LỜI MỞ ĐẦU

Cùng với sự phát triển mọi mặt của khoa học kỹ thuật, tự động hoá trở thành một trong những ngành không thể thiếu được của nền công nghiệp hiện đại. Tự động hoá cho phép nâng cao chất lượng sản phẩm, giảm sức lao động của con người, từ đó dẫn đến giá thành sản phẩm rẻ hơn…Các bộ Vi Điều Khiển ngày càng hiện đại, tốc độ xử lý nhanh hơn, và được ứng dụng rộng rãi khắp trong các ngành công nghiệp.

Một trong những ứng dụng quan trọng của Vi Điển Khiển đó 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.

Trong ngành công nghiệp chế tạo ôtô, nhờ có tự động hoá ta có thể thay thế những nhân công làm việc tại các phân xưởng, công đoạn sản xuất có môi trường độc hại bằng máy móc, làm giảm bớt tác hại đối với người lao động. Không chỉ vậy, nhờ có dây chuyền tự động hoá mà chất lượng sản phẩm làm ra ổn định hơn, giá thành rẻ hơn…Với tầm quan trọng và sự phát triển của công nghệ ô tô nên em đã nhận đề tài " **Xây dựng mô hình điều khiển và giám sát bể sơn điện ly ô tô con**" làm đồ án tốt nghiệp của mình.

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 khoa Điện tự động công nghiệp, đặc biệt là GS.TSKH. Thân Ngọc Hoàn đã giúp đỡ em hoàn thành tốt đồ án này.

CHƯƠNG 1.

GIỚI THIỆU VỀ CÔNG NGHỆ SẢN SUẤT Ô TÔ

1.1. CÔNG NGHỆ SẢN SUẤT Ô TÔ TẠI VIỆT NAM

1.1.1. Tình hình phát triển

Hiện nay, công nghệ sản xuất ô tô tại Việt Nam đang được chú trọng quan tâm, phát triển với những nguồn lực đầu tư mạnh mẽ. Từ công tác đào tạo nghề cho đến việc đầu tư xây dựng các công ty liên doanh sản xuất ô tô với các doanh nghiệp nước ngoài đều đang được xây dựng với quy mô lớn nhằm đưa Việt Nam vào danh sách các nước sản xuất ô tô trên thế giới.

Theo Số liệu của Tổng cục Thống kê cho thấy, số các doanh nghiệp sản xuất ôtô trên lãnh thổ Việt Nam đến cuối năm 2009 là 397 doanh nghiệp; trong đó, có 50 doanh nghiệp lắp ráp ôtô, 40 doanh nghiệp sản xuất khung gầm, thân xe và thùng xe, 210 doanh nghiệp sản xuất linh kiện phụ tùng ôtô và 97 doanh nghiệp sửa chữa ôtô được rải đều trên 44 tỉnh, thành trong cả nước.

Theo công suất thiết kế, năng lực sản xuất của các doanh nghiệp sản xuất ôtô cả nước có tổng công suất sản xuất lắp ráp hiện nay là khoảng 418.000 xe/năm, trong đó số lượng sản xuất xe tải là lớn nhất với hơn 215.000 xe/năm, tiếp theo là sản xuất xe đến 9 chỗ ngồi khoảng 157.000 xe/năm. Sản xuất xe khách chỉ chiếm trên 10,5% năng lực sản xuất xe, còn xe chuyên dùng chỉ chiến trên 0,4%.

Qua các số liệu trên cho thấy hiện nay các nhà máy sản xuất và lắp ráp ôtô trong nước mới chỉ huy động khoảng 50% công suất thiết kế. Tuy nhiên giá trị sản xuất công nghiệp của ngành sản xuất ôtô năm 2009 đạt 19.956 tỷ đồng, chiếm 2,86% so với toàn ngành công nghiệp.

Một số doanh nghiệp lớn đứng hàng đầu về sản xuất lắp ráp ôtô tại Việt Nam hiện nay là Công ty Toyota Việt Nam, Tổng công ty Công nghiệp ôtô Việt Nam, Công ty cổ phần ôtô Trường Hải, Công ty liên doanh ô tô Việt Nam - DAEWOO (VIDAMCO).

Bộ Công Thương đã đề ra định hướng cho sự phát triển của công nghệ ô tô tại Việt Nam hiện nay là phải tập trung phát triển một, hai dòng xe chiến lược để giải quyết bài toán về dung lượng thị trường, phát triển công nghiệp hỗ trợ, trên cơ sở hợp tác với các hãng sản xuất xe lớn và với các nước trong AFTA để từng bước tham gia vào chuỗi sản xuất ôtô của khu vực và thế giới. Xây dựng trung tâm cơ khí ôtô quốc gia để thu hút các nhà đầu tư lớn, có ý định sản xuất ôtô lâu dài tại Việt Nam vào đầu tư nhà máy với quy mô công suất lớn, công nghệ hiện đại, cùng với các doanh nghệp sản xuất linh kiện, phụ tùng và công nghiệp hỗ trợ khác.

Khuyến khích mọi thành phần kinh tế đầu tư vào sản xuất linh kiện, phụ tùng ôtô phù hợp với chiến lược và quy hoạch, đặc biệt là sản xuất động cơ và linh kiện động cơ ôtô, khuyến khích hợp tác sản xuất và chuyển giao công nghệ với các công ty đa quốc gia, tiếp thu công nghệ sản xuất mới không lạc hậu. Tiếp thu và ứng dụng công nghệ tiên tiến, hiện đại, kết hợp với khai thác công nghệ và thiết bị hiện đại có trong nước, đảm bảo đầu tư có hiệu quả. Xây dựng nguồn nhân lực công nghiệp chuyên ngành ôtô chất lượng cao.

1.1.2. Công nghệ sản suất ô tô tại việt nam hiện nay

Việc sản xuất ôtô được thực hiện từ lắp ráp tiến dần đến chế tạo, trong việc lắp ráp cũng thực hiện từ lắp SKD tiến lên CKD1(Completely Knock Down) đến CKD2 sau đó là IKD (Incompletely Knocked Down) với việc nâng dần tỷ lệ các chi tiết, bộ phận chế tạo trong nước. Đối với xe bus, xe tải thì không lắp SKD mà thực hiện ở dạng CKD1 đến CKD2.

- Dạng CKD, CKD nhập vào: Các chi tiết được nhập vào dưới 2 dạng sau:

+ Cụm thành tổng gồm động cơ hộp số, cần chủ động, trục cardan, các cụm điện và điện tử.

+ Các chi tiết như vành, bánh, moayơ, phanh, lốp, giảm xóc… sẽ được lắp ráp tại liên doanh.

- Các chi tiết và bán thành phẩm khác sản xuất tại Việt Nam sẽ được kết hợp lắp ráp hoàn chỉnh tại công ty lắp ráp ô tô.

+ Việc lắp ráp ôtô được tiến hành theo 4 công đoạn sau:

- Hàn thân xe và vỏ xe.

- Sơn.

- Lắp hoàn chỉnh.

- Kiểm tra và hiệu chỉnh.

1.2. CÁC CÔNG ĐOẠN SẢN XUẤT Ô TÔ

1.2.1. Công đoạn hàn lắp thân, vỏ xe

Các bộ phận thân xe, vỏ khung, gầm xe đã được dập định hình sẵn theo từng loại. Xe tải, xe bus, xe du lịch được chuyển tới khu vực hàn lắp bằng xe đẩy tay. Mỗi dây chuyền lắp ráp xe bố trí một hệ thống hàn lắp thân, vỏ xe chuyên dùng.

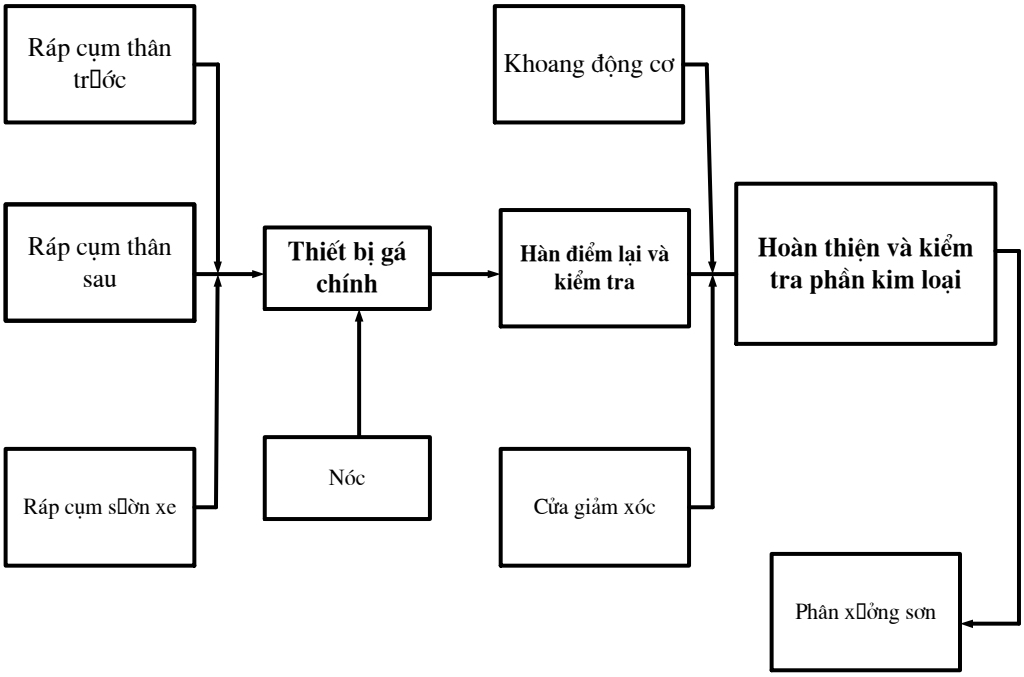
Việc định vị các bộ phận thân, vỏ xe trước khi hàn được thực hiện:

- Gầm xe, khung thân xe được ghép dựng bằng đinh tán.

- Vỏ xe được ghép dựng bằng các đồ gá hàn chuyên dụng.

Các chi tiết rời của thân xe, vỏ xe, gầm xe sau khi được định vị xong được hàn lại bằng máy hàn điểm di động. Các mối nối giữa thân xe, vỏ xe, gầm xe tuỳ từng trường hợp mà sử dụng phương pháp hàn đồ quang dưới lớp khí bảo vệ hoặc hàn hơi ôxi-axetylen.

Sau khi hàn xong toàn bộ thân, vỏ xe được kiểm tra lần cuối để sửa lại các mối hàn chưa đạt yêu cầu và làm sạch các mối hàn để chuyển sang khu vực phốt - phát hoá trước khi sơn. Công nghệ của công đoạn hàn lắp thân, vỏ xe được tóm tắt ở sơ đồ sau:



Hình 1.1: Công nghệ lắp ráp ô tô

1.2.2. Công đoạn sơn xe con

Sau khi hàn lắp xong và hoàn thiện ở phân xưởng thân xe. Thân xe mộc (hàng chưa sơn) được đưa vào bộ phận làm sạch sơ bộ. Dầu mỡ, vảy hàn, bụi bẩn được tẩy rửa bằng những dụng cụ cầm tay, giấy ráp và dung môi sau đó đưa tới phân xưởng sơn bằng xe đẩy trên đường ray.

Trước khi sơn điện ly bằng phương pháp nhúng người ta phải làm sạch bụi bẩn và tạo điều kiện bề mặt cho catốt (tức thân xe) để khi thực hiện công đoạn sơn điện ly được tốt. Thân xe đã làm sạch sơ bộ được đưa đến bộ phận tiền xử lý.

