1. **TỔNG QUAN ĐỀ TÀI**
2. Giới thiệu về đề tài

* Tên đề tài: Thiết kế mạch phát hiện và cảnh báo dò máu trong dây truyền dịch/dẫn dịch sử dụng cảm biến màu.

Hiện nay, trên thế giới có rất nhiều ca dò máu trong khi truyền dẫn dịch mà bác sỹ không thể phát hiện kịp thời để xử lý. Nó đã gây ra rất nhiều tai nạn đáng tiếc .Vì vậy một thiết bị phát hiện dò máu là rất cần thiết trong quá trình truyền dịch/dẫn dịch.

* Yêu cầu : Thiết kế được mạch phát hiện sự rò máu sử dụng LED RBG và quang trở LDR mạch có tính ứng dụng vào trong thực tế, một phần giúp các bác sỹ chăm sóc bệnh nhân tốt hơn.

1. **PHÂN TÍCH VÀTHIẾT KẾ HỆ THỐNG**
2. Ý tưởng thiết kế.

* Thiết kế một hệ thống phát hiện dò máu trong quá trình truyền dịch.
* Hệ thống thiết kế phải có độ chính xác trong mức độ cho phép.
* Hệ thống thiết kế phải có tính ứng dụng trong thực tế.

1. Công cụ thiết kế và lập trình

* Sử dụng công cụ thiết kế mạch Altium.
* Sử dụng công cụ lập trình MPLAB-C18 của Microchip.
* Sử dụng phần mềm mô phỏng Proteus8.

1. Cơ sở lý thuyết và giải thuật nhận diện màu sắc.

3.1 Cơ sở lý thuyết.

* Dựa trên hiện tượng vật lý quang điện trong .

 Hiện tượng tạo thành các electron dẫn và lỗ trống trong chất bán dẫn, do tác dụng của ánh sáng có bước sóngthích hợp  gọi là hiện tượng quang điện trong.

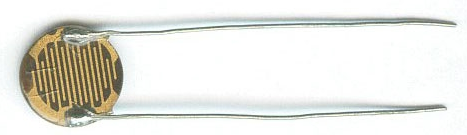
* Quang điện trở.

+ Quang trở là hiện tượng  điện trở của chất bán dẫn giảm đi khi khi bị chiếu sáng.

 + Cấu tạo: Lớp bán dẫn mỏng (cadimi sulfur CdS); tấm cách điện; điện cực bằng kim loại.

+ Hoạt động:

* Nối quang trở với nguồn ( vài volt ).
* Đặt quang trở trong bóng tối: Không có dòng điện.
* Chiếu  ánh sáng kích thích có bước sóng nhỏ hơn giới hạn quang dẫn: xuất hiện dòng điện.
* Điện trở của quang trở giảm rất mạnh khi có ánh  sáng kích thích thích hợp.



* Các định luật về phản xạ và hấp thụ màu sắc của vật thể.
* Lý thuyết về chuyển đổi ADC và lấy mẫu.

3.2 Giải thuật:

Sơ đồ thuật toán:

BLUE

GREEN

RED

RED

LDR

read

ADC

VI ĐIỂU KHIỂN

False True

**Phân tích**

- Đối tượng khi được đưa vào sẽ được chiếu sáng lần lượt bởi 3 LED siêu sáng màu red, green và blue và ánh sáng sẽ được hấp thụ lại bởi quang trở LDR để đưa ra mức điện áp tương ứng.

- Sử dụng bộ TIMER 0 của vi điều khiển để định thời gian sang tắt cho các LED .Tại 1 thời điểm chỉ có 1 LED được sáng và hai LED còn lại phải tắt. Thời gian sáng của LED là 200ms và thời gian tắt là 400ms. Mỗi thời điểm mà LED được bật thì đồng thời vi điều khiển sẽ đọc lại giá trị điện áp từ đầu ra của quang trở LDR .Giá trị điện áp này sẽ được lưu lại qua 1 biến tạm. Sau ba lần sáng lần lượt của 3 LED ta sẽ thu được 3 giá trị điện áp và 3 giá trị này sẽ được so sánh với nhau ,giá trị điện áp nào thấp nhất thì đối tượng sẽ có màu sắc gần với màu sắc của LED có giá trị điện áp đó . Do đó khi một vật màu đỏ đi qua thì giá trị điện áp tại LED màu đỏ sẽ là thấp nhất.

Giải thích:

- Do sự hấp thụ và phản xạ ánh sáng ( vật màu đỏ sẽ hấp thụ các ánh sáng có màu khác và phản xạ lại màu đỏ và tương tự đối với các vật có màu khác). Nên khi 1 vật màu đỏ đi qua thì nó sẽ phản xạ lại ánh sáng màu đỏ , tại thời điểm mà LED màu đỏ sáng thì ánh sáng màu đỏ sẽ có cường độ lớn nhất so với hai màu còn lại và quang trở sẽ hấp thụ mạnh nhất, do đó nội trở của quang trở sẽ giảm xuống thấp nhất. Nên khi qua cầu phân áp thì giá trị của điện áp sẽ thấp nhất. Tương tự đối với vật màu xanh nước biển và xanh lá cây thì giá trị điện áp tại LED blue, green sẽ là thấp nhất.

1. Phân tích thiết kế.

* Hệ thống thiết kế bao gồm cái khối sau:

VI XỬ LÝ

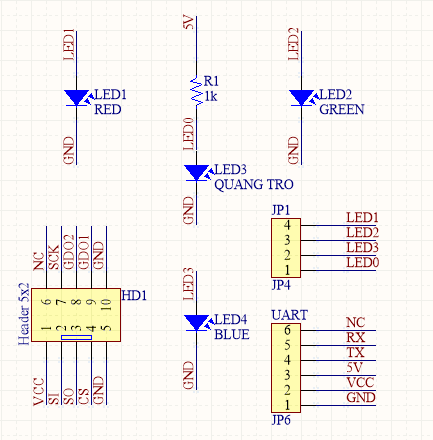
HIỂN THỊ

TÍN HIỆU

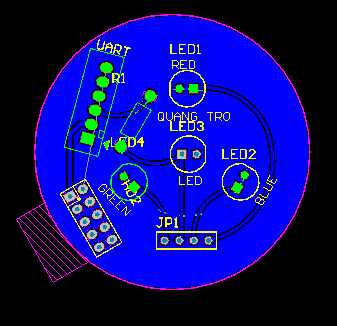
BLUETOOTH

* 1. **KHỐI TÍN HIỆU**

Sơ đồ mạch nguyên lý:



Sơ đồ mạch in:

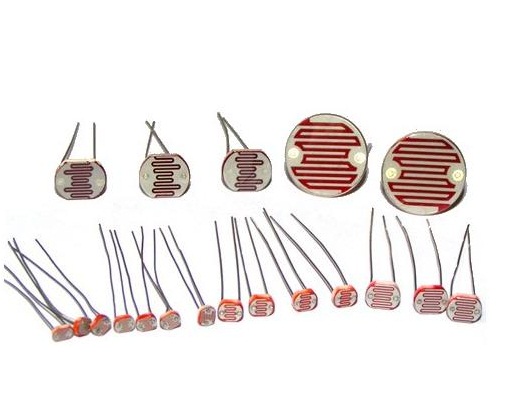


Khối tín hiệu sẽ được chia ra làm hai phần chính:

* + 1. **Khối phát tín hiệu:**
* Bao gồm có 3 LED siêu sáng RED, GREEN, BLUE.



