ĐỒ ÁN TỐT NGHIỆP

GIAO TIẾP VI ĐIỀU KHIỂN VÀ MÁY TÍNH

GVHD: Trần Thái Anh Âu

MỤC LỤC

CHƯƠNG 1 :
TỔNG QUAN VỀ VI ĐIỀU KHIỂN PIC

1.1. Giới thiệu chung về PIC

PIC bắt nguồn từ chữ viết tắt của "Programmable intelligen computer" (Máy tính khả trình thông minh) là sản phẩm của hãng General Instrument đặt cho dòng sản phẩm đầu tiên của họ là PIC 1650. Lúc này Pic dùng để giao tiếp với các thiết bị ngoại vi cho máy chủ 16 bit CP1600, vì vậy người ta gọi PIC với tên là "Peripheral Interface Controller" (bộ điều khiển giao tiếp ngoại vi).

Năm 1985 General Instrument bán bộ phận vi điện tử của họ, và chủ sở hữu mới huỷ bỏ hầu hết các dự án – lúc đó đã quá lỗi thời. Tuy nhiên PIC được bổ sung EEPROM để tạo thành một bộ điều khiển vào ra khả trình. Ngày nay rất nhiều dòng PIC được xuất xưởng với hàng loạt các module ngoại vi được tích hợp sẵn (như :USART, PWM, ADC…) với bộ nhớ chương trình từ 512word đến 32kWord.

PIC sử dụng tập lệnh RISC, với dòng PIC low-end (độ dài mã lệnh 12 bit ví dụ PIC12Cxxx) và mid-range (độ dài mã lệnh 14 bit, ví dụ PIC16Fxxx), tập lệnh bao gồm khoảng 35 lệnh, và 70 lệnh đối với dòng PIC high-end (có độ dài mã lệnh 16bit PIC18Fxxxx). Tập lệnh bao gồm các lệnh tính toán trên các thanh ghi, và các hằng số, hoặc các vị trí ô nhớ, cũng như có các lệnh điều kiện, nhảy, gọi hàm, và các lệnh quay trở về, nó cũng có các chức năng phần cứng khác như ngắt hoặc sleep (chế độ hoạt động tiết kiệm điện). Microchip cung cấp môi trường lập trình MPLAB, nó bao gồm phần mềm mô phỏng và trình dịch ASM.

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

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

Flash và Rom có thể tuỳ chọn 256 byte đến 256 kbybe

Các cổng xuất/ nhập (mức lôgic thường từ 0V đến 5V, ứng với mức logic 0 và 1)

8/16 bit timer

* Các chuẩn giao tiếp ngoại vi nối tiếp đồng bộ/ không đồng bộ

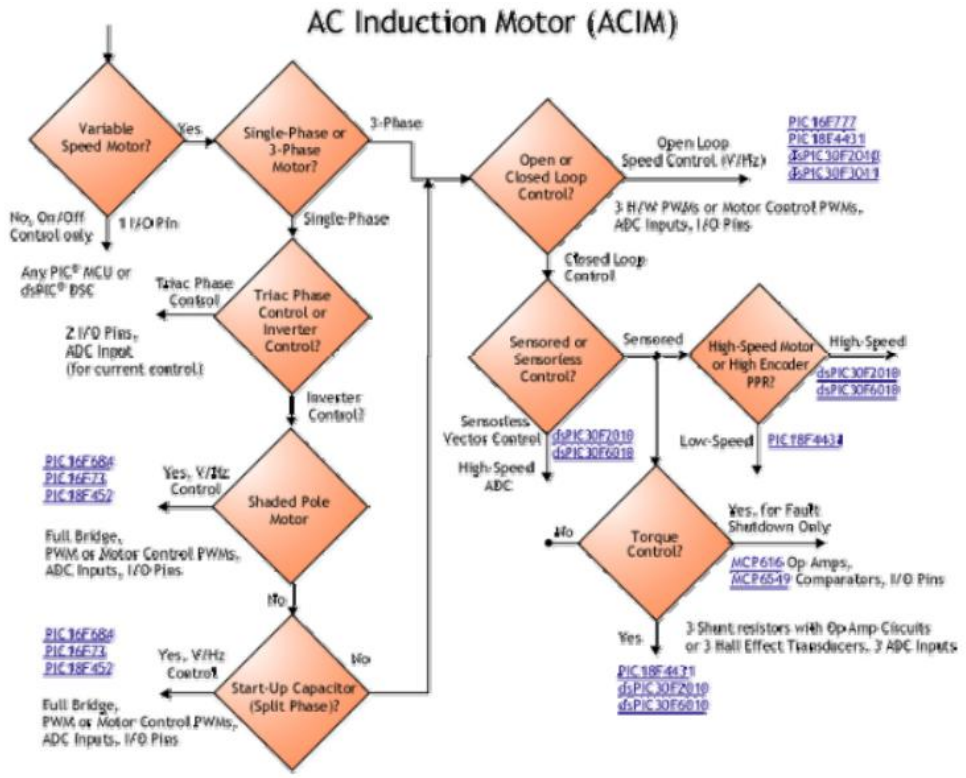
+ Bộ chuyển đổi ADC
+ Bộ so sánh điện áp
+ MSSP Pripheral dùng cho các giao tiếp I2C, SPI và I2S
+ Bộ nhớ nội EEPROM - có thể ghi/ xoá lên tới hàng triệu lần
+ Modul điều khiển động cơ đọc encoder
+ Hỗ trợ giao tiếp USB
+ Hỗ trợ điều khiển Ethernet
+ Hỗ trợ giao tiếp CAN
+ Hỗ trợ giao tiếp LIN
+ Hỗ trợ giao tiếp IrDA