Sau khi trải qua quá trình sơn điện ly, để tạo lớp sơn ED có độ dày 25-32μm, xe được đưa vào bộ phận sấy là hệ thống lò ED OVEN gồm có hai buồng sấy. Tại đây xe được sấy trong 25 phút ở nhiệt độ 165^0C trong buồng sấy sơ bộ và ở 185^0C trong buồng sấy chính. Tiếp theo xe được đưa tới bộ phận đánh bóng và làm sạch những phần sơn không đạt yêu cầu, tại đây thân xe được trát matít, phủ PVC ở gầm và phủ lớp cách âm. Sau đó xe được đưa tới bộ phận tạo lớp sơn phủ đầu tiên. Sau khi đã làm sạch và thổi bụi, xe được

đưa vào buồng sơn phủ lớp đầu. Tại đây lớp sơn phủ được tạo ra nhờ dụng cụ sơn chuyên dụng (súng phun cầm tay).

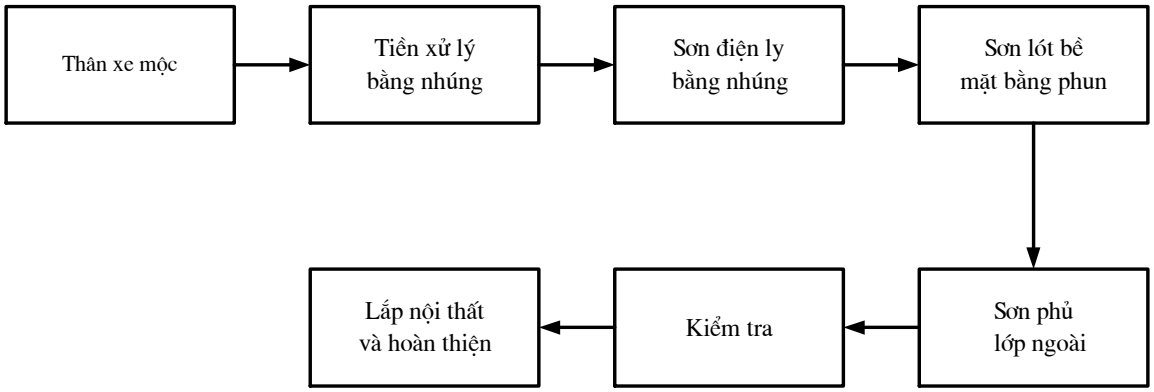
Tiếp theo, xe được đưa tới bộ phận làm sạch lần cuối trước khi đưa vào lò sấy lớp sơn phủ đầu tiên. Lò này là lò PRIMER OVEN gồm hai buồng sấy, xe được đưa tới đây và sấy ở 80^0C trong buồng sấy sơ bộ và ở 100^0C trong buồng sấy chính trong thời gian 25 phút.

Sau đó xe được đưa đến bộ phận mài ướt để đánh bóng lại lớp sơn không đạt yêu cầu của công đoạn sơn phủ lớp đầu. Tiếp theo, khi mài xong xe được đưa vào lò DRY OFF OVEN để sấy khô lớp sơn phủ đầu đã được đánh bóng bằng phương pháp mài ẩm. Tiếp đến xe được đưa vào bộ phận làm sạch bụi bẩn trước khi được đưa vào buồng sơn phủ lớp ngoài cùng. Tại đây sử dụng súng phun sơn cầm tay và các thiết bị chuyên dụng để tạo lớp sơn này.

Công đoạn này được thực hiện xong, thân xe được đưa vào bộ phận làm sạch lớp sơn phủ ngoài không đạt yêu cầu để đưa vào lò sấy TOP OVEN. Khi lớp sơn TOP COAT BOOT được làm sạch xong, xe được đưa tới lò TOP OVEN và được sấy trong vòng 33 phút ở nhiệt độ 110^0C trong buồng sấy sơ bộ, ở 130^0C trong buồng sấy chính. Khi ra khỏi lò này, xe đã được phủ một lớp sơn dày 40÷50μm.

Tiếp theo, xe được đưa đến bộ phận kiểm tra xem có đạt yêu cầu không, nếu đạt yêu cầu thì cho xe ra và chuyển tiếp đến phân xưởng lắp ráp nội thất và hoàn thiện, nếu không đạt yêu cầu thì đem vào bộ phận sửa chữa.

Sơ đồ công nghệ của công đoạn sơn xe ô tô con được trình bày ở hình dưới:



Hình 1.2: Các công đoạn sơn xe ô tô

1.2.3. Công đoạn lắp ráp và hoàn thiện

Công nghệ lắp ráp xe du lịch (xe con) ở giai đoạn SKD, ở giai đoạn này được nhập về ở tình trạng đã làm xong kể cả sơn. Khung chassis khi nhập về đã được lắp hoàn chỉnh. Đông cơ và hệ thống truyền động được gắn liền với nhau, trục đã được lắp sẵn với các cơ cấu liên quan, bánh xe, xăm lốp đã được lắp sẵn.

Các bộ phân bên trong: Ghế, đệm lót, v.v… đều được lắp trước vào thân xe, ống dây nối, ống mềm… đã được lắp tối đa vào khung. Do đó việc lắp ráp các cụm SKD hoàn chỉnh lại với nhau thành xe ôtô hoàn chỉnh chỉ còn là việc lắp ráp các ốc vít. Công việc này được tiến hành bằng tay và bằng dụng cụ vạn năng, ở giai đoạn này nếu cần chỉ sửa chữa mà thôi.

Công đoạn lắp hoàn chỉnh xe con ở giai đoạn SKD: Phần vỏ thân xe sau khi sơn phủ lớp cuối cùng sẽ được chuyển tới bộ phận lắp ráp hoàn chỉnh. Tại đây, việc lắp ráp các bộ phận bên trong thân xe sẽ được tiến hành.

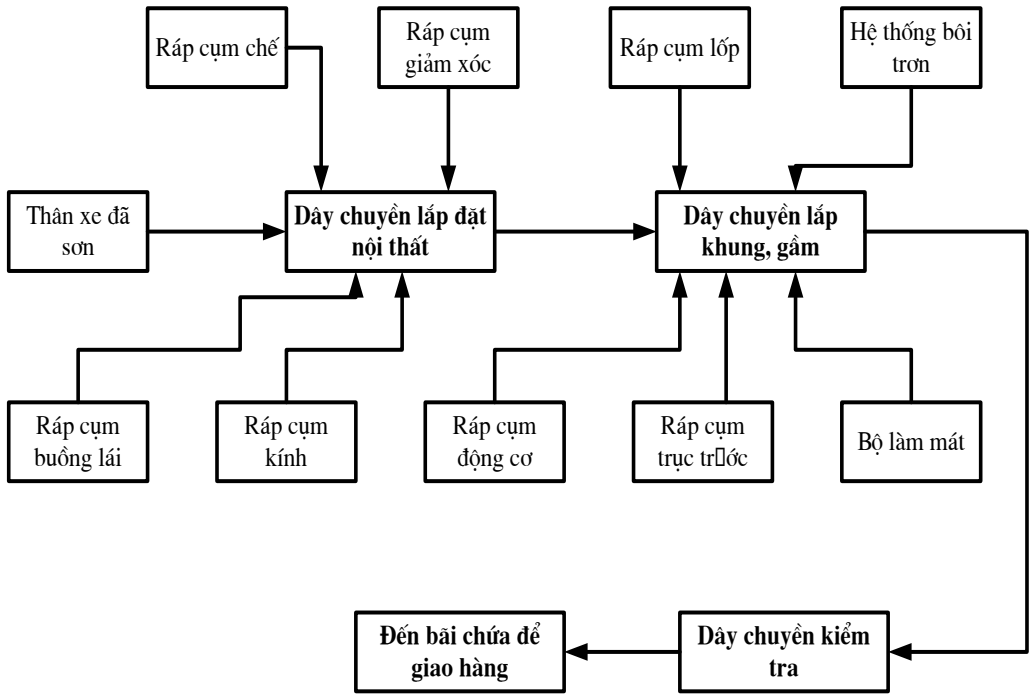
- Lắp ráp các bộ phận chính và các bộ phận phụ của khung chassis.
- Lắp động cơ và hệ thống truyền động.
- Lắp ổ trục và tay phanh vào ổ giữa, trục vi sai.
- Lắp buồng lái : đồng bộ bảng điều khiển, lắp cửa, lắp các bộ phận bên trong như ghế, đệm lót và các bộ phận trang trí.

- Chuyển thân xe đã được lắp ráp hoàn chỉnh các bộ phận bên trong tới bộ phận ghép thân vào khung chassis. Khung chassis được lắp ráp trước, thân được đặt trên khung chassis và tiến hành lắp thân vào khung chassis, sau đó tiếp tục lắp bánh xe.

Trong giai đoạn này sử dụng các dụng cụ lắp ráp vạn năng và chuyên dụng, các tuốc-nơ-vít khí nén.

Việc lắp ráp được tiến hành trên băng chuyền và các thiết bị nâng hạ bằng môônray.

Các công nghệ của công đoạn lắp ráp nội thất và hoàn thiện xe con được tóm tắt theo sơ đồ sau:



Hình 1.3: Các công đoạn lắp ráp hoàn thiện cho xe

1.2.4. Công đoạn kiểm tra

Khi ra khỏi phân xưởng lắp ráp nội thất và hoàn thiện, xe được đưa tới phân xưởng kiểm tra trước khi xuất xưởng và đưa ra bãi chứa để giao hàng. Công đoạn này xe được kiểm tra các công đoạn sau:

- Kiểm tra độ trượt dốc/ phanh/ tốc độ (A.B.S)
- Kiểm tra đèn phía trước.
- Kiểm tra khói.
- Kiểm tra độ kín gas.
- Kiểm tra bán kính quay.
- Kiểm tra độ ổn định.
- Kiểm tra độ lọt nước (tiêu chuẩn là 100%).

1.2.5. Sản phẩm

Sản phẩm chính của công ty liên doanh tại Việt Nam gồm có:

- Xe bus: BG – 150; BS – 090; BS – 106.
- Xe du lịch cỡ nhỏ (xe con): Matiz, Lanos, Nubira, Lengara, Maguz, Lancetti.
- Xe tải cỡ nhỏ

CHƯƠNG 2.

CÔNG NGHỆ SƠN ĐIỆN LY

2.1. QUÁ TRÌNH HÌNH THÀNH VÀ PHÁT TRIỂN CỦA SƠN ĐIỆN LY

2.1.1. Lịch sử của sơn điện ly

Những nghiên cứu phát triển của sơn điện ly được hãng Ford Motor bắt đầu từ năm 1957 dưới sự lãnh đạo của Tiến sĩ George Brewer. Mục đích của những nghiên cứu này là để tìm ra 1 phương pháp chống ăn mòn tốt nhất cho các chi tiết, bộ phận của thân xe ôtô.

Các nhà chế tạo ôtô đã nhận thức rõ ràng rằng quá trình rỉ sét xảy ra bên trong sẽ dần dần phá hỏng các cấu kiện của khung xe. Mặc dù lớp sơn thông thường đã có thể thâm nhập vào tận cùng các hốc của khung xe nhưng chúng lại thường bị tẩy bởi hơi của dung môi trong khi sấy sơn. Vì vậy, nhóm của Tiến sĩ Brewer đã cố gắng tạo nên 1 lớp sơn mà dung môi không thể tẩy chúng được trong suốt quá trình. Những công việc này dẫn đến sự phát triển của sơn điện ly.

Bể sơn đầu tiên của hãng Ford hoạt động vào 4/7/1961 dùng để sơn Lagiăng của bánh xe. Bể sơn nhúng cho thân xe được lắp đặt vào năm 1963. Cả 2 bể này đều sử dụng kiểu kết tủa dương cực.

