LỜI MỞ ĐẦU

Hiện nay, ngành kỹ thuật điện tử và công nghệ thông tin tiến bộ không ngừng. Chúng đang ngày càng phát triển và được ứng dụng trong tất cả các mặt của đời sống. Các thiết bị điện tử dùng Vi Điều Khiển được sử dụng rộng rãi khắp trong các ứng dụng tự động. Nó giúp chúng ta trong mọi công việc cũng như giải trí. Các bộ Vi Điều Khiển ngày càng hiện đại, tốc độ xử lý nhanh hơn, và các ứng dụng rộng hơn.

Một trong những ứng dụng quan trọng của Vi Điển Khiển đó là dựng trong đo lường và điều khiển. Nhờ các loại cảm biến, ứng dụng của đo lường bằng Vi Điều Khiển không chỉ giới hạn trong các đại lượng điện mà cũng mở rộng ra các tín hiệu không phải điện. Sử dụng Vi Điều Khiển chúng ta thu thập các đại lượng cần đo dễ dàng hơn, cụ thể xử lý ngay các đại lượng đó và đưa ra được những kết quả như mong muốn.

Với tầm quan trọng của đo lường bằng Vi Điều Khiển nên em đã nhận đề tài này làm đồ án tốt nghiệp để nghiên cứu và hiểu biết thêm về Vi Điều Khiển và các ứng dụng hay của nó trong cuộc sống thường ngày của chúng ta.

Trong quá trình làm đồ án tốt nghiệp, do sự hạn chế về thời gian, tài liệu và trình độ có hạn nên không tránh khỏi có thiếu sót. Em rất mong được sự đúng góp ý kiến của thầy cô và các bạn để đồ án tốt nghiệp của em được hoàn thiện hơn.

Em xin gửi lời cảm ơn chân thành đến các thầy cô trong Điện tự động công nghiệp, đặc biệt là thầy Nguyễn Trọng Thắng đã giúp đỡ em hoàn thành tốt đồ án này.

CHƯƠNG 1.

TỔNG QUAN VỀ VI ĐIỀU KHIỂN

1.1. TỔNG QUAN VỀ HỌ IC8051

Có 4 bộ vi điều khiển 8 bit chính. Đó là 6811 của Motorola, 8051 của Intel, z8 của Xilog và Pic 16 của Microchip Technology. Mỗi một kiểu loại trên đây đều có một tập lệnh và thanh ghi riêng duy nhất, nếu chúng đều không tương thích lẫn nhau. Cũng có những bộ vi điều khiển 16 bit và 32 bit được sản xuất bởi các hãng sản xuất chíp khác nhau. Với tất cả những bộ vi điều khiển khác nhau thì tiêu chuẩn để lựa chọn các bộ vi điều khiển là:

*) Đáp ứng được nhu cầu tính toán của bài toán một cách hiệu quả về mặt giá thành và đầy đủ chức năng có thể nhìn thấy được. Trong khi phân tích các nhu cầu của một dự án dựa trên bộ vi điều khiển chúng ta phải biết bộ vi điều khiển nào là 8 bit, 16 bit hay 32 bit có thể đáp ứng tôt nhất nhu cầu của bài toán một cách hiệu quả. Nhứng tiêu chuẩn đó là:

- Tốc độ: tốc độ lớn nhất mà vi điều khiển hỗ trợ lạ bao nhiêu.

- Kiểu đóng vỏ: Đóng vỏ kiểu DIP 40 chân hay QFP. Đây là yêu cầu quan trọng đối với yêu cầu về không gian, kiểu láp ráp và tạo mẫu thử cho sản phẩm cuối cùng.

- Công suất tiêu thụ: Điều này đặc biệt khắt khe đối với các sản phẩm dùng pin, ắc quy.

- Dung lượng bộ nhớ Rom và Ram trên chíp.

- Số chân vào ra và bộ định thời trên chíp.

- Khả năng dễ dàng nâng cấp cho hiệu suất cao hoặc giảm công suất tiêu thụ.

- Giá thành cho một đơn vị: Điều này quan trọng quyết định giá thành sản phẩm mà một bộ vi điều khiển được sử dụng.

*) Coppy sẵn các công cụ phát triển phần mềm như các trình biên dịch, trình hợp ngữ và gỡ rối.

*) Nguồn các bộ vi điều khiển có sẵn nhiều và tin cậy. Khả năng sẵn sàng đáp ứng về số lượng trong hiện tại tương lai. Hiện nay các bộ vi điều khiển 8 bit họ 8051 là có số lượng lớn nhất các nhà cung cấp đa dạng như Intel, Atmel, Philip…

1.1.1. Bộ vi điều khiển 8051

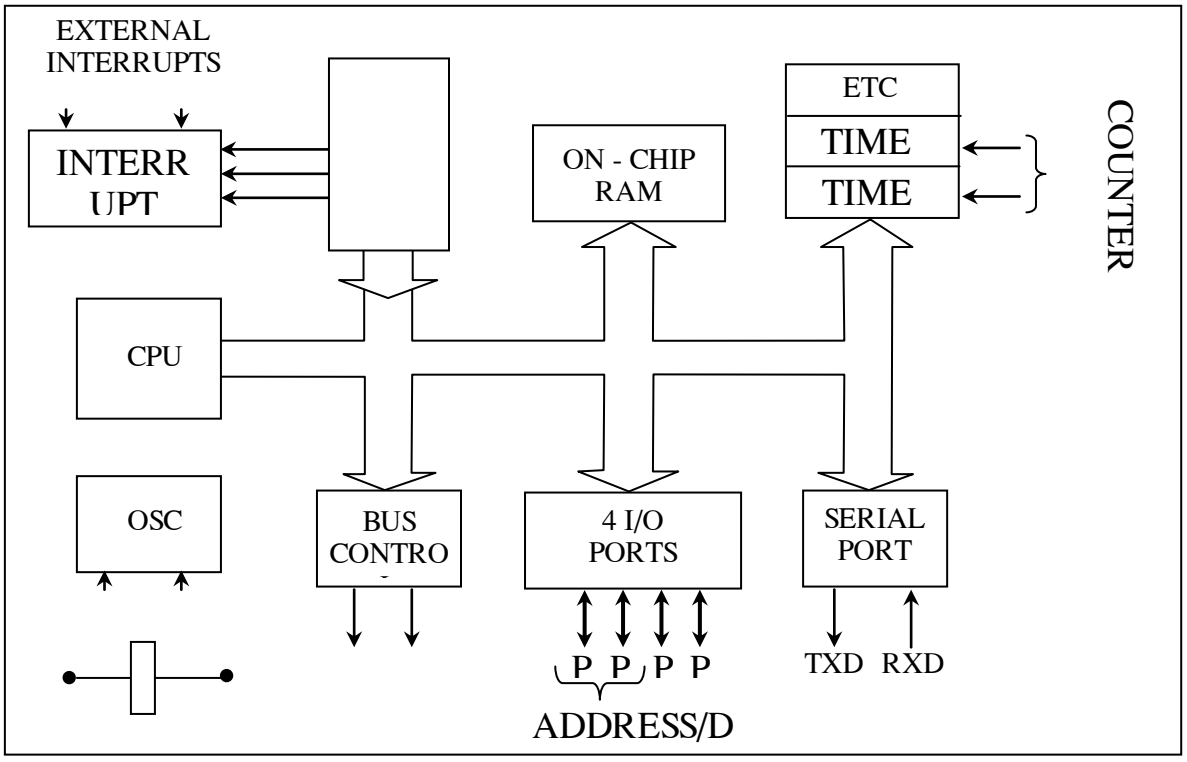
Vào năm 1981 hãng Intel giới thiệu một số bộ vi điều khiển được gọi là 8051. Bộ vi điều khiển này có 128 byte RAM, 4K byte ROM trên chíp, hai bộ định thời, một cổng nối tiếp và 4 cổng (đều rộng 8 bit) vào ra tất cả được đặt trên một chíp. Lúc ấy nó được coi là một 'hệ thống trên chíp'. 8051 là một bộ xử lý 8 bit có nghĩa là CPU chỉ có thẻ làm việc với 8 bit dữ liệu tại một thời điểm. Dữ liệu lớn hơn 8 bit được chia ra thành các dữ liệu 8 bit để cho xử lý. 8051 có tất cả 4 cổng vào ra I/O mỗi cổng rộng 8 bit (hình vẽ). Mặc dù 8051 có một ROM trên chíp cực đại là 64Kbyte, nhưng các nhà sản xuất lúc đó đã xuất xưởng chỉ với 4Kbyte Rom trên chíp.

8051 đã trở nên phổ biến sau khi Intel cho phép các nhà sản xuất khác nhau sản xuất và bán bất kỳ dạng biến thể nào của 8051 mà họ thích với điều kiện họ phải để lại mã tương thích với 8051. Điều này dẫn đến sự ra đời nhiều phiên bản của 8051 với các tốc độ khác nhau và dung lượng Rom trên chíp khác nhau. Điều này quan trọng là mặc dù có nhiều biến thể khác nhau của 8051 về tốc độ và dung lượng nhớ ROM trên chíp nhưng tất cả chúng đều tương thích với 8051 ban đầu về các lệnh. Điều này có nghĩa là nếu ta viết chương trình cho một phiên bản nào đó thì nó cũng sẽ chạy với mọi phiên bản bất kỳ khác mà không phân biệt nó từ hãng sản xuất nào.

Bảng 1.1: Các đặc tính của 8051 đầu tiên.

Đặc tính	Số lượng
ROM trên chíp	4Kbyte
RAM	128 byte
Bộ định thời	2
Các chân vào ra	32
Cổng nối tiếp	1
Nguồn ngắt	6

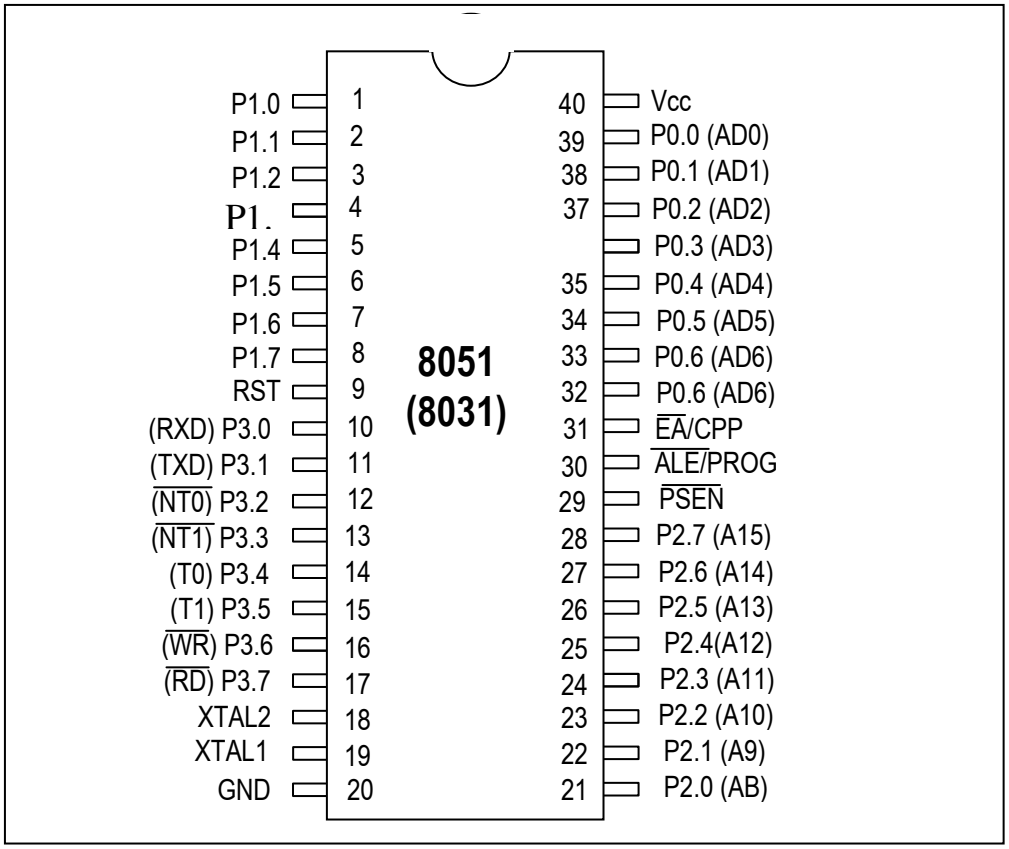
Bộ vi điều khiển 8051 là thành viên đầu tiên của họ 8051, hãng Intel ký hiệu nó là MSC51. Bảng trên là các đặc tính của họ 8051.



Hình 1.1: Bố trí bên trong của 8051

Mô tả chân của 8051 như ***Hình 1.2***. Các thành viên của họ 8051 (ví dụ 8751, 89C51, DS5000) đều có các kiểu đóng vỏ khác nhau, chẳng hạn như hai hàng chân DIP dạng vỏ dẹp vuông QFP và dạng chip không có chân đỡ LLC thì

chúng đều có 40 chân cho các chức năng khác nhau như vào ra I/O, đọc $\overline{RW}$, ghi $\overline{WR}$, địa chỉ, dữ liệu và ngắt. Cần lưu ý rằng một số hãng cung cấp phiên bản 8051 có 20 chân với số cổng vào ra ít hơn cho các ứng dụng yêu cầu thấp hơn. Tuy nhiên, vì hầu hết các nhà phát triển chính sử dụng chíp đóng vỏ 40 chân với hai hàng chân DIP nên ta chỉ tập chung mô tả phiên bản này.



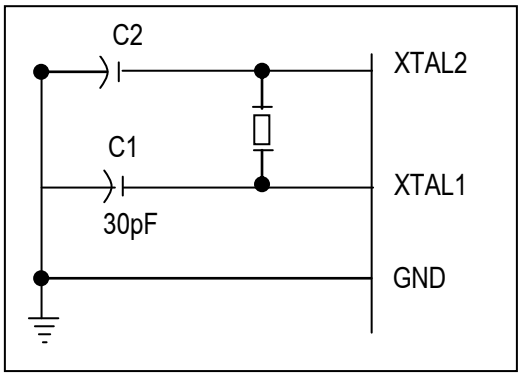
Hình 1.2: Sơ đồ chân của 8051.

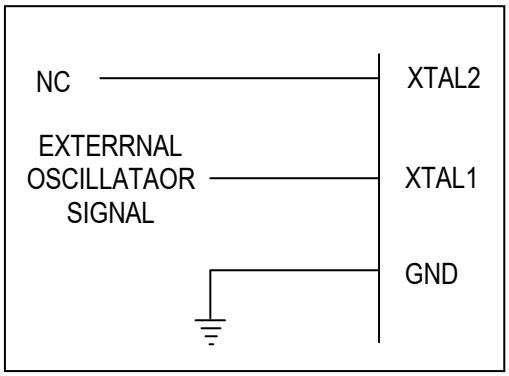
Từ ***Hình 1.2*** ta thấy trong 40 chân có 32 chân dùng cho các cổng P0, P1, P2, P3 với mỗi cổng có 8 chân. Các chân còn lại dành cho nguồn Vcc, đất GND, các chân dao động XTAL1 và XTAL2, khởi động lại RST cho phép chốt địa chỉ ngoài $\overline{EA}$, cho ngắt cất chương trình $\overline{PSEN}$. Trong 8 chân này thì 6 chân Vcc, GND, XTAL1, XTAL2, RST và $\overline{EA}$ được các họ 8031 và 8051 sử dụng. Hay nói cách khác là chúng phải được nối để cho hệ thống làm việc mà không cần biết bộ vi điều khiển thuộc họ 8051 hay 8031. Còn chân $\overline{PSEN}$ và chân ALE được sử dụng trong các hệ thống dựa trên 8031.

- ✓ Chân Vcc và chân GND tương ứng với chân số 40 và chân số 20 cung cấp nguồn (+5V) và nối mass.

