ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO THÍ NGHIỆM

THIẾT KẾ LUẬN LÝ

BÀI TẬP LỚN

LỚP L06, NHÓM 6

GVHD: Huỳnh Hoàng Kha

Thành phố Hồ Chí Minh, tháng 4 năm 2023

ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

BÁO CÁO BÀI TẬP LỚN

THIẾT KẾ LUẬN LÝ

CHỦ ĐỀ:

NGHIÊN CỨU VÀ MÔ PHỎNG ĐÈN TÍN HIỆU GIAO THÔNG Ở ĐƯỜNG LỚN VÀ NHỮNG NƠI GIAO NHAU VỚI ĐƯỜNG SẮT

Lớp: L06 **Nhóm:** 6

Danh sách thành viên:

Họ và tên	**MSSV**	**Mail**
Dương Minh Hiếu	2210978	hieu.duongk22bk@hcmut.edu.vn
Trịnh Thị Mỹ Lệ	2211832	le.trinhmei@hcmut.edu.vn
Nguyễn Ngọc Hà My	2212104	my.nguyen2004@hcmut.edu.vn

MỤC LỤC

LỜI CẢM ƠN

Trong suốt quá trình thực hiện tiểu luận nói trên, nhóm chúng tôi đã nhận được rất nhiều sự quan tâm và ủng hộ, giúp đỡ tận tình của thầy cô, anh chị em và bè bạn.

Ngoài ra, nhóm cũng xin gửi lời cảm ơn chân thành nhất đến thầy Huỳnh Hoàng Kha, là giảng viên trực tiếp hướng dẫn cho đề tài này. Nhờ có các giảng viên hết lòng chỉ bảo mà nhóm đã hoàn thành tiểu luận đúng tiến độ và giải quyết tốt những vấn đề gặp phải. Sự hướng dẫn của các giảng viên đã phát huy tối đa được khả năng của bản thân cũng như mối quan hệ hỗ trợ giữa thầy và trò trong môi trường giáo dục.

Lời cuối, xin một lần nữa gửi lời biết ơn sâu sắc đến các cá nhân, giảng viên đã dành thời gian chỉ dẫn cho nhóm. Đây chính là niềm tin, nguồn động lực to lớn để nhóm có thể đạt được kết quả này.

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

CHƯƠNG I: MỞ ĐẦU

1. Lý do chọn đề tài

Việc chọn đề tài "Nghiên cứu và mô phỏng đèn tín hiệu giao thông ở những nơi giao nhau với đường sắt " là rất cần thiết vì đây là một chủ đề quan trọng trong lĩnh vực an toàn giao thông. Tại các nơi giao nhau với đường sắt đều là những điểm nóng về giao thông, vì vậy việc đảm bảo an toàn giao thông tại nơi này là rất cần thiết. Trong đó, đèn tín hiệu giao thông là một trong những công cụ quan trọng nhất để điều tiết lưu thông và giảm thiểu các tai nạn đáng tiếc xảy ra.

Với sự phát triển của công nghệ, các loại đèn tín hiệu giao thông cũng ngày càng được cải tiến để đáp ứng nhu cầu của người dân. Tuy nhiên, vẫn còn nhiều vấn đề cần được giải quyết, đặc biệt là ở những nơi giao nhau với đường sắt. Vì vậy, việc nghiên cứu và mô phỏng đèn tín hiệu giao thông tại những nơi này sẽ giúp cho các chuyên gia, nhà quản lý giao thông và các cơ quan chức năng có thể đưa ra các giải pháp tối ưu nhằm cải thiện tình hình an toàn giao thông tại điểm nóng này. Đồng thời, nó cũng có thể giúp cho những người tham gia giao thông có thể hiểu rõ hơn về cách thức hoạt động của đèn tín hiệu giao thông và cách sử dụng chúng một cách hiệu quả nhất. Đó là lý do hình thành đề tài của nhóm chúng em.

2. Giới thiệu sơ bộ đề tài

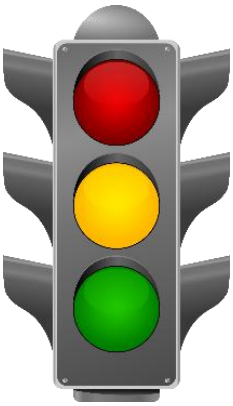
Đề tài " Nghiên cứu và mô phỏng đèn tín hiệu giao thông ở những nơi giao nhau với đường sắt " là một đề tài liên quan đến lĩnh vực giao thông đô thị và công nghệ thông tin. Điểm nhấn của đề tài là việc tìm hiểu về đèn tín hiệu giao thông và áp dụng công nghệ thông tin để phát triển các giải pháp điều tiết giao thông qua việc nghiên cứu và code trên phần mềm rồi từ đó so sánh và đưa ra các giải pháp cụ thể phù hợp với tình hình giao thông hiện tại.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

1. Định nghĩa

1.1 Đèn tín hiệu giao thông

1.1.1 Định nghĩa



Đèn giao thông (còn được gọi tên khác là hệ thống đèn tín hiệu giao thông, đèn điều khiển giao thông, hay đèn xanh đèn đỏ) là một thiết bị được dùng để điều khiển giao thông ở những giao lộ có lượng phương tiện lưu thông lớn (thường là ngã ba, ngã tư đông xe qua lại). Đây là một thiết bị quan t rọng không những an toàn cho các phương tiện mà còn giúp giảm ùn tắc giao thông vào giờ cao điểm. Nó được lắp ở tâm giao lộ hoặc trên vỉa hè. Đèn tín hiệu giao thông có thể hoạt động tự động hay cảnh sát giao thông điều khiển.

1.1.2 Màu tín hiệu

Đèn tín hiệu chính điều khiển giao thông được áp dụng ba loại màu tín hiệu: Xanh, vàng và đỏ.

- Tín hiệu xanh: Cho phép đi.
- Tín hiệu vàng báo hiệu thay đổi tín hiệu của đèn xanh sang đỏ. Tín hiệu vàng bật sáng, phải dừng lại trước vạch dừng, trường hợp đã đi quá vạch dừng hoặc đã quá gần vạch dừng nếu dừng lại thấy nguy hiểm thì được đi tiếp.

Trong trường hợp tín hiệu vàng nhấp nháy là được đi nhưng phải giảm tốc độ, chú ý quan sát, nhường đường cho người đi bộ qua đường hoặc các phương tiện khác theo quy định của Luật Giao thông đường bộ.

- Tín hiệu đỏ: Báo hiệu phải dừng lại trước vạch dừng. Nếu không có vạch dừng thì phải dừng trước đèn tín hiệu theo chiều đi.

1.1.3 Quy định điều khiển đèn tín hiệu

Đèn tín hiệu phải bật từng màu riêng biệt, đèn này tắt mới được bật đèn kia lên, không được bật nhiều màu cùng một lúc. Giữa 2 chiều đường, khi chiều A bật đèn đỏ thì lập tức chiều B phải bật ngay đèn xanh và ngược lại. Khi chuyển từ xanh-đỏ và đỏ-xanh bắt buộc phải bật qua màu vàng, vì màu vàng đệm giữa 2 màu xanh đỏ. Khi bật đèn vàng thì phải bật sáng ở chiều đường sắp dừng.

1.2 Đèn tín hiệu giao thông ở những nơi giao nhau với đường sắt

1.2.1 Định nghĩa

Đèn tín hiệu giao thông ở những nơi giao nhau với đường sắt thường được đặt tại các điểm giao cắt giữa đường bộ và đường sắt để hướng dẫn và điều khiển phương tiện tham gia giao thông qua khu vực này một cách an toàn.Hệ thống đèn tín hiệu giao thông bao gồm một bộ đèn bao gồm các đèn màu đỏ, vàng và xanh lam được sắp xếp theo một thứ tự cụ thể để chỉ dẫn phương tiện giao thông. Khi đèn đỏ sáng, các phương tiện đường bộ phải dừng lại hoàn toàn để đảm bảo an toàn khi tàu đi qua. Khi đèn vàng sáng, các phương tiện đường bộ phải chuyển động chậm và

chuẩn bị dừng lại. Khi đèn xanh sáng, các phương tiện đường bộ được phép tiếp tục di chuyển qua khu vực giao nhau đường sắt một cách an toàn.

Hệ thống đèn tín hiệu giao thông ở những nơi giao nhau với đường sắt rất quan trọng để giảm thiểu nguy cơ tai nạn giao thông và đảm bảo an toàn cho tất cả người tham gia giao thông, bao gồm cả người đi bộ và người lái xe.

1.2.2 Đèn tín hiệu thực tế

Đèn tín hiệu giao thông ở những nơi giao nhau với đường sắt hiện nay được thiết kế và sản xuất theo nhiều mẫu mã và kỹ thuật khác nhau, tùy thuộc vào đặc điểm của khu vực và các yêu cầu kỹ thuật.

Các đèn tín hiệu giao thông hiện đại thường sử dụng công nghệ đèn LED để tiết kiệm năng lượng và tăng độ sáng. Hệ thống cũng có thể được kết hợp với các công nghệ khác như camera giám sát và cảm biến để cải thiện hiệu quả và an toàn khi điều khiển giao thông.

Ngoài ra, có nhiều hệ thống đèn tín hiệu giao thông khác nhau được sử dụng tại các nước khác nhau, ví dụ như hệ thống đèn tín hiệu vòng xuyến ở Châu Âu, hệ thống đèn tín hiệu ngược chiều giữa phương tiện và người đi bộ ở Nhật Bản, và hệ thống đèn tín hiệu xoay chuyển ở một số nước khác.

Tuy nhiên, mục đích chung của các hệ thống đèn tín hiệu giao thông ở những nơi giao nhau với đường sắt vẫn là giảm thiểu nguy cơ tai nạn và đảm bảo an toàn cho tất cả người tham gia giao thông.