Mặc dù thị trường của sơn điện ly sau khi ra đời phát triển một cách vững chắc, nhưng cho đến tận năm 1973, sơn điện ly kiểu kết tủa âm cực ra đời, thị trường mới thực sự bùng nổ. Vào năm 1965, chỉ có 1/100 xe được sơn lót bằng sơn điện ly, đến năm 1970, đã có 10/100 xe và đến nay, hầu hết các xe đều dược sơn lót bằng phương pháp sơn điện ly.

2.1.2. Ưu nhược điểm của sơn điện ly

- Tạo màng bảo vệ để chống rỉ sét tại tất cả các hốc, các vùng bên trong thân xe.

- Hiệu quả sử dụng sơn cao, lên đến 95%. Giảm thiểu lượng sơn thất thoát, đặc biệt nếu đem so sánh với phương pháp sơn phun.

- Việc sử dụng nước trong quá trình sơn đã gần như loại trừ được hệ thống cứu hoả, hệ thống cấp khí nén và gảm được chi phí cho thiết bị, quản lý và vận hành các hệ thống này.

- Do độ nhớt của bể sơn thấp(Ngang bằng với nước) cho nên dễ dàng cho việc bơm và xả trong quá trình sơn.

- Do lớp sơn mới không hoà tan trong nước nên cho phép rửa và thu hồi được cặn sơn.

- Sơn chưa sấy đủ khô để có thể sờ tay được, dễ dàng cho các thao tác bằng tay.

- Khác với sơn bằng phương pháp phun, sơn điện ly không bị chảy trong khi sấy.

- Khác với sơn phun, sơn điện ly không bị tẩy bởi hơi dung môi trong khi sấy.

- Lớp kết tủa được sinh ra một cách liên tục từ phần này đến phần kia.

- Từ khi quá trình là tự động hoá, nhân công lao động trực tiếp giảm rõ rệt.

2.2. CÔNG NGHỆ XỬ LÝ TRƯỚC VÀ SƠN ĐIỆN LY

2.2.1. Xử lý trước

Để tạo điều kiện tốt nhất cho quá trình sơn điện ly, thân xe ô tô phải được trải qua 1 quá trình xử lý trước khi đưa vào sơn.

Hệ thống này gồm 6 bể xử lý với các chức năng cụ thể sau:

* Tẩy dầu mỡ(Degreasing)

Đầu tiên, thân xe từ phân xưởng hàn chuyển đến được lau kỹ bằng dầu hoả. Mục đích của công việc này là để tẩy sạch các lớp bụi kim loại, vảy hàn hoặc keo còn dính trên thân xe.

Sau đó xe được đưa vào nhúng chìm trong bể tẩy đầu mỡ (TK-101) chứa dung dịch kiềm nóng ở 50-60^{0}C. Dưới tác dụng của dòng dung dịch được tạo ra bởi bơm tuần hoàn với áp suất 2 Bar, lưu lượng 120 m^3/h và hoạt chất hoá học của dung dịch kiềm nóng, thân xe được rửa sạch sẽ khỏi các tạp chất bám vào từ các công đoạn sản suất trước như dầu mỡ, bụi bẩn...

Thông số kỹ thuật của bể tẩy dầu mỡ:

- Nhiệt độ làm việc: 50-60^{0}C.
- áp lực bơm tuần hoàn: 2 Bar.
- Lưu lượng bơm: 120 m^3/h
- Độ kiềm tự do: 12-17
- Thể tích dung dịch: 48 m^3
- Thời gian nhúng xe: 3 phút

* Rửa nước thường(Water Rinse):

Thân xe sau khi qua bể tẩy dầu mỡ được đưa vào nhúng chìm trong bể nước sạch(TK-102), dưới tác dụng của các vòi phun và dòng nước tuần hoàn, dung dịch kiềm bám trên xe sẽ được rửa sạch.

Thông số kỹ thuật của bể rửa nước thường:

- áp lực bơm tuần hoàn: 2 Bar.
- Lưu lượng bơm: 78 m^3/h
- Độ pH: 6 – 8
- Thể tích nước: 48 m^3
- Thời gian nhúng xe: 30 s

* Tạo điều kiện bề mặt(Surface Conditioning):

Từ bể rửa TK-102, thân xe được đưa dến nhúng chìm vào bể chứa dung dịch tạo điều kiện bề mặt TK-103. Tại đây, dưới tác dụng của hoá chất, thân xe sẽ sẵn sàng cho quá trình phốt phát hoá tiếp theo.

Thông số kỹ thuật của bể tạo điều kiện bề mặt:

- áp lực bơm tuần hoàn: 2 Bar.
- Lưu lượng bơm: 78 m^3/h
- Độ pH: 8 – 9
- Độ kiềm tổng: 3 – 4.5
- Thể tích dung dịch: 48 m^3
- Thời gian nhúng xe: 30 s

* Phốt phát hoá bề mặt (Phosphating):

Sau khi qua bể tạo điều kiện bề mặt, thân xe đưa tới nhúng chìm trong bể chứa dung dịch phốt phát (TK-104). Quá trình này nhằm mục đích tạo lớp nền để sơn điện ly dễ dàng bám chặt trên bề mặt kim loại.

Thông số kỹ thuật :

- áp lực bơm tuần hoàn P-104: 2 Bar.
- Lưu lượng bơm P-104: 120 m^3/h
- áp lực bơm P-144: 1 bar
- Độ axít tổng: 17-23
- Độ axít tự do: 0.6-0.9
- Hoạt chất: 1.5-2.5
- Thể tích dung dịch: 48 m^3
- Nhiệt độ dung dịch: 40-45°C
- Nhiệt độ nước nóng: 80-90°C
- Thời gian nhúng xe: 180 s

* Rửa nước thường và rửa nước khử ion(Water Rinse & DI Water Rinse):

Sau khi phốt phát hoá bề mặt thân xe, trước khi vào sơn điện ly, xe phải qua 2 công đoạn rửa là rửa bằng nước sạch và nước khử Ion. Mục đích để làm sạch các hoá chất còn bán trên xe, tạo điều kiện tốt nhất cho quá trình sơn điện ly và không làm ảnh hưởng đến chất lượng của bể sơn.

Về nguyên tắc hoạt động của 2 bể này hoàn toàn giống với bể TK-102, chỉ khác thông số kỹ thuật:

Rửa nước thường (TK-105):

- áp lực bơm tuần hoàn: 2 Bar.
- Lưu lượng bơm: 78 m^3/h
- Độ pH: 6 – 8
- Thể tích nước: 48 m^3
- Thời gian nhúng xe: 30 s

Rửa nước khử Ion (TK-106):

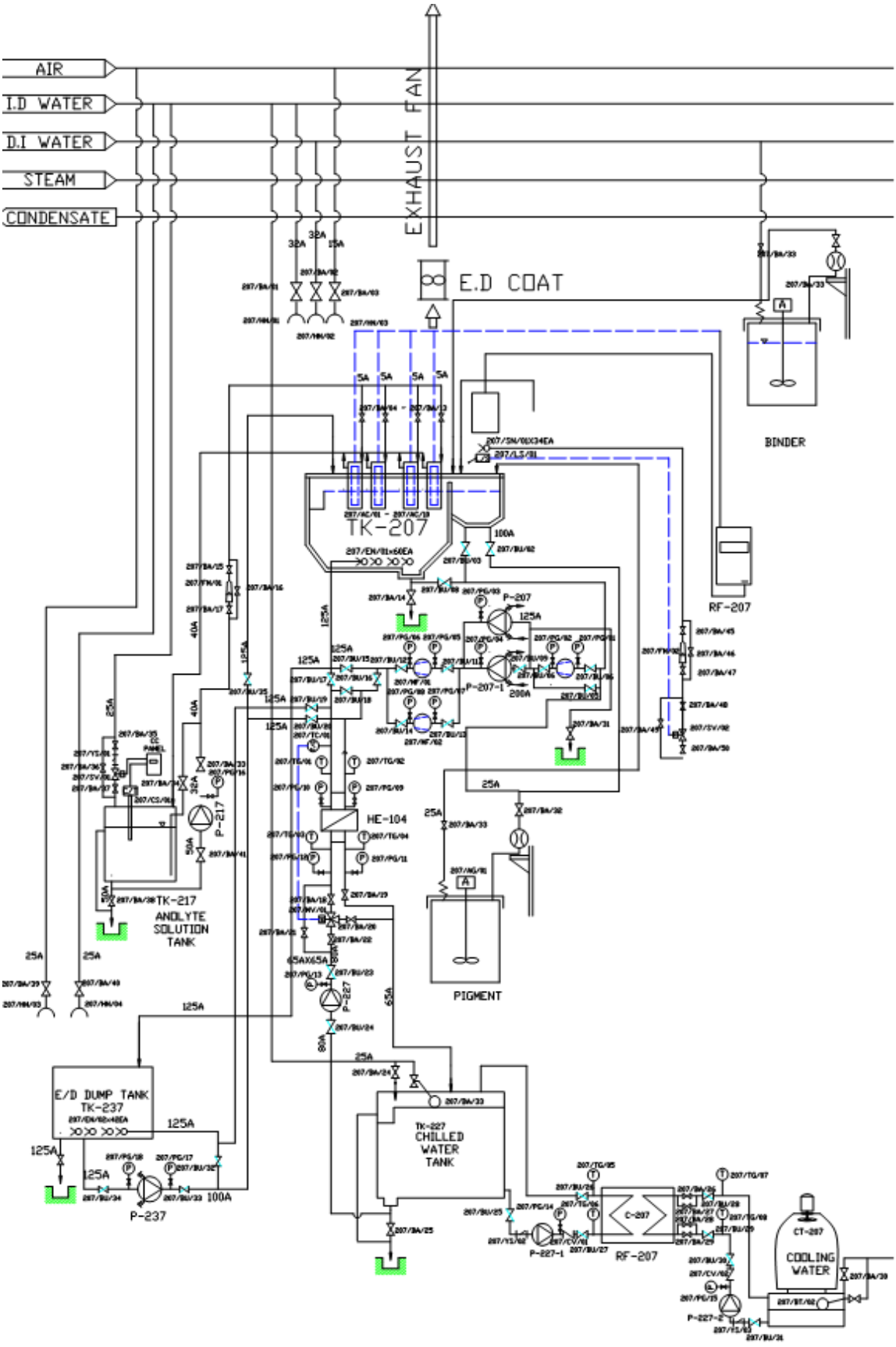
- áp lực bơm tuần hoàn: 2 Bar.
- Lưu lượng bơm: 78 m^3/h
- Độ dẫn điện max: 50 mS
- Thể tích nước: 48 m^3
- Thời gian nhúng xe: 30 s

2.2.2. Sơn điện ly(Electro Deposision)

Đây là quá trình quan trọng nhất trong công nghệ sơn Ôtô. Để hoàn tất quá trình này, thân xe phải trải qua 3 công đoạn:

***** Sơn điện ly

Thân xe sau khi qua quá trình xử lý trước được nhúng chìm trong bể chứa dung dịch sơn điện ly. Dưới tác dụng của dòng điện điện 1 chiều sẽ hình thành 1 lớp sơn bám đều trên bề mặt kim loại của xe.



Hình 2.1: Sơ đồ công nghệ của bể sơn điện ly

Cũng giống như các bể của hệ thống xử lý trước, dung dich sơn được bơm liên tục bằng 2 bơm ly tâm P-207 và P-207-1. Các bơm này hút sơn từ đáy bể chính và ngăn phụ qua phin lọc thô, sau đó sơn được bơm qua 2 phin lọc tinh, qua bộ trao đổi nhiệt đến các vòi phun đẻ tạo dòng chảy tuần hoàn trong bể.

- Nếu độ chênh áp giữa đầu vào và đầu ra của các bộ lọc vượt quá 0.5 bar thì phải vệ sinh hoặc thay thế các bộ lọc này.