✓ Chân XTAL1 (chân 19) và XTAL2 (chân 18): 8051 có bộ dao động trên chíp nhưng nó yêu cầu có một xung đồng hồ ngoài để chạy nó. Bộ dao động thạch anh được nối với XTAL1 và XTAL2 cùng hai tụ điện có giá trị 30pF. Một phía tụ được nối xuống đất như ***Hình 1.3***.

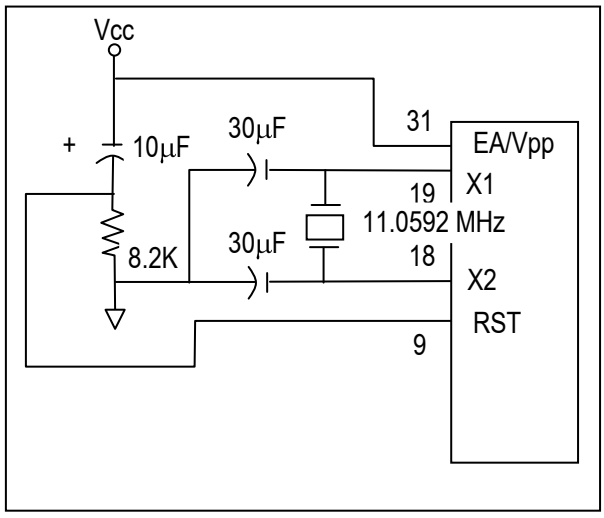
Cần phải lưu ý rằng có nhiều tốc độ khác nhau của họ 8051. Tốc độ được coi như là tần số cực đại của bộ giao động được nối tới chân XTAL. Ta có thể sử dụng một nguồn tần số khác dao động thạch anh chẳng hạn như bộ dao động TTL thì nó sẽ được nối tới chân XTAL1 còn chân XTAL2 để hở như ***Hình 1.4***.

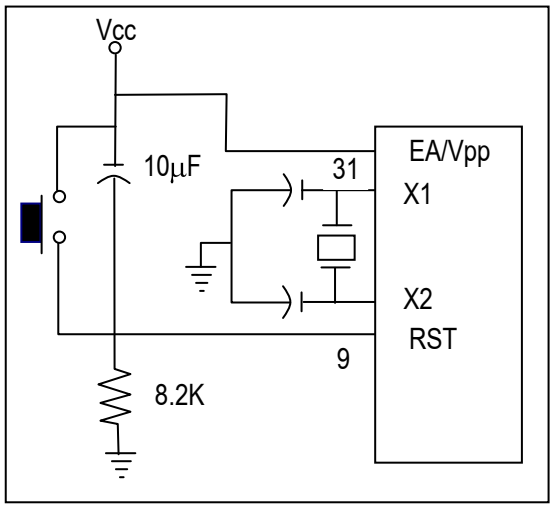




Hình 1.3: XTAL nối với 8051 ***Hình 1.4:*** XTAL nối với dao động ngoài

✓ Chân RST: Chân số 9 là chân tái lập RESET. Nó là chân đầu vào có mức tích cực cao. Khi cấp xung cao tới chân này thì bộ vi điều khiển sẽ tái lập và kết thúc mọi hoạt động. Nó có thể coi như sự tái bật nguồn.





Hình 1.5: Mạch tái bật nguồn RESET. ***Hình 1.6:*** Mạch tái bật nguồn với Debounce.

Muốn mạch RESET làm việc có hiệu quả thì nó phải có tối thiểu 2 chu kì máy. Hay nói cách khác, xung cao phải kéo dài tối thiểu 2 chu kì máy trước khi nó xuống thấp.

- ✓ Chân $\overline{EA}$ (là chân IN): Truy cập bộ nhớ ngoài, chân số 31 trên vỏ chíp như 8751, 89C51 hoặc DS5000 thì chân $\overline{EA}$ được nối với nguồn Vcc. Trường hợp không có ROM trên chíp như 8031 và 8051 thì mã chương trình được lưu cất ở bộ nhớ ngoài, khi đó chân $\overline{EA}$ được nối đất. Như vậy chân này không bao giờ được để hở.
- ✓ Chân $\overline{\text{PSEN}}$ là chân có chức năng cho phép lưu chương trình. Ở hệ thống 8031, khi chương trình cất ở bộ nhớ ROM ngoài thì chân này được nối tới chân OE của ROM.
- ✓ ALE cho phép chốt địa chỉ là chân có mức tích cực cao. Khi nối 8031 tới bộ nhớ ngoài thì cổng 0 cũng được cấp địa chỉ và dữ liệu. Hay nói cách khác, 8031 dồn địa chỉ và dữ liệu qua cổng 0 để tiết kiệm số chân. Chân ALE được sử dụng để phân kênh địa chỉ và dữ liệu bằng cách nối tới chân G của của chíp 73LS373.
- ✓ Nhóm chân cổng vào ra I/O: bốn cổng P0, P1, P2, P3 đều có 8 chân và tạo thành cổng 8 bít. Tất cả các cổng khi RESET đều được cấu hình làm cổng ra. Để làm đầu vào thì cần được lập trình.

Các cổng bình thường là cổng ra. Cổng P0 có thể vừa làm đầu ra, vừa làm đầu vào cổng P0 từ chân 32 đến 39 phải được nối với điện trở kéo 10K bên ngoài. Cổng P1 cũng có 8 chân, từ chân 1 đến chân 8, và có thể sử dụng làm đầu vào hoặc ra. Khác với cổng P0, cổng P1 không cần đến điện trở kéo bên ngoài vì nó đã có điện trở kéo bên trong. Cổng P2 cũng có 8 chân từ chân 21đến 28, và có thể sử dụng làm đầu vào hoặc ra. Cũng giống như cổng P1, cổng P2 không cần điện trở kéo vì bên trong đã có các điện trở kéo. Cổng P3 có 8 chân từ chân 10 đến chân 17. Cổng này có thể sử dụng làm đầu vào hoặc ra. Cũng như chân P1và P2, cổng P3 cũng không cần điện trở kéo.

Bảng 1.2: Chức năng các chân cổng P3**.**

Bít cổng P3	Chức năng	Chân số
P3.0	Nhận dữ liệu (RXD)	10
P3.1	Phát dữ liệu (TXD)	11
P3.2	Ngắt 0(INT0)	12
P3.3	Ngắt 1(INT1)	13
P3.4	Bộ định thời 0 (TO)	14
P3.5	Bộ định thời 1(T1)	15
P3.6	Ghi (WR)	16
P3.7	Đọc (RD)	17

Có hai bộ vi điều khiển thành viên khác của họ 8051 là 8052 và 8031.

1.1.2. Bộ vi điều khiển 8052

Bộ vi điều khiển 8052 là thành viên khác của họ 8051, 8052 có tất cả các đặc tính chuẩn của 8051 ngoài ra nó có thêm 128 byte RAM và một bộ định thời nữa. Hay nói cách khác là 8052 có 256 byte RAM và 3 bộ định thời, nó cũng có 8K byte ROM trên chíp thay vì 4K byte như 8051.

Bảng 1.3: So sánh các đặc tính của các thành viên họ 8051.

Đặc tính	8051	8052	8031
ROM trên chíp	4K byte	8K byte	OK
RAM	128 byte	256 byte	128 byte
Bộ định thời	2	3	2
Chân vào - ra	32	32	32
Cổng nối tiếp	1	1	1
Nguồn ngắt	6	8	6

Qua bảng trên ta thấy thì 8051 là tập con của 8052, nên mọi chương trình viết cho 8051 đều chạy được trên 8052 nhưng điều ngược lại là không đúng.

1.1.3. Bộ vi điều khiển 8031

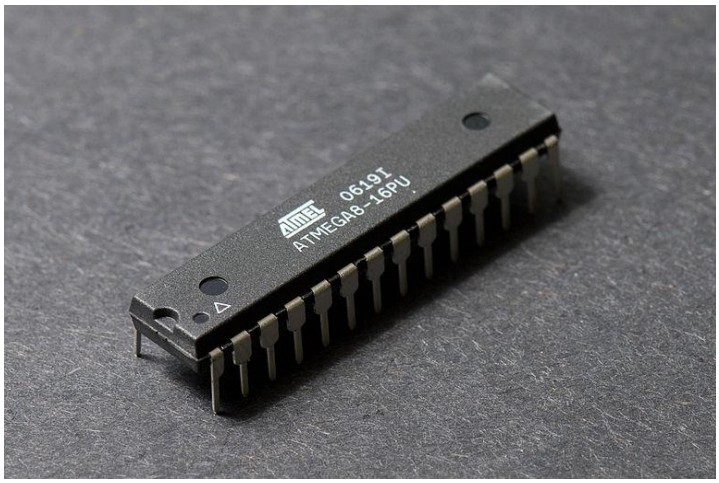
Một thành viên khác của 8051 là chíp 8031. Chíp này không có ROM trên chíp nên để sử dụng chíp này ta phải bổ sung ROM ngoài cho nó, ROM ngoài phải chứa chương trình mà 8031 sẽ nạp và thực hiện. So với 8051 mà chương trình được chứa trong ROM trên chíp bị giới hạn bởi 4K byte, còn ROM ngoài chứa chương trình được gắn vào 8031 thì có thể lớn đến 64K byte. Khi bổ xung cổng, như vậy chỉ còn lại hai cổng để thao tác. Để giải quyết vấn đề này ta có thể bổ xung cổng vào ra cho 8031 bằng cách phối ghép 8031 với bộ nhớ và cổng vào ra chẳng hạn với chíp 8255. Ngoài ra còn có các phiên bản khác nhau về tốc độ của 8031 từ các hãng sản xuất khác nhau.

Bảng 1.4: Các phiên bản của 8051 từ Atmel.

Số linh kiện	ROM	RAM	Chân I/O	Timer	Ngắt	Vcc	Đóng vỏ
AT89C51	4K	128	32	2	6	5V	40
AT89LV51	4K	128	32	2	6	3V	40
AT89C1051	1K	64	15	1	3	3V	20
AT89C2051	2K	128	15	2	6	3V	20
AT89C52	8K	128	32	3	8	5V	40
AT89LV52	8K	128	32	3	8	3V	40

1.2. CÁC HỆ VI ĐIỀU KHIỂN TIÊN TIẾN

1.2.1. Atmel AVR



Hình 1.7: Atmel AVR ATmega8 PDIP

AVR là một kiến trúc Harvard sửa đổi 8-bit RISC đơn chip vi điều khiển (μC) đã được phát triển bởi Atmel vào năm 1996. Các AVR là một trong những họ vi điều khiển đầu tiên sử dụng on-chip bộ nhớ flash để lưu trữ chương trình, trái với One-Time Programmable ROM, EPROM hoặc EEPROM được sử dụng bởi vi điều khiển khác vào lúc đó.

1.2.1.1. Lịch sử họ AVR

Người ta tin vào kiến trúc AVR cơ bản đã được hình thành bởi hai sinh viên tại Viện Công nghệ Na Uy (thứ n) Alf-Egil Bogen và Vegard Wollan.

Các AVR MCU bản gốc đã được phát triển tại một ngôi nhà ASIC thuộc địa phương ở Trondheim, Na Uy, nơi mà hai thành viên sáng lập của Atmel Na Uy đã làm việc như sinh viên. Nó được biết đến như một μRISC (Micro RISC). Khi công nghệ đã được bán cho Atmel, kiến trúc nội bộ đã được phát triển thêm bởi Alf và Vegard tại Atmel Na Uy, một công ty con của Atmel thành lập bởi hai kiến trúc sư. Atmel AVR nói rằng các tên không phải là một từ viết tắt và không phải là bất cứ điều gì đặc biệt. Những người sáng tạo AVR không có câu trả lời dứt khoát về thuật ngữ viết tắt "AVR".

Lưu ý rằng việc sử dụng "AVR" trong bài viết này thường đề cập đến 8-bit RISC dòng vi điều khiển Atmel AVR.

Trong số những thành viên đầu tiên của dòng AVR là AT90S8515, đóng vỏ trong gói 40-pin DIP có chân ra giống như một vi điều khiển 8051, bao gồm địa chỉ BUS multiplexed bên ngoài và dữ liệu. Tín hiệu RESET đã đổi ngược, 8051 RESET mức cao, AVR RESET mức thấp), nhưng khác với đó, chân ra là giống hệtnhau.

1.2.1.2. Tổng quan về thiết bị

AVR là một kiến trúc máy Modified Harvard với chương trình và dữ liệu được lưu trữ trong các hệ thống bộ nhớ vật lý riêng biệt xuất hiện trong không gian địa chỉ khác nhau, nhưng có khả năng đọc ghi dữ liệu từ bộ nhớ bằng cách sử dụng lệnh đặc biệt.

Cơ bản về họ AVR thường chi thành bốn nhóm rộng.

• TinyAVR - chuỗi Attiny

_ 0,5-8 kBbộ nhớ chương trình

_ Đóng vỏ 6-32-chân

_ Tập ngoại vi hữu hạn

• MegaAVR - chuỗi Atmega

_ 4-256 Kb bộ nhớ chương trình

_ Đóng vỏ 28-100-chân

_ Tập lệnh mở rộng (Lệnh nhân và lệnh cho quản lý bộ nhớ lớn hơn).

_ Mở rộng hơn về thiết bị ngoại vi

• XMEGA - chuỗi Atxmega

_ 16-384 kB bộ nhớ chương trình.

_ Đóng vỏ 44-64-100-chân (A4, A3, A1)

_ Mở rộng các tính năng hiệu suất, chẳng hạn như DMA, "Sự kiện hệ thống", và hỗ trợ mật mã.

_ Thiết bị ngoại vi được mở rộng với DACs

• Ứng dụng cụ thể AVR

_ megaAVRs với các tính năng đặc biệt không tìm thấy trên các thành viên khác của gia đình AVR, chẳng hạn như bộ điều khiển LCD, USB, điều khiển, nâng cao PWM, CAN v.v..

_ Atmel At94k FPSLIC (Field Programmable System Level Circuit), một lõi trên AVR với một FPGA. FPSLIC sử dụng SRAM cho mã chương trình AVR, không giống như tất cả các AVRs khác. Một phần do sự khác biệt tốc độ tương đối giữa SRAM

1.2.1.3. Kiến trúc thiết bị

Flash, EEPROM, và SRAM tất cả được tích hợp vào một chip duy nhất, loại bỏ sự cần thiết của bộ nhớ ngoài trong hầu hết các ứng dụng. Một số thiết bị có BUS mở rộng song song để cho phép thêm dữ liệu bổ sung (hoặc mã) bộ nhớ, hoặc bộ nhớ ánh xạ thiết bị. Tất cả các thiết bị có giao tiếp nối tiếp, mà có thể được sử dụng để kết nối EEPROMs nối tiếp chip flash.

1.2.1.4. Program Memory (Flash)

Mã lệnh chương trình được lưu trữ trong bộ nhớ Flash chống xóa (non-volatile Flash). Mặc dù họ là 8-bit MCUs, mỗi lệnh mất 1 hoặc 2 từ 16-bit. Kích cỡ của bộ nhớ chương trình thường được chỉ định trong việc đặt tên của thiết bị chính (ví dụ, dòng ATmega64x có 64 kB của Flash, tuy nhiên ATmega32x chỉ có 32kB).

