

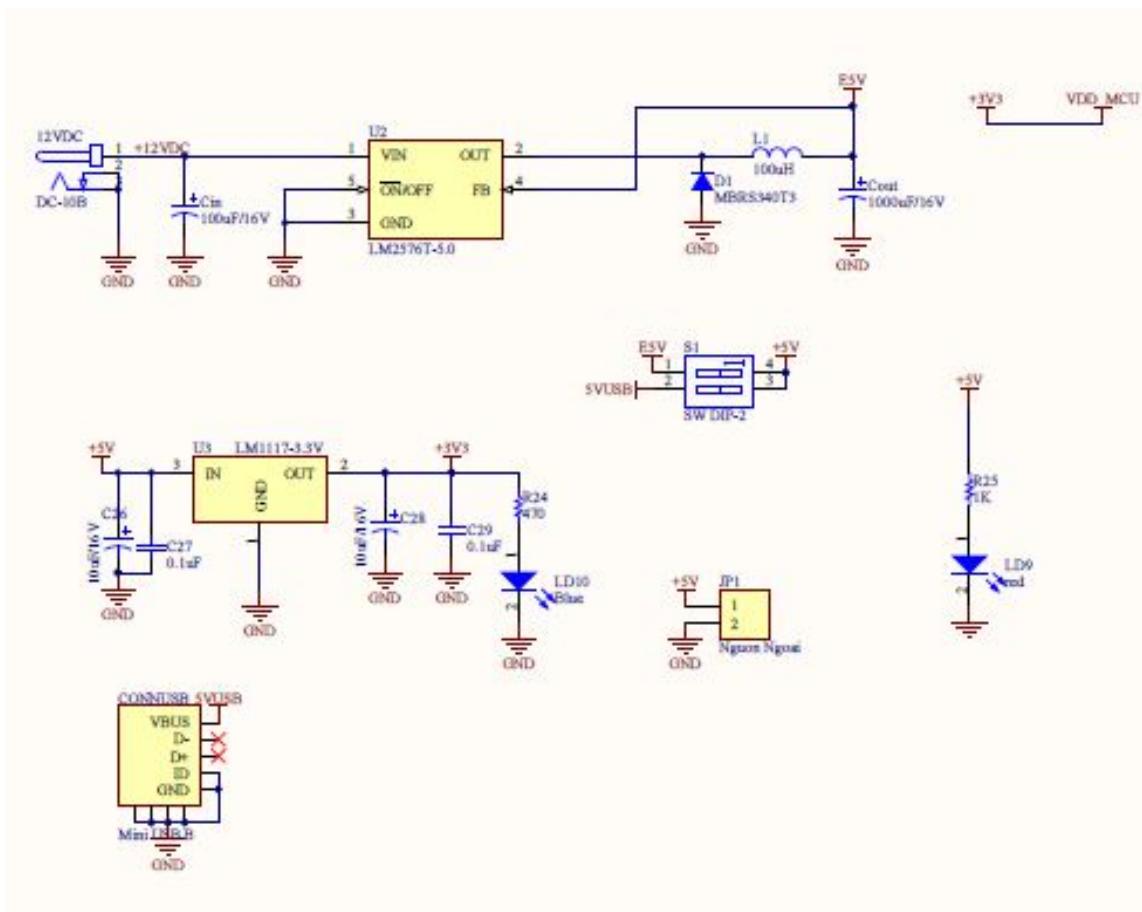
- Sơ đồ toàn mạch chi tiết:

- Sơ đồ mạch khói MCU:



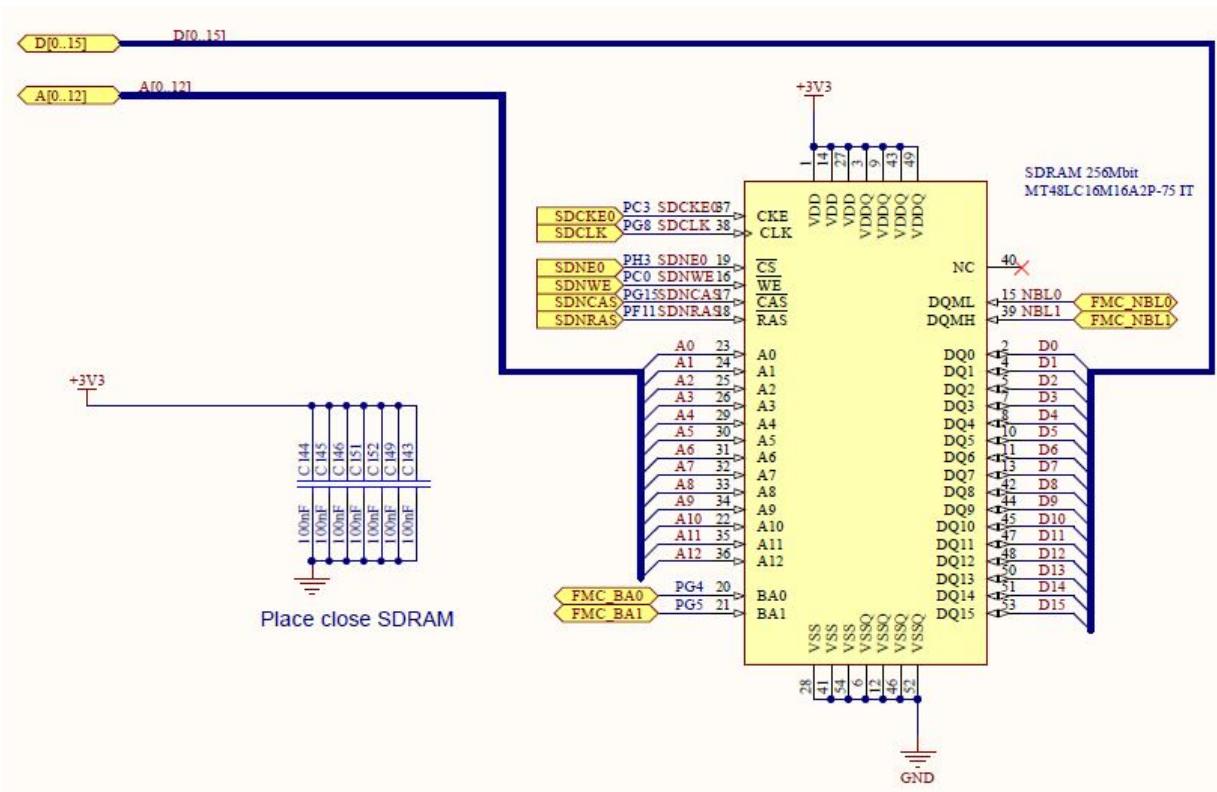
Hình 9.1 Sơ đồ mạch khói MCU

- Sơ đồ mạch khối nguồn:



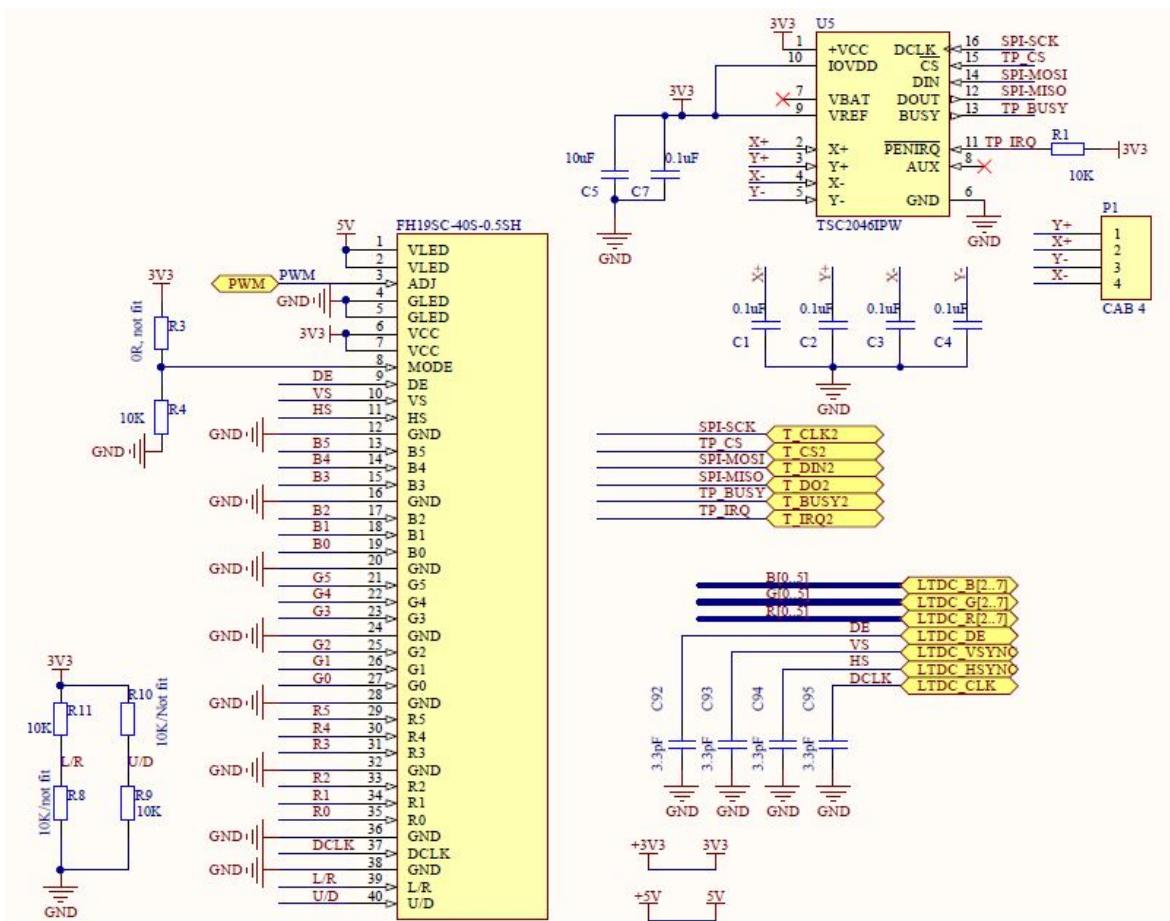
Hình 9.2 Sơ đồ nguyên lý khối nguồn

- Sơ đồ mạch khối bộ nhớ SDRAM:



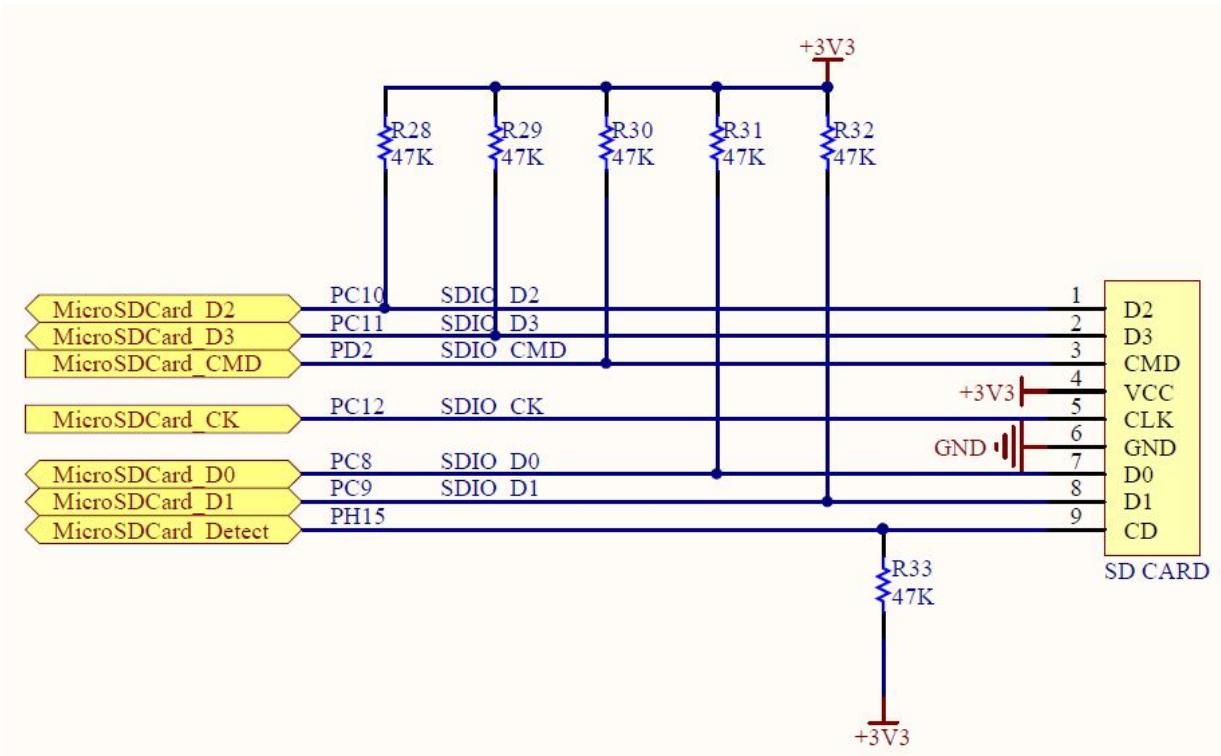
Hình 9.3 Sơ đồ nguyên lý khối bộ nhớ SDRAM

- Sơ đồ mạch khối LCD và Touch:



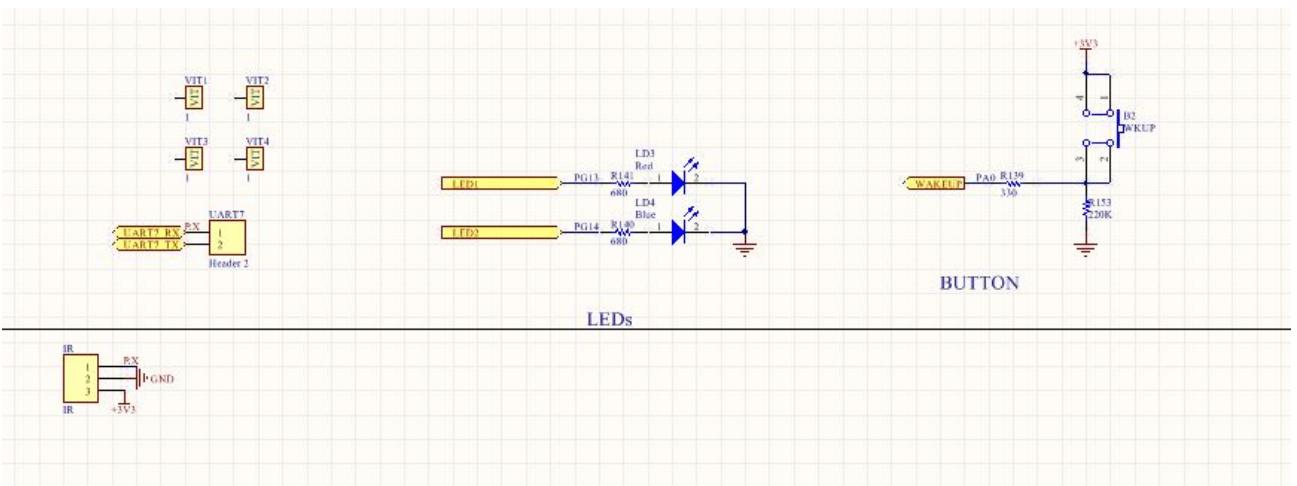
Hình 9.4 Sơ đồ nguyên lý khói LCD và Touch

- Sơ đồ mạch khói thẻ nhớ:



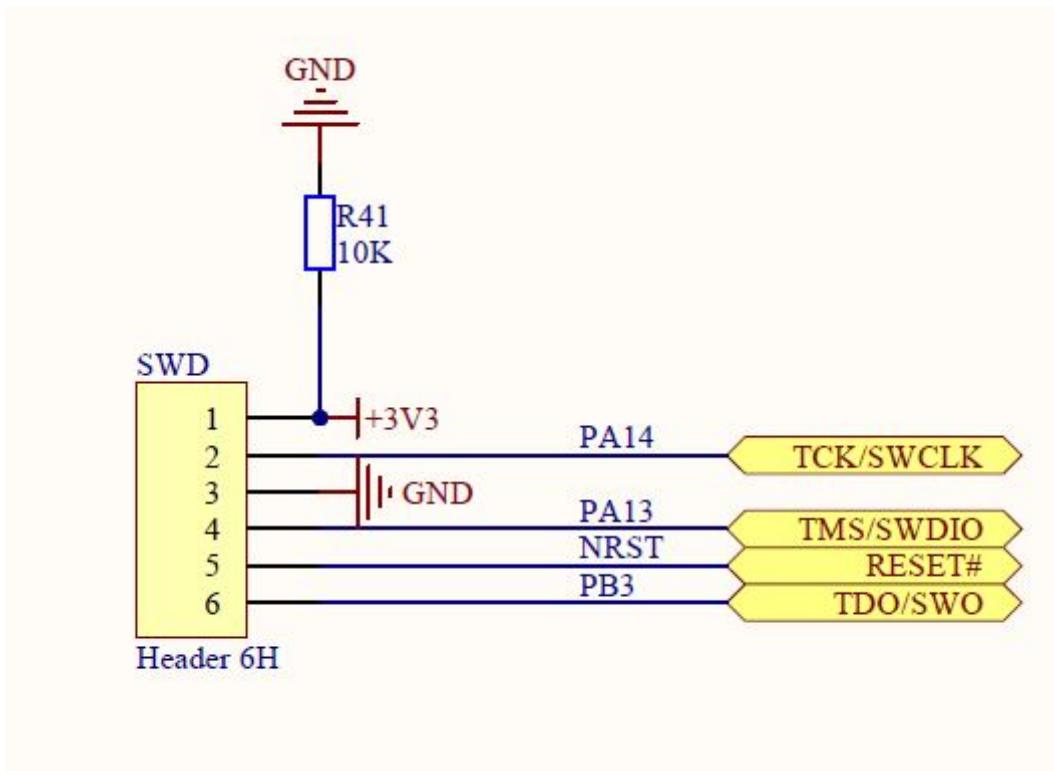
Hình 9.5 Sơ đồ nguyên lý khói thẻ nhớ

- Sơ đồ mạch khói ngoại vi I/O:



Hình 9.6 Sơ đồ nguyên lý khói ngoại vi I/O

- Sơ đồ mạch khối SWD:



Hình 9.7 Sơ đồ nguyên lý khối SWD