TRƯỜNG ĐẠI HỌC TRÀ VINH

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

BÁO CÁO TỔNG KẾT

ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ CẤP TRƯỜNG

TÊN ĐỀ TÀI

THIẾT KẾ ROBOT DÙNG TRONG HỌC TẬP VÀ NGHIÊN CỨU ROBOCON

CHỦ NHIỆM ĐỀ TÀI: Ths. ĐẶNG HỮU PHÚC

ĐƠN VỊ: BỘ MÔN ĐIỆN TỬ - VIỄN THÔNG

Trà Vinh, ngày tháng 11 năm 2012

TRƯỜNG ĐẠI HỌC TRÀ VINH

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

BÁO CÁO TỔNG KẾT

ĐỀ TÀI KHOA HỌC VÀ CÔNG NGHỆ CẤP TRƯỜNG

TÊN ĐỀ TÀI

THIẾT KẾ ROBOT DÙNG TRONG HỌC TẬP VÀ NGHIÊN CỨU ROBOCON

Xác nhận của cơ quan chủ trì
(ký tên và đóng dấu)

Chủ nhiệm đề tài
(ký tên, họ tên)

ĐẶNG HỮU PHÚC

Trà Vinh, ngày 12 tháng 11 năm 2012

LỜI CẢM ƠN

Trước hết, tôi xin chân thành cảm ơn Ban Giám Hiệu, Phòng Khoa học Công nghệ và Đào tạo sau đại học, Phòng Kế hoạch Tài vụ, Khoa Kỹ thuật và Công nghệ của Trường Đại học Trà Vinh đã tạo điều kiện tốt nhất cho tôi thực hiện đề tài nghiên cứu khoa học này.

Tôi xin gởi đến các Thầy và các em sinh viên trong nhóm nghiên cứu đã cùng tôi hoàn thành đề tài nghiên cứu khoa học này.

Cuối cùng cho tôi xin được gởi lời cảm ơn tới những người thân, gia đình, bạn bè và đồng nghiệp đã động viên, khuyến khích, giúp đỡ tôi trong suốt quá trình thực hiện đề tài nghiên cứu này.

Tôi xin chân thành cám ơn!

Chủ nhiệm đề tài

Đặng Hữu Phúc

CHƯƠNG 1:
TỔNG QUAN ĐỀ TÀI

1.1. Sự cần thiết của đề tài

RoBot là sản phẩm được chế tạo ra chủ yếu dựa trên cơ sở của hai yếu tố đó là khoa học và trí tuệ, được tạo ra từ sự tích hợp của nhiều lĩnh vực khoa học và công nghệ như: cơ khí, điện, điện tử và công nghệ thông tin…

Robocon là một cuộc thi rất bổ ích cho sinh viên, kích thích khả năng học hỏi, tư duy và sáng tạo của sinh viên. Cuộc thi này đã có từ rất lâu (năm 2002) và được nhiều trường đại học trong nước tham gia. Tuy nhiên, đối với trường đại học Trà Vình thì cuộc thi này còn rất mới, giáo viên và sinh viên của trường chưa có điều kiện tham gia nên chưa có nhiều kinh nghiệm về Robocon. Điều này đã được thể hiện qua cuộc thi Robocon cấp trường vừa qua, đề thi quá dễ, các robot được thiết kế quá sơ sài và không đạt chuẩn để tham gia cuộc thi cấp khu vực hay cao hơn là cấp quốc gia…

Với mục tiêu tạo ra một sân chơi Robocon cho sinh viên trường đại học Trà Vinh, giúp cho giáo viên và sinh viên có được nhận định tổng quan hơn về Robocon, đồng thời chế tạo ra các mô hình robot với kết cấu cơ khí chắc chắn, hoạt động ổn định nhằm làm mô hình thí nghiệm trực quan về Robocon, giúp cho giáo viên và sinh viên có thể tiếp cận Robocon một cách nhanh chóng. Hướng tới mục tiêu TVU sẽ tham gia vào cuộc thi Robocon quốc gia trong tương lai gần, tác giả đã đề xuất đề tài nghiên cứu **"THIẾT KẾ ROBOT DÙNG TRONG HỌC TẬP VÀ NGHIÊN CỨU ROBOCON"**

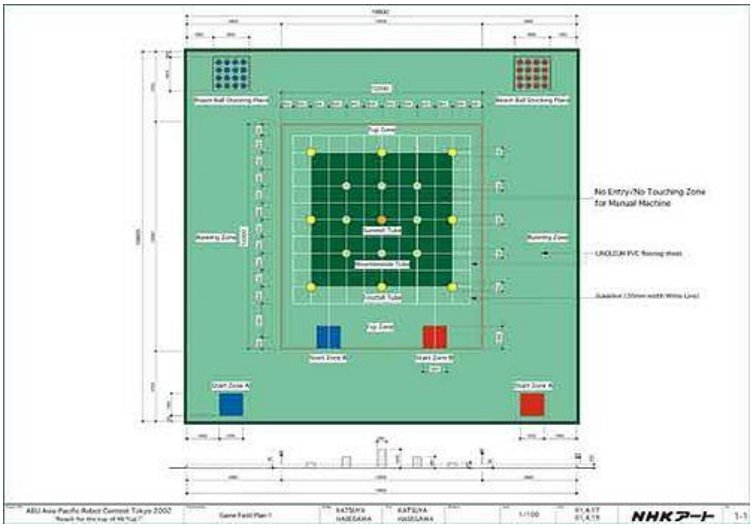
1.2. Các công trình nghiên cứu liên quan

1.2.1. Tổng quan về một số cuộc thi Robocon

Robocon là cuộc thi được khởi xướng tại Nhật Bản. Từ năm 2002, nó trở thành cuộc thi thường niên do Hiệp hội Phát thanh và Truyền hình Châu Á Thái Bình Dương (*Asia-Pacific Broadcasting Union*) tổ chức luân phiên tại các nước thành viên và mang tên ABU Robocon, để cổ vũ cho phong trào sáng tạo robot của thanh niên trong khu vực. Thành viên tại mỗi nước được cử một đội là sinh viên của một trường đại học hay cao đẳng tham dự (ngoại trừ nước đăng cai tổ chức được cử hai đội). Trong đa số trường hợp, đội tham dự ABU Robocon được tuyển ra từ vòng thi trong nước do đài truyền hình thành viên tổ chức với cùng chủ đề nước chủ nhà đưa ra. Mỗi đội sẽ có 2 hoặc 3 robot và sẽ thực thi nhiệm vụ trong vòng 3 phút, đội nào hoàn tất công việc trước (hoặc làm được nhiều việc hơn) đội kia sẽ là đội chiến thắng.

- **Robocon 2002:**

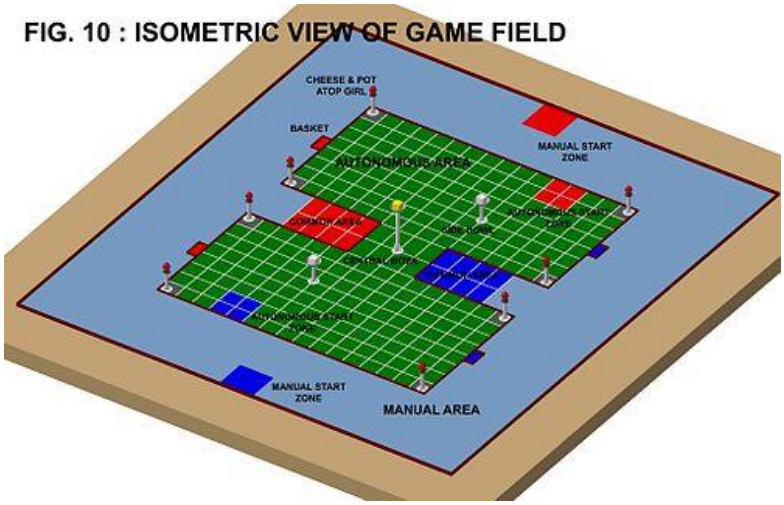
  ROBOCON lần đầu tiên được tổ chức tại Nhật Bản năm 2002 với chủ đề "Chinh phục núi Phú Sĩ". Các Robot di chuyển trên mặt sân bằng phẳng để thực thi nhiệm vụ.



Hình 1.1: Sân chơi Robocon năm 2002

❖ **Robocon năm 2008:**

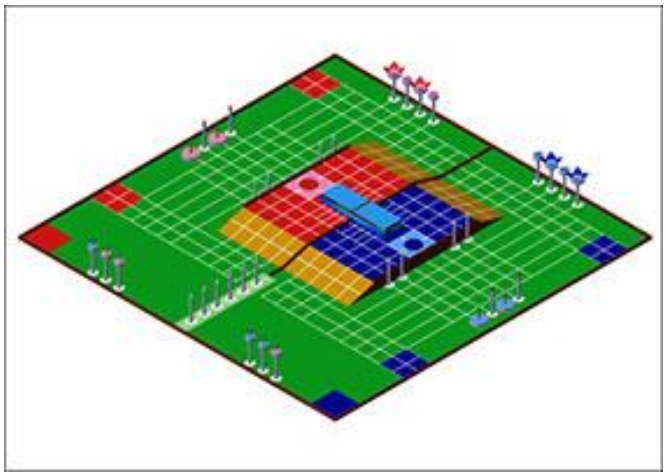
Được tổ chức tại Pune, Ấn Độ với chủ đề *Touch the sky* (Vươn tới bầu trời) và luật chơi khó hơn hẳn các năm trước đó. Các robot thực hiện công việc phức tạp hơn.



Hình 1.2: Sân chơi Robocon năm 2008

❖ **Robocon năm 2011:**

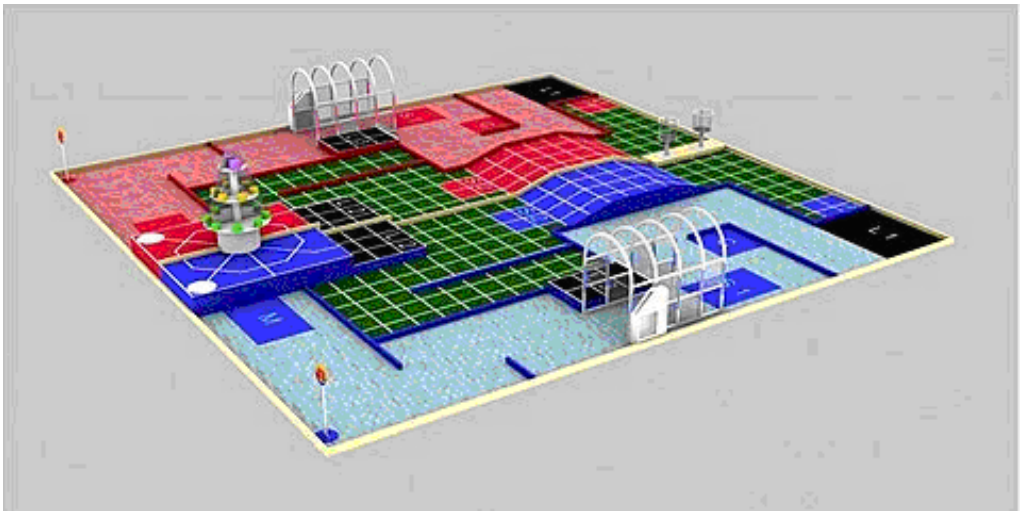
Được tổ chức tại Thái Lan với chủ đề Loy Krathong (là truyền thống của Thái Lan nhằm để tôn vinh các nữ thần của con sông). Các robot phải thực hiện công việc leo dốc, thả quà trên cầu bập bênh.



Hình 1.3: Sân chơi Robocon năm 2011

❖ **Robocon năm 2012:**

Được tổ chức tại HongKong với chủ đề "**Peng On Dai Gat**", Luật thi được dựa trên lễ hội hái bánh bao của người Trung Hoa. Các robot phải chở người, leo dốc, và leo cầu thang để thực thi nhiệm vụ.

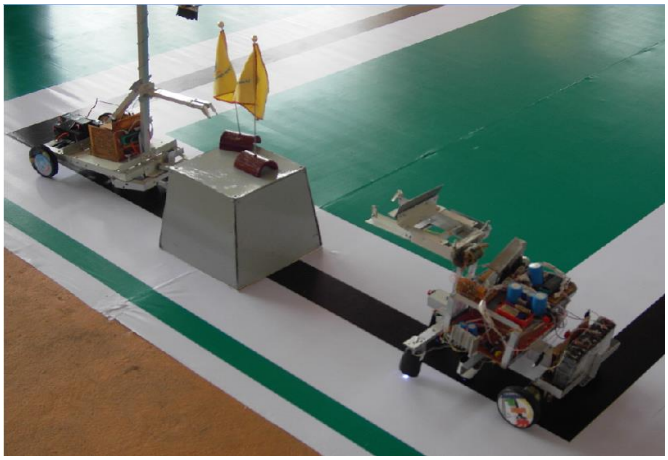


Hình 1.4: Sân chơi Robocon năm 2012

1.2.2. Cuộc thi Robocon cấp trường của trường đại học Trà Vinh

Trường đại học Trà Vinh đã tổ chức 2 cuộc thi Robocon nội bộ vào năm 2011 và 2012.

❖ **Robocon năm 2011:**



Hình 1.5: Robocon TVU năm 2011

❖ **Robocon năm 2012:**

Hình 1.6: Robocon TVU năm 2012

❖ **NHẬN XÉT :**

Cuộc thi Robocon được tổ chức từ năm 2002 đến nay đã được 13 năm, các đề thi nhìn chung có mức độ ngày càng khó, năm sau khó hơn năm trước, các robot phải thực hiện nhiều công việc phức tạp hơn, từ chạy trên địa hình bằng phẳng đến leo dốc, leo cầu thang, chở người… vì vậy đòi hỏi các robot phải được thiết kế ngày càng cải tiến, hoạt động ổn định hơn.

So với 2 cuộc thi Robocon nội bộ của trường đại học Trà Vinh vừa qua, chúng ta dễ dàng nhận ra rằng đề thi quá đơn giản và các robot thiết kế đơn giản, động cơ và bánh xe và các mạch điện bố trí chưa đúng, robot không thể thực hiện được các công việc phức tạp.

1.3. Mục tiêu của đề tài

- Chế tạo một sa bàn thí nghiệm Robot
- Chế tạo một mô hình Robot chạy tự động

- Chế tạo một mô hình Robot điều chỉnh bằng tay

1.4. Phạm vi nghiên cứu

Trong đề tài này, tác giả tìm hiểu về Robocon, phương pháp thiết kế robot tự động và robot điều khiển bằng tay thường sử dụng trong các cuộc thi Robocon, đồng thời nghiên cứu chế tạo một mô hình thí nghiệm Robocon phục vụ cho công việc giảng dạy và nghiên cứu về robot. Đề tài được thực hiện trên cơ sở kế thừa các sản phẩm công nghệ đã được chế tạo và bán trên thị trường như bánh xe, encoder, các board mạch công suất…

1.5. Quy trình thực hiện

Quy trình thực hiện đề tài được thực hiện theo thứ tự các công việc sau:

Bảng 1.1: Quy trình thực hiện đề tài

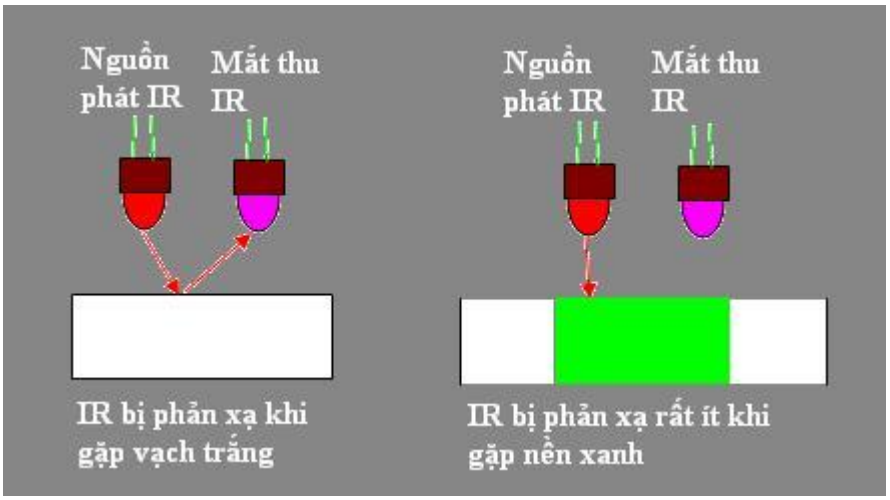
16. Tiến độ thực hiện				
STT	**Nội dung công việc**	**Sản phẩm phải đạt**	**Thời gian**	**Người, cơ quan thực hiện**
1	Tìm kiếm tài liệu (phục vụ việc tính toán lý thuyết và thi công mô hình)	Thông tin về tính toán thiết kế và thi công mô hình	04/2012 đến 05/2012	Đặng Hữu Phúc
2	Mua linh kiện, vật tư, thiết kế và thi công mô hình Robot	Mô hình cơ khí, Board mạch điện	05/2012 đến 07/2012	Đặng Hữu Phúc Đặng Hoàng Vũ
3	Viết chương trình điều khiển	Board mạch và chương trình	07/2012 đến 09/2012	Đặng Hữu Phúc Lê Tấn Cường
4	Chỉnh sửa và viết báo cáo	Báo cáo	09/2012 đến 10/2012	Đặng Hữu Phúc

CHƯƠNG 2:
PHƯƠNG PHÁP VÀ KỸ THUẬT THỰC HIỆN

2.1. Nguyên lý dò đường của Robot [5]

Dò đường hay còn gọi là dò line là phương pháp giúp cho robot có thể di chuyển chính xác đến các mục tiêu đã định trước, Robot sẽ di chuyển theo các vạch trắng hoặc đen có sẵn trên sân thi đấu để đến các mục tiêu và ghi điểm.

Có nhiều phương pháp dò đường khác nhau, ở đây tác giả giới thiệu phương pháp dò đường bám vạch. Đây là phương pháp đơn giản nhưng rất hiệu quả và thường được sử dụng trong cuộc chơi Robocon, phương pháp này được thực hiện dựa trên cơ sở phản chiếu ánh sáng từ thân Robot xuống mặt sân, ánh sáng sẽ phản xạ lại nếu gặp vạch màu trắng và sẽ không phản xạ (hoặc rất ít) nếu gặp vạch màu đen hay vùng màu tối, từ đó giúp Robot xác định được đường đi đến các mục tiêu và thực thi nhiệm vụ do người điều khiển đã lập trình từ trước.



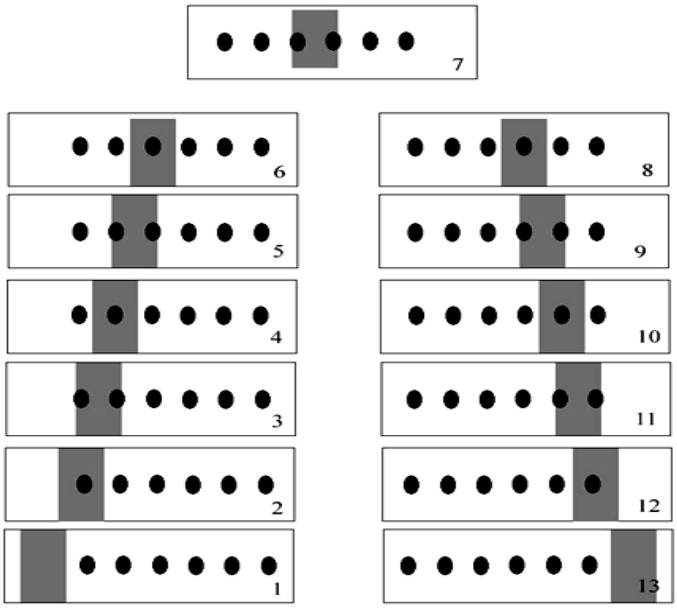
Hình 2.1: Cảm biến nhận biết vạch màu

❖ **Thuật toán cơ bản:**

Nguyên tắc điều khiển đơn giản nhất là robot lệch về phía bên nào thì ta sẽ điều khiển cho robot "bẻ lái" về hướng ngược lại. đây gọi là điều khiển ON-OFF. Cách điều khiển này tuy đơn giản nhưng robot sẽ chạy lắc lư. Như vậy ta sẽ cần một cách điều khiển tốt hơn,cải tiến của cách trên, đó là ta cần phải xem xét thêm là robot đang lệch nhiều hay ít để ta sẽ "bẻ lái" nhiều hay ít. Đây gọi là phương pháp điều khiển tỷ lệ. Ví dụ ta có 6 cảm biến trên robot :3 trái (A1 A2 A3),3 phải (A4 A5 A6). Ta sẽ có các độ lệch tương ứng là +2 ; +1 ; 0 ; 0 ; -1 ; -2 tương ứng với A1 A2 A3 A4 A5 A6. Vậy nếu cảm biến A2 có tín hiệu ta sẽ có độ lệch là +1, A6 tích cực ta sẽ có độ lệch là -2. Như vậy giá trị điều khiển động cơ sẽ tỷ lệ theo độ lệch này :

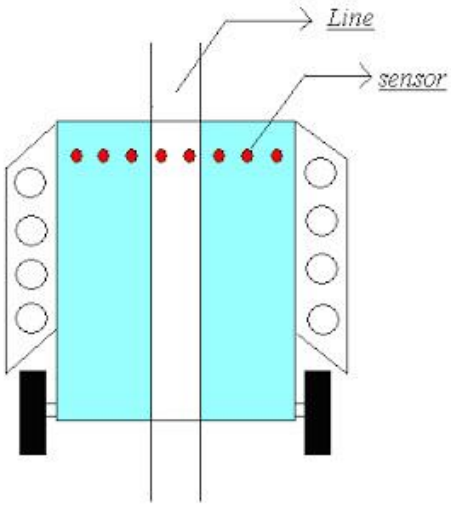
PWM =PWM0 +/- K*độ lệch (1)

Trong đó, K là hệ số tỷ lệ PWM là giá trị điều xung cho động cơ, PWM0 là tốc độ trung bình.

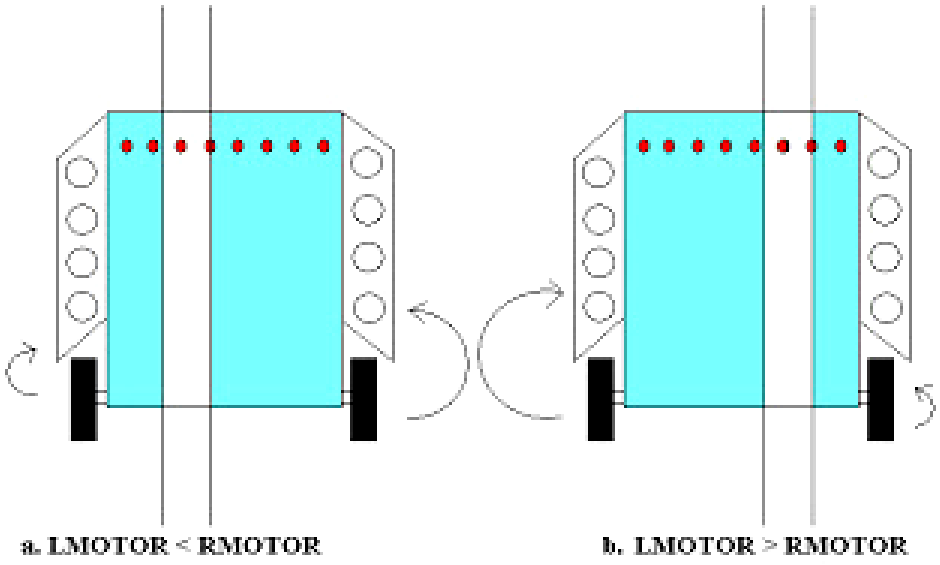


Hình 2.2: Độ lệch line của Robot

Từ các giá trị của cảm biến đưa về vi điều khiển sẻ điều khiển tốc độ các động cơ dò đường một cách phù hợp nhất để đưa robot chạy vào giữa đường line nhằm đảm bảo độ chính xác khi di chuyển.



Hình 2.3: Robot nhận line ở vị trí trung tâm



Hình 2.4: Robot bị lệch line về bên trái và bên phải

Như vậy, sau khi đã xác định được các trường hợp mà khi dò đường robot có thể gặp phải, ta nhận giá trị của cảm biến và so sánh với các trường hợp đã thiết lập, nếu các cảm biến đang rơi vào trường hợp nào thì ta điều khiển động cơ cho từng trường hợp đó.

Từ các giá trị của cảm biến đưa về, vi điều khiển sẻ điều khiển tốc độ các động cơ một cách phù hợp nhất để đưa robot chạy vào giữa đường line nhằm đảm bảo độ chính xác khi di chuyển.

2.2. Vi điều khiển AVR [1],[8]

AVR là họ vi điều khiển 8 bit theo công nghệ mới, với những tính năng rất mạnh được tích hợp trong chip của hãng Atmel theo công nghệ RISC, nó mạnh ngang hàng với các họ vi điều khiển 8 bit khác như PIC, Psoc. Do ra đời muộn hơn nên họ vi điều khiển AVR có nhiều tính năng mới, đáp ứng tối đa nhu cầu của người sử dụng, so với họ 89xx sẽ có độ ổn định tốt hơn, khả năng tích hợp, sự mềm dẻo trong việc lập trình và rất tiện lợi. Hơn nữa, công cụ lập trình (mạch nạp, phần mềm soạn thảo và biên dịch) cho AVR rất đơn giản, dễ sử dụng, có hỗ trợ miễn phí.

❖ **Các tính năng mới của họ AVR:**

Hầu hết các chip họ AVR đều có các tính năng sau:

- Bộ nhớ chương trình Flash có thể lập trình lại rất nhiều lần và dung lượng lớn, có SRAM (Ram tĩnh) và bộ nhớ EEPROM lớn.
- Giao diện nối tiếp Serial Peripheral Interface (SPI).
- Các ngõ vào/ra (I/O) hai hướng.
- Giao tiếp I2C.
- Bộ biến đổi ADC 10 bit.
- Các chế độ tiết kiệm năng lượng như sleep, stand by …
- Một bộ định thời Watchdog.
- Timer/Counter 8 bit, 16 bit tích hợp PWM
- 1 bộ so sánh analog.