2. Miêu tả chi tiết đề tài

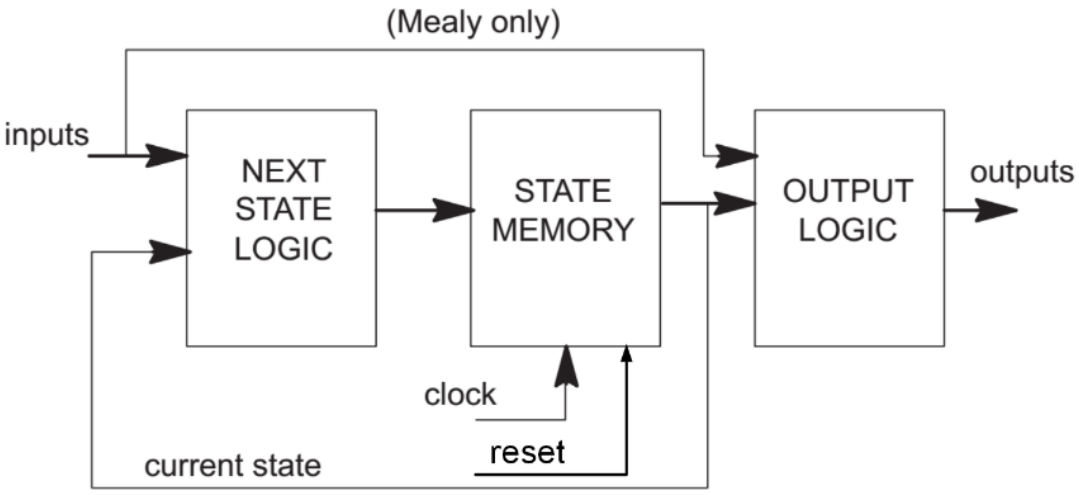
- Tín hiệu đèn giao thông luôn được bật với 3 tín hiệu ĐỎ-VÀNG-XANH(RED-YELLOW-GREEN)
- Khi không có phương tiện giao thông ở các khu vực đường sắt,.. tín hiệu đèn giao thông ở đường sắt sẽ tắt và được xem là chế độ mặc định (default). Thỉnh thoảng sẽ có xe lửa xuất hiện từ đường sắt, mô phỏng bằng sw1, đèn đường sắt sẽ vào chế độ cảnh báo nguy hiểm
- Khi vào đêm khuya, đèn giao thông sẽ chuyển sang màu vàng cho tới khi trời sáng.
- Có 2 swich để mô phỏng 2 chế độ: chế độ ban đêm(sw0) và chế độ khi có tàu chạy ngang(sw1).
    - Sw0 để mô phỏng khi vào ban đêm.
    - Sw1 để mô phỏng xe lửa khi xuất hiện trên đường sắt.
        - Sw1 =0 nếu không có tàu chạy ngang, khi đó tín hiệu đèn đường sắt sẽ tắt như mặc định.
        - Sw1=1 khi có tàu chạy ngang, khi đó đèn đường sắt sẽ bật kết hợp chớp tắt đèn với xung clock 0,25Hz.

3. FSM (Finite State Machine)

3.1 Định nghĩa

- FSM (Finite State Machine) là một mô hình toán học được sử dụng để mô tả hành vi của các hệ thống tự động có hữu hạn trạng thái. Một FSM được biểu diễn bởi một tập hữu hạn các trạng thái, các sự kiện đầu vào và các hành động đầu ra được thực hiện trong các trạng thái khác nhau.
- Một số chức năng chính của FSM bao gồm:
  - Mô tả hành vi của hệ thống.
  - Kiểm tra đúng / sai
  - Tự động hoá các quy trình
  - Phân tích hệ thống
- Trong mỗi trạng thái, FSM có thể chuyển đổi sang một trạng thái khác hoặc giữ nguyên trạng thái hiện tại, tùy thuộc vào sự kiện đầu vào
- FSM được biểu diễn như 1 đồ thị có hướng.
- Ứng dụng của FSM trong lập trình:
  - Mô tả các trạng thái, sự kiện và quá trình chuyển đổi giữa các trạng thái.
  - Sử dụng để quản lý trạng thái của object, hoặc workflow.

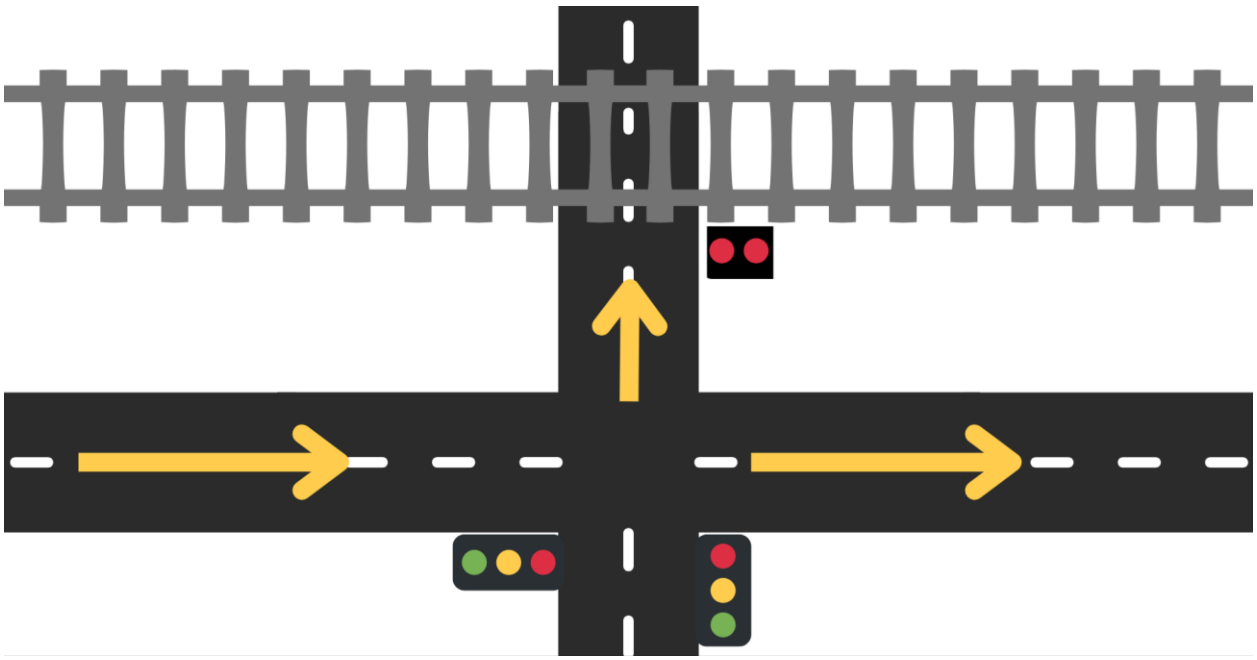
- FSM được sử dụng rộng rãi trong các ứng dụng như thiết kế logic kỹ thuật số, điều khiển máy tính, định tuyến mạng và các hệ thống điều khiển tự động khác.



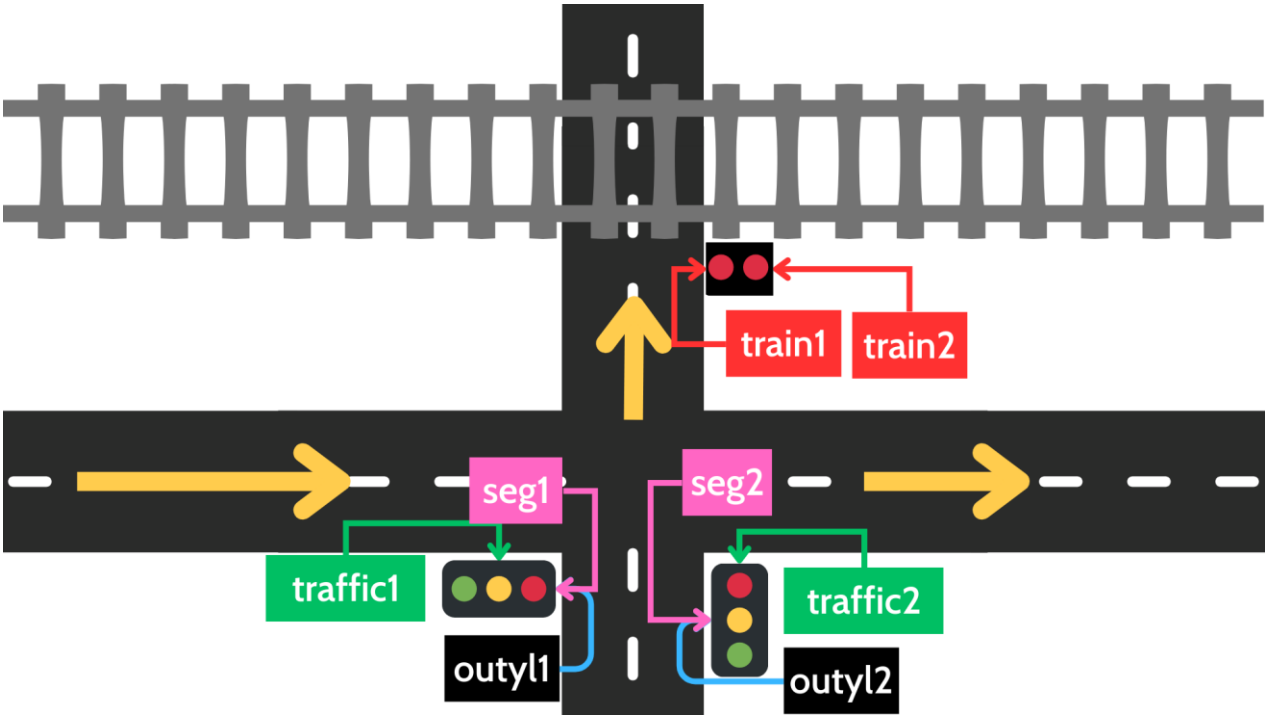
Hình 1: ảnh minh họa mô hình cơ bản của FSM

3.2 Thiết kế

i. Sơ lược



Hình 2: sơ đồ đèn giao thông tại nơi giao nhau với đường sắt



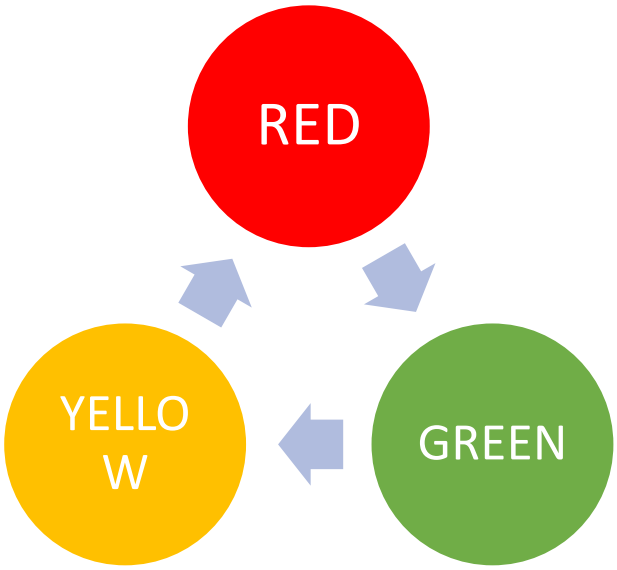
Hình 3: chú thích đèn giao thông tại nơi giao nhau với đường sắt

1.1 Khi switch 0=0 thì tín hiệu đèn giao thông sẽ hoạt động bình thường. Khi switch 1 = 0 thì và tín hiệu đèn ở đường sắt sẽ tắt như mặc định.

