BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÀ RỊA VŨNG TÀU
VIỆN CNTT - ĐIỆN - ĐIỆN TỬ

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI:

ROBOT DÒ LINE ĐIỀU KHIỂN QUA ĐIỆN THOẠI

Họ và tên GVHD : ThS. Nguyễn L.Thanh Tùng

Họ và tên SV : Nguyễn Quốc An

Chuyên ngành : Điện – Điện tử

Lớp : DH13DD

Khóa : 2013 - 2017

Trình độ đào tạo : Đại học

Vũng Tàu, tháng 7 năm 2017

TRƯỜNG ĐẠI HỌC BÀ RỊA-VŨNG TÀU

KHOA ĐIỆN-ĐIỆN TỬ

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

------o0o-----

PHIẾU GIAO ĐỀ TÀI ĐỒ ÁN TỐT NGHIỆP

(Đính kèm Quy định về việc tổ chức, quản lý các hình thức tốt nghiệp ĐH, CĐ ban hành kèm theo Quyết định số 585/QĐ-ĐHBRVT ngày 16/7/2013 của Hiệu trưởng Trường Đại học BR-VT)

Họ và tên sinh viên: Nguyễn Quốc An

MSSV : 13030712 Lớp: DH13DD

Trình độ đào tạo : Đại học

Hệ đào tạo : Chính quy

Ngành : Công nghệ kỹ thuật điện-điện tử

Chuyên ngành : Kỹ thuật điện-điện tử

1. Tên đề tài: *Robot dò line điều khiển qua điện thoại.*

2. Giảng viên hướng dẫn: Th.S. Nguyễn Lương Thanh Tùng

4. Ngày hoàn thành đồ án/ khoá luận tốt nghiệp: 6/2017

Bà Rịa-Vũng Tàu, ngày tháng năm 2017

GIẢNG VIÊN HƯỚNG DẪN
(Ký và ghi rõ họ tên)

SINH VIÊN THỰC HIỆN
(Ký và ghi rõ họ tên)

TRƯỞNG BỘ MÔN
(Ký và ghi rõ họ tên)

TRƯỞNG KHOA
(Ký và ghi rõ họ tên)

LỜI CAM ĐOAN

Tôi xin cam đoan đồ án này tổng quát lại kết quả quá trình nghiên cứu của tôi. Các số liệu, hình ảnh, thông tin trong đồ án đều trung thực, do tôi tìm hiểu, tham khảo từ nhiều nguồn tư liệu. Đồ án này không sao chép các đồ án đã có từ trước.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đề tài của mình. Trường đại học BÀ RỊA-VŨNG TÀU không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

Vũng Tàu, ngày…., tháng …., năm 2017

Người cam đoan:

Nguyễn Quốc An

Nhận xét giáo viên hướng dẫn

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

Vũng Tàu, ngày … ,tháng 07, năm 2017

Giáo viên hướng dẫn

Nguyễn Lương Thanh Tùng

Nhận xét giáo viên phản biện

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

Vũng Tàu, ngày …, tháng 07, năm 2017

Giáo viên phản biện

Phạm Chí Hiếu

LỜI MỞ ĐẦU

Ngày nay, sự phát triển mạnh mẽ của khoa học đời sống, cuộc sống của con người đã thay đổi ngày một tốt hơn, với những trang thiết bị hiện đại phục vụ trong công cuộc công nghiệp hóa, hiện đại hóa. Đặc biệt góp phần không nhỏ đó là ngành kĩ thuật điện – điện tử trong sự nghiệp xây dựng đất nước. Những thiết bị điện, điện tử được phát triển và ứng dụng rộng rãi trong đời sống hằng ngày. Từ những thời gian đầu phát triển vi xử lý đã cho thấy sự ưu việt của nó và cho tới ngày nay tính ưu việt đó ngày càng được khẳng định thêm. Những thành tựu của nó đã có thể biến được những cái tưởng chừng như không thể thành những cái có thể, góp phần nâng cao đời sống vật chất và tinh thần cho con người.

Để góp phần làm sáng tỏ hiệu quả của những ứng dụng trong thực tế của môn vi xử lý, sau một thời gian học tập được các thầy cô trong khoa giảng dạy về các kiến thức chuyên ngành, đồng thời được sự giúp đỡ nhiệt tình của các thầy cô trong khoa Điện-Điện tử, cùng với sự lỗ lực của bản thân, em đã **" Thiết kế robot dò line điều khiển qua điện thoại"** nhưng do thời gian, kiến thức và kinh nghiệm của em còn có hạn nên sẽ không thể tránh khỏi những sai sót . Em rất mong được sự giúp đỡ và tham khảo ý kiến của thầy cô và các bạn nhằm đóng góp phát triển thêm đề tài.

SVTH

Nguyễn Quốc An

LỜI CÁM ƠN

Lời đầu tiên em xin chân thành cảm ơn đến thầy Nguyễn Lương Thanh Tùng đã giúp em rất nhiều trong quá trình thực hiện đồ án này.

Trong quá trình thực hiện đồ án, được sự giúp đỡ tận tình của thầy Nguyễn Lương Thanh Tùng em đã thu được nhiều kiến thức quý báu giúp em rất nhiều trong quá trình học và làm việc trong tương lai: được tiếp xúc với Arduino, Module Bluetooth, Module L298 và thi công mạch in, . . .

Trong quá trình thực hiện đồ án do em chưa có nhiều kinh nghiệm nên không tránh khỏi sai sót. Mong nhận được sự góp ý của các thầy để hoàn thiện hơn.

Một lần nữa em xin chân thành cảm ơn sự giúp đỡ của các quý thầy trong quá trình thực hiện đồ án để em hoàn thành đồ án này.

SVTH

Nguyễn Quốc An

MỤC LỤC

Đề mục **Trang**

Chương 1:

MỞ ĐẦU

1.1 Giới thiệu đề tài

Ngày nay, robotic đã đạt được những thành tựu to lớn trong sản xuất công nghiệp cũng như trong đời sống. Sản xuất robot là nghành công nghiệp trị giá hàng tỉ USD và ngày càng phát triển mạnh, trong các họ robot chúng ta không thể không nhắc tới mobile robot với những đặc thù riêng mà các loại robot khác không có.

Mobile robot có thể di chuyển một cách rất linh hoạt, do đó tạo nên không gian hoạt động lớn và cho đến nay nó đã dần khẳng định vai trò quan trọng không thể thiếu trong nhiều lĩnh vực, thu hút được rất nhiều sự đầu tư và nghiên cứu. Mobile robot cũng được chia ra làm nhiều loại: robot học đường đi, robot dò đường line, robot tránh vật cản, robot tìm đường cho mê cung,…trong số đó robot dò đường line, tránh vật cản dễ dàng ứng dụng nhiều trong cuộc sống. Việc phát triển loại robot này sẽ phục vụ rất đắc lực cho con người.

1.2 Mục đích đề tài

Robot dò line vừa có nhiều ứng dụng trong thực tế vừa dễ dàng để sinh viên vận dụng những kiến thức tiếp thu được trên giảng đường vào nó. Với những kết cấu cơ khí đơn giản nhưng lại có thể kết hợp được với khá nhiều thành phần điện tử (encoder, sensor xác định đường line, sensor đo khoảng cách…) nên những Robot này rất phù hợp để sinh viên học tập và nghiên cứu thêm về ngành Tự động hóa một cách cụ thể.

1.3 Sơ lược các bước thực hiện

- Trước tiên ta phải chế tạo được khung xe của robot. Khung xe phải đảm bảo bền chắc và đạt độ chính xác nhất định về việc bố trí các bánh xe và động cơ thông qua việc vẽ trên phần mềm và cắt CNC.

-Và cuối cùng là công đoạn lập trình dựa trên những kiến thức đã học được.

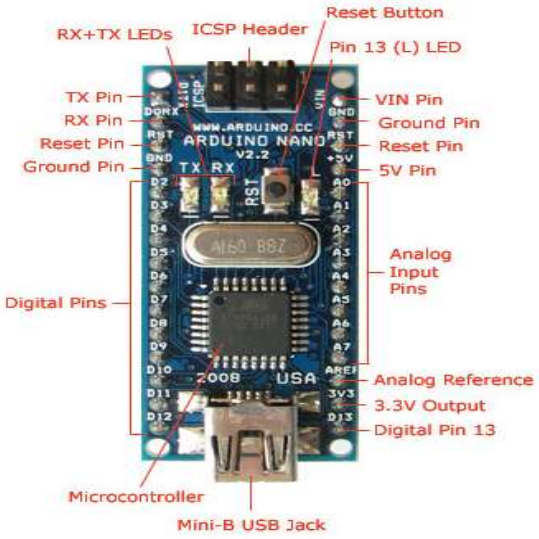
Chương 2:

GIỚI THIỆU ARDUINO VÀ CÁC THÀNH PHẦN CỦA MẠCH

2.1 Giới thiệu về ARDUINO

2.1.1 Sơ Lược về ARDUINO NANO

Khi tiếp xúc với Arduino Nano đó là sự tiện dụng, đơn giản, có thể lập trình trực tiếp bằng máy tính (như Arduino Uno R3) và đặc biệt hơn cả đó là kích thước của nó. Kích thước của Arduino Nano cực kì nhỏ chỉ tương đương đồng 2 nghìn gấp lại 2 lần thôi (1.85cm x 4.3cm), rất thích hợp cho các bạn bắt đầu học vì giá rẻ hơn Arduino Uno nhưng dùng được tất cả các thư việt của mạch này. Bài này nhằm mục đích giới thiệu về mạch Arduino Nano và các thông số kĩ thuật, cùng với đó là những gợi ý ứng dụng khi bắt đầu với mạch này.



Hình 2.1 Arduino Nano

Các thông số kĩ thuật của Arduino Nano hầu như giống hoàn Arduino Uno R3, vì vậy các thư viện trên Arduino Uno đều hoạt động tốt trên Arduino Uno. Tuy nhiên, ở Nano có một lợi thế cực kì quan trọng, nhờ đó Arduino Nano đã được ứng dụng rất nhiều trong các dự án DIY, đó chính là kích thước của nó. Dòng mạch Arduino phổ biến, khi mới bắt đầu làm quen, lập trình với Arduino thì mạch Arduino thường nói tới chính là dòng Arduino UNO.

2.1.2 Một vài thông số của Arduino Nano

Bảng 2.1. Thông số Arduino Nano

Vi Điều khiển	ATmega328 (họ 8bit)
Điện áp hoạt động	5V – DC
Tần số hoạt động	16 MHz
Dòng tiêu thụ	30mA
Điện áp vào khuyên dùng	7-12V – DC
Điện áp vào giới hạn	6-20V – DC
Số chân Digital I/O	14 (6 chân PWM)
Số chân Analog	**8** (độ phân giải 10bit)
Dòng tối đa trên mỗi chân I/O	**40 mA**
Dòng ra tối đa (5V)	500 mA
Dòng ra tối đa (3.3V)	50 mA
Bộ nhớ flash	32 KB (ATmega328) với 2KB dùng bởi bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Kích thước	**1.85cm x 4.3cm**

Các thông số kĩ thuật của Arduino Nano hầu như giống hoàn Arduino Uno R3, vì vậy các thư viện trên Arduino Uno đều hoạt động tốt trên Arduino Uno. Tuy nhiên, ở Nano có một lợi thế cực kì quan trọng, nhờ đó Arduino Nano đã được ứng dụng rất nhiều trong các dự án DIY, đó chính là kích thước của nó.

Đồng thời Nano còn số lượng chân Analog nhiều hơn Uno (2 chân A6, A7 chỉ dùng để đọc) cùng với dùng ra tối đa của mỗi chân IO lên đến 40mA. Nhưng, có một điểm trừ nhẹ cho Nano, đó là mạch này Nano cần đến 2KB bộ nhớ cho bootloader (ở Uno là 0.5KB).

Tuy nhiên, bạn đừng lo lắng, bạn còn đến tận 30KB bộ nhớ flash để lập trình, để dùng hết được 30KB này với tôi, đó là cả "một vấn đề lập trình".

2.1.3 Cổng kết nối với Arduino Nano

Khác với Arduino Nano sử dụng cổng USB Type B, Nano lại sử dụng một cổng nhỏ hơn có tên là mini USB.

Vì sử dụng cổng này nên kích thước board (vê chiều cao) cũng giảm đi khá nhiều, ngoài ra bạn có thể lập trình thẳng trực tiếp cho Nano từ máy tính - điều này tạo nhiều điện thuận lợi cho các bạn mới học.

2.1.4 Lập trình cho Arduino Nano

Cũng tương tự như bên Arduino Uno R3, Arduino Nano sử dụng chương trình Arduino IDE để lập trình, và ngôn ngữ lập trình cho Arduino cũng tên là Arduino (được xây dựng trên ngôn ngữ C).

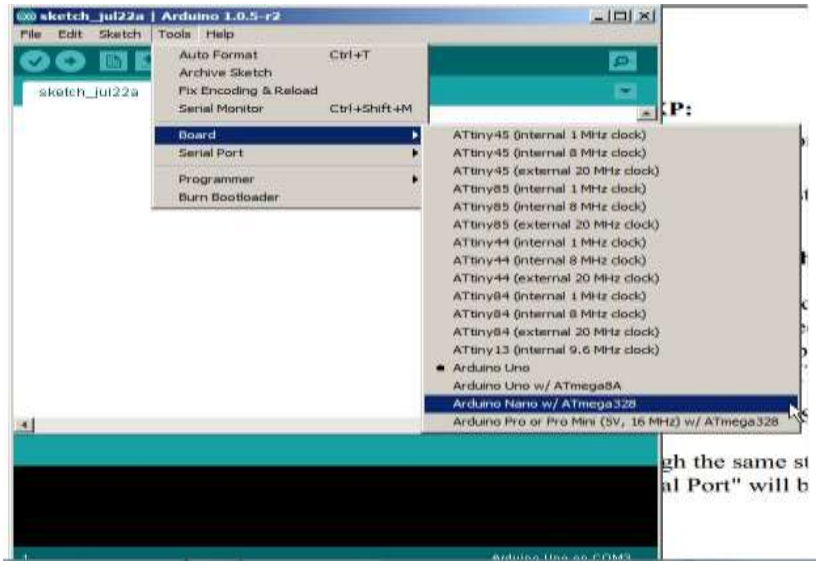
Tuy nhiên, nếu muốn lập trình cho Arduino Nano, bạn cần phải thực hiện một số thao tác trên máy tính. Sau đây, tôi sẽ hướng dẫn bạn từng bước để có thể lập trình cho Arduino Nano.