- Trên bề mặt bể, bố trí các vòi phun dung dịch nước khử Ion và BC để rửa dung dịch sơn bám vào xe sau khi nhúng.

- Các thông số như độ pH, độ dẫn điện, hàm lượng Solid, Binder của bể sơn được đo và phân tích hàng ngày để điều chỉnh bằng hoá chất cho phù hợp tiêu chuẩn.

- Để dễ dàng cho việc bảo dưỡng và sửa chữa bể, người ta lắp đặt 1 bể chứa phụ (TK-237). Khi bảo dưỡng bể chính thì bơm toàn bộ dung dịch sang bể phụ bằng cách khoá van 207/BU/16 và mở van 207/BU/15. Sau khi sửa chữa bảo dưỡng xong, dùng bơm P-237 bơm dung dịch sơn trở lại bể.

Thông số kĩ thuật bể sơn ED:

- Áp lực bơm tuần hoàn P-207, P-207-1	3 Bar
- Lưu lượng bơm P-207, P-207-1	132 m^3/hour
- Nhiệt độ dung dịch sơn ED	28- 35^{0}C
- Thể tích bể sơn ED	48m^3
- Solid	20 – 23 WT%
- Binder	5,0 – 6,2
- Độ pH	5,9 – 6,2
- Độ dẫn điện	1250-1650ms

* Hệ thống dương cực (Anolyte Solution system)

Dương cực của bể sơn ED được chế tạo đặc biệt. Chúng gồm những bản cực hình chữ nhật bên trong chứa đầy dung dịch Anolyte. Một lớp vi màng mỏng ngăn không cho dung dịch Anolyte thấm ra ngoài bể sơn ED nhưng không ngăn các Cation chạy vào từ bể sơn khi có dòng điện 1 chiều chạy qua.

Dung dịch Anolyte chứa trong bể TK-217 được bơm tuần hoàn đến các bản cực bằng bơm P-217. Mức của bể TK-217 được điều chỉnh bằng hệ thống van tự động 207/SV/01. Các thông số như độ pH, dẫn điện của dung dịch Anolyte được đo hàng ngày và xử lý bằng hoá chất. Để tăng hiệu quả của và điều chỉnh chế độ của dương cực, sử dụng hệ thống kiểm soát lưu lượng dung dịch Anolyte 207/FM/01.

Thông số của hệ thống Anolyte:

- Độ PH	2,8 – 3,5
- Độ dẫn điện	4000-7000 ms
- Lưu lượng bơm P – 217	3 m^3/hour
- Áp lực bơm P – 217	1,2 Bar

* Hệ thống điều chỉnh điện áp và chỉnh lưu(IVR và Reetifiev).

Để cung cấp nguồn một chiều với dòng điện lớn(1000A, 380V) cho quá trình sơn ED, người ta lắp đặt hệ thống ổn định điện áp(IVR) và chỉnh lưu có điều khiển(Reetifiev).

- Ổn định điện áp: Đây là một máy biến áp tự ngẫu tự động điều chỉnh điện áp khi tăng tải để đảm bảo điện áp cấp cho chỉnh lưu là 380-400V.

- Chỉnh lưu: Đây là một bộ chỉnh lưu cầu 3 pha công suất lớn (450KVA) tự động điều chỉnh dòng điều khiển tăng dần từ 0-1000A dưới điện áp 380V.

Thông số của hệ thống ổn định điện áp và chỉnh lưu:

- Công suất 450KV
- Dòng điện một chiều max 1000A
- Điện áp làm việc 380-400V

* Bể rửa và thu hồi sơn (TK-208)

Trên thân xe sau khi được sơn ED còn có rất nhiều sơn ED không kết tủa dính ở bên ngoài. Để rửa sạch và thu hồi phần sơn này, người ta nhúng xe vào bể rửa UF(Ultra Filter).

- Độ PH 5,5-6,2
- Độ dẫn điện 800-1300 mS
- áp lực bơm P – 208 1,2 Bar
- Lưu lượng bơm 1200m^3/h
- Thể tích bể 48 m^3
- Thời gian nhúng xe 30s

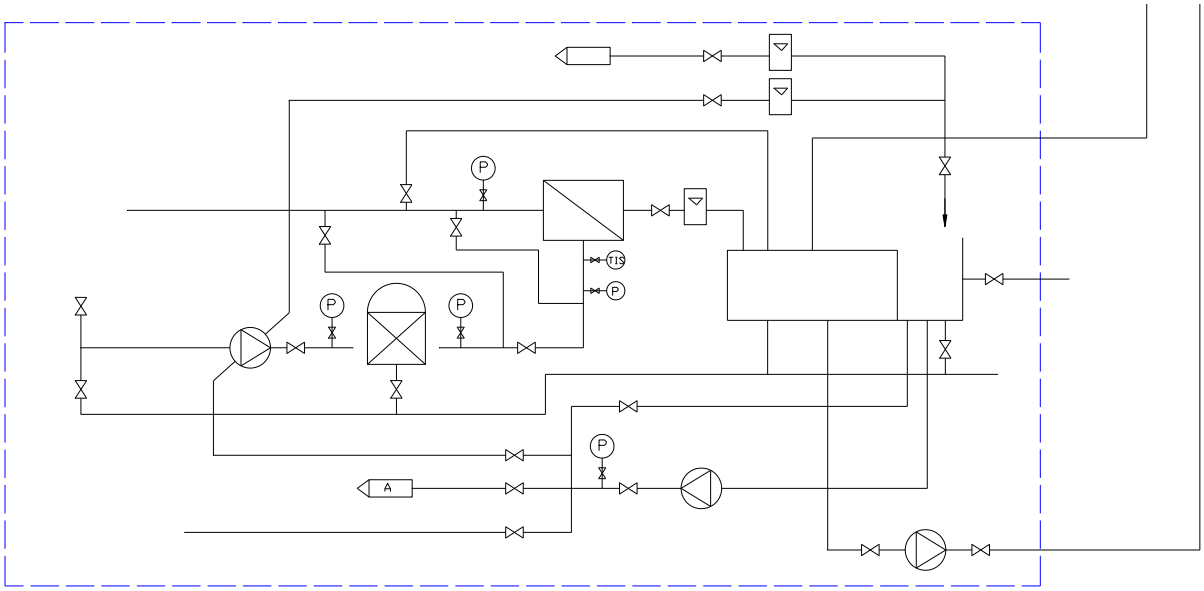
* Hệ thống thu hồi sơn (Paint Recover System, UF-208).

Hệ thống này có tác dụng lọc lại dung dịch sơn trong bể ED một lần nữa và cấp nước làm mát cho bạc bơm tuần hoàn sơn ED. Sơn từ bể chính ED được bơm tuần hoàn M1 bơm qua bộ lọc tinh (Micro Filter, MF) đến bộ tách nước (Ultra Filter, UF).

Đây là một hệ thống gồm 6 cột lọc bao gồm các vi màng chỉ cho phép các phần tử nước thấm qua còn các phần tử khác nếu có phân tử lượng lớn hơn phân tử lượng của nước sẽ không đi qua được. Phần dung dịch sơn sau khi đã tách bớt nước sẽ quay trở lại bể sơn. Phần nước được tách ra khỏi sơn được đưa vào 2 bể T1 (Permeate water tank) và T2 (Seal water tank) rồi chảy về đầu bể TK - 208. Từ cuối bể UF, có một đường ống nối thông với bể ED để dẫn phần dung dịch UF tràn đi. Dung dịch trong bể UF liên tục được nước tách từ sơn ED bổ sung làm loãng ra, phần tràn đi sẽ trộn vào bể sơn và lại được tách nước.

Nước được tách ra từ dung dịch sơn ED còn được bơm tuần hoàn M2 bơm đi làm mát các ổ bạc (Mechanical Seal) của bơm tuần hoàn sơn ED. Một phần khác sẽ được bơm đến các vòi phun rửa xe trực tiếp tại bể TK- 208 bằng bơm M3.

Bộ lọc tinh MF sẽ được thay thế nếu độ chênh áp suất giữa đầu vào và đầu ra vượt quá 0,5 bar.



Hình 2.2: Sơ đồ công nghệ của hệ thông thu hồi sơn

Để kiểm soát khả năng tách nước của UF, người ta đặt 1 lưu lượng kế (Flow meter, FM1) để kiểm soát lưu lượng nước tách được. Nếu lưu lượng nước tách từ sơn ED nhỏ hơn 10l/phút thì phải chạy hệ thống ở chế độ thông rửa bộ tách nước UF.

Để kiểm soát lưu lượng nước làm mát các bạc bơm người ta cũng đặt 2 lưu lượng kế FM2 và FM3, lưu lượng nước làm mát bạc bơm qua FM2 hoặc FM3 khoảng 5l/phút.

Nếu mức nước trong bể T2 (Seal water tank) giảm đột ngột hệ thống sẽ cảnh báo sự cố tại các bạc của các bơm tuần hoàn.

Thông số kỹ thuật:

- Nhiệt độ sơn ED max 40^0C
- áp lực bơm M1 2- 3,8 Bar
- Lưu lượng bơm M1 $24m^3/h$
- Lưu lượng nước tách ra từ UF 10-15 l/phút
- Lưu lượng nước làm mát bạc bơm 3-5 l/phút

* Rửa nước khử Ion(DI Water Rinsse).

Sau khi qua bể rửa UF, thân xe được rửa lại một lần cuối cùng bằng nước DI trước khi vào lò sấy.

Thông số kĩ thuật:

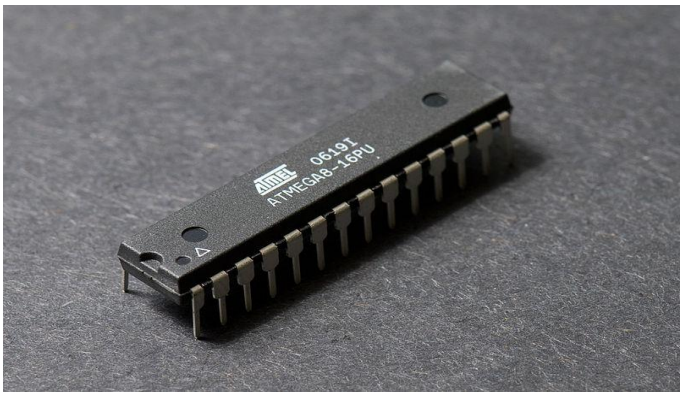
- Độ PH 6 – 8
- Độ dẫn điện 0,4 – 7 ms
- Áp suất bơm P-209 1,2 Bar
- Lưu lượng bơm P-209 $130m^3/h$
- Thể tích bơm $48m^3$
- Thời gian nhúng xe 30s

2.2.3. Sấy sơn ED

Thân xe sau khi sơn xong sẽ được đưa vào lò sấy và được sấy với nhiệt độ lên đến 185^{o}C. Dưới tác dụng của luồng gió nóng, lớp sơn ED trên bề mặt kim loại sẽ khô đi và bám vững chắc vào xe. Đến đây mới chính thức kết thúc quá trình sơn điện ly của thân xe ôtô.

2.3. KHÁI QUÁT ATMEL AVR

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.



Hình 2.3: Atmel AVR ATmega8

2.3.1. 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 hệt nhau.

2.3.2. 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

- 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.

2.3.3. 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).

2.3.4. 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.

2.3.5. 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.

2.3.6. 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.

2.3.7. 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.

2.3.8. 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)

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

+ 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.

+ 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.

+ 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.

+ 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.

+ 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

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

+ Vào capture

• So sánh Analog

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

+ Với 12-bit D / A Converters

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

+ I²C tương thích Two-Wire Interface (TWI)

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

+ Thiết bị giao diện Serial Bus (SPI)

+ 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

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

+ 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

2.4. NGÔN NGỮ LẬP TRÌNH C

2.4.1. Các kiểu toán tử của C

Toán tử gán (=) và 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

2.4.2. 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 dùng 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 địn 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

Atmega8.h và atmega16.h

2.4.3. 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.