Họ vi điều khiển PIC và dsPIC do hãng chế tạo và sản xuất với công nghệ hiện đại, phù hợp cho các ứng dụng đơn giản cho đến phức tạp. Đặc biệt ngoài ngôn ngữ lập trình assembler như các MCU khác, người dùng có thể lập trình PIC trên ngôn ngữ C quen thuộc thông qua các phần mềm hỗ trợ (PIC18C ; CCS C;…….)

Gồm các họ như sau:

- 8 bit:
  +PIC10
  +PIC12
  +PIC16
  +PIC18
- 16 bit:
  +PIC24F
  +PIC24H
  +dsPIC30
  +dsPIC33

Tùy theo các ứng dụng cụ thể mà người dùng có thể chọn ra Chip phù hợp (theo hướng dẫn của nhà sản xuất tại trang chủ của microchip).Trong đó PIC18F4431 là IC chuyên dùng để điều khiển động cơ .



Hình 1.1:Quá trình phát triển PIC

1.2. Tìm hiểu PIC18F4431

1.2.1. Những đặc điểm nổi bậc PIC18F4431:

14 bit Power Control PWM module:

+ Có đến 4 kênh (mỗi kênh gồm 1 cặp xung đối nghịch)
+ Thời gian dead time linh hoạt.
+ update từng duty cycle => ngõ ra PWM đáp ứng nhanh
+….

Motion Feedback Module:

+ Có 3 kênh capture độc lập:
- các chế độ hoạt động linh hoạt cho việc đo đạc độ rụng xung
- Module hỗ trợ Hall Sensor
- Special event trigger cho các module khác
+ Quadrature Encorder interface:
- 2 pha vào và 1 ngõ vào index từ encorder
- hỗ trợ đo đạc vận tốc

High speed, 200Ksps 10-bit A/D Converter:

+ Có 9 kênh A/D
+ 2 kênh lấy mẫu tức thời

+ Lấy mẫu liên tục:1 ; 2 hay 4 kênh được lựa chọn

+ …….

Flexible Oscillator Structure:

+ 4 chế độ thạch anh (hỗ trợ đến 40 MHz)

+ 2 nguồn xung lock ngoài lên đến 40 MHz

+ Chế độ thạch anh nội :

- Có 8 tần số người dùng có thể lựa chọn : từ 31Khz -> 8 MHz

- OSCTUNE có thể bù cho sự lệch tần số ()

+…..

