BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA ĐIỆN-ĐIỆN TỬ

BÁO CÁO TỔNG KẾT
HỆ THỐNG NHÚNG

ĐỀ TÀI ĐÈN GIAO THÔNG SỬ DỤNG PIC 16F877A

Giảng viên hướng dẫn: **VÕ THIỆN LĨNH**

Nhóm: : **13**

Sinh viên thực hiện :**VÕ VĂN TUẤN**

MSSV : **6251020094**

Lớp : **Kỹ thuật Điện tử - Viễn thông**

Khóa : **62**

TP.HCM, ngày 5, tháng 1, năm 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA ĐIỆN-ĐIỆN TỬ

BÁO CÁO TỔNG KẾT
HỆ THỐNG NHÚNG

ĐỀ TÀI ĐÈN GIAO THÔNG SỬ DỤNG PIC 16F877A

Giảng viên hướng dẫn: **VÕ THIỆN LĨNH**

Nhóm: : **13**

Sinh viên thực hiện : **VÕ VĂN TUẤN**

MSSV : **6251020094**

Lớp : **Kỹ thuật Điện tử - Viễn thông**

Khóa : **62**

TP.HCM, ngày 5, tháng 1, năm 2025

MỤC LỤC

LỜI CẢM ƠN

Trong thời đại công nghệ 4.0, hệ thống nhúng đóng vai trò quan trọng trong việc tự động hóa và tối ưu hóa hoạt động trong nhiều lĩnh vực, đặc biệt là giao thông thông minh. Hệ thống đèn giao thông hiện đại không chỉ điều tiết lưu lượng xe mà còn giảm thiểu ùn tắc và đảm bảo an toàn.

Xuất phát từ nhu cầu thực tế, em thực hiện đề tài **"Thiết kế hệ thống đèn giao thông tự động sử dụng vi điều khiển PIC16F877A"** với mục tiêu xây dựng một hệ thống tự động hóa hiệu quả, có khả năng điều khiển tín hiệu giao thông chính xác và đồng bộ. Hệ thống được thiết kế để hoạt động theo chu kỳ thời gian cố định, hoặc tùy chỉnh dựa trên dữ liệu thực tế từ cảm biến lưu lượng xe.

Hệ thống sử dụng các thành phần như vi điều khiển PIC16F877A, đèn LED tín hiệu giao thông, nút nhấn và các cảm biến (nếu cần). Chương trình điều khiển được lập trình để đảm bảo các đèn tín hiệu hoạt động chính xác, dễ dàng tùy chỉnh và mở rộng.

Báo cáo này sẽ trình bày quy trình thiết kế, lập trình và thử nghiệm hệ thống. Em kỳ vọng giải pháp này không chỉ hỗ trợ quản lý giao thông hiệu quả mà còn là nền tảng cho các ứng dụng giao thông thông minh trong tương lai.

LỜI MỞ ĐẦU

Trong thời đại công nghệ 4.0, hệ thống nhúng đang trở thành nền tảng quan trọng trong việc phát triển các giải pháp tự động hóa và tối ưu hóa hoạt động trong nhiều lĩnh vực. Với khả năng tích hợp phần cứng và phần mềm, hệ thống nhúng được ứng dụng rộng rãi trong các lĩnh vực như giáo dục, công nghiệp, và đời sống hàng ngày. Nhờ vào những tiến bộ trong công nghệ vi điều khiển, các hệ thống nhúng ngày càng trở nên thông minh, tiết kiệm năng lượng và chi phí, góp phần nâng cao chất lượng cuộc sống và hiệu suất hoạt động.

Xuất phát từ nhu cầu thực tế và tiềm năng ứng dụng của hệ thống nhúng, em triển khai đề tài "Thiết kế mạch đồng hồ chuông báo điểm giờ cho trường học sử dụng PIC16F877A." Đề tài hướng đến việc xây dựng một hệ thống tự động hóa đơn giản nhưng hiệu quả, giúp nhà trường quản lý thời gian học tập một cách chính xác và đồng bộ. Hệ thống được thiết kế để hiển thị thời gian trên màn hình và tự động kích hoạt chuông báo theo các mốc thời gian đã lập trình trước, nhờ vào khả năng xử lý mạnh mẽ của vi điều khiển PIC16F877A.

Với sự kết hợp giữa phần cứng và phần mềm, hệ thống không chỉ đảm bảo độ chính xác cao mà còn dễ dàng tùy chỉnh theo nhu cầu thực tế. Báo cáo này sẽ trình bày chi tiết từ lý thuyết cơ sở, quy trình thiết kế mạch điện và lập trình vi điều khiển, đến việc thử nghiệm và đánh giá hiệu quả của hệ thống. Em hy vọng rằng giải pháp này sẽ mang lại sự tiện lợi và hiệu quả cao trong việc quản lý thời gian tại các trường học, đồng thời khẳng định vai trò quan trọng của hệ thống nhúng trong việc cải thiện đời sống và công việc.

PHẦN 1: TỔNG QUAN LÝ THUYẾT

1.1 LÝ DO CHỌN ĐỀ TÀI

Giao thông là một vấn đề quan trọng ảnh hưởng trực tiếp đến đời sống, kinh tế và xã hội. Trong bối cảnh đô thị hóa nhanh chóng, việc quản lý giao thông hiệu quả là thách thức lớn đối với các thành phố. Các hệ thống đèn giao thông truyền thống thường hoạt động theo chu kỳ cố định, không đáp ứng linh hoạt trước tình trạng lưu lượng xe thay đổi, dẫn đến ùn tắc, lãng phí thời gian và năng lượng.

Xuất phát từ thực tế này, em lựa chọn đề tài **"Thiết kế hệ thống đèn giao thông tự động sử dụng vi điều khiển PIC16F877A"** với mong muốn tạo ra một giải pháp tự động hóa đơn giản nhưng hiệu quả, giúp cải thiện việc điều tiết giao thông.

Hệ thống được xây dựng nhằm:

- Tự động điều khiển tín hiệu giao thông chính xác, đồng bộ.

- Dễ dàng mở rộng và tích hợp cảm biến lưu lượng để điều chỉnh linh hoạt theo tình hình thực tế.

- Ứng dụng công nghệ nhúng, giúp tối ưu hóa chi phí, tiết kiệm năng lượng và nâng cao độ tin cậy.

- Đề tài không chỉ có giá trị học thuật trong việc nghiên cứu, thiết kế và lập trình hệ thống nhúng, mà còn mang lại ý nghĩa thực tiễn trong việc góp phần giải quyết vấn đề giao thông hiện nay. Em hy vọng rằng đề tài này sẽ là tiền đề cho các giải pháp giao thông thông minh trong tương lai.

1.2 MỤC ĐÍCH THIẾT KẾ

Đề tài **"Thiết kế hệ thống đèn giao thông tự động sử dụng vi điều khiển PIC16F877A"** được thực hiện với các mục đích chính sau:

Tự động hóa hệ thống giao thông:

Thiết kế hệ thống đèn giao thông hoạt động chính xác, tự động chuyển đổi tín hiệu theo thời gian định trước.

Đảm bảo việc điều tiết giao thông hiệu quả, giảm thiểu ùn tắc và tăng cường an toàn giao thông.

Tích hợp công nghệ nhúng:

Áp dụng công nghệ vi điều khiển PIC16F877A để xây dựng hệ thống linh hoạt, dễ lập trình và mở rộng.

Tạo ra một giải pháp giao thông hiện đại, có khả năng tích hợp thêm cảm biến và các công nghệ thông minh trong tương lai.

Tối ưu hóa chi phí và năng lượng:

Thiết kế hệ thống đơn giản, chi phí thấp nhưng đảm bảo tính ổn định và độ bền.

Tiết kiệm năng lượng thông qua các linh kiện tối ưu và thuật toán điều khiển hiệu quả.

Ứng dụng thực tiễn:

Hỗ trợ việc quản lý giao thông tại các giao lộ nhỏ hoặc khu vực dân cư.

Góp phần nâng cao chất lượng sống và cải thiện hạ tầng giao thông trong bối cảnh đô thị hóa.

Thông qua đề tài này, em mong muốn áp dụng kiến thức đã học vào thực tiễn, đồng thời thể hiện vai trò của hệ thống nhúng trong việc giải quyết các vấn đề giao thông hiện đại.

1.3 PHẠM VI VÀ PHƯƠNG PHÁP THIẾT KẾ

1.3.1 Phạm vi thiết kế

Hệ thống điều khiển đèn giao thông cơ bản:Thiết kế hệ thống đèn giao thông cho một ngã tư tiêu chuẩn, bao gồm ba màu đèn (đỏ, vàng, xanh).Thời gian chuyển đổi tín hiệu được lập trình cố định theo các chu kỳ định trước.Sử dụng vi điều khiển PIC16F877A làm trung tâm điều khiển.Tích hợp các linh kiện cơ bản như đèn LED (đại diện cho đèn giao thông), nút nhấn, và nguồn cung cấp điện ổn định.Lập trình vi điều khiển bằng ngôn ngữ C để thực hiện các chức năng chuyển đổi tín hiệu và điều khiển đèn giao thông.Thiết kế chương trình đơn giản, dễ hiểu để có thể mở rộng hoặc thay đổi trong tương lai.Hệ thống mô phỏng trong môi trường thí nghiệm (sử dụng mạch thực tế hoặc phần mềm Proteus).Hướng tới ứng dụng trong giao lộ nhỏ hoặc các khu vực giao thông có lưu lượng xe vừa phải.

1.3.2 Phương pháp thiết kế

Nghiên cứu nguyên lý hoạt động của hệ thống đèn giao thông thông thường.Xác định các yêu cầu cơ bản như: thời gian sáng đèn đỏ, vàng, xanh, điều kiện chuyển đổi tín hiệu và sự đồng bộ giữa các làn đường.Xây dựng sơ đồ khối tổng thể gồm các thành phần chính:Vi điều khiển PIC16F877A làm trung tâm điều khiển.Đèn LED đại diện cho các tín hiệu đèn giao thông (đỏ, vàng, xanh).Nút nhấn giả lập yêu cầu ưu tiên (nút bấm qua đường).Nguồn cung cấp và các linh kiện điện tử hỗ trợ.Sử dụng phần mềm Proteus để mô phỏng mạch điện.Lựa chọn linh kiện phù hợp: LED, điện trở, nút nhấn, nguồn điện, vi điều khiển.Thiết kế mạch trên breadboard hoặc PCB để triển khai thực tế.Viết mã lệnh điều khiển vi điều khiển bằng ngôn ngữ C trên phần mềm MPLAB hoặc MikroC.Cấu trúc chương trình được thiết kế gồm các

phần chính:Thiết lập thời gian cho các tín hiệu đèn giao thông (chu kỳ đèn đỏ, vàng, xanh).Điều kiện kiểm tra và xử lý nút nhấn ưu tiên (cho người đi bộ).Đồng bộ hóa hoạt động của các làn đường.