2.4.4. 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 thực 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 ðýợc bỏ qua.
3) Lệnh được thực hiện. Nó có thể là một lệnh ðõn hoặc là một khối lệnh được bao trong một cặp ngoặc nhọn.
4) Cuối cùng, thực hiện ðể tãng biến ðiều 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.

2.4.5. Cấu trúc cơ bản của của một chương trình C

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.

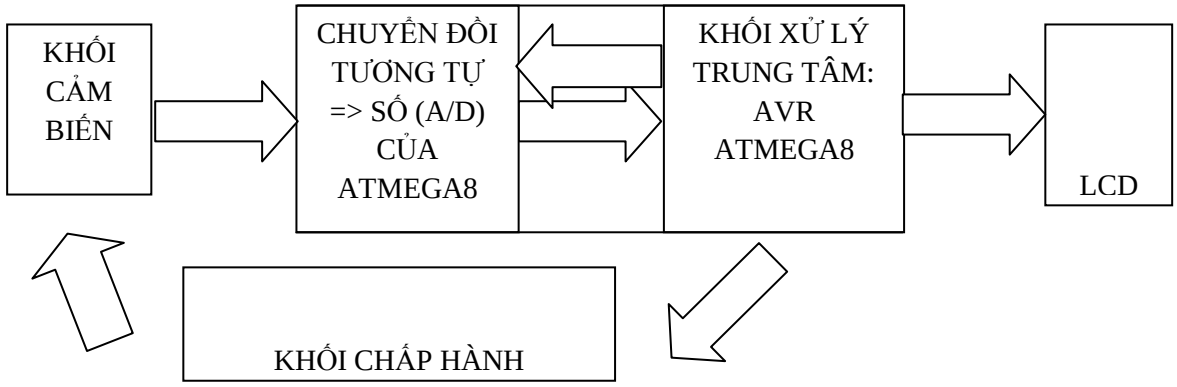
XÂY DỰNG HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT BỂ SƠN ĐIỆN LY Ô TÔ CON

3.1. HỆ THỐNG ĐIỀU KHIỂN VÀ GIÁM SÁT NHIỆT ĐỘ

3.1.1. Nguyên tắc 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 chân ADC của VDK ATmega8, hiển thị dùng LCD. Tín hiệu tương tự từ chân 2 của LM335 được đưa vào chân 23 của ATmega8 để chuyển thành tín hiệu số. Tín hiệu số từ ADC được đưa vào khối xử lý trung tâm của VDK để so sánh với nhiệt độ đặt thực tế. 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 PB1 củaVDK, Tín hiệu ra trên PB2 của VDK qua transitor Q2 để điều khiển IRFZ44N mosfet. Khi mosfet đóng lại thì hệ thống ngừng gia nhiệt, khi mosfet mở hệ thống lại tiếp tục gia nhiệt. Do đó nhiệt độ được giữ ổn định tại giá trị đặt.

3.1.2. SƠ ĐỒ TỔNG QUÁT



Hình 3.1: Sơ đồ khối

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.

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.

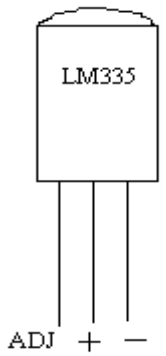
4) Khối hiển thị: Dùng để hiện thị được đưa ở đầu ra của khối xử lý.

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

3.1.3. 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 3.2: 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 -40–100^0C
- Tiêu tán công suất thấp.
- Dòng điện làm việc từ 4.10^{-4}A đến 5.10^{-3}A.

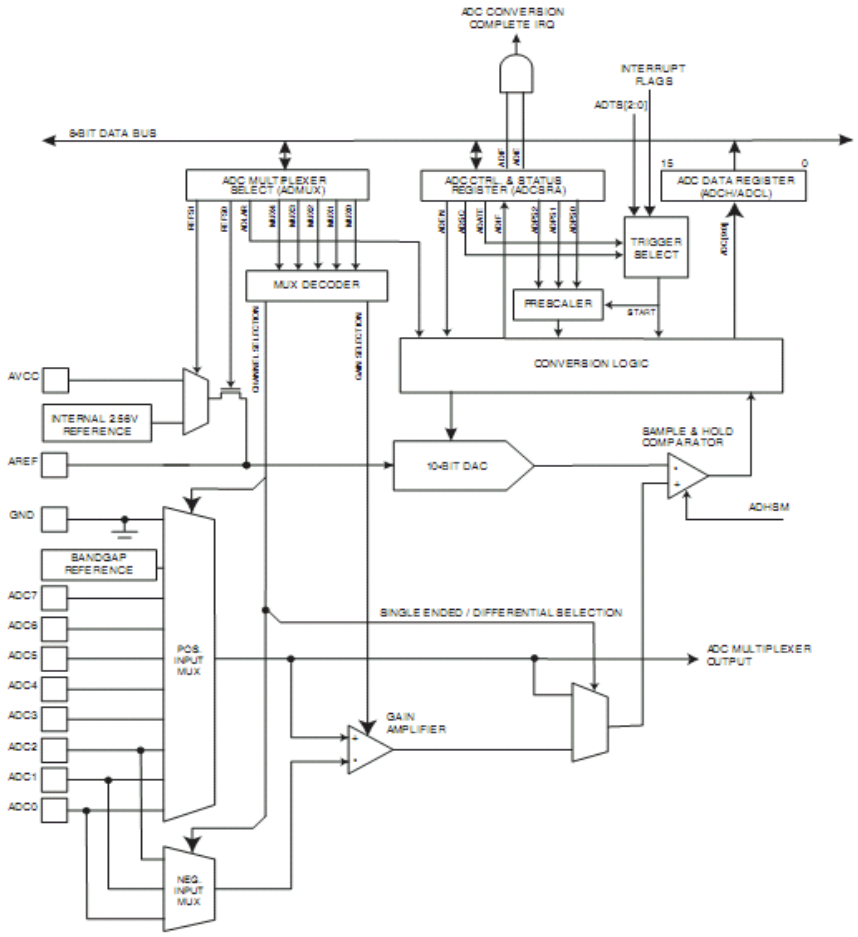
- Dòng ngược 15mA.
- Dòng thuận 10mA.

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

3.1.4. Khối chuyển đổi tương tự sang số

Vi điều khiểnAtmega8 có một bộ biến đổi ADC tích hợp trong chip với các đặc điểm:

- Độ phân giải 10 bit
- Sai số tuyến tính: 0.5LSB
- Độ chính xác +/-2LSB
- Thời gian chuyển đổi:65-260µs
- 6 Kênh đầu vào có thể được lựa chọn
- Có hai chế độ chuyển đổi free running và single conversion
- Có nguồn báo ngắt khi hoàn thành chuyển đổi
- Loại bỏ nhiễu trong chế độ ngủ



Hình 3.3: Sơ đồ bộ biến đổi

Tám đầu vào của ADC là tám chân của PORTA và chúng được chọn thông qua một MUX.

Để điều khiển hoạt động vào ra dữ liệu của ADC và CPU chúng ta có 3 thanh ghi: ADMUX là thanh ghi điều khiển lựa chọn kênh đầu vào cho ADC, ADCSRA là thanh ghi điều khiển và thanh ghi trạng thái của ADC, ADCH và ADCL là 2 thanh ghi dữ liệu.

3.1.4.2. Nguyên tắc hoạt động và lập trình điều khiển

ADC có nhiệm vụ chuyển đổi tín hiệu điện áp tương tự thành tín hiệu số có độ phân giải 10 bit.Với giá trị nhỏ nhất của điện áp đặt ở chân AGND và giá trị cực đại của điện áp tương tự được mắc vào chân AREF. Tám kênh tương tự đầu vào được chọn lựa thông qua ADMUX và ADMUX này được điều khiển bởi thanh ghi ADMUX.

ADC này có thể hoạt động được ở hai chế độ. Đó là chuyển đổi đơn: chỉ chuyển đổi một lần khi có lệnh chuyển đổi và chế độ tự chuyển đổi (Free running mode) đây là chế độ mà ADC tự động chuyển đổi khi được hoạt động và công việc chuyển đổi có tính tuần hoàn (chỉ cần khởi động một lần).

ADC được phép hoạt động nhờ thiết lập bit ADEN. Quá trình chuyển đổi được bắt đầu bằng việc ghi vào bit ADSC mức logic 1 và trong suốt quá trình chuyển đổi bit này luôn được giữ ở mức cao. Khi quá trình chuyển đổi hoàn thành thì bit này được xóa bằng phần cứng và cờ AIDF được bật lên.

Dữ liệu sau khi chuyển đổi được đưa ra thanh ghi dữ liệu ADCL và ADCH, nhưng chú ý khi đọc dữ liệu từ hai thanh ghi này thì đọc ADCL trước rồi mới đọc ADCH. Nếu đọc ADCH trước thì dữ liệu cập nhật có thể ghi đè lên ADCL (Vi điều khiển nghĩ rằng đã đọc xong dữ liệu).

Để điều khiển vào ra dữ liệu với ADC, các bước thực hiện như sau:

Bước 1: Định nghĩa các cổng vào cho tín hiệu tương tự
Xóa bit tương ứng với chân đó trong thanh ghi DDRA. Sau đó loại bỏ điện trở treo bằng cách xóa bit tương ứng ở thanh ghi PORTA.

Bước 2: Chọn kênh tương tự vào (chọn chân vào cho ADC) thông qua thanh ghi ADMUX (có thể thay đổi trong quá trình hoạt động).

Bước 3: Thiết lập các thông số cho ADC

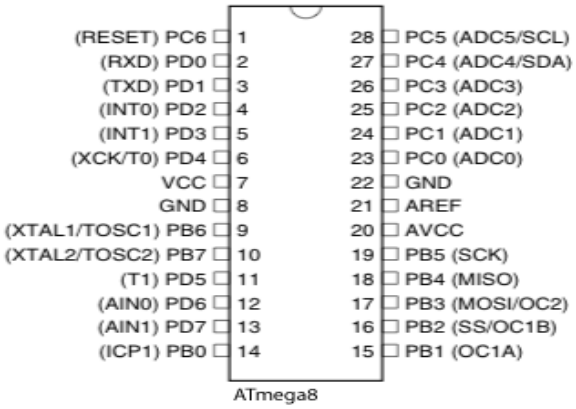
Tốc độ chuyển đổi thông qua xung nhip chuyển đổi.

Chế độ chuyển đổi : đơn hoặc tự động.