1.2.1.5. EEPROM

Hầu như tất cả các vi điều khiển AVR đều có Electrically Erasable Programmable Read Only Memory (EEPROM) để lưu "nửa vĩnh viễn" dữ liệu trữ. Cũng giống như bộ nhớ Flash, EEPROM có thể duy trì nội dung của nó khi được gỡ bỏ. Trong hầu hết các biến thể của kiến trúc AVR, bộ nhớ EEPROM nội bộ này không phải là ánh xạ vào không gian địa chỉ bộ nhớ của MCU. Nó chỉ có thể được truy cập cùng một cách như là thiết bị ngoại vi bên ngoài, thanh ghi sử dụng con trỏ đặc biệt và đọc / ghi hướng dẫn mà làm cho truy cập EEPROM chậm hơn nhiều so với RAM nội bộ khác. Tuy nhiên, một số thiết bị trong dòng SecureAVR (AT90SC) sử dụng một bản đồ EEPROM đặc biệt đến các dữ liệu hoặc bộ nhớ chương trình tùy thuộc vào cấu hình. Dòng XMEGA

cũng cho phép EEPROM ánh xạ vào không gian địa chỉ dữ liệu. Kể từ khi số lượng các lần ghi EEPROM không phải là không giới hạn – Atmel chỉ được 100.000 chu kỳ ghi.

1.2.1.6. Chương trình thực thi

Atmel's AVRs có hai giai đoạn, thiết kế kiểu đường ống (pipeline) duy nhất. Điều này có nghĩa là chỉ lệnh kế tiếp là được lấy khi lệnh này đang thực hiện. Hầu hết các lệnh chỉ mất một hoặc hai chu kỳ đồng hồ, làm cho AVRs tương đối nhanh trong số vi điều khiển 8-bit. Họ AVR của bộ vi xử lý được thiết kế với sự thực hiện hiệu quả của mã C.

1.2.1.7. Tập lệnh

Tập lệnh AVR hơn là trực giao với hầu hết các vi điều khiển tám-bit, đặc biệt là

8051 và vi điều khiển PIC với AVR mà ngày nay đang cạnh tranh. Tuy nhiên, nó không phải là hoàn toàn bình thường:

• Con trỏ ghi X, Y, và Z có khả năng đánh địa chỉ khác với nhau.

• Vị trí thanh ghi R0 đến R15 có khả năng đánh địa chỉ khác hơn vị trí thanh ghi R16 đến R31.

• I / O port 0-31 có khả năng đánh địa chỉ khác so với I / O ports 32-63.

• CLR ảnh hưởng đến các cờ, trong khi SER không, ngay cả khi chúng được lệnh bổ sung. CLR xóa tất cả các bit về không và SER đặt chúng lên một.

• Truy cập dữ liệu chỉ đọc được lưu trong bộ nhớ chương trình (flash) yêu cầu lệnh đặc biệt LPM.

Ngoài ra, một số chip-sự khác biệt cụ thể ảnh hưởng đến các thế hệ mã. Mã con trỏ (bao gồm cả các địa chỉ trở lại stack) là hai byte trên chip lên đến 128 KBytes bộ nhớ flash, nhưng ba byte trên chip lớn hơn, không phải tất cả các chip có số nhân phần cứng; chip với hơn 8 Kbytes flash có nhánh và gọi lệnh với khoảng rộng hơn...

Lập trình cho nó bằng cách sử dụng lập trình C (hoặc thậm chí Ada) trình biên dịch khá đơn giản. GCC đã bao gồm hỗ trợ AVR từ khá lâu, và hỗ trợ được sử dụng lưu rộng rãi. Trong thực tế, Atmel gạ gẫm đầu vào từ các nhà phát triển

chính của trình biên dịch cho vi điều khiển nhỏ, để tích hợp tính năng cho các tập lệnh hữu dụng nhất trong một trình biên dịch cho các ngôn ngữ cấp cao.

1.2.1.8. Tốc độ MCU

Dòng AVR bình thường có thể hỗ trợ tốc độ đồng hồ 0-20 MHz, với một số thiết bị đạt 32 MHz. Hỗ trợ hoạt động thấp hơn thường đòi hỏi một tốc độ giảm. Tất cả gần đây (Tiny và Mega, nhưng không phải 90S) AVRs tích hợp oscillator-chip, loại bỏ sự cần thiết của đồng hồ bên ngoài hoặc mạch dao động. Một số AVRs cũng có một prescaler đồng hồ hệ thống, có thể chia xuống đồng hồ của hệ thống lên đến 1024. Prescaler này có thể được cấu hình lại bằng phần mềm trong thời gian chạy, cho phép tối ưu hóa tốc độ đồng hồ. Vì tất cả các hoạt động (trừ literals) trên thanh ghi R0 - R31 là đơn chu kỳ, các AVR có thể đạt được lên đến 1MIPS mỗi MHz. Tải và lưu trữ vào / ra bộ nhớ mất 2 chu kỳ, phân nhánh phải mất 3 chu kỳ.

1.2.1.9. Những đặc tính

AVRs hiện cung cấp một loạt các tính năng:

• Máy đa chức năng, Bi-directional General Purpose I / O port với cấu hình, built-in pull-up resistors

• Nhiều nội Oscillators, bao gồm cả RC oscillator mà không có bộ phận bên ngoài

• Nội, lệnh Self-Programmable Flash Memory lên đến 256 KB (384 KB trên XMega)

o In-System Programmable sử dụng nối tiếp / song song hạ thế độc quyền hoặc các giao diện JTAG

o Tùy chọn khởi động với bảo vệ Lock Bits độc lập.

• On-chip gỡ lỗi (OCD) hỗ trợ thông qua JTAG hoặc debugWIRE trên hầu hếtcác thiết bị.

o tín hiệu JTAG (TMS, TDI, TDO, và TCK) là multiplexed ngày GPIOs. Những Pin có thể được cấu hình với chức năng như JTAGhoặc GPIO tùy thuộc vào thiết lập của một vài cầu chì (FUSES), có thể được

lập trình thông qua ISP hoặc HVSP. Theo mặc định, AVRs với JTAG đi kèm với giao diện JTAG bật.

o debugWIRE sử dụng chân /RESET như một kênh giao tiếp hai hướng để truy cập vào mạch debug-chip. Đó là hiện nay trên các thiết bị với số lượng chân ít, vì nó chỉ cần một chân.

• Internal Data EEPROM lên đến 4 kB

• Internal SRAM lên đến 8 kB (32 kB trên XMega)

• Ngoài 64KB dữ liệu trên các mô hình không gian nhất định, bao gồm cả Mega8515 và Mega162.

o Trong một số thành viên của loạt XMEGA, dữ liệu không gian bên ngoài đã được tăng cường để hỗ trợ cả hai SRAM và SDRAM. Đồng thời, các dữ liệu địa chỉ, các chế độ đã được mở rộng cho phép lên đến 16MB bộ nhớ của dữ liệu được đề cập trực tiếp.

o AVR thường không hỗ trợ thực thi mã từ bộ nhớ bên ngoài. Một số ASSP bằng cách sử dụng mã AVR làm bộ nhớ hỗ trợ chương trình bên ngoài.

• 8-Bit và 16-Bit Timers

o PWM đầu ra (thời gian chết máy phát điện trên một số thiết bị)

o Vào capture

• So sánh Analog

o 10 hoặc 12-Bit A / D Converters, với multiplex lên đến 16 kênh

o 12-bit D / A Converters

• Một loạt các giao tiếp nối tiếp, bao gồm cả

o I²C tương thích Two-Wire Interface (TWI)

o Thiết bị ngoại vi Synchronous / Asynchronous Serial (UART / USART) (được sử dụng với RS-232, RS-485, và nhiều hơn nữa)

o Thiết bị giao diện Serial Bus (SPI)

o Universal Serial Interface (USI) cho 2 hoặc 3 dây truyền thông đồng bộ nối tiếp.

• Brownout Detection

• Watchdog Timer (WDT)

• Nhiều chế độ tiết kiệm điện (Power-Saving Sleep)

• Điều khiển ánh sáng và điều khiển động cơ (cụ thể là PWM) điều khiển mô hình

• Hỗ trợ CAN Controller

• Hỗ trợ USB Controller

o USB – Full speed (12 Mbit / s) điều khiển phần cứng & Hub với AVR nhúng.

o Cũng sẵn sàng tự do với tốc độ thấp (1,5 Mbit / s) (HID) bitbanging EMULATIONS phần mềm

• Hỗ trợ Ethernet Controller

• Hỗ trợ LCD Controller

• Hoạt động ở mức điện áp thấp, có thể xuống đến 1.8v (đến 0.7v với loại hỗ trợ chuyển đổi DC-DC)

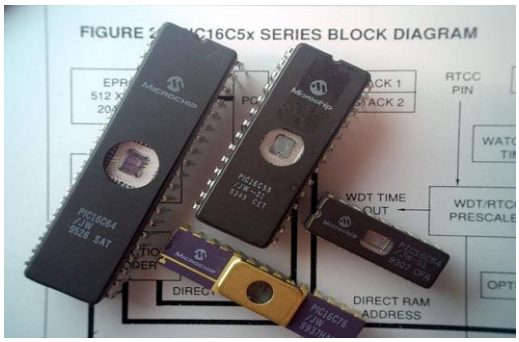
• Thiết bị picoPower

• Bộ điều khiển DMA và truyền thông "Sự kiện hệ thống" ngoại vi.

• Mã hóa và giải mã nhanh, hỗ trợ cho AES và DES

1.2.2. Vi điều khiển PIC

Hình 1.8: PIC 1655A



Hình 1.9: Các dòng PIC khác

PIC là một họ vi điều khiển RISC được sản xuất bởi công ty Microchip Technology. Dòng PIC đầu tiên là PIC1650 được phát triển bởi Microelectronics Division thuộc General Instrument. PIC bắt nguồn là chữ viết

tắt của "Programmable Intelligent Computer" (Máy tính khả trình thông minh) là một sản phẩm của hãng General Instruments đặt cho dòng sản phẩm đầu tiên của họ là PIC1650. Lúc này, PIC1650 được dùng để giao tiếp với các thiết bị ngoại vi cho máy chủ 16bit 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 tốt, nhưng lại kém về các hoạt động xuất nhập, và vì vậy PIC 8-bit được phát triển vào khoảng năm 1975 để hỗ trợ hoạt động xuất nhập cho CP1600. PIC sử dụng microcode đơn giản đặt trong ROM, và mặc dù, cụm từ RISC chưa được sử dụng thời bây giờ, nhưng PIC thực sự là một vi điều khiển với kiến trúc RISC, chạy một lệnh một chu kỳ máy (4 chu kỳ của bộ dao động).

Năm 1985 General Instruments bán bộ phận 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 để tạo thành 1 bộ điều khiển vào ra khả 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.

1.2.2.1. Lập trình cho PIC

PIC sử dụng tập lệnh RISC, với dòng PIC low-end (độ dài mã lệnh 12 bit, ví dụ: PIC12Cxxx) và mid-range (độ dài mã lệnh 14 bit, ví dụ: PIC16Fxxxx), tập lệnh bao gồm khoảng 35 lệnh, và 70 lệnh đối với các dòng PIC high-end (độ dài mã lệnh 16 bit, ví dụ: PIC18Fxxxx). Tập lệnh bao gồm các lệnh tính toán trên các thanh ghi, với các hằng số, hoặc các vị trí bộ nhớ, cũng như có các lệnh điều kiện, lệnh nhảy/gọi hàm, và các lệnh để quay trở về, nó cũng có các tính năng phần cứng khác như ngắt hoặc sleep (chế độ hoạt động tiết kiện điện). Microchip cung cấp môi trường lập trình MPLAB, nó bao gồm phần mềm mô phỏng và trình dịch ASM. Một số công ty khác xây dựng các trình dịch C, Basic, Pascal cho PIC. Microchip cũng bán trình dịch "C18" (cho dòng PIC high-end) và "C30" (cho dsPIC30Fxxx). Họ cũng cung cấp các bản "student edition/demo" dành cho sinh viên hoặc người dùng thử, những version này không có chức năng tối ưu hoá code và có thời hạn sử dụng giới hạn. Những

trình dịch mã nguồn mở cho C, Pascal, JAL, và Forth, cũng được cung cấp bởi PicForth. GPUTILS là một kho mã nguồn mở các công cụ, được cung cấp theo công ước về bản quyền của GNU General Public License. GPUTILS bao gồm các trình dịch, trình liên kết, chạy trên nền Linux, Mac OS X, OS/2 và Microsoft Windows. GPSIM cũng là một trình mô phỏng dành cho vi điều khiển PIC thiết kế ứng với từng module phần cứng, cho phép giả lập các thiết bị đặc biệt được kết nối với PIC, ví dụ như LCD, LED...

1.2.2.2. Một vài đặc tính

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:

• 8/16 bit CPU, xây dựng theo kiến trúc Harvard có sửa đổi

• Flash và ROM có thể tuỳ chọn từ 256 byte đến 256 Kbyte

• Các cổng Xuất/Nhập (I/O ports) (mức logic thường từ 0V đến 5.5V, ứng với logic 0 và logic 1)

• 8/16 Bit Timer

• Công nghệ Nanowatt

• Các chuẩn Giao Tiếp Ngoại Vi Nối Tiếp Đồng bộ/Không đồng bộ USART, AUSART, EUSARTs

• Bộ chuyển đổi ADC Analog-to-digital converters, 10/12 bit

• Bộ so sánh điện áp (Voltage Comparators)

• Các module Capture/Compare/PWM

• LCD

• MSSP Peripheral dùng cho các giao tiếp I^2C, SPI, và I^2S

• Bộ nhớ nội EEPROM - có thể ghi/xoá lên tới 1 triệu lần

• Module Điều khiển động cơ, đọc encoder

• Hỗ trợ giao tiếp USB

• Hỗ trợ điều khiển Ethernet

• Hỗ trợ giao tiếp CAN

• Hỗ trợ giao tiếp LIN

• Hỗ trợ giao tiếp IrDA

• Một số dòng có tích hợp bộ RF (PIC16F639, và rfPIC)

• KEELOQ Mã hoá và giải mã

• DSP những tính năng xử lý tín hiệu số (dsPIC)

1.2.2.3. Họ vi điều khiển PIC 8/16-bit dòng

Vi điều khiển 8-bit

• PIC10, PIC12, PIC14, PIC16, PIC17, PIC18

Vi điều khiển 16-bit:

• PIC24

Bộ điều khiển xử lý tín hiệu số 16-bit (dsPIC):

• dsPIC30

• dsPIC33F

Bộ điều khiển xử lý tín hiệu số 32-bit (PIC32):

• PIC32

CHƯƠNG 2.

NGÔN NGỮ LẬP TRÌNH C CHO VI ĐIỀU KHIỂN

Trong lập trình vi xử lý ngôn ngữ thường dùng là ngôn ngữ lập trình ASM và ngôn ngữ C . Ngôn ngữ lập trình ASM hay lập trình hợp ngữ là ngôn ngữ lập trình trực tiếp cho vi điều khiển (lập trình trực tiếp) còn ngôn ngữ C hay còn gọi là lập trình hướng đối tượng nó gần với ngôn ngữ con người hơn . Điều này có nghĩa là với ASM người lập trình ra lệnh trực tiếp thông qua ngôn ngữ câu lệnh có tính ràng buộc còn ngôn ngữ C sử dụng các cấu trúc điều kiện và vòng lặp theo ý muốn .Nói về ngôn ngữ C thì ưu điểm của ngôn ngữ C là nó dễ hiểu nhưng cấu trúc lại dài và phức tạp so với ngôn ngữ ASM .

Tìm hiểu lập trình C cho 8051.

2.1. GIỚI THIỆU

C là một ngôn ngữ khá mạnh và có rất nhiều người dung. Nhưng với vi xử lý ta chỉ cần biết một vài vấn đề cơ bản sau :

+ Các kiểu toán tử của C .
+ Các kiểu dữ liệu (int , float , double , char , unsigned char , …)
+ Các hàm trong C
+Cấu trúc cơ bản của một chương trình.
+ Cấu trúc điều khiển hay các tập lệnh.

2.2. KIÊN THỨC CƠ BẢN VỀ C

2.2.1. Các kiểu toán tử của C

Toán tử gán (=)

Các toán tử số học (+ , – ,* , / , %)

+ cộng

- trừ

* nhân

/ chia

% lấy phần dư (trong phép chia)

Các toán tử gán phức hợp : (+=, -=, *=, /=, %=, >>=, < a -= 5; tương đương với a = a – 5;

a /= b; tương đương với a = a / b;

a*=2 ; tương đương với a = a*2

……..

