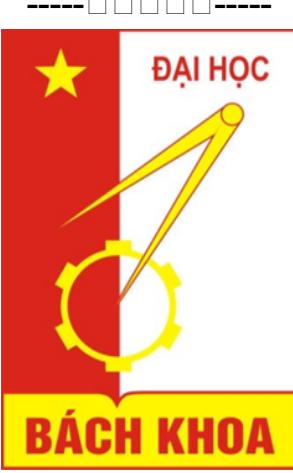


ĐẠI HỌC BÁCH KHOA HÀ NỘI  
TRƯỜNG ĐIỆN - ĐIỆN TỬ



BÁO CÁO KỸ

Đề tài: Điều khiển đèn giao  
STM32F103C8T6

Giảng viên hướng dẫn:

THUẬT VI XỬ LÝ

thông sử dụng

TS. Nguyễn Hoàng Dũng

Nhóm: 21

Sinh viên thực hiện: Nguyễn Thành Trung 20203915

Lớp: 137304

Hà Nội, 3 - 2023

## MỤC LỤC

MỤC LỤC	1
Danh mục Hình vẽ	3
Danh mục bảng biểu	5
Lời nói đầu	6
Lời cam đoan	7
PHÂN CHIA CÔNG VIỆC	8
Tóm tắt báo cáo	9
Chương 1. Tổng quan về đề tài	10
1. Đặt vấn đề	10
2. Giới thiệu thông quan về đề tài	10
2.1 Tổng quan vi xử lý ARM	10
2.2 Tổng quan về cấu trúc Arm Cotex M3	11
2.3 Giới thiệu về STM32 F103C8C6	13
2.4 Giới thiệu các phần mềm sử dụng	17
3. Phân tích yêu cầu thiết kế	22
3.1 Yêu cầu chức năng	22
3.2 Yêu cầu phi chức năng	23
4. Kết luận chương	24
Chương 2. Thiết kế hệ thống	25
1. Đặt vấn đề	25
2. Sơ đồ khái của hệ thống	27
2.1 Khối nguồn	28
2.2 Khối điều khiển	31
2.3 Khối hiển thị	36
2.4 Khối giải mã IC 7447	38
3. Cách thức điều khiển và hoạt động	39
3.1 Cách dùng IC 7447 điều khiển cho led đôi 7 thanh anode chung	39
3.2. Các phương thức giao thức sử dụng	40

4. Lập trình cho STM32F103C8T6	45
4.1 Xây dựng lưu đồ thuật toán	45
4.2 Triển khai code	46
4.3 Chạy mô phỏng trên Proteus	56
5. Tổng kết chương	57
Chương 3. Thử nghiệm và hoàn thiện	58
1. Vẽ mạch nguyên lý và mạch in trên Altium	58
1.1 Xây dựng schematic	58
1.2 Vẽ mạch in PCB	60
2. Hàn mạch	62
3. Chạy thử nghiệm mạch, kết quả	64
4. Đánh giá	66
4.1 Những điều đạt được	66
4.2 Những điều chưa làm được, hạn chế	66
5. Hướng phát triển	67
6. Tổng kết chương	68
KẾT LUẬN BÁO CÁO	69
TÀI LIỆU THAM KHẢO	70

## Danh mục Hình vẽ

### **Chương 1:**

Hình 1. 1 Minh họa vi xử lý ARM	8
Hình 1. 2 Sơ đồ khối ARM	10
Hình 1. 3 Sơ đồ khối ARM Cortex-M3	10
Hình 1. 4 STM32F103C8T6	13
Hình 1. 5 Công cụ STM32 CubeMX	15
Hình 1. 6 Cấu hình trên CubeMX	16
Hình 1. 7 Lập trình trên KeilC	16
Hình 1. 8 Hình minh họa Proteus	17
Hình 1. 9 Thiết kế mạch bằng Proteus	17
Hình 1. 10 Phần mềm Altium Designer	18
Hình 1. 11 Vẽ mạch PCB bằng Altium Designer	19

### **Chương 2:**

Hình 2. 1 Quy trình thiết kế	23
Hình 2. 2 Sơ đồ khối của hệ thống	24
Hình 2. 3 Nguồn 5V-2A	25
Hình 2. 4 IC hạ áp	26
Hình 2. 5 Sơ đồ mạch tham chiếu cho điện áp điều chỉnh	27
Hình 2. 6 Khối điều khiển STM32F103C8T6	28
Hình 2. 7 Sơ đồ khối STM32F103xx	29
Hình 2. 8 Sơ đồ minh họa các cổng kết của STM32F103xx	30
Hình 2. 9 KIT phát triển STM32F103C8T6 Blue Bill	31
Hình 2. 10 ST Link V2	32
Hình 2. 11 Đèn led 7 thanh	33
Hình 2. 12 Sơ đồ đèn led	34
Hình 2. 13 Đèn led đơn	34
Hình 2. 14 IC 7447	35
Hình 2. 15 Sơ đồ kết nối chân IC 7447 với led đôi 7 đoạn anode chung	36
Hình 2. 16 Bảng chuyển đổi BCD ra số thập phân	37
Hình 2. 17 Frame	38
Hình 2. 18 Start bit	38
Hình 2. 19 Parity bit	39
Hình 2. 20 Stop bit	40
Hình 2. 21 Sơ đồ khối SPI của STM32F103	41
Hình 2. 22 Sơ đồ lưu đồ thuật toán lập trình	42
Hình 2. 23 Bảng điều khiển phần mềm STM32CubeMX	43
Hình 2. 24 Sơ đồ cổng kết nối STM32F103C8Tx	44
Hình 2. 25 Sơ đồ minh họa trên phần mềm STM32CubeMX	44
Hình 2. 26 Mạch mô phỏng	52

### **Chương 3:**

Hình 3. 1 Khởi model điều khiển và khởi nguồn	55
Hình 3. 2 Khởi model hiển thị	56
Hình 3. 3 Mạch mô phỏng tông quan	56
Hình 3. 4 Layout 2 mặt 2D	57
Hình 3. 5 Layout mặt trên 3D	58
Hình 3. 6 Layout mặt dưới 3D	58
Hình 3. 7 Mặt trước và mặt sau của khối hiển thị	59
Hình 3. 8 Mặt sau của sản phẩm	60
Hình 3. 9 Mặt trước của sản phẩm	60
Hình 3. 10 Kết quả sau khi chạy thử lần 1	61
Hình 3. 11 Kết quả sau khi chạy thử lần 2	62

## **Danh mục bảng biểu**

Bảng 1: Bảng so sánh các vi xử lý	13
Bảng 2: Cấu hình chi tiết của STM32F103C8T6	15
Bảng 3 :Bảng công việc	23
Bảng 4 :Bảng mô tả chức năng IC hạ áp	27

## **Lời nói đầu**

Ngày nay cùng với sự phát triển đi lên của xã hội, các phương tiện tham gia giao thông cũng gia tăng không ngừng và hệ thống giao thông ngày càng phức tạp. Vì vậy để đảm bao giao thông được an toàn và thông suốt th việc sử dụng các hệ thống tín hiệu để điều khiển và phân luồng tại các nút giao thông là rất cần thiết Nhận thấy đây là vấn đề rất sát thực, với những kiến thức đã được trang bị trong quá trình học tập và nghiên cứu tại trường Đại học Bách Khoa Hà Nội. Chúng em đã lựa chọn đề tài "Thiết kế và mô phỏng hệ thống điều khiển đèn giao thông cho ngã tư "

Trong quá trình thực hiện đồ án chúng em đã nhận được sự chỉ bảo, hướng dẫn tận tình của các thầy cô trong khoa đặc biệt đó là sự chỉ bảo của thầy Chúng em xót trân thành cảm ơn sự chỉ bảo cho các thầy cô!

Trong khi thực hiện dễ ăn do kiến thức còn hạn chế cũng như chúng n china có nhiều điều kiện để đi khảo sát thực tế, với một khoảng thời gian ngắn thực hiện, do vậy mà đồ ăn của chúng em còn nhiều thiếu sót mong các thầy cỏ đóng góp và bô xung ý kiến để đồ ăn của chúng em được hoàn thiện hot.

Chúng em xin chân thành cảm ơn!

## **Lời cam đoan**

Em, cam đoan rằng báo cáo "Thiết kế đèn giao thông bằng STM32F103C8T6" này là công trình nghiên cứu của tôi. Tất cả những thông tin và kết quả được trình bày trong báo cáo này đều là trung thực và chính xác dựa trên nỗ lực nghiêm túc của tôi trong quá trình nghiên cứu và thực hiện.

Em xác nhận rằng tôi đã tham khảo tất cả các nguồn tài liệu một cách hợp lý và đúng đắn. Các kết quả và nhận xét trong báo cáo này là do tôi đánh giá và phân tích một cách khách quan, không bị ảnh hưởng bởi bất kỳ thế lực nào.

Ngoài ra, chúng em cũng cam đoan rằng tôi đã tuân thủ đầy đủ các quy định về trích dẫn và tham khảo tài liệu. Tất cả các nguồn tài liệu được sử dụng trong báo cáo này đều được liệt kê đầy đủ và chính xác trong phần tài liệu tham khảo.

Chúng em hy vọng báo cáo này sẽ có ích cho các nhà nghiên cứu và kỹ sư trong lĩnh vực thiết kế đèn giao thông.

## **Tóm tắt báo cáo**

Báo cáo kỹ thuật vi xử lý tập trung vào trình bày về việc tìm hiểu về vi xử lý STM32. Trong báo cáo này em sẽ trình bày về ứng dụng của STM32F103C8T6 trong việc hoàn thành sản phẩm “Thiết kế đèn giao thông bằng vi điều khiển STM32F103C8T6”

Báo cáo gồm có 3 chương:

Chương 1 : Tổng quan chung về đề tài

- Đặt vấn đề
- Đิ giải quyết vấn đề
- Kết luận chung

Chương 2 : Thiết kế hệ thống

- Đặt vấn đề
- Đิ giải quyết vấn đề :Xây dựng quy trình thiết kế
- Kết luận chung

Chương 3: Thủ nghiệm và hoàn thiện

- Đặt vấn đề
- Đิ giải quyết vấn đề :Xây dựng quy trình thiết kế
- Kết luận chung

# Chương 1. Tổng quan về đề tài

## 1. Đặt vấn đề

Điều khiển đèn giao thông là một trong những ứng dụng điển hình của hệ thống nhúng (embedded systems). Trong thời đại công nghệ 4.0, các hệ thống nhúng như vậy ngày càng được sử dụng rộng rãi để giải quyết các vấn đề trong cuộc sống hàng ngày.

Việc sử dụng hệ thống nhúng trong điều khiển đèn giao thông có nhiều ưu điểm, như làm giảm tình trạng ùn tắc giao thông, giảm nguy cơ tai nạn, tiết kiệm năng lượng và chi phí, cải thiện chất lượng cuộc sống cho người dân.

Trong đề tài này, chúng ta sẽ đề cập đến việc thiết kế và triển khai một hệ thống điều khiển đèn giao thông sử dụng vi điều khiển STM32F103C8T6.

Trong chương I này, chúng ta sẽ đi tìm hiểu tổng quan các khái niệm liên quan đến vi xử lý được dùng trong đề tài lần này của nhóm cụ thể như: vi xử lý Arm, Arm cortex M3, STM32F103C8T6. Cùng với đó là các công cụ, phần mềm hỗ trợ mà nhóm sử dụng để lập trình cho STM32F1C8T6 này.

## 2. Giới thiệu tổng quan về đề tài

### 2.1 Tổng quan vi xử lý ARM



Hình 1. 1 Minh họa vi xử lý ARM

ARM viết tắt của Advanced RISC Machine (trước đây là Acorn RISC Machine) là bộ xử lý dựa trên kiến trúc RISC (Reduced Instruction Set Computer – Máy tính có tập lệnh đơn giản hóa), được phát triển bởi Arm Holdings, Ltd.

Các bộ xử lý có kiến trúc RISC thường yêu cầu ít bóng bán dẫn hơn các bộ xử lý có kiến trúc điện toán CISC (như bộ xử lý x86 có trong hầu hết các máy tính cá nhân), loại bỏ các lệnh không cần thiết, thực hiện nhiều lệnh hơn trên mỗi giây (MIPS) và tối ưu hóa các đường dẫn giúp giảm mức tiêu thụ điện năng, tiết kiệm diện tích, ít tản nhiệt, cung cấp mức hiệu suất vượt trội, lý tưởng cho các thiết bị di động nhỏ gọn như điện thoại thông minh, máy tính xách tay, máy tính bảng.

Bộ vi xử lý ARM được thiết kế để hoạt động hiệu năng nhất có thể, chỉ chấp nhận các lệnh có thể được thực hiện trong một chu kỳ bộ nhớ. Quá trình phổ biến đối với CPU là tìm nạp, giải mã và thực thi lệnh.

ARM Holdings định kỳ phát hành bản cập nhật cho kiến trúc. Các phiên bản kiến trúc ARMv3 đến ARMv7 hỗ trợ không gian địa chỉ 32 bit và số học 32 bit, hầu hết các kiến trúc đều có các hướng dẫn có độ dài cố định 32 bit. Phiên bản Thumb hỗ trợ một tập lệnh có độ dài thay đổi, cung cấp cả hai lệnh 32 và 16 bit để cải thiện mật độ mã. Một số lỗi cũ hơn cũng có thể cung cấp thực thi phần cứng cho mã byte Java. Được phát hành vào năm 2011, kiến trúc ARMv8-A đã thêm hỗ trợ cho không gian địa chỉ 64 bit và số học 64 bit với tập lệnh có độ dài cố định 32 bit mới.

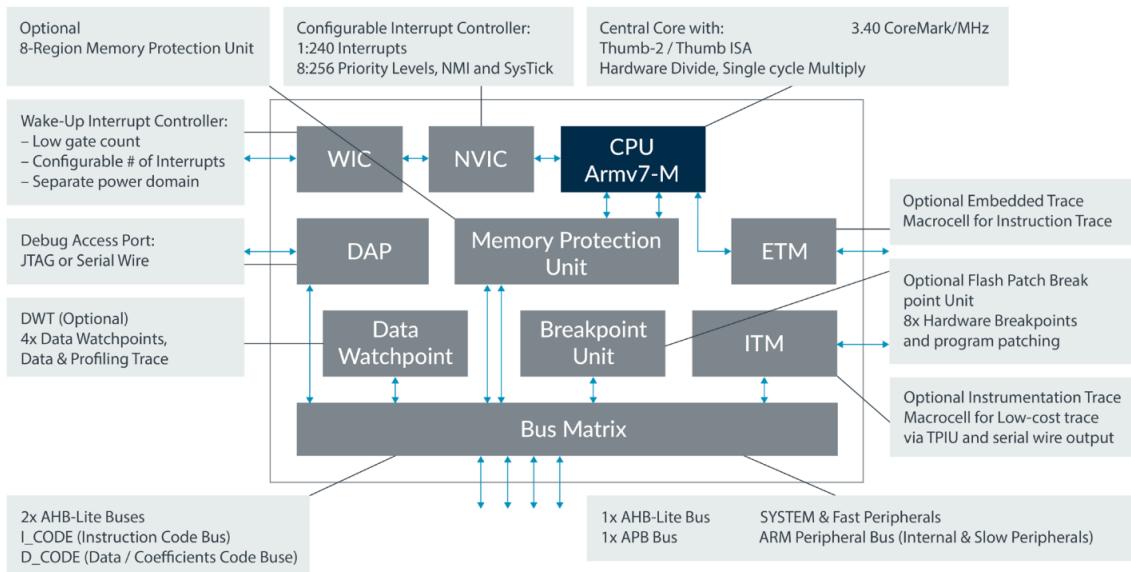
## 2.2 Tổng quan về cấu trúc Arm Cotex M3

Vi xử lý lõi ARM Cortex dựa trên 3 cấu hình của kiến trúc ARMv7

- Cấu hình A: cho các ứng dụng Application, yêu cầu cao chạy trên các hệ điều hành mở và phức tạp như Linux, Android...
- Cấu hình R: cho các ứng dụng thời gian thực Real Time
- Cấu hình M: cho các ứng dụng vi điều khiển Microcontroller

Bộ vi xử lý ARM Cortex-M3 là bộ vi xử lý ARM đầu tiên dựa trên kiến trúc ARMv7-M và được thiết kế đặc biệt để đạt được hiệu suất cao trong các ứng dụng nhúng cần tiết kiệm năng lượng và chi phí, chẳng hạn như các vi điều khiển, hệ thống cơ ô tô, hệ thống kiểm soát công nghiệp và hệ thống mạng không dây.Thêm vào đó là việc lập trình được đơn giản hóa đáng kể giúp kiến trúc ARM trở thành một lựa chọn tốt cho ngay cả những ứng dụng đơn giản nhất.

## Block Diagram

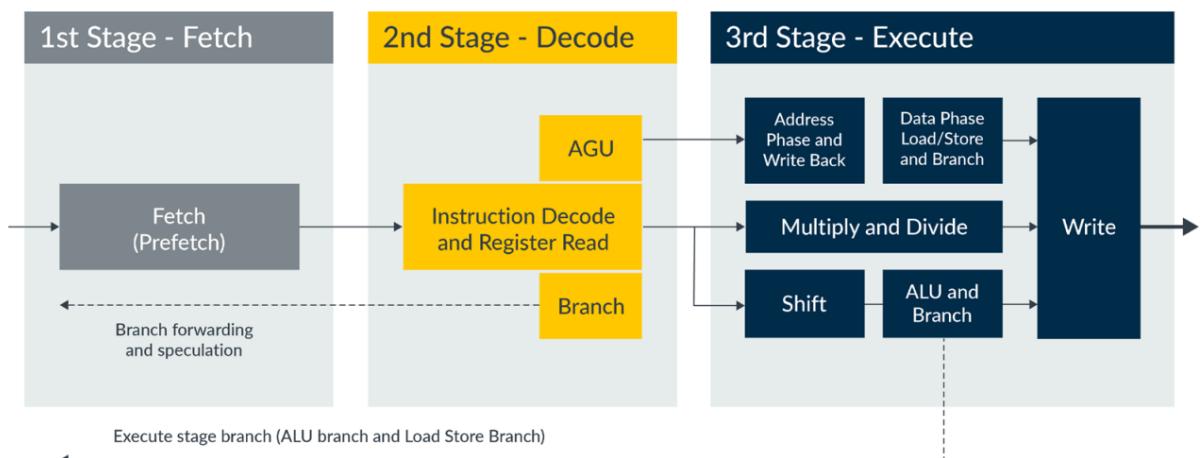


Hình 1. 2 Sơ đồ khói ARM

Bộ vi xử lý ARM Cortex-M3 dựa trên kiến trúc ARMv7-M có cấu trúc thứ bậc. Nó tích hợp lõi xử lý trung tâm, với các thiết bị ngoại vi hệ thống tiên tiến để tạo ra các khả năng như kiểm soát ngắt, bảo vệ bộ nhớ, gỡ lỗi và theo dõi hệ thống.

ARM Cortex M3 dựa trên cấu trúc Harvard, được đặc trưng bằng sự tách biệt giữa vùng nhớ dữ liệu và chương trình. Vì có thể đọc cùng lúc lệnh và dữ liệu từ bộ nhớ, bộ vi xử lý ARM Cortex-M3 có thể thực hiện nhiều hoạt động song song, tăng tốc thực thi ứng dụng.

## Cortex-M3 Pipeline



Hình 1. 3 Sơ đồ khói ARM Cortex-M3

Lõi ARM Cortex có cấu trúc đường ống gồm 3 tầng: Instruction Fetch, Instruction Decode và Instruction Execute. Khi gặp một lệnh nhánh, tầng decode chứa một chỉ thị nạp lệnh suy đoán có thể dẫn đến việc thực thi nhanh hơn. Bộ xử lý nạp lệnh dự định rẽ nhánh trong giai đoạn giải mã. Sau đó, trong giai đoạn thực thi, việc rẽ nhánh được giải quyết và bộ vi xử lý sẽ phân tích xem đâu là lệnh thực thi kế tiếp. Nếu việc rẽ nhánh không được chọn thì lệnh tiếp theo đã sẵn sàng. Còn nếu việc rẽ nhánh được chọn thì lệnh rẽ nhánh đó cũng đã sẵn sàng ngay lập tức, hạn chế thời gian rẽ chỉ còn một chu kỳ.

Lõi Cortex-M3 chứa một bộ giải mã cho tập lệnh Thumb truyền thống và Thumb-2 mới, một bộ phân chia logic ALU(arithmetic logic unit) tiên tiến hỗ trợ nhân chia phần cứng, điều khiển logic, và các giao tiếp với các thành phần khác của bộ xử lý.

Bộ vi xử lý Cortex-M3 là một bộ vi xử lý 32-bit, với độ rộng của đường dẫn dữ liệu 32 bit, các dải thanh ghi và giao tiếp bộ nhớ. Có 13 thanh ghi đa dụng, hai con trỏ ngăn xếp, một thanh ghi liên kết, một bộ đếm chương trình và một số thanh ghi đặc biệt trong đó có một thanh ghi trạng thái chương trình.

