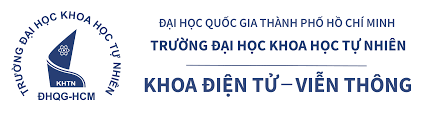
Shape, square

Description automatically generated

****

**ĐỒ ÁN VI ĐIỀU KHIỂN**

**ĐÈN GIAO THÔNG BẰNG PIC16F877A**

**SỬ DỤNG GIAO TIẾP UART**

A picture containing clipart, owl, bird, graphics

Description automatically generated

**Người hướng dẫn:**

Lê Đức Duy

Hoàng Hữu T

Nguyễn Đức Việt Bình

**Người thực hiện:**

Võ Thành Nhân 22200114

NĂM 2023

**Lời mở đầu**

Với sự phát triển nhanh chóng của khoa học công nghệ, làm gia tăng nhu cầu thiết yếu của con người mỗi chúng ta về nhiều mặt, theo đó những thiết bị như điện thoại, laptop, xe máy, ô tô, đồng hồ... là những thiết bị luôn đi kèm với chúng ta mọi lúc mọi nơi trong công việc, sinh hoạt, giải trí, nó là một phần thiết yếu trong thời đại xã hội phát triển này. Vậy đi kèm với những thiết bị diện tử sẽ có những thứ đi kèm với nó, nhắc tới thiết bị này sẽ nghĩ tới những thứ đi kèm như xe cô sẽ có đèn giao thông làm tín hiệu, Điện thoại phục vụ nhu cầu nghe gọi, giải trí, tìm kiếm thông tin nhanh chóng...

Trong đó, giao thông là một nhu cầu không thế thiếu trong mỗi chúng ta, nó phục vụ nhu cầu đi lại của con người, phục vụ con người di chuyển mọi nơi như đi làm, di du lich... Vây đế đảm bảo cho giao thông hoạt động một cách an toàn, giảm thiểu tai nạn giao thông, dặc biệt là những nơi ngã ba, ngã tư và nơi đông dân cư như các thành phố, quận huyện... thì dèn giao thông là công cụ giúp giải quyết vấn đề giảm thiểu tai nạn giao thông, giúp con người di chuyển qua những nơi giao nhau mà không gây tai nạn, gây cản trở giao thông. Đèn giao thông có mặt khắp nơi trên thể giới, đặc biệt là những nơi thành phố đông người, những nơi giao nhau, ta không hề khó bắt gặp những cột đèn giao thông như vậy. Ở bài này chúng ta sẽ nói về cách cấu tạo những cột đèn giao thông, cách thức hoạt động, cách cài đặt 1 bộ đèn giao thông thông qua PIC 16F877A, một loại vi điều khiến được sử dụng phố biến nhất vì tính linh hoạt trong hoat động, tính khả dụng và giá rẻ. Ở bài này ta sẽ nói thông tin chỉ tiết về PIC 16F877A và các linh kiện hỗ trợ việc tạo một bộ dèn giao thông ngoài dời thực và trên phần mềm mô phỏong Proteus.

**MỤC LỤC**

1**. Giới thiệu**

1.1 Tổng quan đề tài 6

1.2 Phân chia công việc 6

**2. Nguyên lý**

2.1 Mô hình ngã tư 7

2.2 Giản đồ thời gian 7

**3. Thiết kế phần cứng**

3.1 Linh kiện 8 3.2 Mô phỏng 19 3.3 Schematic 21

**4. Thiết kế phần mềm**

4.1 Lưu Đồ 24

4.2 Giải thuật 25

**DANH SÁCH ẢNH**

**PHỤ LỤC**

**1. GIỚI THIỆU**

**1.1 Tổng quan đề tài :**

Thiết kế hệ thống đèn giao thông ngã tư bằng pic16f877a , điều khiến bằng giao tiếp UART . Hệ thống có thể chuyển đổi giữa 2 chế độ: Tự động và điều khiển. Ở chế độ tự động thì thời gian của từng đèn được cài đặt trên máy tính thông qua giao tiếp UART, lúc này , đồng hồ led 7 đoạn được bật để thể hiện và đếm ngược thời gian của đèn. Ở chế độ điều khiển, chế độ này sẽ được điều khiển trên máy tính thông qua giao tiếp UART ,đồng hồ led 7 đoạn ở trạng thái tắt.