1.4 NHIỆM VỤ THIẾT KẾ

1.4.1 Phân tích và chọn lựa linh kiện

PIC16F877A: Đây là vi điều khiển chính để điều khiển các tín hiệu đèn giao thông.Nguồn DC (5V): Có thể sử dụng mạch nguồn ổn áp (như LM7805) để cấp nguồn cho vi điều khiển và các linh kiện như Pin hoặc Adapter 9-12V: Làm nguồn đầu vào cho mạch ổn áp.LED đỏ, vàng, xanh lá cây: Sử dụng LED đơn hoặc LED module 3 màu cho các tín hiệu đèn giao thông.Điện trở hạn dòng: Thường từ 220Ω đến 1kΩ, tùy thuộc vào loại LED.Transistor NPN/PNP (như BC547, 2N2222) hoặc MOSFET (nếu điều khiển LED công suất cao).Diode bảo vệ (nếu có relay hoặc tải cảm).Thạch anh và tụ ổn địnhThạch anh 20MHz: Tạo xung nhịp cho PIC16F877A.Tụ gốm 22pF: Dùng với thạch anh.Nút nhấn dùng để giả lập các tình huống như bấm nút đi bộ.Điện trở pull-up hoặc pull-down (10kΩ): Đảm bảo tín hiệu ổn định khi nút nhấn không được bấm.PICkit3 hoặc PICkit4 dùng để nạp chương trình vào PIC16F877A. Module thời gian thực (RTC) (tùy chọn)N nếu cần điều khiển chính xác thời gian, bạn có thể sử dụng module DS1307 hoặc DS3231.Bộ hiển thị LED 7 đoạn hiển thị thời gian đếm ngược hoặc trạng thái đèn giao thông. Phần mềm lập trình viết và biên dịch mã cho PIC16F877A.

1.4.2 Thiết lập và tích hợp phần cứng

Xây dựng mạch điện hoàn chỉnh, tích hợp các thành phần như PIC16F877A, RTC, và led 7 đoạn . Đảm bảo các kết nối giữa linh kiện chính xác, ổn định và cung cấp nguồn điện phù hợp cho toàn hệ thống.

1.4.3 Phát triển phần mềm điều khiển

Lập trình vi điều khiển PIC16F877A để thực hiện các chức năng chính như đọc thời gian từ RTC, thiết kế thuật toán, hiển thị thời gian đếm ngược. Phần mềm được viết bằng ngôn ngữ C và phát triển trên nền tảng CCS C Compiler.

1.5 GIỚI THIỆU VỀ HỆ THỐNG NHÚNG VÀ ỨNG DỤNG ĐIỀU KHIỂN

Hệ thống nhúng là sự kết hợp giữa phần cứng và phần mềm trong các thiết bị điện tử, được thiết kế để thực hiện các chức năng chuyên biệt với độ chính xác và hiệu quả cao. Nhờ khả năng tích hợp và vận hành độc lập, hệ thống nhúng đang được ứng dụng rộng rãi trong nhiều lĩnh vực như tự động hóa, giáo dục, y tế, và công nghiệp. Trong nghiên cứu này, hệ thống nhúng sử dụng vi điều khiển PIC16F877A đóng vai trò cốt lõi trong việc xây dựng mạch đồng hồ chuông báo điểm giờ tự động. PIC16F877A được lựa chọn nhờ tính ổn định, dễ lập trình và khả năng kiểm soát chính xác thời gian, phù hợp với các yêu cầu của hệ thống quản lý thời gian trong trường học hoặc các môi trường làm việc đòi hỏi sự đồng bộ hóa.

1.6 GIỚI THIỆU VỀ VI ĐIỀU KHIỂN 16F877A:

1.6.1 Chức năng và Cấu tạo

PIC16F877A là một Vi điều khiển PIC 40 chân và được sử dụng hầu hết trong các dự án và ứng dụng nhúng. Nó có năm cổng bắt đầu từ cổng A đến cổng E. Nó có ba bộ định thời trong đó có 2 bộ định thời 8 bit và 1 bộ định thời là 16 Bit. Nó hỗ trợ nhiều giao thức giao tiếp như giao thức nối tiếp, giao thức song song, giao thức I2C. PIC16F877A hỗ trợ cả ngắt chân phần cứng và ngắt bộ định thời.
Vi điều khiển này tích hợp nhiều tính năng hữu ích, bao gồm:

Bộ vi xử lý mạnh mẽ: Được trang bị lõi RISC với 35 lệnh tập lệnh, PIC16F877A hoạt động ở tốc độ tối đa 20 MHz, mang lại hiệu suất xử lý ổn định và nhanh chóng trong các ứng dụng điều khiển thời gian thực.

RAM và EEPROM: Có sẵn 368 bytes RAM và 256 bytes EEPROM, PIC16F877A hỗ trợ lưu trữ tạm thời và lưu trữ dữ liệu không bị mất khi tắt nguồn.

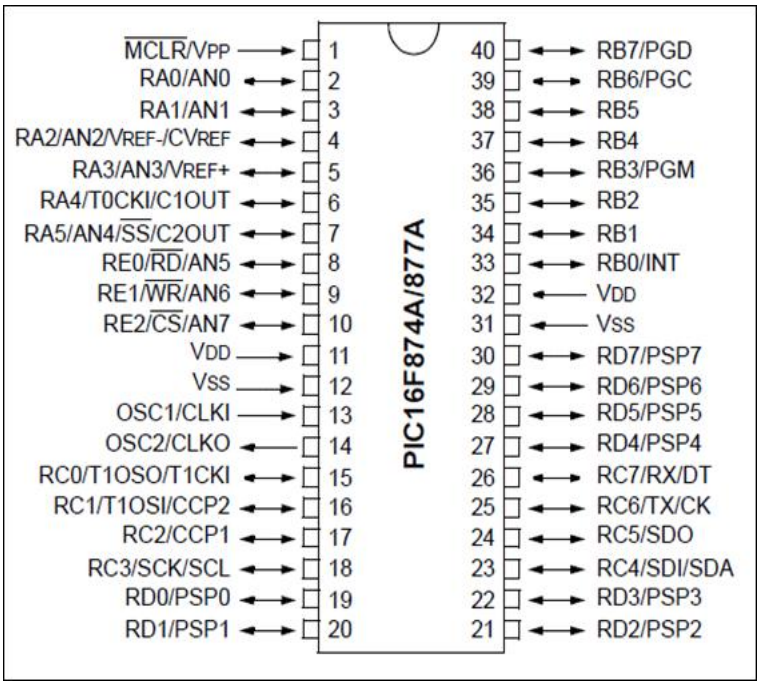
Bộ nhớ Flash: Với 14 KB bộ nhớ Flash, vi điều khiển này cung cấp dung lượng đủ lớn để lưu trữ chương trình điều khiển phức tạp.

Các chân I/O đa dụng: Tích hợp 33 chân GPIO, PIC16F877A hỗ trợ kết nối với nhiều loại cảm biến và thiết bị ngoại vi, mang lại sự linh hoạt trong thiết kế hệ thống nhúng.

Các giao thức giao tiếp: Hỗ trợ các giao thức UART, SPI, I2C, và các bộ chuyển đổi ADC 10-bit (tối đa 8 kênh), vi điều khiển này dễ dàng tích hợp vào các hệ thống đo lường, điều khiển, và giám sát.

PWM và tính năng đặc biệt: PIC16F877A còn cung cấp tính năng PWM, bộ định thời, và tính năng CCP (Capture/Compare/PWM), giúp xử lý tín hiệu và điều khiển chính xác trong các ứng dụng tự động hóa.

Hình 1.1 Hình ảnh Pic16F877A



Hình 1.1.1 Hình ảnh Pic16f877A

Cấu tạo:

- Chân 1: MCLR là chân clear của mạch này. Nó sẽ khởi động lại vi điều khiển và được kích hoạt bởi mức logic thấp, có nghĩa là chân này phải được cấp liên tục một điện áp 5V và nếu cấp điện áp 0V thì PIC16F877A sẽ bị đặt lại.
- Chân 2 RA0/AN0: PORT A có 6 chân, từ chân số 2 đến chân số 7. Tất cả đều là các chân xuất, nhập dữ liệu hai chiều. Chân số 2 là chân đầu tiên của PORT A. Chân này có thể được sử dụng như một chân tương tự (analog) chân AN0. Nó được tích hợp bộ chuyển đổi analog sang digital.
- Chân 3 RA1/AN1: Đầu vào tín hiệu analog 1
- Chân 4 RA2/AN2/Vref-: Có thể hoạt động như đầu vào analog thứ 2 hoặc chân điện áp tham chiếu âm.
- Chân 5 RA3/AN3/Vref+: Có thể hoạt động như đầu vào analog thứ 3 hoặc chân điện áp tham chiếu dương.
- Chân 6 RA0/T0CKI: Với timer 0, chân này hoạt động được như một đầu vào xung clock và đầu ra open drain.
- Chân 7 RA5/SS/AN4: Có thể hoạt động như một đầu vào analog thứ 4. Có cổng nối tiếp đồng bộ và là chân SS cho cổng này.
- Chân 8 RE0/RD/AN5: PORT E bắt đầu từ chân số 8 đến chân số 10 và là cổng I/O hai chiều. Nó còn là cổng analog thứ 5 hoặc là chân RD (tích cực mức logic thấp) cho cổng slave giao tiếp song song

- Chân 9 RE1/WR/AN6: Là đầu vào analog thứ 6 và là chân WR (tích cực mức logic thấp) cho cổng slave giao tiếp song song.
Chân 10 RE2/CS/A7: Là đầu vào analog 7 và là chân CS cho cổng slave song song.
- Chân 11 và 32 VDD: Đây là hai chân cấp nguồn 5V.
- Chân 12 và 31 VSS: Các chân tham chiếu nối đất cho I/O và các chân logic. Chúng nên được nối với 0V hoặc mắc GND.
- Chân 13 OSC1/CLKIN: Là đầu vào bộ dao động hoặc chân đầu vào xung nhịp bên ngoài.
- Chân 14 OSC2/CLKOUT: Đây là chân đầu ra của bộ dao động. Một bộ dao động thạch anh được nối vào giữa hai chân 13 và 14 để cấp xung nhịp bên ngoài cho bộ vi điều khiển. ¼ tần số của OSC1 được OSC2 xuất ra trong chế độ RC. Điều này xác định tốc độ chu kỳ xử lý lệnh.
- Chân 15 RC0/T1OCO/T1CKI: PORT C có 8 chân. Là cổng I/O hai chiều. Trong số đó, chân 15 là chân đầu tiên. Nó có thể là đầu vào xung nhịp của bộ định thời 1 hoặc đầu ra bộ dao động của bộ định thời 2.
- Chân 16 RC11/T1OSI/CCP 2: Là đầu vào dao động của bộ định thời 1 hoặc đầu vào capture 2 / đầu ra so sánh 2 / đầu ra PWM 2.
- Chân 17 RC2/CCP 1: Đầu vào capture 1/ đầu ra so sánh 1/ đầu ra PWM1
- Chân 18 RC3/SCK/ SCL: Đầu ra của chế độ SPI hoặc I2C và có thể là I/O cho bộ dao động nối tiếp đồng bộ.
- Chân 23 RC4/SDI/SDA: Chân dữ liệu trong chế độ SPI hoặc là chân xuất nhập dữ liệu chế độ I2C.
- Chân 24 RC5/SDO: Là chân xuất dữ liệu chế độ SPI.
- Chân 25 RC6/TX/CK: Có thể là chân xung clock đồng bộ hoặc chân truyền không đồng bộ UART.
- Chân 26 RC7/RX/DT: Là chân dữ liệu đồng bộ hoặc chân nhận tín hiệu UART.
- Các chân 19, 20, 21, 22, 27, 28, 29, 30: Tất cả các chân này đều thuộc PORT D, đây là một cổng I/O hai chiều. Khi bus vi xử lý được kết nối, nó có thể hoạt động ncho cổng slave giao tiếp dữ liệu song song.
- Chân 33-40 PORT B: Hai chân này đều thuộc PORTB. Trong đó RB0 có thể được sử dụng làm chân ngắt ngoài và RB6 và RB7 có thể được sử dụng làm chân debugge.