- Giao tiếp USART…

❖ **Một số chip AVR thông dụng:**

- AT90S1200
- AT90S2313
- AT90S2323 and AT90S2343
- AT90S2333 and AT90S4433
- AT90S4414 and AT90S8515
- AT90S4434 and AT90S8535
- AT90C8534
- ATtiny10, ATtiny11 and ATtiny12
- ATtiny15
- ATtiny22
- ATtiny26
- ATtiny28
- ATmega8/8515/8535
- ATmega16
- ATmega161
- ATmega162
- ATmega163
- ATmega169
- ATmega32
- ATmega323
- ATmega103

- ATmega64/128/2560/2561
- AT86RF401.

❖ **Lập trình cho AVR**

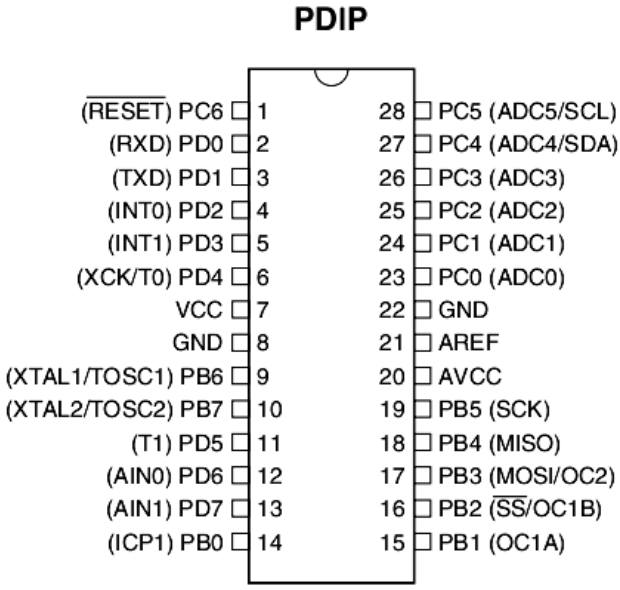
Để lập trình cho AVR, chúng ta có thể sử dụng 2 ngôn ngữ cơ bản là C và ASM. Nhìn chung, 2 ngôn ngữ này có những ưu và nhược điểm riêng. Ngôn ngữ ASM có ưu điểm là gọn nhẹ, giúp người lập trình nắm bắt sâu hơn về phần cứng. Tuy nhiên lại có nhược điểm là phức tạp, khó triển khai về mặt thuật toán, không thuận tiện để xây dựng các chương trình lớn. Ngược lại ngôn ngữ C lại dễ dùng, tiện lợi, dễ debug, thuận tiện để xây dựng các chương trình lớn. Nhưng nhược điểm của ngôn ngữ C là khó giúp người lập trình hiểu biết sâu về phần cứng, các thanh ghi, tập lệnh của vi điều khiển. Hơn nữa, xét về tốc độ, chương trình viết bằng ngôn ngữ C chạy chậm hơn chương trình viết bằng ngôn ngữ ASM, chiếm dụng bộ nhớ nhiều hơn ASM. Tùy vào từng bài toán, từng yêu cầu cụ thể mà ta chọn lựa ngôn ngữ lập trình cho phù hợp.

2.2.1. Chip vi điều khiển ATMEGA8

ATMEGA8 là chip vi điều khiển họ AVR nên có đầy đủ chức năng cơ bản của AVR.

- Có tất cả 28 chân.
- 23 chân I/O chia làm 3 port là B, C, D.
- Nguồn cấp từ 4.5 – 5.5 vol
- 8Kb bộ nhớ Flash, nạp xoá 10.000 lần
- 512 bytes EEPROM, nạp xoá được 100.000 lần
- 1Kb Ram nội
- 32 thanh ghi 8 bit
- 2 timer 8 bit, 1 timer 16 bit
- 3 kênh PWM
- 6 kênh ADC – 10 bit

- Giao tiếp SPI



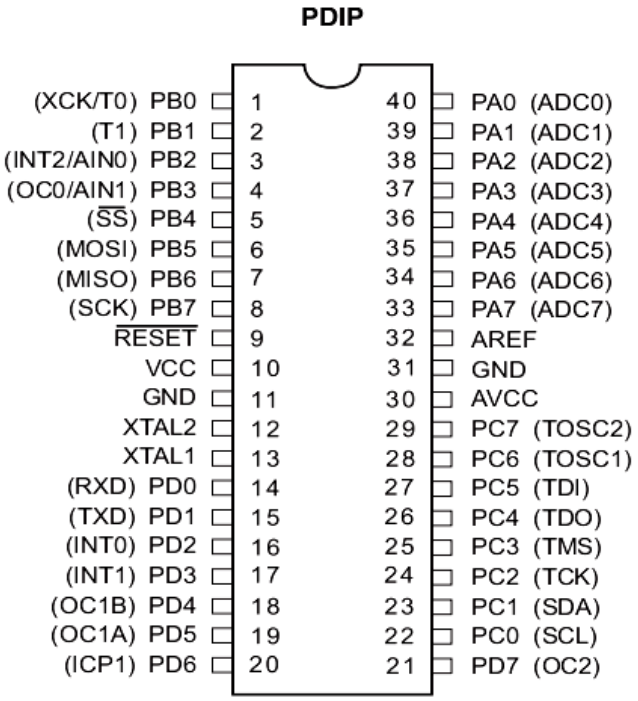
Hình 2.5: Sơ đồ chân ATMEGA8

2.2.2. Chip vi điều khiển ATMEGA32

Cũng như chip ATMEGA8, ATMEGA32 có đầy đủ tính năng của họ AVR, về giá thành so với các loại khác thì giá thành là vừa phải khi nghiên cứu và làm các công việc ứng dụng tới vi điều khiển. Ngoài ra, ATMEGA32 có nhiều chức năng hơn ATMEGA8 như: nhiều I/O hơn, bộ nhớ lớn hơn, …..

❖ **Tính năng :**

- Bộ nhớ 32KB Flash có khả năng đọc, ghi 10000 lần.
- 1024 byte EEPROM có khảnăng đọc, ghi 100000 lần.
- 2KB SRAM.
- 8 kênh đầu vào ADC 10 bit, 4 kênh PWM
- Có 40 chân, trong đó có 32 chân I/O chia làm 4 PORT A,B,C,D. Các chân này đều có chế độ pull_up resistors.
- Giao tiếp SPI, JTAG.
- Nguồn cấp từ 4.5 – 5.5 vol

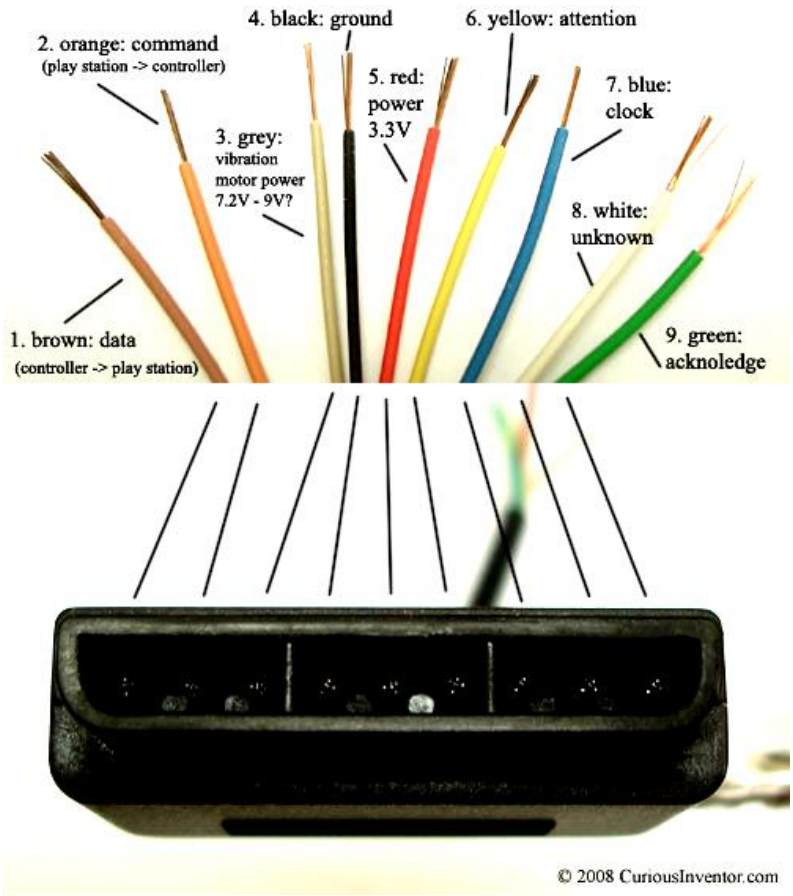


Hình 2.6: Sơ đồ chân ATMEGA32

2.3. Giao tiếp tay game PS2 điều khiển robot [7]

2.3.1. Kết nối phần cứng

Hình 2.7: Tay Game PS2 của Sony



Hình 2.8: Sơ đồ chân của tay Game

Đối với điều khiển robot, chỉ cần dùng các dây sau: Clock, Data, Command, VCC & GND, Attention.

Clock,Data,Command,Attention nối với các chân I/O bất kỳ. Chân Data nên được kéo nguồn bằng điện trở từ 1k-10k.

Clock: xung, đồng bộ hóa quá trình truyền dữ liệu.

Data: dữ liệu từ gamepad về vdk;

Command: dữ liệu từ vdk đến gamepad.

Attention: Chip select

VCC: 3-5V; GND:0V

2.3.2. Cách truyền nhận dữ liệu

Gampad và vi điều khiển truyền và nhận từng byte dữ liệu cùng 1 lúc bằng giao tiếp nối tiếp. Xung Clock được giữ ở mức cao cho đến khi bắt đầu gửi 1byte. Sau đó, Clock sẽ được đưa xuống mức thấp để bắt đầu quá trình truyền và nhận dữ liệu trong thời gian 8 xung Clock. Khi xung Clock ở cạnh xuống, dữ liệu trên đường truyền bắt đầu thay đổi. Khi xung Clock ở cạnh lên, dữ liệu được đọc. Byte có trọng số thấp nhất được truyền trước.

❖ **Các bước truyền nhận dữ liệu**: thông thường trải qua 9 bước.

- Bước 1: Vdk gửi 0x01
- Bước 2: Vdk gửi 0x42
- Bước 3: Vdk gửi 0x00
- Bước 4: Vdk nhận byte thứ 4

(UP,DOWN,RIGHT,LEFT,START,SELECT)

- Bước 5: Vdk nhận byte thứ 5 (□, O , X , Δ , R1,R2,L1,L2)
- Bước 6: Vdk nhận byte thứ 6: Analog bên phải 0x00 = Left 0xFF = Right
- Bước 7: Vdk nhận byte thứ 7: Analog bên phải 0x00 = Up 0xFF = Down
- Bước 8: Vdk nhận byte thứ 8: Analog bên trái 0x00 = Left 0xFF = Right
- Bước 9: Vdk nhận byte thứ 9: Analog bên trái 0x00 = Up 0xFF = Down

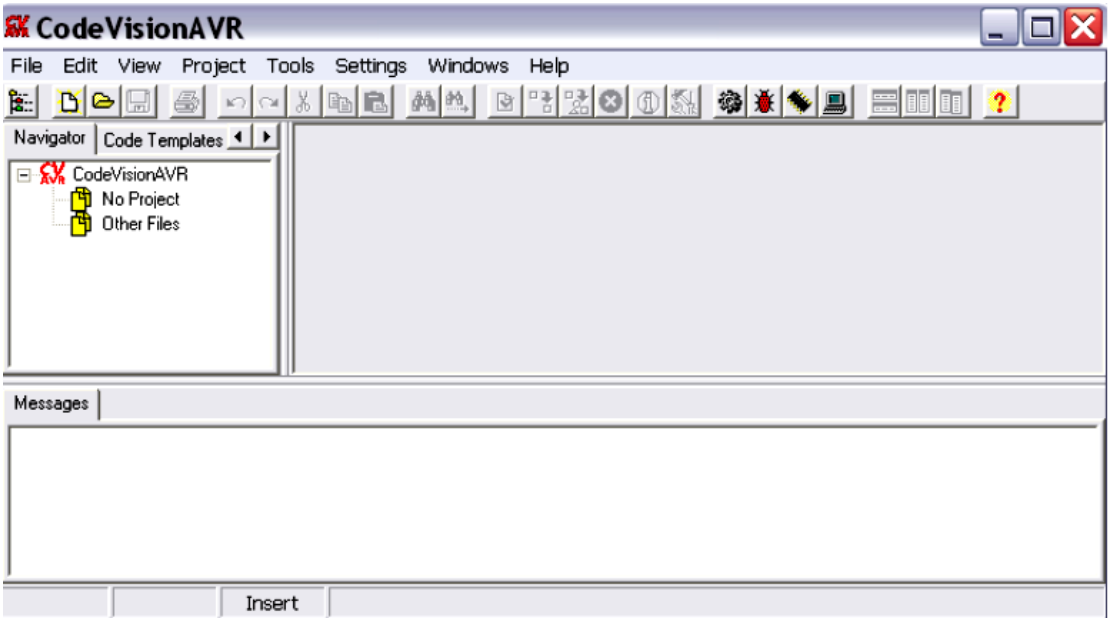
Các nút ấn tích cực mức thấp.

Ví dụ: - Phím UP được ấn thì byte thứ 4 nhận được sẽ là 0b11101111

- Phím vuông được ấn thì byte thứ 5 nhận được sẽ là 0b01111111

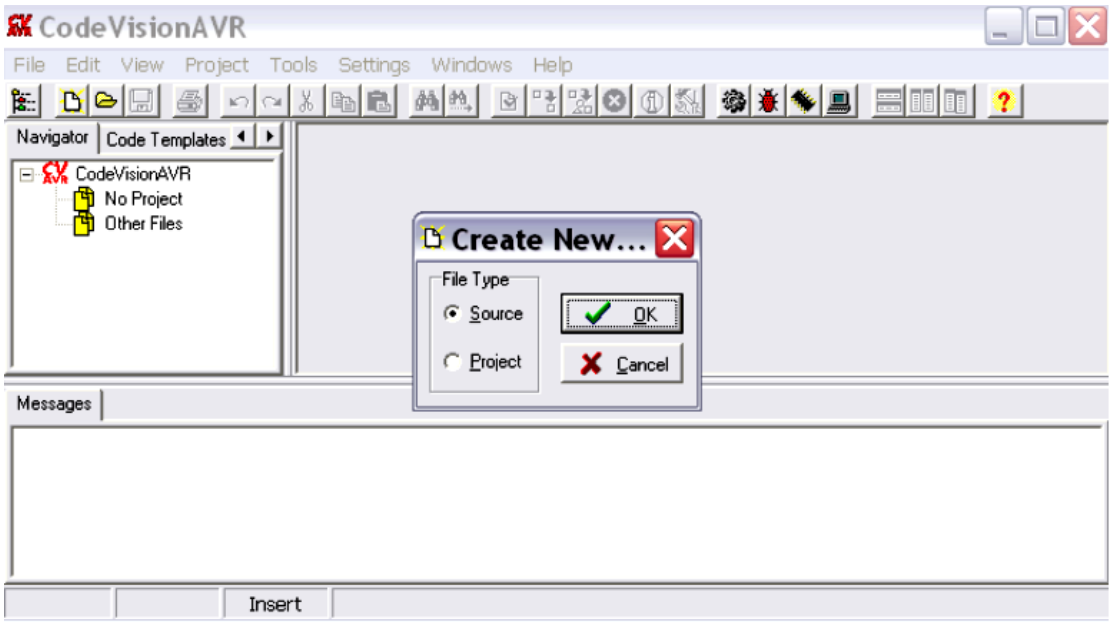
2.4. Phần mềm CodevisionAVR [5],[8],[9]

Có rất nhiều phần mềm lập trình cho AVR, như CodeVisionAVR, IAR, AVRStudio, WinAVR, BascomAVR trong đó CodeVisionAVR là một trong những phần mềm khá nổi tiếng và phổ biến, hỗ trợ nhiều thư viện lập trình. Trong khuôn khổ đề tài này, tác giả sử dụng phần mềm CodeVision để lập trình cho AVR.



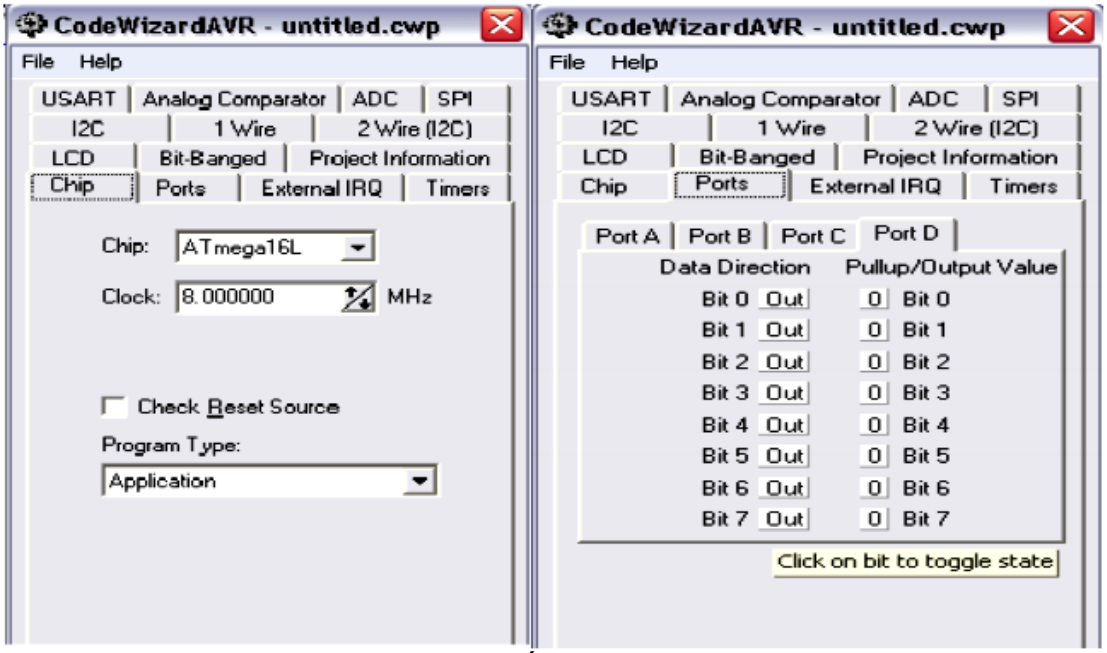
Hình 2.9: Giao diện phần mềm CodeVision

❖ Để tạo Project mới, ta chọn trên menu: **File -> New** được như sau:



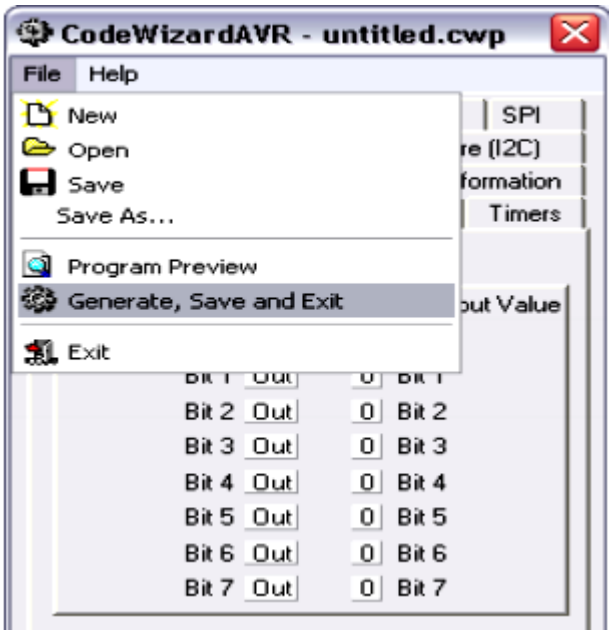
Hình 2.10: Tạo Project mới

Click chọn **Project** sau đó click chuột vào "**OK**" được cửa sổ hỏi xem có sử dụng **CodeWinzardAVR** hay không, ta chọn " **YES**" được giao diện như sau:

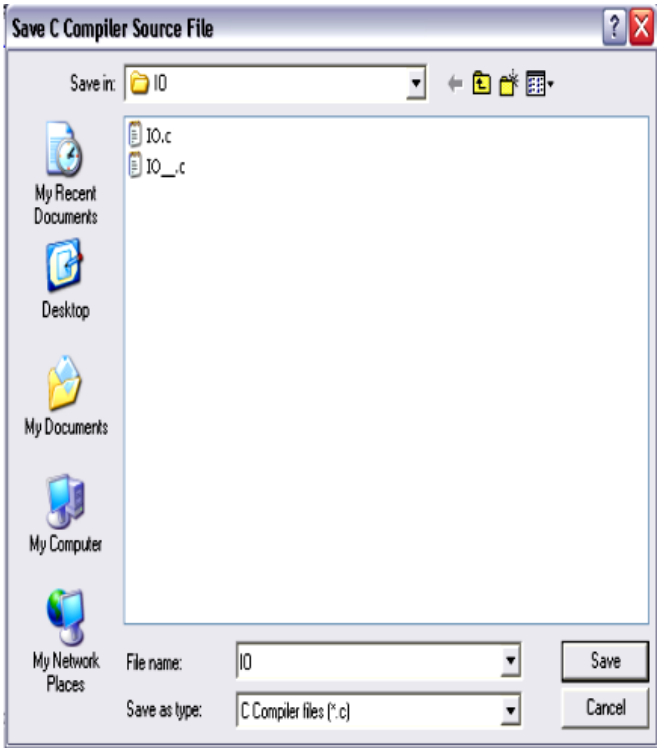


Hình 2.11:Cấu hình chip AVR

Sử dụng chíp AVR nào và thạch anh tần số bao nhiêu ta nhập vào tab Chip. Để khởi tạo cho các cổng IO ta chuyển qua tab Ports. Các chân IO của AVR mặc định trạng thái IN, muốn chuyển thành trạng thái OUT để có thể đưa các mức logic ra ta click chuột vào các nút IN (màu trắng) để nó chuyển thành OUT trong các Tab Port. Sau đó chọn **File** -> **Generate**, **Save and Exit**.

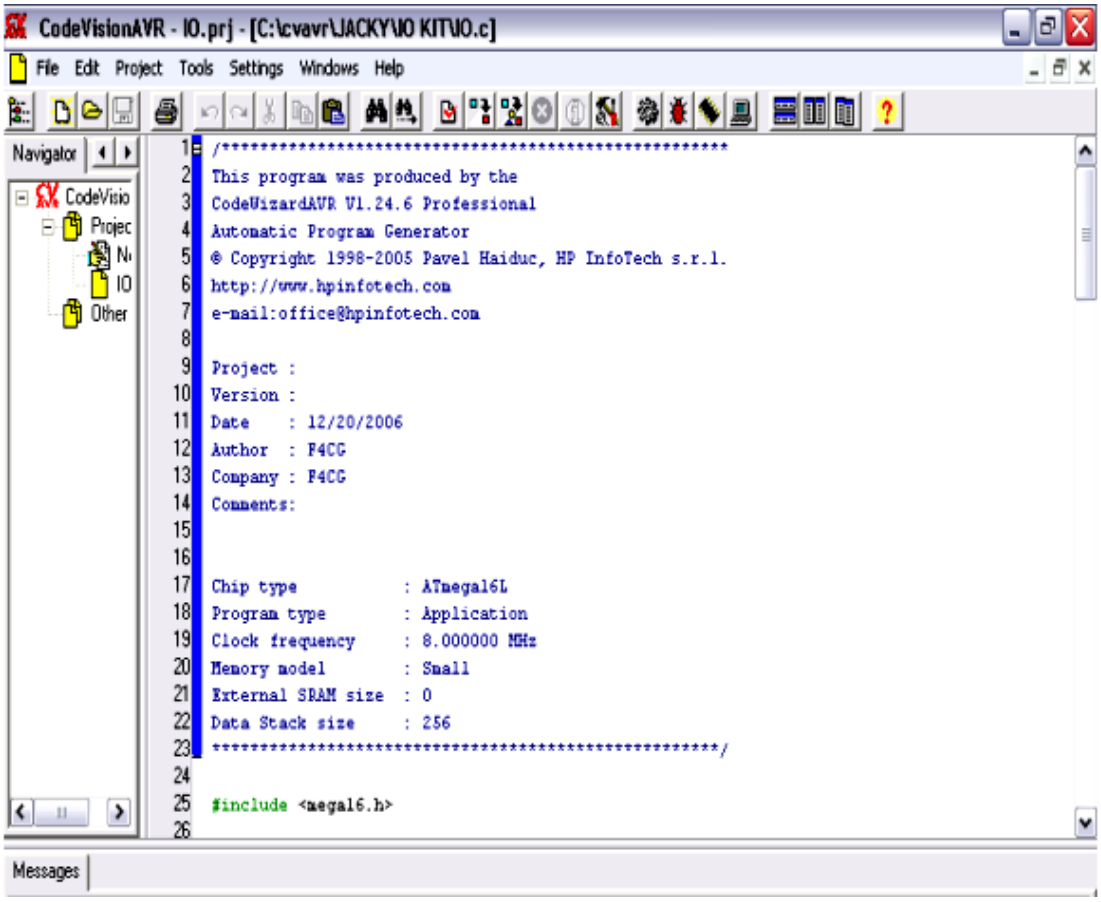


Hình 2.12: Lưu lại cấu hình chip



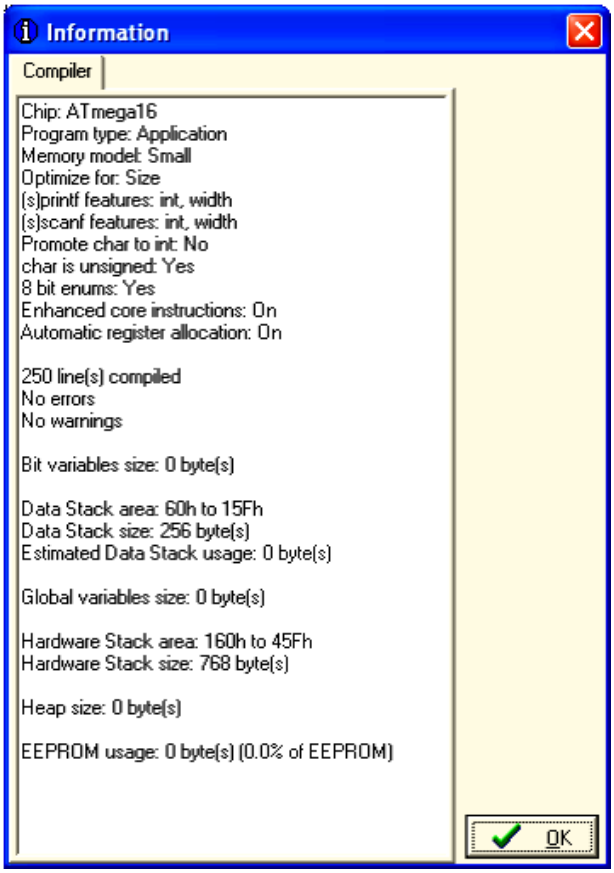
Hình 2.13: Đặt tên Project và lưu vào ổ cứng máy tính

Sau khi đặt tên cho Project, ta chọn " **SAVE**" để lưu Project vào ổ cứng máy tính, sau đó ta sẽ có giao diện soạn thảo chương trình cho AVR.