Tăng và giảm (++ , —)

a++; a+=1; a=a+1;

a–; a+=1 a=a-1

Tiền tố hay hậu tố (++a ; a++)

B=3;

B=3;A=++B;

// A là 4, B là 4

Hay :B=3;

A=B++;

// A là 3, B là 4

Các toán tử quan hệ (= = , != ,< , > , =)

= = Bằng

!= Khác

> Lớn hơn

< Nhỏ hơn > = Lớn hơn hoặc bằng

< = Nhỏ hơn hoặc bằng

Các toán tử logic (!, &&, ||)

! NOT

&& AND

|| OR

Các toán tử thao tác bit (&, |, ^, ~, <>)

& AND Logical AND

| OR Logical OR

^ XOR Logical exclusive OR

~ NOT Đảo ngược bit

<< SHL Dịch bit sang trái >> SHR Dịch bit sang phải

*Thứ tự ưu tiên

1 () [ ] -> .2

++ — tăng/giảm

~ Đảo ngược bit

! NOT

& * Toán tử con trỏ

+ – Dương hoặc âm

3 * / % Toán tử số học

4 + – Toán tử số học

5 << >> Dịch bit

6 < >= Toán tử quan hệ

7 == != Toán tử quan hệ

8 & ^ | Toán tử thao tác bit

9 && || Toán tử logic

10 ?: Toán tử điều kiện

11 = += -= *= /= %=

>>= < 12 , Dấu phẩy

2.2.2. Các kiểu biến dữ liệu

Char : 1byte (-128 ; 127)

Unsigned char : 1byte (0; 255)

Enum : 2byte (-32,768 ; 32,768)

Short : 2byte (-32,768 ; 32,768)

Unsigned short : 2byte (0 ; 65,535)

Int : 2byte (-32,768 ; +32,767)

Unsigned int : 2byte (0 ; 65,535)

Long : 4byte (- 2,147,483,648 ; +2,147,483,647)

Unsigned long : 4byte (0 ; 4,294,697,295)

…….

Khai báo biến:

Cấu trúc :

Kiểu biến Tên biến

VD :

unsigned char x;

Ta cũng có thể gán luôn giá trị ban đầu cho biến. Nghĩa là thay vì:

unsigned char x;

x=0;

ta viết là : unsigned char x=0;

Hoặc ta cũng có thể khai báo nhiêu biến một lúc:

unsigned char x,y,z;

Ngoài ra dung cho vi điều khiển trình biên dich chuyên dụng còn hỗ trợ các biến sau

Dạng biến Số Bit Số Byte Miền giá trị

Bit 1 0 0 ; 1

sbit 1 0 0 ; 1

sfr 8 1 0 đến 255

sf16 16 & ; ;nbs p; 2 ; ; ; ;0 đến 65,535

Trong đó bit có thể dung như các biến trong C nhưng các biến còn lại thì liên quan đến các thanh ghi hoặc địa chỉ cổng cua 8051(có nghĩa là khi khai bao biến kiểu bit thì không cần định địa chỉ trong RAM các biến khác phải định rõ địa chỉ trong RAM vì nó là các dạng biến đặc biệt gọi là special function registers (SFR)

VD: bit kiemtra;

sfr P1_0=0×90

Các SFR được khai báo trong thư viện

at89x51.h và at89x52.h

2.2.3. Các hàm trong C

Có hai loai hàm trong C :

+Hàm trả lai giá trị:

Kiểu giá trị hàm trả lại Tên hàm(Biến truyền vào hàm)

{

// Các câu lệnh xử lý

}

VD;

unsigned char cong(unsigned char x, unsigned char y)

+ Hàm không trả lại giá trị

void Tên hàm(Biến truyền vào hàm)

{

// các câu lệnh xử lý

}

VD:

void cong(unsigned char x,unsigned char y)

{

//các câu lệnh

}

(*) Hàm có thể có biến truyền vào hoặc không

+ Hàm không có biến truyền vào

unsigned char Tên hàm(void)

{

//câu lệnh

}

+ Hàm có biến truyền vào

void Tên hàm(unsigned char x)

{

//các câu lệnh

}

(**) Số biến truyền vào là tùy ý miễn sao là đủ bộ nhớ , các biến ngăn cách nhau bằng dấu “,”.

VD: void Tên hàm(unsigned char x,unsigned char y,unsigned char z)

(***) Ngoài ra trong Keil C còn co một loại hàm là hàm ngắt:

Cấu trúc:

void Tên hàm(void) interrupt nguồn ngắt using băng thanh ghi

{

}

Hàm ngắt không được phép trả lại giá tri hay truyền tham biến vào hàm

Tên hàm : tùy chọn

Interrupt : từ khóa chỉ hàm ngắt

Nguồn ngắt : từ 0 đến 5 theo bảng vecter ngắt

Ngắt do Cờ Địa chỉ vector Nguồn ngắt

Reset hệ thống RST 0000H -

Ngắt ngoài 0 IE0 0003H 0

Timer 0 TF0 000BH 1

Ngắt ngoài 1 IE1 001 3H 2

Timer 1 TF1 001BH 3

Port nối tiếp RI hoặc TI 0023H 4

Timer 2 TF2 hoặc EXF2 002BH 5

Băng thanh ghi trên RAM chon từ 0 đến 3.

2.2.4. Các câu lệnh cơ bản của C

+ Cấu trúc điều kiện: if , else

Cấu trúc if : if (điều kiện) lệnh (đưa ra điều kiện và tuyên bố thưc hiện)

VD : if (x10)

tăng giá trị của x cho đến khi x > 10

Chức năng của nó là hoàn toàn giống vòng lặp while chỉ trừ có một điều là điều kiện điều khiển vòng lặp được tính toán sau khi lệnh được thực hiện, vì vậy lệnh

sẽ được thực hiện ít nhất một lần ngay cả khi điều kiện không bao giờ được thoả mãn .Như ví dụ trên kể cả x >10 thì nó vẫn tăng giá trị 1 lần trước khi thoát

- Vòng lặp for:

Cấu trúc : for (khởi tạo;điều kiện;tăng giá trị) lệnh

và chức năng chính của nó là lặp lại lệnh chừng nào điều kiện còn mang giá trị đúng, như

trong vòng lặp while. Nhưng thêm vào đó, for cung cấp chỗ dành cho lệnh khởi tạo và lệnh tăng. Vì vậy vòng lặp này được thiết kế đặc biệt lặp lại một hành động với một số lần xác định.

Cách thức hoạt động của nó như sau:

1) Khởi tạo được thực hiện. Nói chung nó đặt một giá khí ban đầu cho biến điều khiển. Lệnh này được thực hiện chỉ một lần.

2) Điều kiện được kiểm tra, nếu nó là đúng vòng lặp tiếp tục còn nếu không vòng lặp kết thúc và lệnh được bỏ qua.

3) Lệnh được thực hiện. Nó có thể là một lệnh đơn hoặc là một khối lệnh được bao trong một cặp ngoặc nhọn.

4) Cuối cùng, thực hiện để tăng biến điều khiển và vòng lặp quay trở lại bước kiểm tra điều kiện.

Phần khởi tạo và lệnh tăng không bắt buộc phải có. Chúng có thể được bỏ qua nhưng vẫn phải có dấu chấm phẩy ngăn cách giữa các phần. Vì vậy, chúng ta có thể viết for (;n Bằng cách sử dụng dấu phẩy, chúng ta có thể dùng nhiều lệnh trong bất kì trường nào trong vòng for, như là trong phần khởi tạo. Ví dụ chúng ta có thể khởi tạo một lúc nhiều biến trong vòng lặp:

```
for ( n=0, i=100 ; n!=i ; n++, i– )
{
// các câu lệnh;
}
```

VD: Tạo hàm delayms dung vòng lăp for

```
void delay (unsigned int ms) // ham tao thoi gian tre ms
{
```

unsigned int i ; // hoặc ta có thể khai báo int i j;

unsigned char j ;

for (i=0;i {

for (j=0;j0; n–) {

cout << n << “, “;

if (n==3)

{

cout << “dung dem”; break; //dem den 3 thi dung; } } return 0; } Lệnh continue. Lệnh continue làm cho chương trình bỏ qua phần còn lại của vòng lặp và nhảy sang lần lặp tiếp theo. Ví dụ chúng ta sẽ bỏ qua số 5 trong phần đếm ngược:

#include int main () { for (int n=10; n>0; n–) {

if (n==5) continue;

cout << n << “, “;

}

cout << “FIRE!”;

return 0;

}

Hàm exit.

Mục đích của exit là kết thúc chương trình và trả về một mã xác định. Dạng thức của nó như sau

void exit (int exit code);

exit code được dùng bởi một số hệ điều hành hoặc có thể được dùng bởi các chương trình gọi.

Theo quy ước, mã trả về 0 có nghĩa là chương trình kết thúc bình thường còn các giá trị khác 0 có nghĩa là có lỗi. các lệnh trên chủ yếu chỉ dùng lệnh break để thoát khỏi vòng lặp . Các lệnh khác thường rất ít dược sử dụng

Cấu trúc lựa chọn: switch

Cú pháp của lệnh switch hơi đặc biệt một chút. Mục đích của nó là kiểm tra một vài giá trị hằng cho một biểu thức, tương tự với những gì chúng ta làm ở đầu bài này khi liên kết một vài lệnh if và else if với nhau. Dạng thức của nó như sau:

```
switch (expression)
{
case constant1:
block of instructions 1
break;
case constant2:
block of instructions 2
break;
.
.
.
default:
default block of instructions
}
```

Nó hoạt động theo cách sau: switch tính biểu thức và kiểm tra xem nó có bằng constant1 hay không, nếu đúng thì nó thực hiện block of instructions 1 cho đến khi tìm thấy từ khoá break, sau đó nhảy đến phần cuối của cấu trúc lựa chọn switch. Còn nếu không, switch sẽ kiểm tra xem biểu thức có bằng constant2 hay không. Nếu đúng nó sẽ thực hiện block of instructions 2 cho đến khi tìm thấy từ khoá break. Cuối cùng, nếu giá trị biểu thức không bằng bất kỳ hằng nào được chỉ định ở trên (bạn có thể chỉ định bao nhiêu câu lệnh case tuỳ thích), chương trình sẽ thực hiện các lệnh trong phần default: Nếu nó tồn tại vì phần này không bắt buộc phải có.

2.2.5. Cấu trúc cơ bản của của một chương trình C cho 8051

+ Phần đầu tiên là liệt kê các header file

Các bạn dùng bằng từ khóa

#include “Tên các header”

Hoặc :

#incude

Khi viết theo cách thứ nhất thì trình biên dịch sẽ tìm kiếm file .h hoặc .c này trong thư mục hiện tại chứa dự án của bạn, nếu không có thì sẽ tìm kiếm trong thư mục Inc trong thư mục cài đặt KeilC. Viết theo cách thứ hai thì trình biên dịch sẽ tìm luôn trong thư mục /INC luôn. Để có thể sử dụng đúng các file .h cho các vi điều khiển mở thư mục /inc trong thư mục này có các thư mục con như tên của hãng sản xuất. Ví dụ như của Atmel thì bạn tìm trong thư mục /Atmel thì sẽ thấy được file reg51.h

Phần thứ 2 : Định nghĩa các macro (thiết lập vĩ mô). Cách khai báo sử dụng từ khóa #define. Ví dụ:để khai báo mặc led 1 được nối với chân 0 của port 1 ta viết như sau

#define led1 P1_0

+ Các hàm ngắt như ngắt (timer0, timer1, ngắt nối tiếp, ngắt ngoài)nêu ở phần khai báo biến . Copy lại như sau :

Cấu trúc:

void Tên hàm(void) interrupt nguồn ngắt using băng thanh ghi

{

}

Hàm ngắt không được phép trả lại giá tri hay truyền tham biến vào hàm

Tên hàm : tùy chọn

Interrupt : từ khóa chỉ hàm ngắt

Nguồn ngắt : từ 0 đến 5 theo bảng vecter ngắt

Ngắt do Cờ Địa chỉ vector Nguồn ngắt

Reset hệ thống RST 0000H -

Ngắt ngoài 0 IE0 0003H 0

Timer 0 TF0 000BH 1

Ngắt ngoài 1 IE1 0013H 2

Timer 1 TF1 001BH 3

Port nối tiếp RI hoặc TI 0023H 4

Timer 2 TF2 hoặc EXF2 002BH 5

Băng thanh ghi trên RAM chon từ 0 đến 3.

```
void ngat4(void) interrupt 4 using 2
{
//các câu lệnh
}
```

Cú pháp các ngắt khác cũng tương tự chỉ thay số 4 bằng số thứ tự của ngắt trong bảng vector ngắt.

+ Các hàm con như Delay, khởi tạo,..

Việc gây trễ trong Keil C có nhiều cách khác nhau

- Dùng vòng lặp while for :

Với tần số thạch anh 11.0582 MHz thì mỗi vòng lặp khi các bạn debug sẽ thấy là chúng ta mất thời gian thực khoảng 8.28 us. Do đó để có thể gây trễ 1ms thì các bạn cần dùng xấp xỉ 121 vòng lặp kiểu này. Viết chương trình như sau:

```
//*****************************
void delay (unsigned int ms) // ham tao thoi gian tre ms
{
unsigned int i ;
unsigned char j ; //khai bao bien 1 byte
for (i=0;i {
for (j=0;j {} // khong lam gi ca
}
}
```

- Dùng Timer 0 hoặc Timer 1

Tiếp tục với hàm delay() theo cách dùng bộ định thời thì ta thấy nó cũng giống như ngôn ngữ ASM biên dịch với Topview Simulator .

Dùng bộ định thời có 3 chế độ: chế độ 0, chế độ 1, chế độ 2. Chúng ta sẽ sử dụng chế độ khởi động bộ định thời bằng phần mềm tức TMOD.3 và TMOD.7 =0

Việc xác định chế độ nào phụ thuộc vào giá trị của 2 bit TM1 và TM0 của từng timer(các bạn xem định nghĩa từng bít trong thanh ghi TMOD)

TM1=0, TM0 =0 chế độ 0 : Chế độ định thời 13 bit , số đếm 0000H – 1FFFH

TM1=0, TM0 =1 chế độ 1 : Chế độ định thời 16 bit , số đếm 0000H – FFFFH

TM1=1, TM0 =0 chế độ 2 : Chế độ định thời 8 bit tự động nạp số đếm 00H – FFH

TM1=1, TM0 =1 chế độ 3 : Chế độ định thời chia sẻ số đếm 00H – FFH

VD : Gây trễ 1 ms = 1000us ta dùng chế độ định thời 16 bit sử dụng timer 0

Tdelay=1000 sử dụng calculator của hệ điều hành Windows XP trong Start\Program\Accessories\Calculator ta được

TH0=FC

TL0=18

Vậy chương trình sẽ như sau :

```
void delay(unsigned ms)
{
while (ms–)
{
TMOD=0×01; //dùng timer 0 chế độ 1 ( 16bit )
TH0=0xfc;
TL0=0×18; //hai câu lệnh nạp giá tri đếm
TR0=1; // cho phép timer 0 hoạt động
while (TF0); //chờ TF0=1(cờ tràn =1 )
TF0=0; //xóa cờ tràn
TR0=0; // ngừng Timer
}
}
```

+ Chương trình chính:

```
void main(void)
{
//cấu trúc lệnh điều khiển
}
```

đối tượng của chương trình là vi điều khiển nên hàm main không có giá trị trả về và không có tham số đưa vào.