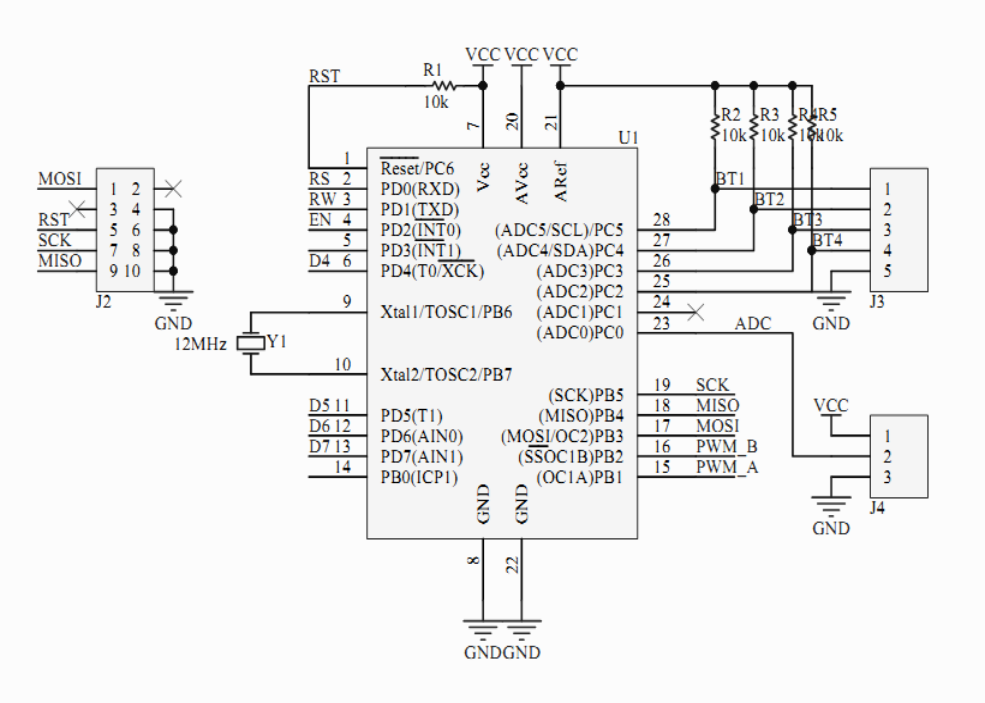
Sử dụng ngắt hoặc không.

Bước 4: Bắt đầu chuyển đổi và đọc dữ liệu.

3.1.5. Khối xử lý trung tâm



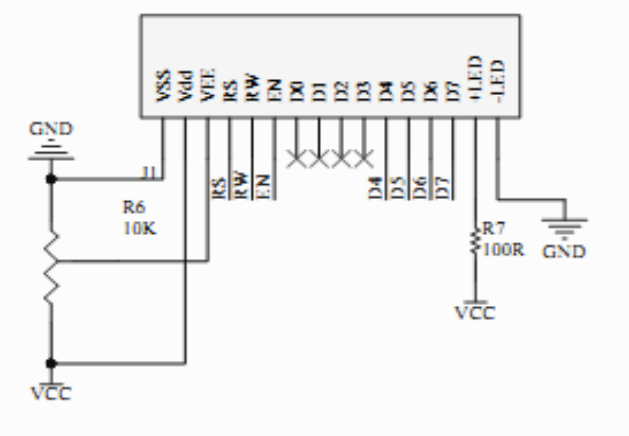
Hình 3.4a: Các chân của ATmega8



Hình 3.4b: ATmega8

3.1.6. 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.



Hình 3.5: Hiển thị LCD

3.1.6.1. Cấu tạo LCD

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)

3.1.6.2. 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.

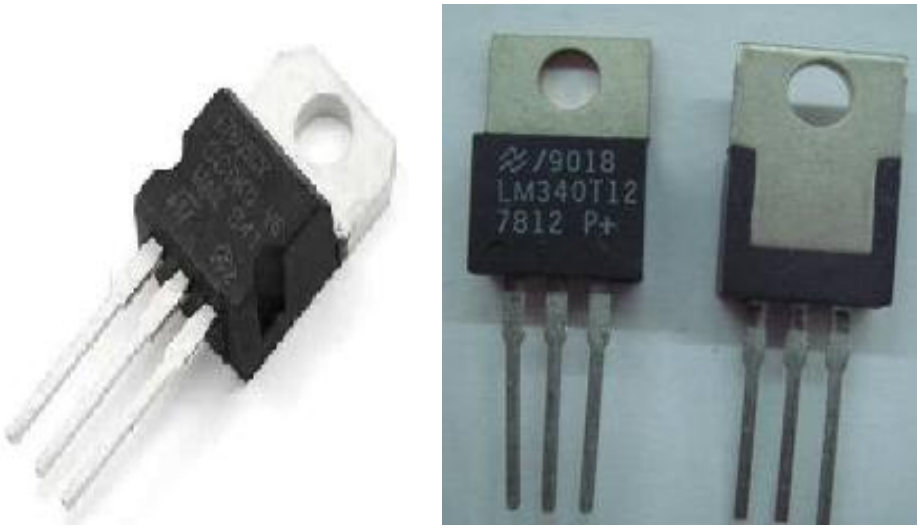
3.1.7. Khối nguồn

3.1.7.1. Biến áp cấp nguồn

Hình 3.6: Biến áp 5A

- Biến áp 5A: Làm nhiệm vụ biến đổi điện áp 220V-50Hz thành điện áp 12V-50Hz.
- Nguồn cấp vào biến áp: 220 VAC
- Nguồn ra 6V, 9V, 12V, 15V, 18V, 24 VAC
- Dòng định mức: 5A

3.1.7.2. IC ổn áp nguồn



Hình 3.7: LM 7805 và LM 7812

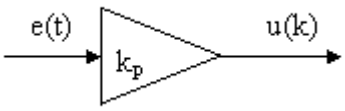
Bộ nguồn nhằm cung cấp điện áp một chiều +5V,+12V,ổn định cho mạch điện. Để tạo được nguồn theo yêu cầu em sử dụng 2 IC ổn áp 7805 và 7812 để tạo ra điện áp ổn định 5 V và 12 V.

Bộ nguồn phải có tính chống nhiễu tốt (Nhất là các xung nhiễu từ động cơ) để tránh làm treo vi điều khiển.

3.1.8. 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.

3.1.8.1. Luật điều khiển tỷ lệ số



Hình 3.8: 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:

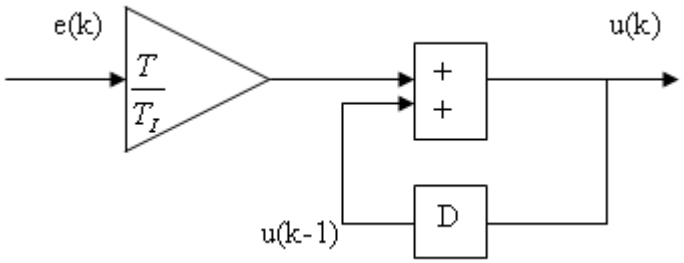
$$u(k) = k_P e(k) \qquad \text{k=0,1,2 ...} \qquad (3.1)$$

3.1.8.2. Luật điều khiển tích phân số

Ta có phương trình sai phân:

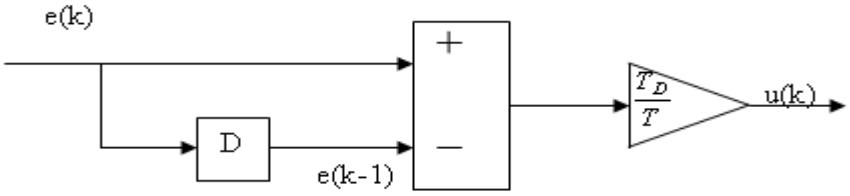
$$u(k) = \frac{T}{T_I} e(k) + u(k-1) \qquad (3.2)$$

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



Hình 3.9: Cấu trúc luật I số.

3.1.8.3. Luật điều khiển vi phân số



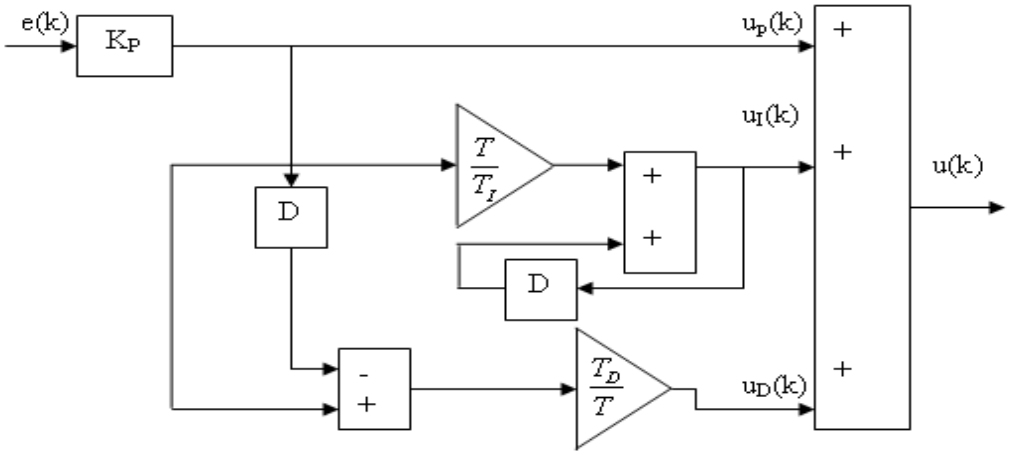
Hình 3.10: Cấu trúc luật D số.

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.

3.1.8.4. Luật điều khiển PID số



Hình 3.11: Cấu trúc luật PID số.

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

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

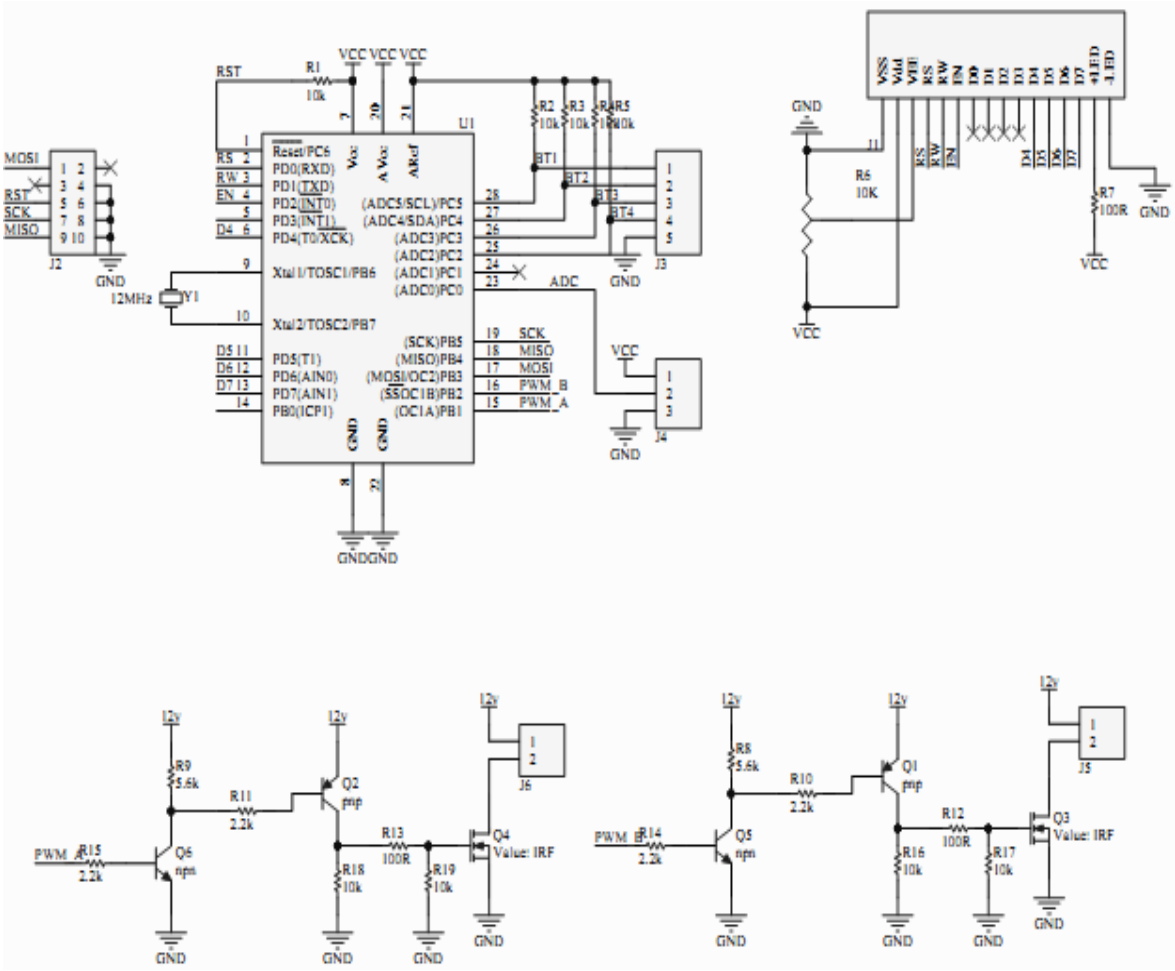
$$u(k) = k_P\left\{(1+\frac{T_D}{T})e(k) - \frac{T_D}{T}e(k-1) + \frac{T}{T_I}e(k) + u_I(k-1)\right\}$$

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

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.