Hình 2.14: Giao diện soạn thảo chương trình

Sau khi soạn thảo chương trình xong, ta tiến hành biên dịch chương trình bằng cách ấn **F9** hoặc vào **menu : Project** $\rightarrow$ **Compile**. Được cửa sổ **Information** như sau:



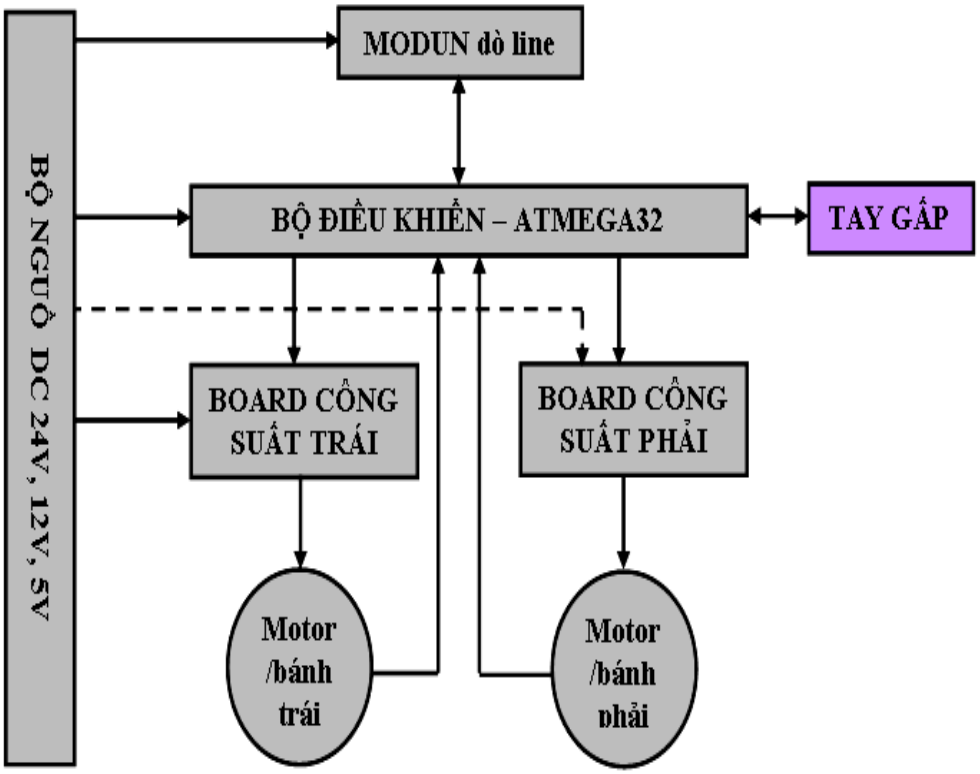
Hình 2.15: Cửa sổ biên dịch

Nếu chương trình không phát hiện lỗi, ta chọn "**OK**" khi đó một file có phần mở rộng là ".Hex" sẽ được tạo ra trong Project, ta dùng file này để nạp vào chip AVR bằng thiết bị nạp.

CHƯƠNG 3
THIẾT KẾ VÀ THI CÔNG

3.1. Robot tự động

3.1.1. Sơ đồ khối

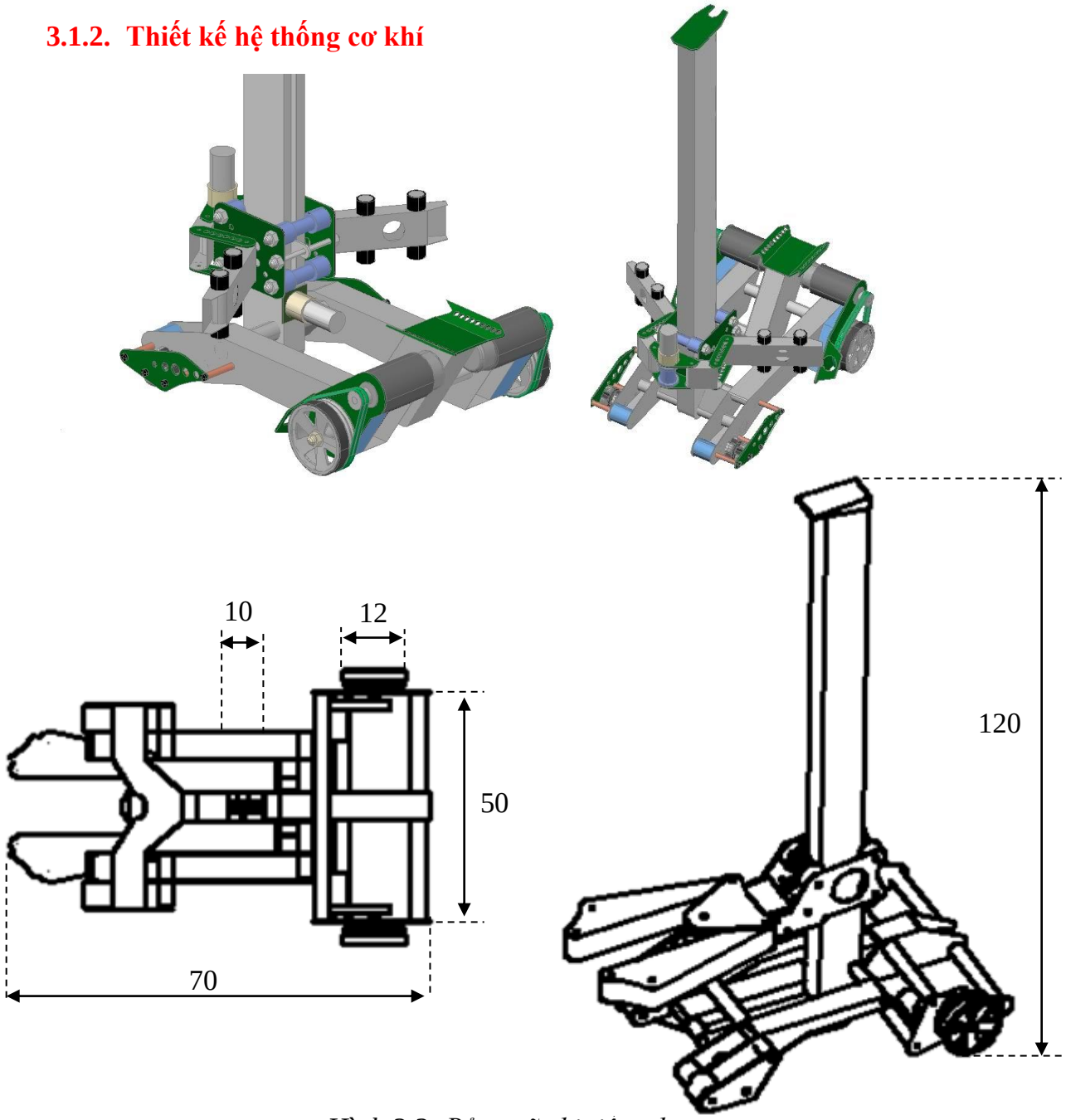


Hình 3.1: sơ đồ khối Robot tự động

- Khối nguồn: cung cấp nguồn DC cho Robot, gồm các nguồn 24v, 12v, 5v.
- Bộ điều khiển: Điều khiển Robot hoạt động theo chương trình do người lập trình định sẵn.
- Board công suất trái và phải: điều khiển 2 động cơ bánh xe trái và bánh xe phải của Robot.

- Modun dò line: thực hiện chức năng dò đường, giúp Robot di chuyển đúng hướng.
- Tay gấp: thực hiện chức năng gấp quà.

3.1.2. Thiết kế hệ thống cơ khí



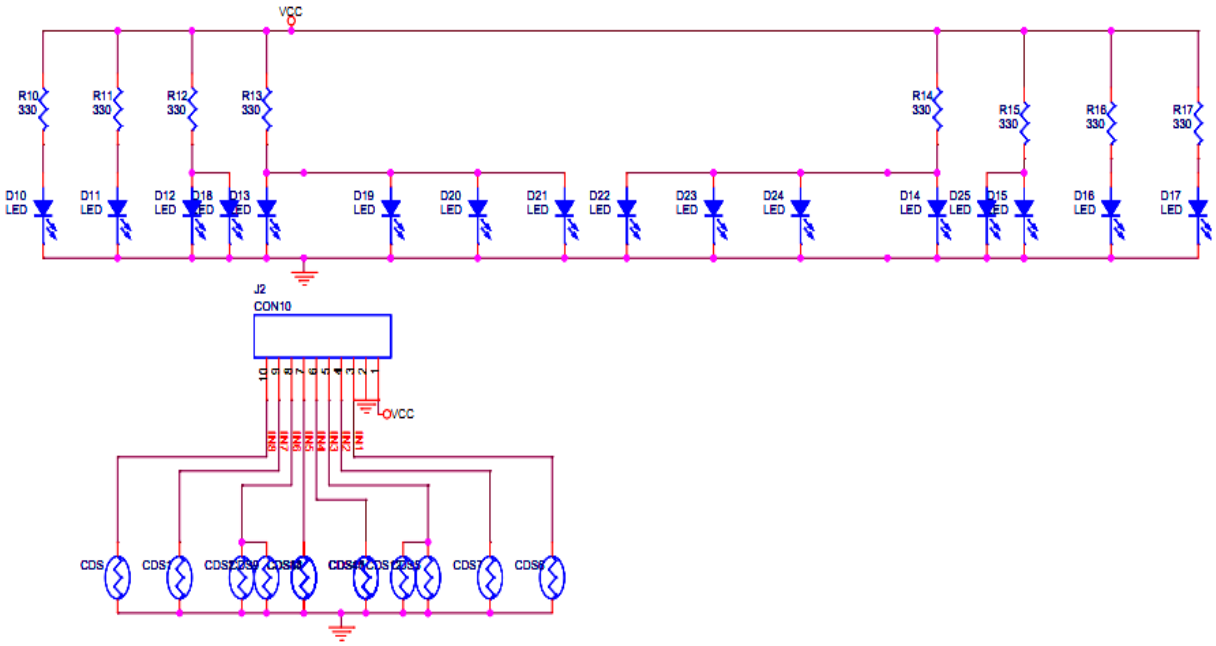
Hình 3.2: Bảng vẽ chi tiết robot

❖ **Yêu cầu thiết kế:**

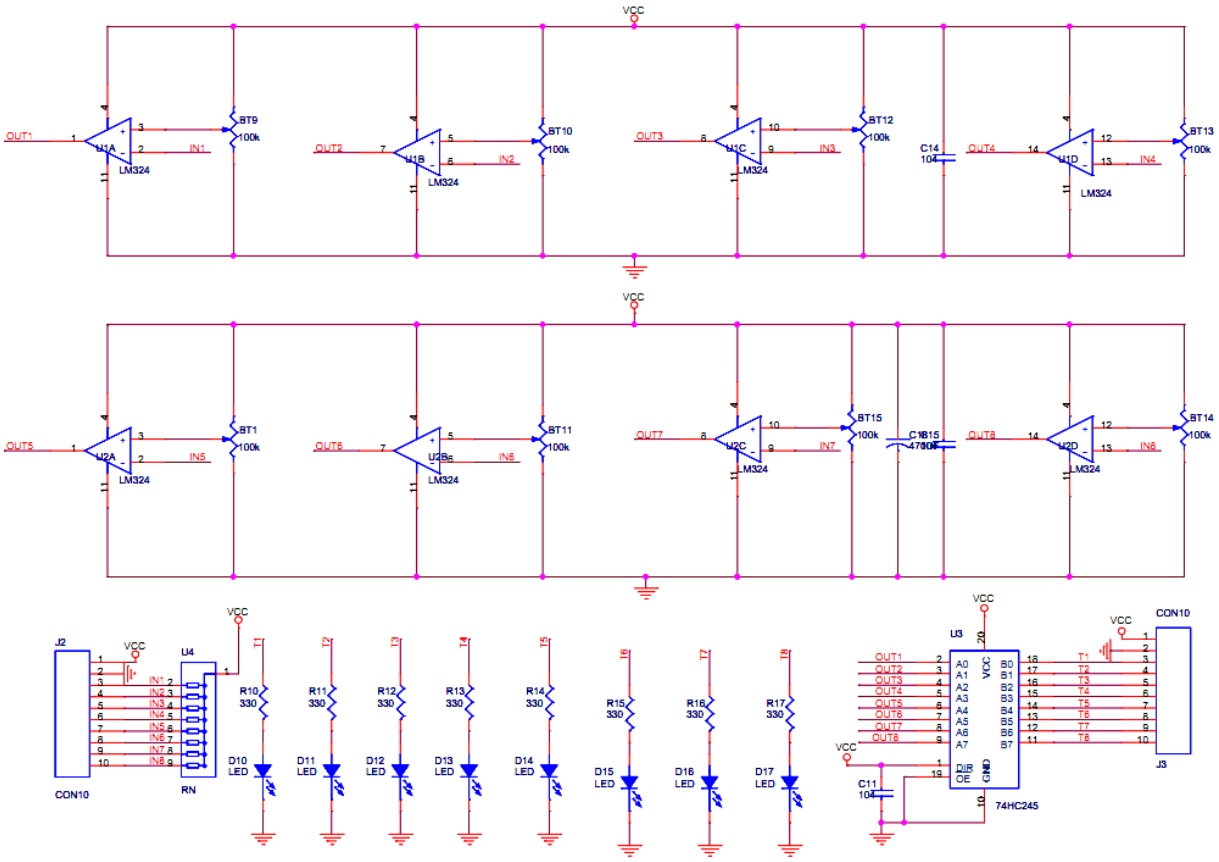
- Khối Robot phải nhỏ 30kg
- Chiều cao từ 1m đến 1,5m
- Chiều rộng từ 0.5m đến 1m
- Tay gấp di chuyển dọc thân
- Bánh xe trước là bánh đa hướng Omi
- Bánh xe sau bằng nhôm đúc

3.1.3. Thiết kế các mạch điện

❖ **Mạch dò đường:**



Hình 3.4a: Board sensor dò line

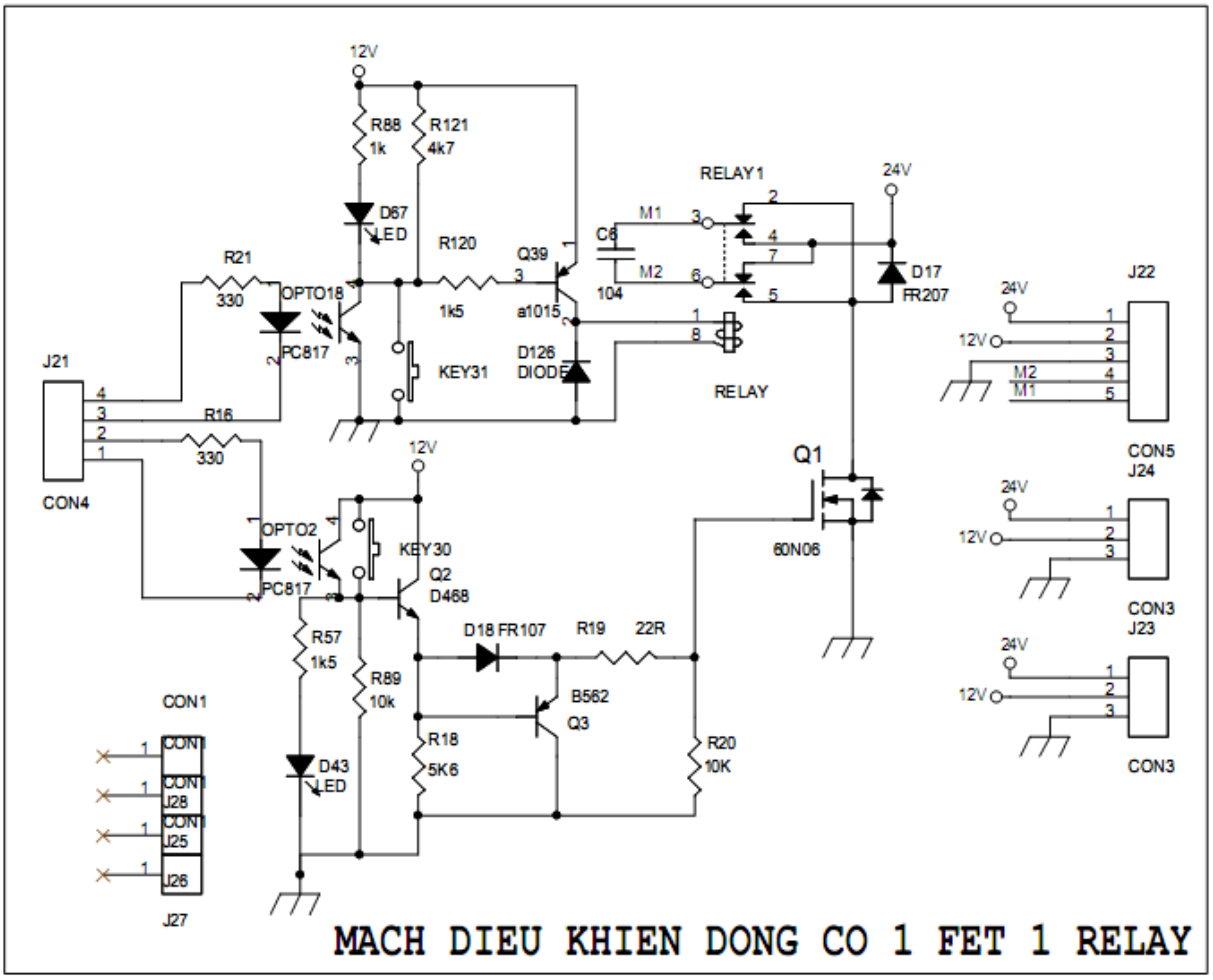


Hình 3.4b: Board sensor dò line

Bộ cảm biến dò line của robot được thiết kế dựa trên cơ sở phản chiếu ánh sáng, nguồn sáng phát ra từ Led và phản chiếu về quang trở kết nối với Op-amp để so sánh mức điện áp, khi phát hiện line, điện áp ngõ ra của cảm biến là mức 1, ngược lại là mức 0. Vi điều khiển sẽ đọc các giá trị điện áp này để điều khiển robot chạy đúng hướng.

- **Mạch công suất:**
  - **Mạch công suất sử dụng loại 1Fet + 1 Relay có Opto cách li để bảo vệ chíp vi điều khiển khi có sự cố xãy ra.**
  - **Fet có chức năng điều khiển đóng ngắt role theo tần số PWM để thay đổi điện áp cấp cho động cơ.**

- **Ngõ vào tín hiệu có 4 chân gồm: PWM + (chân 1), PWM - (chân 2), Dir + (chân 4), Dir - (chân 3).**
- **Ngõ ra gồm có 5 chân gồm: chân nguồn 12v, 24v, chân M1 và M2 để gắn động cơ.**



Hình 3.5: mạch công suất

3.2. Robot điều khiển tay

Robot điều khiển bằng tay có kết cấu gần giống với Robot tự động, tuy nhiên có điểm khác biệt là Robot được điều khiển bằng bộ điều khiển tay, và kết cấu cơ khí được thiết kế chắc chắn hơn, tay gấp đa năng hơn, để đảm bảo khả năng hoạt động linh hoạt của Robot.

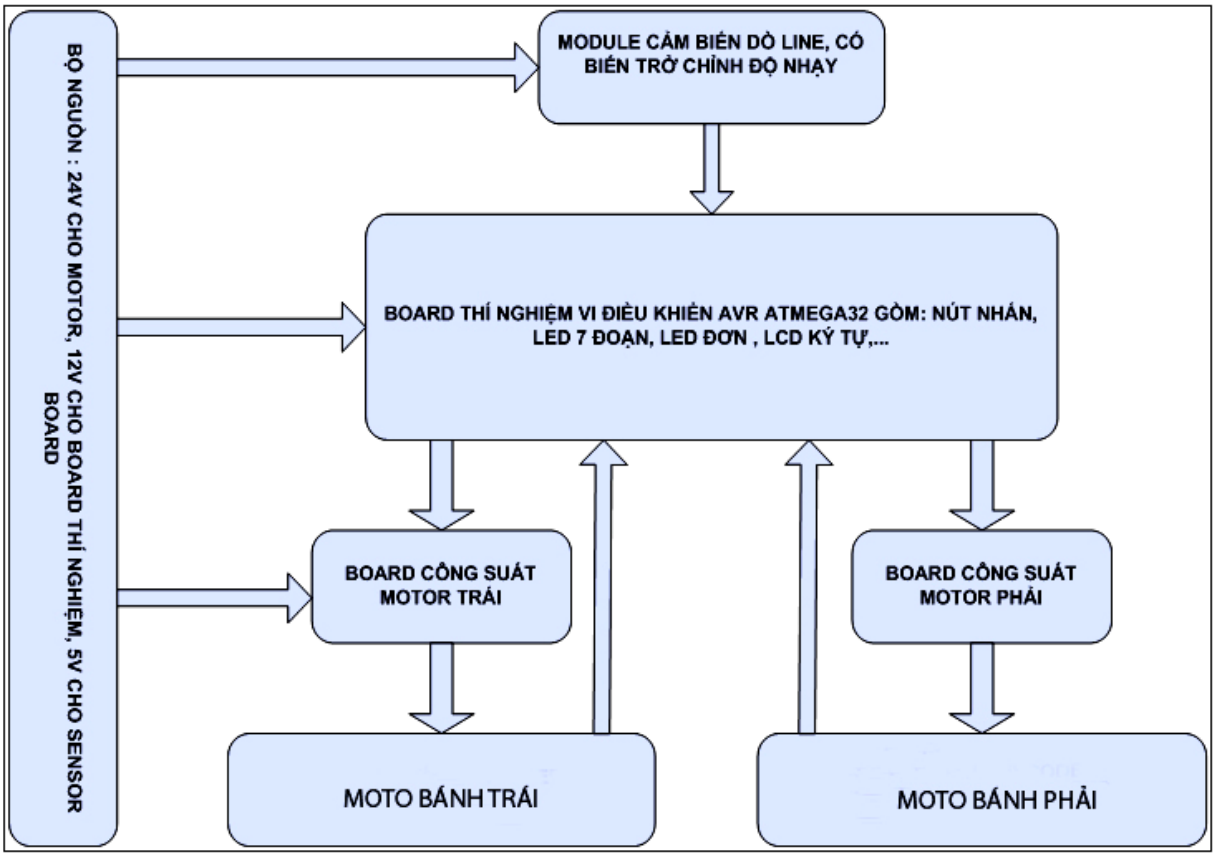
❖ **Bộ điều khiển bằng tay gamepad của sony:**

Hình 3.6: Bộ điều khiển bằng tay

Bộ điều khiển tay với các nút nhấn thực hiện các chức năng sau: Robot chạy thẳng, lùi thẳng, quẹo trái, quẹo phải, cánh tay chạy ra - rút vào, gấp và thả vật.

3.3. Sa bàn thí nghiệm robot

3.3.1. Sơ đồ khối



Hình 3.7: Sơ đồ khối sa bàn thí nghiệm

3.3.2. Chức năng các khối

❖ **Module cảm biến dò line (dò đường đi cho robot):**

- Module này có chức năng nhận biết các vạch trắng và đen, đưa tín hiệu về mức TTL cho board vi điều khiển từ đó điều tốc các motor trái và phải.
- Nguồn cấp cho module là 5V, được cung cấp từ board nguồn.
- Module có 8 led đơn báo trạng thái của 8 kênh: led sáng ứng với vạch trắng, led tắt ứng với màu đen.-Trên module có biến trở dùng chỉnh độ nhạy của sensor.

❖ **Board thí nghiệm vi điều khiển AVR ATMEGA32:**

- Board bao gồm thành phần chính là vi điều khiển ATMEGA32, bao gồm các khối sau:

✓ Khối các ngõ I/O thông dụng:

- 32 ngõ Output với LEDs (high active).
- 32 ngõ Input với nút nhấn (configurable sensitive level).
- 4 LED 7 đoạn.
- 1 TEXT LCD 16*2.
- 1 còi Buzze.
- 32 ngõ IO này cũng được nối ra IO mở rộng của board.

✓ Khối ADC:

- 1 cảm biến nhiệt độ LM35 hoặc DS1820 (Option).
- 1 biến trở xoay potentiometer.

✓ Khối truyền thông UART:

- Tích hợp RS232 transceiver (CMOS ⇔ TTL).
- Hỗ trợ ngõ mở rộng cho card chuyển USB ⇔ RS232.
- Hỗ trợ ngõ mở rộng cho module RF.

✓ Khối RTC:

- Sử dụng thạch anh 32K cung cấp clock cho bộ dao động định thời gian thực.