Đầu tiên, bạn cần cài Driver của Arduino Nano và tải về bản Arduino IDE mới nhất cho máy tính, các bước cài đặt hoàn toàn tương tự như Arduino Uno R3, bạn có thể tham khảo.

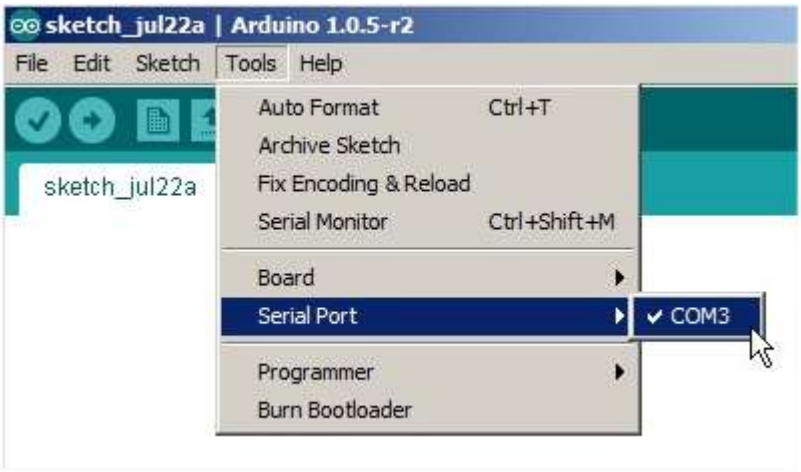
Sau khi cài đặt, bạn sẽ thấy một thông báo dạng "Cổng COMx đã được cài đặt thành công" (chữ "x" này sẽ được thay bằng một số nguyên dương, bạn hãy nhớ lấy số này, vì sau này bạn sẽ dùng cổng COMx này để lập trình cho Arduino Nano).

Mạch Arduino Nano là dòng mạch Arduino phổ biến, khi mới bắt đầu làm quen, lập trình với Arduino thì mạch Arduino thường nói tới chính là dòng Arduino Nano. Hiện dòng mạch này đã phát triển tới thế hệ thứ 3.

Arduino Nano là dòng cơ bản, linh hoạt, thường được sử dụng cho người mới bắt đầu. Bạn có thể sử dụng các dòng Arduino khác như: Arduino Mega, Arduino Nano, Arduino Micro… Nhưng với những ứng dụng cơ bản thì mạch Arduino Nano là lựa chọn phù hợp nhất.



Hình 2.2 Chọn Board Arduino



Hình 2.3 Chọn cổng Arduino

Sau đó, bạn cần lại loại board và cổng Serial mới như hình sau là được. Lưu ý, cổng COM trong hình dưới đây là chỉ là hình minh họa trong máy tính của mình thôi nhé.

Ngôn ngữ Arduino bắt nguồn từ C/C++ phổ biến hiện nay do đó rất dễ học, dễ hiểu. Nếu học tốt chương trình Tin học 11 thì việc lập trình Arduino sẽ rất dễ thở đối với bạn.

2.2 Các thành phần của mạch: LCD 16x2

2.2.1 Hình dáng và kích thước

Có rất nhiều loại LCD với nhiều hình dáng và kích thước khác nhau, trên hình 1 là loại LCD thông dụng.

Hình 2.4 : Hình dáng của loại LCD thông dụng

Khi sản xuất LCD, nhà sản xuất đã tích hợp chíp điều khiển (HD44780) bên trong lớp vỏ và chỉ đưa các chân giao tiếp cần thiết. Các chân này được đánh số thứ tự và đặt tên như hình 2 :

Hình 2.5 : Sơ đồ chân của LCD

Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào(chấp nhận) thanh ghi bên trong nó khi phát hiện một xung (high-to-low transition) của tín hiệu chân E.

Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện cạnh lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.

Chế độ 4 bit : Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7.

Chế độ 8 bit : Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7.

2.2.2 Chức năng của các chân

Bảng 2.2 Chức năng các chân của LCD

Chân	Ký hiệu	Mô tả
1	Vss	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển
2	Vdd	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với VCC=5V của mạch điều khiển
3	Vee	Điều chỉnh độ tương phản của LCD.
4	RS	Chân chọn thanh ghi (Register select). Nối chân RS với logic “0” (GND) hoặc logic “1” (VCC) để chọn thanh ghi.
5	R/W	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic “0” để LCD hoạt động ở chế độ ghi, hoặc nối với logic “1” để LCD ở chế độ đọc.
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E.
7-14	DB0-DB7	Tám đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này :
15	-	Nguồn dương cho đèn nền
16	-	GND cho đèn nền

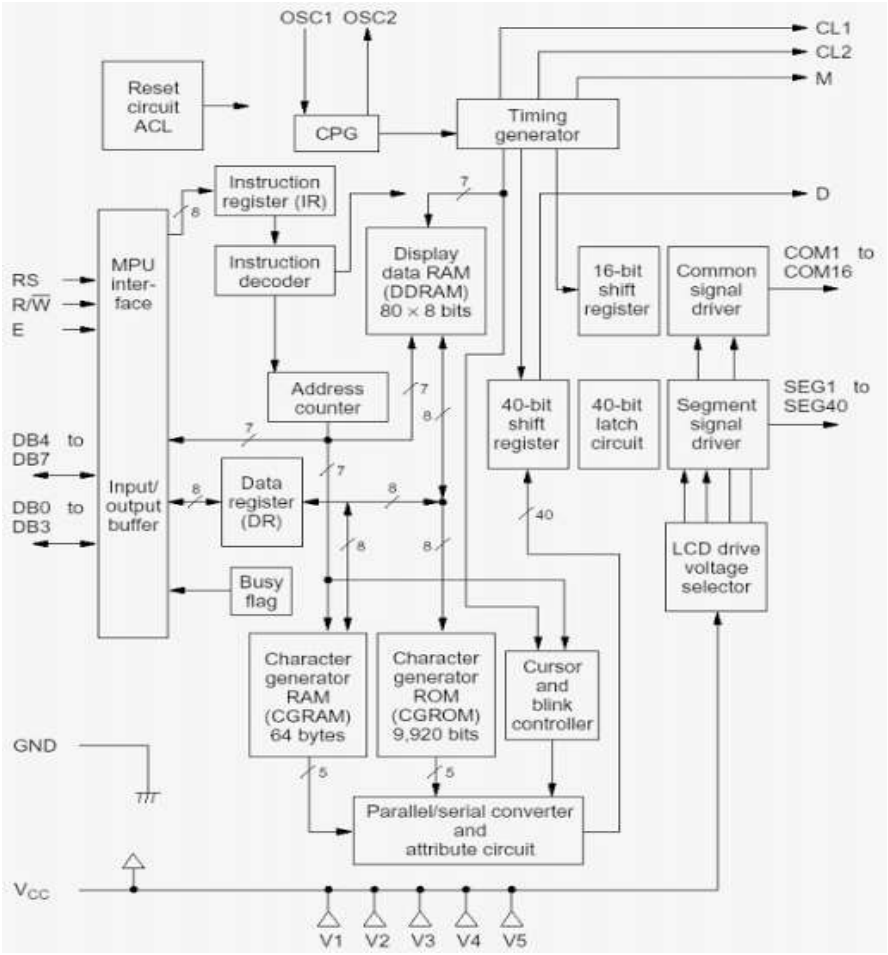
*** Ghi chú : Ở chế độ “đọc”, nghĩa là MPU sẽ đọc thông tin từ LCD thông qua các chân DBx. Còn khi ở chế độ “ghi”, nghĩa là MPU xuất thông tin điều khiển cho LCD thông qua các chân DBx.**

Tăng (I/D=1) hoặc giảm (I/D=0) bộ đếm địa chỉ hiển thị AC 1 đơn vị mỗi khi có hành động ghi hoặc đọc vùng DDRAM. Vị trí con trỏ cũng di chuyển theo sự tăng giảm này.

S : Khi S=1 toàn bộ nội dung hiển thị bị dịch sang phải (I/D=0) hoặc sang trái (I/D=1) mỗi khi có hành động ghi vùng DDRAM. Khi S=0: không dịch nội dung hiển thị. Nội dung hiển thị không dịch khi đọc DDRAM hoặc đọc/ghi vùng CGRAM.

2.2.3 Sơ đồ khối của HD44780

Để hiểu rõ hơn chức năng các chân và hoạt động của chúng, ta tìm hiểu sơ qua chíp HD44780 thông qua các khối cơ bản của nó.



Hình 2.6 : Sơ đồ khối của HD44780

A. Các thanh ghi

Chíp HD44780 có 2 thanh ghi 8 bit quan trọng : Thanh ghi lệnh IR (Instructor Register) và thanh ghi dữ liệu DR (Data Register).

Thanh ghi IR : Để điều khiển LCD, người dùng phải "ra lệnh" thông qua tám đường bus DB0-DB7. Mỗi lệnh được nhà sản xuất LCD đánh địa chỉ rõ ràng. Người dùng chỉ việc cung cấp địa chỉ lệnh bằng cách nạp vào thanh ghi IR. Nghĩa là, khi ta

nạp vào thanh ghi IR một chuỗi 8 bit, chíp HD44780 sẽ tra bảng mã lệnh tại địa chỉ mà IR cung cấp và thực hiện lệnh đó.

VD : Lệnh "hiển thị màn hình" có địa chỉ lệnh là 00001100 (DB7…DB0)

Lệnh "hiển thị màn hình và con trỏ" có mã lệnh là 00001110

Thanh ghi DR : Thanh ghi DR dùng để chứa dữ liệu 8 bit để ghi vào vùng RAM DDRAM hoặc CGRAM (ở chế độ ghi) hoặc dùng để chứa dữ liệu từ 2 vùng RAM này gởi ra cho MPU (ở chế độ đọc). Nghĩa là, khi MPU ghi thông tin vào DR, mạch nội bên trong chíp sẽ tự động ghi thông tin này vào DDRAM hoặc CGRAM. Hoặc khi thông tin về địa chỉ được ghi vào IR, dữ liệu ở địa chỉ này trong vùng RAM nội của HD44780 sẽ được chuyển ra DR để truyền cho MPU.

Bảng 2.3 : Chức năng chân RS và R/W theo mục đích sử dụng

RS	RW	Chức năng
0	0	Ghi vào thanh ghi IR để ra lệnh cho LCD
0	1	Đọc cờ bận ở DB7 và giá trị của bộ đếm địa chỉ DB0-DB6
1	0	Ghi vào thanh ghi DR
1	1	Đọc dữ liệu từ DR

B. Cờ báo bận BF: (Busy Flag)

Khi thực hiện các hoạt động bên trong chíp, mạch nội bên trong cần một khoảng thời gian để hoàn tất. Khi đang thực thi các hoạt động bên trong chip như thế, LCD bỏ qua mọi giao tiếp với bên ngoài và bật cờ BF (thông qua chân DB7 khi có thiết lập RS=0, R/W=1) lên để báo cho MPU biết nó đang "bận". Dĩ nhiên, khi xong việc, nó sẽ đặt cờ BF lại mức 0.

C. Bộ đếm địa chỉ : (Address Counter)

Như trong sơ đồ khối, thanh ghi IR không trực tiếp kết nối với vùng RAM (DDRAM và CGRAM) mà thông qua bộ đếm địa chỉ AC. Bộ đếm này lại nối với 2 vùng RAM theo kiểu rẽ nhánh. Khi một địa chỉ lệnh được nạp vào thanh ghi IR, thông tin được nối trực tiếp cho 2 vùng RAM nhưng việc chọn lựa vùng RAM tương tác đã được bao hàm trong mã lệnh.

Sau khi ghi vào (đọc từ) RAM, bộ đếm AC tự động tăng lên (giảm đi) 1 đơn vị và nội dung của AC được xuất ra cho MPU thông qua DB0-DB6 khi có thiết lập RS=0 và R/W=1 (xem bảng tóm tắt RS - R/W).

Lưu ý: Thời gian cập nhật AC không được tính vào thời gian thực thi lệnh mà được cập nhật sau khi cờ BF lên mức cao (not busy), cho nên khi lập trình hiển thị, bạn phải delay một khoảng tADD khoảng 4uS-5uS (ngay sau khi BF=1) trước khi nạp dữ liệu mới. Xem thêm hình bên dưới.