**1.2 Phân chia công việc trong nhóm:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | schematic | PCB | Code | Tiến độ |
| Chế độ tự động |  |  | Nhân | 100% |
| Chế độ điều khiến |  |  | Giang | 100% |
| Khối nguồn (5v) | Nhân | Nhân |  | 100% |
| Pic16f877a + 4 led giao thông  + 7\_seg + mạch cấp xung | Nhân | Nhân |  | 100% |
| Đổ mạch | Giang | |  |  |  |
| Hàn mạch |  |  |

**2. Nguyên Lí**

**2.1: Mô hình ngã tư.**

A diagram of a traffic light

Description automatically generated

**4**

**3**

**2**

**1**

Mô hình ngã tư

**2.2: Giản đồ thời gian của đèn.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Đèn 1,3 |  |  |  |  |  |  |  |  |  |
| Đèn 2,4 |  |  |  |  |  |  |  |  |  |

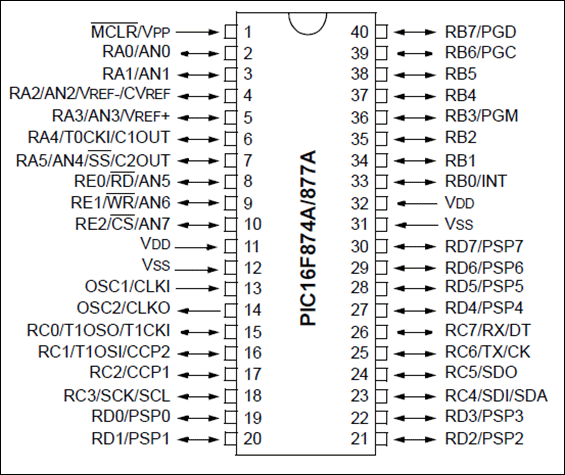
*Đèn ở chế độ tự động*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Đèn 1,3 | Tín Hiệu |  |  |  | Tín Hiệu |  |  |  |
| Đèn 2,4 |  |  |  |  |  |  |