1.6.2 Nguyên lý hoạt động và Điện áp của PIC16F877A

Nguyên lý hoạt động:

- Xử lý tín hiệu: PIC16F877A sử dụng lõi xử lý RISC (Reduced Instruction Set Computing) với 35 lệnh tập lệnh, cho phép thực hiện các tác vụ với tốc độ cao và hiệu quả.

- Bộ nhớ tích hợp:

Bộ nhớ Flash: Lưu trữ chương trình điều khiển với dung lượng 14 KB.

RAM: Cung cấp 368 bytes bộ nhớ tạm thời để xử lý dữ liệu.

EEPROM: Lưu trữ dữ liệu không bị mất khi tắt nguồn, với dung lượng 256 bytes.

- Điều khiển I/O:

Tích hợp 33 chân GPIO, cho phép kết nối với nhiều thiết bị ngoại vi như cảm biến, relay, hoặc màn hình hiển thị.

Có các bộ định thời (Timer) hỗ trợ điều khiển chính xác các tác vụ thời gian thực.

- Giao tiếp với thiết bị ngoại vi:

Hỗ trợ các giao thức UART, SPI, và I2C để giao tiếp với các module khác.

Bộ chuyển đổi ADC (Analog-to-Digital Converter) 10-bit, hỗ trợ đến 8 kênh, giúp đo lường tín hiệu tương tự.

- Chức năng PWM: PIC16F877A hỗ trợ Pulse Width Modulation (PWM), ứng dụng trong điều khiển động cơ, đèn LED, hoặc các thiết bị cần điều chỉnh tín hiệu đầu ra.

PIC16F877A hỗ trợ Pulse Width Modulation (PWM), ứng dụng trong điều khiển động cơ, đèn LED, hoặc các thiết bị cần điều chỉnh tín hiệu đầu ra.

- Ứng dụng đa năng: Được lập trình và điều khiển thông qua công cụ MPLAB IDE và bộ nạp PICkit, vi điều khiển này có thể sử dụng trong hệ thống tự động hóa, robot, hoặc các thiết bị đo lường

Điện áp hoạt động:

- Nguồn cung cấp:

PIC16F877A hoạt động ở mức điện áp từ 4.0V đến 5.5V.

Điện áp tiêu chuẩn thường là 5V để đảm bảo hiệu suất tối ưu.

- Dòng điện tiêu thụ: Tiêu thụ dòng điện thấp, đặc biệt khi hoạt động ở chế độ tiết kiệm năng lượng (Sleep Mode).

- Tương thích mức điện áp: PIC16F877A có thể giao tiếp với các thiết bị hoạt động ở mức điện áp 5V mà không cần mạch chuyển đổi mức điện áp.

- Khi kết nối với cảm biến hoặc thiết bị hoạt động ở điện áp thấp hơn, cần thêm mạch chuyển đổi để đảm bảo tính tương thích.

PHẦN II: THIẾT KẾ PHẦN MỀM

```
//code mô phỏng
#include <16f877a.h>
#fuses HS, NOWDT, NOPROTECT, NOLVP
#use delay(clock=20000000) // 20 MHz
#use rtos(timer=0, minor_cycle=10ms) // Chu k? nh? 10ms

// Các chân k?t nối LED 7-segment
#define SEG_A PIN_B0
#define SEG_B PIN_B1
#define SEG_C PIN_B2
#define SEG_D PIN_B3
#define SEG_E PIN_B4
#define SEG_F PIN_B5
#define SEG_G PIN_B6

#define DIGIT_1_LED1 PIN_D0
#define DIGIT_2_LED1 PIN_D1
#define DIGIT_1_LED2 PIN_C4
#define DIGIT_2_LED2 PIN_D3
efine d1 PIN_A0  // Đèn đỏ giao thông 1
#define v1 PIN_A1  // Đèn vàng giao thông 1
#define x1 PIN_A2  // Đèn xanh giao thông 1
#define d2 PIN_A3  // Đèn đỏ giao thông 2
#define v2 PIN_D4  // Đèn vàng giao thông 2
#define x2 PIN_A5  // Đèn xanh giao thông 2

#define BUTTON_MODE1 PIN_D6 // Nút chế độ 1
#define BUTTON_MODE2 PIN_D7 // Nút chế độ 2

// Mã hóa số trên LED 7-segment
unsigned int8 SEGMENT_CODES[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92,
0x82, 0xF8, 0x80, 0x90};

// Biến trạng thái
int1 mode = 1; // Mặc định chế độ 1
static int8 i1 = 10, i2 = 7; // Biến lưu trữ giá trị LED 7-segment

// Hàm hiện thị chữ số trên LED 7-segment
void display_digit(int8 digit, int8 value, int8 led) {
   // Tắt tất cả các digit
   output_low(DIGIT_1_LED1);
   output_low(DIGIT_2_LED1);
   output_low(DIGIT_1_LED2);
   output_low(DIGIT_2_LED2);
```

```
    // Hiện thị giá trị tương ứng trên LED 7-segment
    output_b(SEGMENT_CODES[value]);

    // Bật digit cần hiện thị
    if (led == 1) {
        if (digit == 0) output_high(DIGIT_1_LED1);
        else output_high(DIGIT_2_LED1);
    } else if (led == 2) {
        if (digit == 0) output_high(DIGIT_1_LED2);
        else output_high(DIGIT_2_LED2);
    }

    // Delay để nhìn thấy hiện thị
    delay_us(500);

    // Tắt tất cả các digit sau khi hiện thị
    output_low(DIGIT_1_LED1);
    output_low(DIGIT_2_LED1);
    output_low(DIGIT_1_LED2);
    output_low(DIGIT_2_LED2);
}

#task(rate=10ms)
void taskLED7() {
    if (mode) {
        static int16 counter = 0;    // Bộ đếm 10ms để tính thời gian
        static int flag = 0;

        counter++;

        if (counter >= 100) { // Mỗi 1 giây (100 * 10ms)
            counter = 0;

            if (i1 > 0) i1--;
            if (i2 > 0) i2--;
            if (i2 == 0 && i1 == 3) i2 = 3;
            if (i2 == 3 && i1 == 0) i1 = 3;
            if (i1 == 0 && i2 == 0) {
                flag = 1 - flag;
                if (flag == 1) { i2 = 10; i1 = 7;}
                else { i2 = 7; i1 = 10;}
            }
        }
        // Hiện thị các giá trị trên LED 7-segment
        display_digit(0, i1 / 10, 1); // Hiện thị chữ số hàng chục của LED 1
```

```
      display_digit(1, i1 % 10, 1); // Hiện thị chữ số hàng đơn vị của LED 1
      display_digit(0, i2 / 10, 2); // Hiện thị chữ số hàng chục của LED 2
      display_digit(1, i2 % 10, 2); // Hiện thị chữ số hàng đơn vị của LED 2
   }
   else { // Chế độ 2
      // Tắt LED 7-segment
      output_low(DIGIT_1_LED1);
      output_low(DIGIT_2_LED1);
      output_low(DIGIT_1_LED2);
      output_low(DIGIT_2_LED2);


   }
}

#task(rate=100ms)
void taskLEDGT() {
   static int8 state = 0;
   static int16 counter = 0;

   if (mode) { // Chế độ 1: Đèn giao thông hoạt động bình thường
      counter++;
      switch (state) {
         case 0:
            output_high(d1);
            output_low(v1);
            output_low(x1);
            output_low(d2);
            output_low(v2);
            output_high(x2);
            if (counter >= 70) { // 7 giây thực hiện
               counter = 0;
               state = 1;
            }
            break;
         case 1:
            output_high(d1);
            output_low(v1);
            output_low(x1);
            output_low(d2);
            output_high(v2);
            output_low(x2);
            if (counter >= 30) { // 3 giây thực hiện
               counter = 0;
               state = 2;
            }
```

```
            break;
         case 2:
            output_low(d1);
            output_low(v1);
            output_high(x1);
            output_high(d2);
            output_low(v2);
            output_low(x2);
            if (counter >= 70) { // 7 giây thực hiện
               counter = 0;
               state = 3;
            }
            break;
         case 3:
            output_low(d1);
            output_high(v1);
            output_low(x1);
            output_high(d2);
            output_low(v2);
            output_low(x2);
            if (counter >= 30) { // 3 giây thực hiện
               counter = 0;
               state = 0;
            }
            break;
      }
   } else { // Chế độ 2: Nháy đèn vàng
      counter++;
      if (counter >= 5) { // Nháy đèn vàng 0.5s
         counter = 0;
         output_toggle(v1);
         output_toggle(v2);
      }
      // Tắt các đèn khác
      output_low(d1);
      output_low(x1);
      output_low(d2);
      output_low(x2);
   }
}

#task(rate=10ms)
void taskBUTTON() {
   static int1 last_button_mode1 = 1;
   static int1 last_button_mode2 = 1;
```

```
    if (!input(BUTTON_MODE1) && last_button_mode1) {
        mode = 1;
        i1 = 10; // Gán LED 1 = 10 khi chuyển sang chế độ 1
        i2 = 7;  // Gán LED 2 = 7 khi chuyển sang chế độ 1
    }

    if (!input(BUTTON_MODE2) && last_button_mode2) {
        mode = 0;
    }

    last_button_mode1 = input(BUTTON_MODE1);
    last_button_mode2 = input(BUTTON_MODE2);
}

void main() {
    set_tris_d(0xC0); // Cấu hình RD6, RD7 là INPUT
    set_tris_b(0x00); // Các chân điều khiển LED 7-segment là OUTPUT
    set_tris_a(0x00); // Các chân điều khiển đèn giao thông là OUTPUT
    output_b(0x00);   // Tắt tất cả các chân B
    output_a(0x00);   // Tắt tất cả các chân A
    rtos_run();       // Chạy RTOS
}
```

Giải thích đoạn code:

2.1 KHAI BÁO

```
#include <16f877a.h>
```

```
#fuses HS, NOWDT, NOPROTECT, NOLVP
#use delay(clock=20000000) // 20 MHz
#use rtos(timer=0, minor_cycle=10ms) // Chu k? nh? 10ms
```

- **Thư viện:** 16f877a.h chứa các định nghĩa và hàm cho vi điều khiển PIC16F877A.
- **Cấu hình fuses:** HS (dao động cao tần), NOWDT (tắt Watchdog Timer), NOPROTECT (tắt bảo vệ mã), NOLVP (tắt lập trình điện áp thấp).
- **Tần số:** 20 MHz.
- **RTOS:** Cấu hình hệ điều hành thời gian thực, sử dụng timer 0 với chu kỳ nhỏ 10ms.