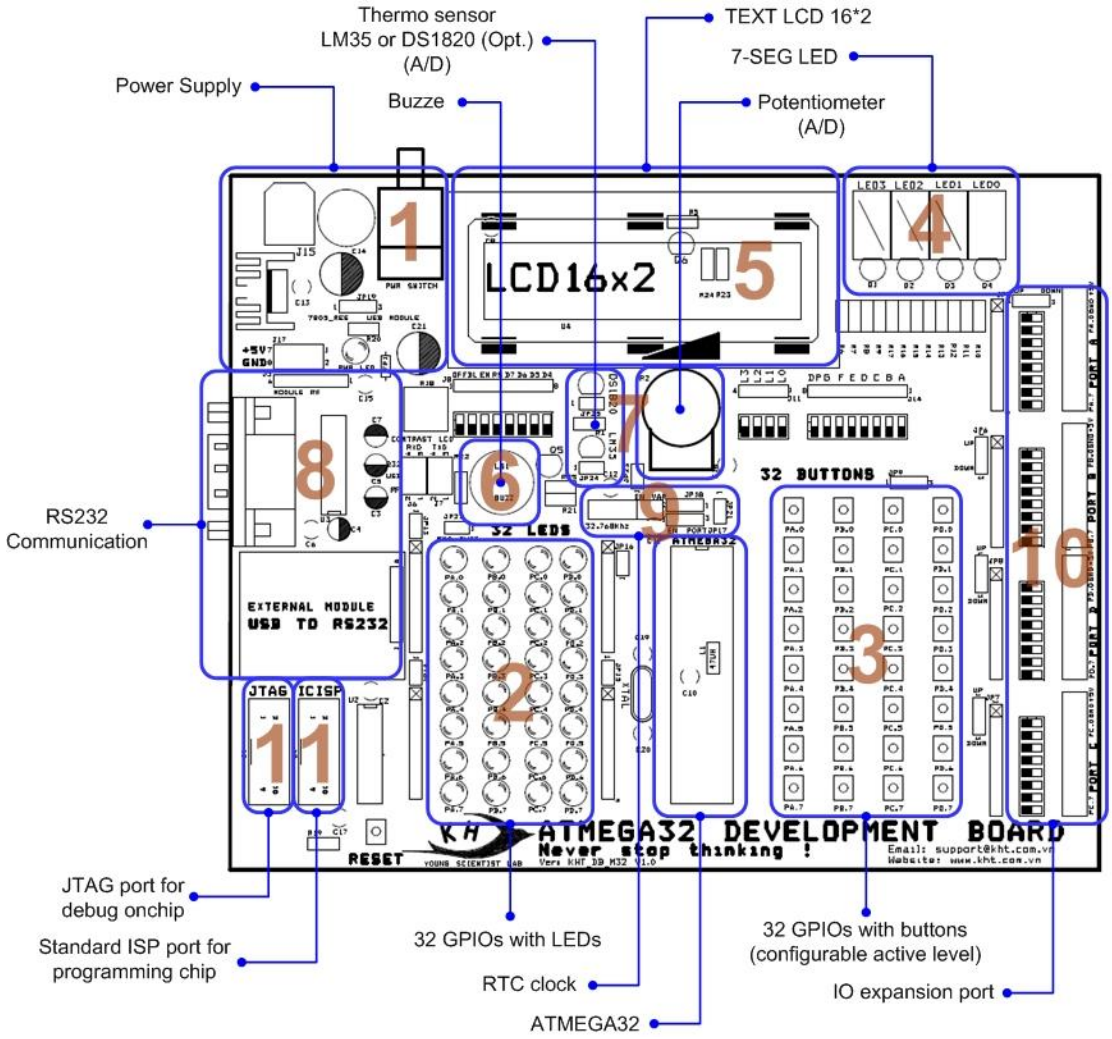
✓ Khối programming và debug port:

- Hỗ trợ 1 ngõ ISP tiêu chuẩn.
- Hỗ trợ 1 ngõ JTAG để debug onchip.

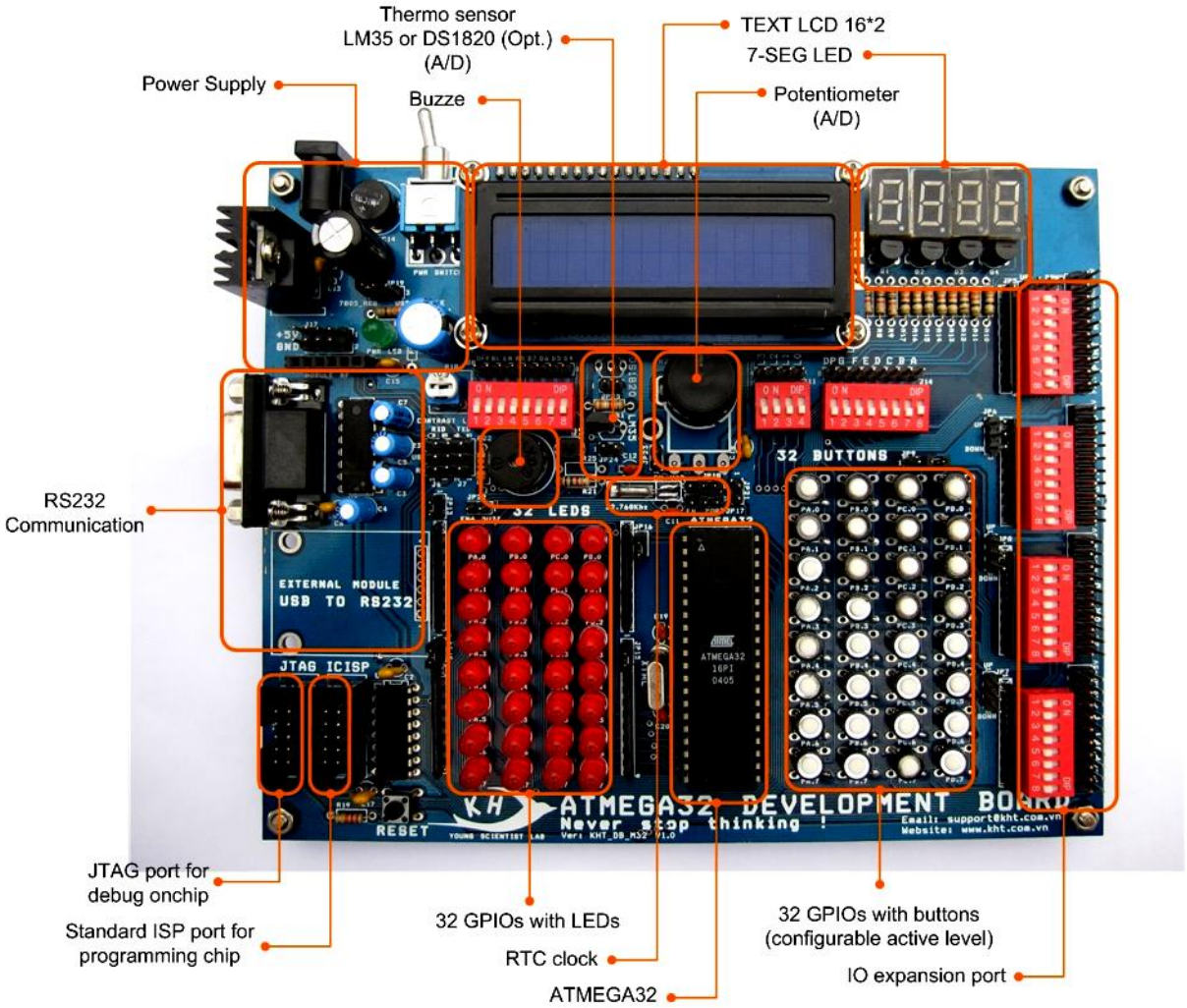
✓ Khối nguồn:

- Sử dụng bộ adapter rời (đi kèm) hoặc lấy nguồn từ card chuyển USB ⇔ RS232.

✓ **Sơ đồ tổng quát như sau:**



Hình 3.8: Sơ đồ tổng quát board thí nghiệm vi điều khiển



Hình 3.9: board thí nghiệm vi điều khiển thực tế

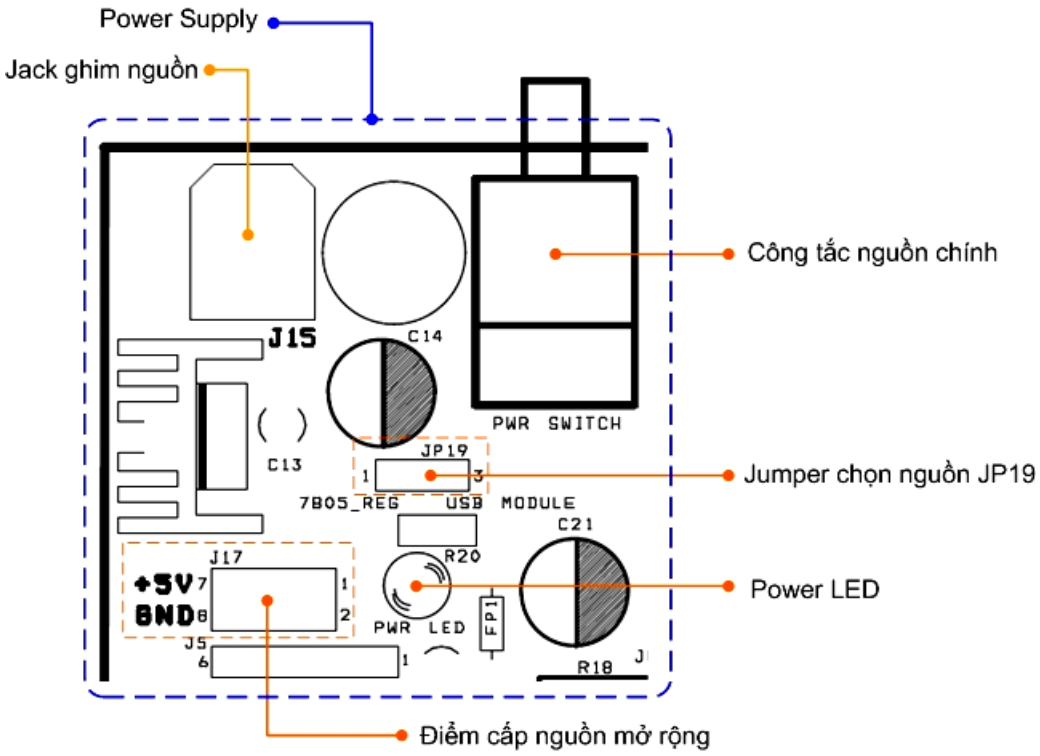
❖ **Board công suất motor :**

- Phần công suất dùng 4 POWER MOFETs IRF3205 có thông số : VDSS = 55V, RDS(on) = 8.0mΩ, ID = 110A
- Điện áp cấp ngõ vào từ +12V đến +40VDC.
- Sử dụng opto xung 6N137, và IC lái MC33883 nên tần số PWM có thể đáp ứng lên đến 100Khz.
- Ngõ vào opto cách ly PWM+, PWM-, DIR+, DIR-.

- Sử dụng IC kích fet chuyên dụng MC33883 của Freescale, nên duty PWM đạt được hiệu suất cao, ưu điểm hơn so với các IC kích fet khác.
- Có Led báo nguồn, Led báo chiều động cơ và PWM.
- Board có 2 nút nhấn DIR và PWM thích hợp cho việc test board bằng tay.

❖ **Khối nguồn:**

Sử dụng bộ adapter rời hoặc lấy nguồn từ card chuyển USB ⇔ RS232.



Hình 3.10: Sơ đồ nguyên lý khối nguồn

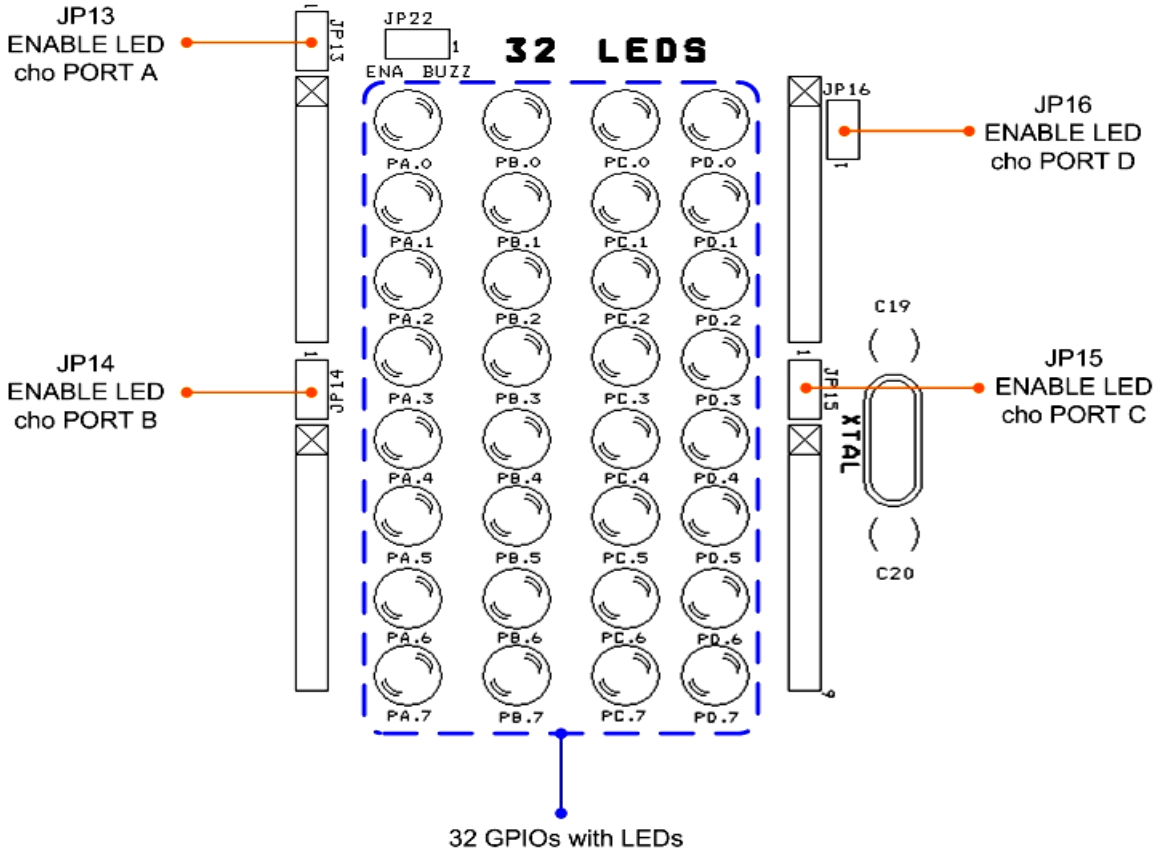
Bảng 3.1: Các chế độ nguồn cấp

Power mode	JP19
Từ adapter rời	1-2 đóng
Từ card chuyển USB ⇔ RS232	2-3 đóng

Khối nguồn cũng cung cấp sẵn các điểm lấy nguồn mở rộng cho các mục đích khác.

❖ **Khối IO với 32 ngõ output LEDs**

32 LEDs được lái trực tiếp từ 32 ngõ IO, tích cực mức cao (high active) và có thể kích hoạt/cấm từng port (8 LEDs) một cách riêng rẽ nhau nhờ vào các jumper ENABLE/DISABLE LED output từ JP13 → JP16.



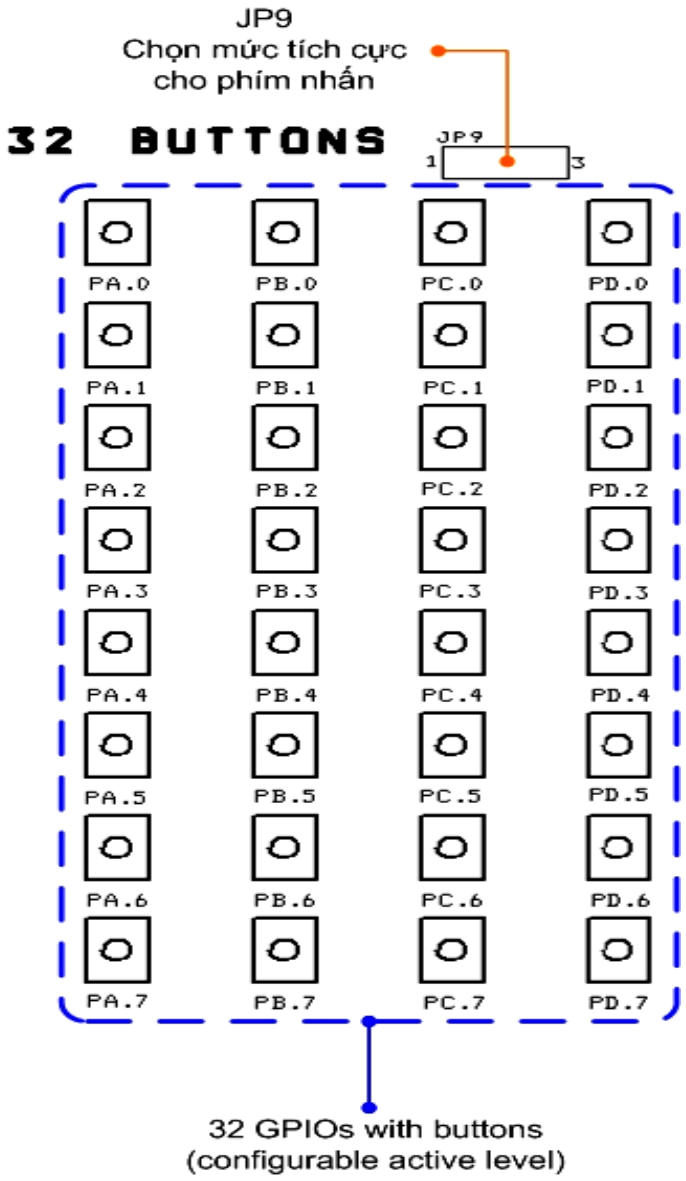
Hình 3.11: Khối Led đơn

❖ **Khối IO với 32 ngõ input phím nhấn**

32 phím nhấn được đưa thẳng vào 32 ngõ IO với mức tích cực có thể hiệu chỉnh được (hi/low) thông qua jumper chọn JP9.

Bảng 3.2: Chọn trạng thái tích cực của nút nhấn

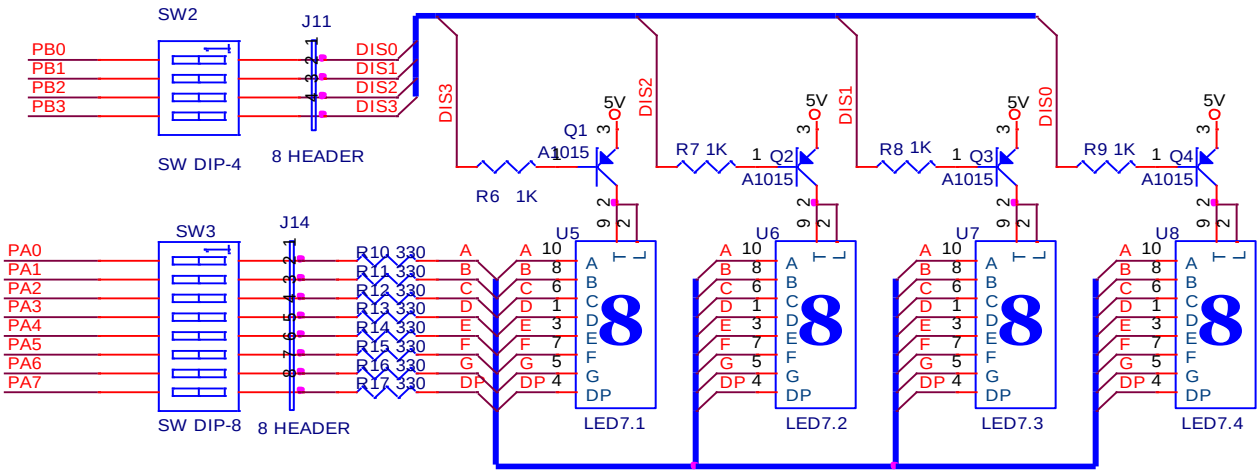
Mức tích cực của phím nhấn	JP9
HIGH	1-2 đóng
LOW	2-3 đóng



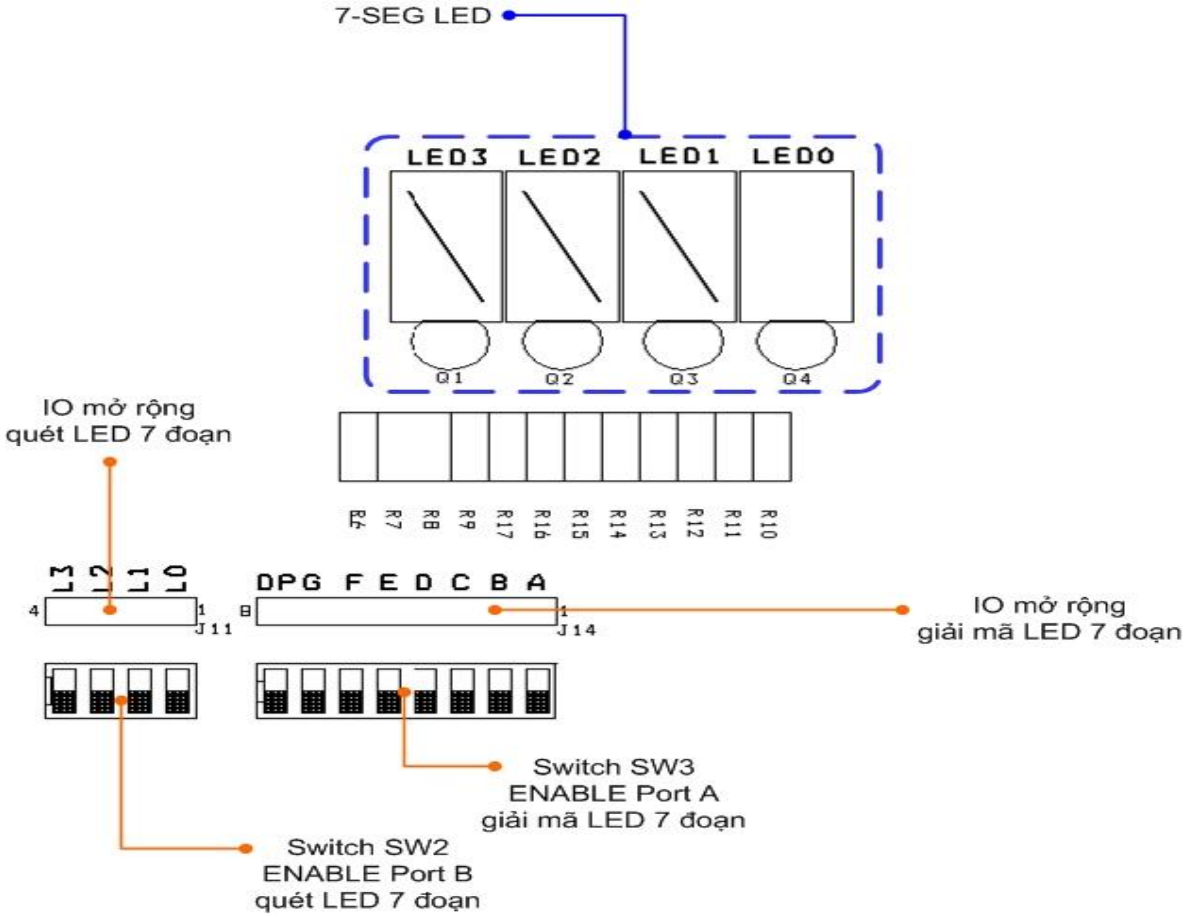
Hình 3.12: Khối nút nhấn

❖ **Khối IO với LED 7 đoạn**

Toàn bộ port A được nối với port mã LED 7 đoạn thông qua switch SW3 và 1 phần port B tham gia quét LED 7 đoạn thông qua switch SW2.