*Đèn ở chế độ điều khiển*

**3. Thiết Kế PHẦN CỨNG**

**3.1: Linh kiện**

 **PIC16f877A ( số lượng 1 )**

**Chức năng các chân**

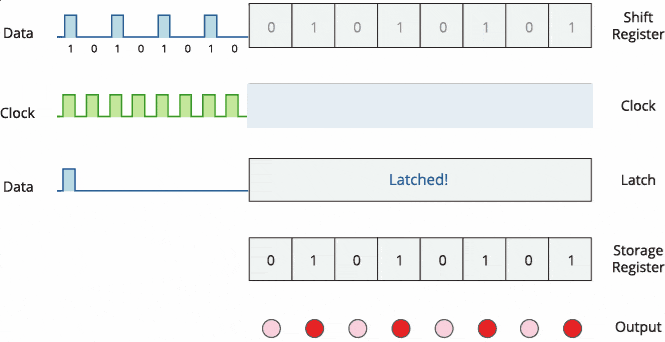
|  |  |  |
| --- | --- | --- |
| **Chân** | **Tên** | **Chức năng** |
| 1 | /MCLR/VPP | – /MCLR: Hoạt động Reset ở mức thấp – VPP : ngõ vào áp lập trình |
| 2 | RA0/AN0 | – RA0 : xuất/nhập số – AN0 : ngõ vào tương tự |
| 3 | RA1/AN1 | – RA1 : xuất/nhập số – AN1 : ngõ vào tương tự |
| 4 | RA2/AN2/VREF-/CVREF | – RA2 : xuất/nhập số – AN2 : ngõ vào tương tự – VREF -: ngõ vào điện áp chuẩn (thấp) của bộ A/D |
| 5 | RA3/AN3/VREF+ | – RA3 : xuất/nhập số – AN3 : ngõ vào tương tự – VREF+ : ngõ vào điện áp chuẩn (cao) của bộ A/D |
| 6 | RA4/TOCKI/C1OUT | – RA4 : xuất/nhập số – TOCKI : ngõ vào xung clock bên ngoài cho timer0 – C1 OUT : Ngõ ra bộ so sánh 1 |
| 7 | RA5/AN4//SS /C2OUT | – RA5 : xuất/nhập số – AN4 : ngõ vào tương tự 4 – SS : ngõ vào chọn lựa SPI phụ – C2 OUT : ngõ ra bộ so sánh 2 |
| 8 | RE0//RD/AN5 | – RE0 : xuất nhập số – RD : điều khiển việc đọc ở port nhánh song song – AN5 : ngõ vào tương tự |
| 9 | RE1//WR/AN6 | – RE1 : xuất/nhập số – WR : điều khiển việc ghi ở port nhánh song song – AN6 : ngõ vào tương tự |
| 10 | RE2//CS/AN7 | – RE2 : xuất/nhập số – CS : Chip lựa chọn sự điều khiển ở port nhánh song song – AN7 : ngõ vào tương tự |
| 11 | VDD | Chân nguồn của PIC |
| 12 | VSS | Chân nối đất |
| 13 | OSC1/CLKI | Ngõ vào dao động thạch anh hoặc xung clock bên ngoài. – OSC1 : ngõ vào dao động thạch anh hoặc xung clock bên ngoài. Ngõ vào Schmit trigger khi được cấu tạo ở chế độ RC; một cách khác của CMOS. – CLKI : ngõ vào nguồn xung bên ngoài. Luôn được kết hợp với chức năng OSC1. |
| 14 | OSC2/CLKO | Ngõ vào dao động thạch anh hoặc xung clock – OSC2 : Ngõ ra dao động thạch anh. Kết nối đến thạch anh hoặc bộ cộng hưởng. – CLKO : ở chế độ RC, ngõ ra của OSC2, bằng tần số của OSC1 và chỉ ra tốc độ của chu kỳ lệnh. |
| 15 | RC0/T1 OCO/T1CKI | – RC0 : xuất/nhập số – T1OCO : ngõ vào bộ dao động Timer 1 – T1CKI : ngõ vào xung clock bên ngoài Timer 1 |
| 16 | RC1/T1OSI/CCP2 | – RC1 : xuất/nhập số – T1OSI : ngõ vào bộ dao động Timer 1 – CCP2 : ngõ vào Capture 2, ngõ ra compare 2, ngõ ra PWM2 |
| 17 | RC2/CCP1 | – RC2 : xuất/nhập số – CCP1 : ngõ vào Capture 1, ngõ ra compare 1, ngõ ra PWM1 |
| 18 | RC3/SCK/SCL | – RC3 : xuất/nhập số – SCK : ngõ vào xung clock nối tiếp đồng bộ/ngõ ra của chế độ SPI – SCL : ngõ vào xung clock nối tiếp đồng bộ/ ngõ ra của chế độ I2C |
| 19 | RD0/PSP0 | – RD0 : xuất/nhập số – PSP0 : dữ liệu port nhánh song song |
| 20 | RD1/PSP1 | – RD1 : xuất/nhập số – PSP1 : dữ liệu port nhánh song song |
| 21 | RD2/PSP2 | – RD2 : xuất/nhập số – PSP2 : dữ liệu port nhánh song song |
| 21 | RD2/PSP2 | – RD2 : xuất/nhập số – PSP2 : dữ liệu port nhánh song song |
| 22 | RD3/PSP3 | – RD3: xuất/nhập số – PSP3 : dữ liệu port nhánh song song |
| 23 | RC4/SDI/SDA | – RC4 : xuất/nhập số – SDI : dữ liệu vào SPI – SDA : xuất/nhập dữ liệu vào I2C |
| 24 | RC5/SDO | – RC5 : xuất/nhập số – SDO : dữ liệu ra SPI |
| 25 | RC6/TX/CK | – RC6 : xuất/nhập số – TX : truyền bất đồng bộ USART – CK : xung đồng bộ USART |
| 26 | RC7/RX/DT | – RC7 : xuất/nhập số – RX : nhận bất đồng USART – DT : dữ liệu đồng bộ USART |
| 27 | RD4/PSP | – RD4: xuất/nhập số – PSP4 : dữ liệu port nhánh song song |
| 28 | RD5/PSP5 | – RD5: xuất/nhập số – PSP5 : dữ liệu port nhánh song song |
| 29 | RD6/PSP6 | – RD6: xuất/nhập số – PSP6 : dữ liệu port nhánh song song |
| 30 | RD7/PSP7 | – RD7: xuất/nhập số – PSP7 : dữ liệu port nhánh song song |
| 31 | VSS | Chân nối đất |
| 32 | VDD | Chân nguồn của PIC |
| 33 | RB0/INT | – RB0 : xuất/nhập số – INT : ngắt ngoài |
| 34 | RB1 | xuất/nhập số |
| 35 | RB2 | xuất/nhập số |
| 36 | RB3 | – RB3 : xuất/nhập số – Chân cho phép lập trình điện áp thấp ICPS |
| 37 | RB4 | – xuất/nhập số – Ngắt PortB |
| 38 | RB5 | – xuất/nhập số – Ngắt PortB |
| 39 | RB6/PGC | – RB6 : xuất/nhập số – PGC : mạch vi sai và xung clock lập trình ICSP – Ngắt PortB |
| 40 | RB7/PGD | – RB7 : xuất/nhập số – PGD : mạch vi sai và dữ liệu lập trình ICSP – Ngắt PortB |