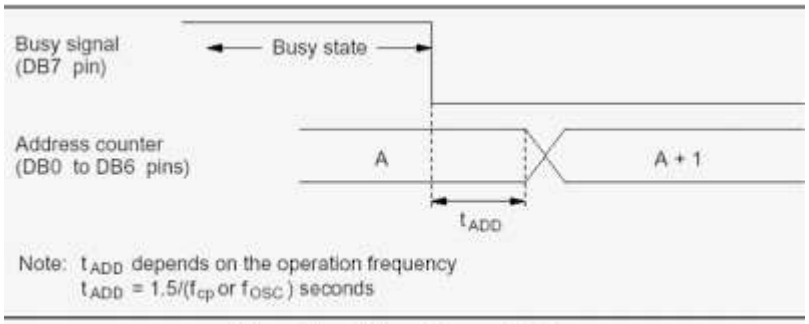


Figure 10 Address Counter Update

Hình 2.7 : Giản đồ xung cập nhật AC

D. Vùng RAM hiển thị DDRAM: (Display Data Ram)

Đây là vùng RAM dùng để hiển thị, nghĩa là ứng với một địa chỉ của RAM là một ô kí tự trên màn hình và khi bạn ghi vào vùng RAM này một mã 8 bit, LCD sẽ hiển thị tại vị trí tương ứng trên màn hình một kí tự có mã 8 bit mà bạn đã cung cấp. Hình sau đây sẽ trình bày rõ hơn mối liên hệ này :

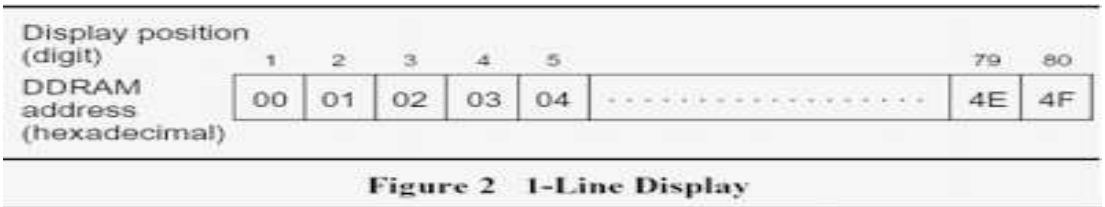


Figure 2 1-Line Display

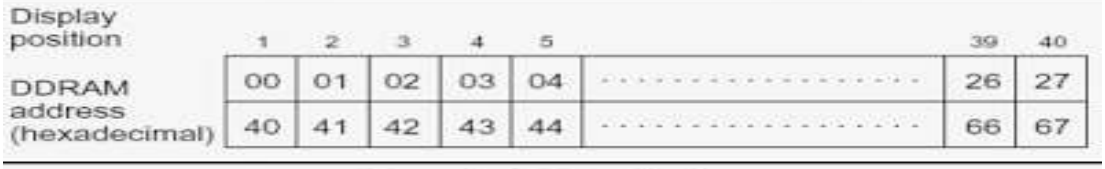


Figure 4 2-Line Display

Hình 2.8 : Mối liên hệ giữa địa chỉ của DDRAM và vị trí hiển thị của LCD

Vùng RAM này có 80x8 bit nhớ, nghĩa là chứa được 80 kí tự mã 8 bit. Những vùng RAM còn lại không dùng cho hiển thị có thể dùng như vùng RAM đa

mục đích. Lưu ý là để truy cập vào DDRAM, ta phải cung cấp địa chỉ cho AC theo mã HEX

E. Vùng ROM chứa kí tự CGROM: Character Generator ROM

Vùng ROM này dùng để chứa các mẫu kí tự loại 5x8 hoặc 5x10 điểm ảnh/kí tự, và định địa chỉ bằng 8 bit.

Tuy nhiên, nó chỉ có 208 mẫu kí tự 5x8 và 32 mẫu kí tự kiểu 5x10 (tổng cộng là 240 thay vì 2^8 = 256 mẫu kí tự). Người dùng không thể thay đổi vùng ROM này.

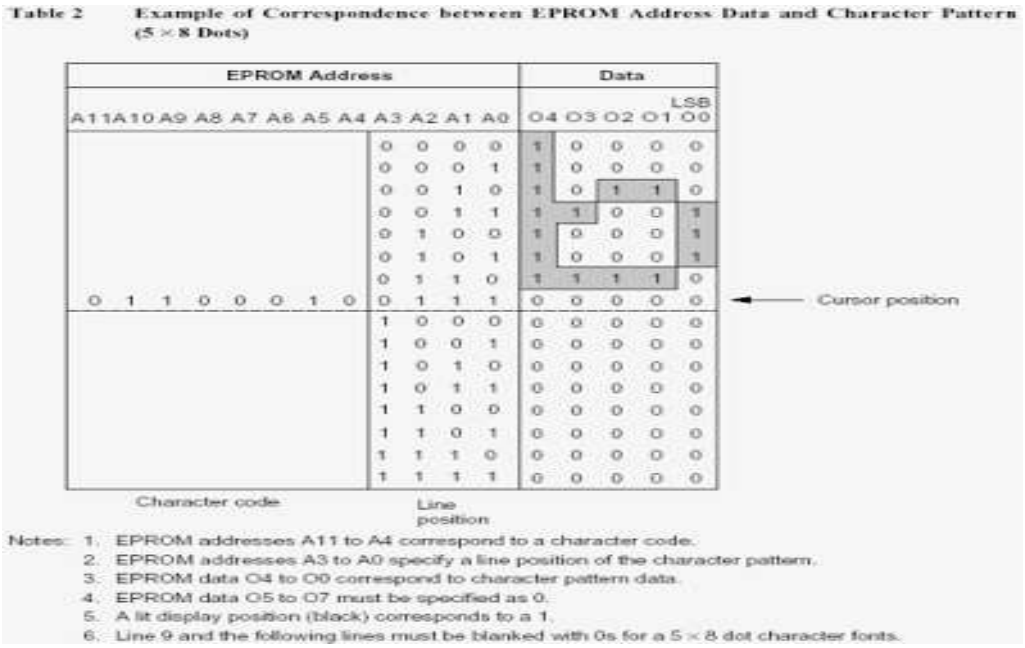


Table 2 Example of Correspondence between EPROM Address Data and Character Pattern (5 × 8 Dots)

EPROM Address												Data					
																LSB	
A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	O4	O3	O2	O1	O0	
								0	0	0	0	1	0	0	0	0	
								0	0	0	1	1	0	0	0	0	
								0	0	1	0	1	0	1	1	0	
								0	0	1	1	1	1	0	0	1	
								0	1	0	0	1	0	0	0	1	
								0	1	0	1	1	0	0	0	1	
								0	1	1	0	1	1	1	1	0	
0	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	← Cursor position
								1	0	0	0	0	0	0	0	0	
								1	0	0	1	0	0	0	0	0	
								1	0	1	0	0	0	0	0	0	
								1	0	1	1	0	0	0	0	0	
								1	1	0	0	0	0	0	0	0	
								1	1	0	1	0	0	0	0	0	
								1	1	1	0	0	0	0	0	0	
								1	1	1	1	0	0	0	0	0	

Character code — Line position

Notes: 1. EPROM addresses A11 to A4 correspond to a character code.
2. EPROM addresses A3 to A0 specify a line position of the character pattern.
3. EPROM data O4 to O0 correspond to character pattern data.
4. EPROM data O5 to O7 must be specified as 0.
5. A lit display position (black) corresponds to a 1.
6. Line 9 and the following lines must be blanked with 0s for a 5 × 8 dot character fonts.

Hình 2.9 : Mối liên hệ giữa địa chỉ của ROM và dữ liệu tạo mẫu kí tự.

Như vậy, để có thể ghi vào vị trí thứ x trên màn hình một kí tự y nào đó, người dùng phải ghi vào vùng DDRAM tại địa chỉ x (xem bảng mối liên hệ giữa DDRAM và vị trí hiển thị) một chuỗi mã kí tự 8 bit trên CGROM. Chú ý là trong bảng mã kí tự trong CGROM ở hình bên dưới có mã ROM A00.

Ví dụ : Ghi vào DDRAM tại địa chỉ "01" một chuỗi 8 bit "01100010" thì trên LCD tại ô thứ 2 từ trái sang (dòng trên) sẽ hiển thị kí tự "b".

Khi MPU ghi thông tin vào DR, mạch nội bên trong chíp sẽ tự động ghi thông tin này vào DDRAM hoặc CGRAM. Hoặc khi thông tin về địa chỉ được ghi vào IR, dữ liệu ở địa chỉ này trong vùng RAM nội của HD44780.

F. Vùng RAM chứa kí tự đồ họa CGRAM

Như trên bảng mã kí tự, nhà sản xuất dành vùng có địa chỉ byte cao là 0000 để người dùng có thể tạo các mẫu kí tự đồ họa riêng. Tuy nhiên dung lượng vùng này rất hạn chế: Ta chỉ có thể tạo 8 kí tự loại 5x8 điểm ảnh, hoặc 4 kí tự loại 5x10 điểm ảnh.

Để ghi vào CGRAM, hãy xem hình 6 bên dưới.

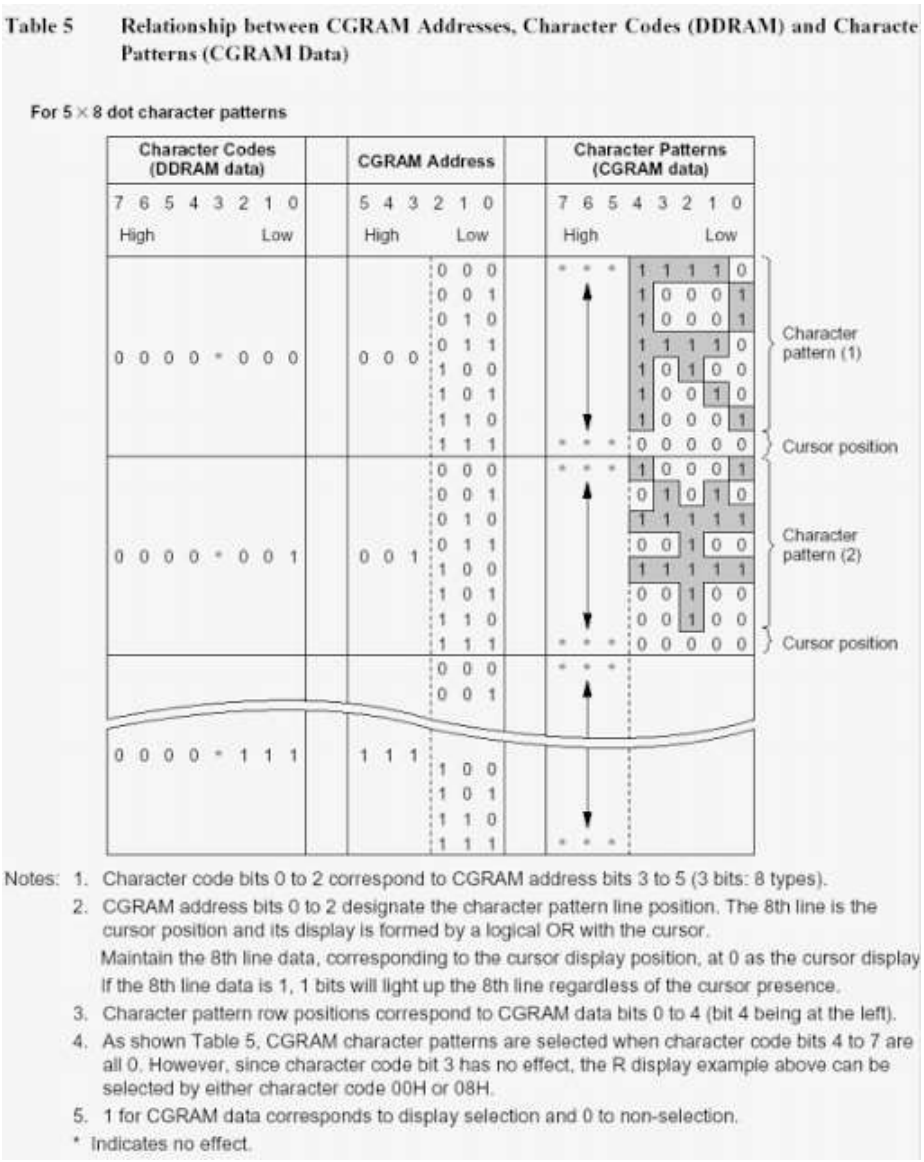


Table 5 Relationship between CGRAM Addresses, Character Codes (DDRAM) and Characte Patterns (CGRAM Data)

For 5 × 8 dot character patterns

Character Codes (DDRAM data)	CGRAM Address	Character Patterns (CGRAM data)	
7 6 5 4 3 2 1 0 (High … Low)	5 4 3 2 1 0 (High … Low)	7 6 5 4 3 2 1 0 (High … Low)	
0 0 0 0 * 0 0 0	0 0 0	* * * 1 1 1 1 0 * * * 1 0 0 0 1 * * * 1 0 0 0 1 * * * 1 1 1 1 0 * * * 1 0 1 0 0 * * * 1 0 0 1 0 * * * 1 0 0 0 1 * * * 0 0 0 0 0	Character pattern (1) Cursor position
0 0 0 0 * 0 0 1	0 0 1	* * * 1 0 0 0 1 * * * 0 1 0 1 0 * * * 1 1 1 1 1 * * * 0 0 1 0 0 * * * 1 1 1 1 1 * * * 0 0 1 0 0 * * * 0 0 1 0 0 * * * 0 0 0 0 0	Character pattern (2) Cursor position
0 0 0 0 * 1 1 1	1 1 1	* * *	

Notes:
1. Character code bits 0 to 2 correspond to CGRAM address bits 3 to 5 (3 bits: 8 types).
2. CGRAM address bits 0 to 2 designate the character pattern line position. The 8th line is the cursor position and its display is formed by a logical OR with the cursor.
   Maintain the 8th line data, corresponding to the cursor display position, at 0 as the cursor display
   If the 8th line data is 1, 1 bits will light up the 8th line regardless of the cursor presence.
3. Character pattern row positions correspond to CGRAM data bits 0 to 4 (bit 4 being at the left).
4. As shown Table 5, CGRAM character patterns are selected when character code bits 4 to 7 are all 0. However, since character code bit 3 has no effect, the R display example above can be selected by either character code 00H or 08H.
5. 1 for CGRAM data corresponds to display selection and 0 to non-selection.

* Indicates no effect.

Hình 2.9 : Mối liên hệ giữa địa chỉ của CGRAM, dữ liệu của CGRAM, và mã kí tự.