### 2.3 Giới thiệu về STM32 F103C8C6

#### 2.3.1 So sánh các chip tiêu biểu thuộc lõi Arm Cotex M3

STM32F1 là dòng chip lõi Arm Cotex M3. Tiêu biểu các dòng trên thị trường như: STM32F103C8T6, STM32F103C6T6, STM32F103C6R6. Dưới đây là bảng so sánh các thông số, tính năng tổng quan về 3 loại trên.

Tính năng	STM32F103C8T6	STM32F103C6T6	STM32F103C6R6
Kiến trúc	ARM Cortex-M3	ARM Cortex-M3	ARM Cortex-M3
Tốc độ xử lý	72 MHz	72 MHz	72 MHz
Flash	64 KB	32 KB	32 KB
RAM	20 KB	10 KB	10 KB
Bộ chuyển đổi ADC	12-bit, 10 channel	12-bit, 10 channel	12-bit, 10 channel
Kết nối	USART, SPI, I2C, CAN, USB	USART, SPI, I2C, CAN, USB	USART, SPI, I2C, CAN, USB

Tính năng	<b>STM32F103C8T6</b>	<b>STM32F103C6T6</b>	<b>STM32F103C6R6</b>
Điện áp hoạt động	2.0-3.6V	2.0-3.6V	2.0-3.6V
Giao diện	LQFP48, LQFP64	LQFP48, LQFP64	LQFP48, LQFP64

Bảng 1: Bảng so sánh các vi xử lý

Như vậy, STM32F103C8T6, STM32F103C6T6 và STM32F103C6R6 có nhiều tính năng chung do cùng thuộc kiến trúc ARM Cortex-M3 và tốc độ xử lý 72MHz. Tuy nhiên, có sự khác biệt về bộ nhớ Flash và RAM giữa chúng. STM32F103C8T6 có bộ nhớ Flash và RAM lớn hơn so với STM32F103C6T6 và STM32F103C6R6.

Do STM32F103C8T6 thông dụng, được sử dụng rộng rãi, có bộ nhớ Flash và RAM lớn hơn nên chúng em quyết định sử dụng chip STM32F103C8T6 để làm bài tập lớn môn học.

### 2.3.2 Chi tiết về STM32F103C8T6

Vi điều khiển STM32F103C8T6 tích hợp nhiều tính năng như bộ chuyển đổi ADC và DAC, các kênh DMA, các giao diện USB, UART, SPI, I2C và các tính năng khác để hỗ trợ các ứng dụng điều khiển, đo lường và giao tiếp.

Vi điều khiển này cũng có tính năng bảo mật như bộ xử lý mã hóa phần cứng, tạo chữ ký số và kiểm tra mã, giúp bảo vệ dữ liệu và đảm bảo tính bảo mật cho các ứng dụng.

Với khả năng xử lý mạnh mẽ, độ ổn định cao và tính linh hoạt trong thiết kế, vi điều khiển STM32F103C8T6 là một trong những lựa chọn phổ biến cho các dự án nhúng và các ứng dụng điều khiển đòi hỏi tính ổn định, độ chính xác cao và tốc độ xử lý nhanh.



Hình 1. 4 STM32F103C8T6

Dưới đây là phần bảng cấu hình chi tiết của STM32F103C8T6:

Bộ nhớ	<ul style="list-style-type: none"><li>- 64 kbytes bộ nhớ Flash(bộ nhớ lập trình).</li><li>- 20kbytes SRAM</li></ul>
Clock, reset và quản lý nguồn	<ul style="list-style-type: none"><li>- Điện áp hoạt động 2.0V -&gt; 3.6V.</li><li>- Power on reset(POR), Power down reset(PDR) và programmable voltage detector (PWD).</li><li>- Sử dụng thạch anh ngoài từ 4Mhz -&gt; 20Mhz.</li></ul>

	<ul style="list-style-type: none"> <li>- Thạch anh nội dùng dao động RC ở mode 8Mhz hoặc 40khz.</li> <li>- Sử dụng thạch anh ngoài 32.768khz được sử dụng cho RTC.</li> </ul>
2 bộ ADC 12 bit với 9 kênh cho mỗi bộ	<ul style="list-style-type: none"> <li>- Khoảng giá trị chuyển đổi từ 0 – 3.6V.</li> <li>- Lấy mẫu nhiều kênh hoặc 1 kênh.</li> <li>- Có cảm biến nhiệt độ nội.</li> </ul>
DMA	<ul style="list-style-type: none"> <li>- 7 kênh DMA.</li> <li>- Hỗ trợ DMA cho ADC, I2C, SPI, UART.</li> </ul>
7 timer	<ul style="list-style-type: none"> <li>- 3 timer 16 bit hỗ trợ các mode IC/OC/PWM.</li> <li>- 1 timer 16 bit hỗ trợ để điều khiển động cơ với các mode bảo vệ như ngắt input, dead-time..</li> <li>- 2 watchdog timer dùng để bảo vệ và kiểm tra lỗi.</li> </ul>
Hỗ trợ 9 kênh giao tiếp bao gồm	<ul style="list-style-type: none"> <li>- 2 bộ I2C(SMBus/PMBus).</li> <li>- 3 bộ USART(ISO 7816 interface, LIN, IrDA capability, modem control).</li> <li>- 2 SPIs (18 Mbit/s).</li> <li>- 1 bộ CAN interface (2.0B Active)</li> </ul>
USB 2.0 full-speed interface	<ul style="list-style-type: none"> <li>- Kiểm tra lỗi CRC và 96-bit ID.</li> </ul>

Bảng 2: Cấu hình chi tiết của STM32F103C8T6

## 2.4 Giới thiệu các phần mềm sử dụng

### 2.4.1 Phần mềm STM32CubeMX và Keil C

STMicroelectronics đã giới thiệu một công cụ có tên STM32CubeMX, tạo code cơ bản theo các thiết bị ngoại vi và board STM32 được chọn. Vì vậy chúng ta không cần phải lo lắng về việc code hóa cho các chương trình điều khiển và thiết bị ngoại vi cơ bản. Hơn nữa code này được tạo có thể sử dụng trong Keil uVision để chỉnh sửa theo yêu cầu. Và cuối cùng code được ghi vào STM32 bằng lập trình ST-Link từ STMicroelectronics.



Hình 1. 5 Công cụ STM32 CubeMX

Công cụ phần mềm này giúp cho công việc phát triển dễ dàng bằng cách giảm giai đoạn phát triển thời gian và chi phí. STM32 Cube gồm STM32CubeMX là một công cụ đồ họa cho phép cấu hình rất dễ dàng các bộ vi điều khiển và bộ vi xử lý STM32, cũng như tạo mã C khởi tạo tương ứng cho lõi Arm Cortex. Code C đó có thể sử dụng trong các môi trường phát triển khác nhau như Keil uVision, GCC, IAR,...

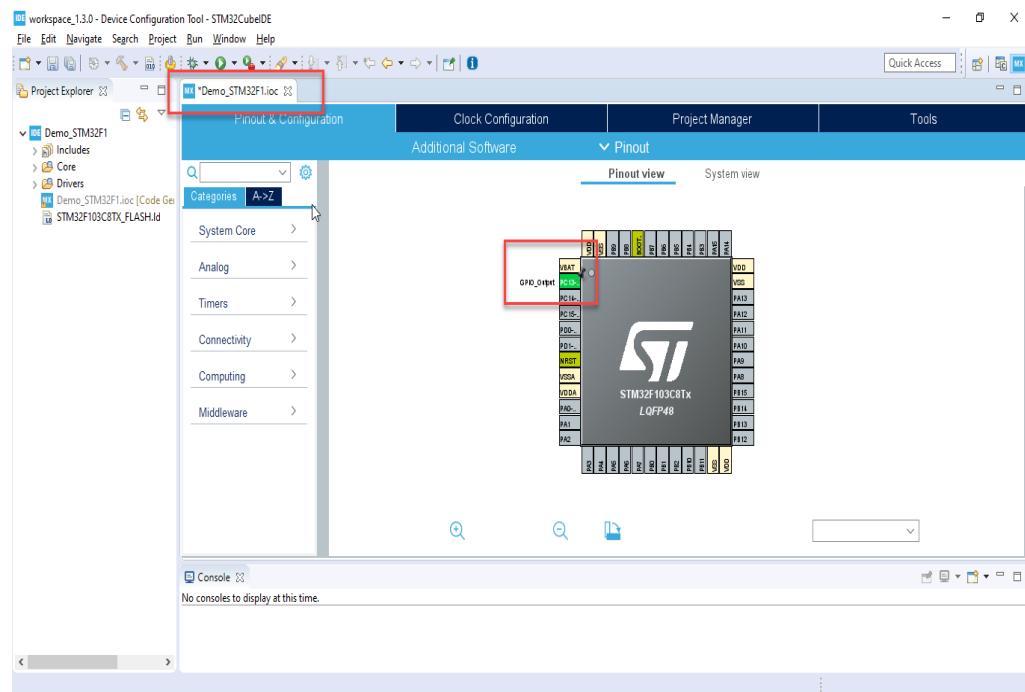
STM32CubeMX có các tính năng sau:

- Chân gõ lỗi
- Trợ giúp thiết lập xung
- Một nguồn
- Một tiện ích thực hiện cấu hình ngoại vi MCU như chân GPIO, USART, v.v.
- Một tiện ích thực hiện cấu hình ngoại vi MCU cho các ngăn xếp phần mềm trung gian như USB, TCP/IP, v.v

Ngày nay trên thị trường có khá nhiều trình biên dịch ngôn ngữ C cho các dòng vi điều khiển, IAR, Keil C ... Những phần mềm này được gọi là môi trường phát triển tích hợp (IDE: viết tắt của Integrated Development Environment). Chúng đóng vai trò như là trình soạn thảo ngôn ngữ C, assembly, cũng như là trình biên dịch, hỗ trợ debug-phát hiện

lỗi và sửa các câu lệnh vừa được viết ra. Ngoài ra chúng cũng hỗ trợ biên dịch những câu lệnh đã được viết ra bởi người sử dụng thành file hex qua đó nạp vào các dòng vi xử lý. Trong đó Keil C là một phần mềm chuyên dụng để tạo ra một môi trường lập trình hỗ trợ cho nhiều dòng vi xử lý từ ARM, AVR, 8051, PIC,.. với 2 ngôn ngữ chủ yếu là C và assembly. Keil C giúp người dùng soạn thảo và biên dịch chương trình C hay assembly thành ngôn ngữ máy để nạp vào vi điều khiển giúp tương tác giữa vi điều khiển và người lập trình.

Quy trình sẽ là cấu hình cơ bản trên Cube MX:



Hình 1. 6 Cấu hình trên CubeMX

Sau đó lập trình nâng cao trên KeilC:

```

91     MX_GPIO_Init();
92
93     /* USER CODE BEGIN 2 */
94
95     /* USER CODE END 2 */
96
97     /* Infinite loop */
98     /* USER CODE BEGIN WHILE */
99     while (1)
100    {
101        HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
102        HAL_Delay(1000);
103    }
104    /* USER CODE END 3 */
105
106 }
107
108 /**
109 */
110 void SystemClock_Config(void)

```

Build Output

```

*** Using Compiler 'V5.06 update 1 (build 61)', folder: 'C:\Keil_v5\ARM\ARMCC\Bin'
Build target 'test'
"test\test.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:03

```

Hình 1. 7 Lập trình trên KeilC

#### 2.4.2 Giới thiệu về Proteus

Proteus là một phần mềm mô phỏng hệ thống điện tử và vi điều khiển, được phát triển bởi Labcenter Electronics. Phần mềm này được sử dụng rộng rãi trong lĩnh vực thiết kế điện tử, điều khiển tự động và hệ thống nhúng.

Proteus cung cấp một môi trường mô phỏng cho các thiết bị điện tử và vi điều khiển. Nó cho phép người dùng thiết kế mạch điện tử, mô phỏng, kiểm tra và phát triển các ứng dụng điều khiển nhúng. Proteus cũng cung cấp một thư viện rộng lớn các linh kiện điện tử và vi điều khiển, bao gồm các vi điều khiển phổ biến của các hãng như Atmel, Microchip, STMicroelectronics, NXP, Texas Instruments, và nhiều hãng khác.

Ngoài ra, Proteus cũng cung cấp các công cụ mô phỏng và phân tích tín hiệu như oscilloscope, logic analyzer, spectrum analyzer, và các công cụ khác để hỗ trợ người dùng kiểm tra và phân tích các tín hiệu điện tử.

Proteus cũng hỗ trợ việc thiết kế các mô-đun vi điều khiển nhúng và các ứng dụng đa nhiệm với các tính năng như debug, simulation và test.

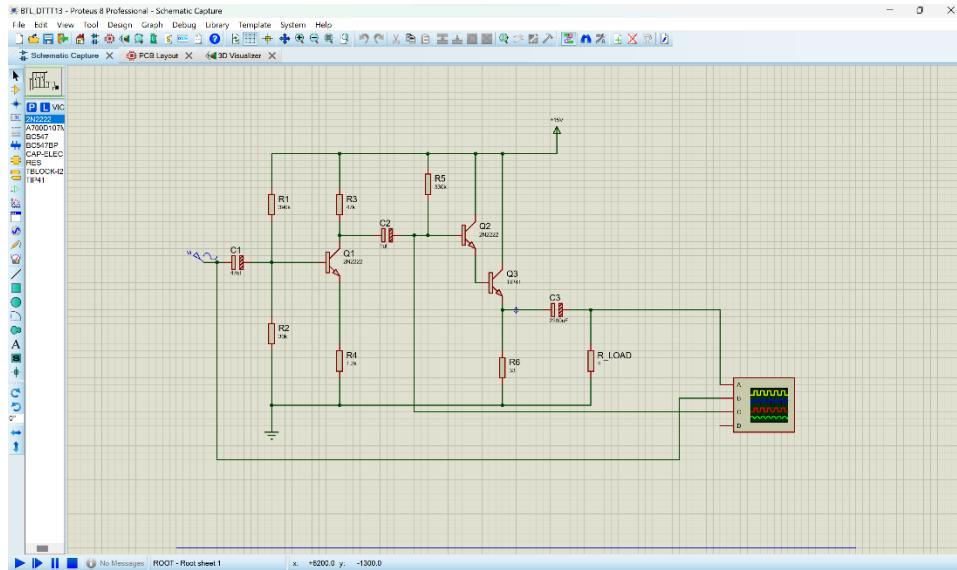


## PROTEUS

Hình 1. 8 Hình minh họa Proteus

Các  
chính sủ  
Proteus  
gồm:

bước  
dụng  
bao



- Bước 1: Khởi động chương trình Proteus Professional
- Bước 2: Mở chương trình ISIS Professional
- Bước 3: Lấy các linh kiện từ thư viện của Proteus
- Bước 4: Đưa linh kiện ra ngoài màn hình thiết kế
- Bước 5: Thay đổi thông số kỹ thuật của linh kiện điện tử cho phù hợp
- Bước 6: Bố trí, sắp xếp lại linh kiện cho hợp lý
- Bước 7: Nối dây
- Bước 8: Kiểm tra sơ đồ mạch nguyên lý

### 2.4.3 Giới thiệu về Altium

Altium là một phần mềm thiết kế điện tử và PCB (Printed Circuit Board) do công ty Altium Limited phát triển. Nó cung cấp một môi trường tích hợp để thiết kế, mô phỏng, và chế tạo các thiết bị điện tử và PCB.

Phần mềm Altium cung cấp một giao diện đồ họa trực quan và dễ sử dụng để thiết kế các mạch điện tử và PCB. Nó cung cấp các tính năng như trình soạn thảo mạch, mô phỏng mạch, tạo mẫu, định nghĩa các ràng buộc PCB, phân bổ địa chỉ, phân tích điện dung, tính toán trở, và nhiều tính năng khác.

Ngoài ra, Altium còn cung cấp các tính năng phân tích và tối ưu mạch, cho phép người dùng kiểm tra tính đúng đắn của mạch, độ ổn định và hiệu suất của các linh kiện và tối ưu hóa mạch để đạt được kết quả tối ưu.

Altium cũng hỗ trợ các công nghệ nhúng hiện đại, cho phép người dùng thiết kế các thiết bị nhúng phức tạp với các tính năng như tích hợp Wi-Fi, Bluetooth, giao thức truyền thông TCP/IP và các thiết bị IoT (Internet of Things).

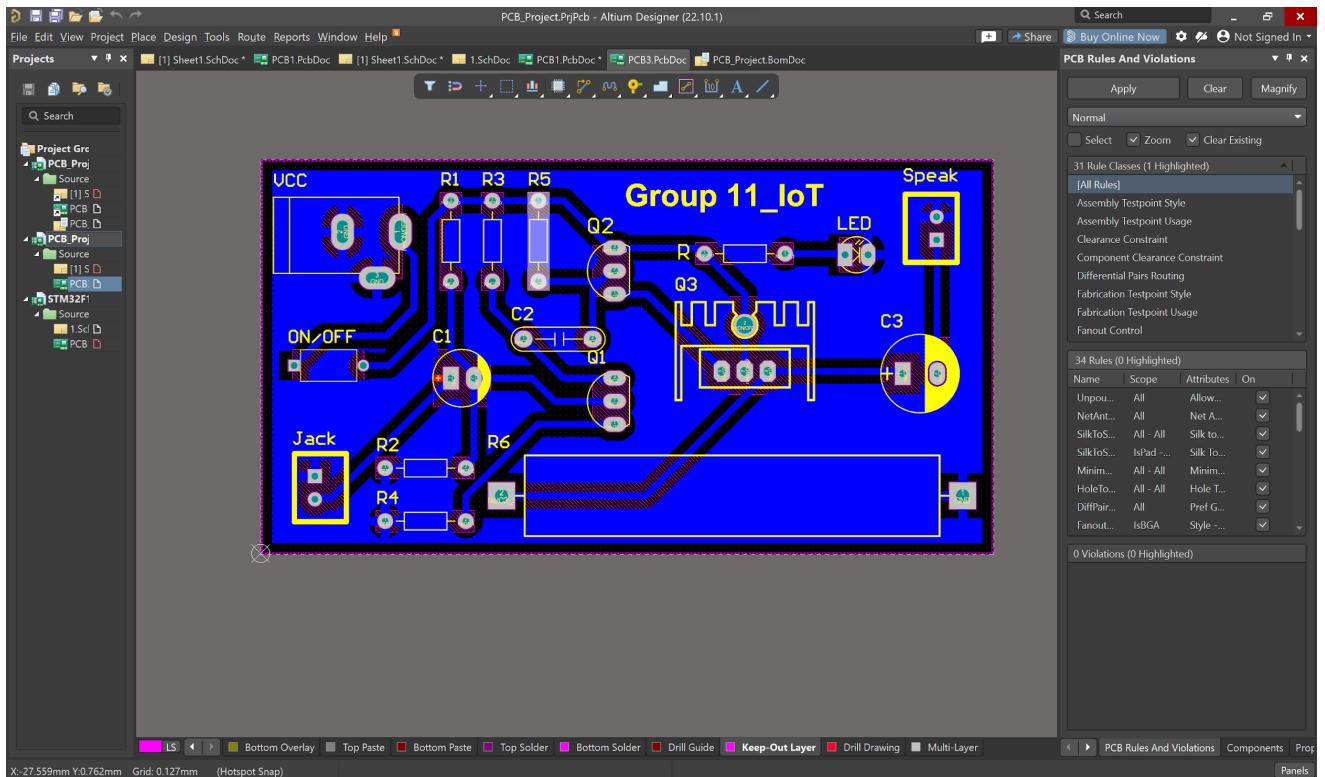


Hình 1. 10 Phần mềm Altium Designer

Việc thiết kế mạch điện tử trên phần mềm altium designer có thể được tóm tắt gồm các bước như sau:

- Đặt ra các yêu cầu bài toán.**
- Lựa chọn linh kiện.**
- Thiết kế mạch nguyên lý.**
- Lựa chọn các chân linh kiện để chuyển sang mạch in Update mạch nguyên lý sang mạch in.**
- Lựa chọn kích thước mạch in Sắp xếp các vị trí các loại linh kiện như điện trở, tụ điện, IC...**
- Đặt kích thước các loại dây nối.**

- Đi dây trên mạch.**
- Kiểm tra toàn mạch.**



Hình 1. 11 Vẽ mạch PCB bằng Altium Designer

### 3. Phân tích yêu cầu thiết kế

#### 3.1 Yêu cầu chức năng

**Chức năng:** Đèn giao thông xác định thời gian hiển thị đèn xanh, đèn đỏ, đèn vàng bằng cách đếm ngược thời gian thông qua led đôi 7 thanh.

Đầu vào: Nguồn 5V DC-1A

Đầu ra: Hiển thị số giây đếm ngược trên led đôi 7 thanh và điều khiển đèn led đơn xanh, đỏ, vàng.