**74HC595 ( số lượng 4 )**

* Điện áp hoạt động: 2< Vin <6V ( = Điện áp đầu ra)
* **Nhiệt độ hoạt động :** -40 đến 125 ‘C
* **Công dụng:** 74HC595 là thanh ghi dịch gồm 16 chân. Nó nhận dữ liệu đầu vào nối tiếp và dữ liệu gửi đi thông qua các chân song song

|  |  |  |
| --- | --- | --- |
| **Số chân** | **Tên chân** | **Mô tả** |
| 1, 2, 3, 4, 5, 6, 7 | Chân output (Q1 đến Q7) | 74hc595 có 8 chân đầu ra, trong đó có 7 chân này. Chúng có thể được kiểm soát nối tiếp |
| 8 | Ground | Nối đất |
| 9 | (Q7’) Serial Output | Chân này được sử dụng để kết nối nhiều hơn một 74hc595 dưới dạng xếp tầng |
| 10 | (MR) Master Reset | Reset tất cả các đầu ra ở mức thấp. Phải giữ ở mức cao để hoạt động bình thường |
| 11 | (SH\_CP) Clock | Đây là chân đồng hồ mà tín hiệu đồng hồ phải được cung cấp từ vi điều khiển hoặc vi xử lý (Cấp Xung) |
| 12 | (ST\_CP) Latch | Chân Latch dùng để cập nhật dữ liệu vào các chân đầu ra. Nó kích hoạt mức cao |
| 13 | (OE) Output Enable | Chân OE được sử dụng để tắt đầu ra. Phải giữ ở mức thấp để hoạt động bình thường |
| 14 | (DS) Serial Data | Đây là chân mà dữ liệu được gửi đến, dựa trên đó 8 đầu ra được điều khiển |
| 15 | (Q0) Output | Chân đầu ra đầu tiên |
| 16 | Vcc | Chân này cấp nguồn cho IC, thường sử dụng + 5V |

* **Chân MR** : khi ở mức LOW sẽ đặt lại Thanh Ghi ,đặt MR ở mức HIGH để 74HC595 hoạt động bình thường
* **Chân OE : khi ở mức LOW thì 74HC595 hoạt động bình thường, đặt OE ở mức HIGH thì đầu ra song song bị ngắt do trở kháng cao – đầu ra nối tiếp không ảnh hưởng.**
* Khi tín hiệu từ Chan DATA đưa vào cùng lúc với CLOCK (xung) ở mức HIGH thì thanh ghi dịch sẽ dịch bit, và khi LATCH ở mức HIGH thì thanh ghi lưu trữ sẽ chốt và xuất dữ liệu vào các chân output



* **Để Kết nối nhiều con 74HC595** thì ta mắc nối tiếp chân Q7’ của con trước vào chân Data của con sau ( nhằm truyền bit) – (có mô phỏng trong file: 74HC595\_S)

Diagram, schematic

Description automatically generated

**Led 7 đoạn 4 số 0.36inch Cathode chung ( số lượng 2 )**

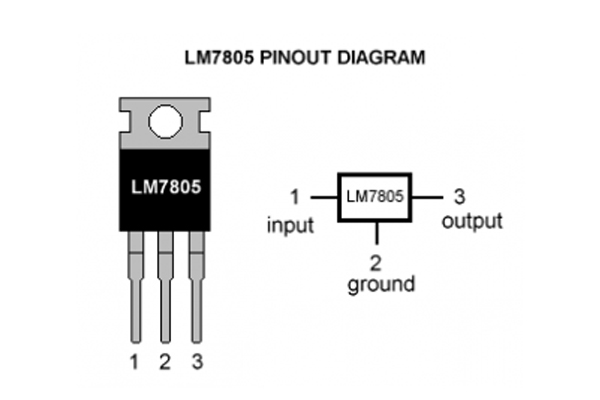
**Thông số kĩ thuật**

* + - K chung (K cathode) âm chung
    - 4 Dot
    - 12 chân
    - Điện áp rơi trên LED là 2.2V
    - Dòng tối đa chạy qua mỗi LED là 25mA
    - A picture containing text, diagram, sketch, drawing

      Description automatically generatedDòng chạy bình thường: 10mA. Nếu nguồn 5V thì mỗi Led phải nối với 1 điện trở 220R (dòng chạy qua mỗi led 13mA)