2.2 KHAI BÁO CHÂN ĐIỀU KHIỂN

```
#define SEG_A PIN_B0
#define SEG_B PIN_B1
#define SEG_C PIN_B2
#define SEG_D PIN_B3
#define SEG_E PIN_B4
#define SEG_F PIN_B5
#define SEG_G PIN_B6

#define DIGIT_1_LED1 PIN_D0
#define DIGIT_2_LED1 PIN_D1
#define DIGIT_1_LED2 PIN_C4
#define DIGIT_2_LED2 PIN_D3

#define d1 PIN_A0  // Đèn đỏ giao thông 1
#define v1 PIN_A1  // Đèn vàng giao thông 1
#define x1 PIN_A2  // Đèn xanh giao thông 1
#define d2 PIN_A3  // Đèn đỏ giao thông 2
#define v2 PIN_D4  // Đèn vàng giao thông 2
#define x2 PIN_A5  // Đèn xanh giao thông 2

#define BUTTON_MODE1 PIN_D6 // Nút chế độ 1
#define BUTTON_MODE2 PIN_D7 // Nút chế độ 2
```

- **Chân kết nối LED 7 :** B0 , B1, B2, B3, B4, B5, B6.
- **Chân dữ liệu LED 7:** D0, D1, C4, D3.
- **Chân kết nối LED giao thông:** A0, A1, A2, A3, D4, A5.
- **Chân kết nối nút bấm:** D6, D7.

2.3 MÃ HÓA SỐ TRÊN LED 7

```
unsigned int8 SEGMENT_CODES[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99,
0x92, 0x82, 0xF8, 0x80, 0x90};
```

Hàm hiện thị chữ số trên LED 7-segment

```
void display_digit(int8 digit, int8 value, int8 led) {
   // Tắt tất cả các digit
   output_low(DIGIT_1_LED1);
   output_low(DIGIT_2_LED1);
   output_low(DIGIT_1_LED2);
   output_low(DIGIT_2_LED2);

   // Hiện thị giá trị tương ứng trên LED 7-segment
   output_b(SEGMENT_CODES[value]);

   // Bật digit cần hiện thị
   if (led == 1) {
      if (digit == 0) output_high(DIGIT_1_LED1);
      else output_high(DIGIT_2_LED1);
   } else if (led == 2) {
      if (digit == 0) output_high(DIGIT_1_LED2);
      else output_high(DIGIT_2_LED2);
   }
   // Delay để nhìn thấy hiện thị
   delay_us(500);
      output_low(DIGIT_1_LED1);
   output_low(DIGIT_2_LED1);
   output_low(DIGIT_1_LED2);
   output_low(DIGIT_2_LED2);
}
```

Hàm sẽ thực hiện lần lượt:

- **Tắt tất cả các digit**: Đảm bảo rằng không có digit nào được bật trước khi hiển thị chữ số mới.
- **Hiển thị giá trị tương ứng trên LED 7-segment**:Sử dụng mã hóa trong mảng SEGMENT_CODES để hiển thị chữ số value mong muốn
- **Bật digit cần hiển thị**: Bật digit cụ thể (digit 0 hoặc 1) của LED được chọn (LED 1 hoặc LED 2).
- **Delay** để nhìn thấy hiển thị.
- **Tắt tất cả các digit sau khi hiển thị**: Đảm bảo rằng tất cả các digit đều tắt sau khi hiển thị xong.

2.3.1 Task 1: Hàm đếm led 7

```
#task(rate=10ms)
void taskLED7() {
  if (mode) {
    static int16 counter = 0;    // Bộ đếm 10ms để tính thời gian
    static int flag = 0;
    counter++;
    if (counter >= 100) { // Mỗi 1 giây (100 * 10ms)
      counter = 0;
      if (i1 > 0) i1--;
      if (i2 > 0) i2--;
      if (i2 == 0 && i1 == 3) i2 = 3;
      if (i2 == 3 && i1 == 0) i1 = 3;
      if (i1 == 0 && i2 == 0) {
        flag = 1 - flag;
        if (flag == 1) { i2 = 10; i1 = 7;}
        else { i2 = 7; i1 = 10;}
      }
    }
    // Hiện thị các giá trị trên LED 7-segment
    display_digit(0, i1 / 10, 1); // Hiện thị chữ số hàng chục của LED 1
    display_digit(1, i1 % 10, 1); // Hiện thị chữ số hàng đơn vị của LED 1
    display_digit(0, i2 / 10, 2); // Hiện thị chữ số hàng chục của LED 2
    display_digit(1, i2 % 10, 2); // Hiện thị chữ số hàng đơn vị của LED 2
  }
```

```
    else { // Chế độ 2
      // Tắt LED 7-segment
      output_low(DIGIT_1_LED1);
      output_low(DIGIT_2_LED1);
      output_low(DIGIT_1_LED2);
      output_low(DIGIT_2_LED2);

      // Gán giá trị LED 7 thứ nhất = 10, LED 7 thứ hai = 7
      display_digit(0, 1, 1); // Hiện thị số 1 (chục) cho LED 7 thứ nhất
      display_digit(1, 0, 1); // Hiện thị số 0 (đơn vị) cho LED 7 thứ nhất
      display_digit(0, 0, 2); // Hiện thị số 0 (chục) cho LED 7 thứ hai
      display_digit(1, 7, 2); // Hiện thị số 7 (đơn vị) cho LED 7 thứ hai
    }
  }
```

Hàm sẽ thực hiện:

- **Chế độ 1 (mode = 1):**
  - **Bộ đếm thời gian:** Tăng giá trị counter mỗi 10ms.
  - **Giảm giá trị:** Khi counter đạt 1 giây (100 lần), giảm các giá trị i1 và i2.
  - **Đổi giá trị:** Nếu i1 và i2 đều là 0, chuyển đổi giá trị giữa 10-7 và 7-10.
  - **Hiển thị số:** Gọi hàm display_digit để hiện thị các số lên LED 7-segment.
- **Chế độ 2 (mode = 0):**
  - **Tắt LED:** Tắt tất cả các digit của LED 7-segment.
  - **Hiển thị giá trị cố định:** Hiển thị giá trị cố định 10 và 7 lên LED 7-segment.

2.3.2 Task 2: Hàm sáng đèn LED giao thông

```
#task(rate=100ms)
void taskLEDGT() {
  static int8 state = 0;
  static int16 counter = 0;
  if (mode) { // Chế độ 1: Đèn giao thông hoạt động bình thường
    counter++;
```

```
switch (state) {
   case 0:
      output_high(d1);
      output_low(v1);
      output_low(x1);
      output_low(d2);
      output_low(v2);
      output_high(x2);
      if (counter >= 70) { // 7 giây thực hiện
         counter = 0;
         state = 1;
      }
      break;
   case 1:
      output_high(d1);
      output_low(v1);
      output_low(x1);
      output_low(d2);
      output_high(v2);
      output_low(x2);
      if (counter >= 30) { // 3 giây thực hiện
         counter = 0;
         state = 2;
      }
      break;
   case 2:
      output_low(d1);
      output_low(v1);
      output_high(x1);
      output_high(d2);
      output_low(v2);
      output_low(x2);
      if (counter >= 70) { // 7 giây thực hiện
```

```
                counter = 0;
                state = 3;
            }
            break;
        case 3:
            output_low(d1);
            output_high(v1);
            output_low(x1);
            output_high(d2);
            output_low(v2);
            output_low(x2);
            if (counter >= 30) { // 3 giây thực hiện
                counter = 0;
                state = 0;
            }
            break;
    }
  } else { // Chế độ 2: Nháy đèn vàng
    counter++;
    if (counter >= 5) { // Nháy đèn vàng 0.5s
        counter = 0;
        output_toggle(v1);
        output_toggle(v2);
    }
    // Tắt các đèn khác
    output_low(d1);
    output_low(x1);
    output_low(d2);
    output_low(x2);
  }
}
```

Hàm sẽ thực hiện:

- **Chế độ 1 (mode = 1): Đèn giao thông hoạt động bình thường**

- Bộ đếm thời gian: Tăng giá trị counter mỗi 100ms.
- Chuyển đổi trạng thái: Dựa trên giá trị state, các đèn giao thông được bật và tắt tương ứng. Các trạng thái lần lượt chuyển đổi sau các khoảng thời gian xác định:
- Trạng thái 0 (7 giây): Đèn đỏ giao thông 1 (d1), đèn xanh giao thông 2 (x2).
- Trạng thái 1 (3 giây): Đèn đỏ giao thông 1 (d1), đèn vàng giao thông 2 (v2).
- Trạng thái 2 (7 giây): Đèn xanh giao thông 1 (x1), đèn đỏ giao thông 2 (d2).
- Trạng thái 3 (3 giây): Đèn vàng giao thông 1 (v1), đèn đỏ giao thông 2 (d2).

- **Chế độ 2 (mode = 0): Nháy đèn vàng**
  - Nháy đèn vàng: Đèn vàng (v1, v2) nháy mỗi 0.5 giây.
  - Tắt các đèn khác: Các đèn giao thông khác (d1, x1, d2, x2) đều tắt.

2.3.3 Task 3: hàm kiểm tra chế độ

```
#task(rate=10ms)
void taskBUTTON() {
   static int1 last_button_mode1 = 1;
   static int1 last_button_mode2 = 1;

   if (!input(BUTTON_MODE1) && last_button_mode1) {
      mode = 1;
      i1 = 10; // Gán LED 1 = 10 khi chuyển sang chế độ 1
      i2 = 7;  // Gán LED 2 = 7 khi chuyển sang chế độ 1
   }

   if (!input(BUTTON_MODE2) && last_button_mode2) {
      mode = 0;
   }
   last_button_mode1 = input(BUTTON_MODE1);
   last_button_mode2 = input(BUTTON_MODE2);
}
```

Hàm sẽ thực hiện:

- **Biến lưu trữ trạng thái nút:**
  - last_button_mode1 và last_button_mode2 lưu trạng thái trước đó của hai nút chế độ (BUTTON_MODE1 và BUTTON_MODE2).
- **Kiểm tra nút chế độ 1:**
  - Khi nút BUTTON_MODE1 được nhấn và trạng thái trước đó là 1, chuyển sang chế độ 1 (mode = 1). Đồng thời, gán giá trị i1 là 10 và i2 là 7.
- **Kiểm tra nút chế độ 2:**
  - Khi nút BUTTON_MODE2 được nhấn và trạng thái trước đó là 1, chuyển sang chế độ 2 (mode = 0).
- **Cập nhật trạng thái nút:**
  - Lưu trạng thái hiện tại của các nút BUTTON_MODE1 và BUTTON_MODE2 để sử dụng trong lần kiểm tra tiếp theo.