2.2.4.Tập lệnh của LCD 16x2

Trước khi tìm hiểu tập lệnh của LCD, sau đây là một vài chú ý khi giao tiếp với LCD :

* Tuy trong sơ đồ khối của LCD có nhiều khối khác nhau, nhưng khi lập trình điều khiển LCD ta chỉ có thể tác động trực tiếp được vào 2 thanh ghi DR và IR thông qua các chân DBx, và ta phải thiết lập chân RS, R/W phù hợp để chuyển qua lại giữ 2 thanh ghi này. (xem bảng 2)

* Với mỗi lệnh, LCD cần một khoảng thời gian để hoàn tất, thời gian này có thể khá lâu đối với tốc độ của MPU, nên ta cần kiểm tra cờ BF hoặc đợi (delay) cho LCD thực thi xong lệnh hiện hành mới có thể ra lệnh tiếp theo.

* Địa chỉ của RAM (AC) sẽ tự động tăng (giảm) 1 đơn vị, mỗi khi có lệnh ghi vào RAM. (Điều này giúp chương trình gọn hơn)

* Các lệnh của LCD có thể chia thành 4 nhóm như sau :

- Các lệnh về kiểu hiển thị. VD : Kiểu hiển thị (1 hàng / 2 hàng), chiều dài dữ liệu (8 bit / 4 bit), …
- Chỉ định địa chỉ RAM nội.
- Nhóm lệnh truyền dữ liệu trong RAM nội.
- Các lệnh còn lại .

Bảng 2.5 : Tập lệnh của LCD

Tên lệnh	Hoạt động
Clear Display	**Mã lệnh :** DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 0 0 0 0 0 0 0 1 Lệnh Clear Display (xóa hiển thị) sẽ ghi một khoảng trống-blank (mã hiện kí tự 20H) vào tất cả ô nhớ trong DDRAM, sau đó trả bộ đếm địa AC=0, trả lại kiểu hiển thị gốc nếu nó bị thay đổi. Nghĩa là : Tắt hiển thị, con trỏ dời về góc trái (hàng đầu tiên), chế độ tăng AC.
Return home	**Mã lệnh :** DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 0 0 0 0 0 0 1 * Lệnh Return home trả bộ đếm địa chỉ AC về 0, trả lại kiểu hiển thị gốc nếu nó bị thay đổi. Nội dung của DDRAM không thay đổi.
Entry mode set	**Mã lệnh :** DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 0 0 0 0 0 1 [I/D] [S]
Display	**Mã lệnh :** DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

<table>
<tr><td>on/off control</td><td>DBx = 0 0 0 0 1 [D] [C] [B]
D: Hiển thị màn hình khi D=1 và ngược lại. Khi tắt hiển thị, nội dung DDRAM không thay đổi.
C: Hiển thị con trỏ khi C=1 và ngược lại.
B: Nhấp nháy kí tự tại vị trí con trỏ khi B=1 và ngược lại.
Chu kì nhấp nháy khoảng 409,6ms khi mạch dao động nội LCD là 250kHz.</td></tr>
<tr><td>Cursor or

display

shift</td><td>Mã lệnh : DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0

DBx = 0 0 0 1 [S/C] [R/L] * *

Lệnh Cursor or display shift dịch chuyển con trỏ hay dữ liệu hiển thị sang trái mà không cần hành động ghi/đọc dữ liệu. Khi hiển thị kiểu 2 dòng, con trỏ sẽ nhảy xuống dòng dưới khi dịch qua vị trí thứ 40 của hàng đầu tiên. Dữ liệu hàng đầu và hàng 2 dịch cùng một lúc. Chi tiết sử dụng xem bảng bên dưới:
<table>
<tr><th>S/C</th><th>R/L</th><th>Hoạt động</th></tr>
<tr><td>0</td><td>0</td><td>Dịch vị trí con trỏ sang trái (Nghĩa là giảm AC một đơn vị).</td></tr>
<tr><td>0</td><td>1</td><td>Dịch vị trí con trỏ sang phải (Tăng AC lên 1 đơn vị).</td></tr>
<tr><td>1</td><td>0</td><td>Dịch toàn bộ nội dung hiển thị sang trái, con trỏ cũng dịch theo.</td></tr>
</table></td></tr>
</table>

Khi thiết lập RS=1, R/W=1,dữ liệu từ CG/DDRAM được chuyển ra MPU thông qua các chân DBx (địa chỉ và vùng RAM đã được xác định bằng lệnh ghi địa chỉ trước đó).

Sau khi đọc, AC tự động tăng/giảm 1 tùy theo thiết lập Entry mode, tuy nhiên nội dung hiển thị không bị dịch bất chấp chế độ Entry mode.

Khi thiết lập RS=1, R/W=0, dữ liệu cần ghi được đưa vào các chân DBx từ mạch ngoài sẽ được LCD chuyển vào trong LCD tại địa chỉ được xác định từ lệnh ghi địa chỉ trước đó (lệnh ghi địa chỉ cũng xác định luôn vùng RAM cần ghi)
Sau khi ghi, bộ đếm địa chỉ AC tự động tăng/giảm 1 tùy theo thiết lập Entry mode.

2.2.5 Giao tiếp giữa LCD và MCU

A. Đặc tính điện của các chân giao tiếp

LCD sẽ bị hỏng nghiêm trọng, hoặc hoạt động sai lệch nếu bạn vi phạm khoảng đặc tính điện sau đây:

Chân cấp nguồn (Vcc-GND)	Min:-0.3V , Max+7V
Các chân ngõ vào (DBx,E,…)	Min:-0.3V , Max:(Vcc+0.3V)
Nhiệt độ hoạt động	Min:-30C , Max:+75C
Nhiệt độ bảo quản	Min:-55C , Max:+125C

Bảng 2.6 : Maximun Rating

Đặc tính điện làm việc điển hình: (Đo trong điều kiện hoạt động Vcc = 4.5V đến 5.5V, T = -30 đến +75C)

Chân cấp nguồn Vcc-GND	2.7V đến 5.5V
Điện áp vào mức cao VIH	2.2V đến Vcc
Điện áp vào mức thấp VIL	-0.3V đến 0.6V
Điện áp ra mức cao (DB0-DB7)	Min 2.4V (khi IOH = -0.205mA)
Điện áp ra mức thấp (DB0-DB7)	Max 0.4V (khi IOL = 1.2mA)
Dòng điện ngõ vào (input leakage current) ILI	-1uA đến 1uA (khi VIN = 0 đến Vcc)
Dòng điện cấp nguồn ICC	350uA(typ.) đến 600uA

Bảng2.7: Miền làm việc bình thường

B. Sơ đồ nối mạch điển hình

- Sơ đồ mạch kết nối giữa mô đun LCD và VĐK 89S52 (8 bit).
- Sơ đồ mạch kết nối giữa môđun LCD và VĐK (4 bit).

C. Bus Timing

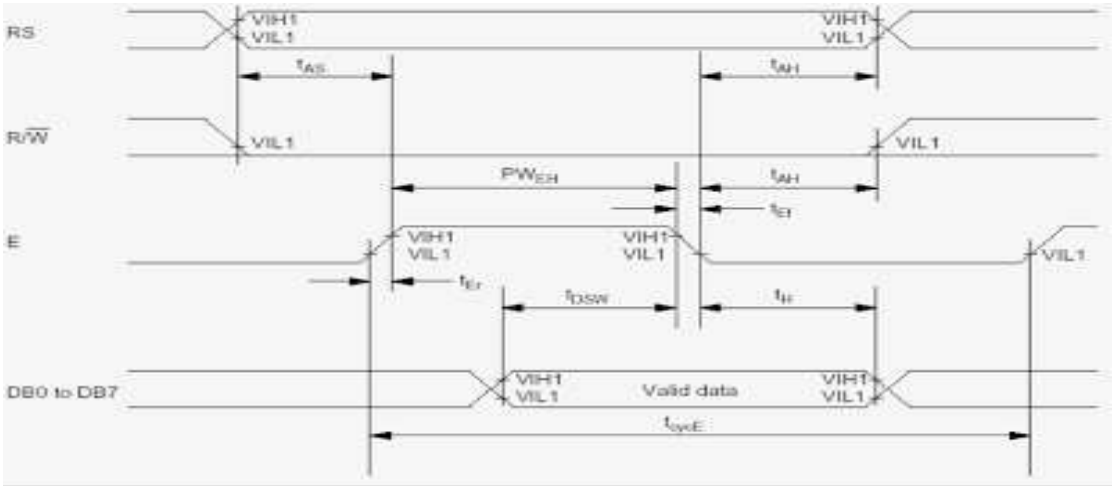


Figure 25 Write Operation

Item	Symbol	Min	Typ	Max	Unit
Enable cycle time	t_{cycE}	500	—	—	ns
Enable pulse width (high level)	PW_{EH}	230	—	—	
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	20	
Address set-up time (RS, R/W to E)	t_{AS}	40	—	—	
Address hold time	t_{AH}	10	—	—	
Data set-up time	t_{DSW}	80	—	—	
Data hold time	t_H	10	—	—	

Hình 2.10 Bus Timing

2.2.6 Khởi tạo LCD

Khởi tạo là việc thiết lập các thông số làm việc ban đầu. Đối với LCD, khởi tạo giúp ta thiết lập các giao thức làm việc giữa LCD và MPU. Việc khởi tạo chỉ được thực hiện 1 lần duy nhất ở đầu chương trình điều khiển LCD và bao gồm các thiết lập sau :

- Display clear : Xóa/không xóa toàn bộ nội dung hiển thị trước đó.
- Function set : Kiểu giao tiếp 8bit/4bit, số hàng hiển thị 1hàng/2hàng, kiểu kí tự 5x8/5x10.
- Display on/off control: Hiển thị/tắt màn hình, hiển thị/tắt con trỏ, nhấp nháy/không nhấp nháy.

• Entry mode set : các thiết lập kiểu nhập kí tự như: Dịch/không dịch, tự tăng/giảm (Increment).

A. Khởi tạo mạch bên trong chip HD 44780

Mỗi khi được cấp nguồn, mạch khởi tạo bên trong LCD sẽ tự động khởi tạo cho nó. Và trong thời gian khởi tạo này cờ BF bật lên 1, đến khi việc khởi tạo hoàn tất cờ BF còn giữ trong khoảng 10ms sau khi Vcc đạt đến 4.5V (vì 2.7V thì LCD đã hoạt động). Mạch khởi tạo nội sẽ thiết lập các thông số làm việc của LCD như sau:

• Display clear : Xóa toàn bộ nội dung hiển thị trước đó.

• Function set: DL=1 : 8bit; N=0 : 1 hàng; F=0 : 5x8

• Display on/off control: D=0 : Display off; C=0 : Cursor off; B=0 : Blinking off.

• Entry mode set: I/D =1 : Tăng; S=0 : Không dịch.

Như vậy sau khi mở nguồn, bạn sẽ thấy màn hình LCD giống như chưa mở nguồn do toàn bộ hiển thị tắt. Do đó, ta phải khởi tạo LCD bằng lệnh.

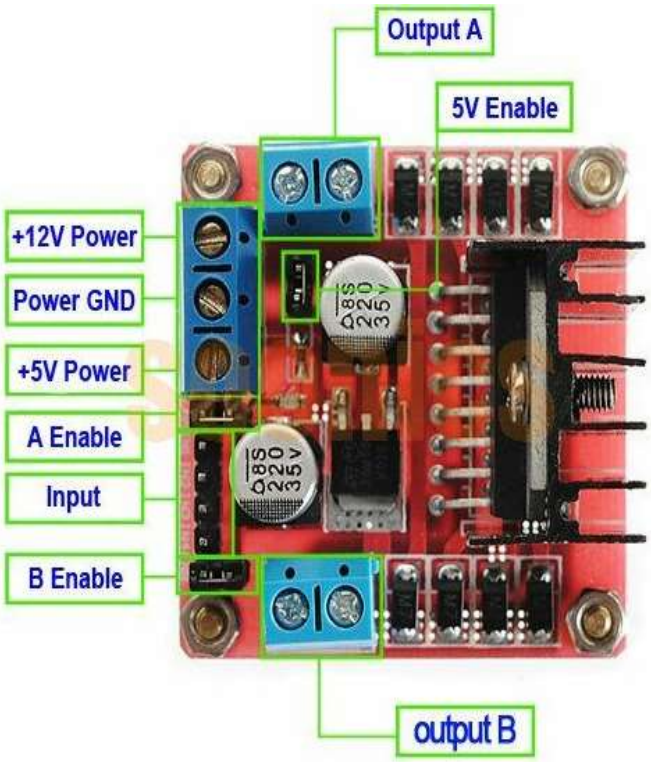
B. Khởi tạo bằng lệnh

Như đã đề cập ở trên, chế độ giao tiếp mặc định của LCD là 8bit (tự khởi tạo lúc mới bật điện lên). Và khi kết nối mạch theo giao thức 4bit, 4 bit thấp từ DB0-DB3 không được kết nối đến LCD, nên lệnh khởi tạo ban đầu (lệnh chọn giao thức giao tiếp – function set 0010****) phải giao tiếp theo chế độ 8 bit (chỉ gởi 4 bit cao một lần, bỏ qua 4 bit thấp). Từ lệnh sau trở đi, phải gởi/nhận lệnh theo 2 nibble.

Lưu ý là sau khi thiết lập function set, bạn không thể thay đổi function set ngoại trừ thay đổi giao thức giao tiếp (4bit/8bit).

Thời gian cập nhật AC không được tính vào thời gian thực thi lệnh mà được cập nhật sau khi cờ BF lên mức cao (not busy), cho nên khi lập trình hiển thị, bạn phải delay một khoảng tADD khoảng 4uS-5uS (ngay sau khi BF=1) trước khi nạp dữ liệu mới.