* Các LED này sẽ được định thời bắng TIMER về thới gian sáng và tắt. Tại một thời điểm chỉ có 1 LED được sáng và hai LED kia sẽ tắt.
* Khi mỗi LED này được bật thì đồng thời giá trị điện áp từ đầu ra của khối thu tín hiệu sẽ được vi điều khiển đọc lại thông qua chân ANALOG , giá trị điện áp này sẽ được lưu lại thông qua qua 1 biến tạm. Sau 3 lần đọc giá trị điện áp tương ứng với 3 lần sáng của 3 led RED, GREEN, BLUE thì các giá trị này sẽ được so sánh với nhau và đưa ra màu sắc của vật thể.
  + 1. **Khối thu tín hiệu**
* Khối thu tín hiệu sử dụng quang trở LDR.

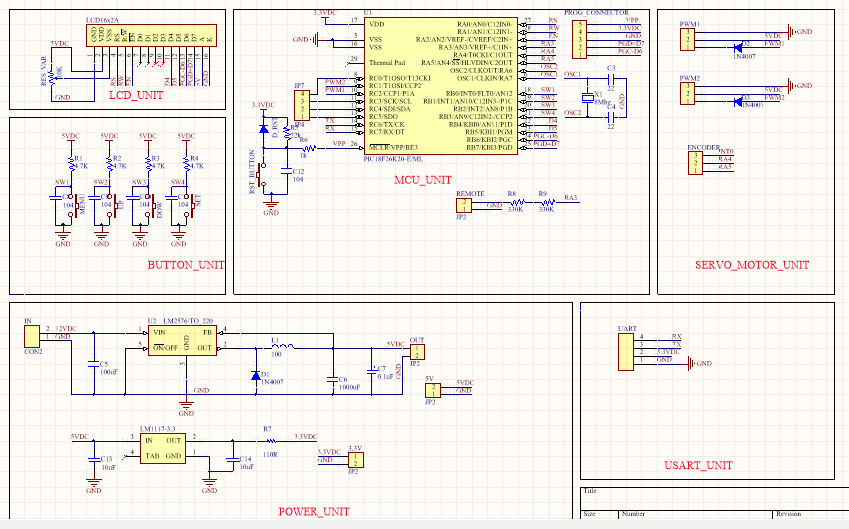


* Quang trở LDR là 1 loại cảm biến ánh sáng đơn giản , nguyên tắc hoạt động dựa trên hiện tượng quang điện trong.
* Nguyên lý làm việc của quang điện trở là khi ánh sáng chiếu vào chất bán dẫn (có thể là Cadmium sulfide – CdS, Cadmium selenide – CdSe) làm phát sinh các điện tử tự do, tức sự dẫn điện tăng lên và làm giảm điện trở của chất bán dẫn. Các đặc tính điện và độ nhạy của quang điện trở dĩ nhiên tùy thuộc vào vật liệu dùng trong chế tạo.
* Khi ánh sáng kích thích chiếu vào LDR thì nội trở của LDR sẽ giảm xuống , tiến về 0 ôm( mạch kín).
* Nhưng khi ánh sáng kích thích ngừng thì nội trở tăng đến vô cùng( hở mạch).

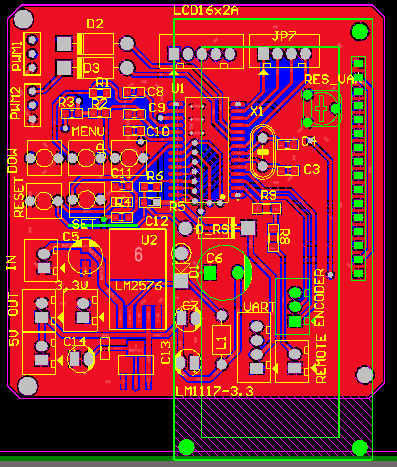
Chức năng:

* Tín hiệu phản xạ từ vật thể được quang trở hấp thụ lại và chuyển đổi thành mức điện áp tương ừng. Giá trị điện áp(tương tự) này được đưa qua bộ chuyển đổi ADC của PIC để được số hóa và chuyển thành giá trị tương ứng.
  1. **KHỐI VI XỬ LÝ**
* Khối vi xử lý có bộ xử lý trung tâm là PIC18F26K20.

Sơ đồ mạch nguyên lý:



Sơ đồ mạch in:



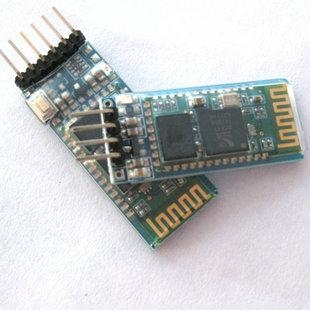
* Bộ xử lý này có các chức năng chính:

+ Chuyển đổi tín hiệu tương tự từ đầu ra của khối thu tín hiệu thành tín hiệu số với tốc độ lấy mẫu khoảng 1KHz.

+ Hiển thị cảnh báo lên.

+ Hiển thị cảnh báo lên điện thoại Android thông qua giao tiếp Bluetooth.

* 1. **KHỐI BLUETOOTH**
* Sử dụng module Bluetooth HC 05.
* Ưu điểm:



* Chức năng chính :

+ Hiển thị cảnh báo dò máu lên điện thoại Android.

* Chuẩn giao tiếp của Module với bộ xử lý là chuẩn UART.
  1. **KHỐI HIỂN THỊ**
* Sử dụng LCD 16x2
* Chức năng chính:

+ Hiển thị những cảnh báo và thông tin chính của hệ thống.

1. **TỔNG QUAN VỀ VI ĐIỀU KHIỂN PIC**
2. PIC là gì?

PIC là một họ vi điều khiển RISC được sản xuất bởi công ty Microchip Technology. Thế hệ PIC đầu tiên là PIC1650 được phát triển bởi Microelectronics Division thuộc General – Instrument.

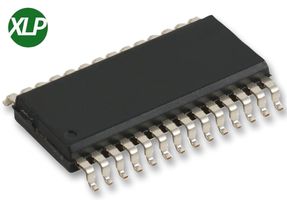
PIC là viết tắt của "Programmable Intelligent Computer" là một sản phẩm của hãng General Instruments đặt cho dòng sản phẩm đầu tiên là PIC1650. Thời điểm đó PIC1650 được dùng để giao tiếp với các thiết bị ngoại vi cho máy chủ 16 bit CP1600, vì vậy, người ta cũng gọi PIC với cái tên "Peripheral Interface Controller" – bộ điều khiển giao tiếp ngoại vi. CP1600 là một CPU mạnh nhưng lại yếu về các hoạt động xuất nhập vì vậy PIC 8-bit được phát triển vào năm 1975 để hỗ trợ cho hoạt động xuất nhập của CP1600.