Quan hệ giữa đầu vào và đầu ra: Nguồn vào cung cấp năng lượng để mạch hoạt động, sau đó tín hiệu điều khiển được truyền từ STM32F103C8T6 qua IC giải mã 7447 đến các led đôi 7 thanh và led đơn xanh, đỏ, vàng và hiển thị chúng.

#### 3.2 Yêu cầu phi chức năng

- Năng lượng:
  - Điện áp: Vi điều khiển STM32F103C8T6 có mức tiêu thụ điện áp từ 2.0V đến 3.6V.
  - Dòng điện: Vi điều khiển STM32F103C8T6 có mức tiêu thụ năng lượng trung bình khoảng 2 mA ở chế độ chờ (standby), và tối đa khoảng 70 mA ở chế độ hoạt động đầy đủ với tần số hoạt động 72 MHz.
- Ngoại quan cơ khí
  - Sản phẩm như một mô hình ngã tư đèn giao thông thu nhỏ.
  - Vật liệu : Thường sản xuất từ silic và các hợp chất bán dẫn.
- Hiệu năng:
  - Vi điều khiển STM32F103C8T6 có khả năng tính toán xử lý với tốc độ cao, với các tính năng như bộ nhớ, giao tiếp nhiều loại, và các tính năng đặc biệt khác.
- Tiêu chuẩn cần tuân theo:
  - Chuẩn an toàn: Vi điều khiển STM32F103C8T6 có thể tuân theo chuẩn an toàn IEC 61508 SIL (Safety Integrity Level) 1, giúp nó được sử dụng trong các ứng dụng yêu cầu độ an toàn cao như trong các thiết bị y tế, xe cộ, máy bay,...
  - Giao thức kết nối: Vi điều khiển STM32F103C8T6 hỗ trợ nhiều giao thức kết nối như SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), USART (Universal asynchronous receiver transmitter), USB (Universal Serial Bus), CAN (Controller Area Network), Ethernet và các giao thức truyền thông không dây.
- Môi trường hoạt động:
  - Nhiệt độ hoạt động từ -10°C đến 50°C.
- Giá cả & chi phí:
  - 60000 đồng cho chip STM32 và 5000 đồng cho mỗi led đôi 7 thanh.

#### **4. Kết luận chương**

Qua chương 1, chúng ta đã tìm hiểu sơ qua tổng quan về các thông số, cấu trúc cơ bản về dòng chip Arm, Arm Cotex M3, STM32, STM32F103C8T6,...

Bên cạnh đó là giới thiệu về các phần mềm hỗ trợ trong việc hoàn thiện sản phẩm bài tập lớn của chúng em như: phần mềm viết code STM32CubeMX kết hợp Keil C, phần mềm mô phỏng Proteus và phần mềm vẽ mạch in PCB Altium.

Cuối cùng, là xác định về yêu cầu chỉ tiêu kỹ thuật của sản phẩm điều khiển đèn giao thông sử dụng STM32F103C8T6.

## Chương 2. Thiết kế hệ thống

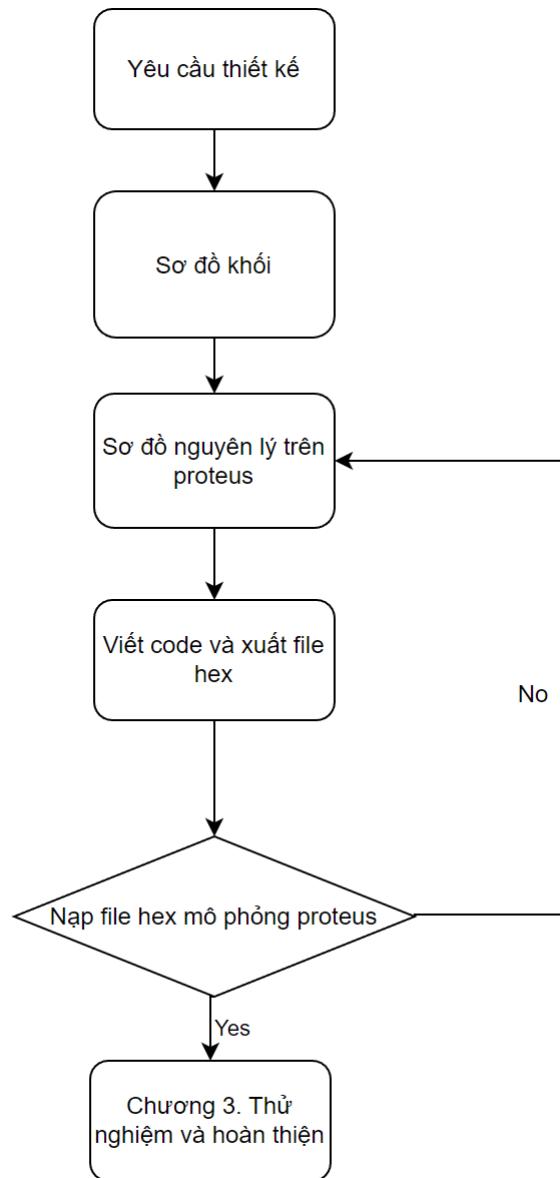
### 1. Đặt vấn đề

Sau khi xác định tổng quan về đề tài, về các chỉ tiêu, yêu cầu thiết kế của hệ thống, chúng em tiến hành đi vào xây dựng quy trình thiết kế sản phẩm và phân chia ra các giai đoạn công việc.

Nội dung	Công việc	Thời gian	Hoàn thành
Tìm hiểu đề tài	Xác định đề tài, yêu cầu thiết kế của mạch điều khiển đèn giao thông	1 tuần	<input checked="" type="checkbox"/>
	Tìm hiểu kiến thức tổng quan về Arm Cotex M3 và VDK STM32F103C8T6	1 tháng	<input checked="" type="checkbox"/>
	Lập kế hoạch chi tiết	1 ngày	<input checked="" type="checkbox"/>
Thiết kế hệ thống	Thiết kế sơ đồ khối, thiết kế chi tiết từng khối	1 ngày	<input checked="" type="checkbox"/>
	Vẽ mạch nguyên lý và mô phỏng trên Proteus	3 ngày	<input checked="" type="checkbox"/>
	Lập trình cho vi điều khiển STM32F103C8T6	1 tuần	<input checked="" type="checkbox"/>
	Nạp code và chạy thử trên proteus	1 ngày	<input checked="" type="checkbox"/>
	Chạy thử trên board với kit STM32F103C8T6	3 ngày	<input checked="" type="checkbox"/>
Hoàn thiện sản phẩm	Thiết kế mạch in trên Altium	1 tuần	<input checked="" type="checkbox"/>
	Đặt mạch và hàn mạch	2 ngày	<input checked="" type="checkbox"/>
	Đo đạc, kiểm tra, đánh giá sản phẩm	3 ngày	<input checked="" type="checkbox"/>
Báo cáo bài tập lớn	Xây dựng và hoàn thành nội dung bản báo cáo	1 tuần	<input checked="" type="checkbox"/>
	Họp bàn chuẩn bị báo cáo trước giảng viên	1 buổi	<input checked="" type="checkbox"/>

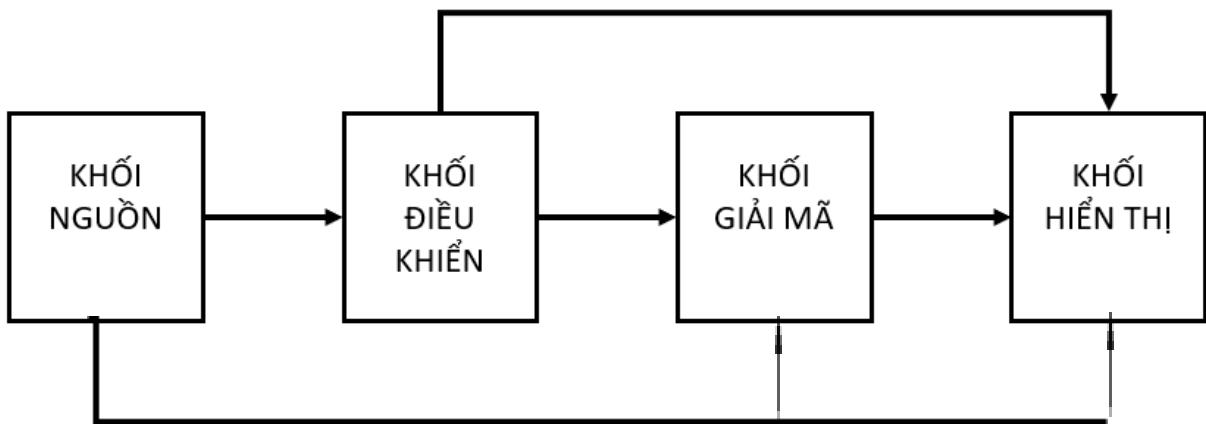
Bảng 3 :Bảng công việc

Trong chương 2 này, chúng em đi xây dựng cụ thể cách thức để có thể mô phỏng đề tài với nạp code thành công trên phần mềm proteus trước khi đi vào tiến hàn thử nghiệm thực tế với linh kiện ở các chương sau.



Hình 2. 1 Quy trình thiết kế

## 2. Sơ đồ khối của hệ thống



Hình 2. 2 Sơ đồ khối của hệ thống

- Khối nguồn: Cung cấp điện áp cho toàn bộ hệ thống và đảm bảo sự ổn định điện áp.
- Khối điều khiển: Điều khiển mọi sự hoạt động của hệ thống, thực hiện chương trình, xử lý các điều khiển vào/ra và truyền thông với các thiết bị bên ngoài.
- Khối giải mã: Sử dụng IC74LS47 để thực hiện việc giải mã BCD sang thập phân và hiển thị các trạng thái thập phân đó trên LED 7 thanh.
- Khối hiển thị: Hiển thị chính xác thời gian trên led 7 thanh và tín hiệu đèn giao thông trên led đơn.

## 2.1 Khối nguồn

### 2.1.1 Adapter 5V DC-2A

Nguồn cung cấp cho mạch vô cùng quan trọng, nó quyết định tính cơ động và thời gian hoạt động của sản phẩm. Do sản phẩm là đèn giao thông, không cần di chuyển nhiều nên tính cơ động cũng không quá quan trọng. Do đó, chọn Adapter 5V – 2A làm nguồn cho sản phẩm giúp chuyển đổi điện 220V AC xuống 5V DC cung cấp điện cho sản phẩm.



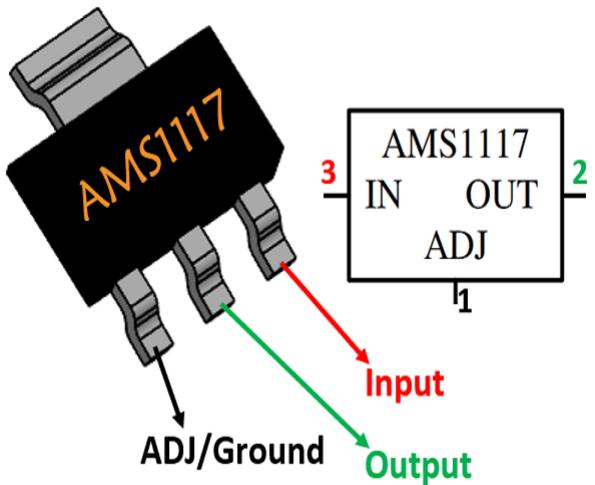
Hình 2. 3 Nguồn 5V-2A

Thông số kỹ thuật:

- Đầu vào: 100 - 220V (50/60Hz).
- Đầu ra : 5VDC/2A.
- Công suất max: 10W.
- Đầu ra jack cắm dc 5.5 x 2.1mm.

### 2.1.2 IC AMS1117

AMS1117 là IC ổn áp 3 chân gói SMD phổ biến có nhiều model cho các yêu cầu điện áp cố định và có thể điều chỉnh. IC có thể cung cấp dòng điện tối đa 1A và điện áp đầu ra có thể thay đổi từ 1,5V đến 5V. Nó cũng có điện áp sụt thấp là 1,3V khi hoạt động ở dòng điện tối đa.



Hình 2. 4 IC hạ áp

Số chân	Tên chân	Mô tả
1	Adj / Ground	Chân này điều chỉnh điện áp đầu ra, nếu là bộ điều chỉnh điện áp cố định thì nó đóng vai trò nối mass.
2	Điện áp đầu ra (Vout)	Điện áp đầu ra điều chỉnh được đặt bởi chân điều chỉnh có thể được lấy từ chân này.
3	Điện áp đầu vào (Vin)	Điện áp đầu vào được điều chỉnh được cấp cho chân này.

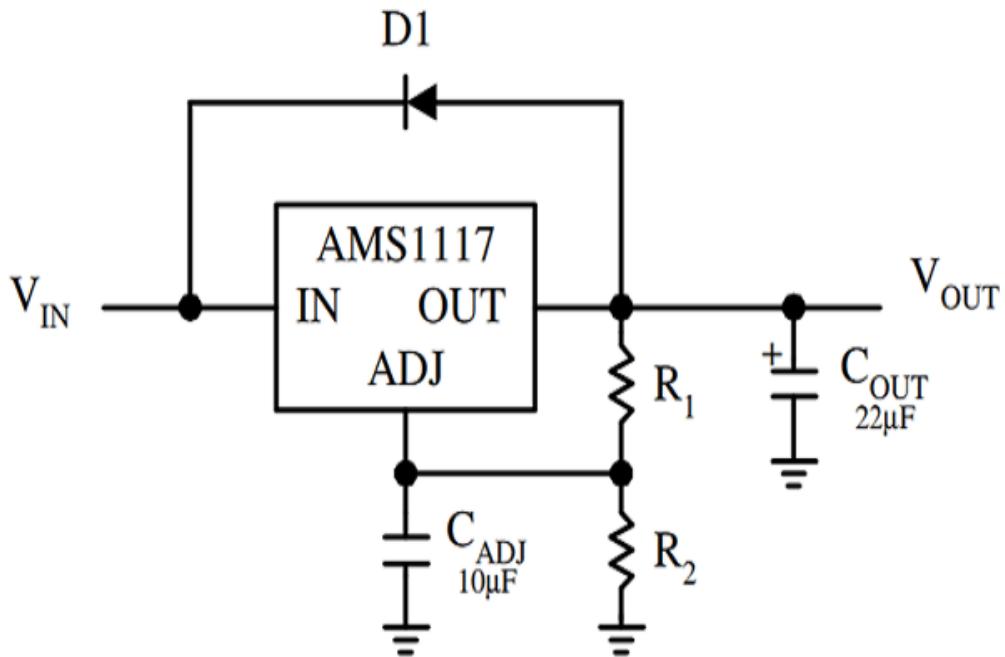
Bảng 4 :Bảng mô tả chức năng IC hạ áp

Thông số, đặc trưng kỹ thuật:

- Bộ điều chỉnh điện áp tuyến tính 3 cực có thể điều chỉnh hoặc cố định
- Bộ điều chỉnh điện áp sụt thấp (LDO)
- Loại điện áp cố định: 1.5V, 1.8V, 2.5V, 2.85V, 3.3V và 5V
- Phạm vi điện áp thay đổi: 1,25V đến 13,8V
- Dòng điện đầu ra là 1000mA
- Điện áp sụt tối đa: 1.3V
- Giới hạn dòng điện tích hợp và bảo vệ nhiệt.
- Nhiệt độ hoạt động lớp tiếp giáp là 125 ° C

Cách sử dụng AMS1117:

Nếu nó là một bộ ổn áp cố định chỉ cần cấp nguồn cho IC thông qua chân Vin và đầu ra được điều chỉnh có thể nhận được trong chân Vout. Chân Adj / Ground trong trường hợp này chỉ hoạt động như một chân mass và được nối mass. Ngoài ra, một tụ điện có thể được thêm vào ở phía đầu ra để lọc nhiễu. Sơ đồ mạch cho một bộ điều chỉnh đầu ra biến đổi ở bên dưới.



Hình 2. 5 Sơ đồ mạch tham chiếu cho điện áp điều chỉnh

Chọn giá trị của R1 và R2 dựa trên điện áp đầu ra cho mạch. Sử dụng công thức:  

$$V_{OUT} = V_{REF} \left( 1 + R_2 / R_1 \right) + I_{ADJ} R_2$$

Đối với bộ ổn áp loại điều chỉnh, chúng ta cần hai điện trở bên ngoài để quyết định điện áp đầu ra của bộ điều chỉnh. Sơ đồ mạch tham chiếu cho điện áp điều chỉnh ở bên dưới, trong đó các điện trở R1 và R2 quyết định điện áp đầu ra của bộ điều chỉnh. Tụ điện CAdj là một linh kiện tùy chọn có thể được thêm vào để cải thiện khả năng loại bỏ gợn sóng nếu được yêu cầu. Hai tụ điện còn lại để lọc nhiễu đầu vào và đầu ra tương ứng.

## 2.2 Khối điều khiển

### 2.2.1 Chip STM32F103C8T6

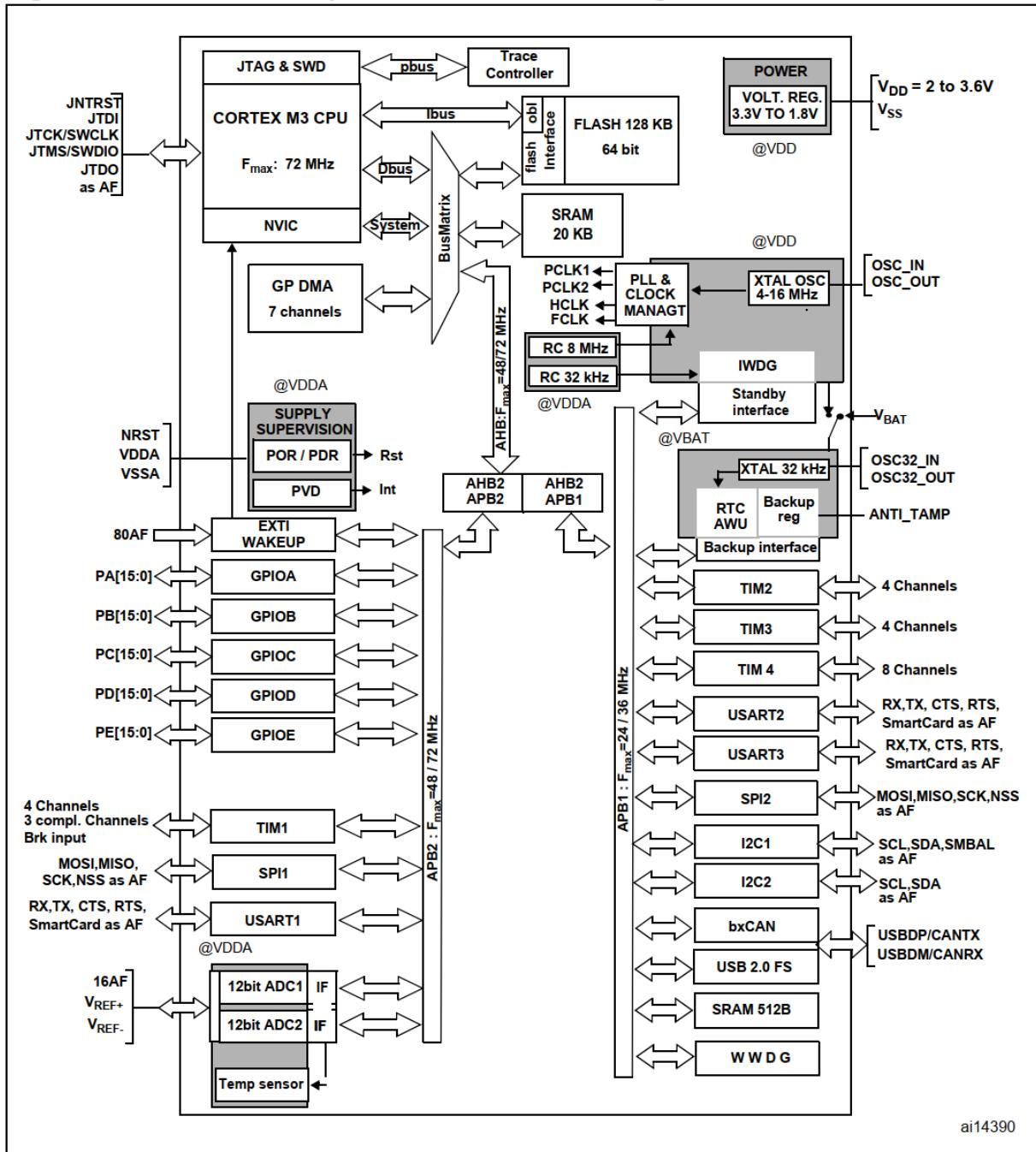


Hình 2. 6 Khối điều khiển STM32F103C8T6

Thông số kỹ thuật:

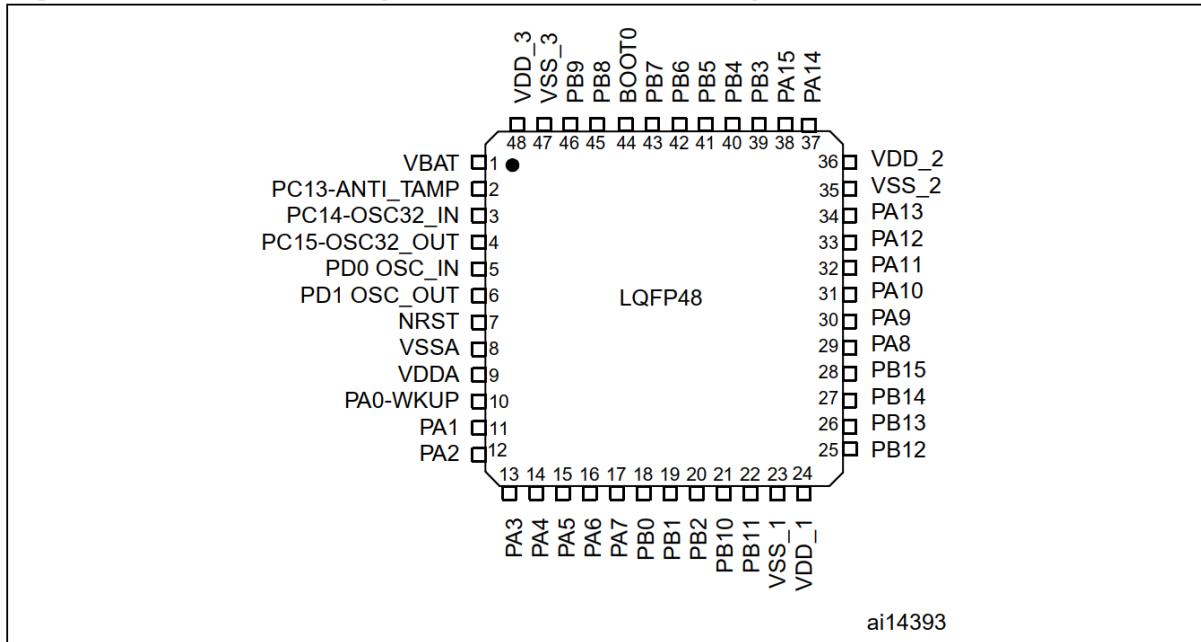
- Kiến trúc vi xử lý 32-bit ARM Cortex-M3
- Tốc độ xử lý lên đến 72 MHz
- Bộ nhớ Flash 64 KB
- Bộ nhớ SRAM 20 KB
- Bộ chuyển đổi ADC 12-bit với tốc độ lên đến 1 MSPS
- Hỗ trợ giao tiếp USART, SPI, I2C, USB, CAN, v.v.
- 2 bộ định thời 16-bit
- 7 chân PWM
- Tính năng watchdog timer
- Nhiệt độ hoạt động từ -40°C đến 85°C
- Vật liệu: nhựa epoxi (màu đen hoặc xám)
- Điện áp hoạt động từ 2V đến 3.6V
- Số lượng chân I/O: 37 chân
- Kích thước: 7mm x 7mm

Figure 1. STM32F103xx performance line block diagram



Hình 2. 7 Sơ đồ khói STM32F103xx

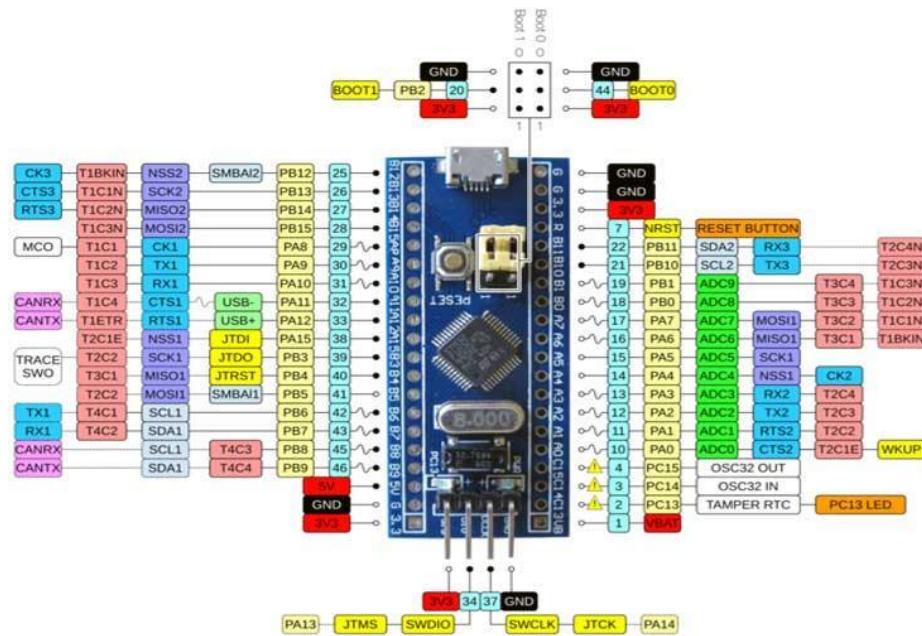
**Figure 4. STM32F103xx performance line LQFP48 pinout**



Hình 2. 8 Sơ đồ minh họa các cổng kết của STM32F103xx

### **2.2.2 KIT phát triển STM32F103C8T6 Blue Bill**

Dùng để chạy thử mạch trên board trắng.



Hình 2. 9 KIT phát triển STM32F103C8T6 Blue Bill

## Thông số kỹ thuật Kit STM32F103C8T6:

- Vị điều khiển: STM32F103C8T6.
  - Điện áp cấp 5VDC qua cổng Micro USB sẽ được chuyển đổi thành 3.3VDC qua IC nguồn và cấp cho Vị điều khiển chính.
  - Tích hợp sẵn thạch anh 8Mhz.
  - Tích hợp sẵn thạch anh 32Khz cho các ứng dụng RTC.
  - Ra chân đầy đủ tất cả các GPIO và giao tiếp: CAN, I2C, SPI, UART, USB,...
  - Tích hợp Led trạng thái nguồn, Led PC13, Nút Reset.
  - Kích thước: 53.34 x 15.24mm

ST Link V2:



Hình 2. 10 ST Link V2

Mạch nạp STM32 ST-Link V2 được sử dụng để nạp chương trình và debug cho Vi điều khiển STM32, mạch nạp có kích thước nhỏ gọn, chi phí thấp, độ bền cao.

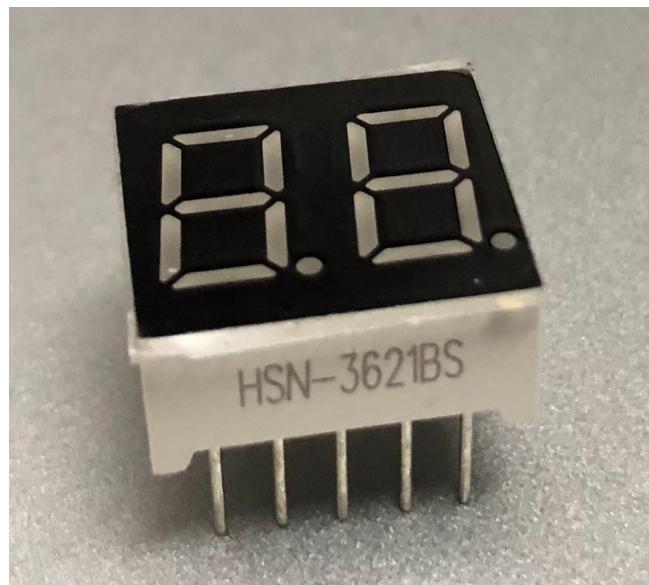
Kết nối chân từ mạch nạp ST-LINK V2 với linh kiện điện tử STM32 theo thứ tự sau:

- RST-NRST
- SWCLK — TCK
- SWDIO — TMS
- GND — GND
- 3.3V — 3.3V

## 2.3 Khối hiển thị

### 2.3.1 Led đôi 7 thanh

Ở khói hiển thị chúng em sử dụng các led đôi 7 thanh mắc anode chung giúp hiển thị số giây với 2 chữ số.



Hình 2. 11 Đèn led 7 thanh

Chân Anode chung là cách mắc: Chân + các led mắc chung lại với nhau.

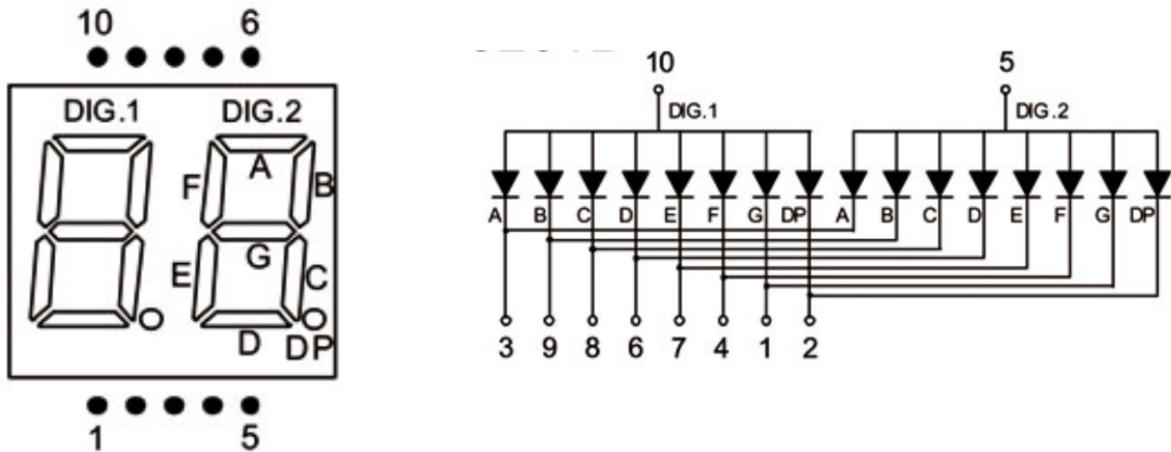
Đối với loại Anode chung:

- Chân 3 và 8 là 2 chân Vcc (nối ngắn mạch lại với nhau, sau đó nối chung với chân anode của 8 led đơn), vậy muốn led nào đó sáng thì chỉ việc nối chân catot xuống mass. Điện áp giữa Vcc và mass phải lớn hơn 1.3V mới cung cấp đủ led sáng, tuy nhiên không được cao quá 3V.
- Trong các mạch thì thường dùng nguồn 5V nên để tránh việc đốt cháy led thì cách đơn giản nhất là mắc thêm trở hạn dòng.

Thông số kỹ thuật:

- Mã sản phẩm: 3261BS
- Kích Thước: 0.36 inch
- Số chân: 10 chân
- Màu led: Đỏ
- Điện áp hoạt động : 1.8 ~2.2V
- Dòng max : 20mA
- Kích thước 14x15mm

- Kiểu LED : LED đôi
- Sơ đồ chân LED 7 đoạn loại dương chung Model 3261BS
- Chân dương chung : Chân 5, Chân 10



Hình 2. 12 Sơ đồ đèn led

### 2.3.2 Led đơn



Hình 2. 13 Đèn led đơn

Thông số kỹ thuật:

- Đường kính: 5mm
- Điện áp led:
  - Đỏ: 1.8 – 2 V
  - Vàng: 1.8 – 2 V
  - Xanh lá: 2.8 – 3.2 V
- Dòng: 10-20mA
- Độ sáng: 2000-4000 milicandela

## 2.4 Khôi giải mã IC 7447

Sử dụng IC 7447 để giải mã led đôi 7 thanh.



Hình 2. 14 IC 7447

Thông số kỹ thuật:

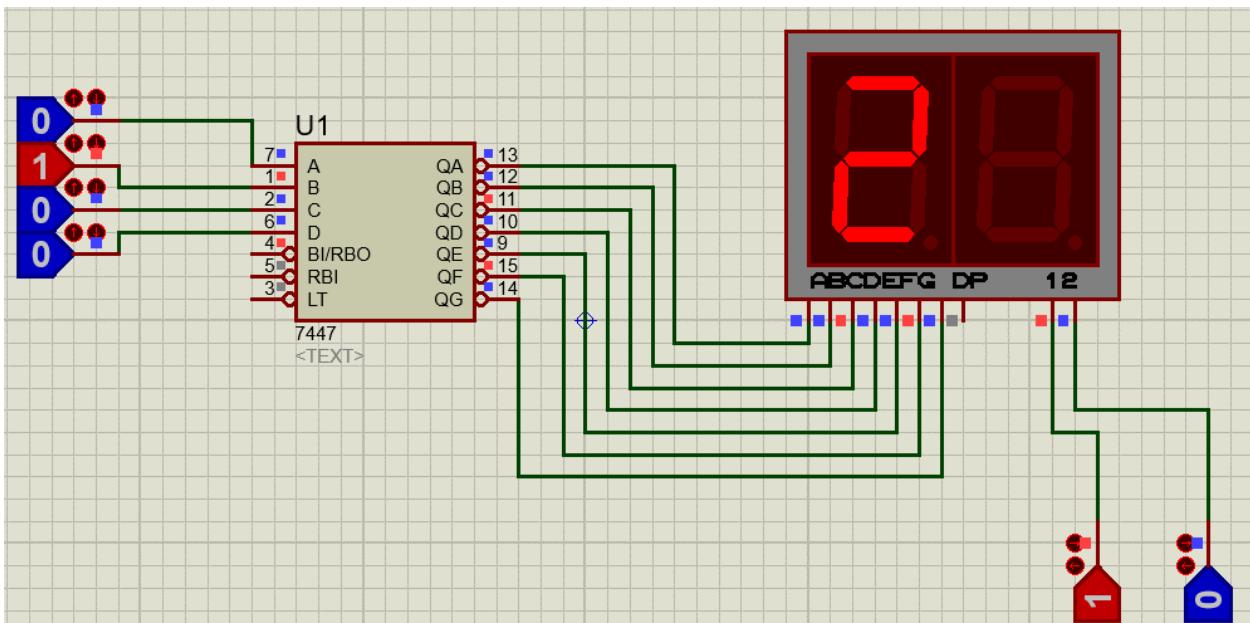
- Loại logic mạch: Bộ giải mã và bộ điều khiển
- Điện áp hoạt động: 4,75 V đến 5,25 V
- Dòng điện tiêu thụ: 24mA
- Nhiệt độ hoạt động: 0°C đến 70°C
- Thời gian trễ: 100s
- Tần số xung nhịp tối đa: 35Mhz
- Số chân cắm: 16
- Số mục: 4
- Số lối ra: 7
- Kích thước: 19,4 x 6,2 x 3,3 mm

### **3. Cách thức điều khiển và hoạt động**

### 3.1 Cách dùng IC 7447 điều khiển cho led đôi 7 thanh anode chung

### 3.1.1 Kết nối

Kết nối các chân của IC 7447 với led đôi 7 đoạn như hình sau:



Hình 2. 15 Sơ đồ kết nối chân IC 7447 với led đôi 7 đoạn anode chung

- Chân số 1, 2, 6, 7 là đầu vào ứng với B, C, D, A
  - Chân số 9, 10, 11, 12, 13, 14, 15 là các chân đầu ra, những chân này sẽ được nối với led 7 thanh để điều khiển chúng.
  - Chân số 8 là chân nối đất GND
  - Chân số 16 là chân cấp nguồn Vcc 5V, không cấp quá nguồn 5V để IC hoạt động bình thường.
  - Chân số 3 LT (Lamp Test) dùng để kiểm tra led 7 đoạn. Nếu chân số 3 nối mass thì led sẽ sáng cùng lúc 7 đoạn. Chân này chỉ dùng để kiểm tra xem led 7 thanh có bị hỏng đoạn nào hay không thôi.
  - Chân số 4 BI/RB0 được nối với mức cao, nếu bị nối với mức thấp thì toàn bộ đèn sẽ không sáng.
  - Chân số 5 RBI nối với mức cao.

### **3.1.2 Phương thức điều khiển**

Sau khi kết nối chân theo sơ đồ nguyên lý ta lập trình điều khiển IC 7447, trước tiên phải cấp mức logic 1 cho led 7 thanh ở 2 chân điều khiển 1,2 để đèn sáng, lúc này led sẽ hiển thị là 00. Sau đó nếu muốn đèn hiển thị số nào thì ta sẽ dựa vào bảng chuyển đổi BCD ra số thập phân để cấp các mức logic tương ứng như sau:

Desimal	Input						BI/RBO	Output						
	$\overline{L}$	$\overline{RBI}$	D	C	B	A		$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$
0	1	1	0	0	0	0	1	0	0	0	0	0	0	1
1	1	X	0	0	0	1	1	1	0	0	1	1	1	1
2	1	X	0	0	1	0	1	0	0	1	0	0	1	0
3	1	X	0	0	1	1	1	0	0	0	0	1	1	0
4	1	X	0	1	0	0	1	1	0	0	1	1	0	0
5	1	X	0	1	0	1	1	0	1	0	0	1	0	0
6	1	X	0	1	1	0	1	1	1	0	0	0	0	0
7	1	X	0	1	1	1	1	0	0	0	1	1	1	1
8	1	X	1	0	0	0	1	0	0	0	0	0	0	0
9	1	X	1	0	0	1	1	0	0	0	1	1	0	0

Hình 2. 16 Bảng chuyển đổi BCD ra số thập phân

### 3.2. Các phương thức giao thức sử dụng

#### 3.2.1 *UART*

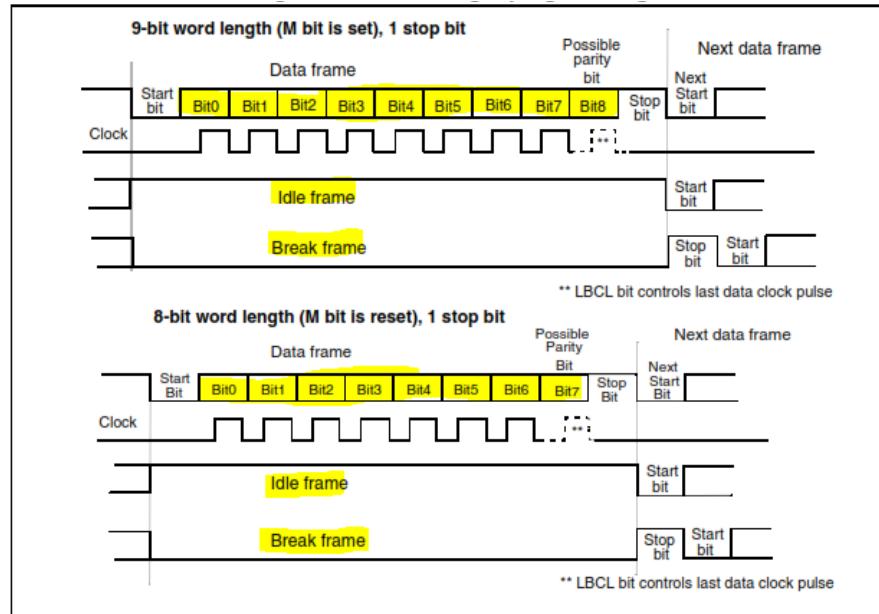
Giao thức này được sử dụng để kết nối với máy tính và các thiết bị khác để truyền và nhận dữ liệu. Chúng ta dùng UART để gửi lệnh điều khiển đèn từ máy tính tới mạch điều khiển đèn giao thông.

##### Các khái niệm quan trọng trong chuẩn truyền thông UART:

**Baudrate:** Số bit truyền được trong 1s, ở truyền nhận không đồng bộ thì ở các bên truyền và nhận phải thống nhất Baudrate.

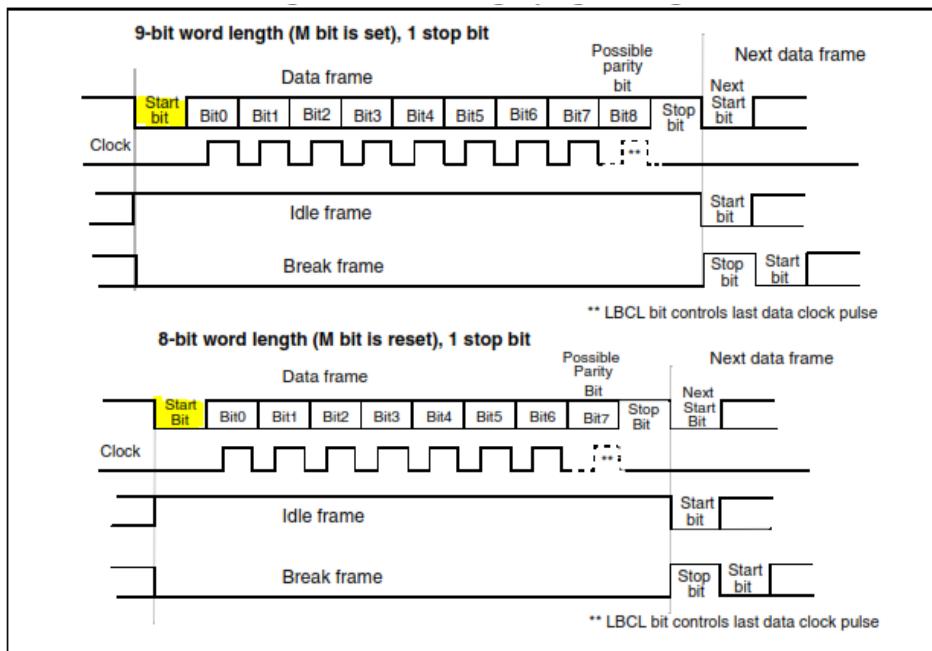
**Frame:** Ngoài việc giống nhau của tốc độ baud 2 thiết bị truyền nhận thì khung truyền của bên cũng được cấu hình giống nhau. Khung truyền quy định số bit trong mỗi lần truyền, bit bắt đầu “Start bit”, các bit kết thúc (Stop bit), bit kiểm tra tính chẵn lẻ (Parity), ngoài ra số bit quy định trong một gói dữ liệu cũng được quy định bởi khung truyền. Có thể thấy, khung truyền đóng một vai trò rất quan trọng trong việc truyền thành công dữ liệu.