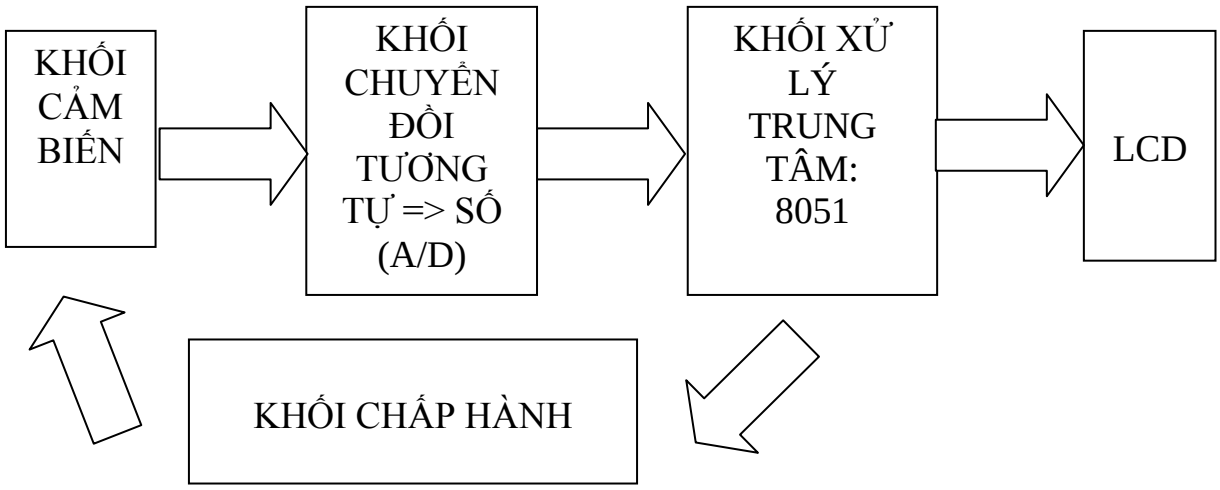
CHƯƠNG 3.

THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN SỐ NHIỆT ĐỘ

3.1. NGUYÊN LÝ HOẠT ĐỘNG

Để đo lường nhiệt độ trong mạch dùng LM335 là loại cảm biến có độ chính xác cao, tầm hoạt động tuyến tính từ -40÷100ºC, tiêu tán công suất thấp. Chuyển đổi từ tương tự sang số dùng IC ADC0804, hiển thị dùng LCD. Tín hiệu tương tự từ chân 2 của LM335 được đưa vào chân 6 (vin+) của ADC0804 để chuyển thành tín hiệu số, chân 1(cs) của ADC tích cực ở mức thấp, R4 và C1 tạo dao động riêng cho ADC. Tín hiệu số từ ADC được đưa vào port 3 của VDK tín hiệu ra từ port 0 được hiển thị bằng LCD. Trở treo RP1 giúp truy nhập được ở port 0. Tín hiệu đầu ra của cảm biến chuyển động sẽ là 0V hoặc 5V phụ thuộc vào tín hiệu đầu vào tức là khi đầu vào được kích thích bởi ánh sáng hồng ngoại thì đầu ra sẽ là 5V. Tín hiệu ra sẽ qua transitor Q1 khuyếch đại rồi đến port 2 củaVDK, Tín hiệu ra trên port 1 của VDK qua transitor Q2 để điều khiển Relay. Khi có ánh sáng thì Relay đóng, khi ko có ánh sáng thi Relay mở. Bộ tạo giao động cho VDK gồm C5, C6, X1.

3.2. SƠ ĐỒ TỔNG QUÁT



Giải thích sơ đồ:

1) Khối cảm biến nhiệt độ: Có chức năng nhận tín hiệu nhiệt độ từ bên ngoài và cảm biến thành tín hiệu điện áp.

2) Khối chuyển đổi tương tự sang số (A/D): Nhận tìn hiệu tương tự ở đầu vào chuyển đổi sang tín hiệu số ở đầu ra 8 bit. (D0 - D7)

3) Khối xử lý trung tâm: Dùng để lấy tín hiệu đầu vào và xử lý sau đó xuất ra tín hiệu cho đầu ra.

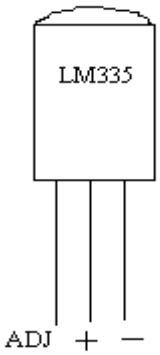
4) Khối hiển thị: Dùng để hiện thị được đưa ở đầu ra của khối xử lý.

5) Khối chấp hành: Thực hiện các tín hiệu từ IC

3.2.1. Khối cảm biến nhiệt độ

Có nhiệm vụ cảm biến nhiệt độ môi trường xung quanh và biến nhiệt độ đó thành đại lượng điện áp ổn định và thay đổi tuyến tính theo nhiệt độ môi trường.

Có rất nhiều loại cảm biến đo nhiệt độ trên thị trường nhưng dễ sử dụng và thông dụng nhất vẫn là LM335.



Hình 3.1: Chân LM335

Các tham số của LM335:

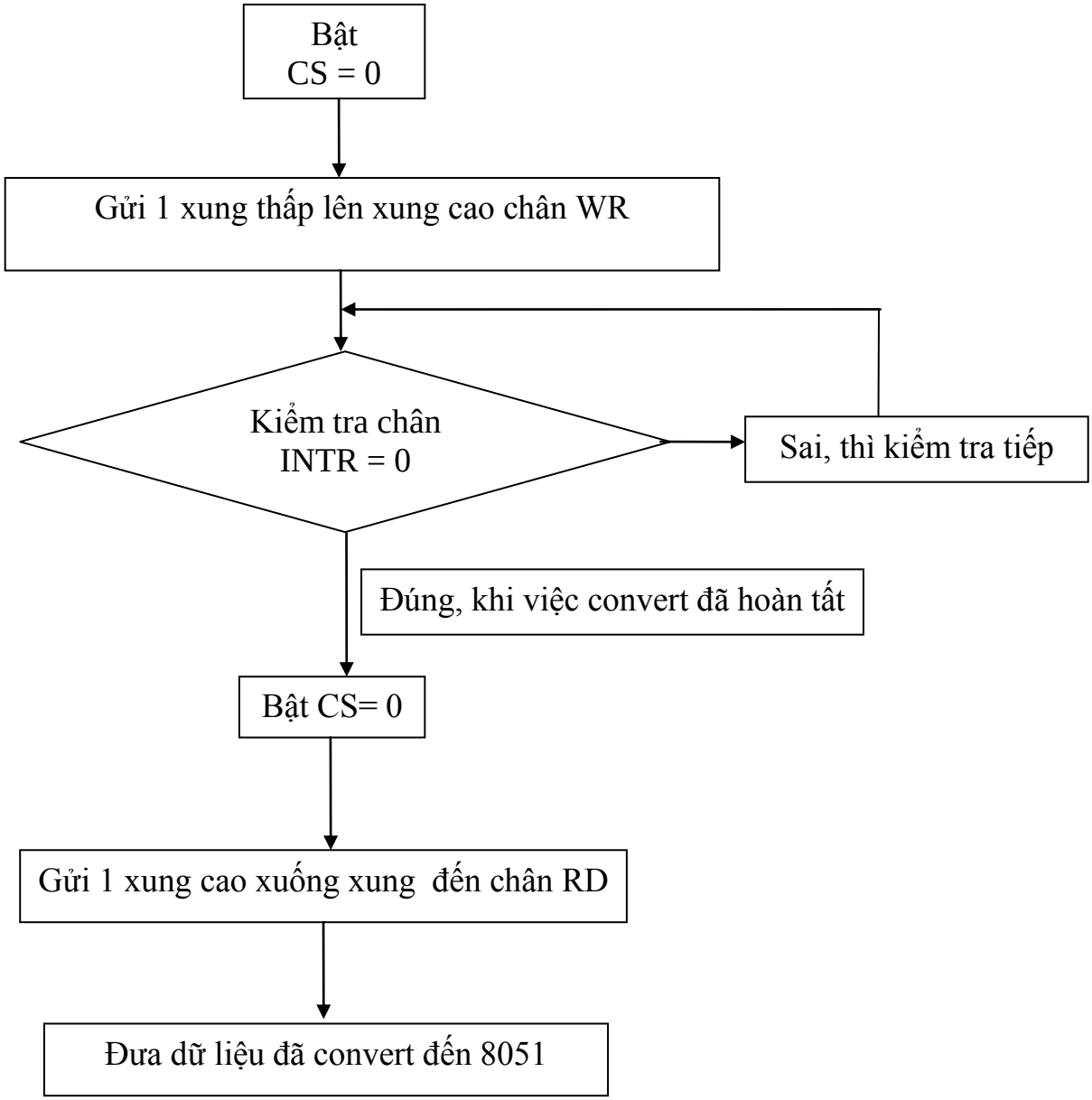
- Có độ biến thiên theo nhiệt độ là:10mv/K^0.
- Có sự ổn định cao, chỉ sai số khoảng 1%.
- Hàm điện áp biến thiên tuyến tính khi đo trong khoảng -40–100^{0}C

- Tiêu tán công suất thấp.
- Dòng điện làm việc từ 4.10^{-4}A đến 5.10^{-3}A.
- Dòng ngược 15mA.
- Dòng thuận 10mA.

Biến thiên của điện áp theo nhiệt độ: Vout=2.73+0.01xT^0C

3.2.2. Khối chuyển đổi tương tự sang số

Thuật toán:

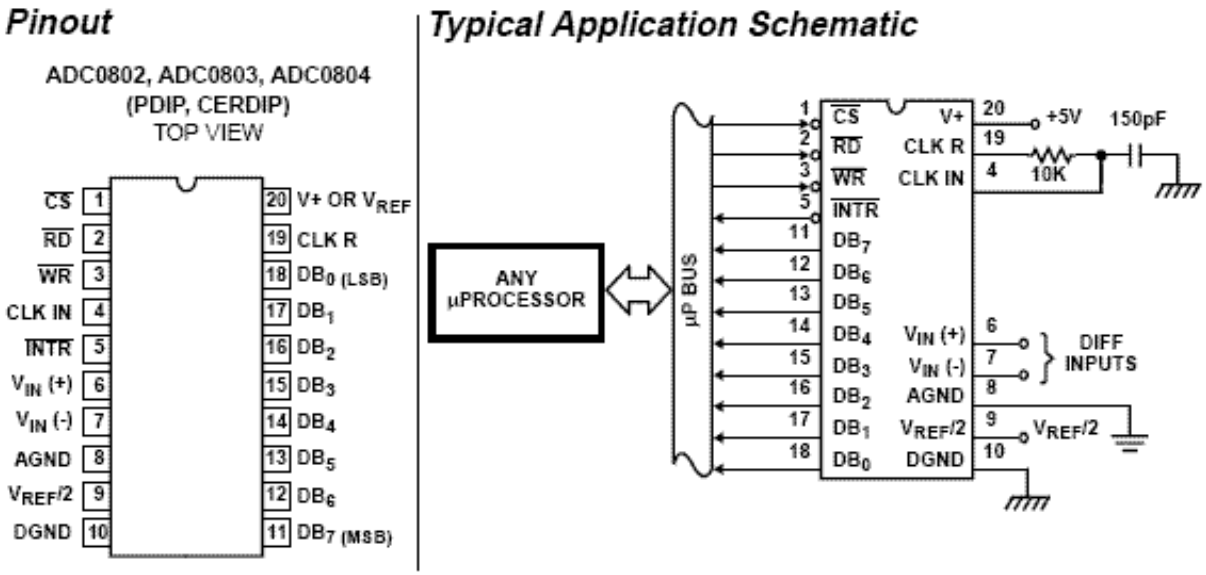


3.2.2.1. Giới thiệu về bộ chuyển đổi ADC

Bộ chuyển đổi ADC là bộ chuyển đổi tín hiệu ở dạng tương tự sang dạng số để có thể làm việc được với CPU.Ứng dụng này chủ yếu mô tả cách thức tối ưu hóa ADC (Analog to Digital Convertor) trong các phần cứng để không làm thay đổi bản chất của nó và làmcho nó hoạt động tốt nhất. Phương pháp này phụ thuộc vào các nhiễu bên trongcủa ADC và các nhiễu bên ngoài như: trở kháng, nguồn, các vòng dây và anten.

3.2.2.2. Tìm hiểu về ADC 0804

Chip ADC0804 là bộ chuyển đổi tương tự số thuộc họ ADC800 của hãng National Semiconductor. Chip này cũng được nhiều hãng khác sản xuất. Chip có điện áp nuôi +5V và độ phân giải 8 bit. Ngoài độ phân giải thì thời gian chuyểnđổi cũng là một tham số quan trọng khi đánh giá bộ ADC. Thời gian chuyển đổiđược định nghĩa là thời gian mà bộ ADC cần để chuyển một đầu vào tương tự thành một số nhị phân. Đối với ADC0804 thì thời gian chuyển đổi phụ thuộcvào tần số đồng hồ được cấp tới chân CLK và CLK IN và không bé hơn 110μs.



Hình 3.2: Cấu tạo ADC 0804

Các chân khác của ADC0804 có chức năng như sau:

- **CS (Chip select)**

Chân số 1, là chân chọn Chip, đầu vào tích cực mức thấp được sử dụng để kích hoạt chip ADC0804. Để truy cập ADC0804 thì chân này phải ở mức thấp.

- **(Read)**

Chân số 2, là một tín hiệu vào, tích cực ở mức thấp. Các bộ chuyển đổi đầu vào tương tự thành số nhị phân và giữ nó ở một thanh ghi trong. RD được sử dụng để có dữ liệu đã được chuyển đổi tới đầu ra của ADC0804. Khi CS = 0 nếu có một xung cao xuống thấp áp đến chân RD thì dữ liệu ra dạng số 8 bit được đưa tới các chân dữ liệu (DB0 – DB7).

- **WR (Write)**

Chân số 3, đây là chân vào tích cực mức thấp được dùng để báo cho ADC biết bắt đầu quá trình chuyển đổi. Nếu CS = 0 khi WR tạo ra từ xung cao xuống xung thấp thì bộ ADC0804 bắt đầu quá trình chuyển đổi giá trị đầu vào tương tự.

V_{in} về số nhị phân 8 bit. Khi việc chuyển đổi hoàn tất thì chân INTR được ADC hạ xuống thấp.

- **CLK IN và CLK R**

CLK IN (chân số 4), là chân vào nối tới đồng hồ ngoài được sử dụng để tạo thờigian. Tuy nhiên ADC0804 cũng có một bộ tạo xung đồng hồ riêng. Để dùngđồng hồ riêng thì các chân CLK IN và CLK R (chân số 19) được nối với một tụ điện và một điện trở (như hình vẽ).

- **Ngắt INTR (Interupt)**

Chân số 5, là chân ra tích cực mức thấp. Bình thường chân này ở trạng thái caovà khi việc chuyển đổi hoàn tất thì nó xuống thấp để báo cho CPU biết l à dữliệu chuyển đổi sẵn sàng để lấy đi. Sau khi INTR xuống thấp, cần đặt CS = 0 v àgửi một xung cao xuống thấp tới chân RD để đưa dữ liệu ra.

- **Vin (+) và Vin (-)**

Chân số 6 và chân số 7, đây là 2 đầu vào tương tự vi sai, trong đó Vin = Vin (+)Vin (-). Thông thường Vin (-) được nối tới đất và Vin (+) được dùng làm đầuvào tương tự và sẽ được chuyển đổi về dạng số.

- **Vcc**

Chân số 20, là chân nguồn nuôi +5V. Chân này còn được dùng làm điện áp thamchiếu khi đầu vào Vref/2 để hở.