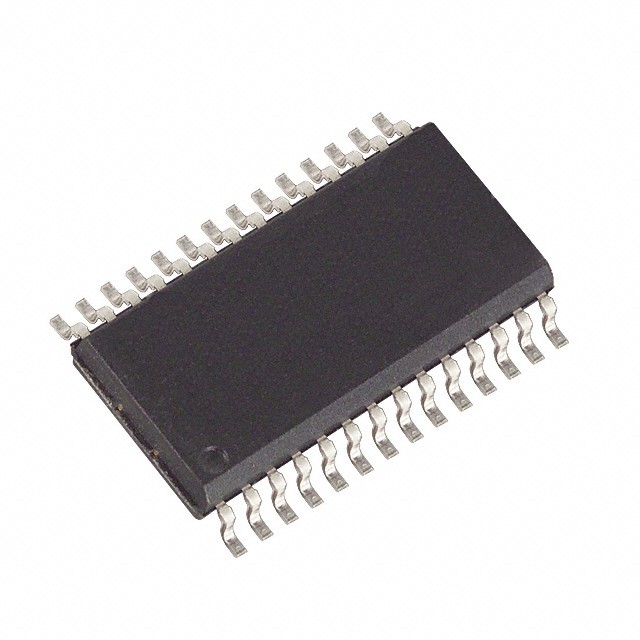
ROM để chứa mã, mặc dù khái niệm RISC chưa được sử dụng thời bấy giờ, nhưng PIC thực sự là một vi điều khiển với kiến trúc RISC, chạy một lệnh với một chu kỳ máy – gồm 4 chu kỳ của Oscillator.

Năm 1985 General Instruments bán công nghệ các vi điện tử của họ, và chủ sở hữu mới hủy bỏ hầu hết các dự án - lúc đó đã quá lỗi thời. Tuy nhiên PIC được bổ sung EEPROM đểtạo thành 1 bộ điều khiển vào ra lập trình.

Ngày nay rất nhiều dòng PIC được xuất xưởng với hàng loạt các module ngoại vi tích hợp sẵn (như USART, PWM, ADC...), với bộ nhớ chương trình từ 512

Word đến 32K Word





1. MỘT SỐ ĐẶC TÍNH CHUNG CỦA VI ĐIỀU KHIỂN

Hiện nay có khá nhiều dòng PIC và có rất nhiều khác biệt về phần cứng, nhưng chúng ta có thể điểm qua một vài nét như sau:

- Là CPU 8/16 bit, xây dựng theo kiến trúc Harvard.

- Có bộ nhớ Flash và ROM có thể tuỳ chọn từ 256 byte đến 256 Kbyte.

- Có các cổng xuất – nhập (I/O ports).

- Có timer 8/16 bit.

- Có các chuẩn giao tiếp nối tiếp đồng bộ/không đồng bộ USART.

- Có các bộ chuyển đổi ADC 10/12 bit.

- Có các bộ so sánh điện áp (Voltage Comparators).

- Có các khối Capture/Compare/PWM.

- Có hỗ trợ giao tiếp LCD.

- Có MSSP Peripheral dùng cho các giao tiếp I²C, SPI, và I²S.

- Có bộ nhớ nội EEPROM - có thể ghi/xoá lên tới 1 triệu lần.

- Có khối Điều khiển động cơ, đọc encoder.

- Có hỗ trợ giao tiếp USB.

- Có hỗ trợ điều khiển Ethernet.

- Có hỗ trợ giao tiếp CAN.

2.1 Các đặc tính USB (Universal Serial Bus) :

- Phù hợp với USB V2.0

- Tốc độ thấp (1.5 Mb/s) và Tốc độ Cao (12 Mb/s)

- Hỗ trợ Điều khiển, Ngắt, Đồng bộ và Truyền Khối

- Hỗ trợ lên tới 32 Endpoints ( 16 Endpoints mỗi chiều)

- 2 bộ nhớ RAM 1-Kbyte cho USB

- Bộ thu-phát USB On-Chip và Bộ ổn áp On-Chip

- Giao tiếp cho Bộ thu-phát USB Off-Chip

- Truyền Port song song (Streaming Parallel Port - SPP) cho truyền dữ

liệu USB ( chỉ thiết bị 40/44 chân).

2.2 Các chế độ quản lý năng lượng :

- Chạy: CPU chạy, thiết bị ngoại vi chạy.

- Chờ: CPU tắt, thiết bị ngoại vi chạy.

- Ngủ: CPU tắt, thiết bị ngoại vi tắt.

- Thoát khỏi chế độ chờ trong 5.8 µs.

- Thoát khỏi chế độ ngủ trong 0.1 µs.

- Oscillator Timer1: 1.1 µs, 32kHz, 2V.

- Watchdog Timer2: 2.1 µs.

- Khởi động Oscillator 2 tốc độ.

Fail-Safe Clock Monitor:

- Cho phép tắt an toàn nếu có bất kì xung clock nào dừng.

2.3 Các đặc tính ngoại vi nổi bật :

Dòng Sink/Source cao : 25 mA/ 25 mA

- 3 External Interrupts

- 4 module Timer ( Timer0 đến Timer3)

- Lên tới 2 Capture/Compare/PWM (CCP) module:

- Capture 16-bit, độ phân giải tối đa 5.2 ns

- Compare 16-bit, độ phân giải tối đa 83.3 ns

- Độ phân giải của PWM là 1 đến 10-bit

- Module Capture/Compare/PWM (ECCP) cải tiến:

- Các chế độ nhiều ngõ ra.

- Có thể chọn độ phân giải

- Có thể lập trình thời gian chờ

- Tự động shutdown và tự động restart.

- Module USART cải tiến:

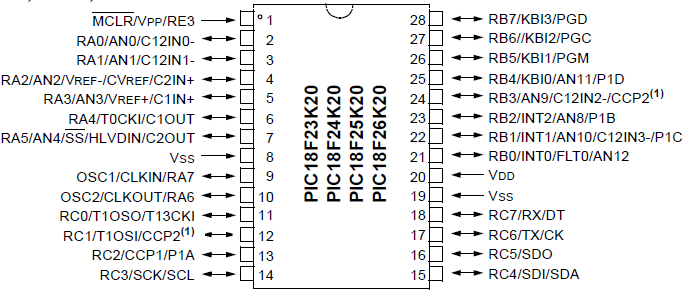
- Hỗ trợ LIN bus

- Module Master Synchronous Serial Port (MSSP) hỗ trợ 3-wire SPI (4

chế độ) và I2C Master và chế độ Slave.