2.3 Các thành phần của mạch: MODULE L298N

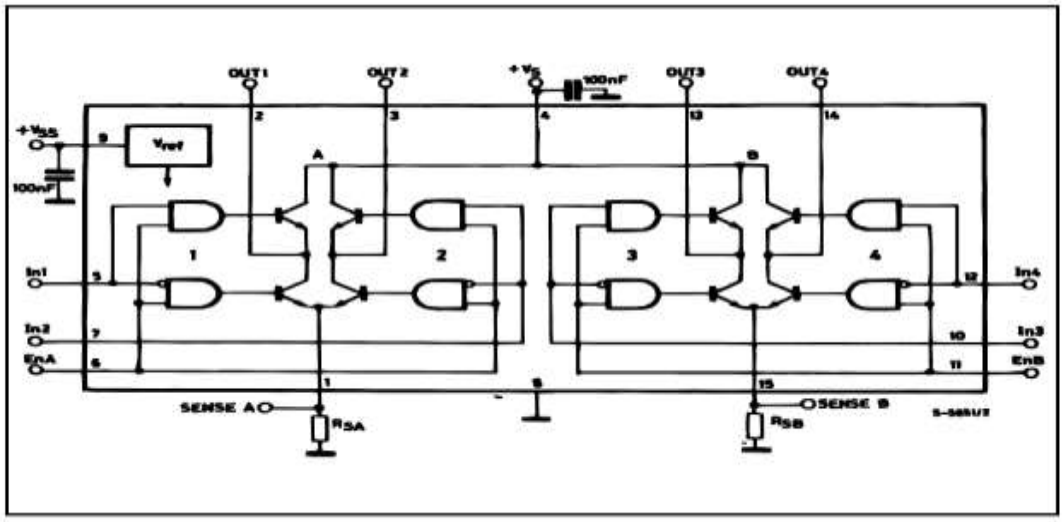


Hình 2.11 Module L298N

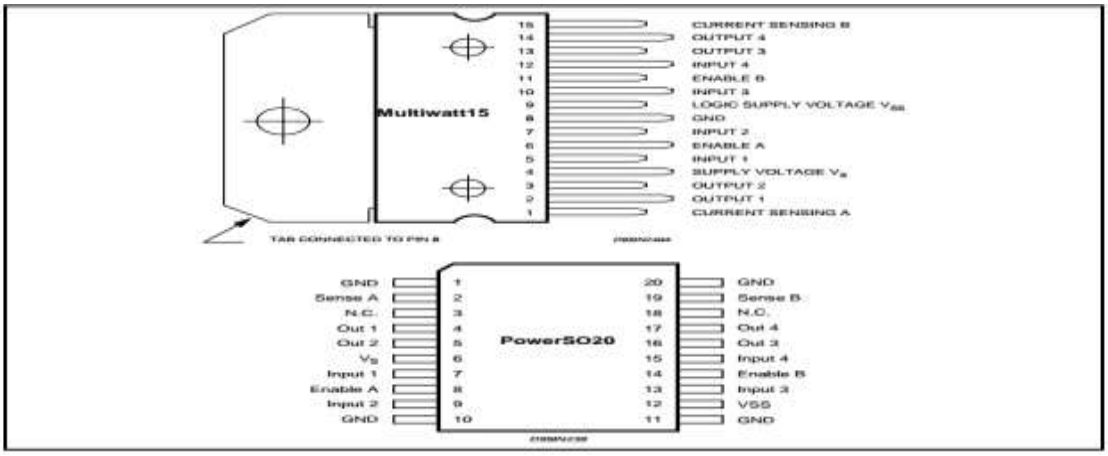
2.3.1 Thông số kỹ thuật

- Driver: L298N tích hợp hai mạch cầu H.
- Điện áp điều khiển: +5 V ~ +12 V
- Dòng tối đa cho mỗi cầu H là: 2A (=>2A cho mỗi motor)
- Điện áp của tín hiệu điều khiển: +5 V ~ +7 V
- Dòng của tín hiệu điều khiển: 0 ~ 36mA (Arduino có thể chơi đến 40mA nên khỏe re nhé các bạn)
- Công suất hao phí: 20W (khi nhiệt độ T = 75 ℃)
- Nhiệt độ bảo quản: -25 ℃ ~ +130 ℃

BLOCK DIAGAM

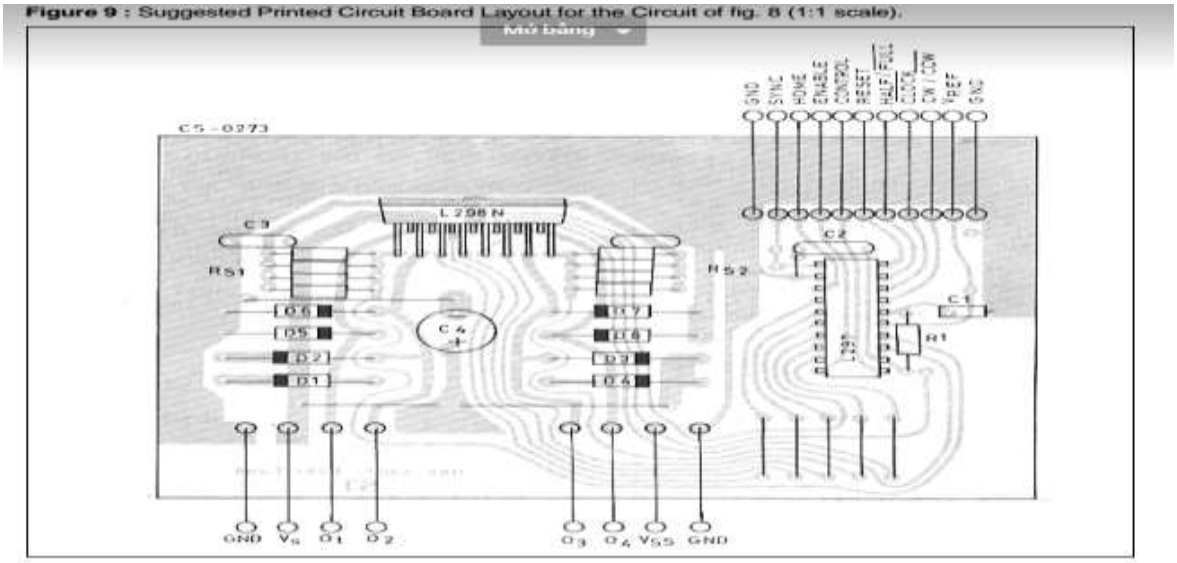


PIN CONNECTIONS



Output A: nối với động cơ A. bạn chú ý chân +, -. Nếu bạn nối ngược thì động cơ sẽ chạy ngược. Và chú ý nếu bạn nối động cơ bước, bạn phải đấu nối các pha cho phù hợp.

Board này gồm 2 phần điều khiển động cơ. Và có thể điều khiển cho 1 động cơ bước 6 dây hoặc 4 dây.

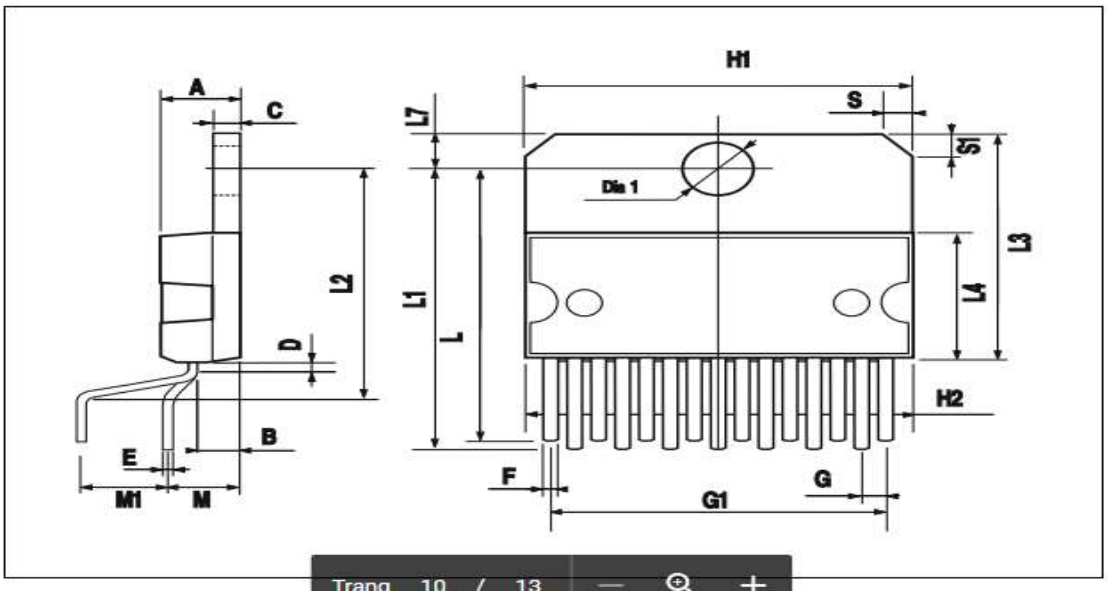


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

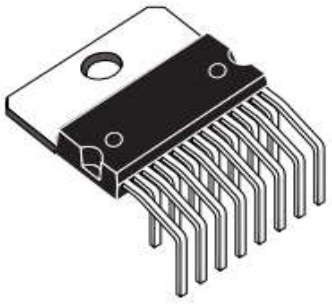
Multiwatt15 V

- 2 Jump A enable và B enable, để như hình, đừng rút ra bạn nhé!
- Gồm có 4 chân Input. IN1, IN2, IN3, IN4. Chức năng các chân này tôi sẽ giải thích ở bước sau.



DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L		20.57			0.810	
L1		18.03			0.710	
L2		2.54			0.100	
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L5		5.28			0.208	
L6		2.38			0.094	
L7	2.65		2.9	0.104		0.114
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



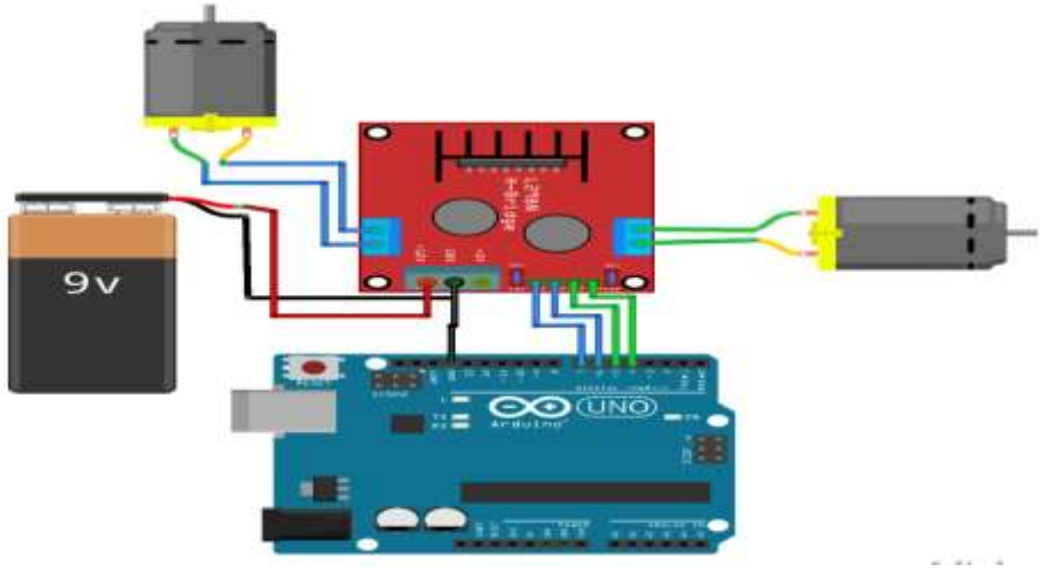
Multiwatt15 H

Bên cạnh đó có jumper 5V, nếu bạn để như hình ở trên thì sẽ có nguồn 5V ra ở cổng 5V power, ngược lại thì không. Bạn để như hình thì ta chỉ cần cấp nguồn 12V vô ở 12V power là có 5V ở 5V power, từ đó cấp cho Arduino

Power GND chân này là GND của nguồn cấp cho Động cơ. Nếu chơi Arduino thì nhớ nối với GND của Arduino

Đây là 2 chân cấp nguồn trực tiếp bạn có thể cấp nguồn 9-12V ở 12V.