1. Void main (){}

```
void main() {
   set_tris_d(0xC0); // Cấu hình RD6, RD7 là INPUT
   set_tris_b(0x00); // Các chân điều khiển LED 7-segment là OUTPUT
   set_tris_a(0x00); // Các chân điều khiển đèn giao thông là OUTPUT
   output_b(0x00);   // Tắt tất cả các chân B
   output_a(0x00);   // Tắt tất cả các chân A
   rtos_run();       // Chạy RTOS
}
```

- **Cấu hình các chân I/O:**
  - set_tris_d(0xC0): Cấu hình các chân RD6 và RD7 làm INPUT (các chân còn lại làm OUTPUT).
  - set_tris_b(0x00): Cấu hình tất cả các chân cổng B làm OUTPUT (điều khiển LED 7-segment).
  - set_tris_a(0x00): Cấu hình tất cả các chân cổng A làm OUTPUT (điều khiển đèn giao thông).
- **Tắt tất cả các chân OUTPUT:**
  - output_b(0x00): Đặt tất cả các chân cổng B ở mức thấp (0).
  - output_a(0x00): Đặt tất cả các chân cổng A ở mức thấp (0).
- **Khởi động RTOS:**
  - rtos_run(): Khởi động hệ điều hành thời gian thực (RTOS), bắt đầu thực hiện các nhiệm vụ được cấu hình trước đó.

PHẦN III: THIẾT KẾ PHẦN CỨNG

3.1 CÁC LINH KIỆN ĐƯỢC SỬ DỤNG

3.1.1 Tụ gốm 22uF

Hình 1.1.2 Hình ảnh tụ gốm 22uF

Tụ gốm là một tụ điện có giá trị cố định, trong đó vật liệu gốm là chất điện môi. Nó được chế tạo từ hai hoặc nhiều lớp gốm sứ xen kẽ và một lớp kim loại hoạt động như các điện cực.

Tụ gốm là một thiết bị không phân cực, do đó bạn có thể nối nó trong mạch điện theo hướng nào cũng được. Vì lý do này nó an toàn hơn so với tụ hóa là tụ phân cực. Đó cũng chính là sự khác nhau giữa tụ gốm và tụ hóa. Nếu bạn để ý hai chân của tụ gốm sẽ thấy nó bằng nhau do nó không phân cực, còn tụ hóa có một chân dài một chân ngắn để xác định hai cực của nó.

Tùy theo thành phần của vật liệu gốm mà ứng dụng khác nhau. Tụ gốm được chia thành hai loại ứng dụng:

Các tụ gốm loại 1 mang lại độ ổn định cao và tiêu hao thấp cho các ứng dụng mạch cộng hưởng.

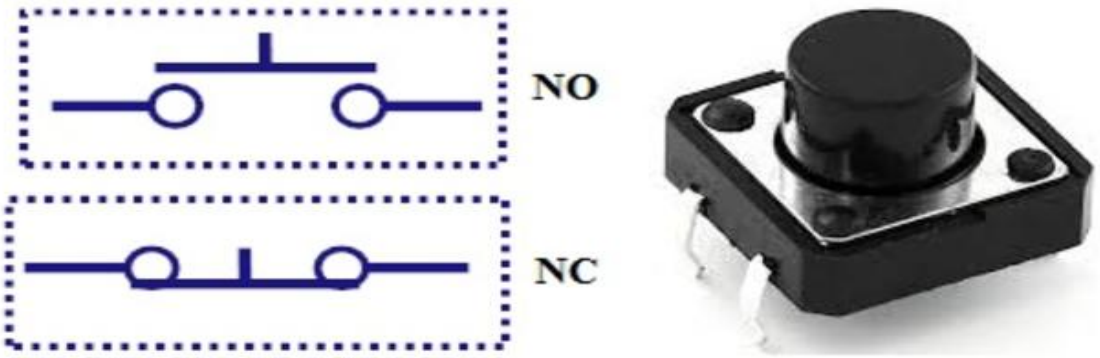
Các tụ gốm loại 2 cung cấp hiệu suất thể tích cao cho các ứng dụng đệm, by-pass và khớp nối.

Tụ gốm, đặc biệt là tụ gốm nhiều lớp (MLCCs), là tụ điện được sản xuất và sử dụng nhiều nhất trong các thiết bị điện tử.

Ký hiệu của tụ gốm như hình bên dưới

3.1.2 Button

3.1.2.1 Giới thiệu nút nhấn



Hình 1.1.3 Hình ảnh nút nhấn

Nút nhấn hay còn gọi là nút ấn dùng để đóng ngắt từ xa các thiết bị điện, các loại máy móc hay một số quá trình trong điều khiển.

Nút nhấn thường được thiết kế trên bảng điều khiển, tủ điện, hay trên hộp nút nhấn… Khi thao tác với nút nhấn phải dứt khoát để mở hoặc đóng mạch điện.

Đa số các nút nhấn được làm bằng nhựa hoặc kim loại. Hình dạng của nút ấn được thiết kế phù hợp với ngón tay hoặc bàn tay để dễ dàng sử dụng và thao tác.

Nút nhấn được thiết kế, sản xuất theo tiêu chuẩn cao, kiểu dáng đẹp, kết cấu chất lượng, dễ lắp đặt và thay thế.

3.1.2.2 Cấu tạo nút nhấn

Hệ thống lò xo,hệ thống các tiếp điểm thường hở (NO) – thường đóng (NC)Vỏ bảo vệ

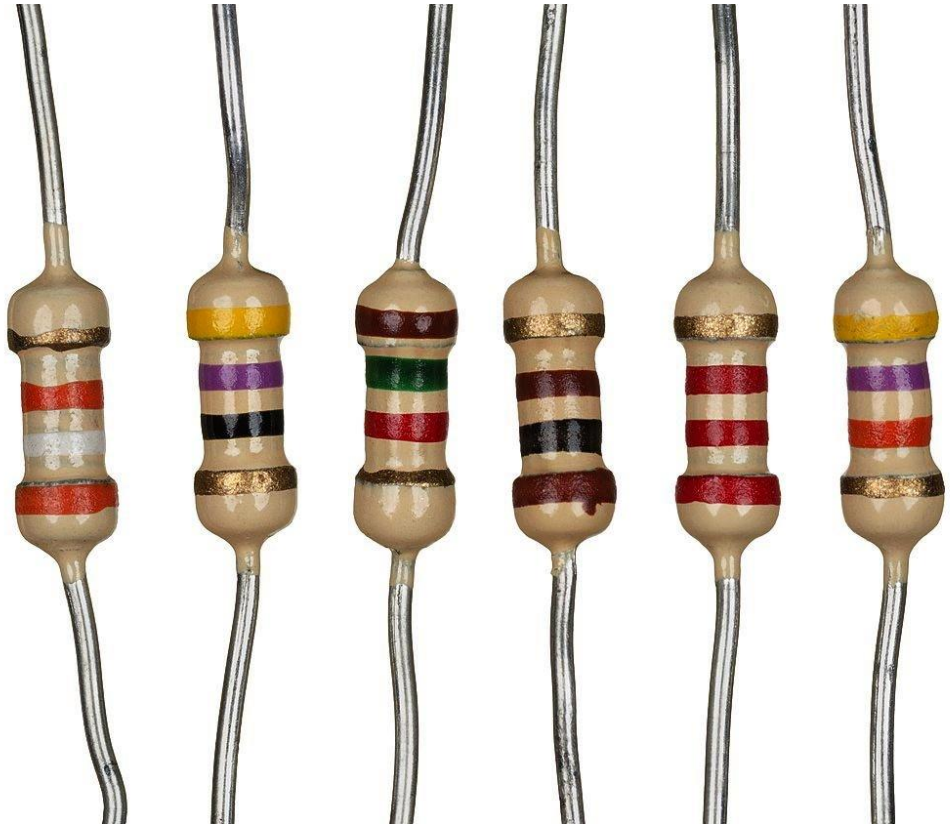
Với nút nhấn nhả: Các tiếp điểm sẽ chuyển trạng thái khi có lực tác động vào nút nhấn. Khi không còn lực tác động tiếp điểm sẽ trở lại trạng thái ban đầu.

Với nút nhấn giữ: Các tiếp điểm sẽ chuyển trạng thái khi có lực tác động vào nút nhấn. Khi không còn lực tác động vào nút ấn, trạng thái tiếp điểm vẫn sẽ duy trì, khi có lực tác động vào nút nhấn lần nữa thì tiếp điểm trở lại trạng thái ban đầu.

3.1.3 Điện trở
3.1.3.1 Giới thiệu điện trở

Khái niệm: Điện trở là một linh kiện điện tử thụ động trong mạch điện có biểu tượng R. Nó là đại lượng vật lý đặc trưng cho tính chất cản trở dòng điện của vật liệu.

– Điện trở kháng được định nghĩa là tỉ số của hiệu điện thế giữa hai đầu vật thể đó với cường độ dòng điện đi qua nó:



Hình 1.1.4 Hình ảnh điện trở

3.1.3.2 Công Thức Tính của Điện Trở Là Gì?

Công thức tính: R=U/I.

Trong đó :

U: là hiệu điện thế giữa hai đầu vật dẫn điện, đo bằng vôn (V).

I: là cường độ dòng điện đi qua vật dẫn điện, đo bằng ampe (A).

R: là điện trở của vật dẫn điện, đo bằng Ohm (Ω).

– Điện trở là gì? Hiểu một cách đơn giản đó là sự cản trở dòng điện của một vật dẫn điện, nếu một vật dẫn điện tốt thì điện trở nhỏ, vật dẫn điện kém thì điện trở lớn, vật cách điện thì điện trở là vô cùng lớn.

– Do đó, bản chất nó là 1 sợi dây dẫn có điện trở rất lớn (thực ra lớn bé còn tùy thuộc vào giá trị của nó), điện trở không phân cực, tức là không phân biệt âm dương.

3.1.4 Led 7segment 2 số Anode chung:

Cấu tạo:

LED 7 đoạn bao gồm 8 LED được kết nối song song để có thể thắp sáng hiển thị số "0, 1, 2, 3, 4, 5, 7, 8, 9, A, b, C, d, E, F, …".

Mỗi đoạn Led được đánh dấu từ A tới G.

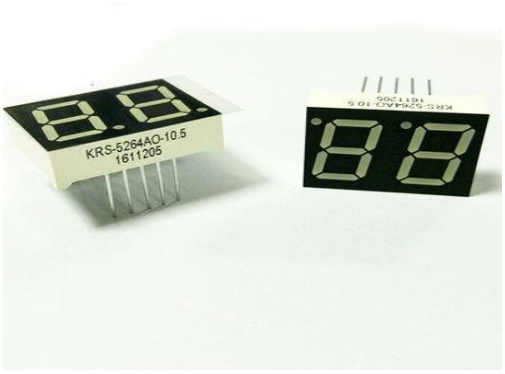
Đoạn thứ tám gọi là "chấm thập phân" (*Decimal Point)* ký hiệu DP được sử dụng khi hiển thị số không phải là số nguyên

Phân loại LED 7 đoạn:

Dựa vào các cực được nối, có thể phân loại LED 7 đoạn như sau:

Loại dương chung (Common Anode): nếu cực dương (anode) của tất cả 8 LED được nối với nhau và các cực âm (cathode) đứng riêng lẻ.