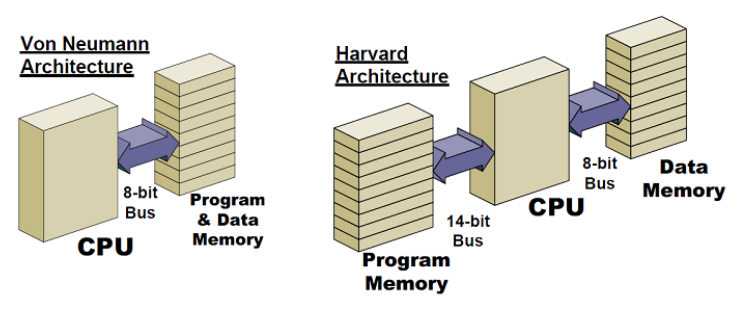
- 10-bit, lên tới 13 kênh ADC với khả năng lập trình thời gian nhận

- 2 bộ so sánh Analog với ngõ vào đa hợp



3.TỔ CHỨC BỘ NHỚ TRONG PIC18F26K20

PIC có kiến trúc bộ nhớ dạng Harvard, một kiến trúc cải tiến so với kiến trúc Von Neumann.

****

Hình: kiến trúc Harvard và kiến trúc Von Neumann

\* Kiến trúc Von Neumann: với kiến trúc này thì bộ nhớ giao tiếp với CPU thông qua 1 bus dữ liệu 8 bit, bộ nhớ có các ô nhớ chứa dữ liệu 8 bit, bộ nhớ vừa lưu trữ chương trình và dữ liệu.

- Ưu điểm: kiến trúc đơn giản.

- Khuyết điểm: do chỉ có 1 bus nên tốc độ truy suất chậm, khó thay đổi dung lượng lưu trữ của ô nhớ.

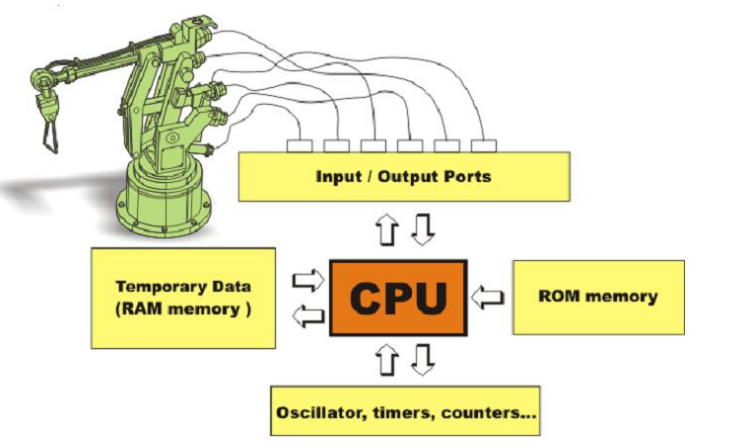
\* Kiến trúc Harvard: với kiến trúc này thì bộ nhớ được tách ra làm 2 loại bộ nhớ độc lập: bộ nhớ lưu chương trình và bộ nhớ lưu dữ liệu, CPU giao tiếp với 2 bộ nhớ độc lập nên cần 2 bus độc lập. Vì độc lập nên có thể thay đổi số bit lưu trữ của từng bộ nhớ mà không ảnh hưởng lẫn nhau. Với PIC thì bộ nhớ chương trình với mỗi ô nhớ lưu trữ 14 bit, còn bộ nhớ dữ liệu với mỗi ô nhớ lưu dữ liệu 8 bit.

- Ưu điểm: do chỉ có 2 bus nên tốc độ truy suất nhanh, tùy ý thay đổi số bit của ô nhớ.

- Khuyết điểm: kiến trúc phức tạp.

4. CẤU HÌNH CÁC PORT CỦA PIC18F26K20

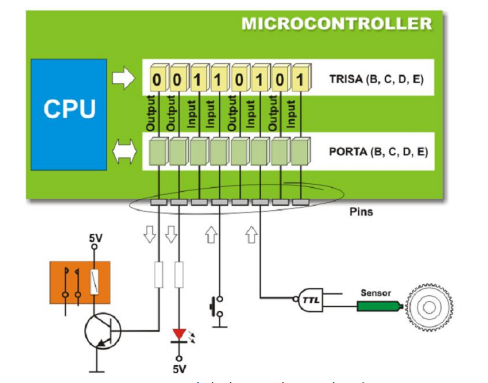
Vi điều khiển có các port để xuất nhập dữ liệu giao tiếp với các đối tượng điều khiển. Tín hiệu điều khiển từ CPU gởi ra port để điều khiển, đồng thời có các port nhận dữ liệu vềđể xử lý. Trong một hệ thống luôn có các tín hiệu vào ra ví như hệ thống điều khiển robo như hình sau:



Mỗi vi điều khiển khác nhau có cấu hình các port cũng khác nhau, phần này sẽ khảo sát các port của vi điều khiển PIC bao gồm port A, B, C, .Mỗi port của vi điều khiển PIC gồm có thanh ghi port và thanh ghi định hướng cho Port ví dụ

như hình 2.10 là PORTA và TRISA. Bit của thanh ghi định hướng TRIS bằng 0 thì port có Chức năng xuất dữ liệu, nếu bằng 1 thì có chức năng nhập dữ liệu.

Chú ý: '0' tương ứng với 'OUT', '1' tương ứng với 'IN'.



Sơ đồ kết nối xuất nhập điều khiển.

1. TIMER/ COUNTER CỦA PIC 18F26K20

5.1. Giới thiệu:

Vi điều khiển PIC 18F4550 có 4 timer T0, T1, T2 và T3. T0,T1 và T3 là timer/counter 6 bit cả 3 đều có bộ chia trước. T2 chỉ là timer/counter 8 bit có bộ chia trước và chia sau phục vụ cho các ứng dụng đặc biệt.

2.6.2. Khảo sát Timer/ Counter:

A. Khảo sát Timer0:

Module Timer0 có những đặc điểm sau:

- Có thể lựa chọn bằng phần mềm chế độ hoạt động Timer/Counter 8-bit hoặc 16-bit

- Có thể đọc và ghi thanh ghi

- Với chế độ 8-bit, có thể lựa chọn chia trước bằng phần mềm

- Có thể chọn lựa nguồn xung clock (trong hoặc ngoài)

- Lựa chọn cạnh lên hoặc xuống của xung clock ngoại

- Ngắt tràn

Ngắt của Timer0 được tạo ra khi thanh ghi TMR0 tràn từ FFh đến 00h ở chế độ 8-bit , hoặc từ FFFFh đến 0000h ở chế đô 16-bit. Việc tràn này làm bit cờ tràn TMR0IF từ 0 lên 1. Ngắt có thể ngăn bằng cách xóa bit cho phép ngắt TMR0IE (INTCON<5>). Trước khi cho phép ngắt lại, TMR0IF phải được xóa bởi phần mềm trong chương trình con ngắt.

Nếu Timer0 tắt trong chế độ Sleep, ngắt TMR0 không thể đánh thức vi xử lý khỏi Sleep.