2.3.2 Nối mạch



Hình 2.11 Sơ đồ nối mạch

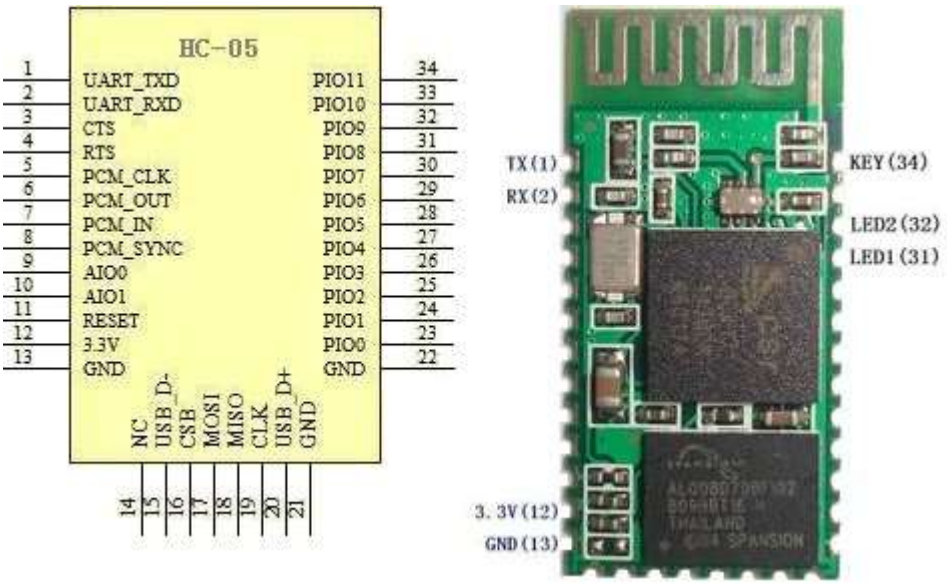
- Nếu bạn điều khiển 2 Động cơ của robot, bạn cần chú ý bài đấu nối Cực +,- của động cơ tương ứng với chân +,- của OUTPUT X.
- Tiếp bạn cấp nguồn cho Module L298 như phần giải thích ở trên. Chú ý chọn Jump cho đúng.
- Nếu bạn dùng 5V và động cơ dưới 1A bạn có thể dùng chân 5V của Arduino, nếu không nguồn cấp cho động cơ ở L298 phải là nguồn riêng để không làm hỏng Arduino của bạn.
- Các chân số D7, D6, D5 và D4 của Arduino sẽ nối tương ứng với IN1, IN2, IN3 và IN4 của L298.
- Chiều quay của động cơ được điều khiển bằng cách xuất các đầu ra HIGH hoặc LOW tại các chân IN_x.
  - Ví dụ với Động Cơ A: Logic HIGH ở IN1 và IN2 Logic LOW sẽ làm động cơ quay 1 hướng nếu đặt Logic ngược lại sẽ làm động cơ quay theo hướng khác.
  - Bạn cần phải nhớ, đây là làm động cơ chỉ quay hết công suất mà thôi. Nếu muốn thay đổi tốc độ của nó, bạn cần phải băm xung bằng các chân có hỗ trợ PWM trên Arduino (những chân có dấu ~).

- Để hiêu rõ, bây giờ mình sẽ giúp các bạn tưởng tượng nhé:

- Tưởng tượng, chân IN1 là chân OutA.1, chân IN2 là chân OutA.2.
- Bạn cấp cực dương vào IN1, cực âm vào IN2 => motor quay một chiều (chiều 1).
- Bạn cấp cực âm vào IN1, cực dương vào IN2 => motor quay chiều còn lại (chiều 2)!
- Cực dương ở đây là điện thế 5V, cực âm ở đây là điện thế 0V. Hiện điện thế được tính là điện thế ở IN1 trừ hiệu điện thế IN2.
- Giả sử, hiệu điện thế 5V sẽ là mạnh nhất trong việc điều khiển động cơ. Như vậy, chỉ cần hạ hiệu điện thế xuống là động cơ sẽ bị yếu đi.
- Và nếu hiệu điện thế < 0 => động cơ sẽ đảo chiều!

2.4 Module Bluetooth

2.4.1 Sơ đồ chân



Hình 2.12 Module Bluetooth

- Điện áp hoạt động: 3.3V.

- Module có 2 chế độ làm việc (có thể lựa chọn chế độ làm việc bằng cách thay đổi trạng thái chân 34 KEY):

- Đáp ứng theo lệnh: khi làm việc ở chế độ này, các bạn có thể gửi các lệnh AT để giao tiếp với module.

- Module HC05 có thể nhận 1 trong 3 chức năng: Master, Slave, Loopback (có thể lựa chọn các chức năng bằng lệnh AT).

- Giao tiếp với module bằng giao tiếp nối tiếp không đồng bộ qua 2 đường RX và TX, vì vậy các bạn có thể sử dụng PC với chuẩn RS232 hoặc các dòng vi điều khiển để giao tiếp.

- Bằng cách thay đổi trạng thái chân 34 (KEY), bạn có thể cấu hình chế độ hoạt động cho module:

- Để module làm việc ở chế độ kết nối tự động: KEY phải ở trạng thái Floating (trạng thái không kết nối).

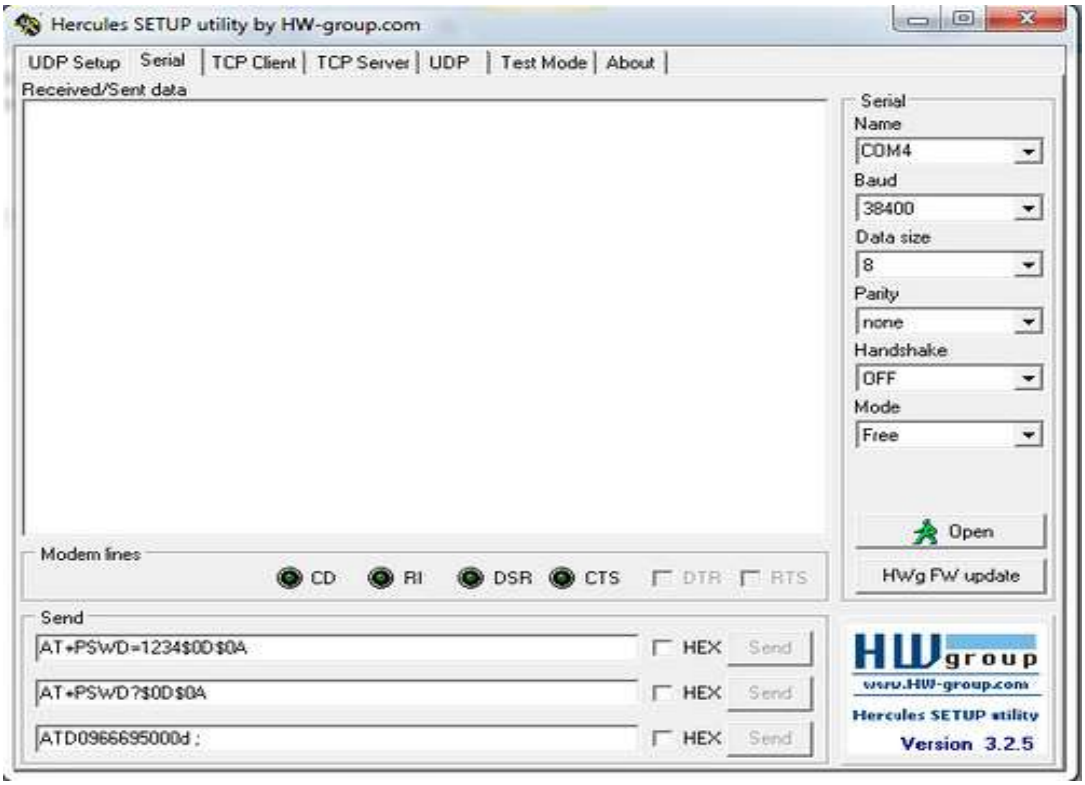
- Để module làm việc ở chế độ đáp ứng theo lệnh: KEY = ‘0’ (kết nối xuống đất)□Cấp nguồn cho module □ chuyển KEY = ‘1’ (kết nối lên VCC) □ lúc này có thể sử dụng các lệnh AT để giao tiếp.

2.3.2 Giao tiếp với Module Bluetooth HC05

Giao tiếp với HC05 bằng các lệnh AT sử dụng phần mềm Hercules Setup Untility trên PC.

Cài đặt phần mềm **Hercules Setup Untility**, sau đó mở ứng dụng, chọn **Serial**, giao diện giao tiếp với cổng nối tiếp sẽ hiện ra:

- Tích hợp RF switch, balun, 24dBm PA, DCXO, and PMU
- Tích hợp bộ xử lý RISC, trên chip bộ nhớ và giao diện bộ nhớ bên ngoài
- Tích hợp bộ vi xử lý MAC / baseband
- Chất lượng quản lý dịch vụ
- Giao diện I2S cho độ trung thực cao ứng dụng âm thanh
- On-chip thấp học sinh bỏ học điều chỉnh tuyến tính cho tất cả các nguồn cung cấp nội bộ



Hình 2.13 Giao tiếp HC05

Các bạn cấu hình cổng vào, chế độ, khung dữ liệu, tốc độ Baud cho cổng nối tiếp.

- Thiết lập module HC05 hoạt động ở chế độ đáp ứng theo lệnh. Ở chế độ này, các bạn có thể cấu hình và kiểm soát module của mình.

- Các bạn kết nối module Bluetooth với PC bằng USB TO COM PL2303 như sau:

RX (màu trắng) à TX của module HC05.

TX (màu xanh lá cây) à RX của module HC05.

ØVCC à 5.0.

ØGND à GND.

- Sử dụng các lệnh AT để giao tiếp với module thông qua hercules setup.

- Module bluetooth HC05 master / slave dùng để thiết lập kết nối Serial giữa 2 thiết bị bằng sóng bluetooth. Điểm đặc biệt của module bluetooth HC-05 là module có thể hoạt động được ở 2 chế độ: MASTER hoặc SLAVE. Trong khi đó, bluetooth module HC-06 chỉ hoạt động ở chế độ SLAVE.

+ Ở chê độ SLAVE: bạn cần thiết lập kết nối từ smartphone, laptop, usb bluetooth để dò tìm module sau đó pair với mã PIN là 1234. Sau khi pair thành công, bạn đã có 1 cổng serial từ xa hoạt động ở baud rate 9600.

+ Ở chế độ MASTER: module sẽ tự động dò tìm thiết bị bluetooth khác (1 module bluetooth HC-06, usb bluetooth, bluetooth của laptop...) và tiến hành pair chủ động mà không cần thiết lập gì từ máy tính hoặc smartphone.

2.3.3 Module bluetooth HC05

Module bluetooth HC05 được điều khiển bằng tập lệnh AT để thực hiện các tác vụ mong muốn. Để bluetooth module chuyển từ chế độ thông thường qua điều khiển bằng AT, ta có 2 cách như sau:

+ Cấp nguồn cho module bluetooth (Vcc và Gnd) đồng thời cấp mức điện áp cao (=Vcc) cho chân KEY của module bluetooth. Khi đó giao tiếp bằng tập lệnh AT với module bằng cổng Serial (Tx và Rx) với baud rate là 38400**.** (khuyên dùng)

+ Cấp nguồn cho module bluetooth trước, sau đó cấp mức điện áp cao cho chân KEY của module bluetooth. Lúc này bạn có thể giao tiếp với module bằng tập lệnh AT với baud rate là 9600.

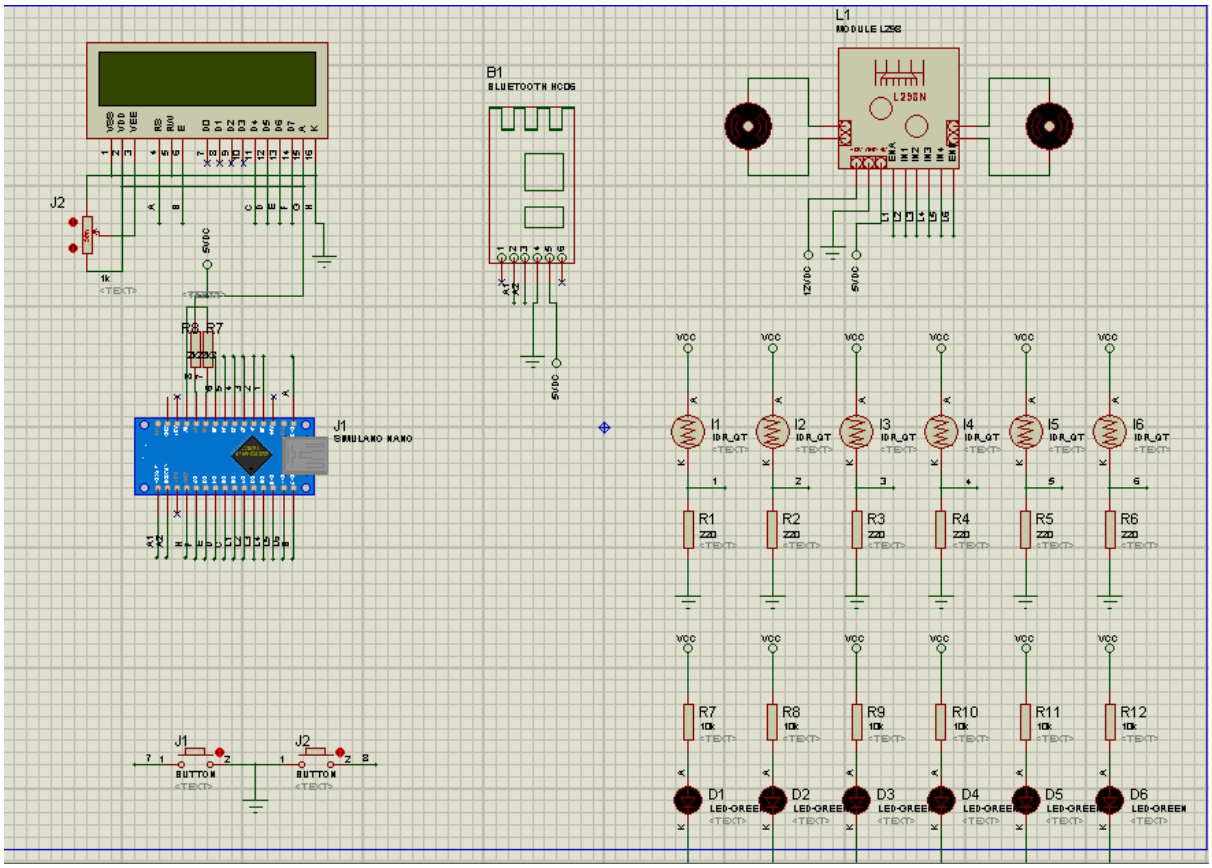
Sau khi pair thành công với thiết bị bluetooth khác, đèn trên module bluetooth HC05 sẽ nhấp nháy chậm cho thấy kết nối Serial đã được thiết lập.

Nguồn cung cấp cho module bluetooth là nguồn từ 3.6V đến 6V. Quá áp sẽ gây cháy module. Ngoài ra module tương thích với các vi điều khiển 5V mà không cần chuyển đổi mức giao tiếp 5V về 3.3V như nhiều loại module bluetooth khác.