- Idle frame: Đường truyền UART ở mức “1”, để xác nhận hiện tại đường truyền dữ liệu trống, không có frame nào đang được truyền đi.
- Break frame: Đường truyền UART ở mức “0”, để xác nhận hiện tại trên đường truyền đang truyền dữ liệu, có frame đang được truyền đi.



Hình 2. 17 Frame

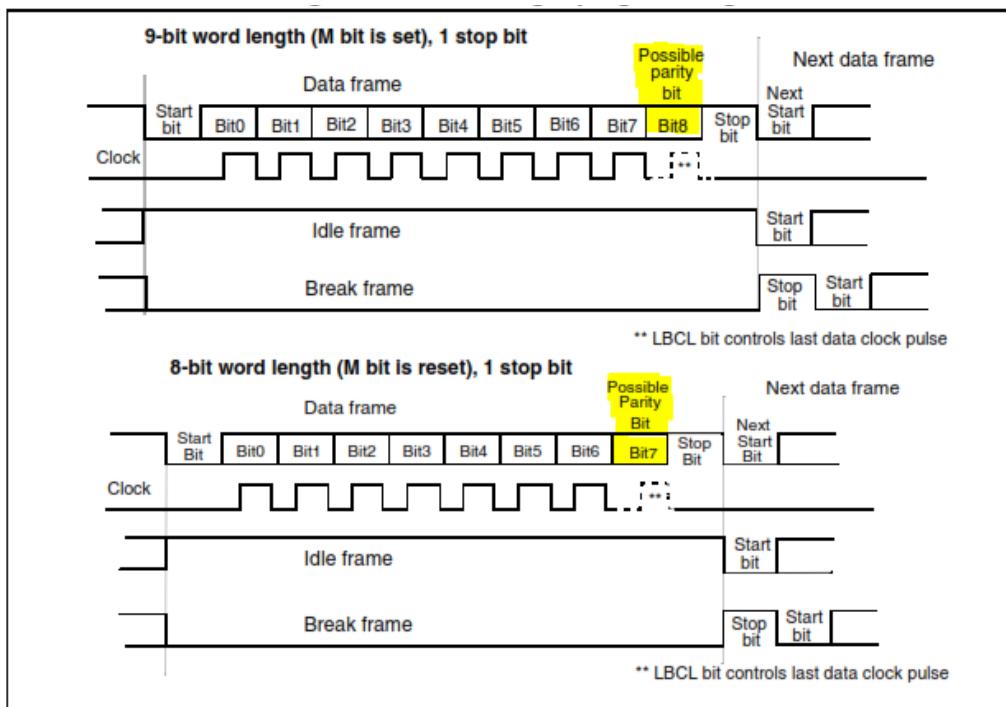
**Start bit:** Bit đầu tiên được truyền trong một frame, bit này có chức năng báo cho bên nhận rằng sắp có một gói dữ liệu truyền đến. Đường truyền UART luôn ở trạng thái cao mức “1” cho đến khi chip muốn truyền dữ liệu đi thì nó gửi bit start bằng cách kéo xuống mức “0”. Như vậy start bit giá trị điện áp 0V và phải bắt buộc có bit start trong khung truyền.



Hình 2. 18 Start bit

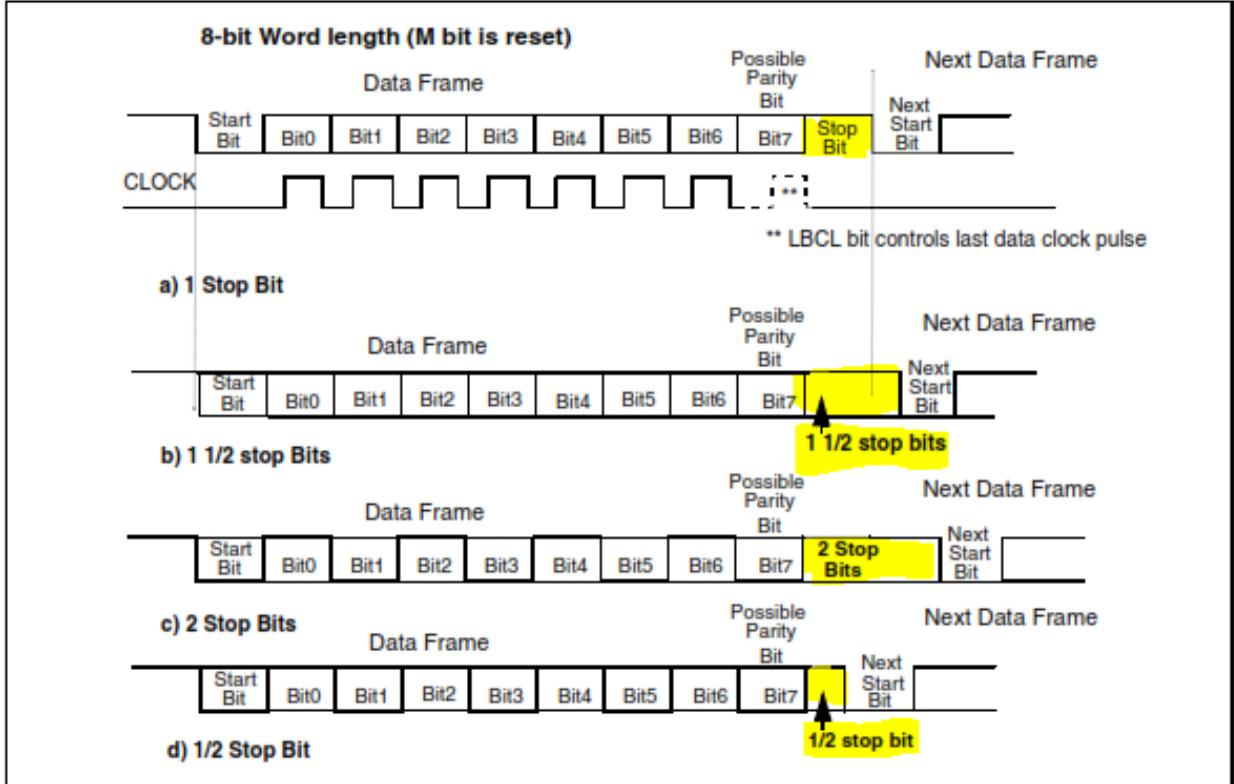
**Data:** Data hay dữ liệu là thông tin mà chúng ta nhận được trong quá trình truyền và nhận. Data trong STM32 có quy định khung truyền là 8bit hoặc 9bit. Trong quá trình truyền UART, bit có trọng số thấp nhất (LSB – least significant bit – bên phải) sẽ được truyền trước và cuối cùng là bit có ánh hưởng cao nhất (MSB – most significant bit – bên trái)

**Parity bit:** Parity dùng để kiểm tra dữ liệu truyền có đúng hay không. Có 2 loại Parity đó là Parity chẵn (even parity) và parity lẻ (odd parity). Parity chẵn nghĩa là số bit 1 trong trong data truyền cùng với bit Parity luôn là số chẵn, ngược lại nếu Parity lẻ nghĩa là số bit 1 trong data truyền cùng với bit Parity luôn là số lẻ. Bit Parity không phải là bit bắt buộc và vì thế chúng ta có thể loại bỏ bit này ra khỏi khung truyền.



Hình 2. 19 Parity bit

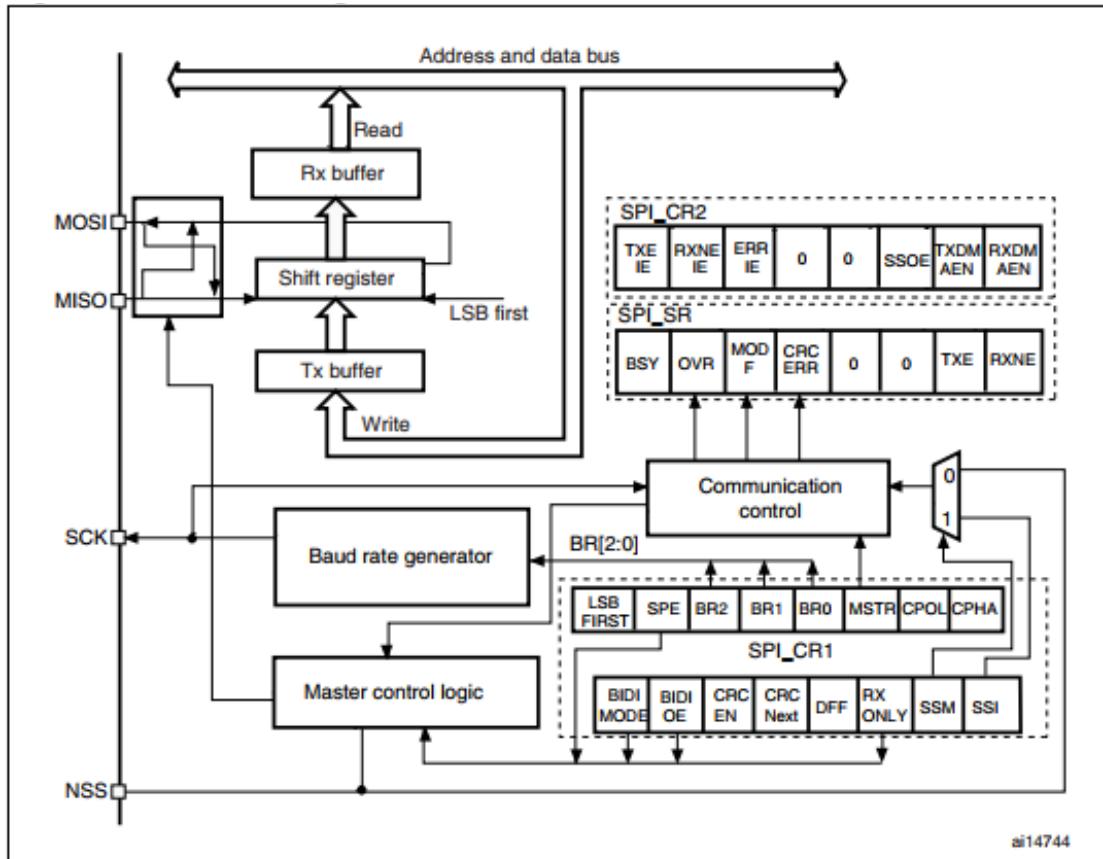
**Stop bits:** Stop bits là một bit báo cáo để cho bộ truyền/nhận biết được gói dữ liệu đã được gửi xong. Stop bits là bit bắt buộc phải có trong khung truyền. Stop bits có thể là 1bit, 1.5bit, 2bit, 0.5bit tùy thuộc vào ứng dụng UART của người sử dụng.



Hình 2. 20 Stop bit

### 3.2.2 SPI

Giao thức này được dùng để kết nối LED 7 thanh hiển thị số đếm thời gian đèn đỏ, xanh và vàng. Mạch điều khiển sẽ gửi các lệnh và dữ liệu đến màn hình led thông qua SPI để hiển thị thời gian đếm ngược.



Hình 2. 21 Sơ đồ khói SPI của STM32F103

- SCK: Xung tín hiệu, là tín hiệu đầu ra của Master và là tín hiệu đầu vào của Slave.
- MISO: Dữ liệu Master In / Slave Out. Chân này có thể được sử dụng để truyền dữ liệu ở chế độ Slave và nhận dữ liệu ở chế độ Master.
- MOSI: Master Out / Slave In data. Chân này có thể được sử dụng để truyền dữ liệu ở chế độ Master và nhận dữ liệu ở chế độ Slave.
- NSS: Chọn Slave. Đây là chân tùy chọn để chọn chế độ Master / Slave. Chân này hoạt động như một "chip chọn" để cho phép SPI master giao tiếp với từng Slave và tránh tranh chấp trên các đường dữ liệu.

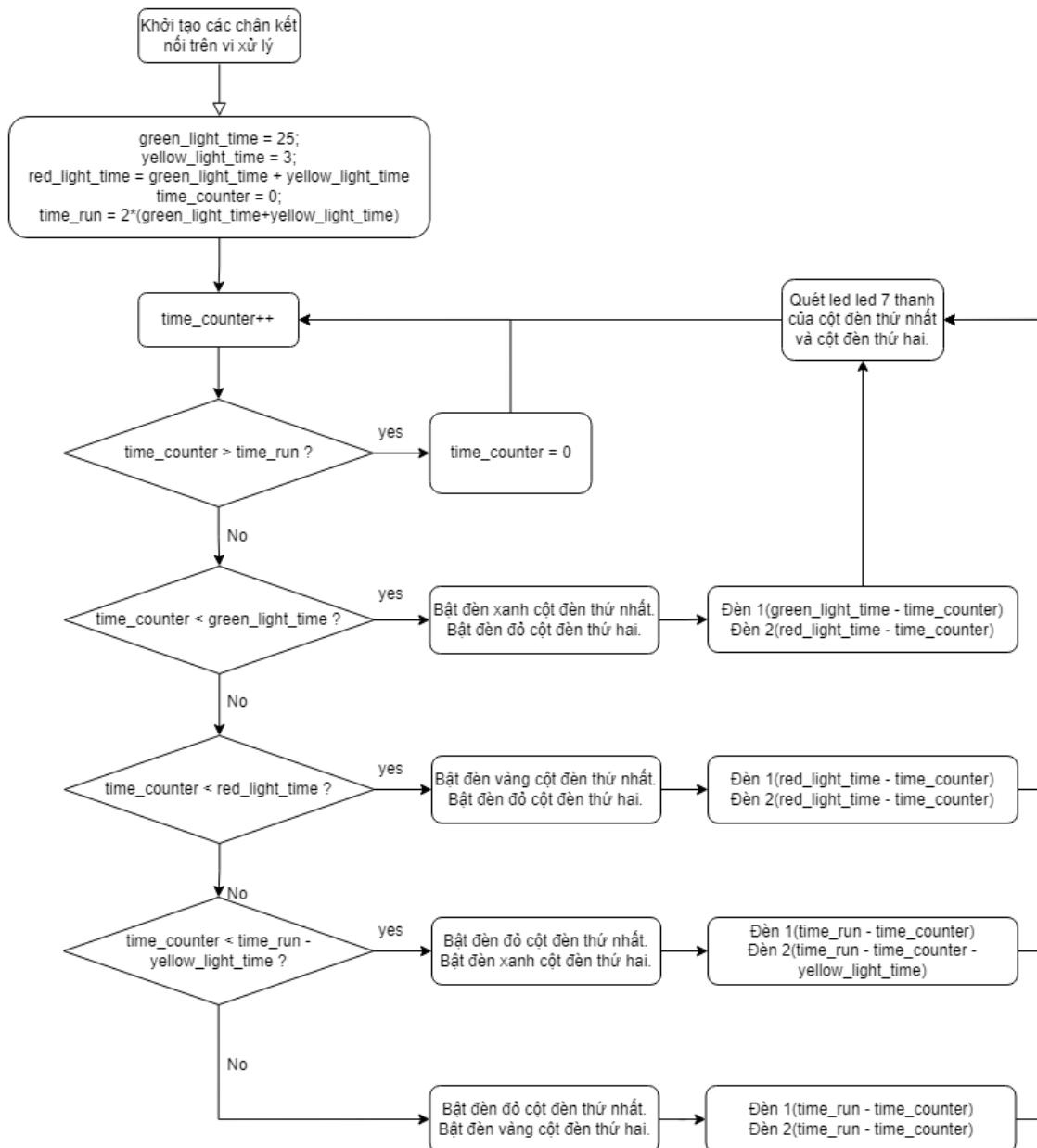
Chip STM32F103 hỗ trợ cho chúng ta 2 bộ SPI, lần lượt là SPI1, SPI2 tương ứng với các chân như sau.

	SPI1	SPI2
SCK	PA5	PB13
MISO	PA6	PB14
MOSI	PA7	PB15
SS	PA4	PB12

Bảng 7. Mô tả các cổng kết nối của chip STM32F103

## 4. Lập trình cho STM32F103C8T6

### 4.1 Xây dựng lưu đồ thuật toán



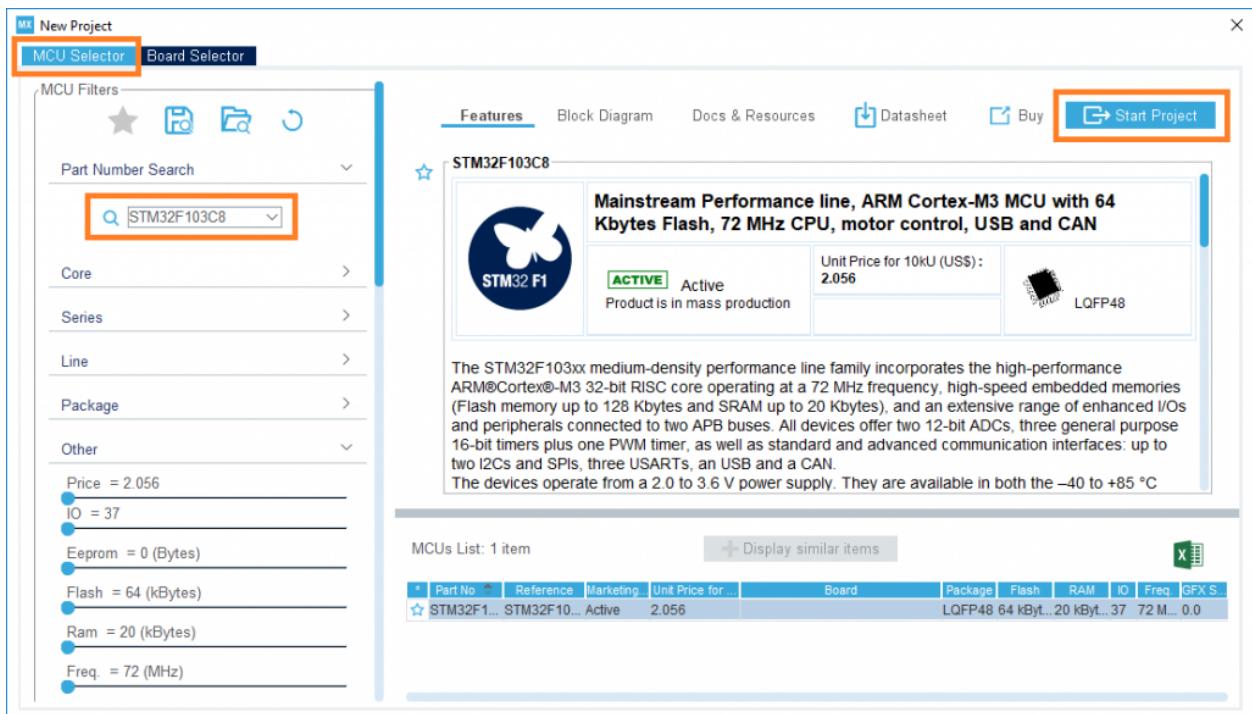
Hình 2. 22 Sơ đồ lưu đồ thuật toán lập trình

## 4.2 Triển khai code

Các bước lập trình:

Bước 1: Tạo File project mới bằng phần mềm STM32CubeMX

Bước 2: Chọn vi điều khiển muốn sử dụng ở mục MCU Selector. Ở project này ta sẽ chọn vi điều khiển STM32F103C8T6.