Peripheral Highlights:

+ Chịu dòng cao : sink/source (25mA/25ma)

+ 3 nguồn ngắt ngoài

+ 2 module Capture / Compare / PWM (CCP)

- Capture 16 bit, độ phân giải tối đa 6.25 ns ($T_{CY}/6$)

- Compare 16 bit, độ phân giải tối đa 100 ns (T_{CY})

- PWM output: độ phân giải từ 1 -> 10 bít

+ Module USART:

-Hỗ trợ RS-485, RS-232 và LIN1.2

- Auto weak-up on start bit

- Auto-Bound detect

+ RS-232 sử dụng khối dao động nội (ko cần thạch anh ngoài)

Những đặc điểm chính:

+ Là CPU sử dụng tập lệnh RISC và có tốc độ xử lý cao , công suất thấp nhờ sử dụng công nghệ CMOS FLASH/EEPROM.

+ Tập lệnh có 75 lệnh .

+ Một chu kỳ lệnh bằng 4 chu kỳ xung . Sử dụng bộ dao động 40 Mhz thì chu kỳ lệnh là 0,1 us .

+ Tần số bộ dao động cho phép tới 40Mhz.

+ 8K x 14 word bộ nhớ FLASH lập trình.

+ 768 byte bộ nhớ RAM , trong đó bộ nhớ EEPROM lên đến 256 byte.

+ Trang bị tới 34 ngắt với 8 cấp độ ngắt

+ 5 port I / O.

+ Trang bị 3 bộ định thời: 2 bộ 8 bit,1 bộ 16 bit.

+ 2 module Capture/Compare/PWM.

+ Bộ chuyển đổi 10 bit ADC với tốc độ 5-10us.

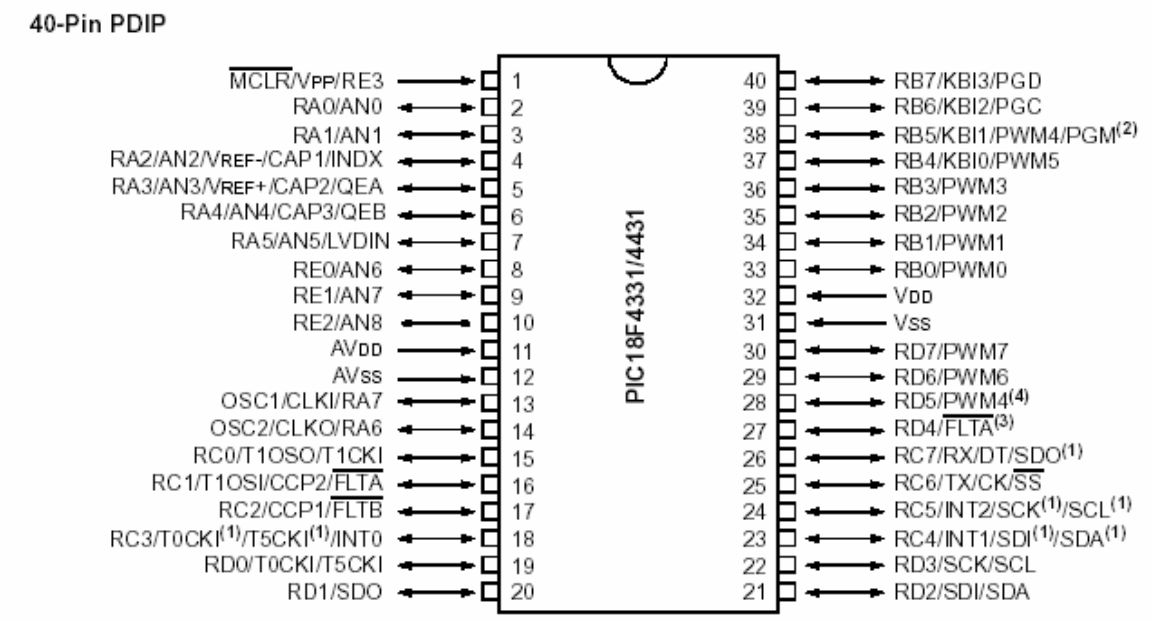
+ Cổng serial đồng bộ với chế độ SPI(Master) và I2C (Master/Slave) thực hiện bằng phần cứng .

+ Chế độ chuyển nhận đồng bộ/bất đồng bộ với 9 bit địa chỉ kiểm tra.