Chương 3:

NGUYÊN LÝ TỔNG QUÁT, LƯU ĐỒ GIẢI THUẬT VÀ CHƯƠNG TRÌNH ĐIỀU KHIỂN

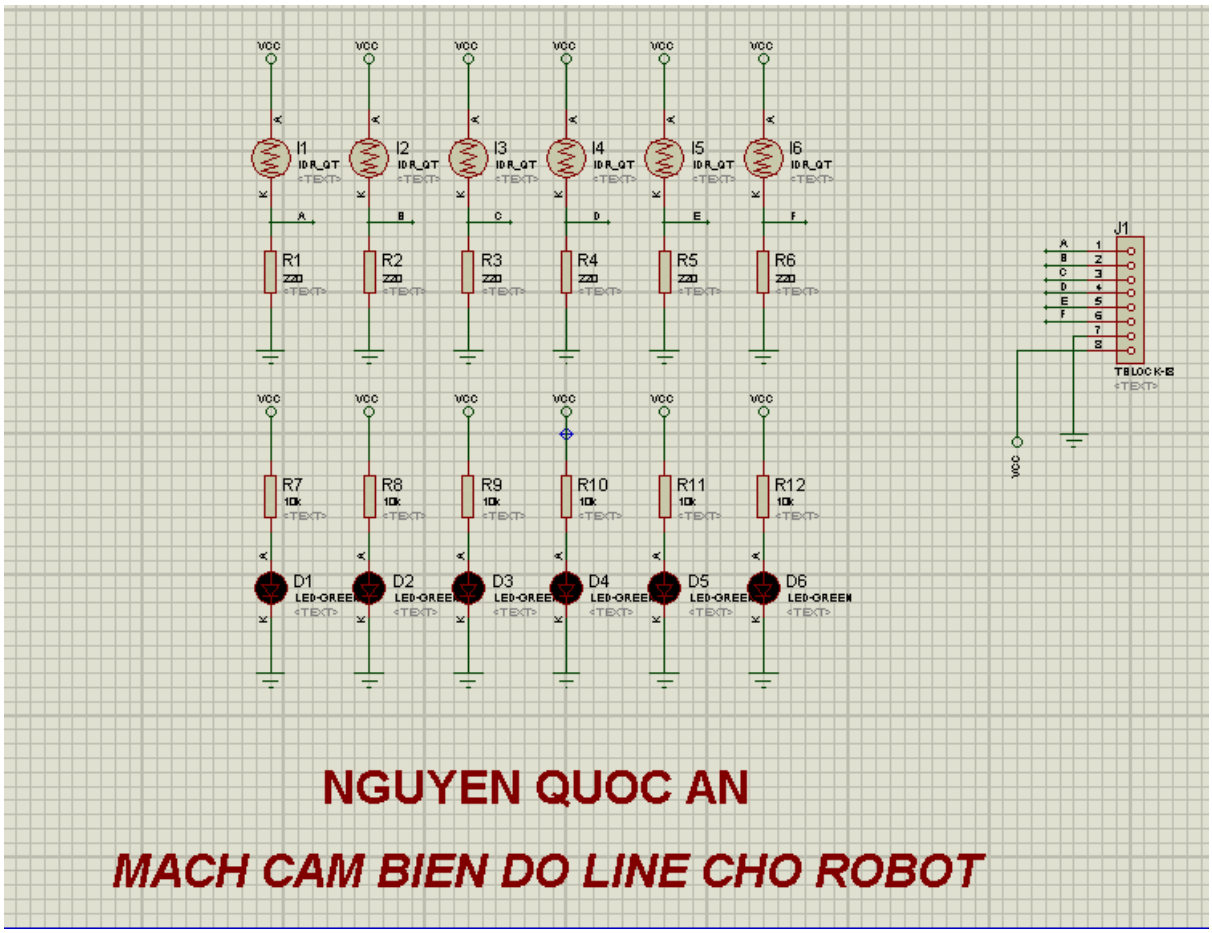
3.1 Nguyên lý tổng quát



Nguyên lý tổng quát của toàn mạch bao gồm các bộ phận kết nối với nhau gồm có : Module L298N, Module Bluetooth HC05, Arduino Nano, màn hình LCD 16x2, cảm biến sensor của mạch.

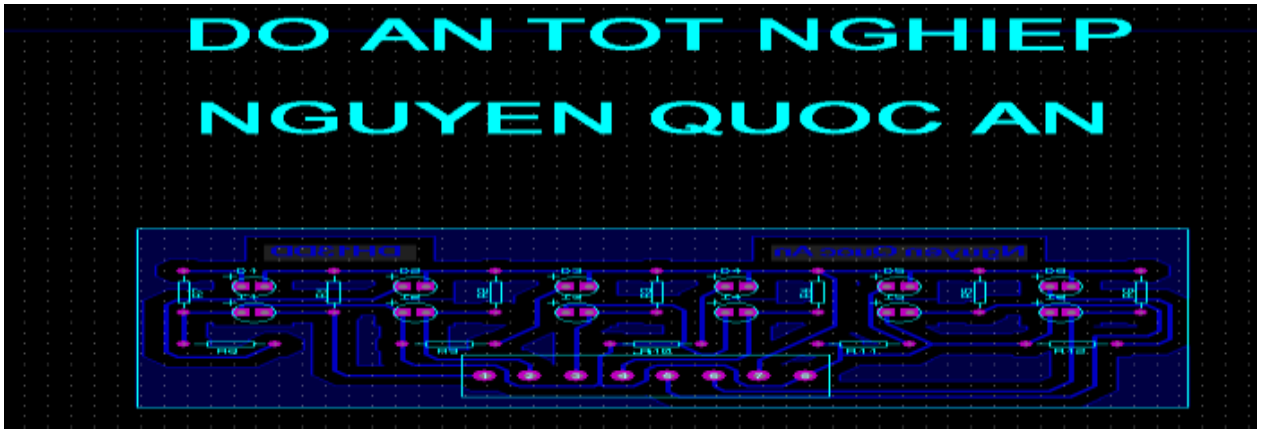
3.2 Mạch cảm biến dò line

3.2.1 Nguyên Lý

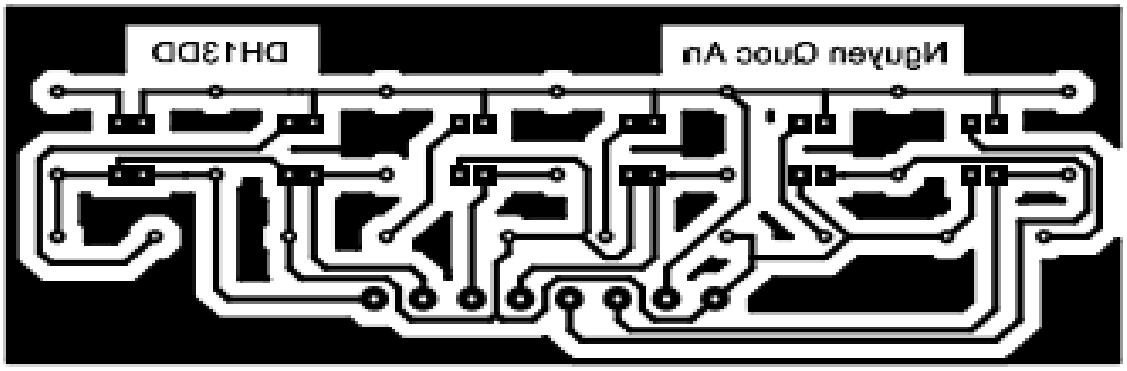


Gồm có 6 led để soi đường và 6 cảm biến quang dùng để cảm biến và phát tín hiệu đến Arduino Nano điều khiển robot.Là một phần rất quang trọng của robot. Vì nhờ có mạch cảm này mà robot có thể di chuyển theo đường line và có thể hoạt động một cách ổn định nhất.