Loại âm chung (Common Cathode): nếu cực âm (cathode) của tất cả 8 LED được nối với nhau và các cực dương (anode) đứng riêng lẻ.



Hình 1.1.5 Hình ảnh led 7 đoạn

❖ **Thông số kỹ thuật**

- Loại: anode chung
- LED: 0.36 inch
- Màu: đỏ
- Điện áp:3.3-5V

3.1.4 Module Led giao thông

3.4.1.1 Giới thiệu module Led

Mạch LED đèn giao thông là một trong những module LED được sử dụng làm các mô hình điện tử, với 3 bóng đèn LED để mô phỏng cột đèn giao thông chính xác.

Hình 1.1.6 Hình ảnh module đèn giao thông

THÔNG SỐ KĨ THUẬT

- Kích thước: 56*21*11mm
- Màu sắc: đỏ, vàng, xanh
- LED: 3 led đục - đường kính bóng led 8mm
- Điện áp: 5V
- Trọng lượng: 25 gram
- 4 Chân nối: GND, red,blue,green

3.1.5 THẠCH ANH 12MHZ; 32.768KHZ

3.1.5.1 Giới thiệu chung

Thạch anh là bộ dao động khá ổn định để tạo ra tần số dao động cho vi điều khiển.

Thông số kĩ thuật:

- Tần số: 12MHz, 32.768KHZ.
- Dung sai tần số: ±20ppm đến ±30ppm.
- Điện dung tải: 10pF - 30pF.
- Dải nhiệt độ hoạt động: -20°C đến +70°C.

Hình 1.1.7 Hình ảnh thạch anh

3.1.5.2 Nguyên lý hoạt động

Thạch anh hoạt động dựa trên hiện tượng dao động cơ học. Khi được kích thích bởi một điện áp xoay chiều, thạch anh sẽ dao động với một tần số xác định. Tần số này phụ thuộc vào kích thước và hình dạng của miếng thạch anh.

- Dao động tạo tần số: Điện áp áp dụng lên thạch anh tạo ra dao động cơ học, và dao động này chuyển đổi lại thành tín hiệu điện với tần số tương ứng.
- Ổn định tần số: Thạch anh có khả năng giữ tần số ổn định cao, ít bị ảnh hưởng bởi nhiệt độ và các yếu tố môi trường.

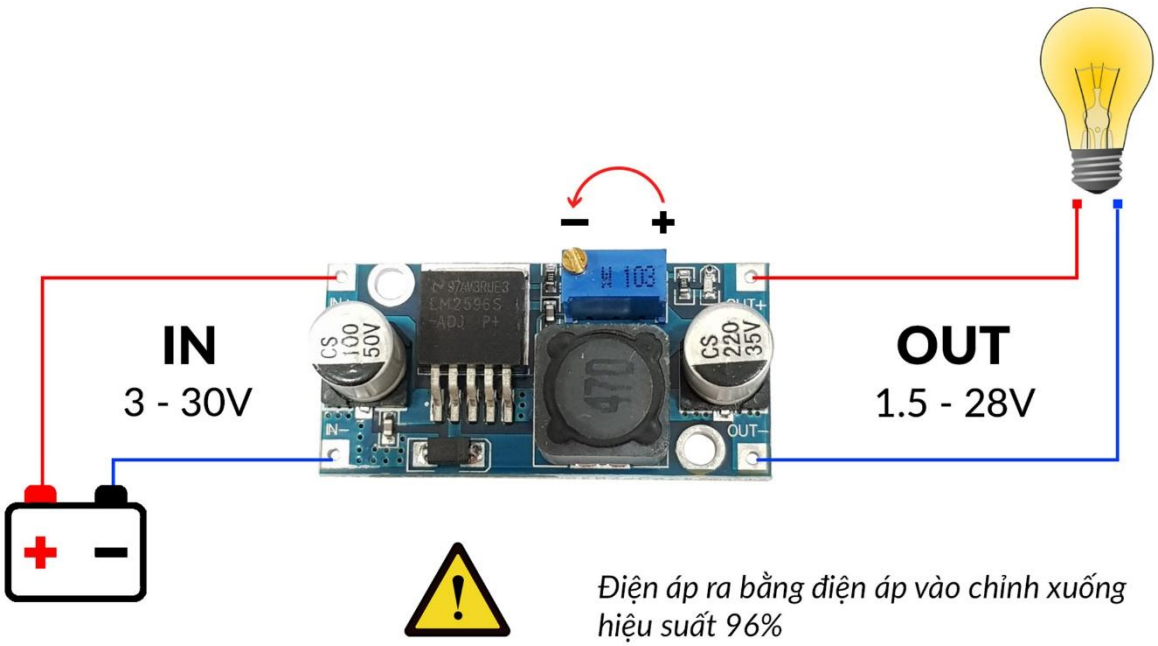
3.1.5.3 Ứng dụng

- Thiết bị truyền thông: Sử dụng trong các bộ phát sóng vô tuyến và các thiết bị truyền thông để tạo ra tần số sóng mang chính xác.
- Thiết bị công nghiệp: Được sử dụng trong các thiết bị đo lường và kiểm tra, hệ thống điều khiển công nghiệp.
- Thiết bị y tế: Sử dụng trong các thiết bị siêu âm và các hệ thống chẩn đoán y tế khác.

3.1.6 Module hạ áp LM2596

Giới thiệu chung

Mạch giảm áp LM2596 nhỏ gọn có khả năng giảm áp từ 30V xuống 1.5V mà vẫn đạt hiệu suất cao (92%) . Thích hợp cho các ứng dụng chia nguồn, hạ áp, cấp cho các thiết bị như camera, motor, robot,…



Hình 1.1.8 Hình ảnh module hạ áp LM2596

Thông số kỹ thuật

- Điện áp đầu vào: Từ 3V đến 30V.
- Điện áp đầu ra: Điều chỉnh được trong khoảng 1.5V đến 30V.
- Dòng đáp ứng tối đa là 3A.
- Hiệu suất: 92%
- Công suất: 15W
- Kích thước: 45 (dài) * 20 (rộng) * 14 (cao) mm

3.1.7 NGUỒN PIN 9V

Giới thiệu chung

Pin 9v Panasonic là loại pin chất lượng cao được sử dụng trong các thiết bị đo, mix karaoke… Pin sử dụng chất liệu kiềm cho thời gian lưu trữ điện vượt trội gấp nhiều lần so với pin than. Pin 9v Panasonic có tuổi thọ của pin cao đến 5 năm.

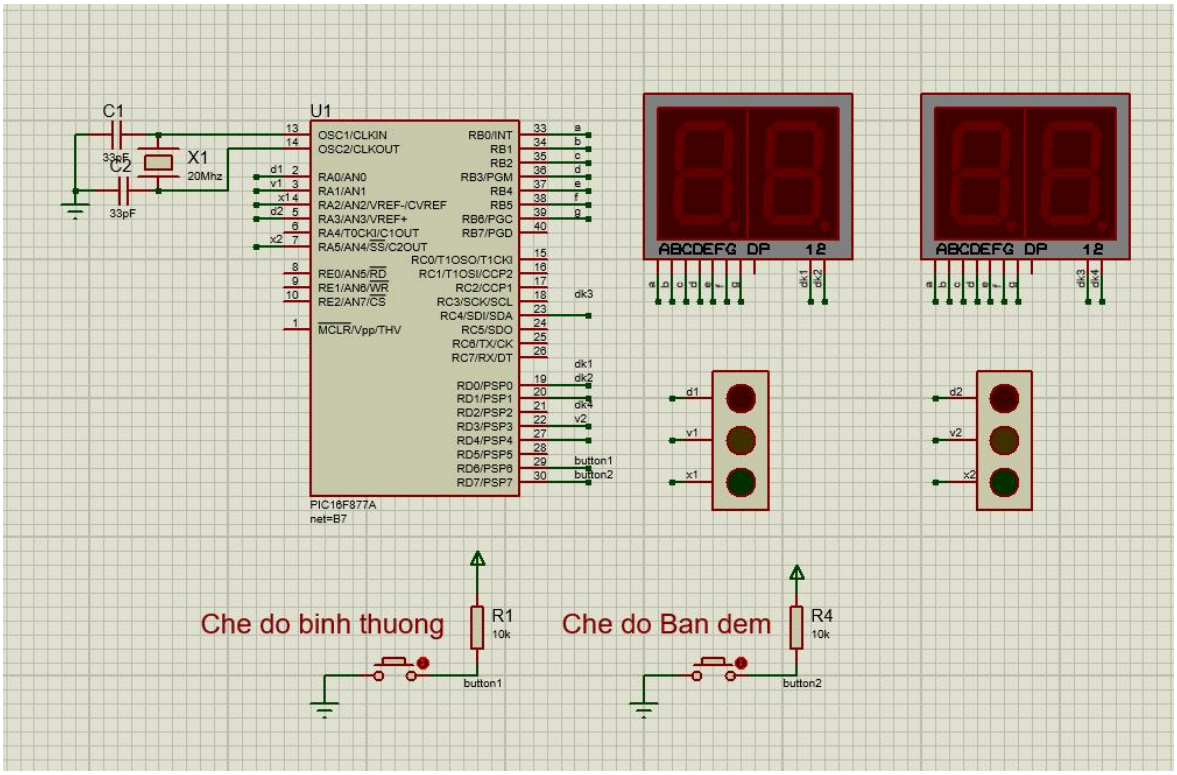
Hình 1.1.9 Hình ảnh Pin 9V

THÔNG SỐ KỸ THUẬT

- TT sản phẩm: Pin 9V Panasonic
- Chất liệu: Alkaline
- Điện áp: 9V
- Hãng sản xuất : Pin Sac Panasonic
- Loại pin: Pin vuông
- Hạn sử dụng được in trên thân pin