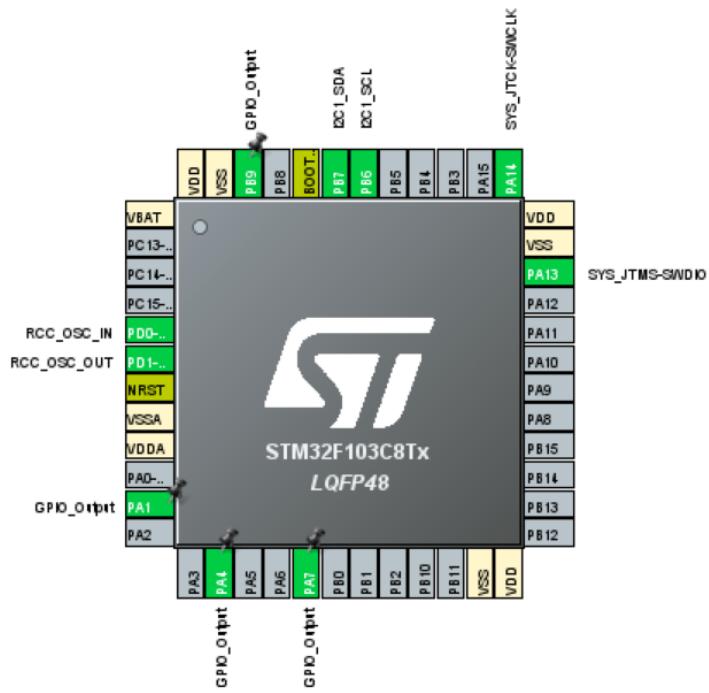


Hình 2. 23 Bảng điều khiển phần mềm STM32CubeMX

Bước 3: Cấu hình cho vi điều khiển STM32F103C8T6: Ở đây ta sẽ cấu hình phương thức nạp code bằng cách chọn “System Core” → “SYS” → “Debug” → Serial Wire để nạp 24 chương trình cho vi điều khiển qua 2 chân SWDIO và SWCLK (đây chính là 2 chân của mạch nạp ST-Link V2 kết nối với vi điều khiển). Tiếp theo ta sẽ Cấu hình cho vi điều khiển sử dụng bộ dao động ngoại “Reset and Clock control (RCC)” → “High Speed Clock (HSE)” → “Crystal/ Ceramic Resonator” (chọn bộ dao động thạch anh 8MHZ tương ứng với High Speed Clock, còn Low Speed Clock sẽ tương ứng với thạch anh 32,768 MHZ).

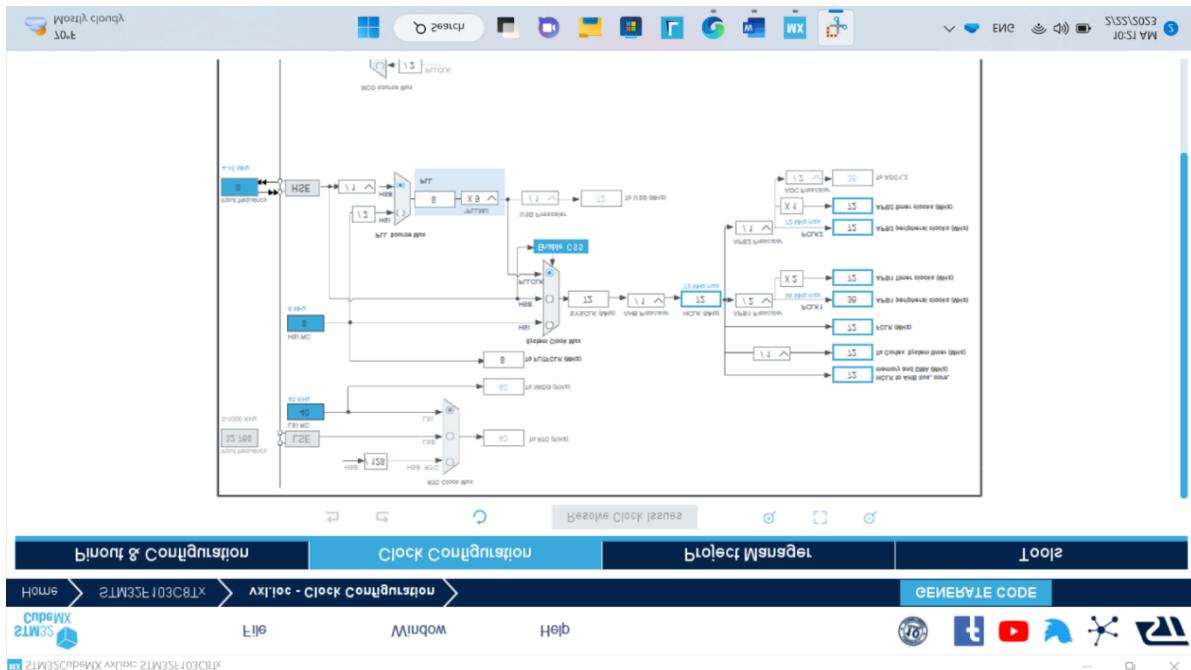
Cấu hình ngoại vi: Việc cấu hình các ngoại vi như Input, Output, External Interrupt, ADC, TIMER, UART... có thể được thực hiện bằng cách chuột phải để chọn chân trực tiếp và kích chuột trái vào chân mà mình muốn cài đặt. Với đề tài này ta sẽ chọn như sau:

- Chân PA1, PA4, PA5, PB9 tương ứng với 4 GPIO\_Output.
- Chọn Timers
- Chọn Connectivity



Hình 2. 24 Sơ đồ cổng kết nối STM32F103C8Tx

Tiếp theo, tại mục Clock Configuration: ta cấu hình lựa chọn nguồn tạo dao động và tần số hoạt động cho vi điều khiển (Bộ xử lý trung tâm – CPU và Peripherals – các ngoại vi) thông qua Clock tree. (Kết hợp với cấu hình RCC tại System Core). Ở đây ta sẽ chọn tần số đầu vào thạch anh là 8MHZ, HSE và PLLCLK, ta sẽ sử dụng HCLK 72MHZ.



Hình 2. 25 Sơ đồ minh họa trên phần mềm STM32CubeMX

Bước 4: Lưu và project và chọn MDK-ARM V5 để sử dụng trên Keil C.

Dưới đây là Source code:

```
#define LED7SEG_A GPIO_PIN_0
#define LED7SEG_B GPIO_PIN_1
#define LED7SEG_C GPIO_PIN_2
#define LED7SEG_D GPIO_PIN_3

#define LED1    GPIO_PIN_12
#define LED2    GPIO_PIN_13
#define LED3    GPIO_PIN_14
#define LED4    GPIO_PIN_15

#define LED_DO    GPIO_PIN_5
#define LED_VANG  GPIO_PIN_4
#define LED_XANH  GPIO_PIN_3

#define LED_DO1   GPIO_PIN_9
#define LED_VANG1 GPIO_PIN_8
#define LED_XANH1 GPIO_PIN_7

#define PORT_LED7SEG_CODE GPIOA
#define PORT_LED      GPIOB

#define delayTime1      5

int green_light_time = 25;
int yellow_light_time = 3;
int red_light_time;
int time_counter = 0;
int time_run;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

void led7seg( int y){
    switch(y){
        case 0 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_RESET);
                  HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_RESET);
                  HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_RESET);
                  HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_RESET);
                  break;
        case 1 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_SET);
                  HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_RESET);
                  HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_RESET);
                  HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_RESET);
                  break;
        case 2 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_RESET);
                  HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_SET);
                  HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_RESET);
```

```

        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_RESET);
        break;
    case 3 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_RESET);
        break;
    case 4 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_RESET);
        break;
    case 5 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_RESET);
        break;
    case 6 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_RESET);
        break;
    case 7 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_RESET);
        break;
    case 8 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_SET);
        break;
    case 9 : HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_A, GPIO_PIN_SET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_B, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_C, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED7SEG_CODE, LED7SEG_D, GPIO_PIN_SET);
        break;
    }
}
void show7Seg2(int c, int x){
    int x1, x2;
    x1 = x/10;
    x2 = x%10;
    if(c == 1){
        HAL_GPIO_WritePin(PORT_LED, LED1, GPIO_PIN_SET);
        led7seg(x1);
        HAL_Delay(delayTime1);
        HAL_GPIO_WritePin(PORT_LED, LED1, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PORT_LED, LED2, GPIO_PIN_SET);
    }
}

```

```

    led7seg(x2);
    HAL_Delay(delayTime1);
    HAL_GPIO_WritePin(PORT_LED, LED2, GPIO_PIN_RESET);
} else {
    HAL_GPIO_WritePin(PORT_LED, LED3, GPIO_PIN_SET);
    led7seg(x1);
    HAL_Delay(delayTime1);
    HAL_GPIO_WritePin(PORT_LED, LED3, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(PORT_LED, LED4, GPIO_PIN_SET);
    led7seg(x2);
    HAL_Delay(delayTime1);
    HAL_GPIO_WritePin(PORT_LED, LED4, GPIO_PIN_RESET);

}
}

int main(void)
{
    HAL_Init();

/* Configure the system clock */
SystemClock_Config();

/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_SET);

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_RESET);

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);

    time_run = 2*(green_light_time+yellow_light_time);
    red_light_time = green_light_time + yellow_light_time;
/* USER CODE END 2 */

```

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    time_counter++;
    if(time_counter > time_run)
        time_counter = 0;
    if(time_counter < green_light_time){
        // cot 1
        HAL_GPIO_WritePin(GPIOB, LED_XANH, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, LED_VANG, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, LED_DO, GPIO_PIN_SET);

        // cot 2
        HAL_GPIO_WritePin(GPIOB, LED_XANH1, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, LED_VANG1, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, LED_D01, GPIO_PIN_RESET);

        for(int i=0; i<50; i++){
            show7Seg2(1, green_light_time - time_counter);
            show7Seg2(0, red_light_time - time_counter);
        }
    } else if(time_counter < red_light_time){
        // cot 1
        HAL_GPIO_WritePin(GPIOB, LED_XANH, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, LED_VANG, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, LED_DO, GPIO_PIN_SET);

        // cot 2
        HAL_GPIO_WritePin(GPIOB, LED_XANH1, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, LED_VANG1, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, LED_D01, GPIO_PIN_RESET);

        for(int i=0; i<50; i++){
            show7Seg2(1, red_light_time - time_counter);
            show7Seg2(0, red_light_time - time_counter);
        }
    } else if(time_counter < time_run-yellow_light_time){
        // cot 1
        HAL_GPIO_WritePin(GPIOB, LED_XANH, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, LED_VANG, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, LED_DO, GPIO_PIN_RESET);

        // cot 2
        HAL_GPIO_WritePin(GPIOB, LED_XANH1, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB, LED_VANG1, GPIO_PIN_SET);
    }
}

```

```

    HAL_GPIO_WritePin(GPIOB, LED_DO1, GPIO_PIN_SET);

    for(int i=0; i<50; i++){
        show7Seg2(1, time_run - time_counter);
        show7Seg2(0, time_run - time_counter - yellow_light_time);
    }
} else {
    // cot 1
    HAL_GPIO_WritePin(GPIOB, LED_XANH, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, LED_VANG, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, LED_DO, GPIO_PIN_RESET);

    // cot 2
    HAL_GPIO_WritePin(GPIOB, LED_XANH1, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, LED_VANG1, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, LED_DO1, GPIO_PIN_SET);

    for(int i=0; i<50; i++){
        show7Seg2(1, time_run - time_counter);
        show7Seg2(0, time_run - time_counter);
    }
}

}

void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

/** Initializes the RCC Oscillators according to the specified parameters
* in the RCC_OscInitTypeDef structure.
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSISState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                            |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;

```

```

RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if(HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
{
    Error_Handler();
}

static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15
        |GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_7
        |GPIO_PIN_8|GPIO_PIN_9, GPIO_PIN_RESET);

    /*Configure GPIO pins : PA0 PA1 PA2 PA3 */
    GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /*Configure GPIO pins : PB12 PB13 PB14 PB15
       PB3 PB4 PB5 PB7
       PB8 PB9 */
    GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15
        |GPIO_PIN_3|GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_7
        |GPIO_PIN_8|GPIO_PIN_9;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

}

```

```
void Error_Handler(void)
{
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