3.2.2 Layout:



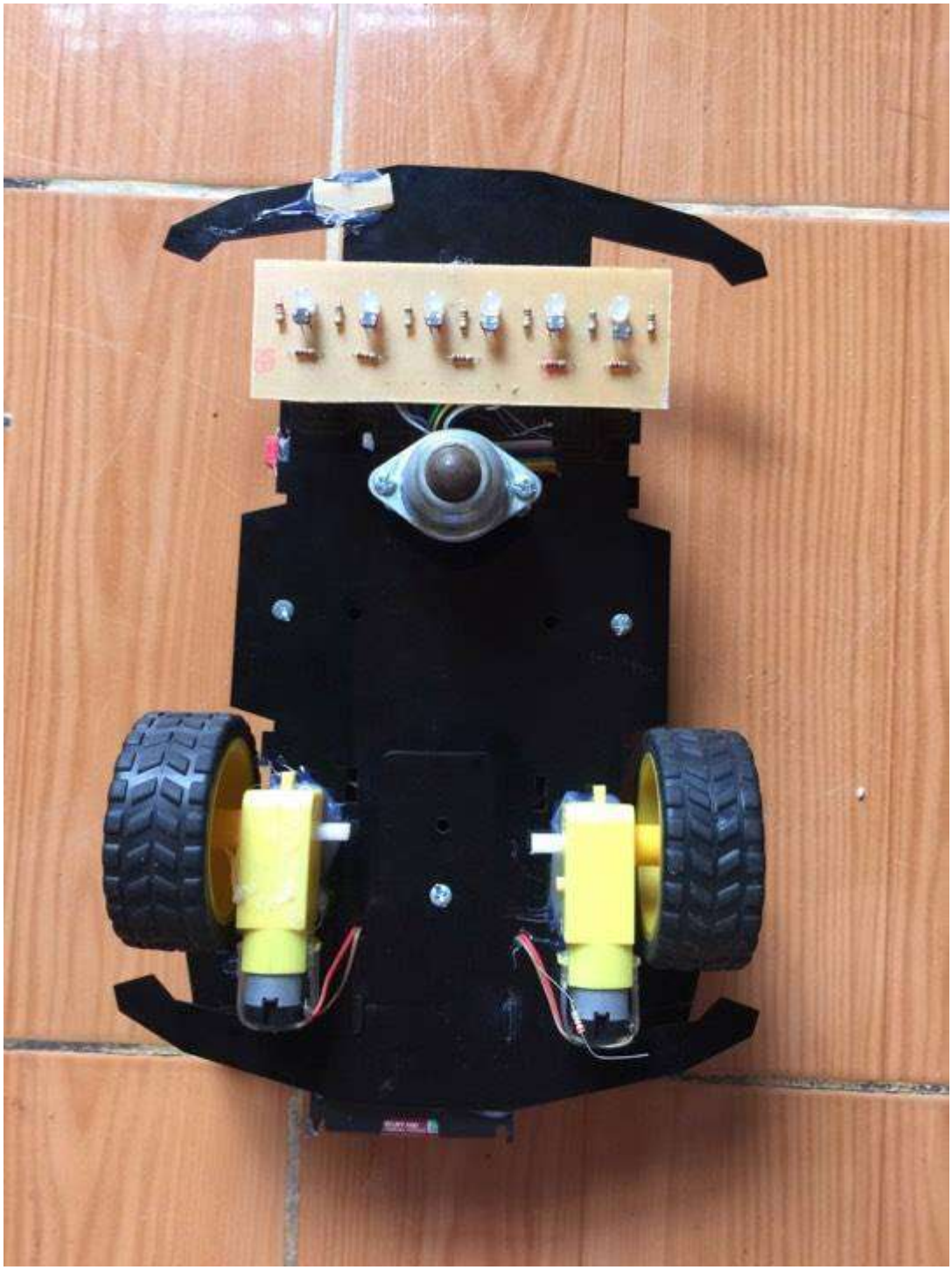
3.2.3 Mạch in



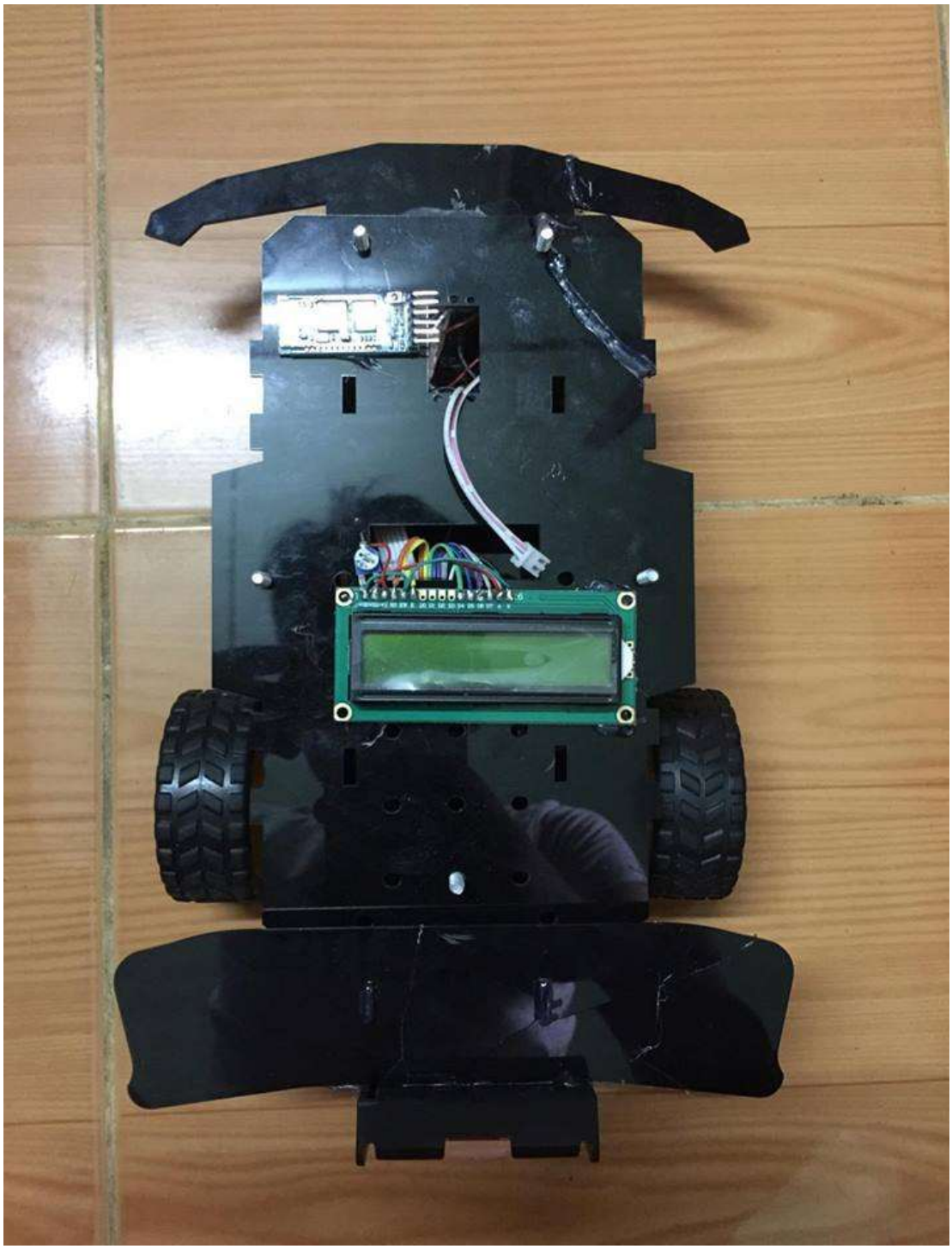
3.2.4 Mạch hoàng chỉnh

3.3 Sản phẩm sau khi hoàn thiện:

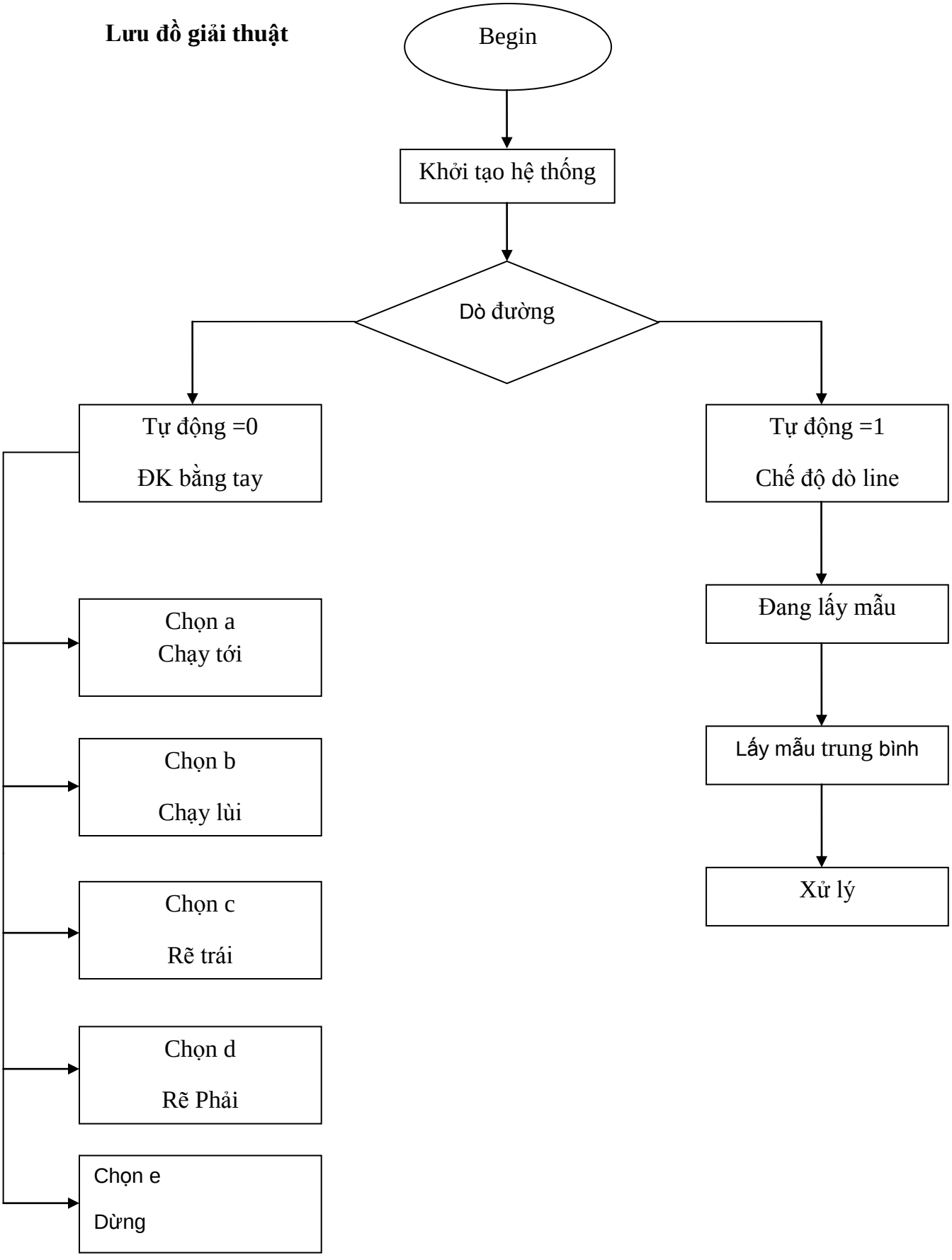
Mặt dưới của robot dò line



Mặt trên của robot dò line



Lưu đồ giải thuật



3.4 Chương trình điều khiển

```
byte inpinA=3;

byte inpinA1=2;

byte inpinB=5;

byte inpinB1=4;

byte PWMA=10;

byte PWMB=9;

#define cb_1 A0

#define cb_2 A1

#define cb_3 A2

#define cb_4 A3

#define cb_5 A4

int i=0,n=0,j=0;

int max_cb[6]={0,0,0,0,0,0};

int min_cb[6]={1023,1023,1023,1023,1023,1023};

int tb_cb[6];

int nut_nhan=53;

int val_cb[6];

int val_th[6];

int truong_hop;
```

```
int tocdo=100;
int vach_ngang=0;
int value=0,tudong=0;
void setup(){
  Serial.begin(9600);
   pinMode(12,INPUT);
  pinMode(inpinA,OUTPUT);
  pinMode(inpinA1,OUTPUT);
  pinMode(inpinB,OUTPUT);
  pinMode(inpinB1,OUTPUT);
  pinMode(PWMA,OUTPUT);
  pinMode(PWMB,OUTPUT);
  pinMode(nut_nhan,INPUT_PULLUP);
void lay_mau(){
  Serial.println("dang lay mau");
    val_cb[0] = analogRead(A0);
    val_cb[1] = analogRead(A1);
    val_cb[2] = analogRead(A2);
    val_cb[3] = analogRead(A3);
    val_cb[4] = analogRead(A4);
```

```
    val_cb[5] = analogRead(A5);
  for(i=0; i<6; i++){
          if(max_cb[i]<val_cb[i])
          {max_cb[i]=val_cb[i];}
          if(min_cb[i]>val_cb[i])
          {min_cb[i]=val_cb[i];}
 }
   delay(10);

}
void trung_binh(){
  Serial.println("dang lay trung binh");
   for(i=0; i<6; i++)
   {
      tb_cb[i]=(max_cb[i]+min_cb[i])/2;
   }

}
void xu_li(){
val_cb[0] = analogRead(A0);
```

```
    val_cb[1] = analogRead(A1);
    val_cb[2] = analogRead(A2);
    val_cb[3] = analogRead(A3);
    val_cb[4] = analogRead(A4);
     val_cb[5] = analogRead(A5);
  ///////////////
  for(int e=0;e<6;e++)
   {
    if( val_cb[e]< tb_cb[e])
    { val_th[e]=0; }
    else
    {  val_th[e]=1;}

   }
     ///////////////

if((val_th[0]==1)&&(val_th[1]==1)&&(val_th[2]==1)&&(val_th[3]==1)&&(val_th[4]==1)&&(val_th[5]==1))
    {truong_hop=0;Serial.println("qua vach ngang");}

if((val_th[0]==0)&&(val_th[1]==0)&&(val_th[2]==0)&&(val_th[3]==0)&&(val_th[4]==0)&&(val_th[5]==0))
```

```
{truong_hop=1;Serial.println(" mat line");}

if((val_th[0]==1)&&(val_th[1]==1)&&(val_th[2]==0)&&(val_th[3]==0)&&(val_th[4]==0)&&(val_th[5]==0))

{truong_hop=2;Serial.println("lech trai1!!!");}

if((val_th[0]==0)&&(val_th[1]==1)&&(val_th[2]==0)&&(val_th[3]==0)&&(val_th[4]==0)&&(val_th[5]==0))

{truong_hop=3;Serial.println("lech trai1!!!");}

if((val_th[0]==0)&&(val_th[1]==1)&&(val_th[2]==1)&&(val_th[3]==0)&&(val_th[4]==0)&&(val_th[5]==0))

{truong_hop=4;Serial.println("lech trai1!!!");}

if((val_th[0]==0)&&(val_th[1]==0)&&(val_th[2]==1)&&(val_th[3]==0)&&(val_th[4]==0)&&(val_th[5]==0))

{truong_hop=5;Serial.println("lech trai1!!!");}

if((val_th[0]==0)&&(val_th[1]==0)&&(val_th[2]==1)&&(val_th[3]==1)&&(val_th[4]==0)&&(val_th[5]==0))

{truong_hop=6;Serial.println("khong lech");}

if((val_th[0]==0)&&(val_th[1]==0)&&(val_th[2]==0)&&(val_th[3]==1)&&(val_th[4]==0)&&(val_th[5]==0))
```

```
{truong_hop=7;Serial.println("lech phai1!!!");}

if((val_th[0]==0)&&(val_th[1]==0)&&(val_th[2]==0)&&(val_th[3]==1)&&(val_th[4]==1)&&(val_th[5]==0))
{truong_hop=8;Serial.println("lech phai1!!!");}

if((val_th[0]==0)&&(val_th[1]==0)&&(val_th[2]==0)&&(val_th[3]==0)&&(val_th[4]==1)&&(val_th[5]==1))
{truong_hop=9;Serial.println("lech phai1!!!");}

if((val_th[0]==0)&&(val_th[1]==0)&&(val_th[2]==0)&&(val_th[3]==0)&&(val_th[4]==0)&&(val_th[5]==1))
{truong_hop=10;Serial.println("lech phai1!!!");}
}
void loop(){

if (Serial.available())
{
 value = Serial.read();
}
 switch (value)
 {
```

```
case 'a' : chay_toi(100,100); Serial.println("chay toi");break;
case 'b' : chay_lui(100,100); Serial.println("chay lui");break;
case 'c' : trai(100,100); Serial.println("re trai");break;
case 'd' : phai(100,100); Serial.println("re phai");break;
case 'e' : dung(); Serial.println("dung");break;
case 'f' : tudong=0; Serial.println("che do dieu hien bang tay");delay(2000);break;
case 'g' : tudong=1; Serial.println("che do chay do line");delay(2000);break;

}
if(tudong==1)
{
 xu_li();
}
}
void doc_cam_bien(){
}
void dung(){
 digitalWrite(inpinA,LOW);
 digitalWrite(inpinA1,LOW);
```

```
  digitalWrite(inpinB,LOW);
  digitalWrite(inpinB1,LOW);
  analogWrite(PWMA,0);
  analogWrite(PWMB,0);
}
void chay_lui(int vt,int vt1){
  digitalWrite(inpinA,HIGH);
  digitalWrite(inpinA1,LOW);
  digitalWrite(inpinB,LOW);
  digitalWrite(inpinB1,HIGH);
  analogWrite(PWMA,vt);
  analogWrite(PWMB,vt);
}
void chay_toi(int vt,int vt1){
  digitalWrite(inpinA,LOW);
  digitalWrite(inpinA1,HIGH);
  digitalWrite(inpinB,HIGH);
  digitalWrite(inpinB1,LOW);
  analogWrite(PWMA,vt);
  analogWrite(PWMB,vt1);
```

```
}
void phai(int vt, int vt1){
  digitalWrite(inpinA,LOW);
  digitalWrite(inpinA1,HIGH);
  digitalWrite(inpinB,LOW);
  digitalWrite(inpinB1,HIGH);
  analogWrite(PWMA,vt);
  analogWrite(PWMB,vt1);
}
void trai(int vt, int vt1){
  digitalWrite(inpinA,HIGH);
  digitalWrite(inpinA1,LOW);
  digitalWrite(inpinB,HIGH);
  digitalWrite(inpinB1,LOW);
  analogWrite(PWMA,vt);
  analogWrite(PWMB,vt1);
}
```

Chương 4:

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết quả

- Kết quả tương đối chính xác

- Tốc độ chậm

- Những chỗ rẽ mạnh thì xe còn sai sót

- Mỗi lần cần chạy tự động thì cần lấy mẫu cho xe

4.2 Hướng phát triển

- Cải tiến xe nhanh hơn, tải trọng lớn hơn nhưng cũng bám đường tốt hơn

- Xây dựng giải thuật hoàn chỉnh hơn.

- Có thể cải tiến xe kết hợp dò đường và tránh vật cản đồng thời nhớ đường và tìm đường đi nhanh nhất.

TÀI LIỆU THAM KHẢO

- Dientuvietnam.net

- Hoclamrobot.com

- Codientu.org

- Giáo trình lý thuyết tự động – Nguyễn Thị Phương Hà(chủ biên), Huỳnh Thái Hoàng

-Bài giảng hệ thống điều khiển – Nguyễn Vĩnh Hảo