B. Khảo sát Timer1:

Module Timer1 timer/counter có những đặc điểm sau:

- Lựa chọn chế độ 16-bit timer hoặc counter bằng phần mềm.

- Đọc và ghi lại các thanh ghi 8-bit (TMR1H và TMR1L).

- Chọn lựa nguồn xung clock với thiết bị tạo xung clock hoặc lựa chọn bộ dao động .

Timer1 nội.

- Ngắt tràn

- Module Reset ở Triger CCP sự kiện đặc biệt.

Cặp thanh ghi TMR1 (TMR1H:TMR1L) tăng từ 0000h đến FFFFh và quay lại 0000h. Nếu ngắt TMR1 cho phép, nó tạo ra việc tràn và set cờ tràn TMR1IF. (PIR1<0>) lên 1. Ngắt có thể bật hoặc tắt bằng set hoặc xóa TMR1IE(PIE1<0>).

C. Khảo sát Timer2:

Module Timer2 có những đặc điểm sau:

- Timer 8-bit.

- Có thể dùng phần mềm chia trước (1:1, 1:4 và 1:16) và chia sau (1:1 đến 1:16)

- Ngắt ở TMR2 đến khi trùng PR2.

- Lựa chọn xung clock lên cho module MSSP.

Timer 2 có thể tạo ra lựa chọn thiết bị ngắt. Ngõ ra tín hiện của Timer2 yêu cầu ngõ ra 4-bit của counter/postcaler. Ngắt có được cho phép bằng cách điều chỉnh bit TMR2IE (PIE1<1>). 16 chế độ chia trước có thể lựa chọn bằng các bit điều khiển chia trước T2OUTPS3:T2OUTPS0 (T2CON<6:3>).

D. Khảo sát Timer3:

Module timer/counter Timer3 có những đặc điếm sau:

- Lựa chọn bằng phần mềm chế độ 16-bit timer hoặc counter.

- Đọc và ghi các thanh ghi 8-bit (TMR3H và TMR3L).

- Lựa chọn nguồn xung clock với thiết bị tạo xung hoặc bộ dao động Timer1 bên trong.

- Ngắt tràn

Cặp thanh ghi TMR3 (TMR3H:TMR3L) tăng từ 0000h đến FFFFh và tràn ở 0000h. Ngắt Timer3 nếu cho phép sẽ tạo ra tràn và làm bit cờ tràn lên 1 TMR3IF (PIR2<1>). Ngắt cóthể cho phép bằng cách điều chỉnh bit TMR3IE (PIE2<1>).

1. **LINH KIỆN VÀ CÁC THÔNG SỐ KỸ THUẬT**
2. LED siêu sáng RBG.

1.1 Lý do.



LED siêu sáng có:

* Công suất phát trung bình từ 0.5 W- 1.5 Wcó thể lên tới 5W.
* Cường độ chiếu sáng lớn.
* Quang thông rộng.
* Tuổi thọ của LED cao có thể lên tới 100 nghìn giờ.
* Giá thành hợp lý phù hợp với sinh viên.

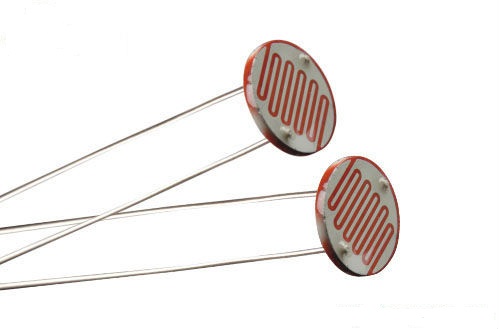
1.2 Các thông số kỹ thuật

+ Điện áp hoạt động tốt nhất từ 3.0V – 3.2 V.

+ Dòng qua LED từ 13mA – 20mA.

+ Công suất phát khoảng 1.2W.

1. QUANG TRỞ
   1. Lý do



* 1. Thông số kỹ thuật
* Điện trở trong của quang trở khoảng từ 3-4KΩ khi chưa có ánh sáng chiếu vào.
* Điện trở khi có ánh sáng chiếu vào có thể xuống tới 400Ω - 9KΩ.
* Dòng tối đa 750mA.
* Nhiệt độ hoạt động: -60­oC – +70oC.

1. PIC18F26K20

Thông số kỹ thuật:

+ Điện áp hoạt động 3.0V – 3.6V.

+ Dòng 25mA. Ở chế độ sleep :100nA – 1.8V

+Nhiệt độ hoạt động

1. HC-05

Thông số kỹ thuât:

+ Điện áp hoạt động 3.3V.

+ Dòng 25mA – 30mA.

+ Nhiệt độ làm việc -20oC – 70oC.

1. Điện trở phân áp

Sử dụng điện trở 2.2KΩ.

Lý do: Do nội trở của quang điện trở là vào khoảng 3KΩ nên khi được được chiếu sáng thì điện trở của nó giảm xuông khoảng 100- 200Ω. Lúc đo qua cầu phân áp thì tín hiệu điện áp của nó không quá nhỏ ta có thể đọc nó dễ dàng.

1. **PHẦN MỀM.**

**Thư viện**

LCD:

/\*

\* File: LCD.h

\* Author: Chu

\*

\* Created on April 30, 2014, 7:06 PM

\*/

#ifndef LCD\_H

#define LCD\_H

#include"p18f26k20.h"

#include"delays.h"

#define LCD\_RS LATAbits.LATA0

#define LCD\_RW LATAbits.LATA1

#define LCD\_E LATAbits.LATA2

#define LCD\_D4 LATBbits.LATB4

#define LCD\_D5 LATBbits.LATB5

#define LCD\_D6 LATBbits.LATB6

#define LCD\_D7 LATBbits.LATB7

//#define lcd\_busy LATBbits.LATB7

#define LCD\_clear 0x01 //Xoa man hinh LCD

#define LCD\_home 0x02 //Tro ve dau dong

#define LCD\_normal 0x06 //Che do nhap du lieu binh thuong

#define LCD\_normal\_s 0x07 //Che do nhap du lieu binh thuong, dich man hinh

#define LCD\_off 0x08 //Tat man hinh

#define LCD\_on 0x0C //Bat man hinh

#define LCD\_on\_crsr 0x0E //Bat man hinh va con tro

#define LCD\_on\_blink 0x0F //Bat man hinh va con tro (nhap nhay)

#define LCD\_4b2l 0x28 //Che do 2 dong, giao tiep 4 bit du lieu

#define LCD\_homeL1 0x80 //Ve dau dong 1

#define LCD\_homeL2 0xC0 //Ve dau dong 2

#define lcd\_line\_two 0x40

#define lcd\_line\_one 0x00

//---------------------------------------------------------------------------------------------------------