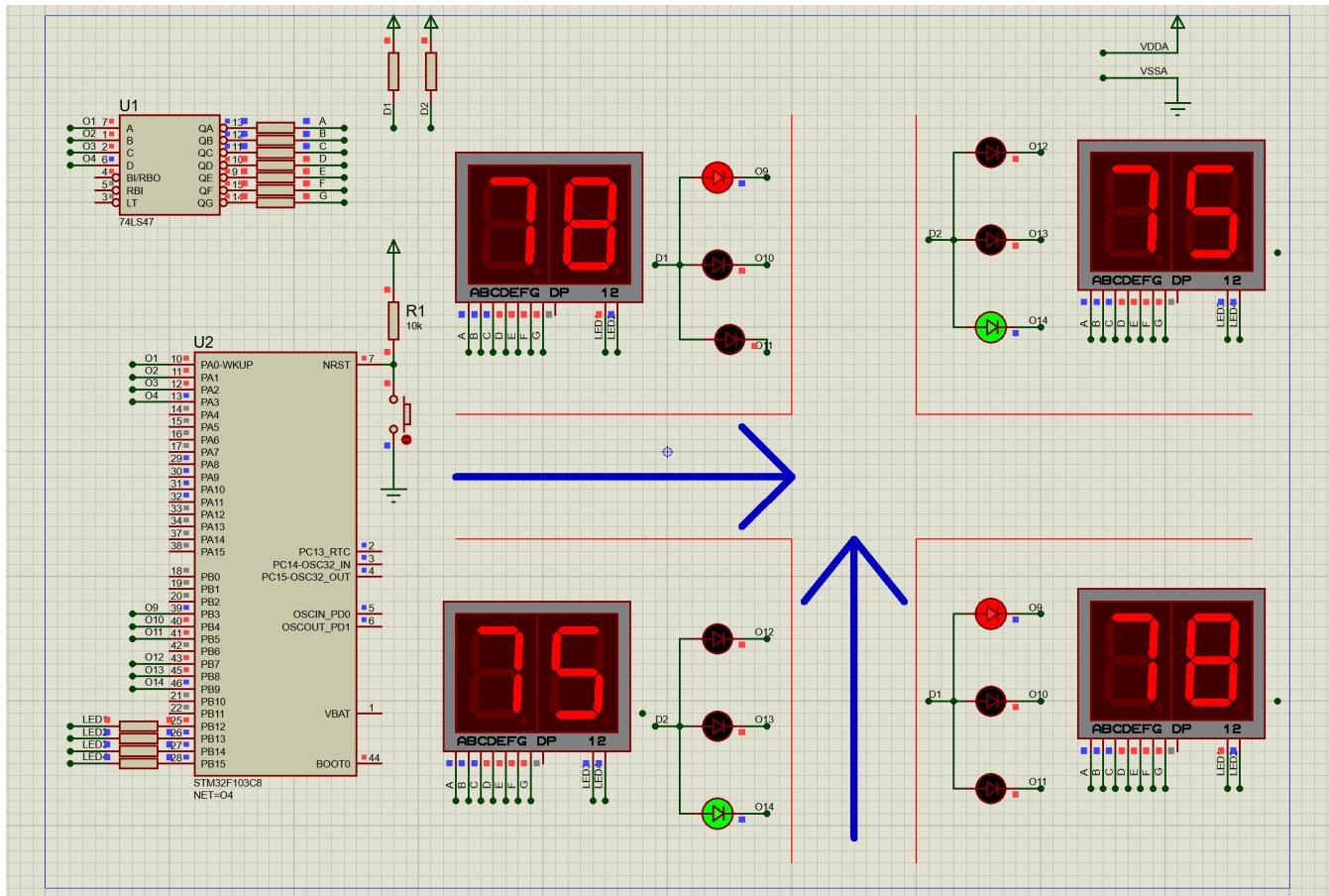
#ifndef USE_FULL_ASSERT

void assert_failed(uint8_t *file, uint32_t line)
{
}

#endif /* USE_FULL_ASSERT */
```

### 4.3 Chạy mô phỏng trên Proteus

- Tổng quan các bước mô phỏng: Tạo Project mới □ Nhấn phím P trên bàn phím để vào chế độ lựa chọn linh kiện ở trong thư viện □ Tìm kiếm các linh kiện mà nhóm đã lựa chọn □ Vẽ mạch kết nối các linh kiện và nguồn.
- Nhấn vào STM32F103C8T6: nhập 8MHz (là tần số dao động của thạch anh nội nhóm sử dụng), sau đó tìm lấy file hex có được từ run code để nạp vào cho STM32F103C8T6.
- Chạy và xem thử kết quả mô phỏng.



Hình 2. 26 Mạch mô phỏng

## **5. Tổng kết chương**

Qua chương 2, chúng ta đã xác định phân chia rõ chi tiết công việc và quy trình thiết kế sản phẩm từ yêu cầu chức năng, phi chức năng đến hoàn thiện thành công mô phỏng proteus.

Bên cạnh đó, xác định và phân tích rõ chức năng các khối của sản phẩm, các thông số của linh kiện được sử dụng. Hiểu rõ các phương thức hoạt động, kết nối và điều khiển được sử dụng trong mạch. Triển khai lưu đồ thuật toán và viết code, xuất file hex thành công để nạp lên proteus.

## Chương 3. Thử nghiệm và hoàn thiện

### 1. Vẽ mạch nguyên lý và mạch in trên Altium

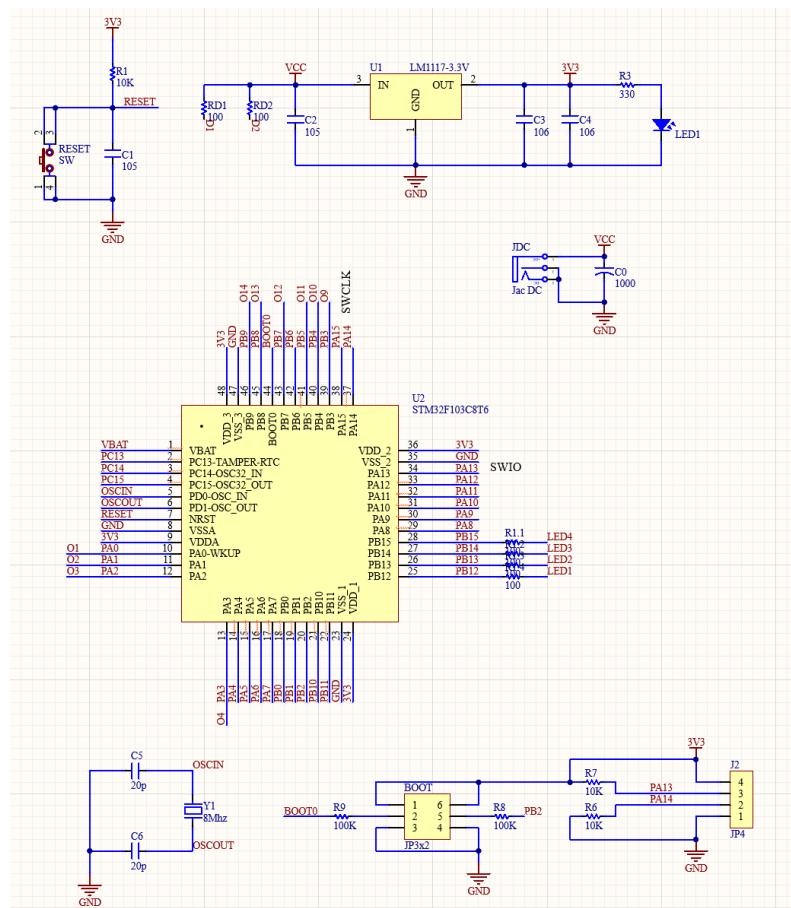
#### 1.1 Xây dựng schematic

Triển khai các khối thành các model:

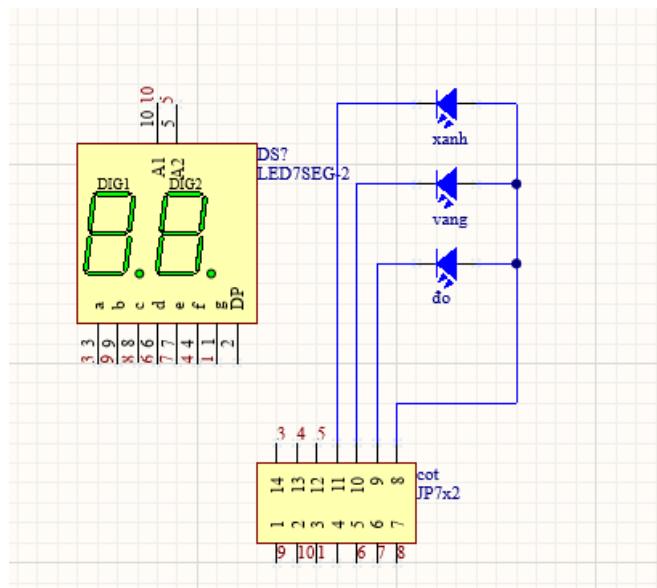
- Khối điều khiển và khói nguồn: Chip STM32F103C8T6, thạch anh 8MHz, boot,.IC AMS 1117 và jack nguồn 5VDC hoặc jack nạp và cấp nguồn bằng ST Link
- Khối giải mã: 1 IC7447
- Khối model led: gồm 4 khói với mỗi khói gồm 1 led đôi 7 thanh anode chung và 3 led đơn xanh, đỏ, vàng.

Tạo Project mới □ Tạo Schematic trong project này □ Chọn components □ chọn thư viện và tìm kiếm các linh kiện.

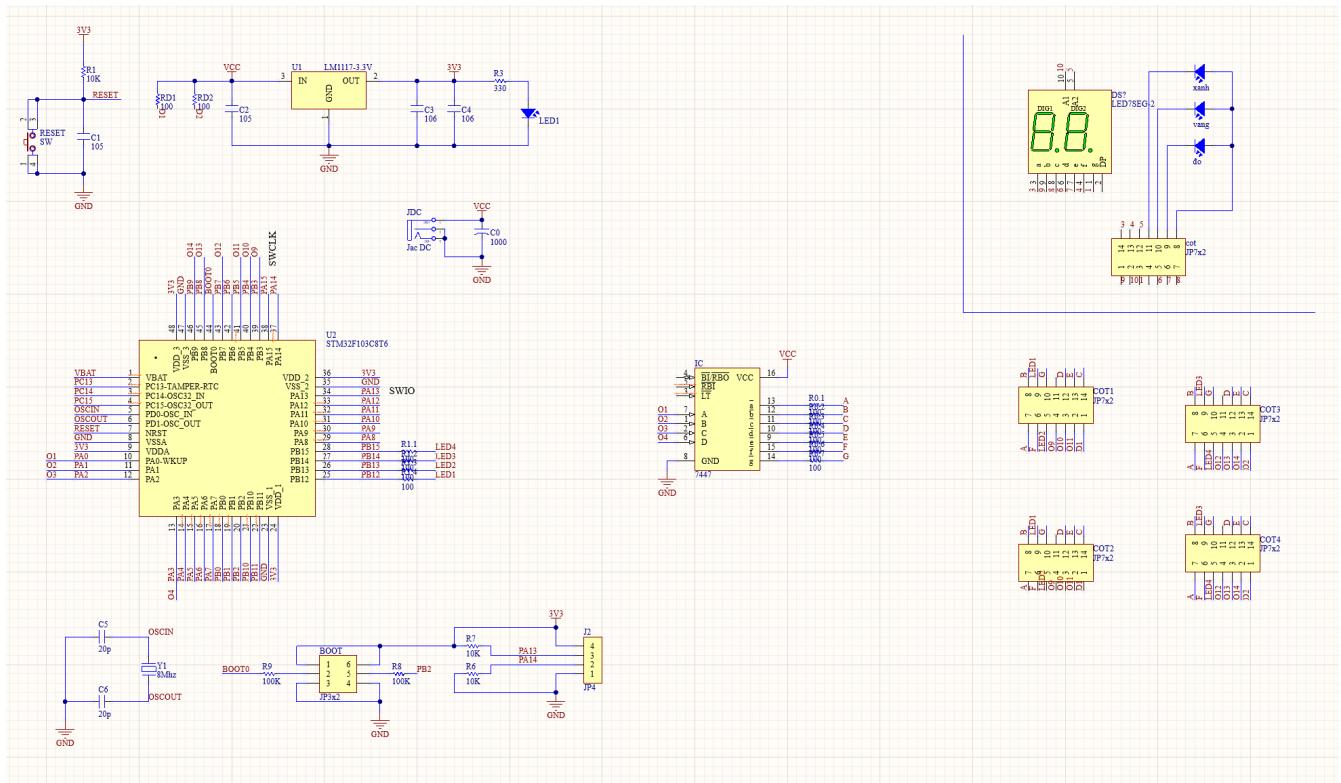
Chú ý lựa chọn các thông số có kích thước đúng, hình dáng thực tế ở mục footprint. Tiến hành chia khói kết nối dây cho mạch rõ ràng.



Hình 3. 1 Khối model điều khiển và khói nguồn



Hình 3.2 Khối model hiển thị

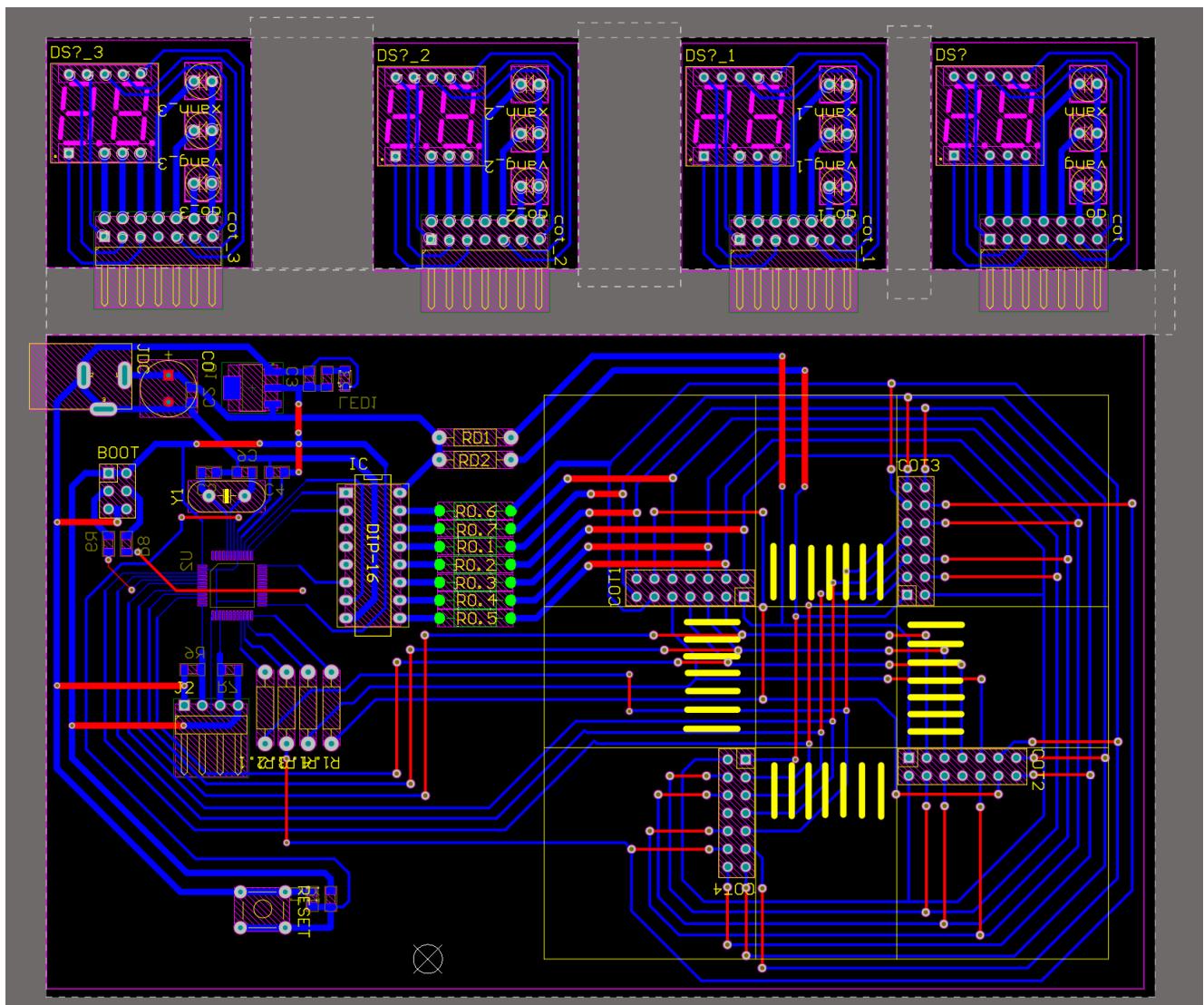


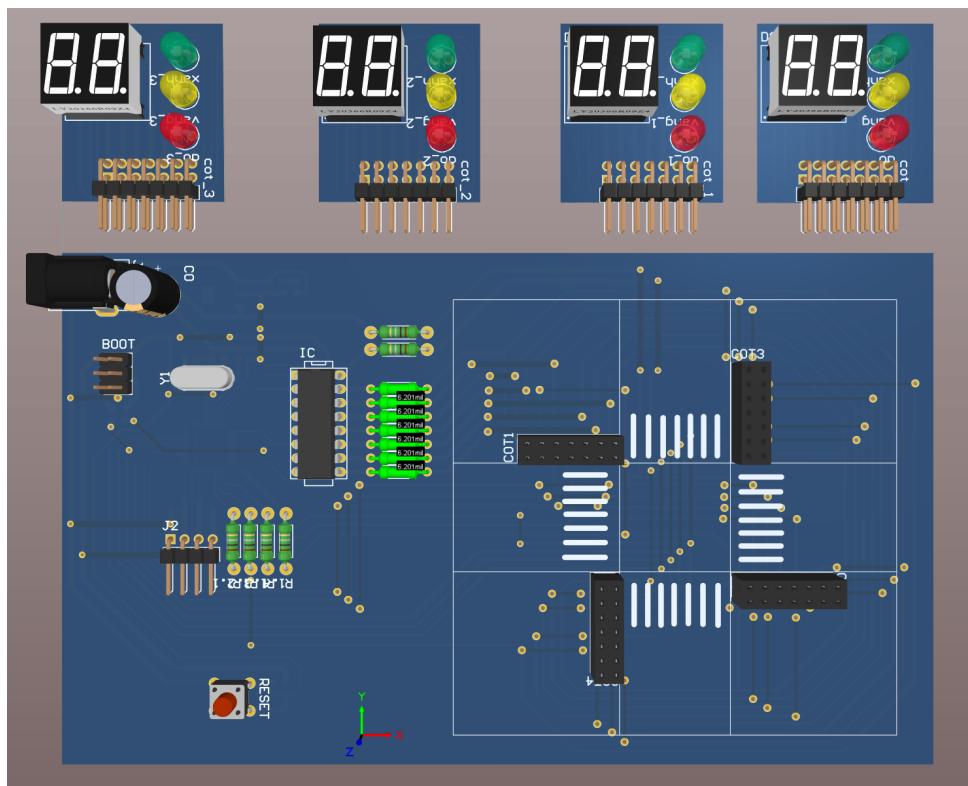
Hình 3. 3 Mạch mô phỏng tổng quan

## 1.2 Vẽ mạch in PCB

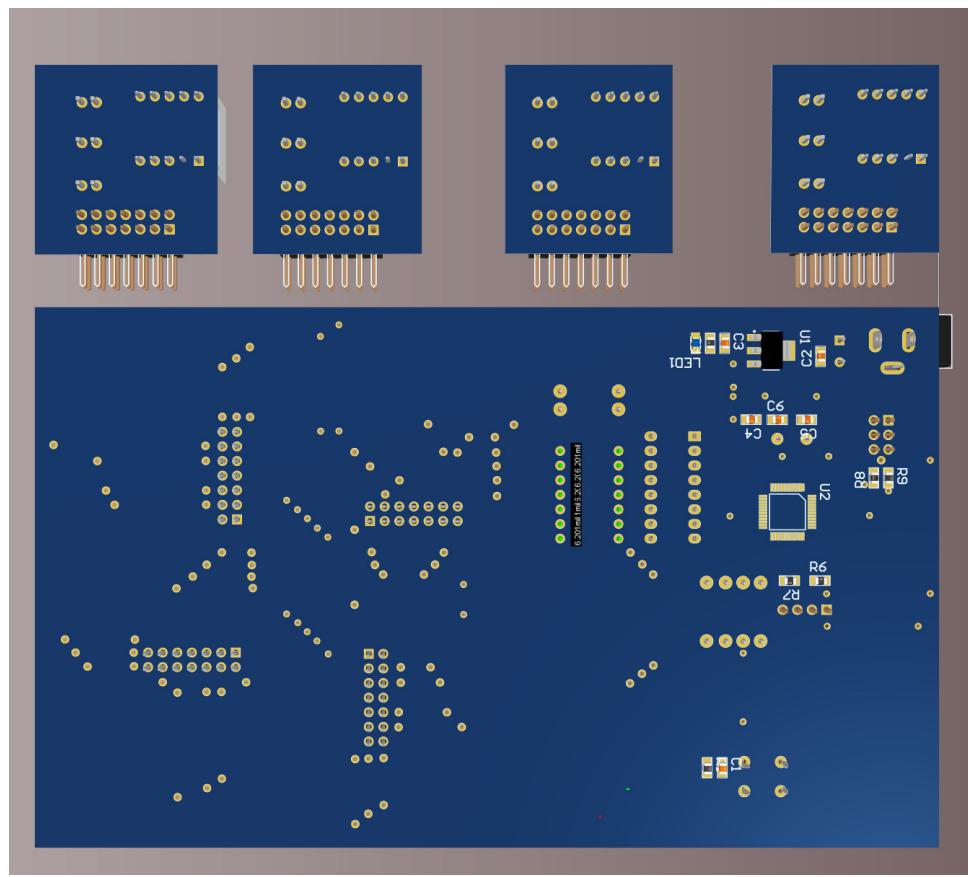
Sau khi vẽ xong mạch nguyên lý và build kiểm tra lỗi và thành công thì chúng ta sẽ chuyển qua bước vẽ mạch in PCB.

- Tạo Mạch in PCB trong file project này (cùng chứa với mạch nguyên lý tương ứng trên).
- Sau đó tiến hành sắp xếp linh kiện phù hợp sao cho đi dây dễ dàng. Vì mạch phức tạp chúng em quyết định đi dây 2 mặt (2 lớp layout) □ Đặt luật đi dây như độ rộng dây, chế độ phủ chân,... □ Check lại bằng hình 2D và 3D mong muốn.





Hình 3. 5 Layout mặt trên 3D



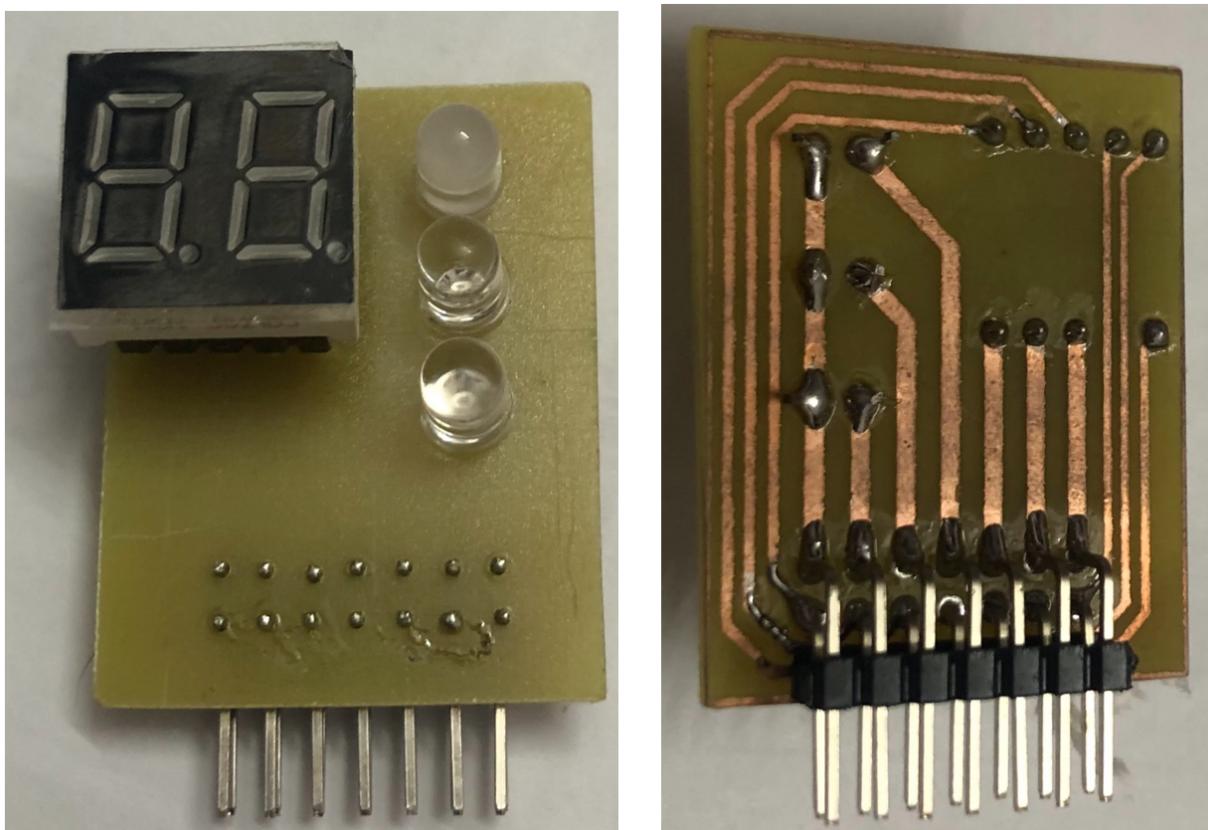
Hình 3. 6 Layout mặt dưới 3D

## 2. Hàn mạch

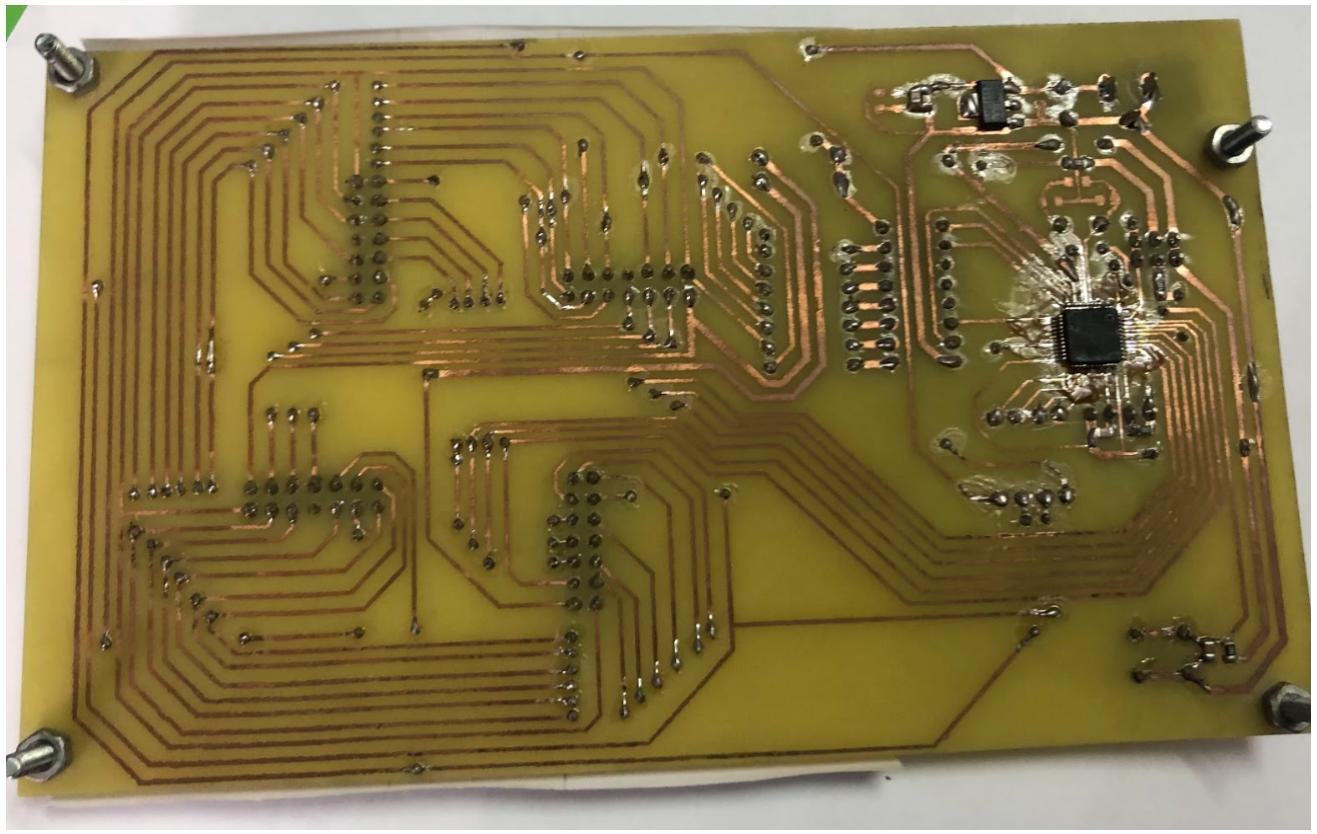
Sau khi có mạch in, chúng em sẽ tiến hành làm mạch. Chúng em lựa chọn đi dây mặt dưới và câu dây đồng ở mặt trên vì chi phí tiết kiệm và không ảnh hưởng đến mạch quá nhiều.

Tiếp đến sẽ là phần hàn linh kiện lên mạch, phải sử dụng nhiều kỹ thuật hàn linh kiện bé như chân dán, chân cắm, mối hàn bé,... không để hàn lâu nóng ảnh hưởng đến linh kiện chịu quá nhiệt hỏng đặc biệt là chip STM32F103C8T6.

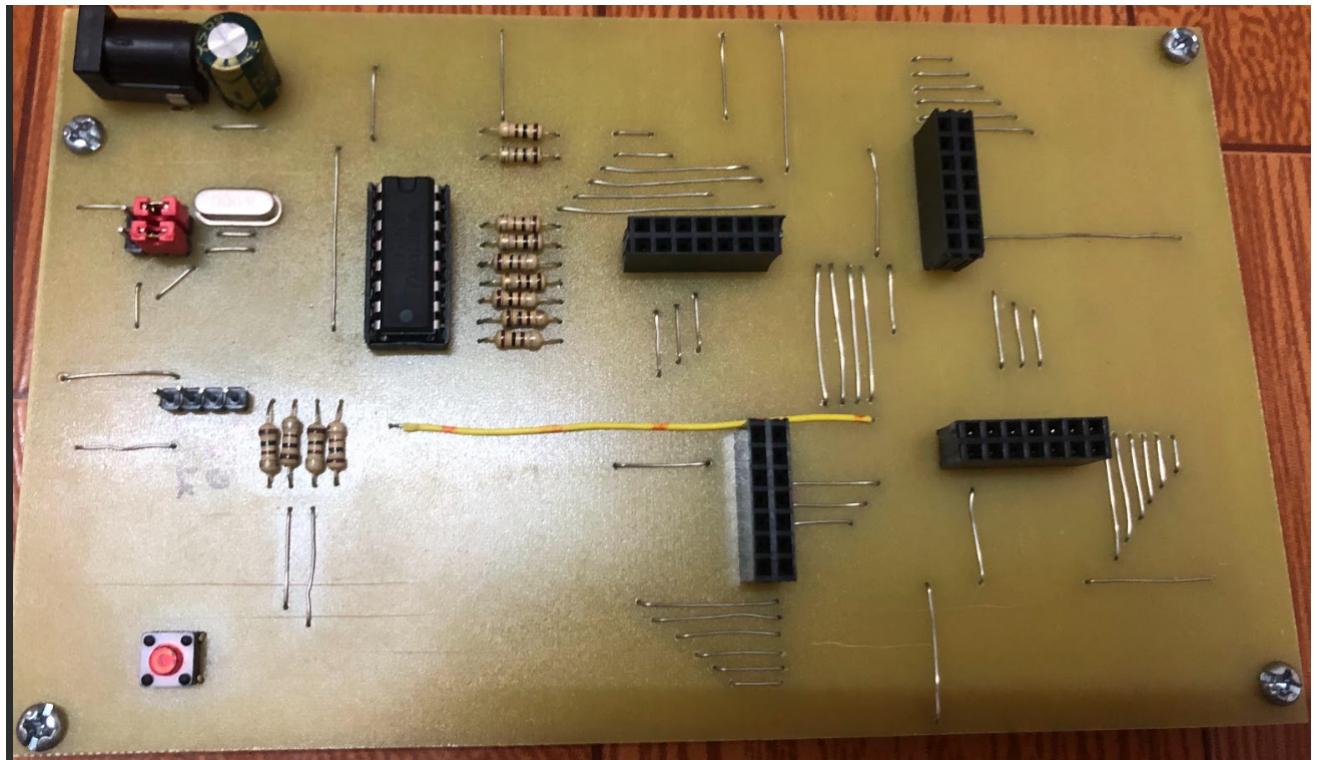
Kết quả thu được mạch:



Hình 3. 7 Mặt trước và mặt sau của khối hiển thị



Hình 3. 8 Mặt sau của sản phẩm



Hình 3. 9 Mặt trước của sản phẩm

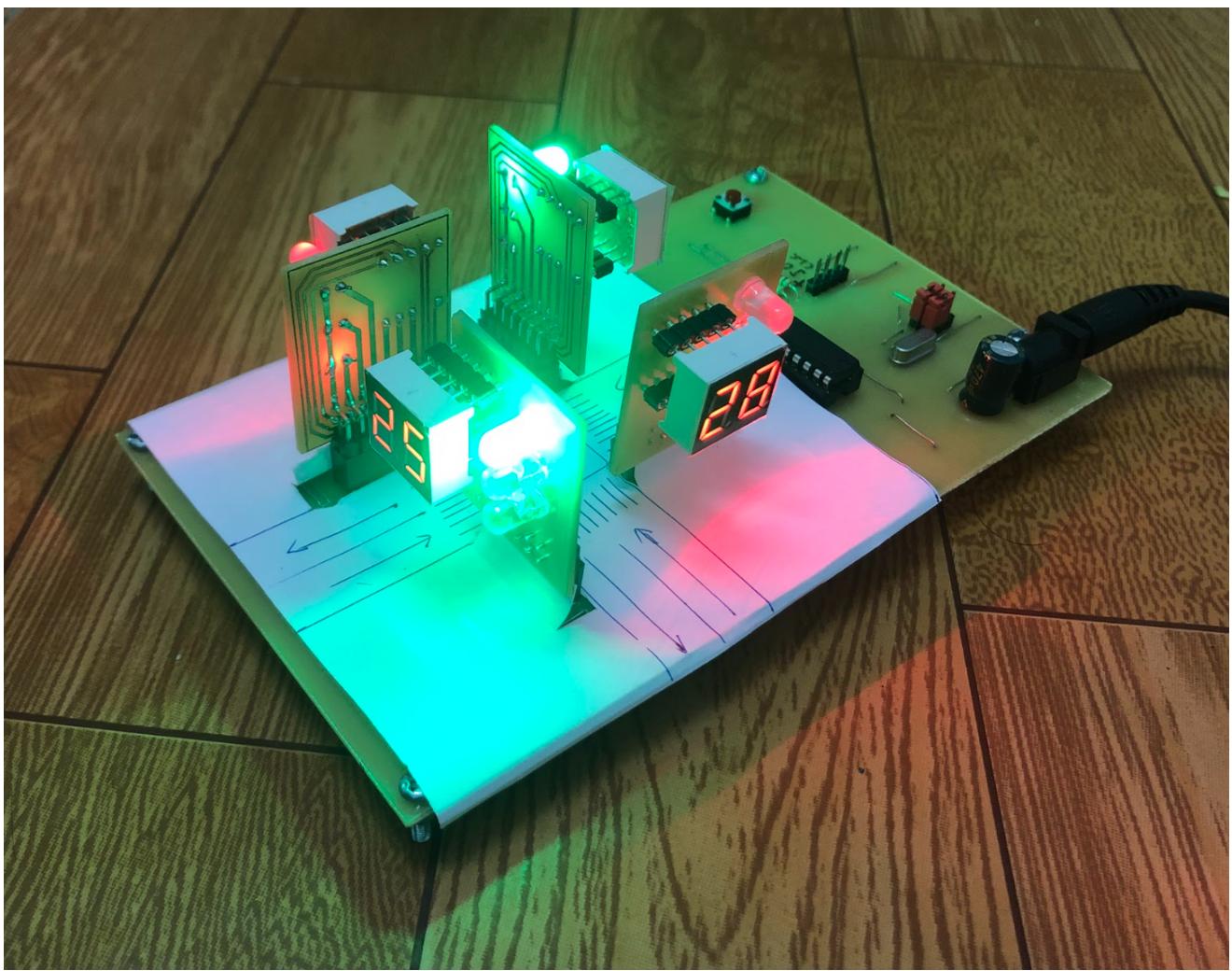
### 3. Chạy thử nghiệm mạch, kết quả

Môi trường:

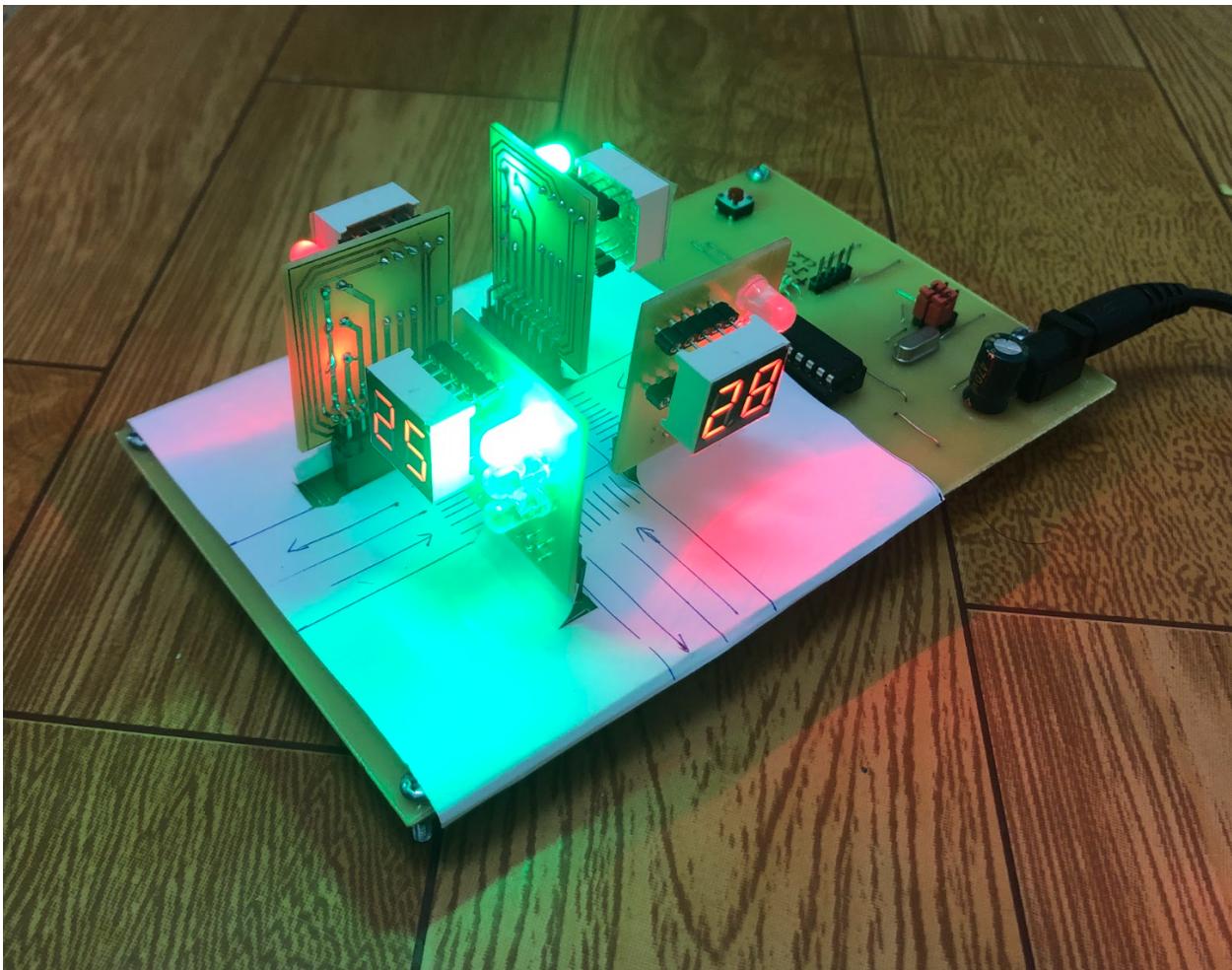
- Phòng ngủ, nhiệt độ 25 độ C.
- Thời gian chạy lần 1 là 1 phút, lần 2 là 5 phút.

Đầu vào: Cáp nguồn 5VDC qua jack DC bằng Adapter.

Kết quả:



Hình 3. 10 Kết quả sau khi chạy thử lần 1



Hình 3. 11 Kết quả sau khi chạy thử lần 2

- Đã hoàn thiện được sản phẩm: đi dây thành công, dựng lên được mô hình quen thuộc thực tế.
- Mạch hoạt động ổn định:
  - o Sáng rõ nét tất cả các đèn, không xảy ra mờ nét, đèn hỏng.
  - o Các IC và chip STM32F103C8T6 hoạt động ở nhiệt độ ổn định (không có hiện tượng nóng lên) sau thời gian dài.
  - o Các chân nạp như ST Link, Boot, nút reset đều hoạt động bình thường.
  - o Bảo quản mạch nơi khô ráo và cho chạy mạch 5 ngày liên tục mỗi ngày 10 phút và tăng dần thì mạch đều hoạt động ổn định, không có hiện tượng hỏng linh kiện, oxi hóa dây đồng.

## **4. Đánh giá**

### **4.1 Nhữnđiềuđạtđượct**

- Đáp ứng được cơ bản các chỉ tiêu đề ra ban đầu về sản phẩm
- Mạch nhỏ gọn, đi dây rõ ràng, khoa học.
- Mạch có độ bền cao, không bị oxi hóa dây đồng.
- Mạch tích hợp được có chân nạp ST Link, boot, reset.
- Có thể cấp nguồn theo nhiều cách.
- Mô hình đẹp, có tính thân thuộc với thực tế.
- Kết hợp được việc sử dụng nhiều linh kiện với nhau để hoàn thiện nên mạch.
- Hàn mạch với mối chân hàn bé với những linh kiện dán, cắm cỡ bé,...
- Đảo bảo được nhưng tiêu chuẩn về an toàn kỹ thuật.
- Sự dụng được chip STM32F103C8T6 mà không cần đến kit phát triển blue pill.

### **4.2 Nhữnđiềuchưalàmdượct, hạn ché**

- Chưa hiểu kỹ về kiến thức nên một số bước, công đoạn phải thực hiện lại nhiều lần gây mất thời gian.
- Thiếu sự hiểu ý giữa các thành viên giai đoạn đầu khiến việc triển khai đè tài bài tập lớn gặp đôi chút khó khăn.