- Hàng chính hãng bảo hành lưu trữ theo quy định hãng sản xuất

3.2 SƠ ĐỒ ĐẤU NỐI PHẦN CỨNG

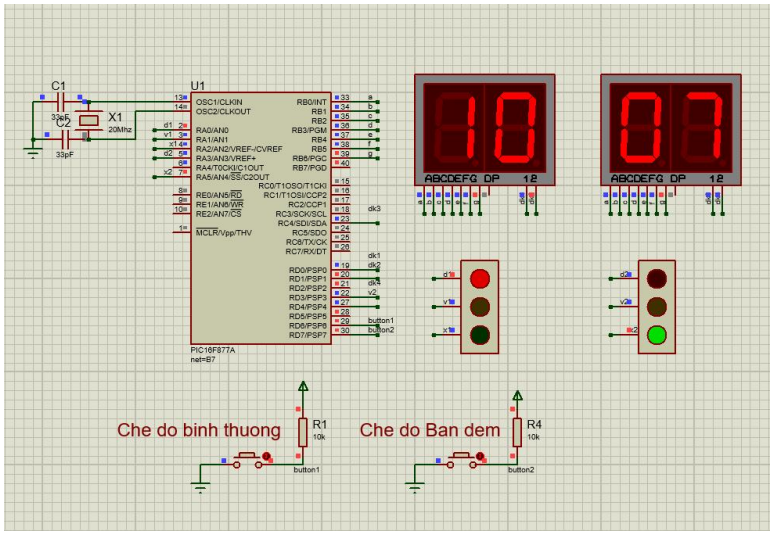


1.2.1 Hình ảnh thiết kết mạch bằng phần mềm Protues8

3.3 NGUYÊN LÝ HOẠT ĐỘNG

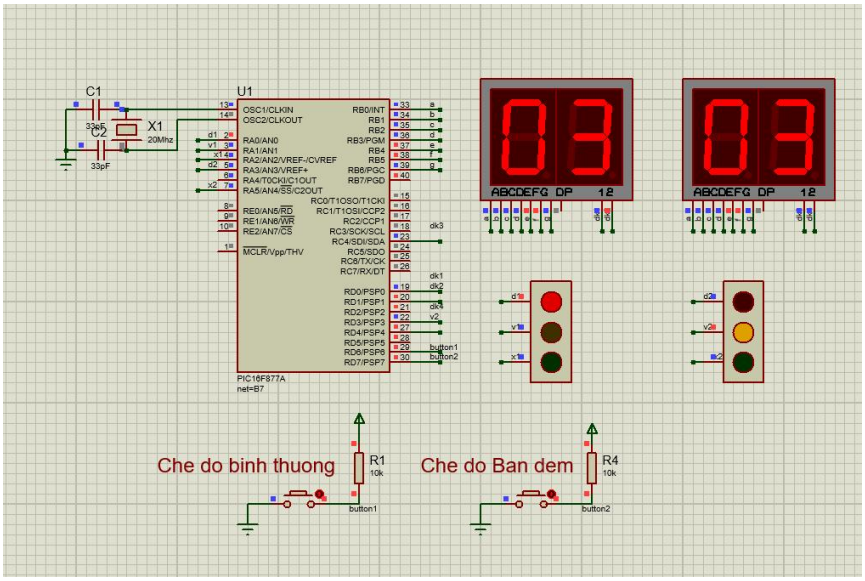
3.3.1 LED giao thông

- **Với chế độ 1**
  - Ban đầu mạch sẽ hiển thị:
    - LED7_1 là số 10.
    - LED7_2 là số 7.
    - LED giao thông _1: đỏ (sáng), vàng (tắt), xanh (tắt).
    - LED giao thông _1: đỏ (tắt), vàng (tắt), xanh (sáng).



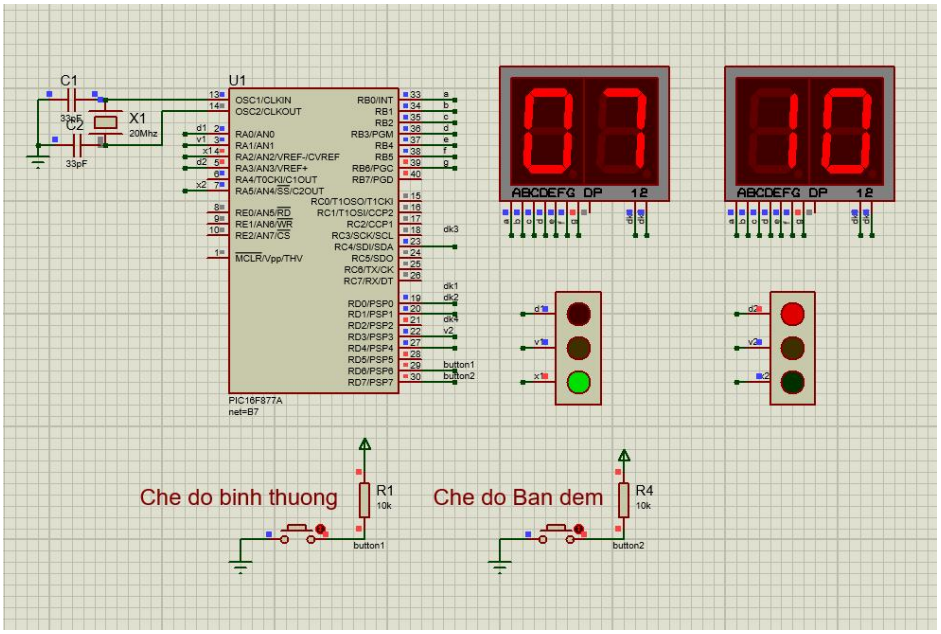
1.2.2 Hình ảnh chế độ 1

- Sau 7 giây :
  - LED7_1 là số 3.
  - LED7_2 là số 3.
  - LED giao thông _1: đỏ (sáng), vàng (tắt), xanh (tắt).
  - LED giao thông _1: đỏ (tắt), vàng (sáng), xanh (tắt).



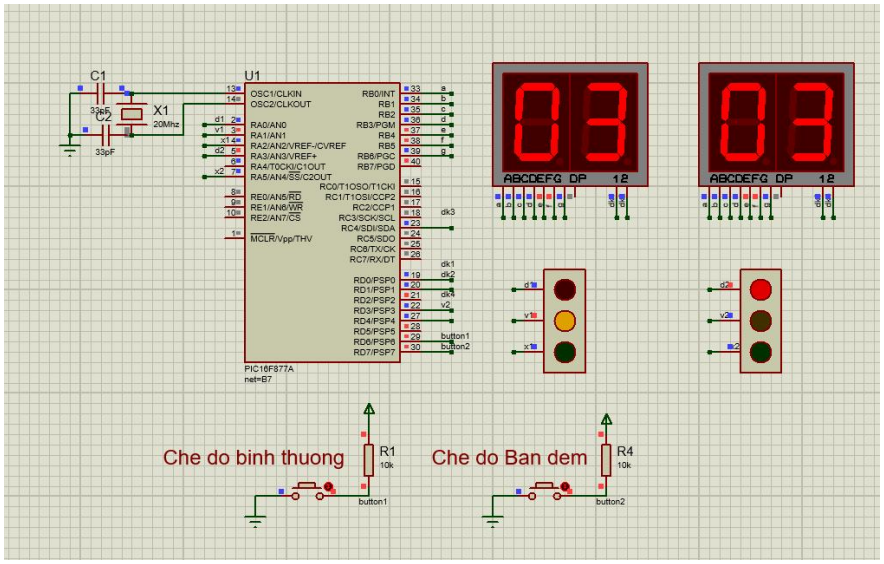
1.2.3 Hình ảnh sau 7 giây

- Sau 3 giây :
  - LED7_1 là số 7.
  - LED7_2 là số 10.
  - LED giao thông _1: đỏ (tắt), vàng (tắt), xanh (sáng).
  - LED giao thông _1: đỏ (sáng), vàng (tắt), xanh (tắt).



1.2.4 Hình ảnh sau 3 giây

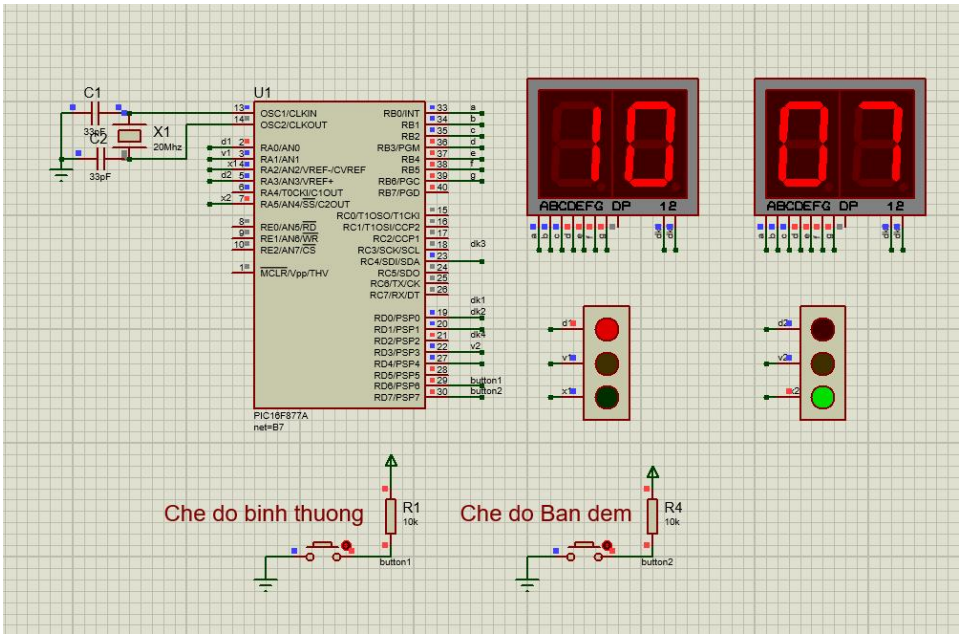
- Sau 7 giây :
  - LED7_1 là số 3.
  - LED7_2 là số 3.
  - LED giao thông _1: đỏ (tắt), vàng (sáng), xanh (tắt).
  - LED giao thông _1: đỏ (sáng), vàng (tắt), xanh (tắt).



1.2.5 Hình ảnh sau 7 giây

- Sau 3 giây :
  - LED7_1 là số 10.

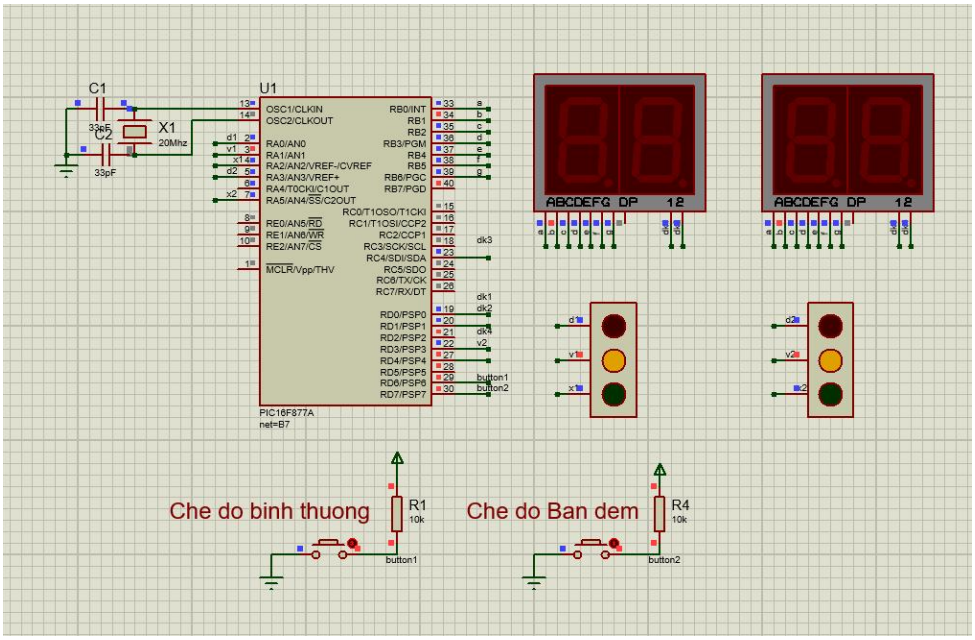
- LED7_2 là số 7.
- LED giao thông _1: đỏ (sáng), vàng (tắt), xanh (tắt).
- LED giao thông _1: đỏ (tắt), vàng (tắt), xanh (sáng).