**LM7805 ( số lượng 1 )**

|  |  |
| --- | --- |
| **Datasheet** | 7805 |
| **Chân** | 3 |
| **Điện áp ngõ ra** | 5v |
| **Điện áp ngõ vào** | 7V – 18V DC |
| **Dòng ngõ ra** | 1A |
| **Nhiệt độ hoạt động** | 0°C – 125°C |
| **Công suất cực đại** | 5W |

 **Thông số kỹ thuật**

|  |  |
| --- | --- |
| Thông số kỹ 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,Yellow,Green |

 **Module đèn giao thông**

|  |  |
| --- | --- |
| **Linh kiện khác** | **Số lượng** |
| **Cầu diode** | **1** |
| **BJT (NPN)** | **8** |
| **Thạch anh 20Mhz** | **1** |
| **Nút bấm** | **1** |
| **Led vàng** | **1** |
| **Led đỏ** | **1** |

**Tụ Điện:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1nF** | **2200uF** | **100nF** | **33pF** |
| **Tụ gốm** | **1** |  | **1** |  |
| **Tụ Hóa** |  | **2** |  | **2** |

**Trở:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **10kΩ** | **1kΩ** | **220Ω** |
| **Trở** | **2** | **1** | **8** |

**3.2: Mô phỏng**

**A computer screen shot of a circuit board

Description automatically generated with low confidence Mạch mô phỏng trên proteus**

A screenshot of a computer

Description automatically generated **Chế độ tự động:**

Chế độ điều khiển:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**3.3: Schematic**

-Bản vẽ schematic gồm 5 khối : khối nguồn, khối chân cắm module , khối vi điều khiển, khối đèn led, khối cấp xung

**Khối Nguồn:** Sử dụng cầu diode để chỉnh lưu điện áp đầu vào , Tụ C1 để lọc điện áp đầu vào cấp cho IC ổn áp LM7805, C2 dùng để lọc điện áp đầu ra, C3 để lọc nhiễu

A screenshot of a computer

Description automatically generated**Khối cắm đèn**A diagram of a power supply

Description automatically generated**:** dùng để cắm module đèn giao thông

**Khối thạch anh:** A diagram of a circuit

Description automatically generateddùng để cấp xung cho mạch.

**Khối đèn led:** khối sử dụng 74hc595 để hỗ trợ xuất giá trị thời gian ra led 7 đoạn, C1815 dùng để khuếch đại điện áp .

A diagram of a circuit board

Description automatically generated

**Khối Vi điều khiển:** sử dụng Pic 16F877A làm vi điều khiển chính, nút bấm dùng để reset vi điều khiển, Header sử dung cho giao tiếp UART

A screenshot of a computer

Description automatically generated

**4. THIẾT KẾ PHẦN MỀM**

**4.1: LƯU ĐỒ THUẬT TOÁN**

Nhận tín hiệu

Tự động

Điều khiển

Sai

Đúng

Đổi loại đèn ?

Đổi chế độ ?

Nhận tín hiệu

Sai

Đúng

Sai

Đúng

Đúng

Đổi Thời gian ?

Đổi chế độ ?

Nhận tín hiệu

Chọn loại đèn

Xanh || Đỏ

Vàng

Vàng || Đỏ

Đếm ngược thời gian

Xanh || Đỏ

Đếm ngược thời gian

Nhập thời gian

Tự động/ Điều khiển

**4.2: GIẢI THUẬT**

Sơ lược về biến và hàm:

|  |  |  |  |
| --- | --- | --- | --- |
| **Biến** | **Kiểu dữ liệu** | **Chức năng** | **Mô tả** |
| count | Short | Xác định số chữ số của thời gian đèn xanh và vàng | Số chữ số là : giá trị của count + 1 |
| val | Char | Nhận chức năng của người sử dụng | Nhập “M” để chọn chế độ:   - Nhập “X” để chọn chế độ tự động,rồi nhập thời gian đèn,nếu muốn thay đổi thời gian thì ấn “C”  -Nhập “O” để chọn chế độ điều khiển. |
| Str[] | Short [3] | Chuỗi kí tự thời gian của đèn vàng hoặc xanh | Sau khi nhấn “X” , thì hiện sau bản nhập thời gian, cho từng đèn, ngay tại lúc nhập thì str[] lưu chuỗi kí từ đó. |
| Ginput | Short | Thời gian đèn xanh |  |
| Yinput | Short | Thời gian đèn vàng |  |
| In, check | Short | Kiểm tra tín hiệu đèn ở chế độ điều khiển |  |
| Ngan, tram, chuc, donvi | Short | Thời gian đèn sáng, xác định số cần xuất ra led | Thời gian đèn được biểu diễn tách riêng thành Ngàn, Trăm , Chục và Đơn vị để dịch bit và xuất ra led 7 đoạn |
| rc , gc , yc | Char | Bật tắt đèn | Các biến dc gán giá trị 1, sau 1 bán kì thì đảo giá trị ( = 0 ) để tắt đèn. |
| nc | Char | Điều kiện xuất led 7 đoạn | Nếu nc=1 thì 7seg\_1 xuất thời gian đèn xanh rồi vàng còn 7seg\_2 sẽ xuất thời gian đèn đỏ |
| Green | Short | Thời gian đèn xanh được gán vào vòng lặp, dùng để xuất ra led 7 đoạn | Biến được gán thời gian đèn đã nhập và trừ 1 với delay 950ms ( tương đối với 1 giây) để giảm dần giá trị mỗi giây và xuất ra led 7 đoạn  Red = Ginput + Yinput, |
| Yellow | Short | Thời gian đèn vàng được gán vào vòng lặp, dùng để xuất ra led 7 đoạn |
| Red | Short | Thời gian đèn Đỏ được gán vào vòng lặp, dùng để xuất ra led 7 đoạn |