//Cac prototype cho cac chuong trinh con

void Init\_PORTS(void);

void Init\_LCD(void);

void LCD\_OK(void);

void LCD\_CMD(unsigned char cmd);

void LCD\_DATA(unsigned char dat);

void lcd\_putstr(char \*buffer);

void lcd\_gotoxy(int x, int y);

void khoi\_tao(void);

void LCD\_IN\_INTER(unsigned int d);

void LCD\_IN\_FLOAT( float d);

void lcd\_clear(void);

void lcd\_int(unsigned int a);

void lcd\_putch(char data);

#endif /\* LCD\_H \*/

#include "LCD.h"

void Init\_PORTS(void)

{

TRISAbits.TRISA0 = 0;

TRISAbits.TRISA1 = 0;

TRISAbits.TRISA2 = 0;

TRISBbits.TRISB4 = 0;

TRISBbits.TRISB5 = 0;

TRISBbits.TRISB6 = 0;

TRISBbits.TRISB7 = 0;

}

void Init\_LCD(void)

{

LCD\_CMD(0x02); //Dat che do giao tiep 4-bit, man hinh 2 dong

Delay1KTCYx(1);

LCD\_CMD(0x28); //Tat man hinh

Delay1KTCYx(1);

LCD\_CMD(0x0C); //Che do nhap du lieu binh thuong

Delay1KTCYx(1);

LCD\_CMD(0x06 ); //Bat man hinh, hien con tro nhap nhay

Delay1KTCYx(1);

//LCD\_CMD(0x01); //Xoa man hinh

Delay1KTCYx(1);

}

void LCD\_OK(void)

{

unsigned u;

LCD\_E = 1; //Tao xung Enable

for (u = 0; u < 2; u++); //keo dai toi thieu 1 us

LCD\_E = 0;

}

void LCD\_CMD(unsigned char cmd)

{

unsigned temp1;

LCD\_RW = 0;

LCD\_RS = 0; //Dat che do xuat lenh

temp1 = cmd >> 4; //Lay 4 bit cao

LCD\_D4 = temp1 & 1; //Xuat 4 bit cao

LCD\_D5 = (temp1 & 2) >>1;

LCD\_D6 = (temp1 & 4) >>2;

LCD\_D7 = (temp1 & 8) >>3;

LCD\_OK();

temp1 = cmd & 0x0F; //Lay 4 bit thap

LCD\_D4 = temp1 & 1; //Xuat 4 bit thap

LCD\_D5 = (temp1 & 2) >>1;

LCD\_D6 = (temp1 & 4) >>2;

LCD\_D7 = (temp1 & 8) >>3;

LCD\_OK();

Delay1KTCYx(5);

}

void LCD\_DATA (unsigned char dat)

{

unsigned temp2;

LCD\_RW = 0;

LCD\_RS = 1; //Dat che do xuat du lieu

temp2 = dat >> 4; //Lay 4 bit cao

LCD\_D4 = temp2 & 1; //Xuat 4 bit cao

LCD\_D5 = (temp2 & 2) >>1;

LCD\_D6 = (temp2 & 4) >>2;

LCD\_D7 = (temp2 & 8) >>3;

LCD\_OK();

temp2 = dat & 0x0F; //Lay 4 bit thap

LCD\_D4 = temp2 & 1; //Xuat 4 bit thap

LCD\_D5 = (temp2 & 2) >>1;

LCD\_D6 = (temp2 & 4) >>2;

LCD\_D7 = (temp2 & 8) >>3;

LCD\_OK();

Delay1KTCYx(5);

}

void lcd\_clear(void)

{

LCD\_CMD(0x01);

}

void lcd\_putch(char data)

{

LCD\_DATA(data);

}

void lcd\_putstr(char \*buffer)

{

while(\*buffer != '\0')

{

LCD\_DATA(\*buffer); /\* calling another function \*/

/\* to write each char to the lcd module \*/

buffer++;

Delay1KTCYx(2);

};

}

//-----------Chuong trinh con nhay den cot x, dong y-------------------------------------------------------------------

void lcd\_gotoxy(int x, int y)

{

int address;

if(y != 1) address = lcd\_line\_two;

else address=0;

address += x-1;

LCD\_CMD(0x80|address);

Delay1KTCYx(2);

}

void LCD\_IN\_INTER(unsigned int d)

{

unsigned i,j,k,l;

l=d/1000;//nghin

k=(d%1000)/100;//tram

j=((d%1000)%100)/10;//chuc

i=((d%1000)%100)%10;//donvi

LCD\_DATA(0x30+l);

LCD\_DATA(0x30+k);

LCD\_DATA(0x30+j);

LCD\_DATA(0x30+i);

}

void lcd\_int(unsigned int a)

{

unsigned int x,y, z;

x=a/100;

y=(a/10)%10;

z = a%10;

if(x>=1)

{

LCD\_DATA(0x30 +x);

LCD\_DATA(0x30+y);

LCD\_DATA(0x30+z);

}

else

{

LCD\_DATA(0);

LCD\_DATA(0x30+y);

LCD\_DATA(0x30+z);

}

}

//=======giu so thuc len lcd=============================

void LCD\_IN\_FLOAT( float d)

{

char ch[]=".";

unsigned int e, i,j,k,l;

// d=round(d,1);

e=(int)(d\*10);

l=e/1000;//nghin

k=(e%1000)/100;//tram

j=((e%1000)%100)/10;//chuc

i=((e%1000)%100)%10;//donvi

LCD\_DATA(0x30+l);

LCD\_DATA(0x30+k);

LCD\_DATA(0x30+j);

lcd\_putstr(ch);

LCD\_DATA(0x30+i);

}

Timer:

/\*

\* File: Timer.h

\* Author: Chu

\*

\* Created on May 26, 2014, 11:39 AM

\*/

#ifndef TIMER\_H

#define TIMER\_H

#include"p18f26k20.h"

// Function

void OSC\_config(void)

{

// config OSC= 4MHz

OSCCONbits.IRCF2=1;

OSCCONbits.IRCF1=0;

OSCCONbits.IRCF0=1;

}

void INT0\_config(void)

{

RCONbits.IPEN=0;//disable priority levels interrupt

INTCONbits.GIE=1;// disable global interrupt

INTCONbits.PEIE=1;

INTCONbits.INT0IE=1;

INTCONbits.INT0IF=0;

}

void interrupt\_config(void)

{

RCONbits.IPEN=0;//disable priority levels interrupt

INTCONbits.GIE=1;// disable global interrupt

INTCONbits.PEIE=1;

}

void timer0\_config(void)

{

T0CON=0x82;// prescaler = 1:8

TMR0H=0xF0;

TMR0L=0x00;

INTCONbits.GIEL=1;// disable global interrupt

INTCONbits.GIEH=1;

INTCONbits.TMR0IE=1;

INTCONbits.TMR0IF=0;

}

void timer1\_config(void)

{

T1CON=0x81;

/\*

T1CONbits.T1CKPS1=0;

T1CONbits.T1CKPS0=0;

T1CONbits.TMR1CS=0;

\*/

RCONbits.IPEN = 1;

IPR1bits.TMR1IP = 0;