+ Cổng song song (PSP) 8bit .
+ Các chế độ định địa chỉ:trực tiếp , gián tiếp , và tương đối.
+ Cho phép đọc/ghi bộ nhớ chương trình .
+ Có chế độ bảo vệ mã lập trình .
+ Chế độ SLEEP(tạm nghỉ) để tiết kiệm điện năng .
+ Cho phép chọn lựa chế độ dao động (nội , ngoại).
+ 2 chân cho phép gỡ rối hoạt động của vi điều khiển.
+ Lập trình thông qua cổng serial với điện thế chỉ 5 V.
+ Tầm điện thế hoạt động rộng: từ 2 đến 5.5V. Dòng cấp khoảng 25mA.
+ Được sản xuất với nhiều loại khác nhau cho cùng 1 mã vi điều khiển, tuỳ thuộc vào số tính năng được trang bị thêm . Các kiểu đế cắm:PDIP(40 chân), PLCC và QFP (44 chân).

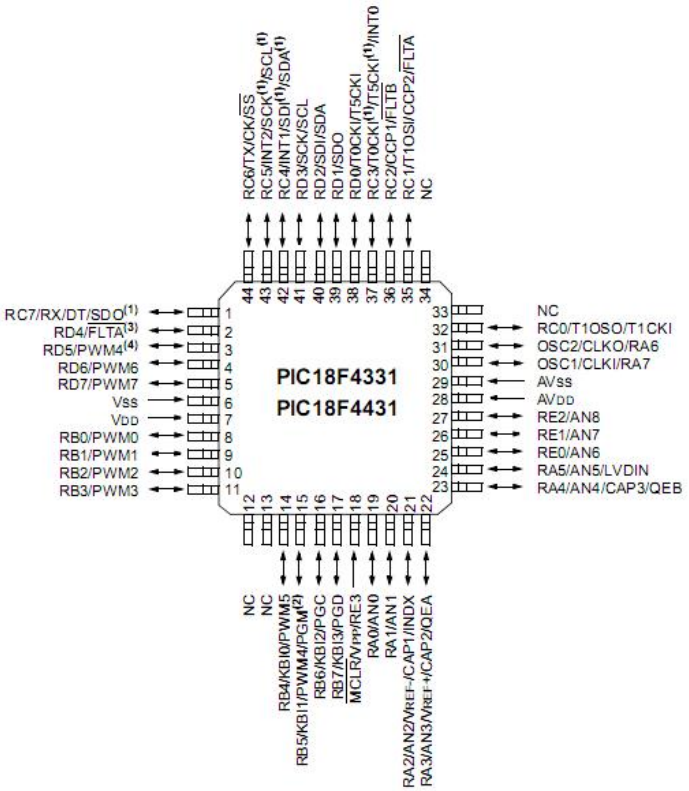
1.2.2. Tóm tắt phần cứng:

1.2.2.1. Sơ đồ chân MCU PIC18F4431 :



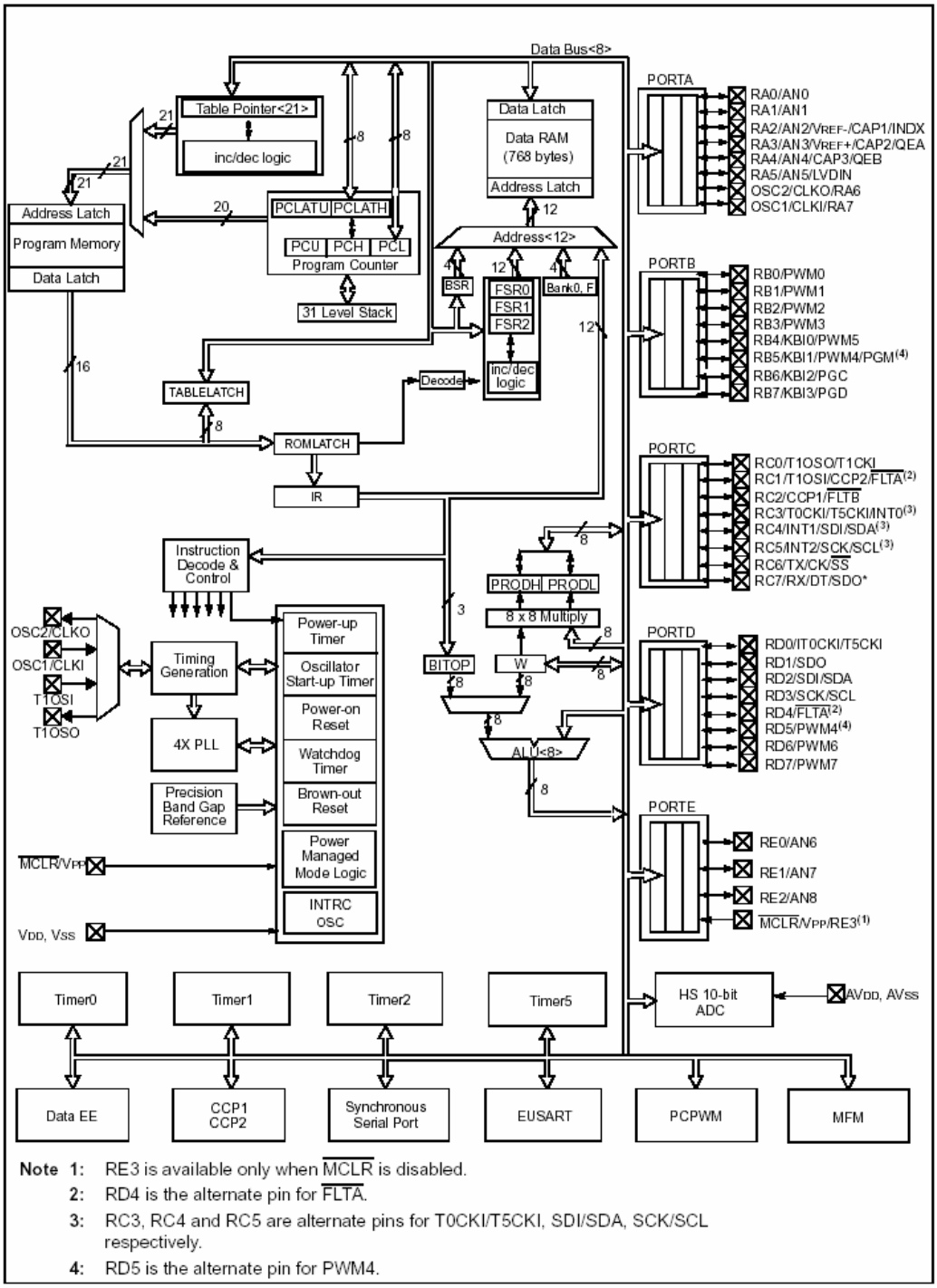
Hình 1.2.2.1:Sơ đồ chân PIC18f4431

1.2.2.2. Sơ đồ các khối chức năng



Hình 1.2.2.2.1:Sơ đồ chân PIC18f4431

FIGURE 1-2: PIC18F4331/4431 BLOCK DIAGRAM



Note 1: RE3 is available only when MCLR is disabled.
2: RD4 is the alternate pin for FLTA.
3: RC3, RC4 and RC5 are alternate pins for T0CKI/T5CKI, SDI/SDA, SCK/SCL respectively.
4: RD5 is the alternate pin for PWM4.

Hình 1.2.2.2.2 :Sơ đồ cấu trúc cơ bản PIC18f4431

1.2.2.3. Chức năng của từng chân

a)_PORT A:

+ Là port I/O . Có tất cả 6 chân, từ RA0 đến RA5.Trong đó RA2 và RA3 có thể dùng tiếp nhận điện áp Vref+ và Vref-.

+ RA4 còn là ngõ vào xung clock cho Timer0. RA5 có thể làm chân chọn slave cho port serial đồng bộ.

b)_PORT B:

+ Là port I/O ,có thể được lập trình bởi phần mềm để làm chức năng kéo lên cho tất cả ngõ vào.

+ RB0 có thể làm chân ngắt ngoài.

+ RB3 có thể làm ngõ vào lập trình điện thế thấp.

+ Các chân còn lại có thể làm ngõ vào ngắt trên chân, lập trình với xung và dữ liệu serial.

c)_PORT C:

+ Là port I/O, có 8 chân:

+ RC0 dùng làm ngõ ra bộ dao động Timer1 hoặc ngõ vào xung timer1.

+ RC1, RC2 có cùng 3 chức năng: làm ngõ ra PWM / chân Compare(so sánh) / chân capture (lấy mẫu).RC1 còn là ngõ vào bộ dao động Timer1.