|  |  |  |  |
| --- | --- | --- | --- |
| **Hàm** | **Kiểu dữ liệu** | **Chức năng** | **Mô tả** |
| uart\_init(int baudrate) | Void | Cài đặt UART |  |
| uart\_tx(char ch) | Void | Gửi tín hiệu vào UART |  |
| uart\_rx() | Char | Gửi tín hiệu UART vào vi điều khiển |  |
| uart\_sent(char \*p) | Void | Gửi tín hiệu từ vi điều khiển ra UART |  |
| clearScreen() | Void | Xóa màn hình giao diện trên mô phỏng |  |
| Init() | void | Cài đặt chế độ ngõ vào, ngõ ra của chân vi điều khiển |  |
| number[10] | int | Mã hóa số từ [0..9] để xuất ra led 7 đoạn |  |
| nUm\_OUT(int num)  nUm\_OUT2(int num) | Void | Cấp xung để cho thanh ghi dịch và đưa giá trị thời gian vào thanh ghi dịch | Đưa RB0/RB3 lên HIGH rồi hạ xuống LOW |
| lat() | Void | Chố và xuất dữ liệu đầu ra | Đưa RB1 lên HIGH rồi hạ xuống LOW |
| hold() | Void | Đưa giá trị đèn led về 0 | Kết hợp giữa nUm\_OUT(0b00000000) và lat() |
| seg( int input )  seg2( int input ) | void | Xuất giá trị thời gian ra led 7 đoạn , seg là thời gian của đèn 1,3 còn seg2 là thời gian của đèn 2,4 | Xuất giá trị số theo từng cột bằng nUm\_OUT( ),lat(), và hold( ). |
| pOw(short number) | int | Xác định trọng số của thời gian , chia số nhập vào thành Ngàn/trăm/chục/đơn vị |  |
| Input(int count) | void | Nhập thời gian đèn | Dùng giao tiếp UART |
| \_\_interrupt()isr\_uart(void) | Void | Ngắt chương trình chính để đi tới tác vụ khác rồi quay lại |  |

**Xử lí tín hiệu UART:**

Dùng biến **RCREG** ( 1 biến nhận giá trị của **uart\_rx()** ) để nhận giá trị gửi từ UART. Nếu giá trị đọc được là “M” thì bắt đầu chọn chế độ, nếu giá trị đọc được tiếp theo “X” thì vào chế độ tự động, còn là “O” thì vào chế độ điều khiển. Nếu muốn thay đổi chế độ thì gửi tín hiệu “M”

Ở chế độ điều khiển , nếu giá trị đọc được tiếp theo là “A” thì Đèn 1,3 sẽ là đèn xanh, còn “D” thì Đèn 2,4 là đèn xanh . Đèn tiếp tục giữ tín hiệu như trên cho tới khi có tín hiệu mới từ UART.

Ở chế độ tự động, nhận giá trị đèn xanh và vàng qua UART, sau đèn giao thông sáng theo thời gian cài đặt, đèn xanh đầu tiên mặc định là Đèn 1,3 . Sau đó, nếu muốn thay đổi thời gian đèn thì gửi tín hiệu “C” ,nếu muốn thay đổi chế độ thì gửi tín hiệu “M”

**Xuất led:**

Thời gian được nhập từ UART thông qua **input()** để chuyển thành số nguyên và chia trọng số ,sau đó đi qua **seg/seg2** để xuất ra led 7 đoạn

CODE

#pragma config FOSC = HS // Oscillator Selection bits (HS oscillator)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = OFF // Brown-out Reset Enable bit (BOR disabled)