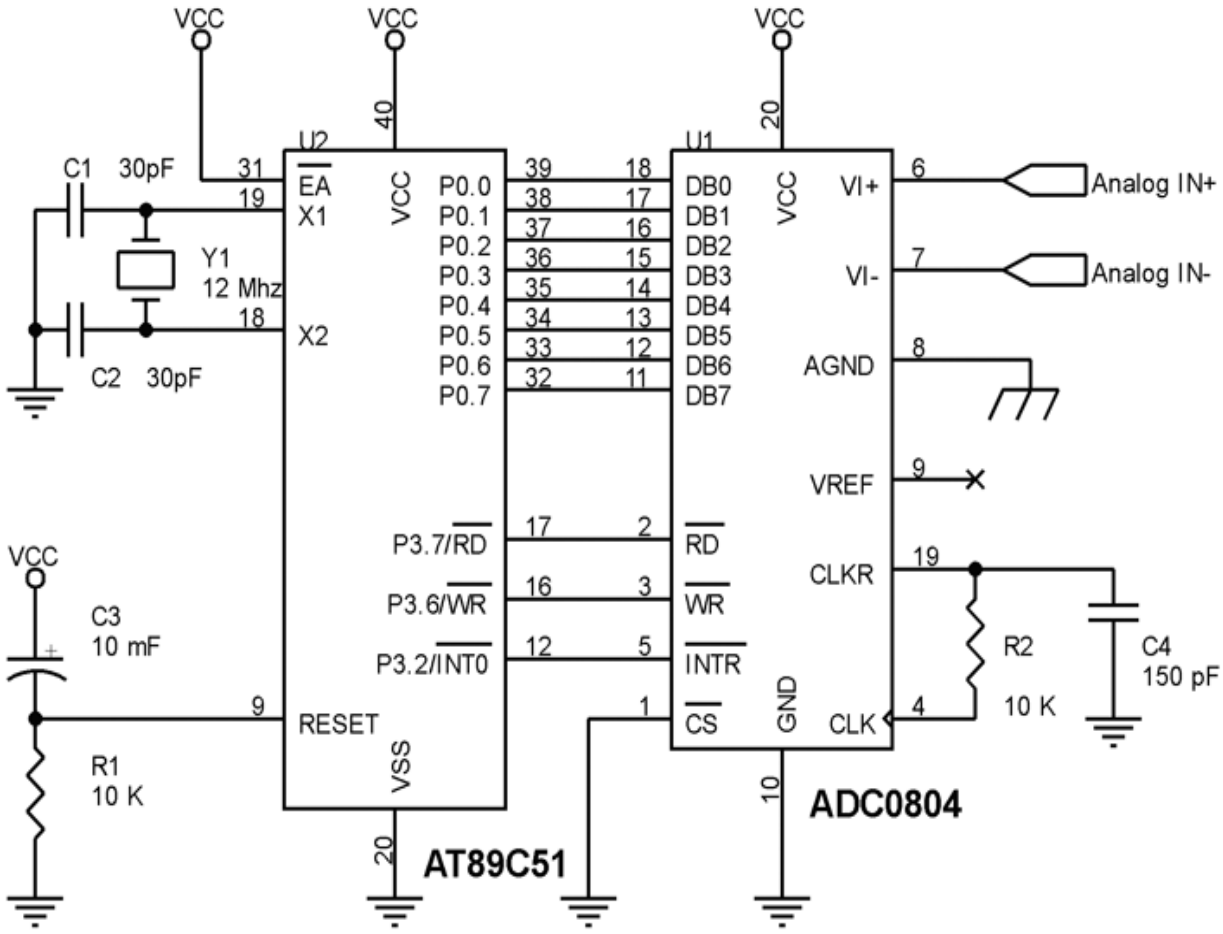
- **Vref/2**

Chân số 9, là chân điện áp đầu vào được dùng làm điện áp tham chiếu. Nếu chânnày hở thì điện áp đầu vào tương tự cho ADC0804 nằm trong dải 0 - +5V. Tuynhiên, có nhiều ứng dụng mà đầu vào tương tự áp đến Vin khác với dải 0 - +5V.Chân Vref/2 được dùng để thực hiện các điện áp đầu ra khác 0 - +5V.

- **D0 - D7**

D0 - D7, chân số 18 – 11, là các chân ra dữ liệu số (D7 là bit cao nhất MSB và D0 là bit thấp nhất LSB). Các chân này được đệm ba trạng thái và dữ liệu đã được chuyển đổi chỉ được truy cập khi chân CS = 0 và chân RD đưa xuống mức thấp.

3.2.3. Khối xử lý trung tâm



Hình 3.3: Ghép nối IC 8051 và ADC 0804

3.2.4. Khối hiển thị

Trong các ứng dụng của vi điều khiển thì LCD đóng vai trò quan trọng nó là bộ phận giao tiếp giữa người và thiết bị. Có rất nhiều loại LCD khác nhau của các hãngkhác nhau. Có loại LCD 8x1,8x2,16x2...Ngày nay, hầu hết các bộ hiển thị LCD thông minh đều tuân theo một tiêu chuẩn chung. Tùy theo yêu cầu về hiển thị thông tin mà ta chọn loại nào cho phù hợp. Trong đồ án này em dùng LCD loại 16x2 2 dòng 16 kí tự trên một dòng. Do loại này dễ dùng và giá thành cũng phải chăng nên em dùng để hiển thị.

3.2.4.1. Cấu tạo

Bảng 3.1: Chức năng các chân của LCD 16x2:

Chân số	Ký hiệu	Mức logic	I/O	Chức năng
1,15	Vss	-	-	Nguồn cung cấp (GND)
2,16	Vdd	-	-	Nguồn cung cấp (+5V)
3	Vee	-	I	Điện áp để điều chỉnh tương phản
4	RS	0/1	I	Lựa chọn thanh ghi 0 = thanh lệnh ghi 1 = Thanh ghi dữ liệu
5	R/W	0/1	I	0 = Thanh ghi vào LCD module 1= Đọc từ LCD module
6	E	1,1=>0	I	Tín hiệu cho phép 0 = Vô hiệu hóa 1 = Hoạt động Từ 1 xuống 0: Bắt đầu đọc ghi
7	DB1	0/1	I/O	Data bus line 0(LSB)
8	DB2	0/1	I/O	Data bus line1
9	DB3	0/1	I/O	Data bus line2
10	DB4	0/1	I/O	Data bus line3
11	DB5	0/1	I/O	Data bus line 4
12	DB6	0/1	I/O	Data bus line 5
13	DB7	0/1	I/O	Data bus line 6
14	DB8	0/1	I/O	Data bus line 7(MSB)

3.2.4.2. Nguyên tắc hiển thị ký tự trên LCD

Một chương trình hiển thị ký tự trên LCD sẽ đi theo bốn bước sau:

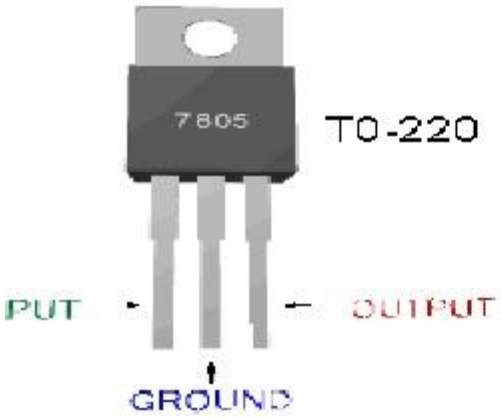
1) Xóa toàn bộ màn hình.

2) Đặt chế độ hiển thị.

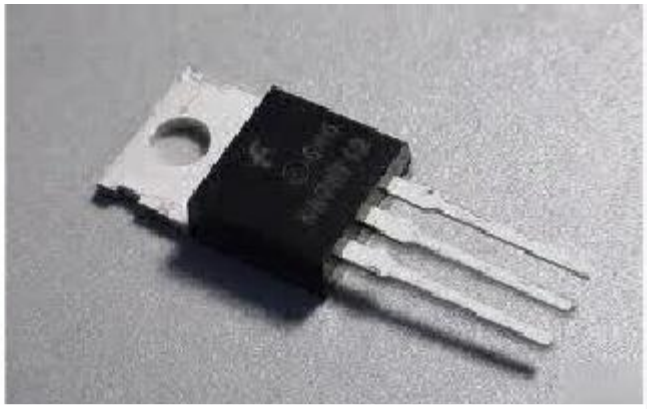
3) Đặt vị trí con trỏ (nơi bắt đầu của ký tự hiển thị).

4) Hiển thị ký tự.

3.2.5. Khối nguồn:



Hình3.4: 7805



Hình 3.5: 7812

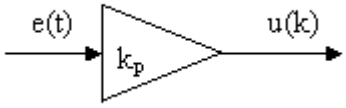
Hình 3.6: Biến áp 3A

- Biến áp 3A: Làm nhiệm vụ biến đổi điện áp 220V-50Hz thành điện áp 12V-50Hz
- 7812, 7805: Làm nhiệm vụ ổn định điện áp chuẩn 12V, 5V theo yêu cầu của mạch điều khiển

3.3. CÁC LUẬT ĐIỀU KHIỂN SỐ

Yêu cầu thiết kế được đặt ra là bộ PID số phải có tính linh hoạt cao, có nghĩa là phải có giao điều khiển các đối tượng công nghiệp theo luật P, I, PI, PD và có thể lựa chọn tham số của các luật phù hợp với đối tượng thiết kế. Luật PID số phải được thiết kế gọn gàng, thời diện thân thiện với người sử dụng. Thông qua HMI, người sử dụng có thể chọn luật điều khiển dễ dàng. Ví dụ như có thể gian xử lý lệnh phải nhanh để làm tăng tính thời gian thực cho thiết bị điều khiển.

3.3.1. Luật điều khiển tỷ lệ số



Hình 3.7: Cấu trúc luật P số.

Đây là luật điều khiển có thể thiết kế đơn giản nhất. Dãy u(k) được tính từ dãy e(k) theo công thức:

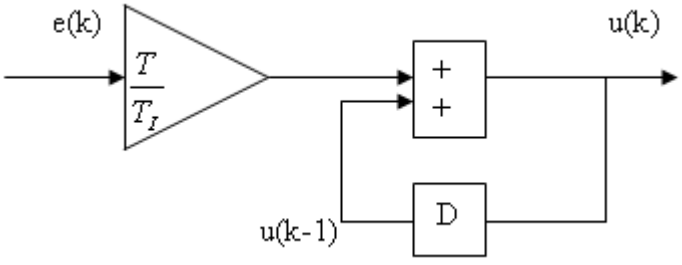
$$u(k) = k_P e(k) \qquad \text{k=0,1,2 ...} \qquad (3.1)$$

3.3.2. Luật điều khiển tích phân số

Ta có phương trình sai phân:

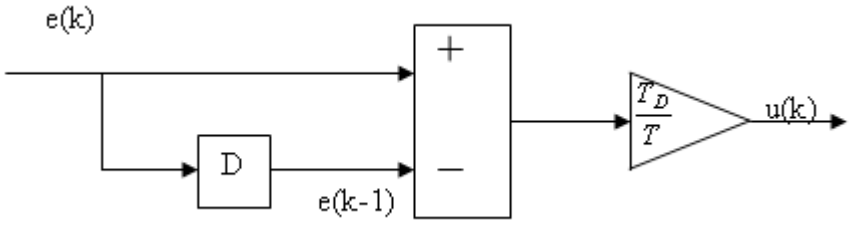
$$u(k) = \frac{T}{T_I} e(k) + u(k-1) \qquad (3.2)$$

Trong đó T là thời gian trích mẫu (Sample Time)



Hình 3.8: Cấu trúc luật I số.

3.3.3. Luật điều khiển vi phân số



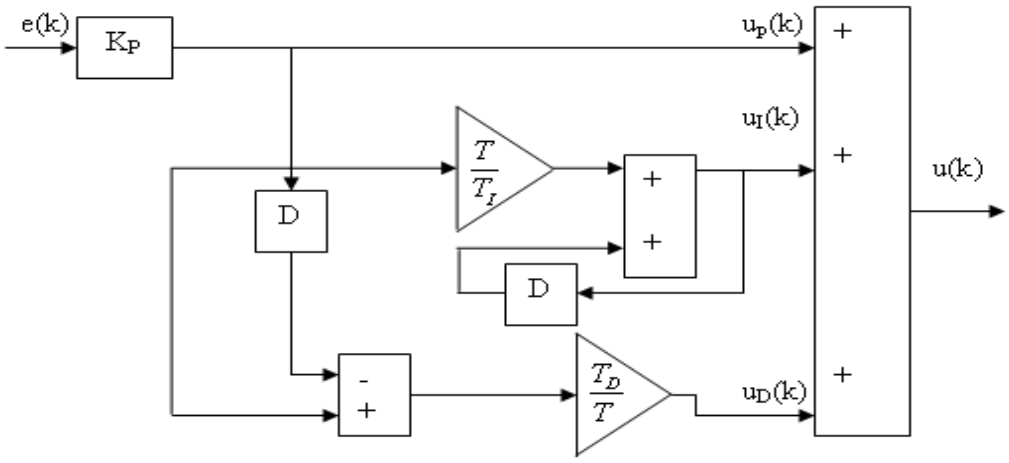
Hình 3.9: Cấu trúc luật D số.

Thường các bộ điều khiển theo luật vi phân số được cài đặt theo các phương trình sai phân sau:

$$u(k) = \frac{T_D}{T}[e(k) - e(k-1)] \qquad (3.3)$$

Trong đó T là thời gian trích mẫu.

3.3.4. Luật điều khiển PID số



Hình 3.10: Cấu trúc luật PID số.

Từ cấu trúc PID số trong Hình 3.5, ta có:

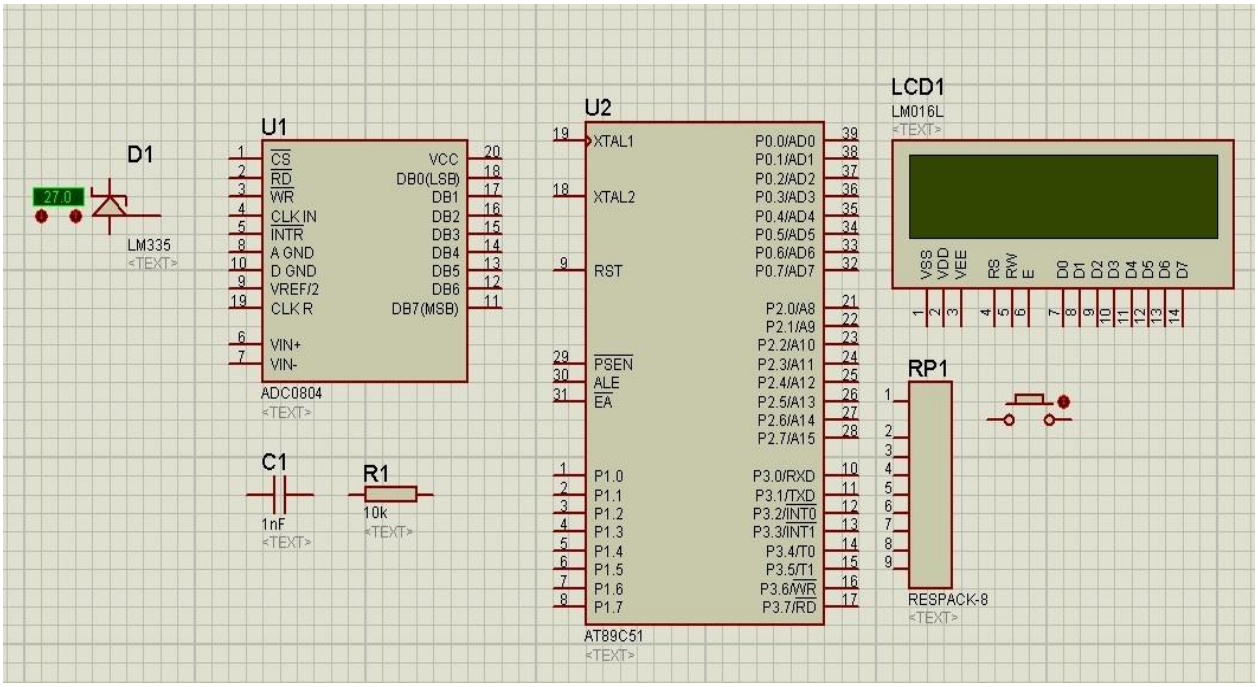
$$u(k) = k_P\left\{e(k) + \frac{T}{T_I}e(k) + u_I(k-1) + \frac{T_D}{T}\ e(k) - e(k-1)\right\} \qquad (3.4)$$

$$u(k) = k_P\left\{(1+\frac{T_D}{T})e(k) - \frac{T_D}{T}e(k-1) + \frac{T}{T_I}e(k) + u_I(k-1)\right\}$$

$$u(k) = k_P\left\{(1+\frac{T_D}{T}+\frac{T}{T_I})e(k) - \frac{T_D}{T}e(k-1) + u_I(k-1)\right\}$$

Luật điều khiển PID số trong công thức trên được lựa chọn để cài đặt cho bộ điều khiển được chế tạo trên chip PIC.