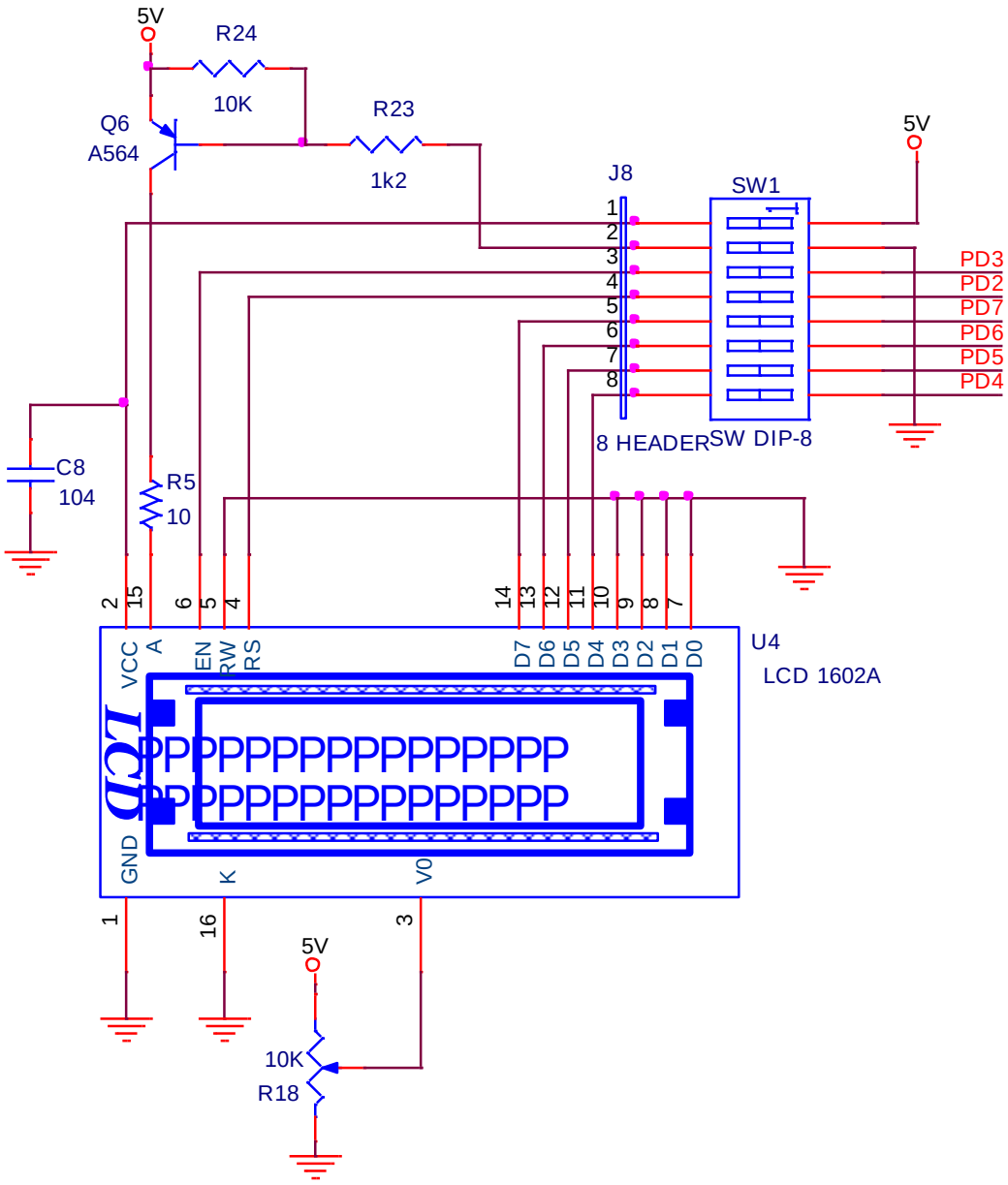
7 SEGMENT LEDS



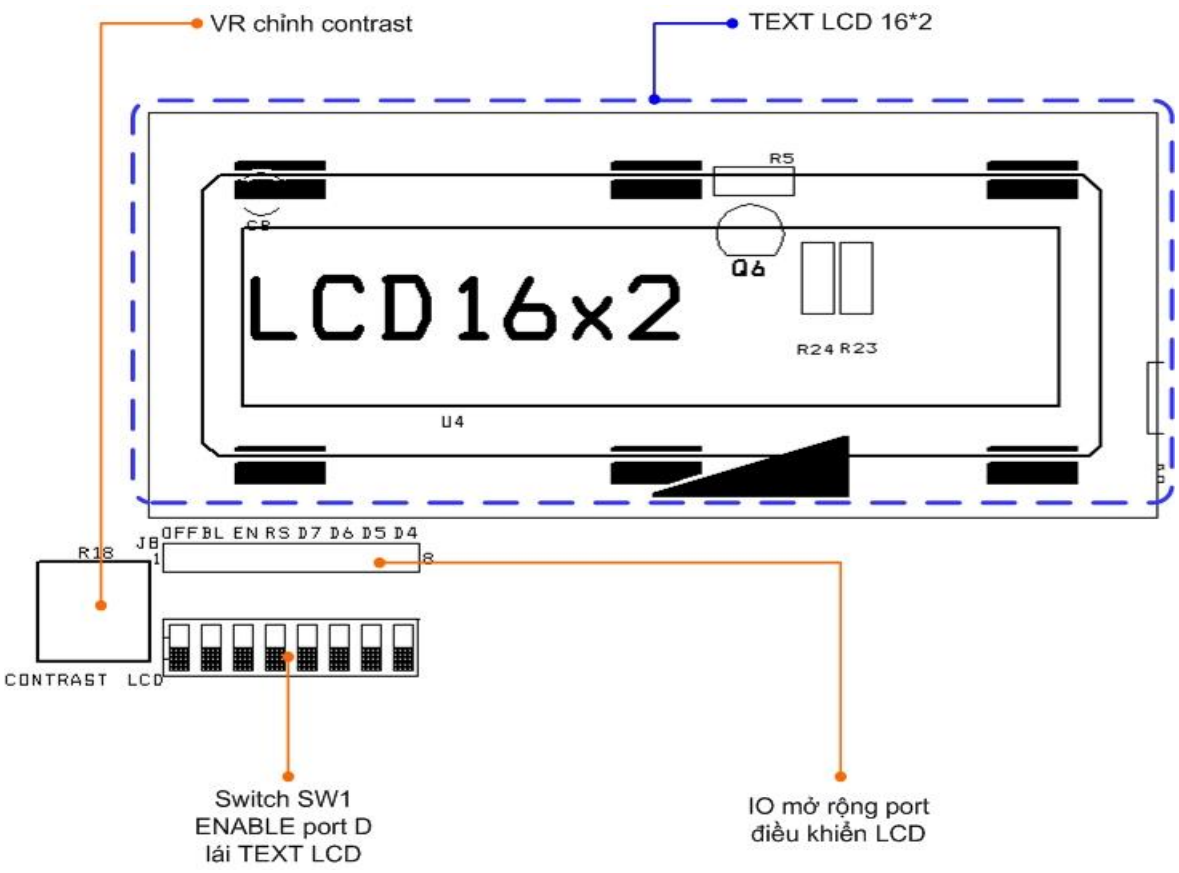
Hình 3.13: Sơ đồ nguyên lý khối Led 7 đoạn

❖ **Khối IO với TEXT LCD 16*2**

1 phần port D tham gia lái TEXT LCD thông qua switch SW1

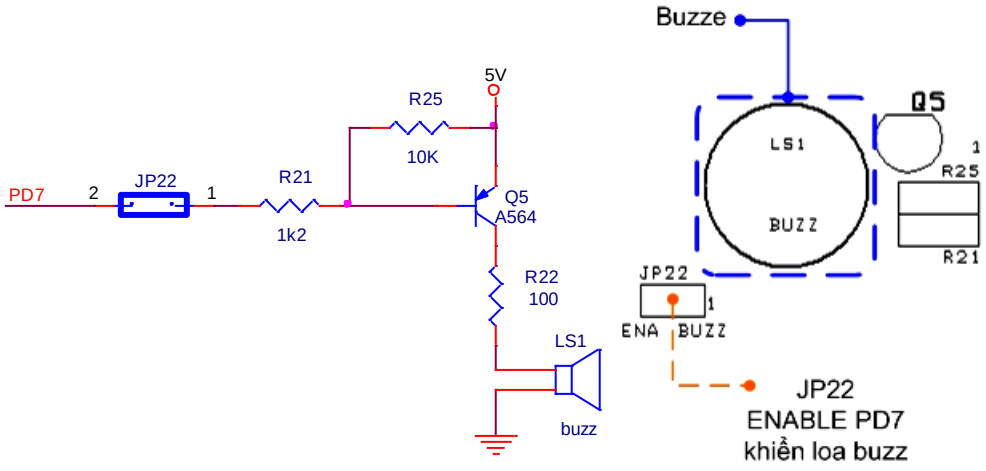


Hình 3.14a: Sơ đồ nguyên lý khối LCD



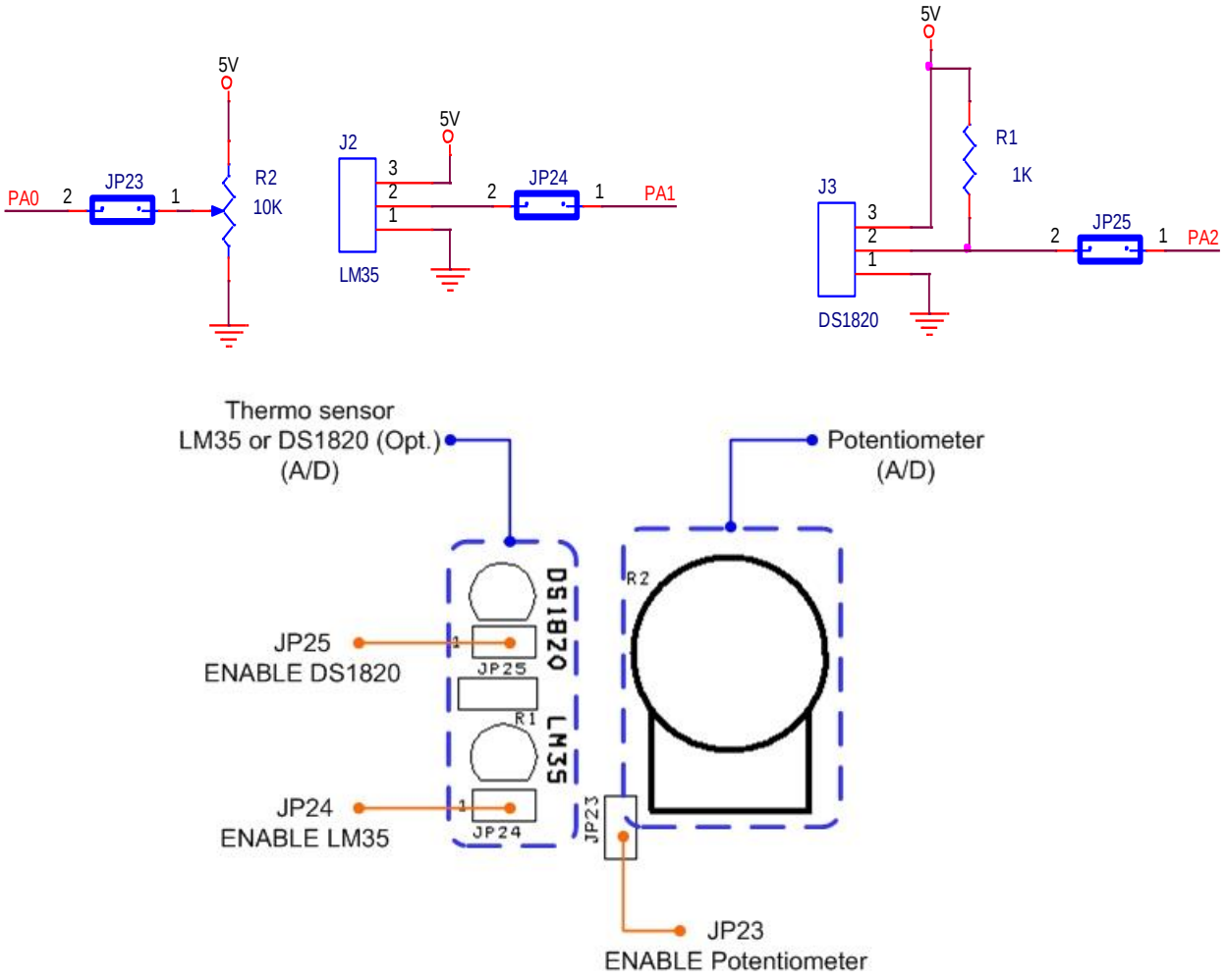
Hình 3.14b: Sơ đồ khối LCD

❖ **Ngõ output với loa buzze**



Hình 3.15: Sơ đồ nguyên lý khối loa buzze

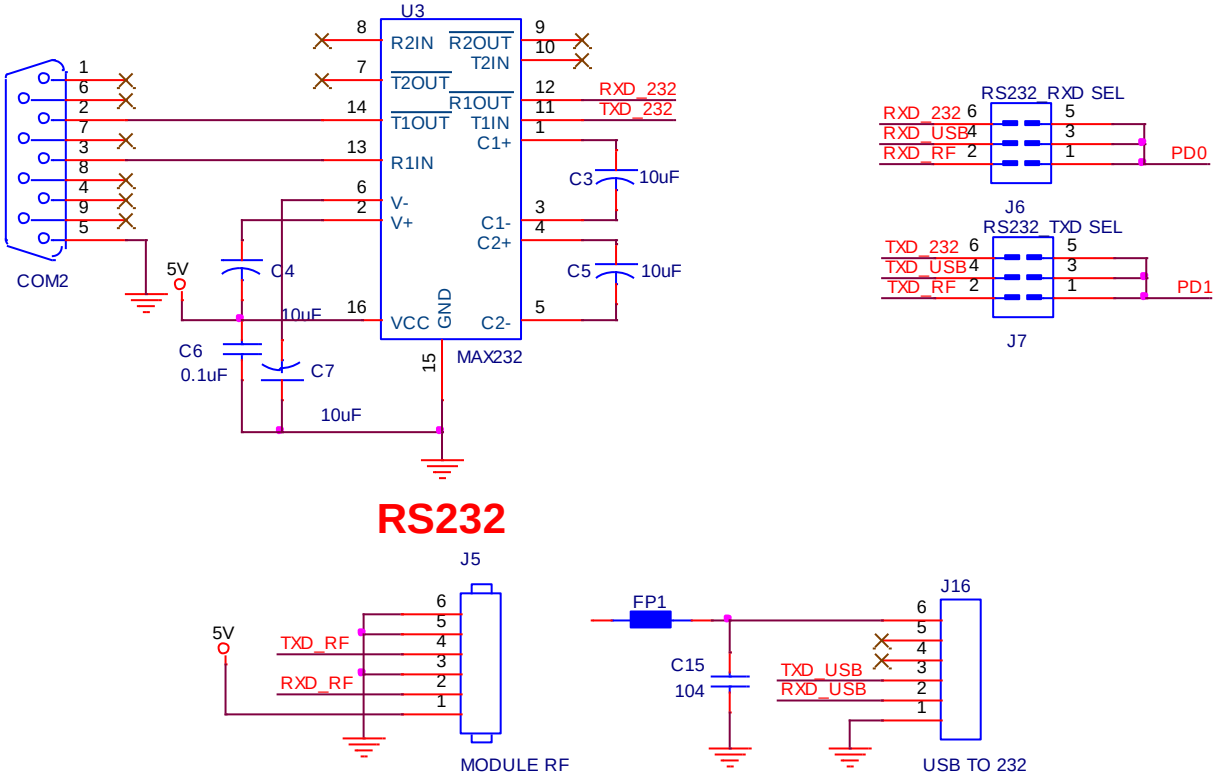
❖ **Khối ADC với LM35 (hoặc DS1820) và biến trở xoay potentiometer**



Hình 3.16: Sơ đồ nguyên lý khối ADC

Biến trở xoay potentiometer và 2 cảm biến nhiệt độ LM35, DS1820 lần lượt nối với 3 kênh ngõ vào bộ chuyển đổi ADC0, ADC1 và ADC2 thông qua các jumper JP23, JP24 và JP25.

❖ **Khối truyền thông UART**



Hình 3.17a: Sơ đồ nguyên lý khối UART

Khối truyền thông nối tiếp UART hỗ trợ 3 chế độ hoạt động:

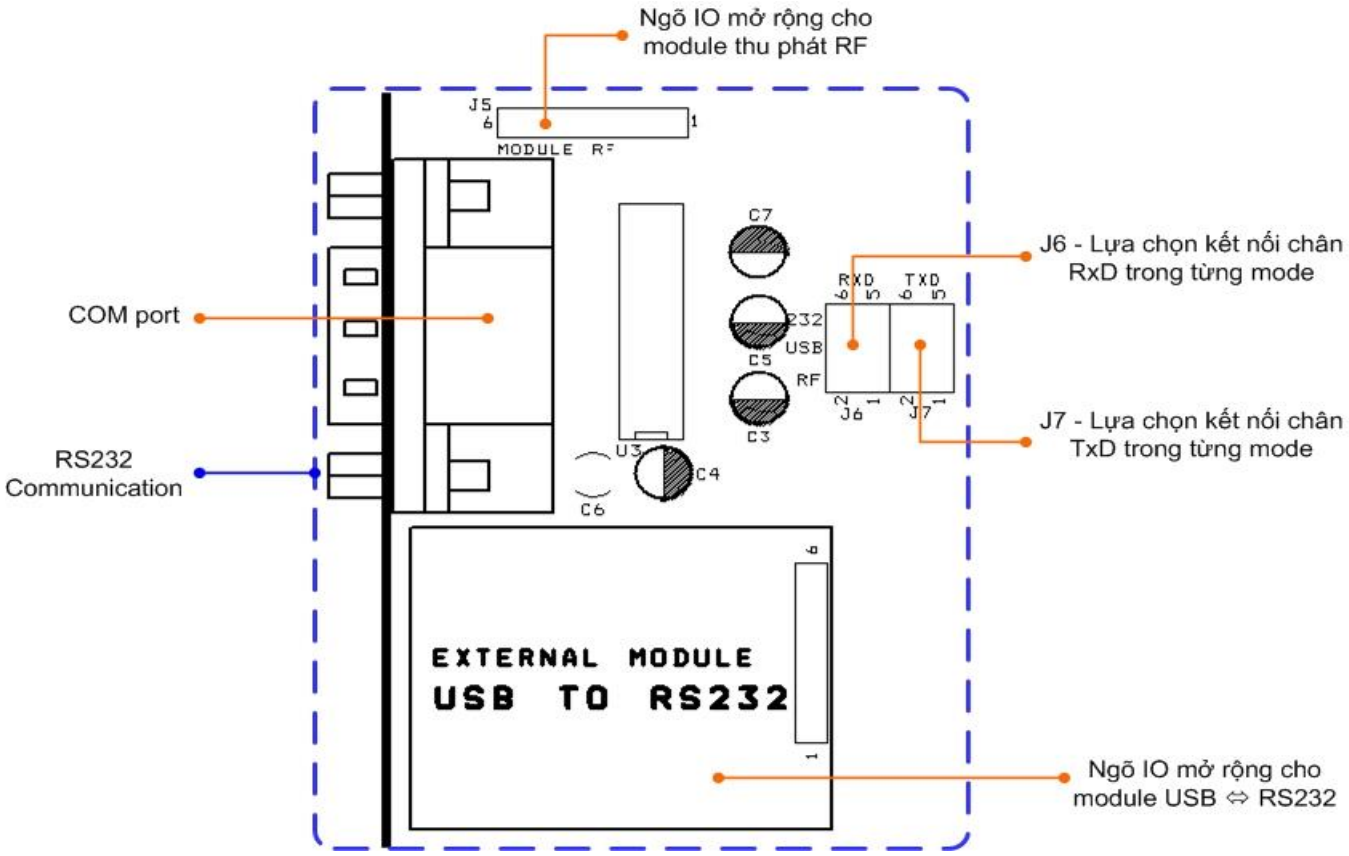
- ✓ Hoạt động thông qua RS232 transceiver (TTL ⇔ CMOS): chức năng UART trên chip ATMEGA32 lúc này kết nối với COM port thông qua RS232 transceiver để chuyển đổi qua lại giữa 2 chuẩn điện áp CMOS và TTL (thường sử dụng kết nối với PC).
- ✓ Hoạt động thông qua module thu phát wireless RF: đây là module rời, nhận tín hiệu RF chuyển đổi sang tín hiệu UART (TTL) và ngược lại để giao tiếp với chức năng UART trên ATMEGA32.
- ✓ Hoạt động thông qua card chuyển USB ⇔ RS232: cũng là module rời, chuyển đổi giao tiếp USB sang RS232. Ở mode này, nguồn cấp cho board có thể lấy trực tiếp từ card chuyển.

2 jumper chọn J6 và J7 phải thiết lập tương ứng với mỗi mode hoạt động trên theo bảng 3.3.

Lưu ý: trên cùng 1 jumper, chỉ duy nhất 1 mode thiết lập ở cùng 1 thời điểm (đóng), các mode còn lại không được kích hoạt (phải để hở).

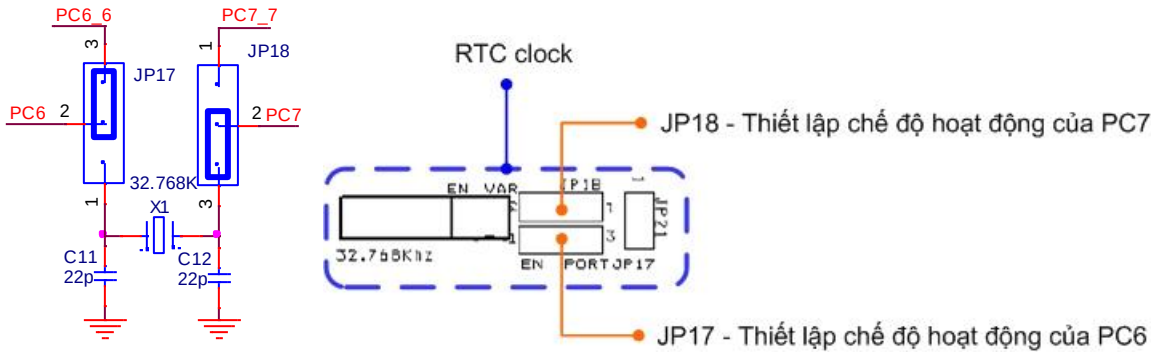
Bảng 3.3: Sử dụng các chế độ UART

Mode	J6	J7
Thông qua RS232 transceiver	5-6 đóng	5-6 đóng
Thông qua card RF	1-2 đóng	1-2 đóng
Thông qua card chuyển USB ⇔ RS232	3-4 đóng	3-4 đóng



Hình 3.17b: Sơ đồ khối UART

❖ **Khối RTC clock**



Hình 3.18: Sơ đồ nguyên lý khối RTC

Chân PC6 và PC7 có thể thiết lập như chân RTC clock input hoặc chân chức năng port thông thường thông qua jumper chọn JP17 và JP18 với chi tiết như trong bảng bên dưới.

Bảng 3.4: Sử dụng các chế độ RTC

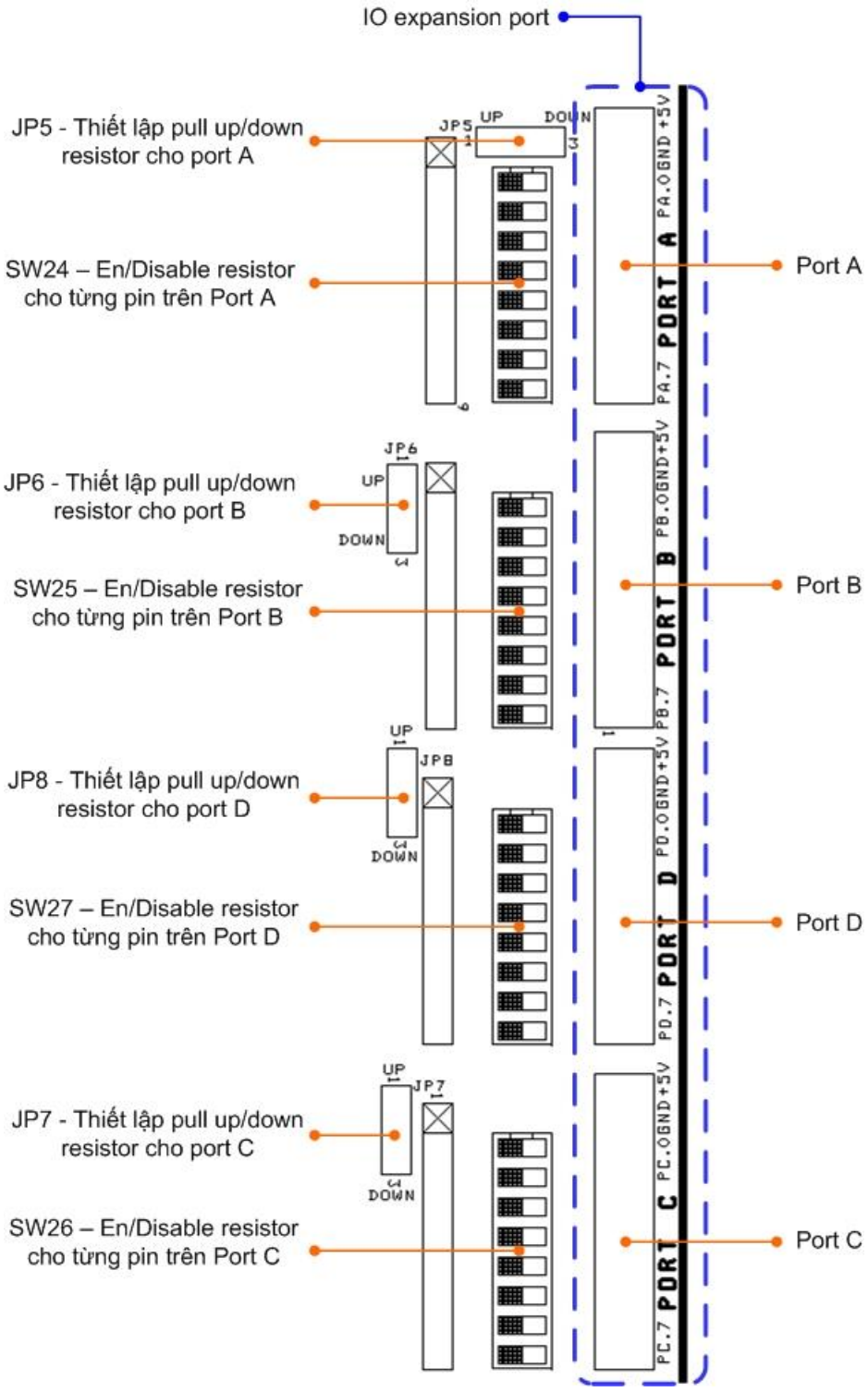
Mode	JP17	JP18
RTC clock input	1-2 đóng	2-3 đóng
Chức năng port thông thường	2-3 đóng	1-2 đóng

❖ **Khối IO mở rộng**

Toàn bộ 32 IO trên ATMEGA32 đều được kết nối trực tiếp ra ngõ IO mở rộng với tùy chọn điện trở pull up/down (thông qua các jumper chọn JP5, JP6, JP7, JP8) có thể thiết đặt độc lập cho từng pin riêng rẽ trên cùng 1 port (thông qua các switch chọn SW24, SW25, SW26 và SW27).