#pragma config LVP = OFF // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)

#pragma config CPD = OFF // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF // Flash Program Memory Code Protection bit (Code protection off)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#define \_XTAL\_FREQ 20000000

//#include <math.h>

short count=2;

char val;

short check=1;

short in = 0;

short str[3];

short Ginput=0;

short Yinput=0;

void uart\_init(int baudrate){

TRISCbits.TRISC6 = 0;

TRISCbits.TRISC7 = 1;

SPBRG = ((\_XTAL\_FREQ/16)/baudrate) - 1;

TXSTA = 0B00100100;

RCSTA = 0B10010000;

}

void uart\_tx(char ch){

while(!TXIF);

TXREG = ch;

}

char uart\_rx(){

if(OERR){//==1

CREN = 0;

CREN = 1;

}

while(!RCIF);

return RCREG;

}

void uart\_sent(char \*p){

while(\*p){

uart\_tx(\*p++);

}

}

void init () //setup pin

{

TRISB = 0b00000000;

PORTB = 0b00000000;

TRISCbits.TRISC0=1;

TRISCbits.TRISC1=1;

TRISCbits.TRISC2=1;

TRISD = 0b00000000;

PORTD = 0b00000000;

PIE1bits.RCIE = 1; // Kích ho?t ng?t RDA

INTCONbits.PEIE = 1; // Kích ho?t ng?t ngo?i vi

INTCONbits.GIE = 1; // Kích ho?t ng?t toàn c?c

TXSTAbits.TXEN = 1; // B?t ch? ?? truy?n

RCSTAbits.SPEN = 1; // B?t c?ng n?i ti?p

RCSTAbits.CREN = 1; // B?t ch? ?? nh?n

}

void clearScreen()

{ while (!TXIF); // Ch? ??n khi thanh ghi truy?n s?n sàng

TXREG = 0x0C; // G?i ký t? ASCII c?a l?nh CLS (0x0C)

}

int number[10]={

0b11111100, //0

0b01100000, //1

0b11011010, //2

0b11110010, //3

0b01100110, //4

0b10110110, //5

0b10111110, //6

0b11100000, //7

0b11111110, //8

0b11110110, //9

};

void lat(){//Latch data

// Latch: RB2

// clock :RB0

RB2 = 1;

\_\_delay\_us(200);

RB2 = 0;

}

void nUm\_OUT(int num){ //d?ch bit Light\_1,3

// Latch: RB2

// clock :RB0

// Data:RB1

for (int i = 0;i<8;i++)

{

RB1 = (num >> i)&0b00000001;

RB0 = 1;

\_\_delay\_us(200);

RB0 = 0;

\_\_delay\_us(200);

}

}

void nUm\_OUT2(int num){//d?ch bit Light\_2,4

// Latch: RB2

// clock :RB0

// Data:RB1

for (int i = 0;i<8;i++)

{ RB3 = (num >> i)&0b00000001;

RB0 = 1;

\_\_delay\_us(200);

RB0 = 0;

\_\_delay\_us(200); }

}

void hold(){

nUm\_OUT(0b00000000);

nUm\_OUT(0b00000000);

lat();

\_\_delay\_us(200);

}

void seg( int input ) // out\_7Seg Light\_1,3

{ // number of wait time

short ngan;

short tram;

short chuc;

short donvi;

ngan=input/1000;

tram=(input/100)%10;

chuc=(input%100)/10;

donvi=(input%10);

nUm\_OUT( 0b00000000);

nUm\_OUT( 0b01110000);

nUm\_OUT( number[ngan]);

lat();

hold();

nUm\_OUT( 0b10110000);

nUm\_OUT( number[tram]);

lat();

hold();

nUm\_OUT( 0b11010000);

nUm\_OUT( number[chuc]);

lat();

hold();

nUm\_OUT( 0b11100000);

nUm\_OUT( number[donvi]);

lat();

hold();

}

void seg2( int input ) // out\_7Seg Light\_2,4

{ // number of wait time

short ngan;

short tram;

short chuc;

short donvi;

ngan=input/1000;

tram=(input/100)%10;

chuc=(input%100)/10;

donvi=(input%10);

nUm\_OUT2( 0b00000000);

nUm\_OUT2( 0b01110000);

nUm\_OUT2( number[ngan]);

lat();

hold();

nUm\_OUT2( 0b10110000);

nUm\_OUT2( number[tram]);

lat();

hold();

nUm\_OUT2( 0b11010000);

nUm\_OUT2( number[chuc]);

lat();

hold();

nUm\_OUT2( 0b11100000);

nUm\_OUT2( number[donvi]);

lat();

hold();