3.2. SƠ ĐỒ NGUYÊN LÝ

3.2.1. Mạch duy trì nhiệt độ của bể sơn



Hình 3.12: Sơ đồ nguyên lý

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:

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

* Các linh kiện trong mạch

1) Cảm biến: LM335

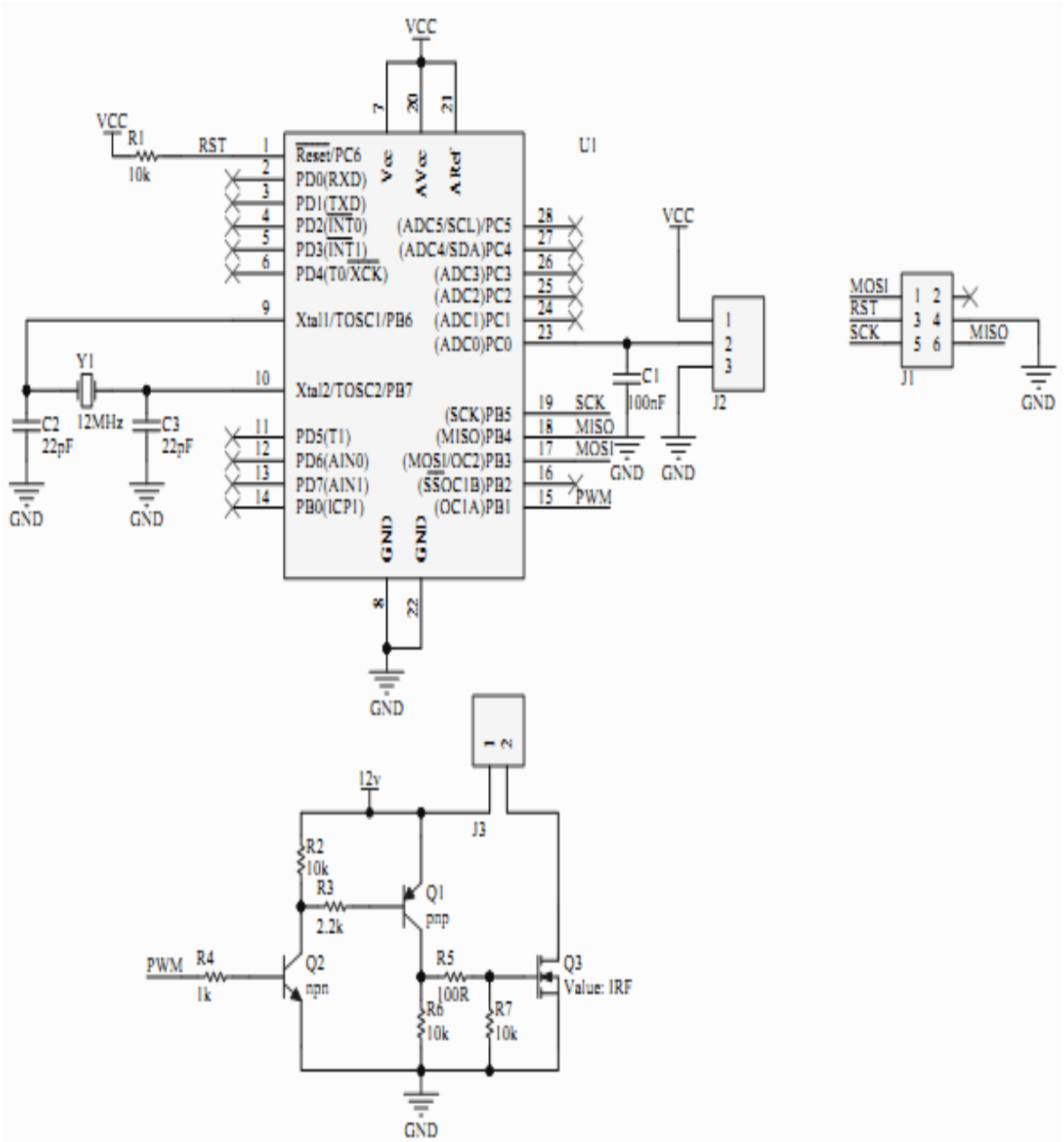
2) Xử lý trung tâm: ATmega8

3) Khối hiển thị: LCD LM016L

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

5) Mosfet IRFZ44N

3.2.2. Mạch điều khiển tốc độ quạt sấy



Hình 3.13: Sơ đồ nguyên lý

1) Cảm biến: LM335

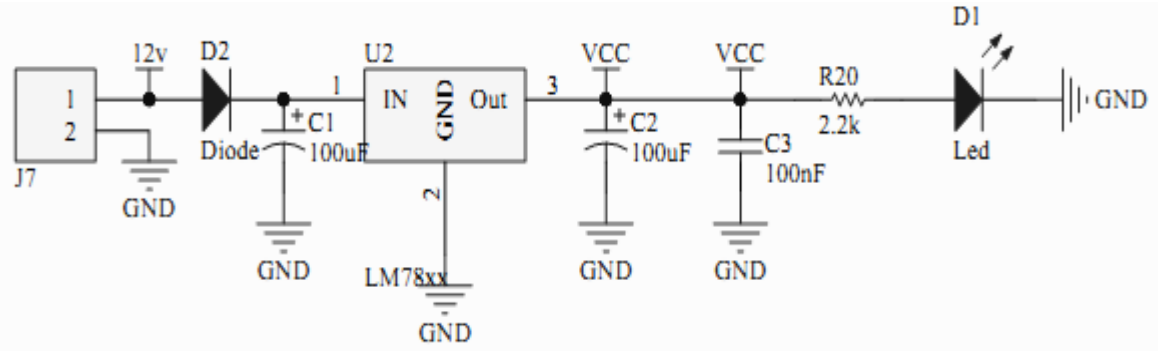
2) Xử lý trung tâm: ATmega8

3) Điện trở, tụ điện, transistor, diode…

4) Động cơ

5) Mosfet IRFZ44N

3.2.3. Mạch tạo nguồn nuôi

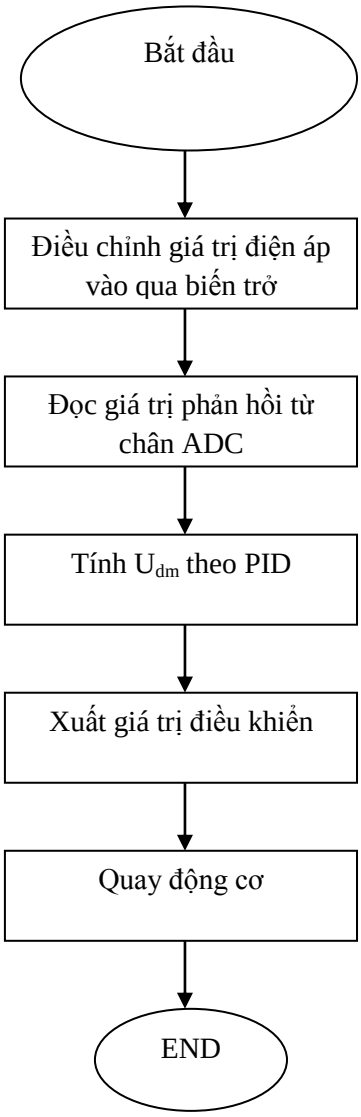


Hình 3.14: Mạch nguồn

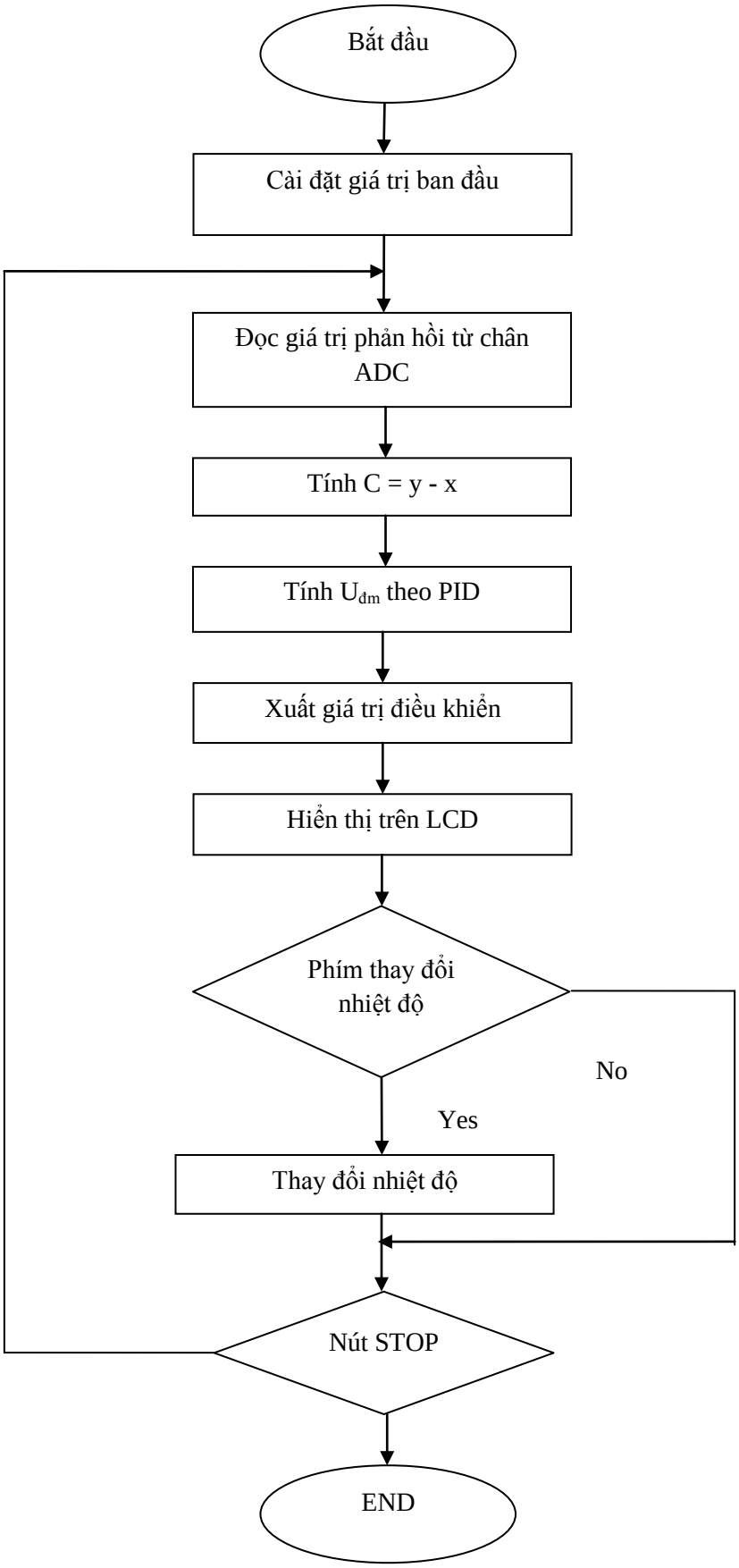
Sử dụng LM7805 và LM7812 để tạo điện áp ra ổn định ở 5V và 12V