1.2.6 Hình ảnh sau 3 giây

Sau đó lặp lại chu kỳ

- **Với chế độ 2:**
  - Led vàng ở 2 đèn cùng sáng nahys với chu kỳ 500ms.
  - Các Led còn lại tắt.

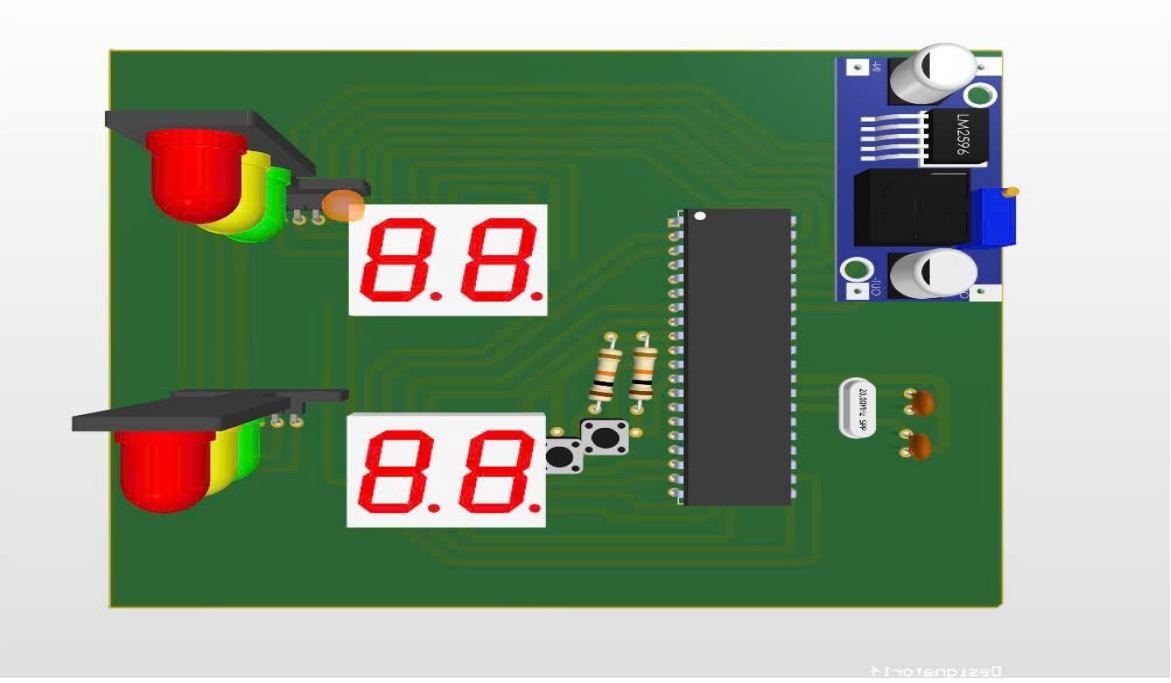


1.2.7 Hình ảnh chế độ 2

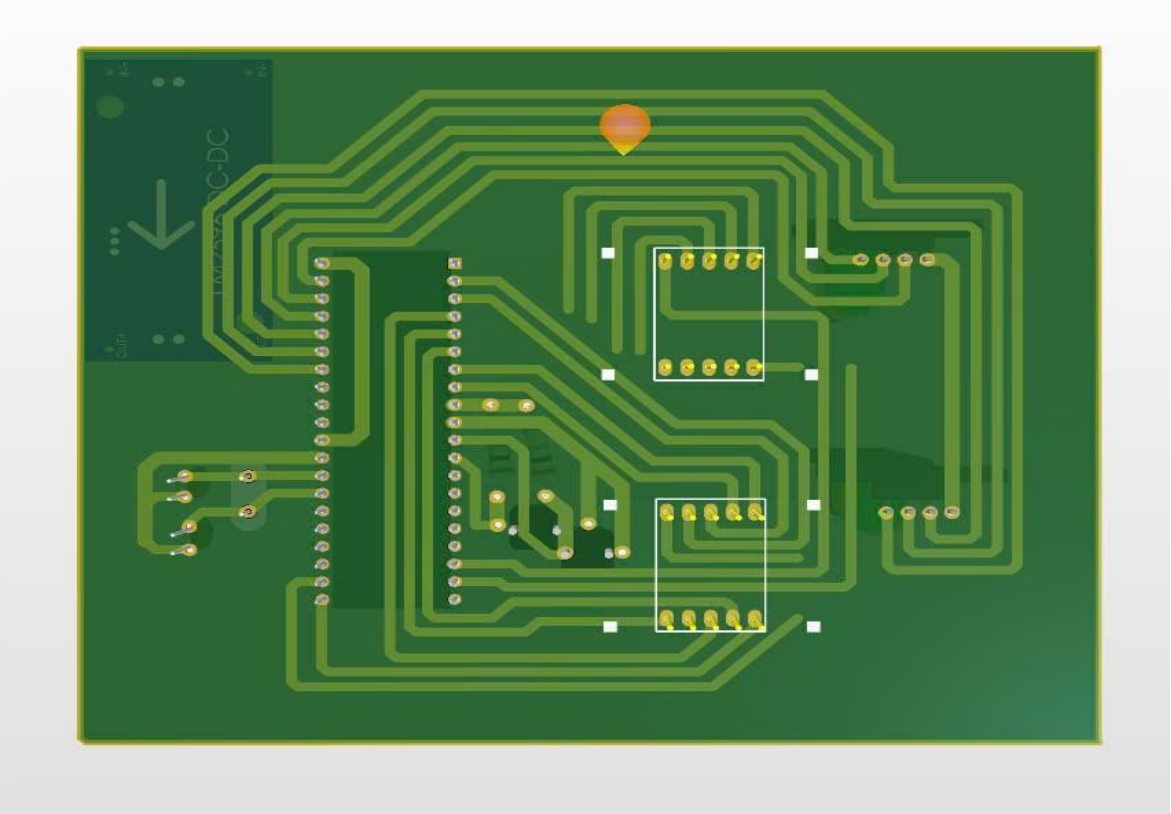
3.3.2 LED 7 đoạn

- **Chế độ mặc định và biến trạng thái:**
  - mode: Xác định chế độ hoạt động (1 hoặc 0).
  - i1 và i2: Lưu trữ các giá trị hiển thị trên hai LED 7-segment.
- **Hàm hiển thị chữ số trên LED 7-segment:**
- **Hàm display_digit hiển thị một chữ số (value) lên một trong các digit (digit) của LED 7-segment (led).**
- **Nhiệm vụ taskLED7:**
  - Chế độ 1 (mode = 1):
    - Bộ đếm thời gian: Tăng giá trị counter mỗi 10ms.
    - Giảm giá trị: Khi counter đạt 1 giây (100 lần), giảm các giá trị i1 và i2.
    - Đổi giá trị: Nếu i1 và i2 đều là 0, chuyển đổi giá trị giữa 10-7 và 7-10.
    - Hiển thị số: Gọi hàm display_digit để hiển thị các số lên LED 7-segment.
- **Chế độ 2 (mode = 0):**
  - Tắt LED: Tắt tất cả các digit của LED 7-segment.
  - Hiển thị giá trị cố định: Hiển thị giá trị cố định 10 và 7 lên LED 7-segment

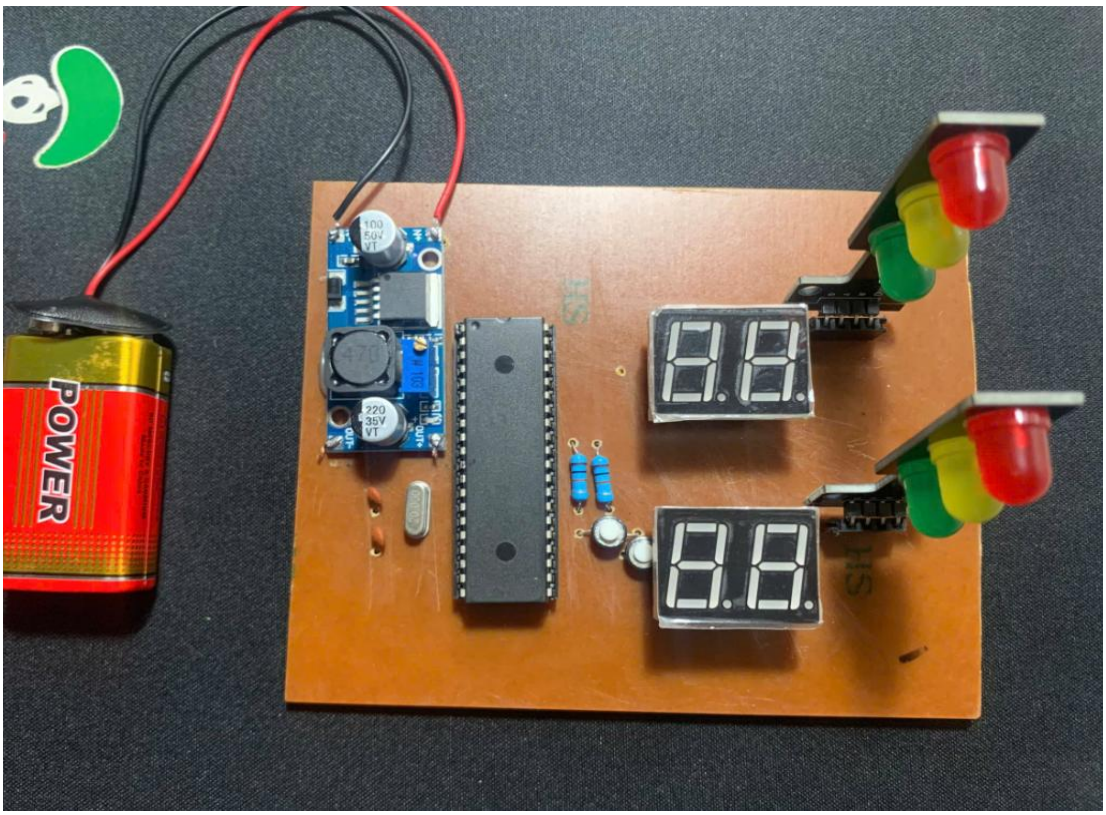
3.3.3: Mô phỏng trên Altium



Hình 1.2.8 Mô phỏng 3D trên Altium mặt trước



Hình 1.2.9 Mô phỏng 3D trên Altium mặt sau



1.2.10 Hình ảnh mô hình thực tế

TÀI LIỆU THAM KHẢO

- https://www.youtube.com/results?search_query=PIC16F877A+traffic+light
- https://www.alldatasheet.com/view.jsp?Searchword=Pic16f877a%20datasheet&gad_source=1&gclid=CjwKCAiA-Oi7BhA1EiwA2rIu2-N_5P0520OvMiREfZ0owCgAu9t1gcIhKCfbQP98PH5PWICNWVIoORoCf1kQAvD_BwE

- https://www.sciencedirect.com/book/9781856177504/designing-embedded-systems-with-pic-microcontrollers