INTCONbits.GIEL=1;

INTCONbits.GIEH=1;

PIR1bits.TMR1IF=0;

PIE1bits.TMR1IE=1;

TMR1H=0x00;

TMR1L=0x00;

}

void timer3\_config(void)

{

T3CON=0x81;

/\*

T3CONbits.RD16=1;

T3CONbits.T3CCP2=0;

T3CONbits.T3CCP1=1;

T3CONbits.T3CKPS1=0;

T3CONbits.T3CKPS0=0;

T3CONbits.TMR3CS=0;

\*/

INTCONbits.GIE=1;

INTCONbits.PEIE=1;

PIR2bits.TMR3IF=0;

PIE2bits.TMR3IE=1;

TMR3H=0x00;

TMR3L=0x00;

T3CONbits.TMR3ON=1;

}

#endif /\* TIMER\_H \*/

UART:

/\*

\* File: UART.h

\* Author: Chu

\*

\* Created on June 2, 2014, 6:03 PM

\*/

#ifndef UART\_H

#define UART\_H

#include"p18f26k20.h"

void UART\_Config(void);

void uart\_puts(const char \*s);

void uart\_putc(char c);

char uart\_getc();

void uart\_gets(char \*s);

void UART\_isr();

void uart\_interrupt();

void Init\_Rec\_UART(void);

void Init\_Tran\_UART(void) ;

void UART\_Config(void)

{

/\* relates crystal freq to baud rate - see above and PIC18f26k20 data sheet under async 'USART'

BRGH=1, Fosc=3.6864MHz BRGH=1, Fosc=4MHz BRGH=1, Fosc=8MHz BRGH=1, Fosc=16MHz

---------------------- --------------------- ----------------- ------------------

Baud SPBRG Baud SPBRG Baud SPBRG Baud SPBRG

1200 191 1200 207.3 1200 415.7 9600 103

2400 95 2400 103.2 2400 207.3 19200 51

4800 47 4800 51.1 4800 103.2 38400 25

9600 23 9600 25.0 9600 51.1 57600 16

19200 11 19200 12.0 19200 25.0 115200 8

38400 5 38400 5.5 38400 12.0

57600 3 57600 3.3 57600 7.7

115200 1 115200 1.2 115200 3.3

\*/

//you can comment these #assert statements out if you dont want error checkin

BAUDCONbits.BRG16 =1;//16-bit Baud Rate Generator is used (SPBRGH:SPBRG)

SPBRGH:SPBRG = 207;//

TXSTAbits.BRGH = 1; //data rate for sending/receiving

TXSTAbits.SYNC =0; //Truyen khong dongbo

RCSTAbits.SPEN =1; //enable serial port pin // Enable transmitter

TRISCbits.TRISC6 = 0; // RC6 is pin trasmit

TRISCbits.TRISC7 = 1; //RC7 is pin receive

TXREG =0x00; //lam rong bo truyen

}

void Init\_Tran\_UART(void)

{

BAUDCONbits.CKTXP=0; // Idle state for transmit (TX) is high

TXSTAbits.TXEN =1; // cho phep truyen uart

TXREG = 0x00; // lam rong thanh ghi truyen

}

void Init\_Rec\_UART(void)

{

RCONbits.IPEN=1;

IPR1bits.RCIP=0;

PIE1bits.RCIE=1; // cho phep ngat khi nhan uart

INTCONbits.GIE=1; // cho phep ngat toan cuc khi IPEN =1 set boi thanh ghi RCON

INTCONbits.PEIE=1; // cho phep ngat ngoai vi

RCSTAbits.CREN =1; // cho phep nhan uart khong dong bo

PIR1bits.RCIF=0;//tat co ngat

RCREG = 0x00;

}

void uart\_putc(char c)

{

while(!TXSTAbits.TRMT);//bo truyen rong

TXREG = c;//gui ky tu

}

void uart\_puts(const char \*s)

{

while(\*s != '\0')

{

uart\_putc(\*s);

s++;

};

}

char uart\_getc()

{

while(!PIE1bits.RCIE);

return RCREG;//cho phep nhap du lieu

}

void uart\_gets(char \*s)

{

\*s = uart\_getc();

while(\*s!='\0')

{

s++;

\*s = uart\_getc();

};

}

#endif /\* UART\_H \*/

Main:

/\*

\* File: TranRF315.c

\* Author: Chu

\*

\* Created on May 1, 2014, 12:57 AM

\*/

#include "LCD.h"

#include "stdlib.h"

#include "stdio.h"

#include "Timer.h"

#include "UART.h"

// CONFIG1H

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

#pragma config FCMEN = ON // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor enabled)

#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled)

// CONFIG2L

#pragma config PWRT = ON // Power-up Timer Enable bit (PWRT enabled)

#pragma config BOREN = SBORDIS // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware only (SBOREN is disabled))

#pragma config BORV = 18 // Brown Out Reset Voltage bits (VBOR set to 1.8 V nominal)

// CONFIG2H

#pragma config WDTEN = OFF // Watchdog Timer Enable bit (WDT is controlled by SWDTEN bit of the WDTCON register)

#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H

#pragma config CCP2MX = PORTC // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)

#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)

#pragma config LPT1OSC = OFF // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation)

#pragma config HFOFST = OFF // HFINTOSC Fast Start-up (The system clock is held off until the HFINTOSC is stable.)

#pragma config MCLRE = ON // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)

// CONFIG4L

#pragma config STVREN = ON // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)

#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

// CONFIG5L

#pragma config CP0 = OFF // Code Protection Block 0 (Block 0 (000800-003FFFh) not code-protected)

#pragma config CP1 = OFF // Code Protection Block 1 (Block 1 (004000-007FFFh) not code-protected)

#pragma config CP2 = OFF // Code Protection Block 2 (Block 2 (008000-00BFFFh) not code-protected)

#pragma config CP3 = OFF // Code Protection Block 3 (Block 3 (00C000-00FFFFh) not code-protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected)

#pragma config CPD = OFF // Data EEPROM Code Protection bit (Data EEPROM not code-protected)

// CONFIG6L

#pragma config WRT0 = OFF // Write Protection Block 0 (Block 0 (000800-003FFFh) not write-protected)

#pragma config WRT1 = OFF // Write Protection Block 1 (Block 1 (004000-007FFFh) not write-protected)

#pragma config WRT2 = OFF // Write Protection Block 2 (Block 2 (008000-00BFFFh) not write-protected)

#pragma config WRT3 = OFF // Write Protection Block 3 (Block 3 (00C000h-00FFFFh) not write-protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected)

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot Block (000000-0007FFh) not write-protected)

#pragma config WRTD = OFF // Data EEPROM Write Protection bit (Data EEPROM not write-protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection Block 0 (Block 0 (000800-003FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR1 = OFF // Table Read Protection Block 1 (Block 1 (004000-007FFFh) not protected from table reads executed in other blocks)

#pragma config EBTR2 = OFF // Table Read Protection Block 2 (Block 2 (008000-00BFFFh) not protected from table reads executed in other blocks)