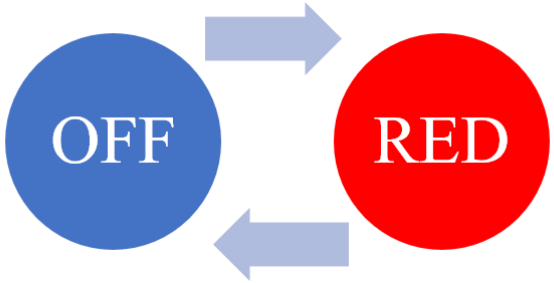
1.2 Mỗi đường sẽ có 3 tín hiệu đèn tương ứng với 3 màu: đỏ/xanh/vàng được ký hiệu là RED/YELLOW/GREEN

1.3 Đèn giao thông sẽ hoạt động bình thường chuyển trạng thái đỏ>> xanh>>vàng.

1.4 Đèn đường sắt sẽ có 2 trạng thái: bật đèn đỏ>> tắt



Hình 4: thứ tự chuyển tín hiệu đèn giao thông traffic1, traffic2 tại nơi giao nhau với đường lớn



Hình 5: thứ tự chuyển tín hiệu đèn giao thông train1, train2 tại nơi giao nhau với đường sắt

State	Signals	
	Traffic 1	**Traffic 2**
S0	GREEN	RED
S1	YELLOW	RED
S2	RED	GREEN
S3	RED	YELLOW

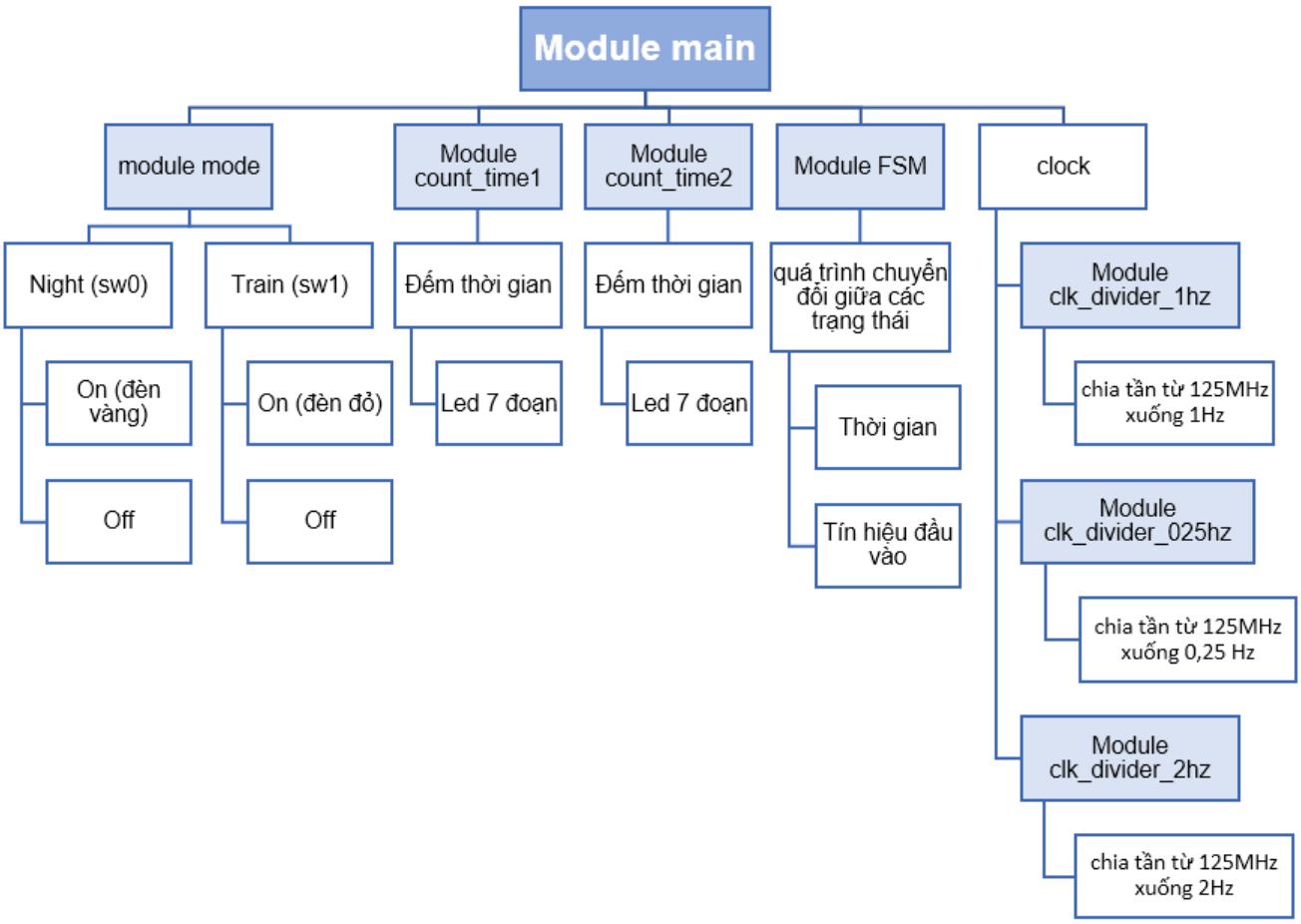
Bảng 1: bảng chuyển trạng thái của traffic 1 và traffic 2

	Switch 0 (sw0)		Switch 1(sw1)	
	Sw0=0	Sw0=1	Sw1=0	Sw1=1
Trạng thái	Ban ngày	Ban đêm	Không có tàu	Có tàu
Tín hiệu	Đỏ>>xanh>>vàng	vàng	off	Đỏ

Bảng 2: bảng mô tả chi tiết của nút bấm switch 0 và switch 1

STT	Led 7 đoạn	STT	Led 7 đoạn	STT	Led 7 đoạn
1	0000_0001	11	0001_0001	21	0010_0001
2	0000_0010	12	0001_0010	22	0010_0010
3	0000_0011	13	0001_0011	23	0010_0011
4	0000_0100	14	0001_0100	24	0010_0100
5	0000_0101	15	0001_0101	25	0010_0101
6	0000_0110	16	0001_0110	26	0010_0110
7	0000_0111	17	0001_0111	27	0010_0111
8	0000_1000	18	0001_1000	28	0010_1000
9	0000_1001	19	0001_1001	29	0010_1001
10	0001_0000	20	0010_0000	30	0011_0000
		88	1000_1000	100	1010_1010

Bảng 3: bảng hiện số lên led 7 đoạn mô phỏng



Hình 6: hình sơ đồ khối

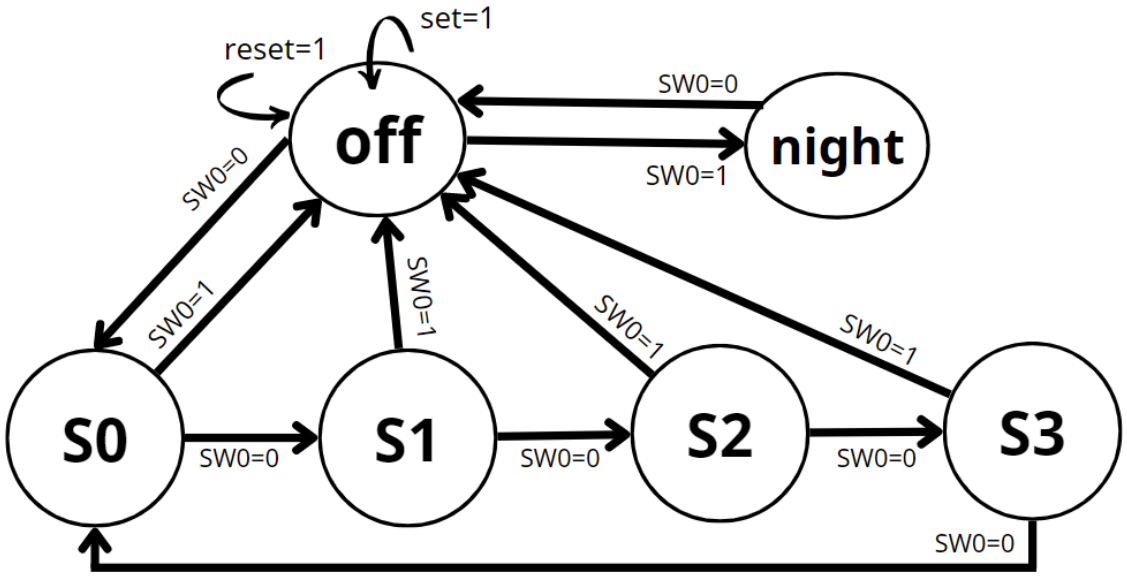
Mã hóa trạng thái:

state	Mã hóa 3 bit nhị phân
S0	3'd0
S1	3'd1
S2	3'd2
S3	3'd3
off	3'd4

Bảng 4: bảng mã hóa trạng thái

ii. phân tích quy trình

1. đèn giao thông

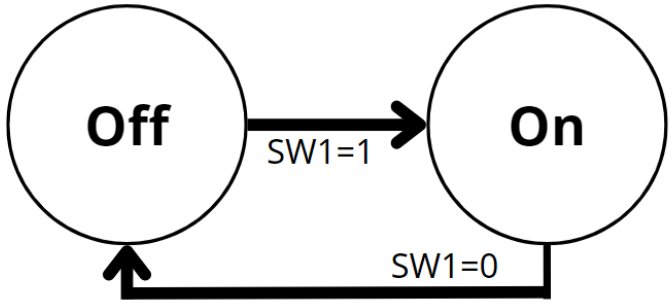


Hình 7: FSM điều khiển đèn giao thông

- Gồm 6 trạng thái lớn: ban đêm (night), off, S0, S1, S2 và S3.
- Theo bảng 1 phân tích, khi vào chế độ night, đèn tín hiệu sẽ chuyển sang tín hiệu vàng, còn tín hiệu đỏ và xanh sẽ chuyển sang trạng thái OFF. Trạng thái được duy trì cho tới khi trời sáng.
- Ở trạng thái bình thường, các chế độ được duy trì theo thứ tự:
  - Khi đèn tín hiệu ở đường 1 (đèn 1) ở trạng thái GREEN và khi đèn tín hiệu ở đường 2 (đèn 2) ở trạng thái RED (S0), thì xe ở đường 1 có thể di chuyển qua đường 2 mà không gặp nguy hiểm.
  - Khi đèn tín hiệu ở đường 1 chuyển sang trạng thái YELLOW (S1), nghĩa là sắp chuyển sang trạng thái RED, xe ở đường 1 nên chuẩn bị dừng lại và không nên vượt qua đường 2.