- Do mới lần đầu tiếp xúc STM32F103C8T6, mà nó lại là một dòng vi xử lý đòi hỏi nhiều kiến thức và thời gian tìm hiểu nên còn gặp nhiều khó khăn.

## **5. Hướng phát triển**

Với hướng tiếp tục phát triển định hướng đèn giao thông: tích hợp sản phẩm sử dụng thêm các vi điều khiển, ngoại vi khác để kết hợp và xây dựng mô hình nhiều chức năng hơn như:

- Có thể thêm các nút nhấn chức năng ở trên mạch để có thể đi vào lựa chọn nhiều chế độ như chỉnh số giây mặc định của mỗi đèn màu.
- Kết hợp với các vi điều khiển khác như ESP8266, ESP32,... để đưa dữ liệu lên web, app. Điều khiển số giây, các chế độ, tốc độ bằng điện thoại hoặc laptop.
- Đưa thêm các cảm biến, thu thập, dùng các công nghệ như AI, IoT,..để nhận diện được các sự kiện như có vật thể kích thước lạ ở giữa đường và gửi thông tin về máy chủ.

Với hướng phát triển sử dụng dòng STM32: sử dụng STM32F103C8T6 nói riêng và dòng STM32 nói chung vào các sản phẩm thực tiễn, đa dạng hơn nữa. Làm việc với nhiều tầng cấu trúc để cho ra các sản phẩm lớn, tích hợp, nhiều ứng dụng mang lại sự hiệu quả và thú vị như dùng trong công nghiệp, nông nghiệp, môi trường lớn,...

## **6. Tổng kết chương**

Qua chương 3, chúng em đã xây dựng được quy trình từ việc chạy thành công mô phỏng trên proteus đến việc hoàn thiện sản phẩm thông qua các bước như: thử nghiệm trên board trắng, vẽ mạch trên altium, hàn mạch, chạy thử,...

Bên cạnh đó, đã đánh giá được kết quả của sản phẩm, nhìn nhận lại những điều đã đạt được và chưa đạt được từ đó đề xuất được các định hướng phát triển sản phẩm trong tương lai.

## KẾT LUẬN BÁO CÁO

Qua bài tập lớn môn kỹ thuật vi xử lý với đề tài sử dụng chip Arm Cotex để ứng dụng trong điều khiển của thầy. Nhóm em đã lên những ý tưởng ngay từ ban đầu để đi đến chọn lựa đề tài cụ thể là làm về mô hình điều khiển đèn giao thông sử dụng STM32F1. Với những kiến thức tích lũy từ trước cộng thêm những kiến thức quan trọng của môn học đã giúp chúng em có cái nhìn tổng quan về đề tài, biết các khái quát, tìm hiểu, triển khai về một vấn đề. Hiểu rõ và xây dựng bộ khung thiết kế chuẩn kỹ thuật từ giai đoạn ý tưởng đến hình thành nên sản phẩm. Được tiếp cận, hiểu biết thêm về dòng chip xử lý Arm Cotex nói chung và STM32F103C8T6 nói riêng. Với những kiến thức tổng quan chung nhất. Tất nhiên, việc làm bài tập lớn về đề tài này nó chỉ giúp chúng em có cái tiếp xúc sơ khai nhất về dòng chip của hàng ST này chứ chưa thể hiểu sâu, rộng được những kiến thức hay, thú vị khác về dòng chip này do thời gian có giới hạn.

Bên cạnh đó, qua bài tập lớn của môn học còn giúp chúng em hoàn thiện hơn nữa kỹ năng làm việc nhóm: phia chia công việc và thời hạn, bàn luận, phản biện, phân chia công việc theo cá nhân, phân chia công việc chung cả nhóm,...chia sẻ những kiến thức, kinh nghiệm với nhau để cùng nhau hoàn thiện tốt bài tập lớn.

Khoảng thời gian trải qua môn học trong suốt học kỳ dài 4 tháng này đã giúp cho mỗi cá nhân trong nhóm có cho riêng mình những kinh nghiệm quý báu khác nhau. Góp phần ý nghĩa cho việc hoàn thiện các kỹ năng chuyên môn, kỹ năng mềm cho bản thân trong chặng đường còn đầy thử thách phía trước.

## TÀI LIỆU THAM KHẢO

- [1] <https://deviot.vn/tutorials/stm32f1.23165131/huong-dan-cai-dat-stm32-cubemx-va-keil-c.04048071>
- [2] <https://developer.arm.com/Processors/Cortex-M3>
- [3] [https://www.alldatasheet.com/view.jsp?Searchword=Stm32f103c8t6&gclid=Cj0KCQjwwtWgBhDhARIsAEMcxえA6uYi-IWBHJPVtQuEAscRX8-IRIfqsoWucGxI3W-d4dYHMLCY3vOMaArN0EALw\\_wkB](https://www.alldatasheet.com/view.jsp?Searchword=Stm32f103c8t6&gclid=Cj0KCQjwwtWgBhDhARIsAEMcxえA6uYi-IWBHJPVtQuEAscRX8-IRIfqsoWucGxI3W-d4dYHMLCY3vOMaArN0EALw_wkB)
- [4] <https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html>
- [5]  
<https://dammedientu.vn/proteus-8-7-full-key-full-huong-dan-download-va-cai-dat-moi-nhat>
- [6] <https://dientuviet.com/tai-lieu-huong-dan-su-dung-altium-designer/>
- [7] [https://www.alldatasheet.com/view.jsp?Searchword=Datasheet%207447&gclid=Cj0KCQjwwtWgBhDhARIsAEMcxえBFDgW5Pr2Fm1LeggwNz56\\_vd-R3ghq0AmUwPPgJ9FGA5EygbLWpAwaAomREALw\\_wkB](https://www.alldatasheet.com/view.jsp?Searchword=Datasheet%207447&gclid=Cj0KCQjwwtWgBhDhARIsAEMcxえBFDgW5Pr2Fm1LeggwNz56_vd-R3ghq0AmUwPPgJ9FGA5EygbLWpAwaAomREALw_wkB)
- [8] [https://www.alldatasheet.com/view.jsp?Searchword=Ams1117%20datasheet&gclid=Cj0KCQjwwtWgBhDhARIsAEMcxえBGvX\\_ze02WroUA3\\_CzWqk6I4ofU4Dw2YuGA03GOdiyOFSoR1GlME8aAuhEALw\\_wkB](https://www.alldatasheet.com/view.jsp?Searchword=Ams1117%20datasheet&gclid=Cj0KCQjwwtWgBhDhARIsAEMcxえBGvX_ze02WroUA3_CzWqk6I4ofU4Dw2YuGA03GOdiyOFSoR1GlME8aAuhEALw_wkB)
- [9]  
[https://www.infineon.com/dgdl/Infineon-LED7SEG\\_User\\_Module-Software%20Module%20Datasheets-v01\\_02-EN.pdf?fileId=8ac78c8c7d0d8da4017d0fa9f7d51aa](https://www.infineon.com/dgdl/Infineon-LED7SEG_User_Module-Software%20Module%20Datasheets-v01_02-EN.pdf?fileId=8ac78c8c7d0d8da4017d0fa9f7d51aa)
- [10] <https://embedds.com/programming-stm32f10x-io-port-pins/>
- [11] System-on-Chip Design with Arm® Cortex®-M Processors: Reference Book
- [12] Embedded Systems: Introduction to Arm® Cortex™-M Microcontrollers , Fifth Edition
- [13] STM32 Arm Programming for Embedded Systems (Mazidi & Naimi ARM)