}

int pOw(short number) // exponent

{ short result = 1.0;

for (int i = 0; i < number; i++)

result \*= 10;

return result;

}

void input(int count )

{ TXREG= '\r';

uart\_sent(" Enter the time for the green light: ");

// input Red light

while(count != -1)

{ str[count]=(int)uart\_rx() - 48;

Ginput+=(str[count]\*pOw(count) );

count--; }

uart\_sent(" second ");

count=2;

TXREG= '\r';

uart\_sent(" Enter the time for the Yellow light: ");

// input Yellow light

while(count != -1)

{ str[count]=(int)uart\_rx() - 48;

Yinput+=(str[count]\*pOw(count));

count--; }

uart\_sent(" second ");

TXREG= '\r';

TXREG= '\r';

uart\_sent(" #Press C to change time ");

TXREG= '\r';

uart\_sent(" #Press M to change mode ");

}

void \_\_interrupt() isr\_uart(void)

{

if (PIR1bits.RCIF) // Ki?m tra c? ng?t RDA

{ if( RCREG == 'M')

{ uart\_sent(" Mode: ");

TXREG= '\r';

uart\_sent(" -press X for Automatic counter ");

TXREG= '\r';

uart\_sent(" -press O for Controller ");

TXREG='\r';

}

if( RCREG == 'X')

{ TXREG= '\r';

val='X';

clearScreen();

uart\_sent(" ---AutoMode--- ");

input( count );

}

if ( RCREG == 'C' )

{ clearScreen();

uart\_sent(" \*\*Change ");

TXREG= '\r';

val='X';

Ginput=0;

Yinput=0;

input(count);

}

if( RCREG == 'O' )

{ TXREG= '\r';

val='O';

clearScreen();

uart\_sent(" ---ControlMode--- ");

uart\_sent("CHON CHE DO ");

}

if (RCREG == 'D' )

{ uart\_sent("CHE DO DEN 13 XANH "); val = 'D'; }

if (RCREG == 'A' )

{ uart\_sent("C rôi"); val = 'A'; }

PIR1bits.TXIF = 0; // Xóa c? ng?t TXIF

PIR1bits.RCIF = 0; // Xóa c? ng?t RDA

}

}

void main(void) {

init();

uart\_init(9600);

// Timer\_Configuaration();

char rc=1;

char gc=1;

char yc=1;

char nc=1;

uart\_sent("Press M");

while(1)

{

if( val == 'X' ) //Auto

{

short Green =Ginput;

short Yellow =Yinput;

short Red=Ginput+Yinput;

while( Green != 0) // turn on green light

{

RD0=gc; //Green \_ light 1,3

RD1=0 ; //Yellow \_ light 1,3

RD2=!rc; //Red \_ light 1,3

RD3= !gc; //Green \_ light 2,4

RD4= 0; //Yellow \_ light 2,4

RD5=rc; //RED \_ light 2,4

if( nc == 1)

{

seg(Green );

seg2(Red);

}

if (nc == 0)

{

seg(Red);

seg2(Green);

}

Green--;

Red--;

\_\_delay\_ms(950);

}

while( Yellow != 0) // turn on Yellow light

{

RD0=0; //Green \_ light 1,3

RD1=yc ; //Yellow \_ light 1,3

RD2=!rc; //Red \_ light 1,3

RD3= 0; //Green \_ light 2,4

RD4= !yc; //Yellow \_ light 2,4

RD5=rc; //RED \_ light 2,4

if( nc == 1)

{

seg(Yellow);

seg2(Red);

}

if (nc == 0)

{

seg(Red);

seg2(Yellow);

}

Yellow--;

Red--;

\_\_delay\_ms(950);

}

rc=!rc;

gc=!gc;

yc=!yc;

nc=!nc;

}

if(val == 'D')// uart\_sent("CHE DO DEN 13 XANH ");

{

RD1 = 0 ;

RD2 = 0;

RD3 = 0;

RD4 = 0;

if( in == 0)

{ RD1 = 1;RD4 = 1; \_\_delay\_ms(1000); in = 1; }

RD5 = 1;

RD0 = 1 ;

}

if(val == 'A')//uart\_sent("CHE DO DEN 24 XANH ");

{ RD0 = 0 ;

RD1 = 0 ;

RD4 = 0;

RD5 = 0;

if( in == 1 || check == 1)

{ RD1 = 1;RD4 = 1; \_\_delay\_ms(1000); in =0;check =0;}

RD2 = 1;

RD3 = 1;

}

}

return;

}