- Khi đèn tín hiệu ở đường 2 chuyển sang trạng thái GREEN (S2), xe trên đường 2 được phép đi qua mà không gặp nguy hiểm, trong khi đèn tín hiệu ở đường 1 sẽ chuyển sang trạng thái RED.
- Cuối cùng, khi đèn tín hiệu ở đường 2 chuyển sang trạng thái YELLOW (S3), nghĩa là sắp chuyển sang trạng thái RED, các xe trên đường 2 nên chuẩn bị dừng lại và không nên vượt qua đường 1.

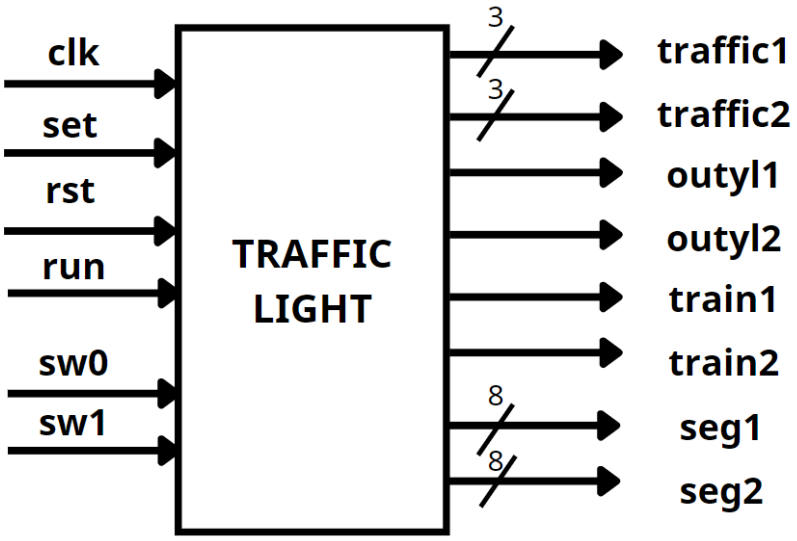
2. Đèn sắt



Hình 8: FSM điều khiển đèn sắt

- khi có tàu chạy qua, đèn sắt sẽ on, tức là sẽ mở tín hiệu đèn đỏ nhấp nháy cảnh báo các phương tiện giao thông.
- Ngược lại, khi không có tàu chạy qua, đèn sẽ tắt, các phương tiện di chuyển bình thường.

iii. Phân tích tổng quan



Hình 9: Sơ đồ tín hiệu giao tiếp của bộ điều khiển đèn giao thông

Trong đó:

- Clock **(clk)** là xung clock giúp các thành phần trong mạch hoạt động đúng cách và đồng bộ với nhau.
- Reset (**rst**) là chân để thiết lập lại giá trị ban đầu của bộ đếm.
- **Set** là chân để vào chế độ chỉnh thời gian.
- **Run** là chân để cập nhập lại giá trị sau khi chỉnh thời gian.
- **Sw0** và **sw1** lần lượt là chế độ ban đêm (night) và có tàu đi qua (train).
- **Traffic1** [2:0] và **traffic2** [2:0] lần lượt là tín hiệu điều khiển đèn trên đường 1 và 2.
- **Outyl1** và **outyl2** lần lượt là là tín hiệu chuyển đèn vàng nhấp nháy ở chế độ ban đêm ở đường 1 và đường 2.
- **Train1** và **train2** lần lượt là tín hiệu điều khiển đèn đường sắt khi chuyển sang chế độ có tàu đi ngang qua.
- **Seg1** [7:0] và **seg2** [7:0] lần lượt là tín hiệu đèn ở led 7 đoạn.

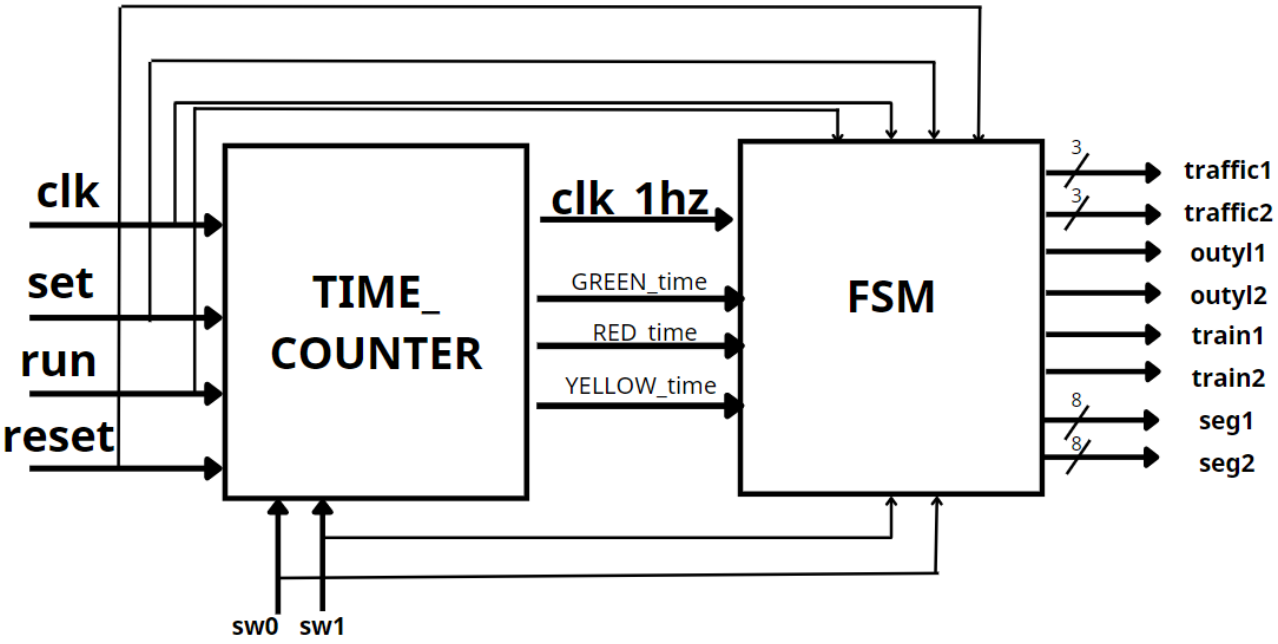
Tín hiệu	
RED	001
YELLOW	010
GREEN	100

Bảng 5: Bảng chuyển trạng thái đèn giao thông

Bảng mã hóa giá trị ngõ ra tương ứng với các trạng thái đèn như sau:

Traffic 1	Traffic 2	Traffic1 [2:0]	Traffic2 [2:0]
GREEN	RED	100	001
YELLOW	RED	010	001
RED	GREEN	001	100
RED	YELLOW	001	010

Bảng 6: Bảng giá trị ngõ ra bộ điều khiển tương ứng với trạng thái đèn



Hình 10: Sơ đồ khối của bộ điều khiển đèn giao thông

Căn cứ trên hoạt động mà yêu cầu đề ra, sơ đồ khối của bộ điều khiển đèn giao thông như sau:

- TIME_COUNTER: dùng để đếm xung clock xác định thời gian duy trì trạng thái của đèn. Đây là thời gian có thể cấu hình được trước khi biên dịch như yêu cầu đặt ra. Ba tín hiệu GREEN_time, RED_time và YELLOW_time báo thời điểm kết thúc của trạng thái GREEN, RED và YELLOW đi với clock 1 HZ.
- FSM: là máy trạng thái sẽ tạo ngõ ra traffic1, traffic2, outyl1, outyl2, train1, train2, seg1, seg2 và FSM_

CHƯƠNG III: CÁC HÀM CƠ BẢN CỦA VERILOG TRONG VIVADO

1. Tổng quan về VIVADO

Vivado là một phần mềm thiết kế logic trên thiết bị lập trình Logic (FPGA) và các hệ thống tùy chỉnh (SoC) của Xilinx. Vivado cung cấp một môi trường tích hợp đầy đủ cho các giai đoạn của quá trình thiết kế, từ mô phỏng đến đánh giá và triển khai trên thiết bị thực tế. Nó cung cấp các công cụ và tính năng để tối ưu hóa hiệu suất, tiết kiệm thời gian và tăng độ chính xác trong thiết kế. Vivado cũng hỗ trợ nhiều ngôn ngữ lập trình, bao gồm Verilog, VHDL và SystemVerilog.