+ RC3 là ngõ vào xung tuần tự đồng bộ/ hoặc ra (với chế độ SPI và I2C).

+ RC4 làm chân nhận data (chế độ SPI) hay data I/O (chế độ I2C).

+ RC5 có thể xuất data SPI (chế độ SPI).

+ RC6 có thể làm chân phát bất đồng bộ (USART) hoặc xung đồng bộ.

d)_PORT D:

+ Là port I/O ,có thể làm port slave song song khi giao tiếp với 1 bus vi xử lý.

e)_PORT E:

+ Port I/O này thường dùng điều khiển chọn/đọc/ghi cho port slave song song.

f)_Các chân khác:

+ Chân 13(OSC1/CLKIN) tiếp nhận xung ngoài cho bộ dao động thạch anh bên trong.

+ Chân 14(OSC2/CLKOUT) làm ngõ ra bộ dao động thạch anh.Ở chế độ RC,chân này có tần số bằng ¼ của OSC1.

+ Chân 1 : làm ngõ vào reset .

+ Chân 12, 31 là nối đất Vss.

+ Chân 11, 32 là chân cấp nguồn Vdd.

Mô tả các I/O trích từ datasheet :

TABLE 1-3: PIC18F4331/4431 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	TQFP	QFN			
MCLR/VPP/RE3	1	18	18			Master Clear (input) or programming voltage (input).
MCLR				I	ST	Master Clear (Reset) input. This pin is an active-low. Reset to the device.
VPP				P		Programming voltage input.
RE3				I	ST	Digital input. Available only when MCLR is disabled.
OSC1/CLKI/RA7	13	30	32			Oscillator crystal or external clock input.
OSC1				I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise.
CLKI				I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
RA7				I/O	TTL	General purpose I/O pin.
OSC2/CLKO/RA6	14	31	33			Oscillator crystal or clock output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General purpose I/O pin.
						PORTA is a bidirectional I/O port.
RA0/AN0	2	19	19			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	3	20	20			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1.
RA2/AN2/VREF-/CAP1/INDX	4	21	21			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
VREF-				I	Analog	A/D Reference Voltage (Low) input.
CAP1				I	ST	Input capture pin 1.
INDX				I	ST	Quadrature Encoder Interface Index input pin.
RA3/AN3/VREF+/CAP2/QEA	5	22	22			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
VREF+				I	Analog	A/D Reference Voltage (High) input.
CAP2				I	ST	Input capture pin 2.
QEA				I	ST	Quadrature Encoder Interface channel A input pin.
RA4/AN4/CAP3/QEB	6	23	23			
RA4				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
CAP3				I	ST	Input capture pin 3.
QEB				I	ST	Quadrature Encoder Interface channel B input pin.
RA5/AN5/LVDIN	7	24	24			
RA5				I/O	TTL	Digital I/O.
AN5				I	Analog	Analog input 5.
LVDIN				I	Analog	Low-voltage Detect input.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power; OD = Open-Drain (no diode to VDD)

TABLE 1-3: PIC18F4331/4431 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	TQFP	QFN			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/PWM0	33	8	9			
RB0				I/O	TTL	Digital I/O.
PWM0				O	TTL	PWM output 0.
RB1/PWM1	34	9	10			
RB1				I/O	TTL	Digital I/O.
PWM1				O	TTL	PWM output 1.
RB2/PWM2	35	10	11			
RB2				I/O	TTL	Digital I/O.
PWM2				O	TTL	PWM output 2.
RB3/PWM3	36	11	12			
RB3				I/O	TTL	Digital I/O.
PWM3				O	TTL	PWM output 3.
RB4/KBI0/PWM5	37	14	14			
RB4				I/O	TTL	Digital I/O.
KBI0				I	TTL	Interrupt-on-change pin.
PWM5				O	TTL	PWM output 5.
RB5/KBI1/PWM4/ PGM	38	15	15			
RB5				I/O	TTL	Digital I/O.
KBI1				I	TTL	Interrupt-on-change pin.
PWM4				O	TTL	PWM output 4.
PGM				I/O	ST	Low-voltage ICSP programming entry pin.
RB6/KBI2/PGC	39	16	16			
RB6				I/O	TTL	Digital I/O.
KBI2				I	TTL	Interrupt-on-change pin.
PGC				I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	40	17	17			
RB7				I/O	TTL	Digital I/O.
KBI3				I	TTL	Interrupt-on-change pin.
PGD				I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power; OD = Open-Drain (no diode to VDD)