3.3. SƠ ĐỒ THUẬT GIẢI

3.3.1. Sơ đồ thuật giải của mạch điều khiển tốc độ quạt sấy



3.3.2. Sơ đồ thuật giải của mạch duy trì nhiệt độ bể sơn



3.4. CHƯƠNG TRÌNH ĐIỀU KHIỂN

3.4.1. Chương trình điều khiển mạch duy trì nhiệt độ bể sơn

```
#include <mega8.h>
#include <delay.h>
#define OPM OCR1AL
#define Fan OCR1BL

#define Mode    PINC.5
#define Up      PINC.4
#define Down    PINC.3
#define On_Off  PINC.2

char Data_LM35_1=0, ev_1=0, ev_2=0, fb=0;
bit run=0, Display=0;
int time_delay=0;
int PWM=0;

unsigned char value_tep=0, Data_LM35=0, Keypad=0, k=0, speed=0, t=0;

// Alphanumeric LCD functions
#include <alcd.h>

#define FIRST_ADC_INPUT 0
#define LAST_ADC_INPUT 0
unsigned int adc_data[LAST_ADC_INPUT-FIRST_ADC_INPUT+1];
#define ADC_VREF_TYPE 0x00

// ADC interrupt service routine
// with auto input scanning
interrupt [ADC_INT] void adc_isr(void)
{
static unsigned char input_index=0;
// Read the AD conversion result
adc_data[input_index]=ADCW;
// Select next ADC input
if (++input_index > (LAST_ADC_INPUT-FIRST_ADC_INPUT))
```

```
  input_index=0;
ADMUX=(FIRST_ADC_INPUT | (ADC_VREF_TYPE &
0xff))+input_index;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
}


// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
TCNT0=0x08;
// Place your code here
   t++;
   if(t==2 && run==1)//Neu dat 20ms va dc nap toc do
   {
      t=0;
      for(k=0;k<10;k++)
      {
         Data_LM35=(Data_LM35+((adc_data[0]-558.558)/2.048))/2;//5v
      }
      fb=Data_LM35;// Sao chep
      ev_1=value_tep-fb;// Sai lech hien tai
      if(fb<value_tep)
      {
         Fan=0;// Tinh toan PWM;// Tat quat gio
         PWM=PWM+(2*ev_1)+(0.001*(ev_1+ev_2));
         if(PWM<256 && PWM>=0)    OPM=PWM;
      }
      else if(fb>value_tep)
      {
         PWM=PWM+(2*ev_1)+(0.001*(ev_1+ev_2));
         if(PWM<256 && PWM>=0)      Fan=PWM, OPM=0;//  Bat quat gio
      }
```

```
        else
        {
           OPM=0;
           Fan=0;
           PWM=0;
        }

        ev_2=ev_1;// Sai lech 2
        Display=1;
     }
     else if(t==2 && run==0)
     {
        t=0;
     }
     if(Mode==0 && time_delay==0)//   Neu an nut cai dat
     {
        Keypad=1;// Bat dau vao che do cai dat
     }
     else if(Up==0 && time_delay==0)//    Neu Up
     {
        Keypad=2;// Xac nhan tang gia tri
     }
     else if(Down==0 && time_delay==0)//    Neu Down
     {
        Keypad=3;// Xac nhan giam gia tri
     }
     if(Keypad >0 && time_delay==0)   time_delay=20;//  Chong rung phim
     else if(time_delay>0)    time_delay--;

}


// Declare your global variables here

void lcd_put_int(int num)//Xuat 1 so nguyen ra LCD
{
   int temp;
```

```
    unsigned char i = 0, c[5];
    temp = num;
    if (temp != 0) {
        if (temp < 0){
            lcd_putchar('-');
            temp = - temp;
        }
        while(temp){
            c[i++] = temp%10;
            temp /= 10;
        }
        while(i) lcd_putchar(c[--i] + '0');
    }
    else lcd_putchar('0');
}


void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=Out Func1=Out
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=0 State1=0
State0=T
PORTB=0x00;
DDRB=0x06;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 11.719 kHz
TCCR0=0x05;
TCNT0=0x08;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 46.875 kHz
// Mode: Ph. correct PWM top=0x00FF
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xA1;
TCCR1B=0x04;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
```

```
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x01;

// USART initialization
// USART disabled
UCSRB=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 187.500 kHz
// ADC Voltage Reference: AREF pin
ADMUX=FIRST_ADC_INPUT | (ADC_VREF_TYPE & 0xff);
ADCSRA=0xCE;

// SPI initialization
// SPI disabled
SPCR=0x00;
```

```
// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTD Bit 0
// RD - PORTD Bit 1
// EN - PORTD Bit 2
// D4 - PORTD Bit 4
// D5 - PORTD Bit 5
// D6 - PORTD Bit 6
// D7 - PORTD Bit 7
// Characters/line: 16
lcd_init(16);

// Global enable interrupts
#asm("sei")
run=0;
Display=1;
Data_LM35=((adc_data[0]-558.558)/2.048);//5v
Fan=0;
while (1)
      {
      // Place your code here
        if(Display==1)
        {
           Display=0;
           lcd_clear();//    Xoa LCD
           lcd_gotoxy(1,0);
           lcd_putsf("NHIET DO: ");
           lcd_put_int(Data_LM35);
           lcd_gotoxy(3,1);
           lcd_putsf("CONFIG: ");
           lcd_put_int(value_tep);
        }
```

```
if(Keypad==1)
{
run=0;
Display=0;
lcd_clear();
lcd_gotoxy(2,0);
lcd_putsf("SETUP= ");
lcd_gotoxy(8,0);
Keypad=0;// Xac nhan da vao che do cai dat toc do

speed=Data_LM35;
lcd_put_int(speed);
while(Keypad !=1)//    Neu chua an xac nhan
{
   if(Keypad>1)//    Neu nhap so
   {
      if(Keypad==2)   speed++;// Neu an tang
      else if(Keypad==3) speed--;// Neu an giam
      lcd_gotoxy(8,0);
      lcd_putsf("  ");
      lcd_gotoxy(8,0);
      lcd_put_int(speed);
      Keypad=0;
   }
}
PWM=0;
run=0;
t=0;
lcd_clear();
lcd_gotoxy(2,0);
lcd_putsf("HOLD = ");
lcd_gotoxy(8,0);
lcd_put_int(speed);

delay_ms(20);
Display=0;
Keypad=0;
```

```
      lcd_clear();

      value_tep=speed;// Quy doi ra so xung
      run=1;//    Cho phep khoi dong voi toc do moi sai dat
      }
    }
}
```

3.4.2. Chương trình điều khiển mạch điều khiển tốc độ quạt sấy

```
#include <avr/io.h>
#include <mega8.h>
#include <until/delay.h>
#include <avr/interrupt.h>
#include <math.h>
#include <stdio.h>
 #ifndef  cbi
        #define  cbi(port,bit)       (port) &=~(1 << (bit))
#endif
#ifndef  sbi
        #define  sbi(port,bit)       (port) |=~(1 << (bit))
//Dinh nghia cac duong dieu khien motor
#define  MOTOR_DDR  DDRD
#define  MOTOR_PORT PORTD
#define  MOTOR_DIR    6
#define  MOTOR_EN     7
#define  Sampling_time          25          //thoi gian lay mau(ms)
#define  inv_Sampiling_time     40          // |/Saampling_time
#define  PWM_period             8000        // cycle=1ms, f=8MHz
Volatile long int Pulse, pre_Pulse;
Volatile long int rSpeed, Err, pre_Err, Kp=8, Kd=10, Ki=1;//for speed control
```

```
Volatile long int pPart=0,  iPart=0, dPart=0; //PID gains
Volatile long int Ctrl_speed=5;          //van toc can dieu khien
Volatile long int Output;
Volatile unsigned char sample_count=0;
//dieu khien van toc bang PID
Void Motor_Speed_PID(long int des_speed) {
        rSpeed=Pulse-pre_pulse;          //tinh van toc (trong sampling time)
        pre_Pulse=Pulse;            //luu gia tri Pulse: so xung
        Eri=des_Speed-abs(rSpeed);      //tinh error (loi)
//cac thanh phan cua PID
        pPart=Kp*Err;
        dPart=Kd*(Err-pre_Err)*inv_Sampling_time;
        iPart +=Ki*Sampling_time*Err/1000;
        Output +=pPart+dPart+iPart;     //cong thuc duoc bien doi vi la dieu
khhien van toc
//saturation
        If (Output>=PWM_Period) Output=PWM_Period-1;
        If (Output<=0) Output=1;
        OCR1A=Output;   //gan duty cycle cho OCR1A: update PWM
        Pre_Eri=Err;         //luu gia tri error
}
\int main(void){
        //Encoder va cac can nap toc do
        DDRB=0x00;        //set PORTB as a input port to use the T0 input pin
and INT2
        PORTB=0xFF;      //dien tro keo len (nhat la encoder)
```

```
//Motor
MOTOT_DDR=0xF0;
sbi(MOTOR_PORT, MOTOR_DIR);
MCUCSR|=(0<<ISC2);  //ngat INT2 la ngat canh xuong-Falling Edge
GICR |=(1<<INT2);        //enable INT2
//dung timer 2 lam bo dinh thoi25ms, sampling time
TCCR2=(1<<CS21)|(1<<CS20);//CS22=1, CS21=1, CS20=1: Prescaler=1024
TCNT2=60;          //gan gia tri khoi tao cho T/C2 de duoc 25ms ( truong hop
f =8MHz)
TIMSK=(1<<TOIE2);              //cho phep ngat khi co tran o T/C2
//dung timer1 lam PWM generator, Fast PWM mode 14: ICR1 chua time
period
TCCR1A=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
TCCR1B=(1<<WGM13)|(1<<WGM12)|(1<<CS10);
//khoi dong gia tri PWM
OCR1A=1;
ICR1=PWM_Period;
sbi(MOTOR_PORT,MOTOR_EN);//khoi dong motor
sei ( );
        }
}
ISR (TIMER2_OVF_vect){      //update sampling time
        TCNT2=60;                  //gan gia tri khoi tao cho T/C2
        Sample_count++
        Motor_Speed_PID(Ctrl_Speed);
```

```
}
ISR(INT2_vect){
        If (bit_is_set(PINB,0)) Pulse++;
        else Pulse--;
}
```

3.5. MỘT SỐ HÌNH ẢNH VỀ MÔ HÌNH

Hình 3.15: Quạt sấy

Hình 3.16: Bể sơn điện ly

Hình 3.17: Đang gia nhiệt bể sơn

Hình 3.18: Bên trong buồng sấy

KẾT LUẬN

Sau thời gian nghiên cứu và tìm hiểu, với sự hướng dẫn của GS.TSKH Thân Ngọc Hoàn 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 AVR Atmega8
- 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 hệ 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, rơle 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 LCD,trên LED 7 đoạn, 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:

- 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.

Em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

1. DKS GROUP (2010), Giáo trình AVR, DKS GROUP biên soạn.
2. Phạm Minh Hà (2004), ***Kỹ thuật mạch điện tử***, Nhà xuất bản khoa học và kỹ thuật.
3. Bùi Xuân Hòa, Bùi Hồng Huế (2009), ***Hướng dẫn thực hành vi điều khiển AVR***, Nhà xuất bản xây dựng.
4. Ngô Diên Tập (2009), ***Kỹ thuật vi điều khiển AVR***, Nhà xuất bản khoa học và kỹ thuật.
5. Các trang web của Việt Nam các bạn có thể truy nhập:

   www.dientuvietnam.net

   www.dientuvienthong.net

   www.webdien.com

   www.tailieu.vn

   www.hocavr.com/

MỤC LỤC