2. Tổng quan về ngôn ngữ VERILOG

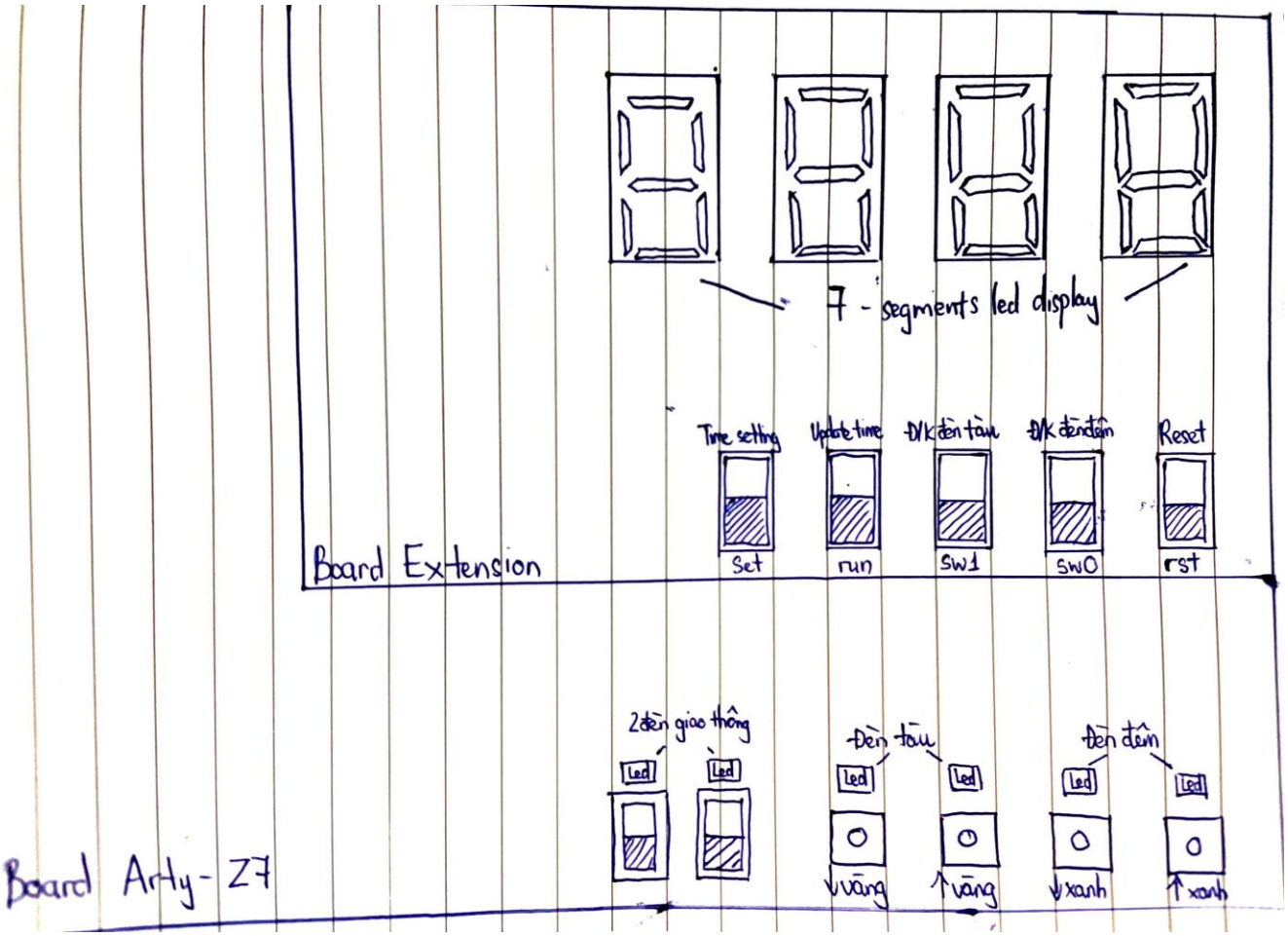
Verilog HDL là một trong hai ngôn ngữ mô phỏng phần cứng thôngdụng nhất, được dùng trong thiết kế IC, ngôn ngữ kia là VHDL. HDL chophép mô phỏng các thiết kế dễ dàng, sửa chữa lỗi, hoặc thực nghiệm bằngnhững cấu trúc khác nhau. Các thiết kế được mô tả trong HDL là những kỹthuật độc lập, dễ thiết kế, dễ tháo gỡ, và thường dễ đọc hơn ở dạng biểu đồ,đặc biệt là ở các mạch điện lớn.

Verilog thường được dùng để mô tả thiết kế ở bốn dạng:Thuật toán (một số lệnh giống ngôn ngữ C như: if, case, for,while…).Chuyển đổi thanh ghi (kết nối bằng các biểu thức Boolean). Các cổng kết nối(cổng: OR, AND, NOT…).Chuyển mạch (BJT, MOSFET)Ngôn ngữ này cũng chỉ rõ cách thức kết nối, điều khiển vào/ra trong mô phỏng.

CHƯƠNG IV: CODE HOÀN CHỈNH, WAFEFORM VÀ THỬ NGHIỆM

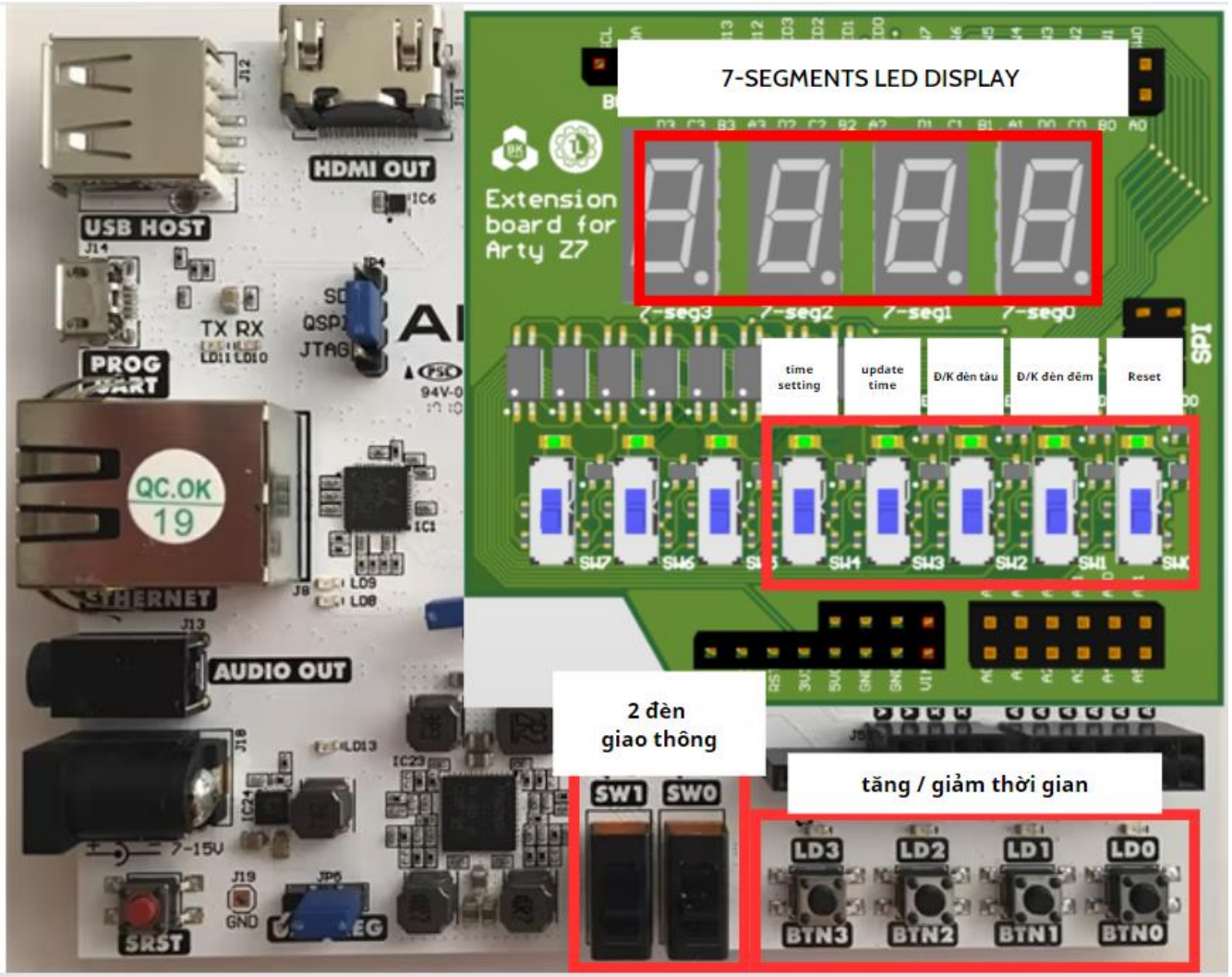
1. Các nút mô phỏng trên board ARTY-Z7 và Extension board for Arty Z7