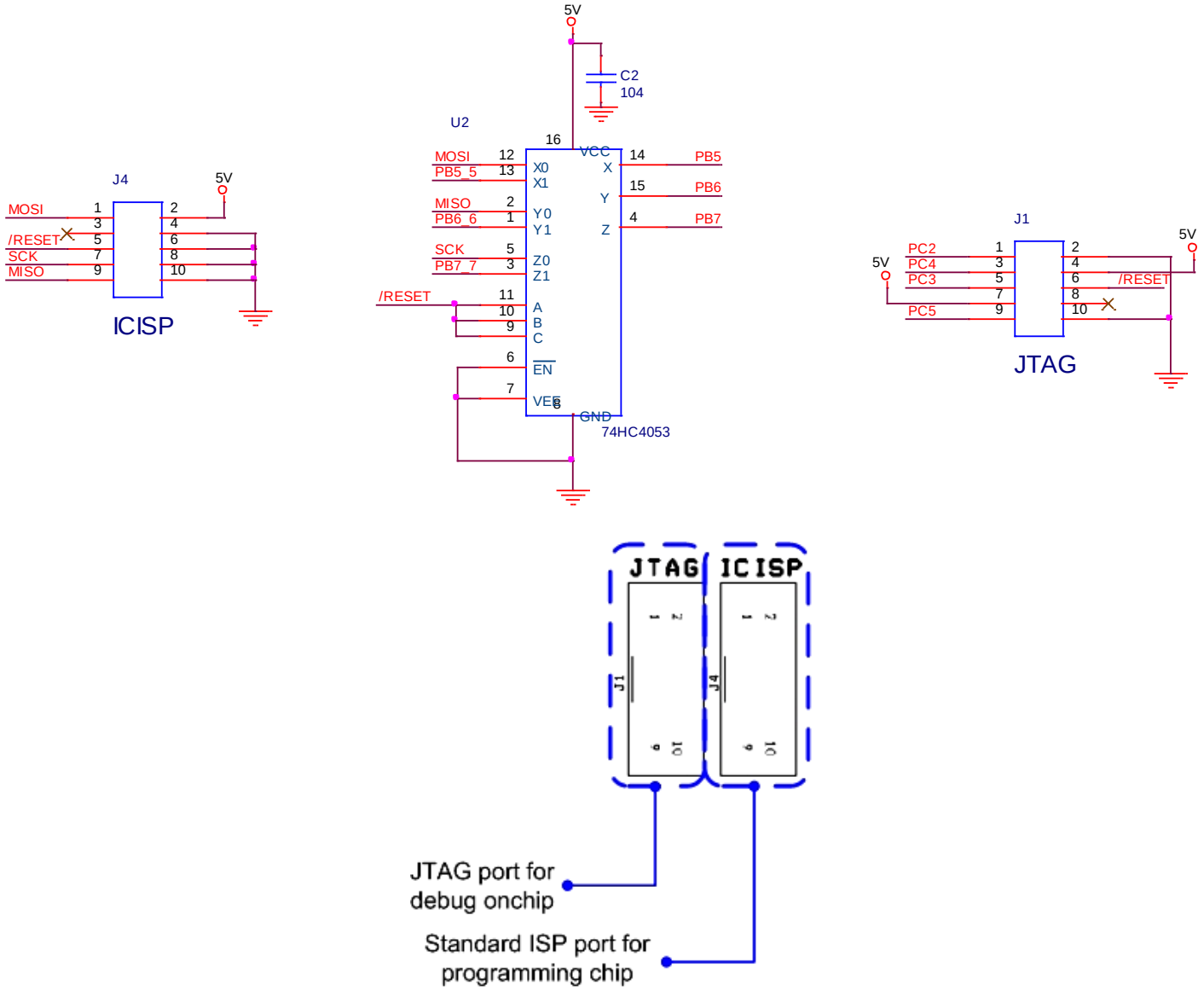
Bảng 3.5: Sử dụng các chế độ điện trở pull up/down

Resistor mode	JP5 cho port A JP6 cho port B JP7 cho port C JP8 cho port D
Pull UP	1-2 đóng
Pull DOWN	2-3 đóng



Hình 3.19: Sơ đồ khối Port mở rộng

❖ **Cổng nạp ISP và cổng debug JTAG**



Hình 3.20: Sơ đồ nguyên lý khối cổng nạp ISP và JTAG

Trên board hỗ trợ 1 cổng nạp tốc độ cao, tương thích hoàn toàn với công cụ nạp USB AVR Prog hoặc STK.

Ngoài ra còn có cổng JTAG phục vụ debug onchip cho các yêu cầu nâng cao trong tương lai.

3.3.3. Các bài thí nghiệm trên sa bàn

- Các bài thí nghiệm có thể xây dựng trên sa bàn bao gồm:
  - Xử lý lọc nhiễu và đọc cảm biến dò line hiển thị ra led đơn trên bộ thí nghiệm.
  - Điều khiển tốc DC motor.

Thực hiện các bài thí nghiệm AVR cơ bản như: led đơn, led 7đoạn, LCD, nút nhấn, ADC...

CHƯƠNG 4:

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

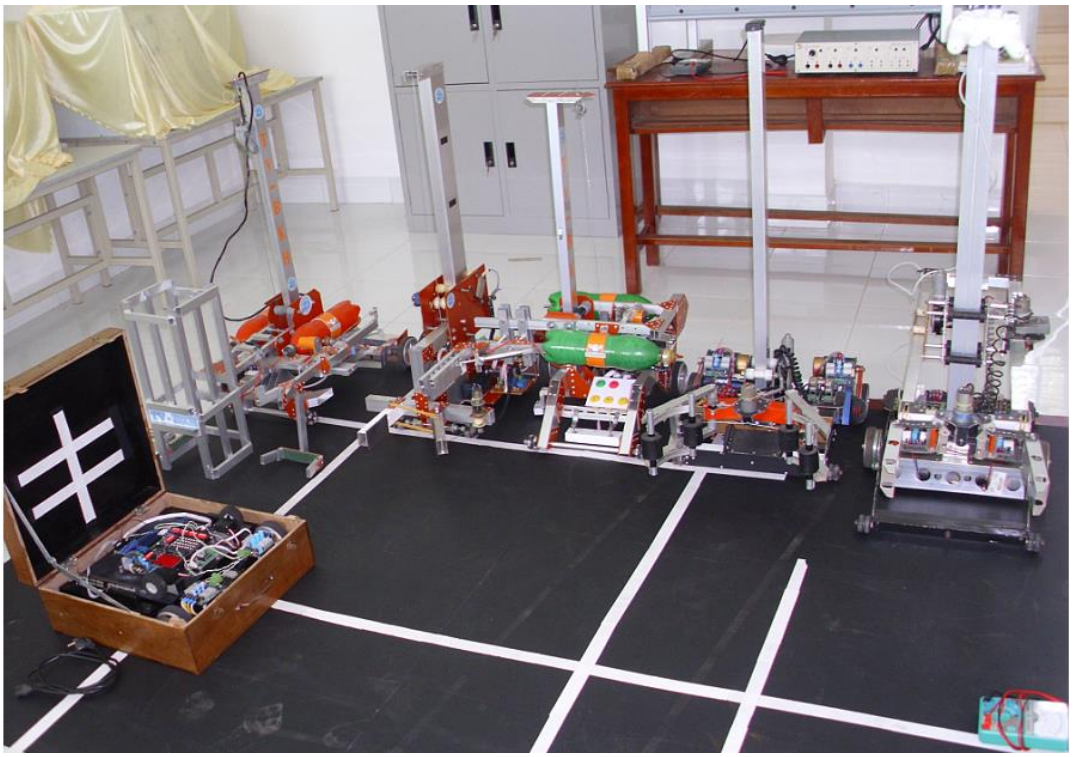
4.1. Kết đạt được

Đề tài đã hoàn thành với 6 sản phẩm được tạo ra vượt mức đăng ký ban đầu. Các sản phẩm của đề tài bao gồm:

- 1 sa bàn thí nghiệm Robot tự động.
- 3 Robot tự động.
- 2 Robot điều khiển bằng tay.

Trong đó, có 2 robot tự động và 1 robot điều khiển bằng tay đã tham gia vào cuộc thi Robocon cấp Trường của Khoa Kỹ thuật và Công nghệ tổ chức vừa qua.

❖ **Các sản phẩm đã thi công:**



Hình 4.1: Các sản phẩm của đề tài

4.2. Ưu khuyết điểm

- **Ưu điểm:**
  - Các sản phẩm được thiết kế chắc chắn, bền đẹp
  - Hoạt động tốt và ổn định cao
  - Có tháo ráp hoặc cải tiến dễ dàng
  - Làm mô hình thí nghiệm trực quan, giúp ích cho việc giảng dạy và nghiên cứu.
- **Khuyết điểm:**
  - Các mô hình robot thiết kế đơn giản.
  - Các robot chỉ thực hiện các bài thí nghiệm đơn giản như dò đường, gấp thả vật.
  - Chưa giao tiếp được modul Analog của tay game.

4.3. Hướng phát triển của đề tài

– Tiếp tục nghiên cứu, cải tiến mô hình cơ khí và chương trình điều khiển để robot có khả năng hoạt động linh hoạt hơn.

– Thực hiện giao tiếp modul Analog của tay game.

– Nghiên cứu các kỹ thuật dò đường khác như la bàn số, xử lý ảnh…

TÀI LIỆU THAM KHẢO

[1] Ngô Diên Tập (2003), *Kỹ Thuật Vi điều Khiển Với AVR*, NXB Khoa học và Kỹ thuật.

[2] Gerhard Schmidt (2003), *Beginners Introduction to theAssembly Language of ATMEL-AVR-Microprocessors*, Avr-Asm-Tutorial.

[3] Pavel Haiduc (2008), *CodevisionAVR V2.03.5 User Manual*, HP InfoTech S.R.L.

[4] Richard Barnett, Larry O'Cull, Sarah Cox (2006), *Embedded C Programming and the Atmel AVR*, Canada.

Trang web

[5] http://hlab.com.vn/index.php?option=com_content&view=article&id=45%3Ak-thut-do-dng-cho-robocon&catid=38%3Arobocon&Itemid=106&lang=vi (20/8/2012)

[6] http://store.curiousinventor.com/guides/PS2/#hardware (20/8/2012)

[7] http://www.gamesx.com/controldata/psxcont/psxcont.htm#DATA (20/8/2012)

[8] http://www.hocavr.com (20/8/2012)

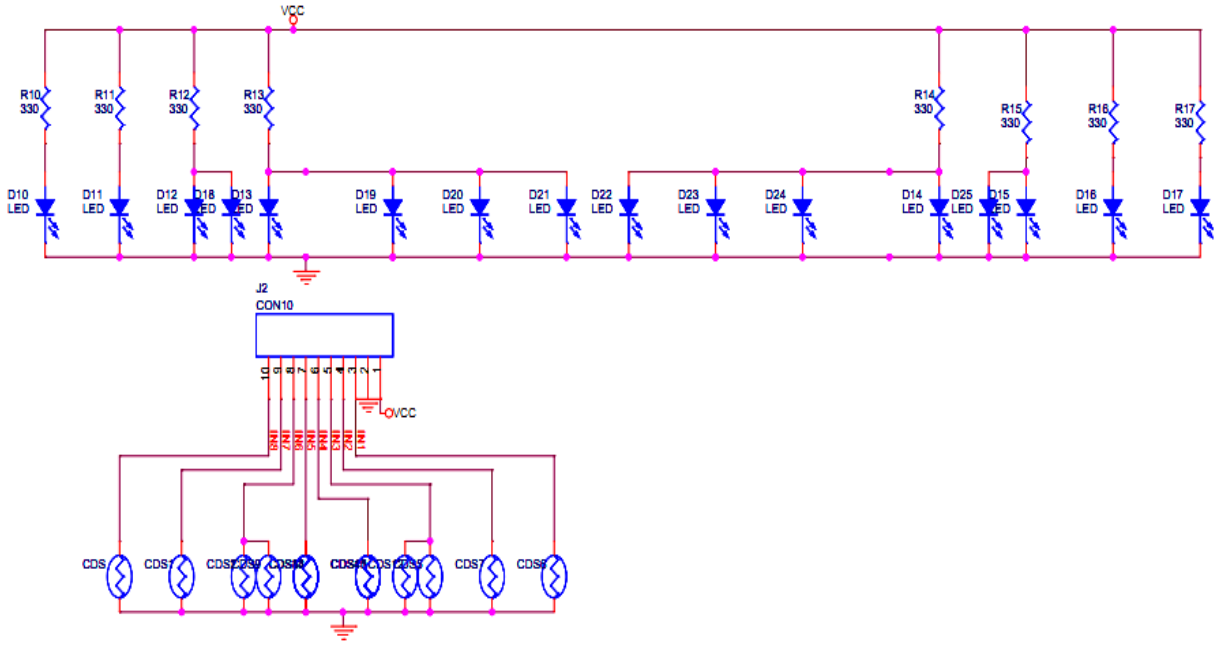
[9] http://dks.edu.vn/?course_sid=c9f0f895fb98ab9159f51fd0297e236d

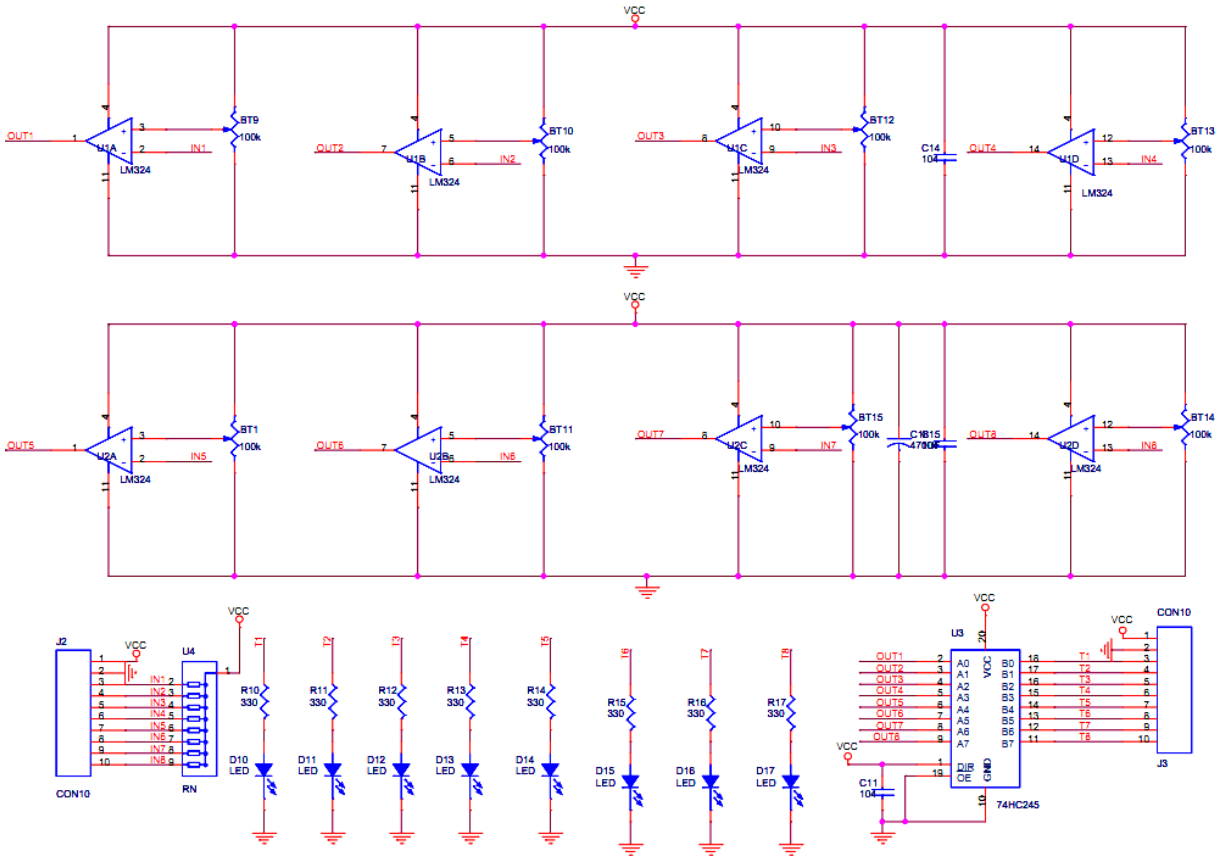
[10] http://www.360-books.com/ebooks/book-store/khoa-hoc-ky-thuat/giao-trinh-avr.html (20/8/2012)

PHỤ LỤC 1

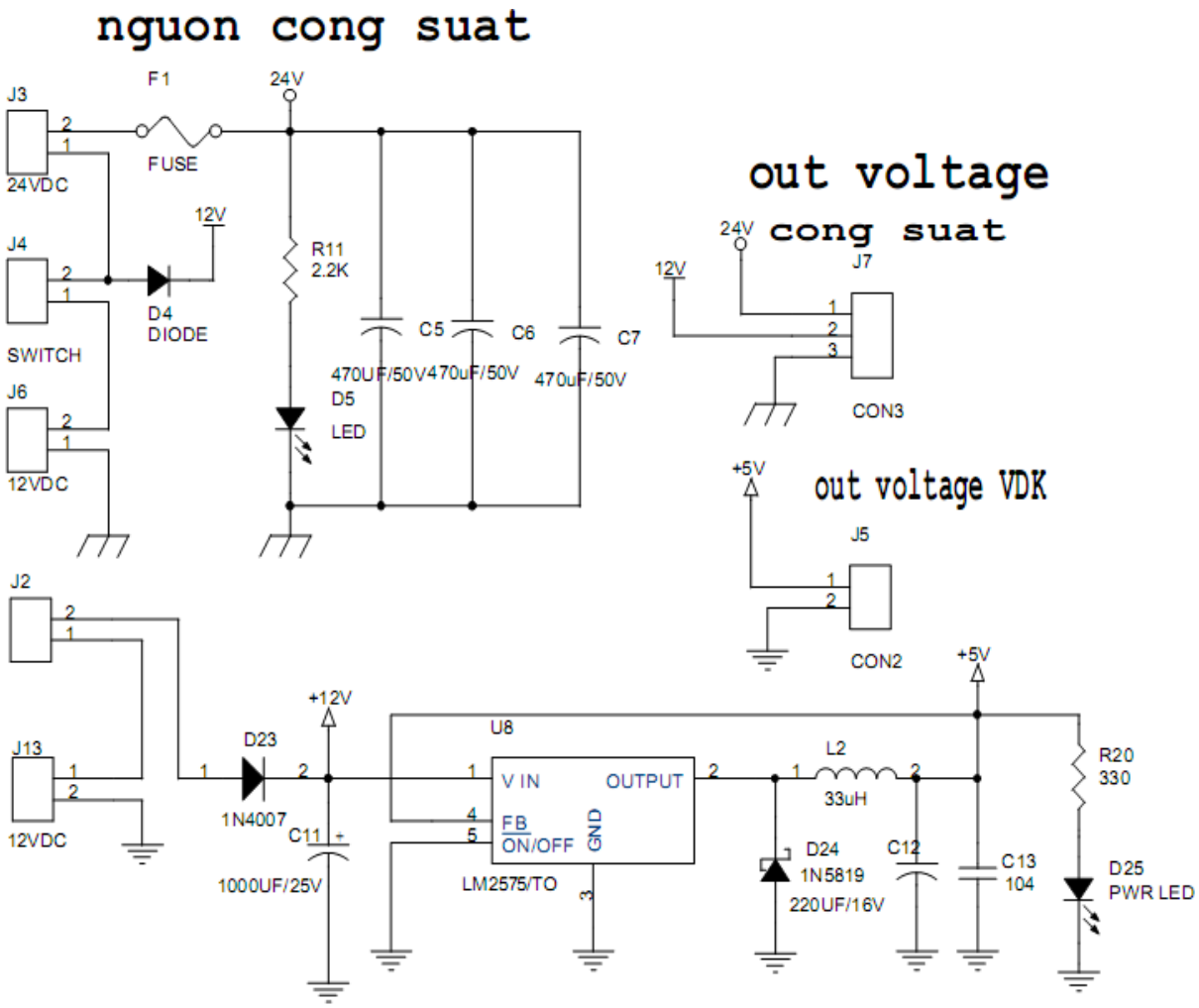
CÁC SƠ ĐỒ MẠCH ĐIỆN

1. Board cảm biến dò line

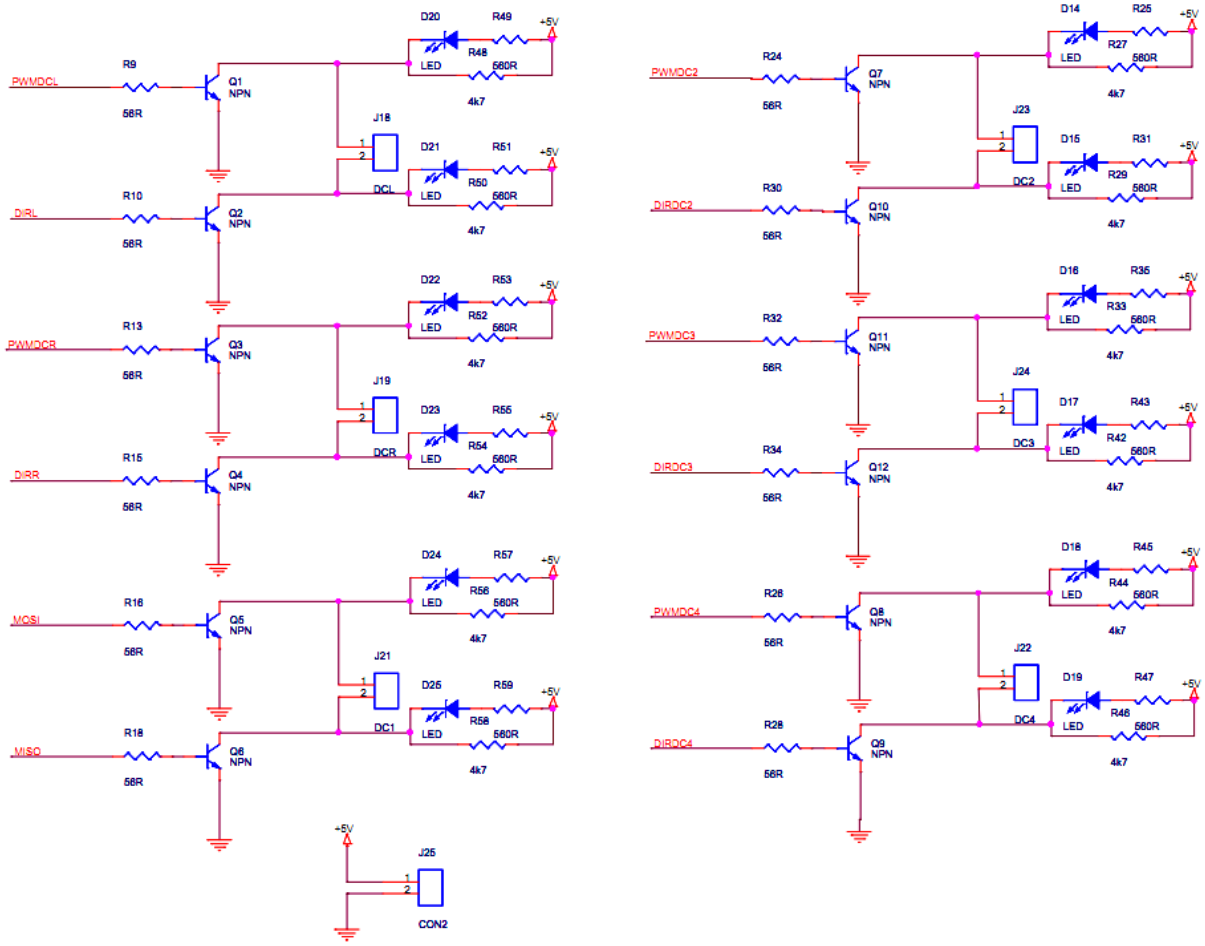


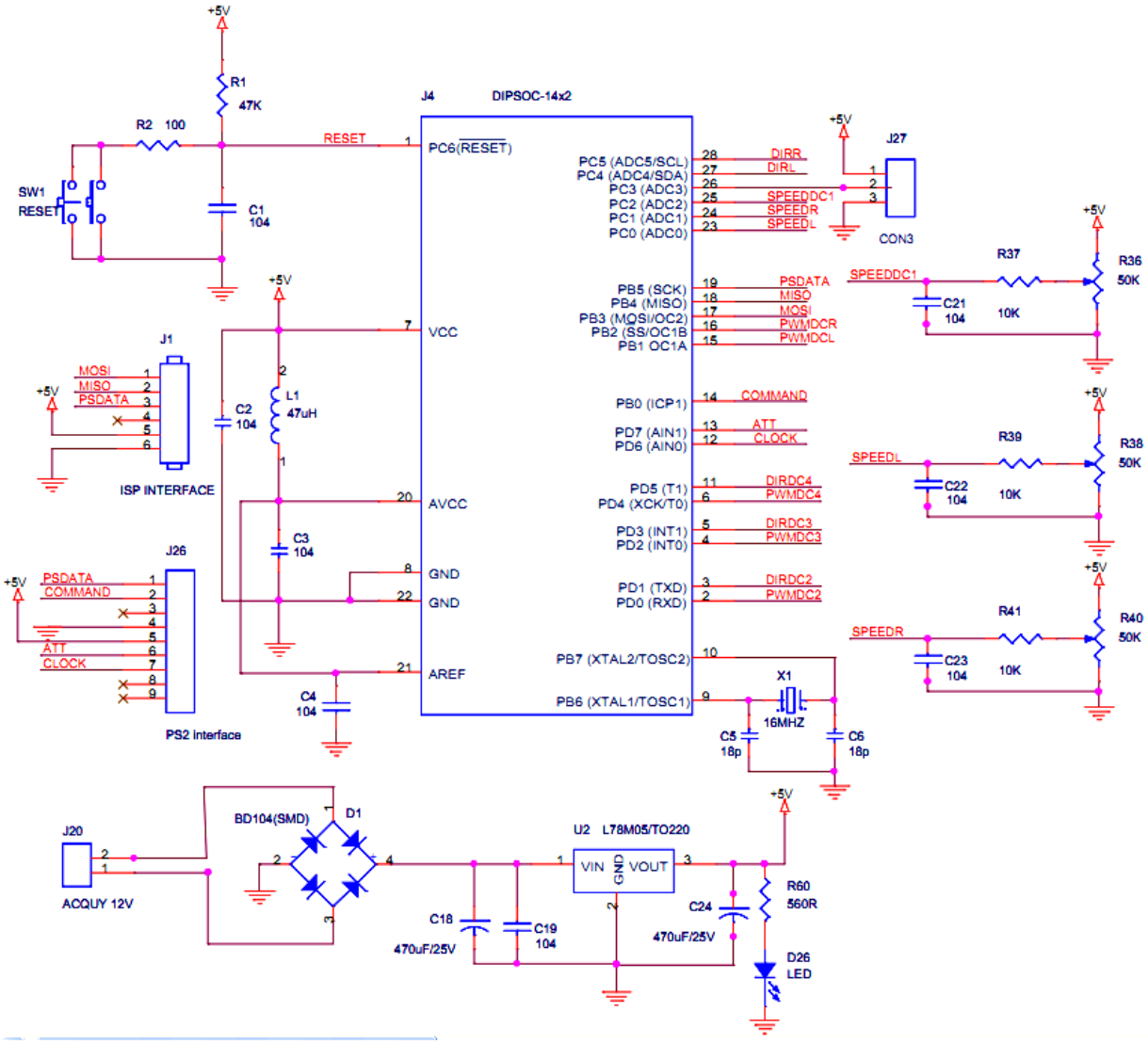

VCC
OUT1
U1A
LM324
IN1
BT9
100k
OUT2
U1B
IN2
BT10
OUT3
U1C
IN3
BT12
C14
104
OUT4
U1D
IN4
BT13
OUT5
U2A
IN5
BT1
OUT6
U2B
IN6
BT11
OUT7
U2C
IN7
BT15
C15
OUT8
U2D
IN8
BT14
J2
CON10
U4
RN
R10
330
D10
LED
R11
D11
R12
D12
R13
D13
R14
D14
R15
D15
R16
D16
R17
D17
U3
74HC245
C11
104
J3