3.4. CÁC LINH KIỆN TRONG MẠCH



Hình 3.11: Các linh kiện trong mạch.

1) Cảm biến: LM335

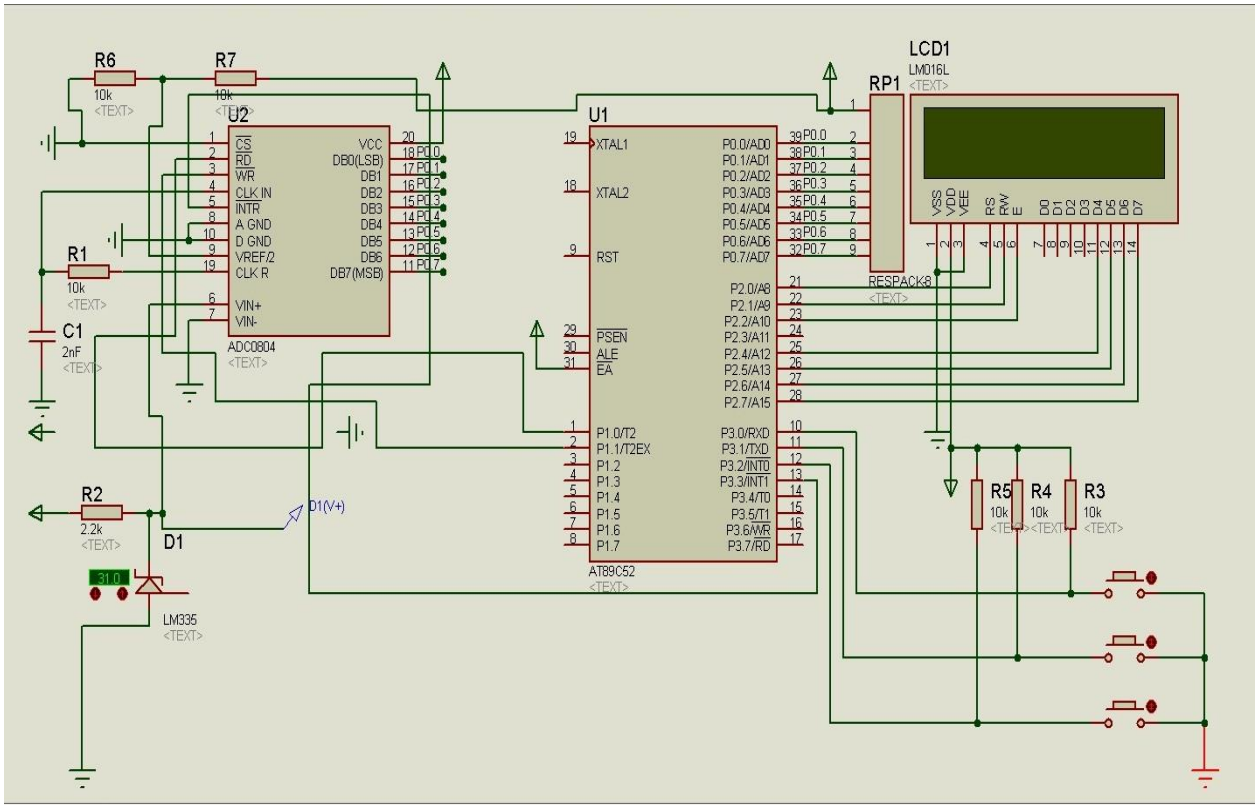
2) IC chuyển đổi tương tự sang số: ADC 0804

3) Xử lý trung tâm: AT89C51

4) Khối hiển thị: LCD LM016L

5) Điện trở, tụ điện, nút bấm, transistor, diode…

3.5. SƠ ĐỒ NGUYÊN LÝ



Hình 3.12: Mạch cảm biến nhiệt độ dùng LM335.

Đo nhiệt độ tại đối tượng thông qua sensor nhiệt LM335. LM335 là sensor đo nhiệt độ với đầu ra là 10mV/^{0}K, do đó để đo độ C ta cần có công thức chuyển đổi giá trị từ độ K sang độ C. Vì ta dùng ADC của PIC là 10 bit nên giá trị số lớn nhất là 1023. Vref=Vcc, giả thiết là VCC=5V nên tại 0^{o}C hay 273^{o}K thì đầu ra của LM335 có giá trị là 2.73V. Như vậy khi muốn tính toán ra độ C ta cần phải trừ đi mức điện áp là 2.73V.

Ví dụ: Nhiệt độ là 30^{o}C = 303^{o}K, mức điện áp tương ứng là

out = 303 x 10mV/^{o}K =3.03V.

Ta tính toán giá trị đọc được từ ADC.

- Với ADC 10 bit (V_in là điện áp đưa vào chân ADC của PIC):

V_in = 5V => ADC_value = 1023

V_in = 2.73V => ADC_value = (1023/5)x2.73=558.6 (tương ứng 0^0)

mặt khác do V_ref = VCC = 5V nên ADC_value = 1 tương ứng với 5/1023=4.9mV÷5mV. Trong khi đó LM335 cho ra điện áp là 10mV/1^{o}K nên để

giá trị ADC thay đổi 1 đơn vị thì nhiệt độ phải thay đổi là 0.5^{o}K (hay gần 5mV)

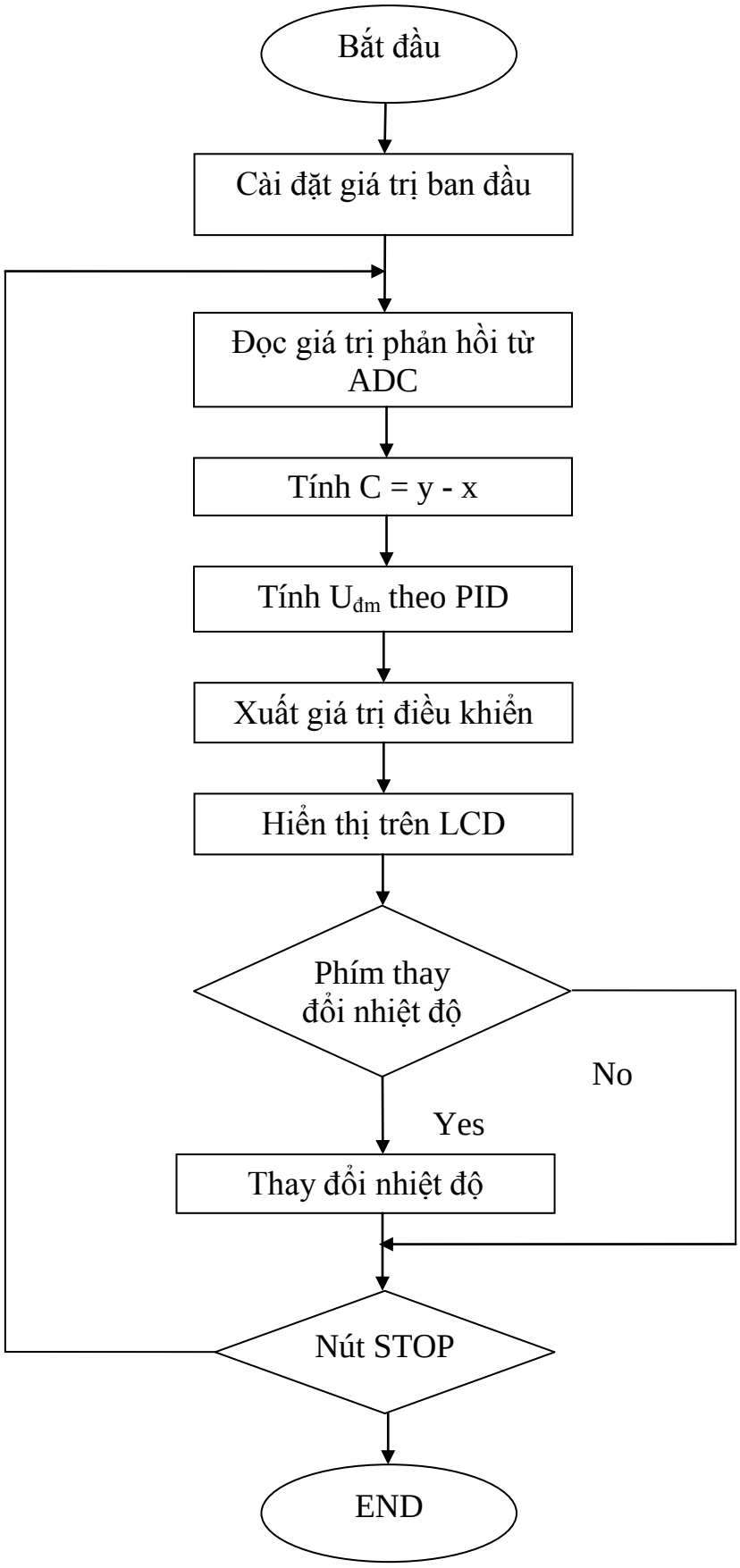
Từ đó ta có công thức đầy đủ sau để tính giá trị ^{o}C:

$$T[C] = \frac{(ADC\_value - 558.6).5V}{1023x10mV}$$

Vậy ta có công thức rút gọn là:

$$T[C] = \frac{(ADC\_value - 558.6)}{2.046}$$

3.6. SƠ ĐỒ THUẬT GIẢI



3.7. CHƯƠNG TRÌNH ĐIỀU KHIỂN

```
#include <REGX52.H>
#include<delay.h>
#include<lcd4bit.h>

#define W P1_1//  Chan yeu cau xuat du lieu chuyen doi
#define R P1_0//  Chan yeu cau bat dau lay mau

#define Mode        P3_0//        Mode
#define Up          P3_1//        Up
#define Down        P3_2//        Dowm
#define Fan  P1_2//         Dowm

unsigned char i=0;
unsigned int time=0,dl=0;
unsigned int nhiet_do_M=0,nhiet_do=0;
float data_temp=0,data_in=0;
bit
enable_sampling=0,enable_display=0,Mode_M=0,Up_M=0,Down_M=0,enable
_time=0,check_point=0;
void init(void)
{
        P3=0xFF;
        P1=0xFF;
        P0=0xFF;
        W=1;
        R=1;
        TMOD=0x02;//  Chon timer 0 che do 8 bit tu lap lai
        TH0=0xC8;//  Thoi gian ngat ~100us_Clock=24MHz
        EA=1;//  Cho phep ngat toan cuc
```

```
        ET0=1;//  Dung ngat dinh thoi timer 0
        EX1=1;//  Dung ngat ngoai cua timer 1
        IT1=1;//  Ngat theo suon xuong
        TR0=1;//  Khoi dong timer 0
}
void sampling()
{
        if(enable_sampling==1)//  Neu dc phep lay mau
                {
                W=0;//  Xung tich cuc thap
                delay_ms(100);
                W=1;//  Tra lai tich cuc cao
                enable_sampling=0;//  Thoi lay mau_Cho lan lay mau sau
                }
}
//------------------------------Ham cai dat nhiet do------------------------------
void Setup(void)
{
        lcd_clear();//          Xoa LCD
        Mode_M=0;//         Da thuc hien Mode
        lcd_putsf("SetPoint: ");
        lcd_put_int(nhiet_do);
        nhiet_do_M=nhiet_do;
        while(Mode_M==0)
        {
                if(Up_M==1 || Down_M==1)
                {
                        if(Up_M==1 && nhiet_do_M<80)
lcd_gotoxy(0,10),nhiet_do_M++,lcd_put_int(nhiet_do_M);//      Tang nhiet do
```

```
                    else if(Up_M==1 && nhiet_do_M>=80)
        lcd_gotoxy(0,10),nhiet_do_M=25,lcd_put_int(nhiet_do_M);//    Quay
vong cai dat
                    else if(Down_M==1 && nhiet_do_M>25)
lcd_gotoxy(0,10),nhiet_do_M--,lcd_put_int(nhiet_do_M);//       Giam nhiet do
                    else if(Down_M==1 && nhiet_do_M<=25)
        lcd_gotoxy(0,10),nhiet_do_M=80,lcd_put_int(nhiet_do_M);//    Quay
vong cai dat
                    Up_M=0,Down_M=0;
                }
        }
        if(nhiet_do>nhiet_do_M)P3_7=0,Fan=1;//    Neu nhiet do cai dat nho hon
nhiet hien tai thi tat
        else if(nhiet_do<nhiet_do_M)    P3_7=1,Fan=0;//    Neu nhiet do cai dat
lon hon nhiet hien tai thi bat
        lcd_clear();//          Xoa LCD
        nhiet_do=(data_temp-139.8)/0.512;//5v
        lcd_putsf("SetPoint: ");
        lcd_put_int(nhiet_do_M);//   Hien thi gia tri moi cai dat
        enable_display=0;
        lcd_gotoxy(1,0);//  Xuong hang duoi
        lcd_putsf("Feedback: ");
        lcd_put_int(nhiet_do);
        Mode_M=0;//         Da thuc hien Mode
        check_point=1;//    Xac nhan cai dat
}

void main(void)
{
        init();//  Goi ham khoi tao ban dau
```

```
        lcd_init();//  Goi ham khoi tao LCD
        delay_ms(100);
        lcd_clear();
        P3_7=0;//     Tat soi dot
        Fan=0;//   Tat quat
        while(1)
        {
                if(enable_sampling==1)
                {
                        sampling();//  Goi ham lay mau_Cu sau 0.5s thi lay mau
nhiet do 1 lan
                }
                if(enable_display==1)
                {
                        lcd_clear();
                        nhiet_do=(data_temp-139.8)/0.512;//5v
                        lcd_putsf("SetPoint: ");
                        enable_display=0;
                        lcd_gotoxy(1,0);//  Xuong hang duoi
                        lcd_putsf("Feedback: ");
                        lcd_put_int(nhiet_do);
                        if(check_point==1)//  Xac nhan cai dat
                        {
                                if(nhiet_do>nhiet_do_M)Fan=1,P3_7=0;//    Neu nhiet
do cai dat nho hon nhiet hien tai thi tat
                                else if(nhiet_do<nhiet_do_M)    P3_7=1,Fan=0;//  Neu
nhiet do cai dat lon hon nhiet hien tai thi bat
                                lcd_gotoxy(0,10);
                                lcd_put_int(nhiet_do_M);
                        }
```

```
            }
            if(Mode_M==1)    Setup(),Mode_M=0;//      Da thuc hien Mode
        }
}


void timer_0(void) interrupt 1//Ngat timer 0
{
        time++;
        if(time==3000 && enable_sampling==0)//Neu dat 100us*200=20ms va
da lay mau thanh cong truoc do thi lay mau tiep
        {
            enable_sampling=1;//Ra lenh lay mau
            time=0;
        }
        else if(time==3000 && enable_sampling==1)
        {
            time=0;
        }
        if(Mode==0 && Up==1 && Down==1 && enable_time==0)// Neu an
Mode
        {
            Mode_M=1;// Xac nhan
            enable_time=1;// Bat dau tinh thoi gian chong rung phim
            dl=4000;
        }
        else if(Up==0 && Down==1 && Mode==1  && enable_time==0)//
        Neu an Up
        {
```

```
            Up_M=1;// Xac nhan
            enable_time=1;// Bat dau tinh thoi gian chong rung phim
            dl=4000;
        }
        else if(Down==0 && Up==1 && Mode==1  && enable_time==0)//
        Neu an Down
        {
            Down_M=1;// Xac nhan
            enable_time=1;// Bat dau tinh thoi gian chong rung phim
            dl=4000;
        }
        if(enable_time==1)
        {
            dl--;
            if(dl==0)      enable_time=0;//   Neu het thoi gian thi cho phep an
nut
        }
}
void ext_1(void) interrupt 2//Ngat ngoai 0
{
R=0;
data_in=P0;//Lay du lieu tu ADC
if(data_in!=data_temp)    enable_display=1;
data_temp=data_in;// Copy
R=1;
}
```

3.8. XÂY DỰNG MÔ HÌNH

Hình 3.11: Tổng thể mô hình

Thực hiện điều khiển mạch như sau:

1) Bắt đầu khi cấp nguồn cho hệ thống cảm biến sẽ đo nhiệt độ hiện tại trong buồng.