1.1 Mô phỏng trên giấy



Hình 11: mô phỏng các nút chức năng trên giấy nháp

1.2 Mô phỏng thực tế



Hình 12: mô phỏng các nút chức năng trên board thực tế

2. Code hoàn chỉnh

```
1  `timescale 1s / 1ms
2  //change mode "night" or "train"
3  module mode(clk_05hz,clk_2hz,sw0, sw1, outyl1, outyl2, train1, train2);
4      input clk_05hz,clk_2hz,sw0, sw1;
5      output reg outyl1, outyl2, train1, train2;
6      always @ (clk_05hz,clk_2hz,sw0,sw1) begin
7          if (sw0 == 1) begin
8              if (sw1 == 1) begin
9                  outyl1 <= clk_05hz;
10                 outyl2 <= clk_05hz;
11                 train1 <= clk_2hz;
12                 train2 <= ~clk_2hz;
13             end
14             else begin
15                 outyl1 <= clk_05hz;
16                 outyl2 <= clk_05hz;
17                 train1 <= 1'b0;
18                 train2 <= 1'b0;
19             end
20         end
21         else begin
22             if (sw1 == 1) begin
23                 outyl1 <= 1'b0;
```

```
24              outyl2 <= 1'b0;
25              train1 <= clk_2hz;
26              train2 <= ~clk_2hz;
27          end
28          else begin
29              outyl1 <= 1'b0;
30              outyl2 <= 1'b0;
31              train1 <= 1'b0;
32              train2 <= 1'b0;
33          end
34      end
35  end
36  endmodule
37  //counter for traffic light 1 and traffic light 2
38  module count_time1(clk_1hz,G1,Y1,sw0,set,rst,seg,run);
39  input clk_1hz,sw0,set,rst,run;
40  input [7:0] G1,Y1;
41  output [7:0] seg;
42  reg [7:0] pre_count=8'd7,count=8'd7;
43  assign seg[7:0] =       count == 8'd100 ? 8'b1010_1010 :
44                          count == 8'd88 ?  9'b1000_1000 :
45                          count == 8'd30 ?  8'b0011_0000 :
46                          count == 8'd29 ?  8'b0010_1001 :
47                          count == 8'd28 ?  8'b0010_1000 :
```

```
48                  count == 8'd27 ?  8'b0010_0111 :
49                  count == 8'd26 ?  8'b0010_0110 :
50                  count == 8'd25 ?  8'b0010_0101 :
51                  count == 8'd24 ?  8'b0010_0100 :
52                  count == 8'd23 ?  8'b0010_0011 :
53                  count == 8'd22 ?   8'b0010_0010 :
54                  count == 8'd21 ?   8'b0010_0001 :
55                  count == 8'd20 ?  8'b0010_0000 :
56                  count == 8'd19 ?  8'b0001_1001 :
57                  count == 8'd18 ?  8'b0001_1000 :
58                  count == 8'd17 ?  8'b0001_0111 :
59                  count == 8'd16 ?  8'b0001_0110 :
60                  count == 8'd15 ?  8'b0001_0101 :
61                  count == 8'd14 ?  8'b0001_0100 :
62                  count == 8'd13 ?  8'b0001_0011 :
63                  count == 8'd12 ?  8'b0001_0010 :
64                  count == 8'd11 ? 8'b0001_0001 :
65                  count == 8'd10 ? 8'b0001_0000 :
66                  count == 8'd9 ?  8'b0000_1001 :
67                  count == 8'd8 ?  8'b0000_1000 :
68                  count == 8'd7 ?  8'b0000_0111 :
69                  count == 8'd6 ?  8'b0000_0110 :
70                  count == 8'd5 ?  8'b0000_0101 :
71                  count == 8'd4 ?  8'b0000_0100 :
```

```
72                  count == 8'd3 ?  8'b0000_0011 :
73                  count == 8'd2 ?  8'b0000_0010 :
74                                   8'b0000_0001 ;
75      always @(posedge clk_1hz) begin
76      if(rst) begin
77      pre_count<= 8'd7;
78      count<= 8'd88;
79      end
80      else if(run) begin
81      pre_count<= G1;
82      count<=8'd88;
83      end
84      else if (sw0) begin
85      pre_count <= G1;
86       count <= 8'd100;
87      end
88      else if (set) begin
89      pre_count <= G1;
90       count <= G1;
91      end
92      else if (count > 1) begin
93        if(count==8'd100|| count==8'd88) count<=G1;
94        else
95        count <= count - 1;
```

```
96      end
97      else begin
98          case (pre_count)
99          G1+Y1: begin
100          count <= G1;
101          pre_count<=G1;end
102          G1: begin
103          count <= Y1;
104          pre_count <=Y1;end
105          Y1: begin
106          count <= G1+Y1;
107          pre_count <=G1+Y1;
108          end
109          default:begin
110          pre_count<= G1;
111          count<= G1+1;
112          end
113       endcase
114  end
115  end
116  endmodule
117  module count_time2(clk_1hz,G1,Y1,sw0,set,rst,seg,run);
118  input clk_1hz,sw0,set,rst,run;
119  input [7:0] G1,Y1;
```

```
120  output [7:0] seg;
121  reg [7:0] pre_count=8'd10,count=8'd10;
122  assign seg[7:0] =       count == 8'd100 ? 8'b1010_1010 :
123                          count == 8'd88 ?  9'b1000_1000 :
124                          count == 8'd30 ?  8'b0011_0000 :
125                          count == 8'd29 ?  8'b0010_1001 :
126                          count == 8'd28 ?  8'b0010_1000 :
127                          count == 8'd27 ?  8'b0010_0111 :
128                          count == 8'd26 ?  8'b0010_0110 :
129                          count == 8'd25 ?  8'b0010_0101 :
130                          count == 8'd24 ?  8'b0010_0100 :
131                          count == 8'd23 ?  8'b0010_0011 :
132                          count == 8'd22 ?  8'b0010_0010 :
133                          count == 8'd21 ?  8'b0010_0001 :
134                          count == 8'd20 ?  8'b0010_0000 :
135                          count == 8'd19 ?  8'b0001_1001 :
136                          count == 8'd18 ?  8'b0001_1000 :
137                          count == 8'd17 ?  8'b0001_0111 :
138                          count == 8'd16 ?  8'b0001_0110 :
139                          count == 8'd15 ?  8'b0001_0101 :
140                          count == 8'd14 ?  8'b0001_0100 :
141                          count == 8'd13 ?  8'b0001_0011 :
142                          count == 8'd12 ?  8'b0001_0010 :
143                          count == 8'd11 ?  8'b0001_0001 :
```

```
144                 count == 8'd10 ? 8'b0001_0000 :
145                 count == 8'd9 ?  8'b0000_1001 :
146                 count == 8'd8 ?  8'b0000_1000 :
147                 count == 8'd7 ?  8'b0000_0111 :
148                 count == 8'd6 ?  8'b0000_0110 :
149                 count == 8'd5 ?  8'b0000_0101 :
150                 count == 8'd4 ?  8'b0000_0100 :
151                 count == 8'd3 ?  8'b0000_0011 :
152                 count == 8'd2 ?  8'b0000_0010 :
153                                  8'b0000_0001 ;
154   always @(posedge clk_1hz) begin
155    if(rst) begin
156    pre_count <= 8'd10;
157       count <= 8'd88;
158       end
159     else if(run) begin
160       pre_count <= G1+Y1;
161       count <= 8'd88;
162     end
163     else if (sw0) begin
164    pre_count <= G1+Y1;
165       count <= 8'd100;
166     end
167     else if(set) begin
```

```
168     pre_count <= G1+Y1;
169     count <= G1+Y1;
170     end
171     else if (count > 1) begin
172     if(count==8'd100|| count==8'd88) count=G1+Y1;
173     else
174       count <= count - 1;
175     end
176     else begin
177         case (pre_count)
178         G1+Y1: begin
179         count <= G1;
180         pre_count<=G1;end
181         G1: begin
182         count <= Y1;
183         pre_count <=Y1;end
184         Y1: begin
185         count <= G1+Y1;
186         pre_count <=G1+Y1;
187         end
188         default:begin
189         pre_count<= G1+Y1;
190         count<= G1+Y1+1;
191         end
```

```
192       endcase
193     end
194   end
195   endmodule
196 //Finate State Machine
197 module FSM(traffic1,traffic2,clk_1hz,rst,set,run,sw0,G1,Y1);
198 input clk_1hz,set,sw0,rst,run;
199 input [7:0] G1,Y1;
200 output reg [2:0] traffic1,traffic2;
201 reg [2:0] state=3'd0, next_state=3'd0;
202 reg [7:0] counter=8'd1;
203   parameter
204         RED           = 3'b001,
205         YELLOW        = 3'b101,
206         GREEN         = 3'b100;
207     parameter         //      traffic1      traffic2
208         S0 = 3'd0,    //      GREEN           RED
209         S1 = 3'd1,    //      YELLOW          RED
210         S2 = 3'd2,    //      RED             GREEN
211         S3 = 3'd3,    //      RED             YELLOW
212         OFF =3'd4;
213    always @(state) begin
214         case (state)
215             S0: begin
```

```
216                 traffic1 = GREEN;
217                 traffic2 = RED;
218               end
219           S1: begin
220                 traffic1 = YELLOW;
221                 traffic2 = RED;
222               end
223           S2: begin
224                 traffic1 = RED;
225                 traffic2 = GREEN;
226               end
227           S3: begin
228                 traffic1 = RED;
229                 traffic2 = YELLOW;
230               end
231           OFF:begin
232                 traffic1 = 3'b000;
233                 traffic2 = 3'b000;
234               end
235           default: begin
236                 traffic1 = 3'b000;
237                 traffic2 = 3'b000;
238                 end
239       endcase
```

```
240     end
241     always @(posedge clk_1hz) begin
242         if(rst) begin
243             state<=OFF;
244             counter<=8'd2;
245              end
246         else if(run) begin
247             state<=OFF;
248              end
249         else if(set) begin
250             state<=OFF;
251              end
252         else if(sw0) begin
253             state<=OFF;
254              end
255         else begin
256             state <= next_state;
257              end
258     case (state)
259        S0:if (counter < G1) begin
260               counter <= counter + 8'd1;
261              end else begin
262                next_state <= S1;
263                counter <= 8'd1;
```

```
264                 end
265         S1:if (counter < Y1) begin
266                 counter <= counter + 8'd1;
267             end else begin
268                 next_state <= S2;
269                 counter <= 8'd1;
270             end
271         S2:   if (counter < G1) begin
272                 counter <= counter + 8'd1;
273             end else begin
274                 next_state <= S3;
275                 counter <= 8'd1;
276             end
277         S3:  if (counter < Y1) begin
278                 counter <= counter + 8'd1;
279             end else begin
280                 next_state <= S0;
281                 counter <= 8'd1;
282             end
283         OFF: begin
284                 next_state<=S0;
285                 counter<=8'd2;
286                 end
287     endcase
```

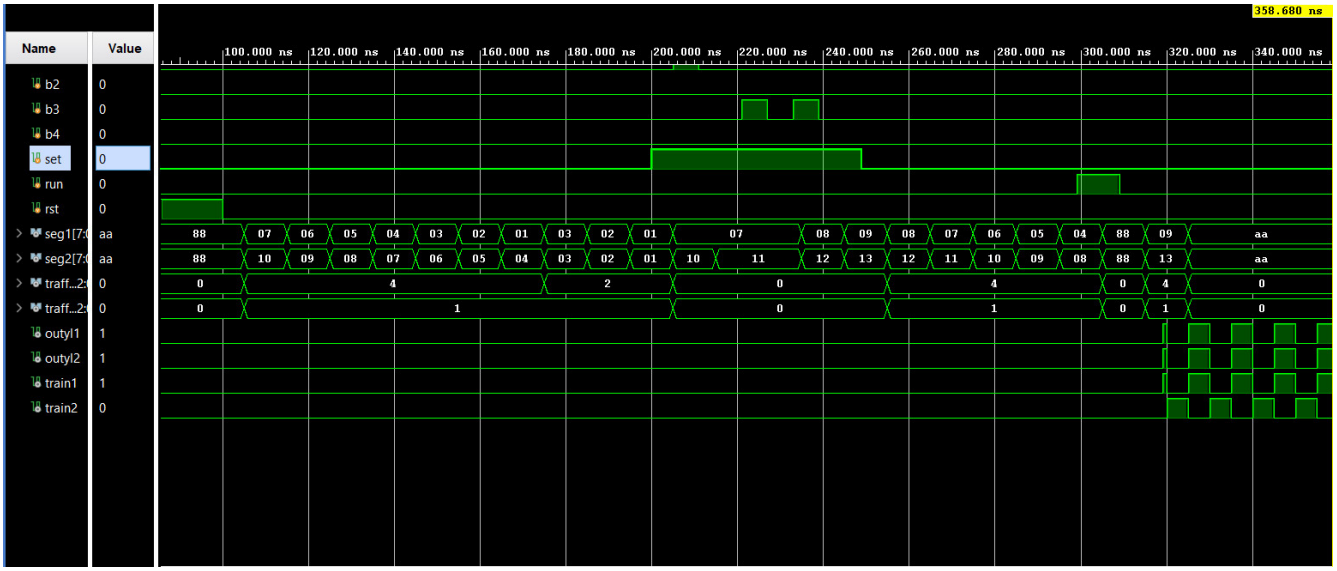
```
288   end
289   endmodule
290   //clk divide from 125MHz to 1 Hz
291   module clk_divider_1hz(
292      input clock_in,
293      output reg clock_out);
294      reg[27:0] counter=28'd0;
295   parameter DIVISOR = 28'd125000000;
296   always @(posedge clock_in)
297   begin
298    counter <= counter + 28'd1;
299    if(counter>=(DIVISOR-1))
300     counter <= 28'd0;
301    clock_out <= (counter<DIVISOR/2)?1'b1:1'b0;
302   end
303   endmodule
304   //clk divide from 125MHz to 0,5 Hz
305   module clk_divider_05hz(
306        input clock_in,
307   output reg clock_out);
308   reg[27:0] counter=28'd0;
309   parameter DIVISOR = 28'd250000000;
310   always @(posedge clock_in)
311   begin
```

```
312    counter <= counter + 28'd1;
313    if(counter>=(DIVISOR-1))
314     counter <= 28'd0;
315    clock_out <= (counter<DIVISOR/2)?1'b1:1'b0;
316   end
317   endmodule
318   //clk divide from 125MHz to 2 Hz
319   module clk_divider_2hz(
320        input clock_in,
321   output reg clock_out);
322   reg[27:0] counter=28'd0;
323   parameter DIVISOR = 28'd62500000;
324   always @(posedge clock_in)
325   begin
326    counter <= counter + 28'd1;
327    if(counter>=(DIVISOR-1))
328     counter <= 28'd0;
329    clock_out <= (counter<DIVISOR/2)?1'b1:1'b0;
330   end
331   endmodule
332   //top_module
333   module main_control(
334   input clk,sw0,sw1,bgreen1,bgreen2,byellow1,byellow2,set,rst,run,
335    output [7:0] seg1,
```