2. Board mạch nguồn công suất

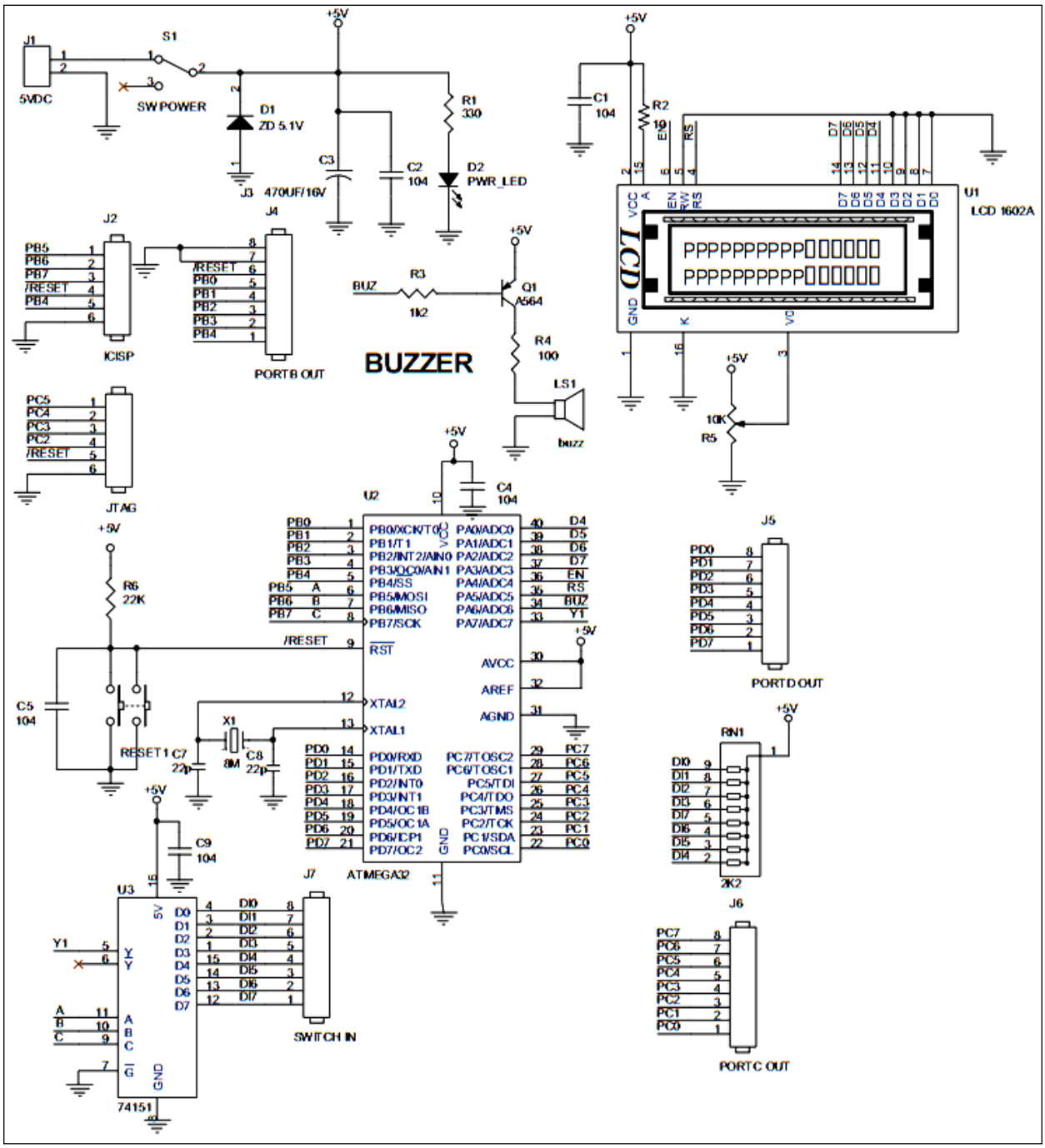


3. Board điều khiển Manual Robot

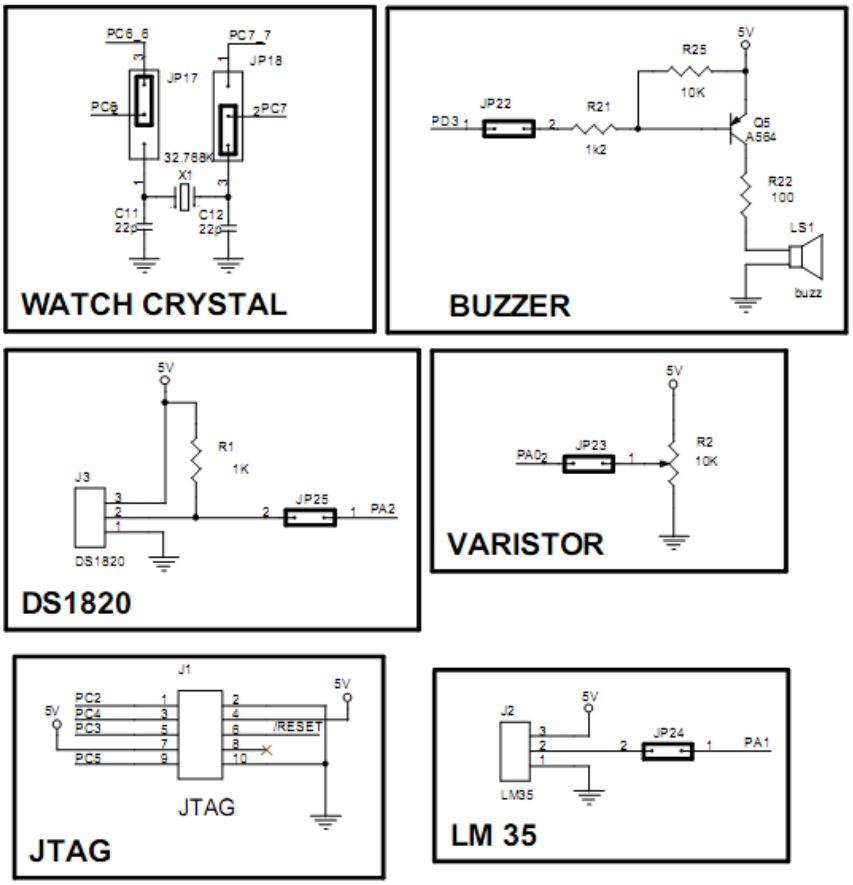


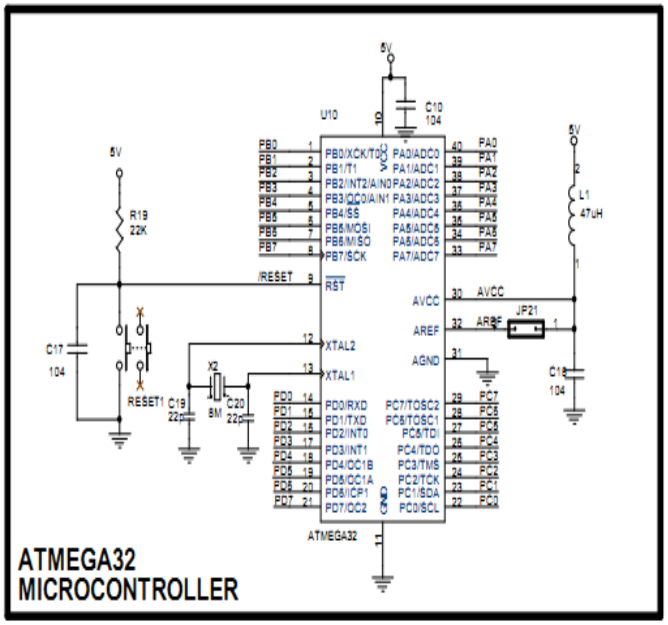

+5V
R1
47K
R2 100
RESET
SW1
RESET
C1
104
J4
DIPSOC-14x2
PC6(RESET)
PC5 (ADC5/SCL)
PC4 (ADC4/SDA)
PC3 (ADC3)
PC2 (ADC2)
PC1 (ADC1)
PC0 (ADC0)
DIRR
DIRL
SPEEDDC1
SPEEDR
SPEEDL
J27
CON3
J1
MOSI
MISO
PSDATA
ISP INTERFACE
C2
104
L1
47uH
C3
104
VCC
AVCC
GND
GND
AREF
C4
104
PB5 (SCK)
PB4 (MISO)
PB3 (MOSI/OC2)
PB2 (SS/OC1B)
PB1 OC1A
PSDATA
MISO
MOSI
PWMDCR
PWMDCL
PB0 (ICP1)
COMMAND
PD7 (AIN1)
PD6 (AIN0)
ATT
CLOCK
PD5 (T1)
PD4 (XCK/T0)
DIRDC4
PWMDC4
PD3 (INT1)
PD2 (INT0)
DIRDC3
PWMDC3
PD1 (TXD)
PD0 (RXD)
DIRDC2
PWMDC2
PB7 (XTAL2/TOSC2)
PB6 (XTAL1/TOSC1)
X1
16MHZ
C5
18p
C6
18p
J26
PSDATA
COMMAND
ATT
CLOCK
PS2 Interface
R37
R36
50K
C21
104
10K
SPEEDDC1
R39
R38
50K
C22
104
10K
SPEEDL
R41
R40
50K
C23
104
10K
SPEEDR
J20
ACQUY 12V
BD104(SMD)
D1
U2 L78M05/TO220
VIN
GND
VOUT
C18
470uF/25V
C19
104
C24
470uF/25V
R60
560R
D26
LED

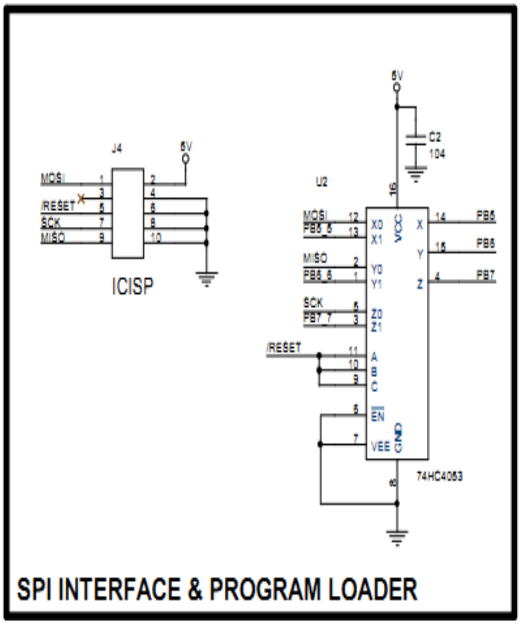
4. Board điều khiển Auto Robot

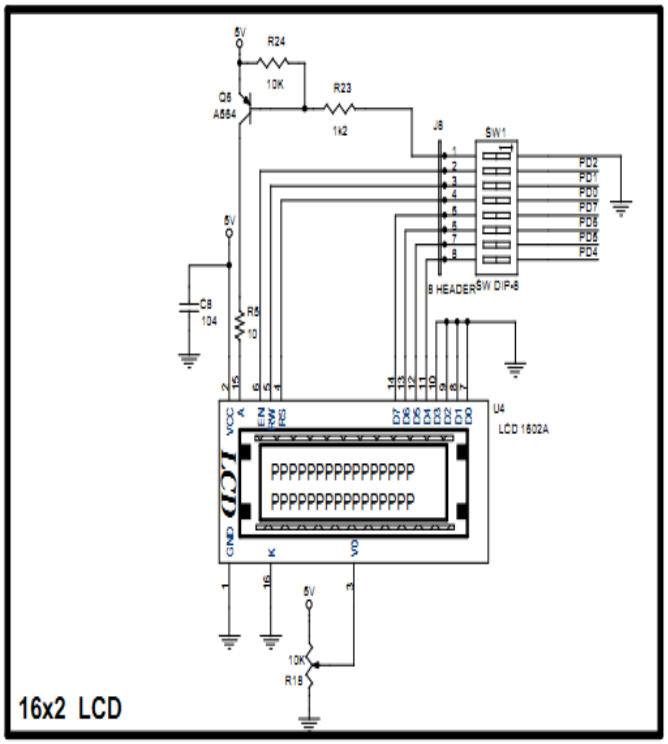


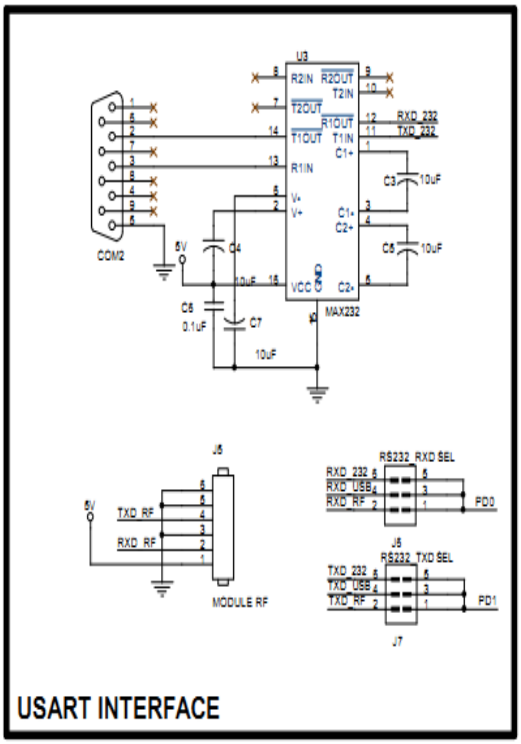
5. Board mạch bộ thí nghiệm

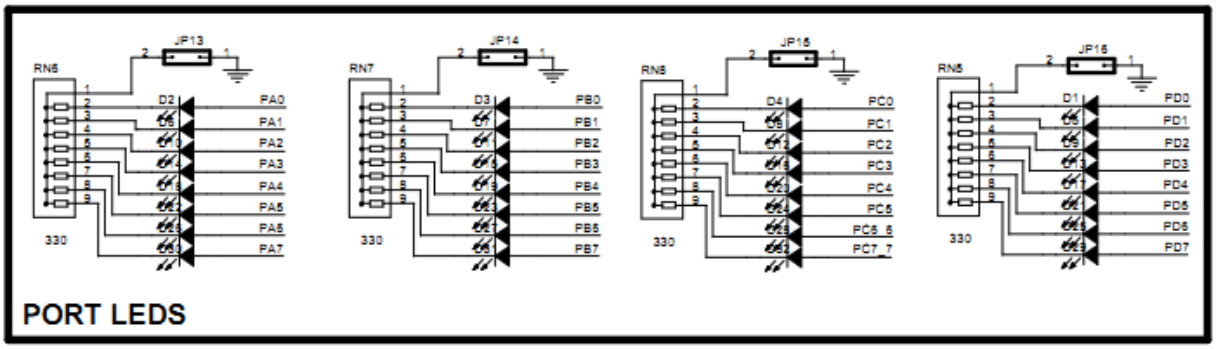


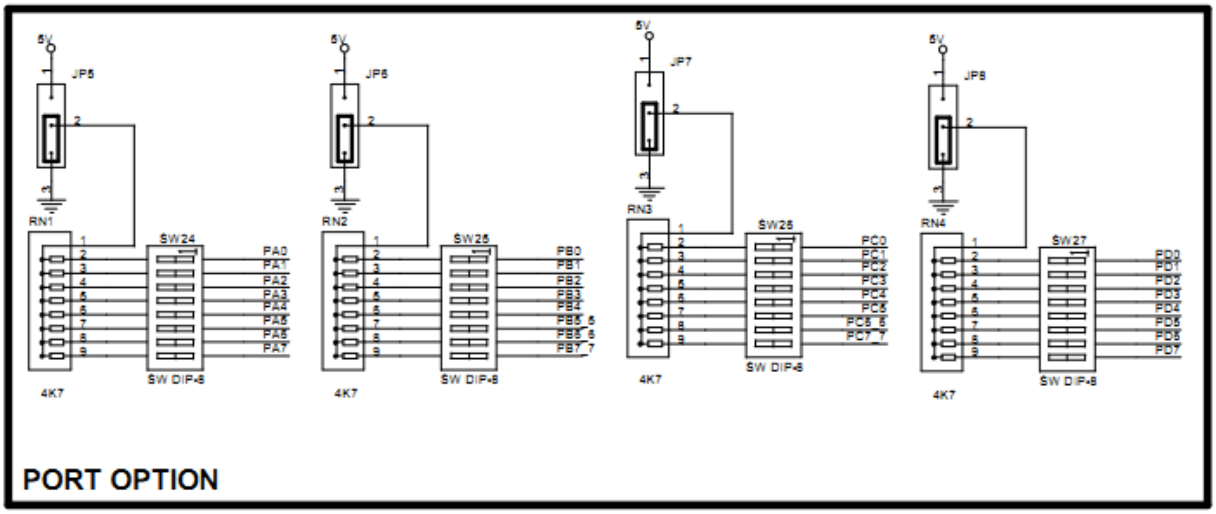

ATMEGA32
MICROCONTROLLER

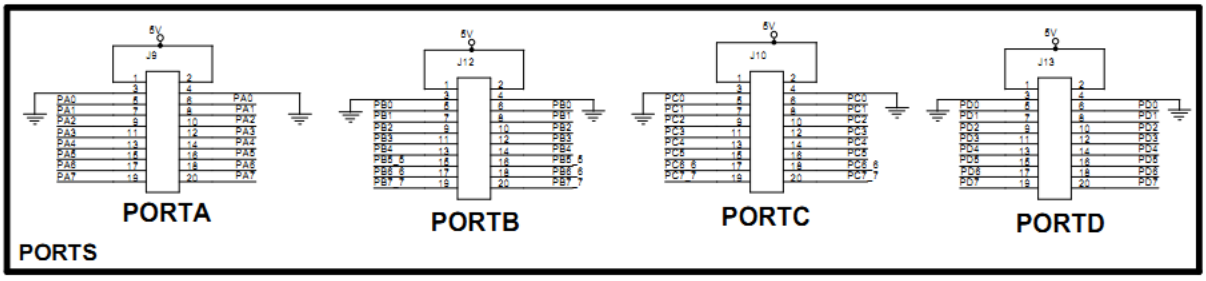

ICISP
74HC4053
SPI INTERFACE & PROGRAM LOADER

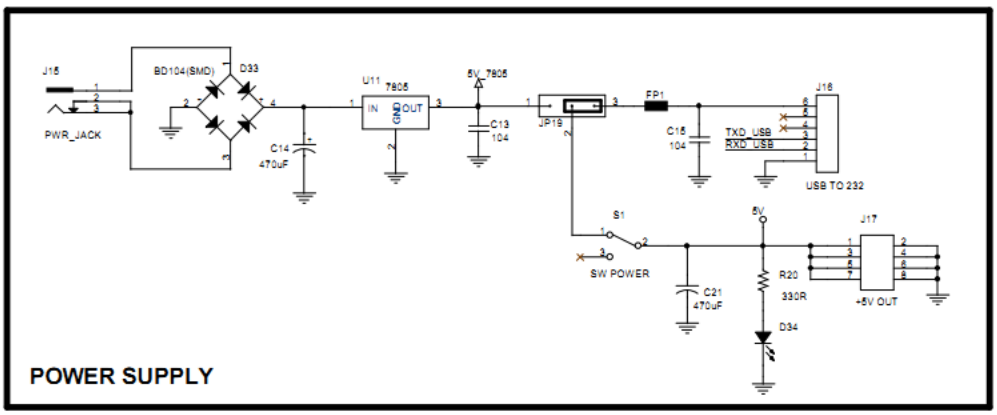

LCD 1602A
16x2 LCD

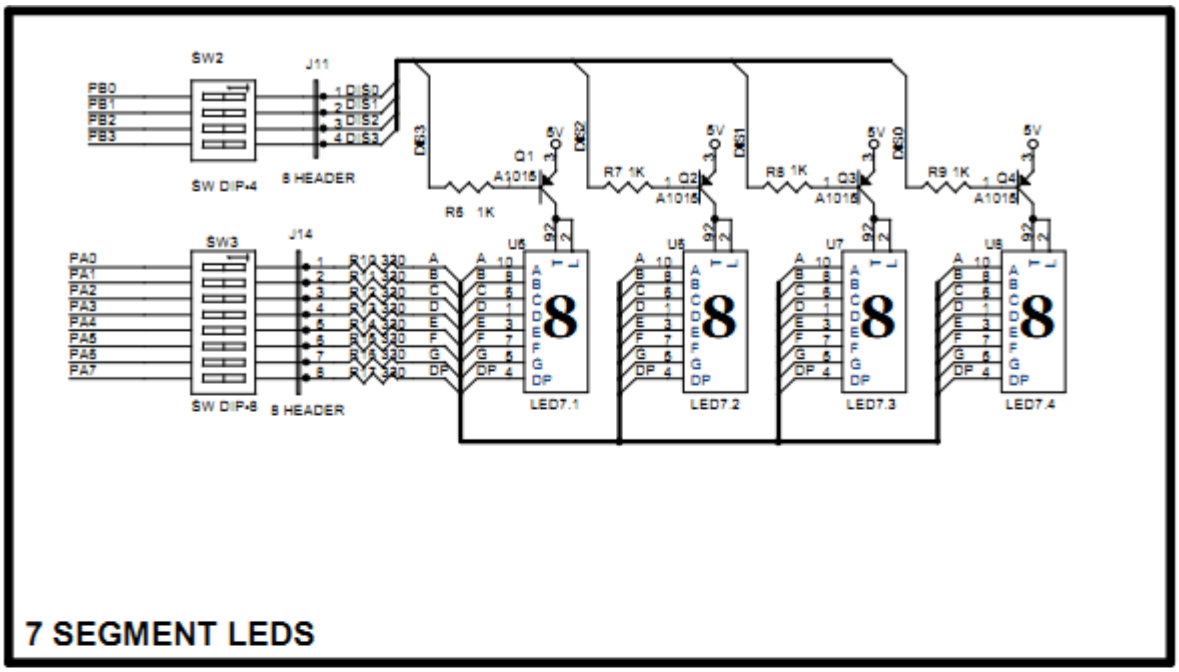

MAX232
MODULE RF
USART INTERFACE

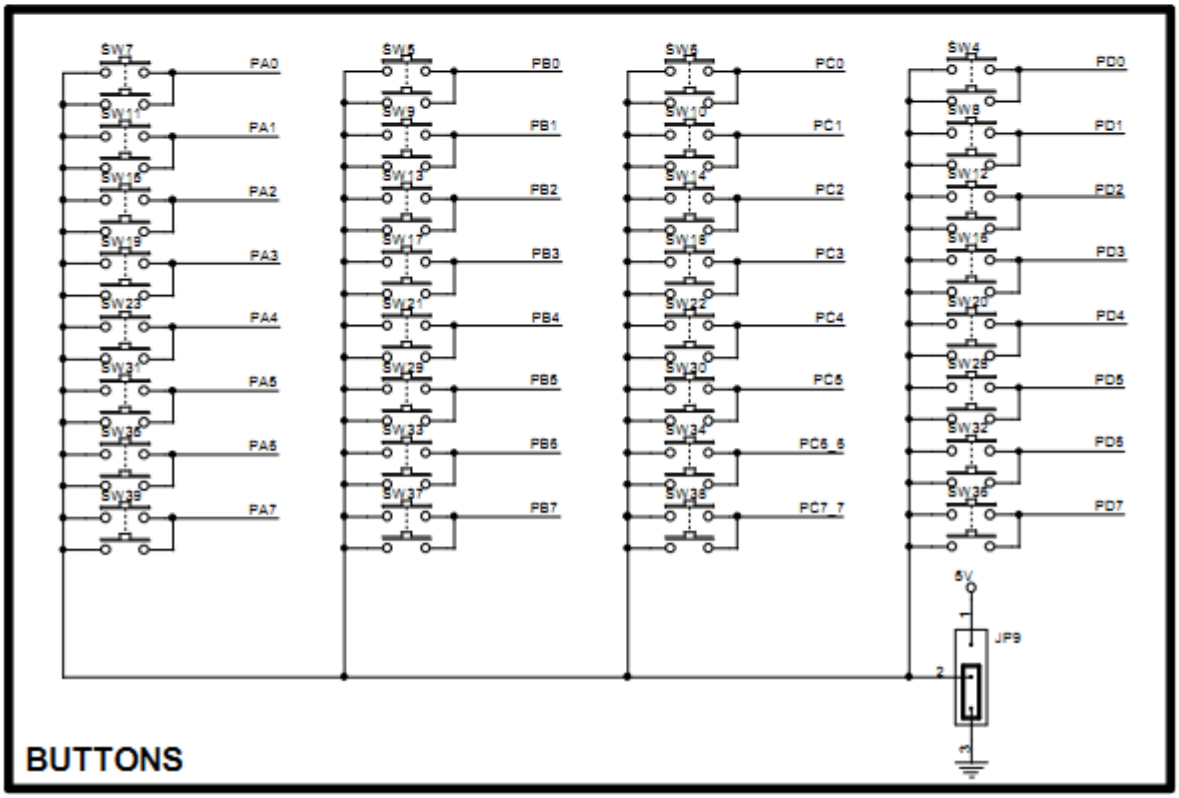

JP13
JP14
JP15
JP16
RN5
RN7
RN8
RN6
D2
D3
D4
D1
330
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6_6
PC7_7
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
PORT LEDS


5V
JP5
JP6
JP7
JP8
RN1
RN2
RN3
RN4
SW24
SW25
SW26
SW27
4K7
SW DIP-8
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5_5
PB6_6
PB7_7
PC0
PC1
PC2
PC3
PC4
PC5
PC6_6
PC7_7
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
PORT OPTION