#pragma config EBTR3 = OFF // Table Read Protection Block 3 (Block 3 (00C000-00FFFFh) not protected from table reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot Block (000000-0007FFh) not protected from table reads executed in other blocks)

#define GREEN LATAbits.LATA4

#define BLUE LATCbits.LATC3

#define RED LATCbits.LATC4

#define USE\_ADC\_H

void init\_ADC(void);

void ADC\_isr(void);

char title[] = "BLT - Cam bien va ky thuat do luong - Phat hien do mau";

char temp[] = "Phat Hien Do Mau";

char reds[] = "Mau do ";

char warning[] = "Phat hien do mau ";

char error[] = "Other ";

char str1[] = "Do mau";

unsigned int result;

char temperature;

unsigned char a,b;

unsigned char flag = 0;

unsigned int red, green, blue;

int count = 0;

void main(void) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Init PORT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

INTCON2bits.RBPU = 0;//PB0-->PB7 have weak pull-up to Vcc

WPUB = 0x0f;

TRISCbits.TRISC4 = 0;

TRISCbits.TRISC3 = 0;

TRISAbits.TRISA4 = 0;

RED =0;

GREEN =0;

BLUE =0 ;

timer0\_config();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Init LCD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Init\_PORTS();

Init\_LCD();

Delay10TCYx(10);

lcd\_clear();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Init ADC\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

init\_ADC();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*UART\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Delay10TCYx(10);

lcd\_gotoxy(1,1);

lcd\_putstr(temp);

ADCON0bits.GO = 1;//Start conversion by setting the GO/~DONE bit

while(1)

{

if(flag)

{

count++;

if((count >0) && (count<=4))

{

RED =1;

GREEN =0;

BLUE =0 ;

red = ADRESH;

red = red << 8;

red = red | ADRESL;

ADCON0bits.GO = 1;

}

if((count >4) && (count<=8))

{

RED =0;

GREEN =1;

BLUE =0 ;

green = ADRESH;

green = green << 8;

green = green | ADRESL;

ADCON0bits.GO = 1;

}

if((count >8) && (count<=12))

{

RED =0;

GREEN =0;

BLUE =1;

blue = ADRESH;

blue = blue << 8;

blue = blue | ADRESL;

ADCON0bits.GO = 1;

}

if(count > 12) count =0;

if(red <green && red <blue)

{

lcd\_gotoxy(1,2);

lcd\_putstr(error);

}

else

{

lcd\_gotoxy(1,2);

lcd\_putstr(str1);

}

flag = 0;

}

};

}

void init\_ADC(void)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Configure port\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

TRISBbits.TRISB2 = 1;

ANSEL = 0x00; // Set PORT RA0 is analog input

ANSELH = 0x01; // Set PORT AN8 to AN12 as Digital I/O

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Configure ADC module\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Select ADC conversion clock

//Select result format

//Select acquisition delay

ADCON2 = 0x88; //ADCS<2:0> = 001 => Conversion clock = Fosc/8

//Right justified

//ACQT<2:0> = 010 => Acquisition Time = 4 Tad

//Configure voltage reference

ADCON1 = 0x00; //negative voltage reference supplied by Vss

//positive voltage reference supplied by VREF+ = 3.3V

//Select ADC input channel

ADCON0 = 0x20; //CHS<3:0> = 0000 => select channel AN8

//Turn on ADC module

ADCON0bits.ADON = 1;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Configure ADC interrupt (optional)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Clear ADC interrupt flag

PIR1bits.ADIF = 0;

//Enable ADC interrupt

PIE1bits.ADIE = 1;

//Enable peripheral interrupt

INTCONbits.PEIE = 1;

//Enable global interrupt

INTCONbits.GIE = 1;

}

#pragma code interrupt\_vector=0x08

void interrupt\_vector()

{

\_asm

goto ADC\_isr

\_endasm

}

#pragma interrupt ADC\_isr

void ADC\_isr(void)

{

if(PIR1bits.ADIF == 1)

{

PIR1bits.ADIF = 0;

}

if( INTCONbits.TMR0IF ==1)

{

flag = 1;

TMR0H=0xF0;

TMR0L=0x00;

INTCONbits.TMR0IF=0;

}

}

**Mục lục Trang**

1. **TỔNG QUAN ĐỀ TÀI 1**
2. **PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 2**
3. **Ý tưởng thiết kế 3**
4. **Công cụ lập trình 3**
5. **Cơ sở lý thuyết 3**

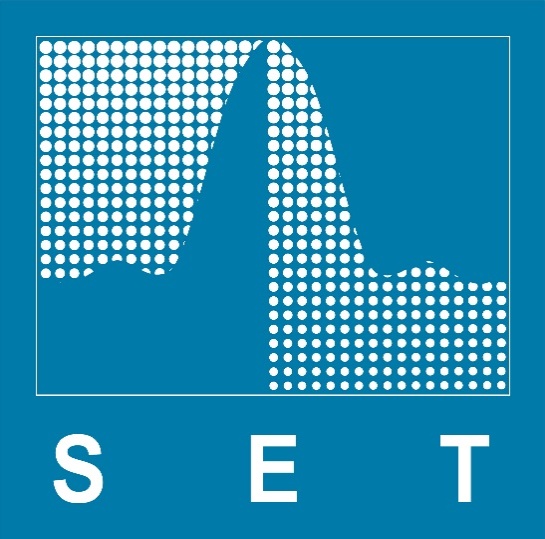
**3.1 Cơ sở lý thuyết 3**

**3.2 Thuật toán 4**

1. **Phân tích thiết kế 5**
   1. **Khối tín hiêu 5**
   2. **Khối Vi xử lý 6**
   3. **Khối Bluetooth 8**
   4. **Khối hiển thị 10**
2. **TỔNG QUAN VỀ VI ĐIỀU KHIỂN PIC**
3. **PIC là gì 11**
4. **Các đặc tính 12**
5. **Cấu hình port 16**
6. **Cấu hình Timer 17**
7. **LINH KIÊN VÀ THÔNG SỐ KỸ THUẬT 18**
8. **LED 19**
9. **Quang trở 20**
10. **HC-05 20**
11. **PIC 21**
12. **PHẦN MỀM 22 10**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  
Viện Điện tử - Viễn thông**

===== ✡✡✡=====



**BÁO CÁO BÀI TẬP LỚN**

**CẢM BIẾN VÀ KỸ THUẬT ĐO LƯỜNG**

*Đề tài: Thiết kế mạch phát hiện dò máu trong truyền dịch/ dẫn dịch.*

***Giáo viên hướng dẫn : Ths.Nguyễn Việt Dũng***

***Sinh viên thực hiện:***

|  |  |  |
| --- | --- | --- |
| Chu Trọng Khanh | Điện tử 5 – K55 | 20101700 |
| Vũ Quang Hiệu | Điện tử 8 – K53 | 20081002 |

Hà Nội, 06/2014