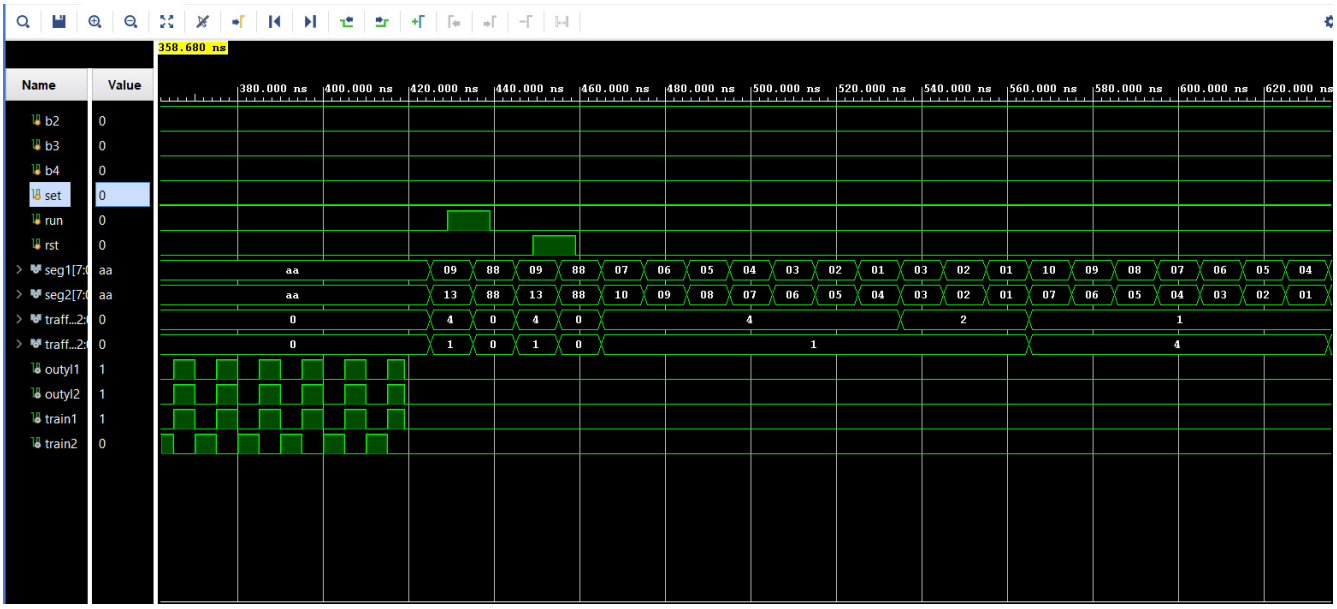
```
336   output [7:0] seg2,
337   output [2:0] traffic1,traffic2,
338   output outyl1,outyl2,
339   output train1,train2);
340   wire clk_1hz,clk_2hz,clk_05hz;
341   reg [7:0] Y_to_R_delay=8'd3;
342   reg [7:0] G_to_Y_delay=8'd7;
343   //
344   //////////////////////////////////////////////////*****/////////////////////////////////////
345   //PROGRAM:
346   //clock
347       clk_divider_05hz divider05(clk,clk_05hz);
348       clk_divider_1hz divider1(clk,clk_1hz);
349       clk_divider_2hz divider2(clk,clk_2hz);
350   //
351   //mode::counter
352      count_time1
c2(.clk_1hz(clk_1hz),.G1(G_to_Y_delay),.Y1(Y_to_R_delay),.sw0(sw0),.set(set)
,.seg(seg1),.rst(rst),.run(run));
353      count_time2
c1(.clk_1hz(clk_1hz),.G1(G_to_Y_delay),.Y1(Y_to_R_delay),.sw0(sw0),.set(set)
,.seg(seg2),.rst(rst),.run(run));
354   //
355   //mode::mode"night"||mode"train"
```

```
356      mode m1(.clk_2hz(clk_2hz),.clk_05hz(clk_05hz),.sw0(sw0), .sw1(sw1), .outyl1(outyl1), .outyl2(outyl2), .train1(train1), .train2(train2));
357  //
358  //machine::fsm
359      FSM f1(.traffic1(traffic1),.traffic2(traffic2),.clk_1hz(clk_1hz),.rst(rst),.run(run),.set(set),.sw0(sw0),.G1(G_to_Y_delay),.Y1(Y_to_R_delay));
360  //
361  //setting::increase/decrease time of lights
362      always @(posedge clk_1hz)
363      begin
364      if(rst) begin
365      Y_to_R_delay<=8'd3;
366      G_to_Y_delay<=8'd7;
367      end
368      else if(set) begin
369      if(byellow1) Y_to_R_delay<=Y_to_R_delay+8'd1;
370      else if(byellow2) Y_to_R_delay<=Y_to_R_delay-8'd1;
371      else if(bgreen1) G_to_Y_delay<=G_to_Y_delay+8'd1;
372      else if(bgreen2) G_to_Y_delay<=G_to_Y_delay-8'd1;
373      end
374      end
375  //
376  endmodule
```

3. Waveform

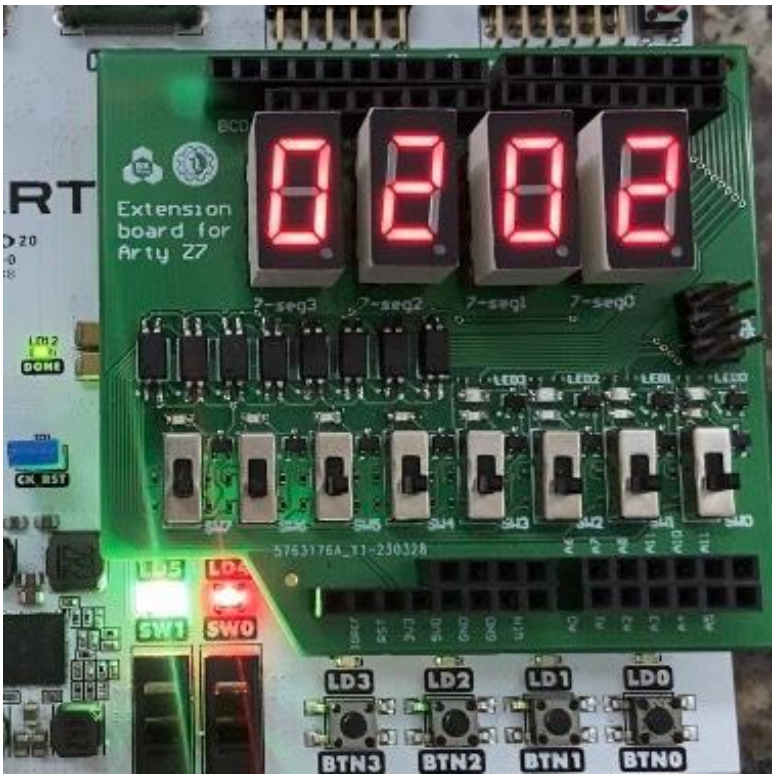


Hình 13: waveform



Hình 14: waveform

4. Mô phỏng thực tế

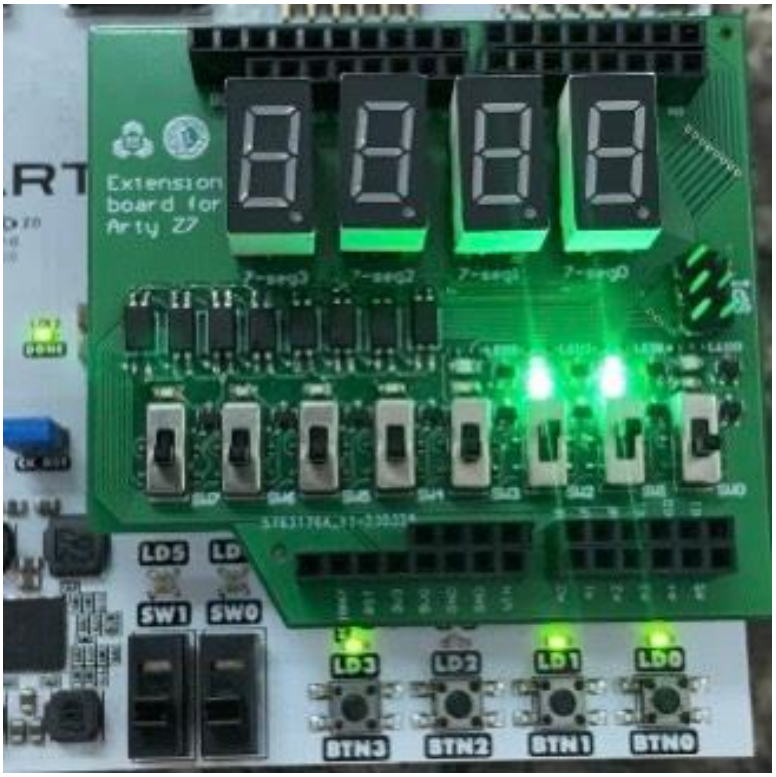


Hình 15:

tín hiệu đèn ở trạng thái bình thường

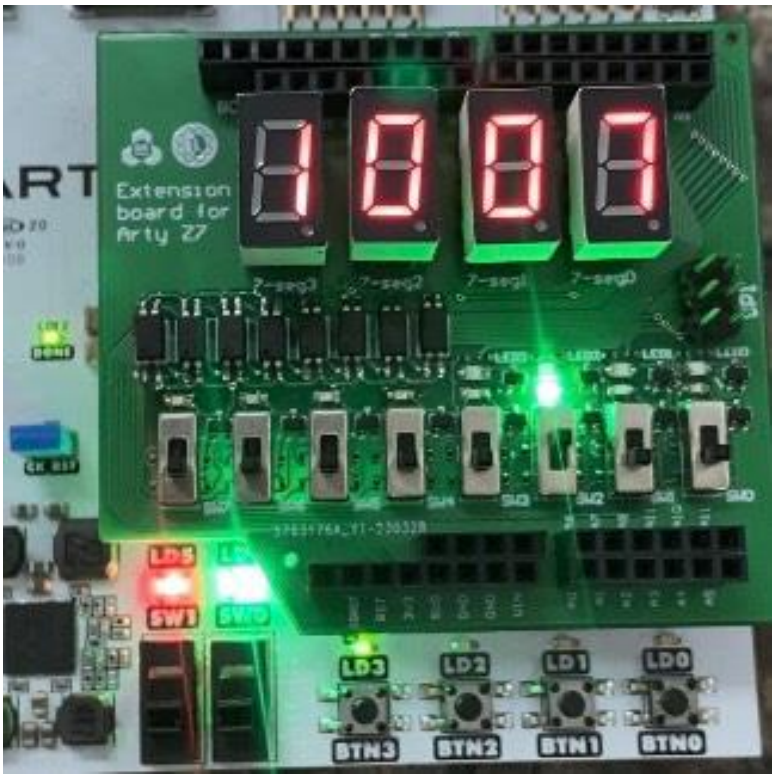
Hình 16:

tín hiệu đèn ở trạng thái ban đêm (sw0=1)