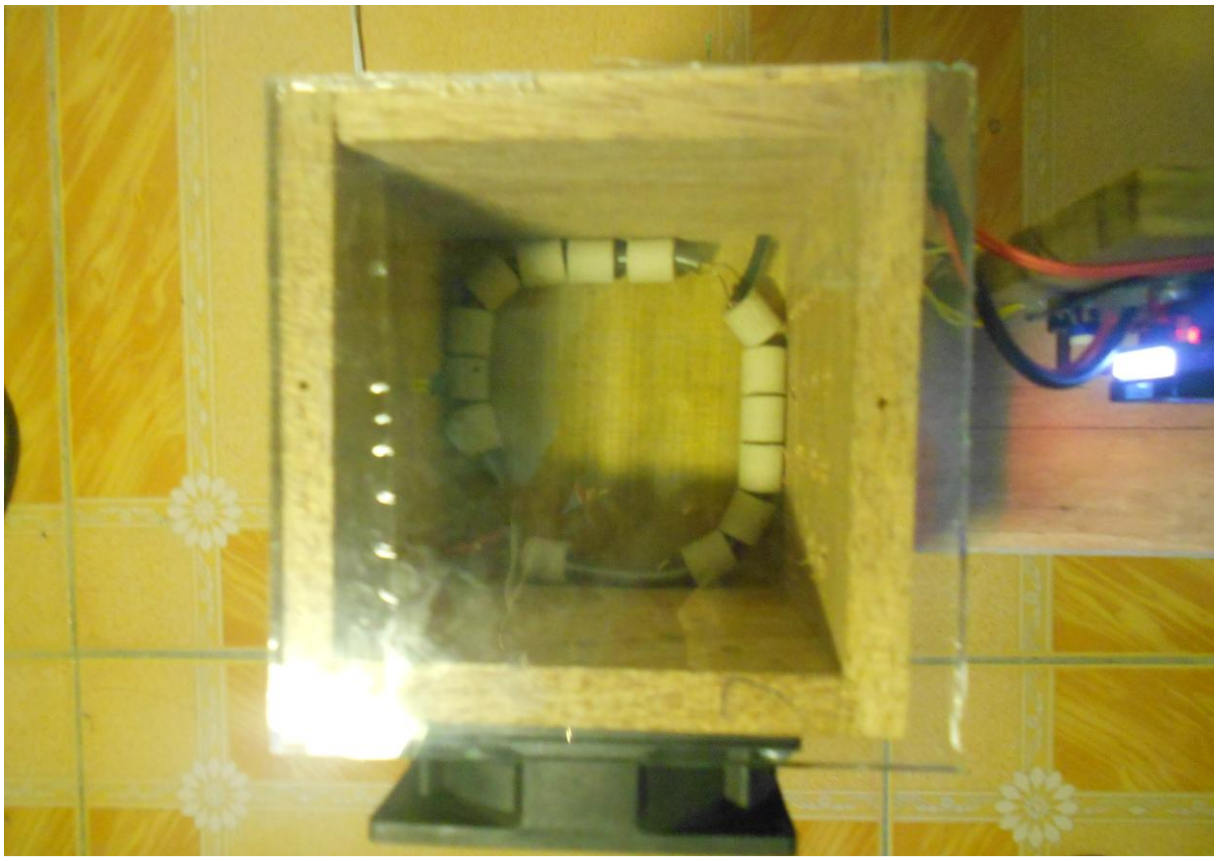
   Bấm MENU để cài đặt nhiệt độ:

2) Tăng nhiệt độ:

- Cấp nhiệt cho gia nhiệt
- Cảm biến sẽ thực hiện đo nhiệt độ với bước là 2ºC và thể hiện trên LCD
- Khi đạt đến nhiệt độ đặt hệ thống sẽ dừng cấp nhiệt cho gia nhiệt

3) Giảm nhiệt độ:

- Không cấp nhiệt cho gia nhiệt
- Tương tự cảm biến sẽ đo nhiệt độ với bước là 2ºC được thể hiện trên LCD
- Trong quá trình giảm nhiệt độ hệ thống sẽ khởi động quạt tản nhiệt
- Khi đạt đến nhiệt độ đặt hệ thống sẽ ngừng quạt tản nhiệt

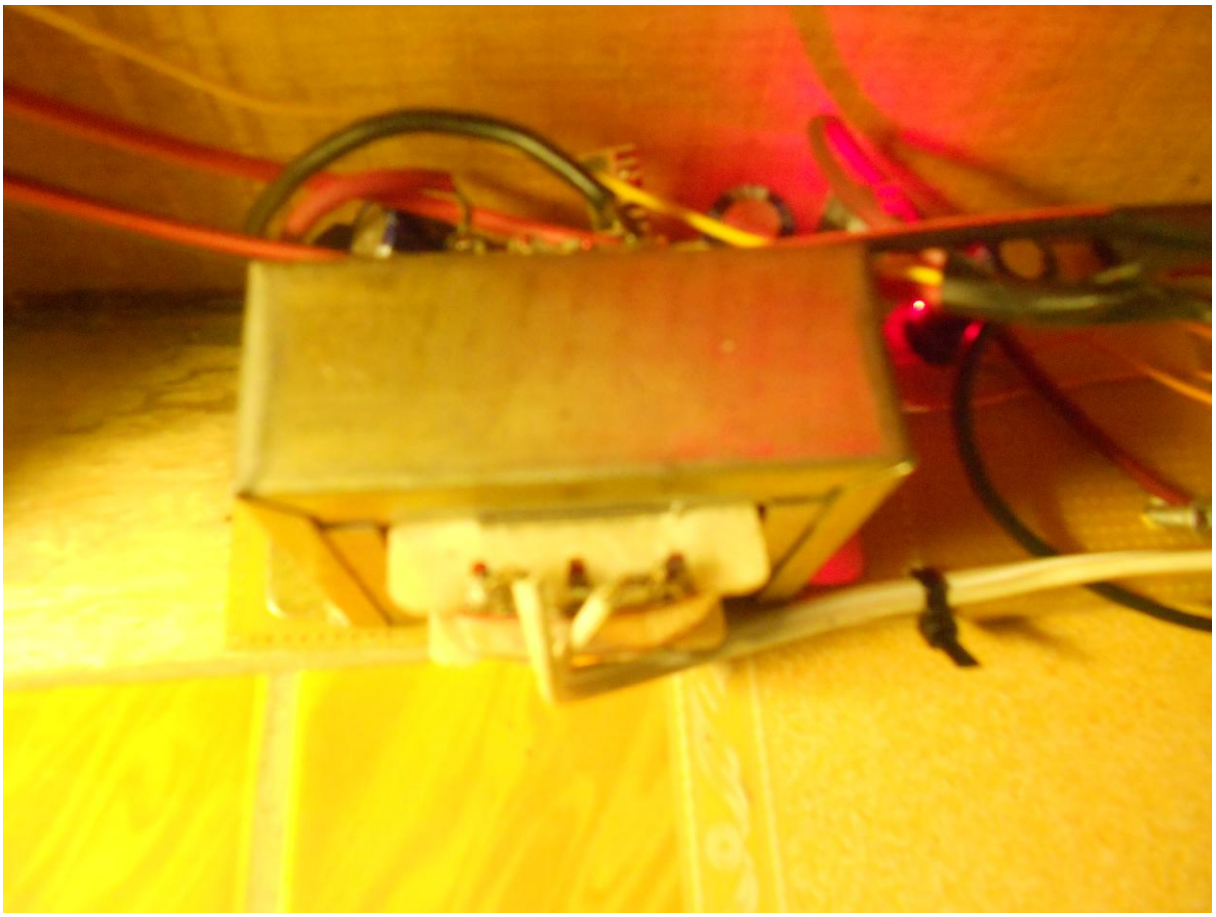


Hình 3.12: Hệ thống gia nhiệt

Hệ thống gia nhiệt bao gồm:

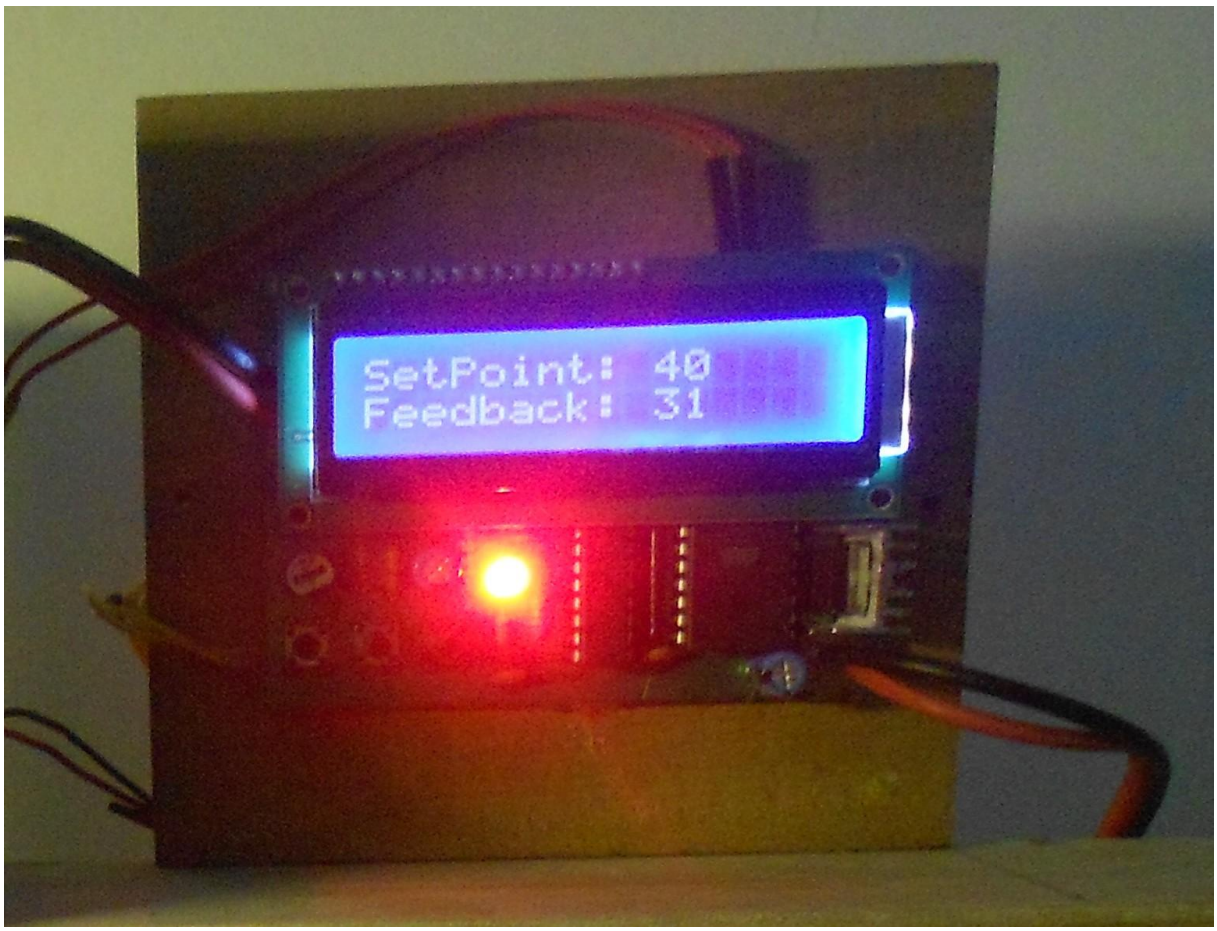
1) Dây gia nhiệt và gốm cách điện
2) Buồng kín
3) Quạt tản nhệt
4) Giá đỡ

Do lần đầu thực hiện nên em chọn dây gia nhiệt chưa đáp ứng được với yêu cầu của mạch thiết kế. Làm nhiệt độ tăng chậm



Hình 3.13: Mạch nguồn của hệ thống

Cung cấp nguồn điện cho hệ thống gia nhiệt và hệ thống điều khiển với 2 điện áp là 12V và 5V.



Hình 3.14: Mạch điều khiển

Giao diện điều khiển 3 nút ấn:

1) MENU
2) Tăng
3) Giảm

KẾT LUẬN

Sau thời gian nghiên cứu và tìm hiểu, với sự hướng dẫn của thầy Nguyễn Trọng Thắng và sự giúp đỡ của thầy cô trong khoa Điện tự động trường đại học Dân Lập Hải Phòng, em đã hoàn thành được đồ án của mình.

Qua đồ án này em đã thu được những kết quả sau:

- Hiểu được phương pháp đo lường qua vi điều khiển IC 8051
- Biết được phương pháp lập trình C phục vụ cho vi điều khiển.
- Tìm hiểu được các loại cảm biến thông dụng trong đo lường.
- Xây dựng được một hệt thống đo lường cơ bản.

Mở rộng đề tài:

- Thiết kế hệ thống điều khiển, giám sát nhiệt lò công nghiệp…
- Kết hợp các thiết bị vi điều khiển có dải băm xung lớn hơn như thyristor… Và các thiết bị contactor, role để hoạt động với điện thế cao áp dụng trong công nghiệp
- Chọn cảm biến có thang đo lớn hơn
- Hiển thị nhiệt độ trên LED 7 đoạn, LED ma trận, giao diện máy tính…

*) Ưu điểm:

- Hệ thống hoạt động ổn định
- Giao diện LCD và nút điều khiển thân thiện
- Khả năng áp dụng vào thực tiễn cao

*) Nhược điểm:

- Tính toán về dây gia nhiệt và diện tích buồng kín chưa chính xác nên nhiệt tỏa ra chậm
- Thiết bị gia nhiệt hoạt động dòng 1 chiều 12V nên chỉ áp dụng với các loại máy không đòi hỏi nhiệt độ quá cao.

Do hạn chế về kiến thức, kinh nghiệm và tài liệu nên không tránh khỏi những thiếu sót. Em rất mong được thầy cô và các bạn giúp đỡ để học hỏi được nhiều hơn nữa.

Chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

1. Tống Văn On, ***Họ vi điều khiển 8051***, Nhà xuất bản Lao Động và Xã Hội.
2. Nguyễn Tăng Cường, Phan Quốc Thắng (2004), ***Cấu trúc và lập trình họ vi điều khiển 8051***, Nhà xuất bản khoa học và kỹ thuật.
3. Phạm Minh Hà (2004), ***Kỹ thuật mạch điện tử***, Nhà xuất bản khoa học và kỹ thuật.
4. Các trang web của Việt Nam các bạn có thể truy nhập:

   www.dientuvietnam.net

   www.dientuvienthong.net

   www.webdien.com

   www.tailieu.vn

MỤC LỤC