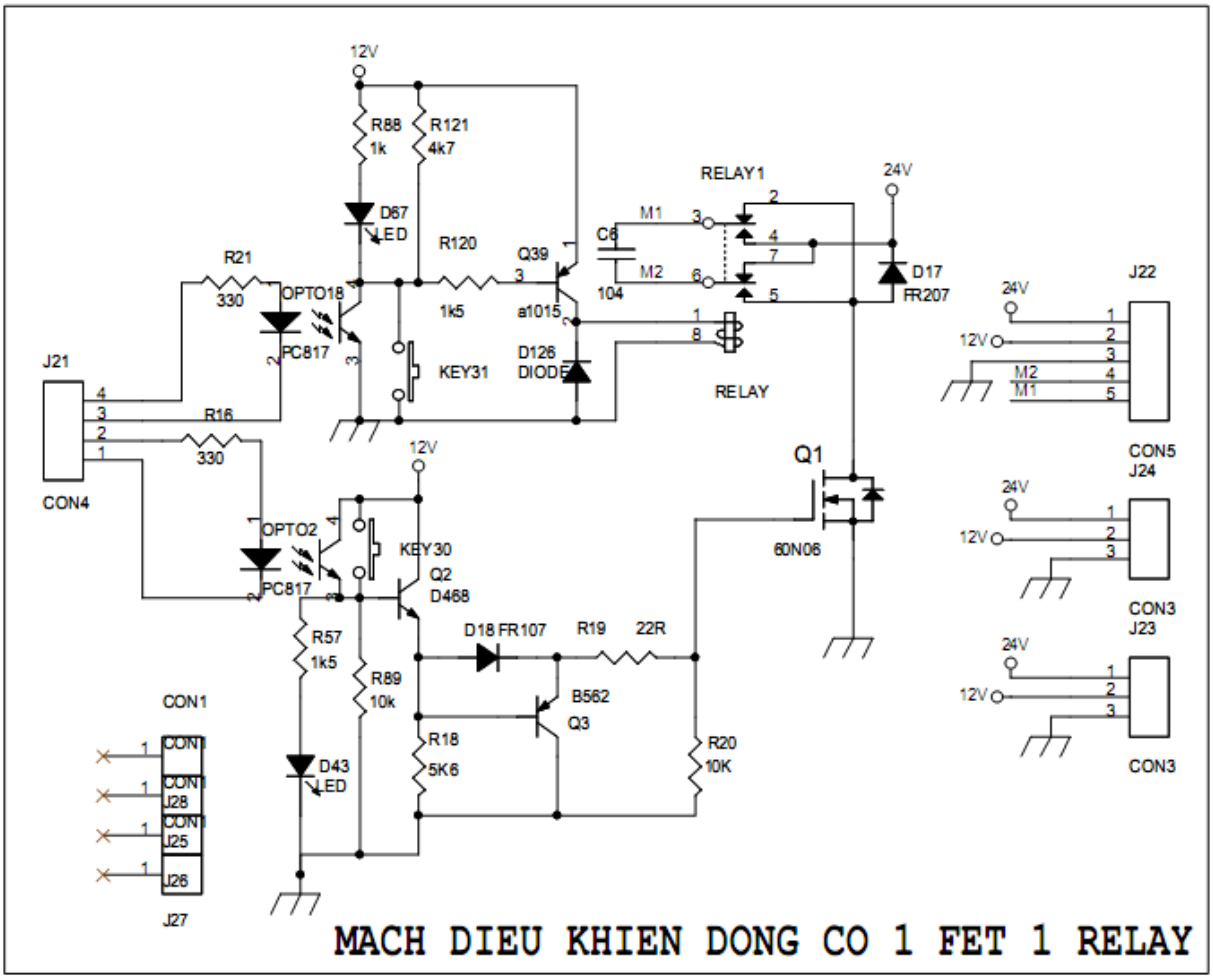

5V
J9
J12
J10
J13
PORTA
PORTB
PORTC
PORTD
PORTS


J15
PWR_JACK
BD104(SMD)
D33
U11
7805
5V_7805
IN
GND
OUT
C14
470uF
C13
104
JP19
FP1
C15
104
TXD_USB
RXD_USB
J16
USB TO 232
S1
SW POWER
C21
470uF
R20
330R
J17
+5V OUT
D34
POWER SUPPLY


SW2
J11
PB0
PB1
PB2
PB3
SW DIP-4
8 HEADER
R5 1K
R7 1K
R8 1K
R9 1K
Q1
Q2
Q3
Q4
A1015
SW3
J14
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
SW DIP-8
8 HEADER
LED7.1
LED7.2
LED7.3
LED7.4
7 SEGMENT LEDS


PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6_6
PC7_7
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
JP9
BUTTONS

6. Các Board mạch công suất



PHỤ LỤC 2

CHƯƠNG TRÌNH ĐIỀU KHIỂN

1. Chương trình điều khiển Robot tự động

```
*****************************************************/

#include <mega32.h>
#include <delay.h>
////////////////////////////////////////////////////
#define sensor  PINC
///////////////////////////////////////////////////
#define pwmL    OCR1A
#define DirL    PORTD.3
///////////////////////////////////////////////////
#define pwmR    OCR1B
#define DirR    PORTD.2
//////////////////////////////////////////////////
#define pwmDC_up_down   PORTD.6
#define Dir_up_down   PORTD.7
/////////////////////////////////////////////////
#define pwmDC_kep_tha  PORTD.1
#define Dir_kep_tha    PORTD.0
/////////////////////////////////////////////////
#define limit_kep   PINB.2
#define limit_tha   PINB.4
#define limit_down   PINB.3
////////////////////////////////////////////////
#define p7 0b00000001
#define p6 0b00000011
#define p5 0b00000010
#define p4 0b00000110
#define p3 0b00000100
#define p2 0b00001100
#define p1 0b00001000
#define tt 0b00011000
#define t1 0b00010000
#define t2 0b00110000
#define t3 0b00100000
```

```
#define t4 0b01100000
#define t5 0b01000000
#define t6 0b11000000
#define t7 0b10000000
#define rangoai 0b00000000
#define ngatu 0b11111111
//////////////////////////////////////////////

bit erL,erR;
void speed(unsigned char L,unsigned char R);
void kep ();
void tha ();
void up ();
void down () ;
void buzzer (unsigned char t);  // tham so t la so lan kich buzzer
void doline();
void xoay_phai();          // xoay phai danh cho ham xoay 180
void xoay_phai1();         // xoay phai danh cho ham xoay 90
void xoay_trai1();         // xoay trai danh cho ham xoay 90
void xoay_180();           // xoay robot 180 do
void xoay_phai_90();       // xoay phai robot 90 do
void xoay_trai_90();       // xoay trai robot 90 do
void offset_phai();        // chinh robot ve line trung tam khi lech phai
void offset_trai();        // chinh robot ve line trung tam khi lech trai

////////////////////////////////////////////////////////////////////////////

// Declare your global variables here

/*   CHUONG TRINH MAU
 Chuong trinh main gom co cac hoat dong sau:
 1. robot do line gap nga tu, dung lai
 2. gap qua, nang qua len
 3. xoay 180 do, quay tro lai vi tri xuat phat
*/

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

```
// Input/Output Ports initialization
// Port A 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
PORTA=0xF0;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=P State3=P State2=P State1=T State0=T
PORTB=0x1C;
DDRB=0x00;

// Port C 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
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0xFF;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 250.000 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=0x03;
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
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// 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 disabled
ADCSRA=0x00;

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

// TWI initialization
// TWI disabled
TWCR=0x00;
buzzer(2);
while (1)
      {
      down();
      delay_ms(50);
      tha();
      buzzer(1);
       while(sensor!=ngatu)
       {
        doline();
       }
       speed(0,0);
       delay_ms(1000);
       kep();
       up();
       buzzer(1);
       delay_ms(1000);
       buzzer(1);
       xoay_180();

        delay_ms(300);
```

```
        buzzer(1);
        while(sensor!=ngatu)
       {
        doline();
       }

       speed(0,0);
       while(1);

      }
}
//////////////////////////////////////////////////////
void speed(unsigned char L,unsigned char R)
{
   pwmL=L*2.54;
   pwmR=R*2.54;
}
//////////////////////////////////////////////////////
void kep ()
{
     Dir_kep_tha = 0;
     delay_ms(20);
  while(limit_kep)
  {

    pwmDC_kep_tha = 1;
  }
   pwmDC_kep_tha = 0;
}
//////////////////////////////////////////////////////
void tha ()
{
    Dir_kep_tha = 1;
     delay_ms(50);
  while(limit_tha)
  {

    pwmDC_kep_tha = 1;
  }
   pwmDC_kep_tha = 0;
```

```
}
//////////////////////////////////////////////////////
void up ()
{
    Dir_up_down = 1;
    delay_ms(50);
    pwmDC_up_down = 1;
    delay_ms(3000);
   pwmDC_up_down = 0;
}
//////////////////////////////////////////////////////
void down ()
{
 while(limit_down)
 {
    Dir_up_down = 0;
    delay_ms(50);
    pwmDC_up_down = 1;
 }
   pwmDC_up_down = 0;
}
//////////////////////////////////////////////////////
void buzzer (unsigned char t)
{
   char i;
   for(i=0;i<t;i++)
   {
      PORTA.6=0;
      delay_ms(60);
      PORTA.6=1;
      delay_ms(60);
   }
}
//////////////////////////////////////////////////////
void doline()
{
   DirL = DirR = 0;
    switch (sensor)
   {
    case p7:
```

```
    speed(70,30);
    erR=0;
    erL=1;
    break;
case p6:
    speed(70,30);
    erR=0;
    erL=1;
    break;
case p5:
    speed(70,40);
    erR=0;
    erL=1;
    break;
case p4:
    speed(70,40);
    break;
case p3:
    speed(70,50);
    break;
case p2:
    speed(70,60);
    break;
case p1:
    speed(70,60);
    break;
case tt:
    speed(77,70);
    break;
case t1:
    speed(60,70);
    break;
case t2:
    speed(60,70);
    break;
case t3:
    speed(50,70);
    break;
case t4:
    speed(40,70);
```

```
    break;
case t5:
    speed(40,70);
    erR=1;
    erL=0;
    break;
case t6:
    speed(30,70);
    erR=1;
    erL=0;
    break;
case t7:
    speed(30,70);
    erR=1;
    erL=0;
    break;
case rangoai:
    if(erR==1)
    {
    speed(0,70);
    }
    else if(erL==1)
    {
    speed(70,0);
    }

    break;
 default:
 {
    if(erR==1)
    {
    speed(0,70);
    }
    else if(erL==1)
    {
    speed(70,0);
    }
    speed(70,70);
    break;
 }
```

```
   };

}
////////////////////////////////////////////////
void xoay_phai()
{
  DirL =0;
  DirR = 1;
  speed(55,40);
}
////////////////////////////////////////////////
void xoay_phai1()
{
  DirL =0;
  DirR = 1;
  speed(60,45);
}
////////////////////////////////////////////////
void xoay_trai1()
{
  DirL =1;
  DirR = 0;
  speed(60,45);
}
////////////////////////////////////////////////
void offset_phai ()
{
 while(sensor!= t4)
 {
   if(sensor==tt) break;
     DirL =0;
    DirR = 1;
   speed(30,30);
 }
  speed(0,0);
  delay_ms(200);
  while(sensor!= p4)
 {
   if(sensor==tt) break;
    DirL =1;
```

```
    DirR = 0;
   speed(30,30);
 }
  speed(0,0);
}
////////////////////////////////////////////////
void offset_trai ()
{
 while(sensor!= p7)
 {
   if(sensor==tt) break;
    DirL =1;
    DirR = 0;
   speed(30,30);
 }
  speed(0,0);
  delay_ms(200);
 while(sensor!= t7)
 {
   if(sensor==tt) break;
     DirL =0;
    DirR = 1;
   speed(30,30);
 }
  speed(0,0);


}
////////////////////////////////////////////////
void xoay_180(void)
{
char i;
    offset_phai();
    while(i<=3)
       {


         xoay_phai();
         if(sensor==tt){ while(sensor==tt); i++; }
         delay_ms(50);
```

```
      }
  speed(0,0);
  delay_ms(300);
  offset_trai();
}
//////////////////////////////////////////////////

void xoay_phai_90()
{
  char a;
    while(a<2)
      {

        xoay_phai1();
       if(sensor==tt){ while(sensor==tt); a++; }
      }

     while(sensor!= t4)
    {
       DirL =0;
       DirR = 1;
       speed(30,30);
     }
       speed(0,0);
      delay_ms(300);
      offset_trai();

}
//////////////////////////////////////////////////////////
void xoay_trai_90()
{
  char a;
    while(a<2)
      {

        xoay_trai1();
       if(sensor==tt){ while(sensor==tt); a++; }
      }

     while(sensor!= p4)
```

```
    {
      DirL =1;
      DirR = 0;
      speed(30,30);
     }
      speed(0,0);
     delay_ms(300);
     offset_phai();
}
```

2. Chương trình điều khiển Robot điều khiển bằng tay

```
**************************************************/

#include <mega8.h>
#include <delay.h>

// Declare your global variables here
#define Psdata      PINB.5
#define Pscmd       PORTB.0
#define Psatt       PORTD.7
#define Psclk       PORTD.6
////////////////////////////////////
#define PwmL        OCR1A          //toc do dong co ben trai
#define DirL        PORTC.4        //chieu dong co trai
#define PwmR        OCR1B          //toc do dong co ben phai
#define DirR        PORTC.5 //chieu dong co ben phai
////////////////////////////////////
#define PwmDC1      PORTB.3 //OCR2
#define DirDC1      PORTB.4
////////////////////////////////////
#define tay_kep  PORTD.0
#define Dir_tay_kep  PORTD.1
////////////////////////////////////
#define tay_truot   PORTD.2
#define DirDC3      PORTD.3
////////////////////////////////////
#define kep_token   PORTD.4
#define DirDC4      PORTD.5
////////////////////////////////////
#define toi         0
#define lui         1
#define kep         1
#define tha         0

#define dorong      100                //tinh theo phan tram

#define do_rong     0xff*dorong/100     //tinh theo ti le

#define loc_nhieu 10
// Data Types defines
```

```
typedef unsigned char byte;
typedef unsigned int word;

static word timecount;

enum state{setup,runpro};
enum
PS{NOPRESS,SLT,START,UP,RIGHT,DOWN,LEFT,TRIGLE,O,X,SQUARE,L2,R2
,L1,R1};

static byte newnumpressed=NOPRESS;

static byte Psbyte4,Psbyte5,Rightjoylr,Rightjoyud,Leftjoylr,Leftjoyud;

unsigned char
t_l1,t_l2,t_r1,t_r2,t_up,t_down,t_left,t_right,t_triangle,t_square,t_start,t_select,t_x;
static byte Ps_access (byte Tdata);
byte PS_state();

void  select();
void  start();
void  up();
void  right();
void  down();
void  left();
void  triangle();
void  circle();
void  x();
void  square();
void  l1();
void  r1();
void  l2();
void  r2();

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
  //1ms timer0
  // Reinitialize Timer 0 value
  TCNT0=0x06;
```

```
 if (timecount++==1000)
 {
    timecount=0;
 }
 if ((timecount%5)==0)
 {
      newnumpressed=PS_state();
      if(t_r1>0)      t_r1--;
      if(t_l1>0)      t_l1--;
      if(t_l2>0)      t_l2--;
      if(t_r2>0)      t_r2--;
      if(t_up>0)      t_up--;
      if(t_down>0)    t_down--;
      if(t_right>0)   t_right--;
      if(t_left>0)    t_left--;
      if(t_start>0)   t_start--;
      if(t_select>0)  t_select--;
     if(t_triangle>0)  t_triangle--;
     if(t_x>0)        t_x--;

 }

}

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

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=T State6=T State5=T State4=0 State3=0 State2=0 State1=0 State0=0
PORTB=0x21;
DDRB=0b11011111;

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

```
DDRC=0xff;

// Port D initialization
// Func7=Out Func6=Out Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=0 State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0xc0;
DDRD=0xFF;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 125.000 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=0x03;
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: 125.000 kHz
// Mode: Fast PWM top=0xFF
// OC2 output: Inverted PWM
ASSR=0x00;
//TCCR2=0x7C;     //125Khz
TCCR2=0x7B;     //250Khz
TCNT2=0x00;
```

```
OCR2=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 125.000 kHz
TCCR0=0x03;
TCNT0=0x00;

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

// Global enable interrupts
#asm("sei")
tay_kep=tay_truot=kep_token=tha;
while (1)       //chuong trinh chinh. Tac dung quet game pad
//vi dieu khien lien tuc doc du lieu tu game pad de biet nut nao duoc nhan
//du lieu tra ve thong tin trang thai cac nut nhan duoc luu trong cac thanh ghi:
//Psbyte4 Psbyte5
      {
          // PwmL=Leftjoylr;
        if((Psbyte4 ==0xFF)&&(Psbyte5 ==0xFF))
          {
                if((t_r2==0)&&(t_start==0)) DirR=0;
                if((t_l2==0)&&(t_start==0)) DirL=0;
                if(t_left==0) Dir_tay_kep=0;
                if((t_l1==0)&&(t_l2==0)&&(t_select==0)&&(t_start==0)) PwmL=0;
                if((t_r1==0)&&(t_r2==0)&&(t_select==0)&&(t_start==0)) PwmR=0;
                if((t_up==0)&&(t_down==0))
                {
                      PwmDC1=1;
                      OCR2=0;
                      TCCR2=0;
                      DirDC1=0;
                }

                if((t_right==0)&&(t_left)==0) tay_kep=0;
                if((t_right==0)&&(t_left)==0)  Dir_tay_kep=0;
                if((t_x==0)&&(t_triangle)==0) tay_truot=0;
                if((t_x==0)&&(t_triangle)==0) DirDC3 =0;
```

```
                DirDC4=0;

                t_square=100;

            }

            else
            {
                if((t_r2==0)&&(t_start==0)) DirR=0;
                if((t_l2==0)&&(t_start==0)) DirL=0;
                if(t_left==0) Dir_tay_kep=0;
                if((t_l1==0)&&(t_l2==0)&&(t_select==0)&&(t_start==0)) PwmL=0;
                if((t_r1==0)&&(t_r2==0)&&(t_select==0)&&(t_start==0)) PwmR=0;
                if((t_up==0)&&(t_down==0))
                {
                    PwmDC1=1;
                    OCR2=0;
                    TCCR2=0;
                    DirDC1=0;
                }
                //tat nut khi nhan cung luc
                triangle();
                circle();
                x();
                square();

                up();
                right();
                down();
                left();
                l1();
                r1();
                l2();
                r2();
                start();
                select();
            }
        }

}
```

```
//vi dieu khien se gui 1 byte yeu cau game pad tra ve du lieu trang thai cua game pad
byte Ps_access (byte Tdata)
{
  byte i,data=0;

  for (i=0;i<8;i++)
  {
     Pscmd=(Tdata>>i)&0x01;
     Psclk=0;
     delay_us(20);
     data|=(Psdata<<i);
     Psclk=1;
     delay_us(20);
  }
return data;
}

byte PS_state()
{
#asm("cli")
      Psbyte4=0xFF;
      Psbyte5=0xFF;
      Psatt=0;
      //byte1
      Ps_access(0x01);
      //byte2
      switch(Ps_access(0x42))
      {
         case 0x41:  //byte 3
                  if  (Ps_access(0x00) == 0x5A )
                  {
                    //byte 4
                    Psbyte4 = Ps_access(0x00);
                    //byte 5
                    Psbyte5= Ps_access(0x00);
                  }
         case 0x73:
                  //byte 3
                  if  (Ps_access(0x00) == 0x5A )
```

```
                {
                        //byte 4
                        Psbyte4=Ps_access(0x00);
                        //byte 5
                        Psbyte5=Ps_access(0x00);
                        //byte 6
                        Rightjoylr =Ps_access(0x00);
                        //byte 7
                        Rightjoyud =Ps_access(0x00);
                        //byte 8
                        Leftjoylr=Ps_access(0x00);
                        //byte 9
                        Leftjoyud=Ps_access(0x00);
                 }
        }
#asm("sei")
        Psatt=1;
        return 0;
  }
//=============================================================
void r1()
{
        if((Psbyte5&8)==0)
        {
                t_r1=loc_nhieu;
                PwmR=254;
        }
}
//=============================================================
void r2()
{
        if((Psbyte5&2)==0)
        {
                t_r2=loc_nhieu;
                DirR=1;
                PwmR=254;
        }
}
//=============================================================
void l1()
```

```
{
	if((Psbyte5&4)==0)
	{
		t_l1=loc_nhieu; //khong bo duoc, anh huong do on dinh dong co
		PwmL=254;
	}
}
//============================================================
void l2()
{
	if((Psbyte5&1)==0)
	{
		t_l2=loc_nhieu;
		DirL=1;
		PwmL=254;
	}
}
//============================================================

void  select()
{	if((Psbyte4&1)==0)
	{
		t_select=loc_nhieu;
		PwmL=PwmR=254;
	}
}
//============================================================
void  start()
{
	if((Psbyte4&8)==0)
	{
		t_start=loc_nhieu;
		PwmL=PwmR=254;
		DirL=DirR=1;
	}
}
//============================================================
void  up()
{
```

```
      if((Psbyte4&16)==0)
      {
            t_up=loc_nhieu;
            TCCR2=0x7C;
            OCR2=255;
      }
}
//=============================================================
void  down()
{
      if((Psbyte4&64)==0)
      {
            t_down=loc_nhieu;
            TCCR2=0x7C;
            OCR2=255;
            DirDC1=1;
      }
}
//=============================================================
void  right()
{
      if((Psbyte4&128)==0)
      {
           t_right=loc_nhieu;
           tay_kep=1;
      }
}
//=============================================================
void  left()
{
      if((Psbyte4&32)==0)
      {
            t_left=loc_nhieu;
            tay_kep=1;
            Dir_tay_kep=1;
      }
}
//=============================================================
void  triangle()
{
```

```
	if((Psbyte5&16)==0)
	{
		t_triangle=loc_nhieu;
		tay_truot=1;
	}
}
//=============================================================
void  x()
{
	if((Psbyte5&64)==0)
	{
	   t_x=loc_nhieu;
	   DirDC3=1;
	   tay_truot=1;
	}

}
//=============================================================
void  square()
{
	if((Psbyte5&128)==0)
	{
		if(t_square>0)  t_square--;
		else
				{
					kep_token=kep;
				}
	}
}
//=============================================================
void  circle()
{
	if((Psbyte5&32)==0)
		{
			kep_token=tha;
		}
}
```

3. Chương trình Test sa bàn thí nghiệm

**********/

```
#include <mega32.h>

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

#define PWML OCR1A
#define PWMR OCR1B
#define DIRL PORTD.6
#define DIRR PORTD.7
#define BUZZ PORTD.3
#define STARTBUT PIND.2
#define SENSORS PINC

#define FORW 0
#define REV 1

#define MAXSPEED 255

#define center 0b00011000

#define r1    0b00001000
#define r2    0b00001100
#define r3    0b00000100
#define r4    0b00000110
#define r5    0b00000010
#define r6    0b00000011
#define r7    0b00000001

#define l1    0b00010000
#define l2    0b00110000
#define l3    0b00100000
#define l4    0b01100000
#define l5    0b01000000
#define l6    0b11000000
#define l7    0b10000000
```

```
#define err    0b00000000

#define quarter 0b00111100

char cnt;
void sound(char count)
{
  while(count--!=0)
  {
     BUZZ=0;
     delay_ms(100);
     BUZZ=1;
     delay_ms(100);

  }

}

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
   switch (SENSORS)
   {
      case (center) : PWML=PWMR=MAXSPEED;
                      break;
      case (r1) :     PWML=MAXSPEED;
                      PWMR=MAXSPEED*6/7;
                      break;
      case (r2) :     PWML=MAXSPEED;
                      PWMR=MAXSPEED*5/7;
                      break;
      case (r3) :     PWML=MAXSPEED;
                      PWMR=MAXSPEED*4/7;
                      break;
      case (r4) :     PWML=MAXSPEED;
                      PWMR=MAXSPEED*3/7;
                      break;
      case (r6) :     PWML=MAXSPEED;
                      PWMR=MAXSPEED*2/7;
                      break;
```

```
    case (r7) :     PWML=MAXSPEED/2;
                    PWMR=MAXSPEED*1/7;
                    break;

    case (l1) :     PWMR=MAXSPEED;
                    PWML=MAXSPEED*6/7;
                    break;
    case (l2) :     PWMR=MAXSPEED;
                    PWML=MAXSPEED*5/7;
                    break;
    case (l3) :     PWMR=MAXSPEED;
                    PWML=MAXSPEED*4/7;
                    break;
    case (l4) :     PWMR=MAXSPEED;
                    PWML=MAXSPEED*3/7;
                    break;
    case (l6) :     PWMR=MAXSPEED;
                    PWML=MAXSPEED*2/7;
                    break;
    case (l7) :     PWMR=MAXSPEED/2;
                    PWML=MAXSPEED*1/7;
                    break;

    default:        if ((SENSORS&0b00111100)==quarter)
                    {
                      sound(1);
                      ++cnt;
                      while ((SENSORS&0b00111100)==quarter);
                    }
  }

}

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

// Input/Output Ports initialization
// Port A 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
PORTA=0x00;
DDRA=0x00;

// Port B 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
PORTB=0x00;
DDRB=0x00;

// Port C 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
PORTC=0xFF;
DDRC=0x00;


// Port D initialization
// Func7=In Func6=In Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=0 State4=0 State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x30;
DDRD.6=1;
DDRD.7=1;
DDRD.3=1;
PORTD.2=1;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 115.200 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x03;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 115.200 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=0x03;
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
// INT2: Off
MCUCR=0x00;
MCUCSR=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 disabled
ADCSRA=0x00;

// 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 - PORTA Bit 0
// RD - PORTA Bit 1
// EN - PORTA Bit 2
// D4 - PORTA Bit 4
// D5 - PORTA Bit 5
// D6 - PORTA Bit 6
// D7 - PORTA Bit 7
// Characters/line: 16
lcd_init(16);



lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("HELLO");
```

```
DIRL=DIRR=FORW;
while(STARTBUT!=0) sound(1);
// Global enable interrupts
#asm("sei")
while (1)
      {
        if (cnt==2)
        {
          #asm("cli")
          PWML=PWMR=MAXSPEED;
          DIRL=FORW;
          DIRR=REV;
          delay_ms(5000);
          #asm("sei")
          DIRL=DIRR=FORW;
          cnt=0;
        }
      }
}
```

MỤC LỤC

TRANG

DANH MỤC CÁC HÌNH VẼ

TRANG

DANH MỤC CÁC BẢNG

TRANG

DANH MỤC CHỮ VIẾT TẮT

JTAG:	Joint Test Action Group
SPI :	Serial Peripheral Interface Bus
UART :	Universal Asynchronous Receiver/Transmitter
PWM :	Pulse-width modulation
RTC :	real-time clock
RF:	Radio frequency
CMOS:	Complementary Metal-Oxide-Semiconductor
TTL :	Transistor–transistor logic
JP:	JUMPER
SW:	SWITCH