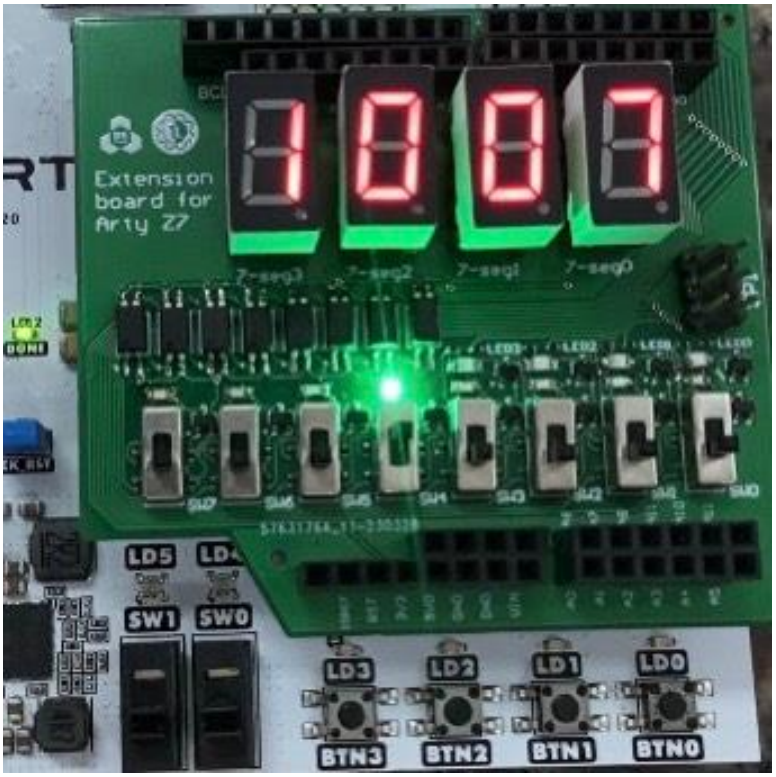
Hình 17:

tín hiệu đèn ở trạng thái ban đêm(sw0=1) và trạng thái tàu đi qua(sw1=1)



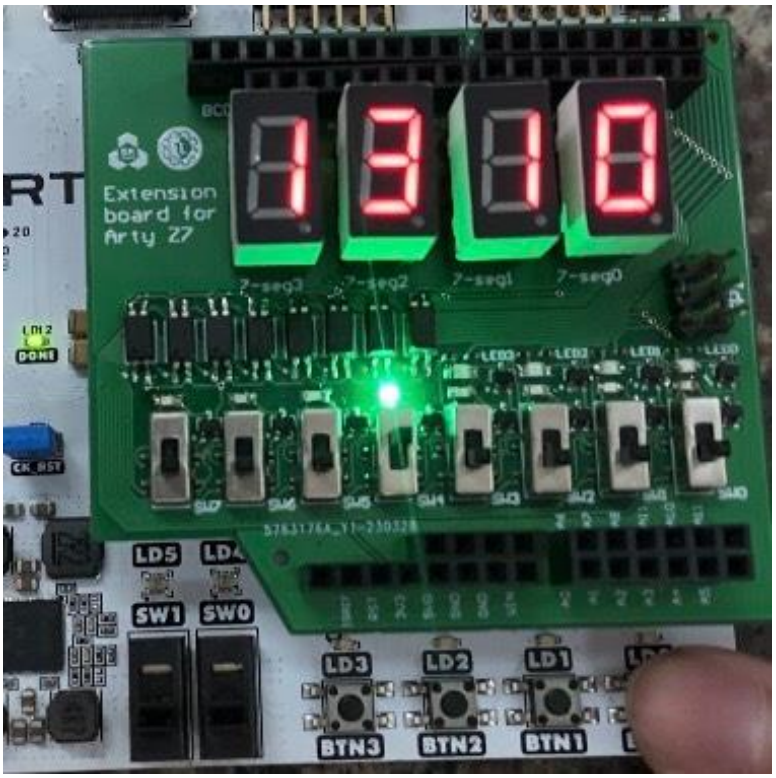
Hình 18:

tín hiệu đèn ở trạng thái tàu đi qua(sw1=1)



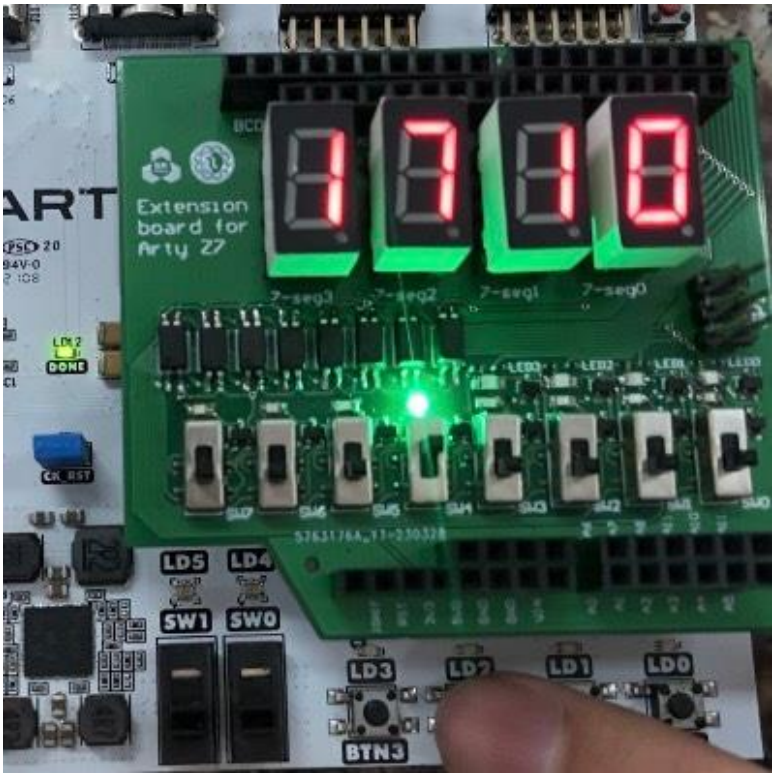
Hình 19:

tín hiệu đèn ở chế độ chỉnh thời gian(set=1)



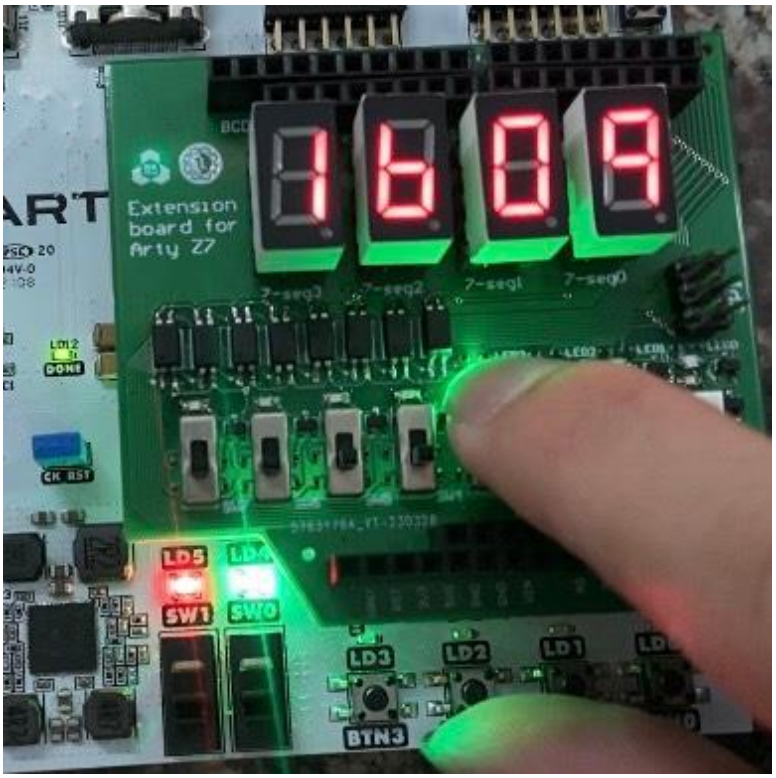
Hình 20:

tín hiệu đèn ở chế độ chỉnh thời gian(set=1) và đang chỉnh tang thời gian đèn tín hiệu xanh



Hình 21:

tín hiệu đèn ở chế độ chỉnh thời gian(set=1) và đang chỉnh tang thời gian đèn tín hiệu vàng



Hình 22:

tín hiệu đèn ở đang cập nhập lại sau khi thiết lập.

Hình 23:

tín hiệu đèn ở trạng thái bình thường sau khi thiết lập

CHƯƠNG V: KẾT LUẬN

Tổng kết báo cáo cho thấy rằng việc nghiên cứu và mô phỏng đèn tín hiệu giao thông tại các ngã tư giao nhau với đường sắt là một chủ đề quan trọng trong việc tối ưu hóa giao thông đô thị. Việc tăng tần suất đèn tín hiệu giao thông tại các ngã tư giao nhau với đường sắt có thể giảm thời gian chờ đợi và tăng khả năng thông suốt của giao thông tại những nơi này.

Việc sử dụng các hệ thống đèn tín hiệu thông minh cũng là một giải pháp hiệu quả để cải thiện tính an toàn và hiệu quả của giao thông tại những nơi này. Điều chỉnh độ dài của các chu kỳ tín hiệu giao thông tại các ngã tư giao nhau với đường sắt cũng là một yếu tố quan trọng để đảm bảo tính an toàn và hiệu quả của giao thông.