TABLE 1-3: PIC18F4331/4431 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	TQFP	QFN			
						PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI	15	32	34			
RC0				I/O	ST	Digital I/O.
T1OSO				O	—	Timer1 oscillator output.
T1CKI				I	ST	Timer1 external clock input.
RC1/T1OSI/CCP2/$\overline{FLTA}$	16	35	35			
RC1				I/O	ST	Digital I/O.
T1OSI				I	CMOS	Timer1 oscillator input.
CCP2				I/O	ST	Capture2 input, Compare2 output, PWM2 output.
$\overline{FLTA}$				I	ST	Fault interrupt input pin.
RC2/CCP1/$\overline{FLTB}$	17	36	36			
RC2				I/O	ST	Digital I/O.
CCP1				I/O	ST	Capture1 input/Compare1 output/PWM1 output.
$\overline{FLTB}$				I	ST	Fault interrupt input pin.
RC3/T0CKI/T5CKI/INT0	18	37	37			
RC3				I/O	ST	Digital I/O.
T0CKI				I	ST	Timer0 alternate clock input.
T5CKI				I	ST	Timer5 alternate clock input.
INT0				I	ST	External interrupt 0.
RC4/INT1/SDI/SDA	23	42	42			
RC4				I/O	ST	Digital I/O.
INT1				I	ST	External interrupt 1.
SDI				I	ST	SPI Data in.
SDA				I/O	ST	I^2C Data I/O.
RC5/INT2/SCK/SCL	24	43	43			
RC5				I/O	ST	Digital I/O.
INT2				I	ST	External interrupt 2.
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C mode.
RC6/TX/CK/$\overline{SS}$	25	44	44			
RC6				I/O	ST	Digital I/O.
TX				O	—	USART Asynchronous Transmit.
CK				I/O	ST	USART Synchronous Clock (see related RX/DT).
$\overline{SS}$				I	ST	SPI Slave Select input.
RC7/RX/DT/SDO	26	1	1			
RC7				I/O	ST	Digital I/O.
RX				I	ST	USART Asynchronous Receive.
DT				I/O	ST	USART Synchronous Data (see related TX/CK).
SDO				O	—	SPI Data out.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power; OD = Open-Drain (no diode to VDD)

TABLE 1-3: PIC18F4331/4431 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number DIP	Pin Number TQFP	Pin Number QFN	Pin Type	Buffer Type	Description
						PORTD is a bidirectional I/O port, or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
RD0/T0CKI/T5CKI	19	38	38			
RD0				I/O	ST	Digital I/O.
T0CKI				I	ST	Timer0 external clock input.
T5CKI				I	ST	Timer5 input clock.
RD1/SDO	20	39	39			
RD1				I/O	ST	Digital I/O.
SDO				O	—	SPI Data out.
RD2/SDI/SDA	21	40	40			
RD2				I/O	ST	Digital I/O.
SDI				I	ST	SPI Data in.
SDA				I/O	ST	I^2C Data I/O.
RD3/SCK/SCL	22	41	41			
RD3				I/O	ST	Digital I/O.
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C mode.
RD4/$\overline{\text{FLTA}}$	27	2	2			
RD4				I/O	ST	Digital I/O.
$\overline{\text{FLTA}}$				I	ST	Fault interrupt input pin.
RD5/PWM4	28	3	3			
RD5				I/O	ST	Digital I/O.
PWM4				O	TTL	PWM output 4.
RD6/PWM6	29	4	4			
RD6				I/O	ST	Digital I/O.
PWM6				O	TTL	PWM output 6.
RD7/PWM7	30	5	5			
RD7				I/O	ST	Digital I/O.
PWM7				O	TTL	PWM output 7.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power; OD = Open-Drain (no diode to VDD)

Pin Name	Pin Number DIP	Pin Number TQFP	Pin Number QFN	Pin Type	Buffer Type	Description
						PORTE is a bidirectional I/O port.
RE0/AN6	8	25	25			
RE0				I/O	ST	Digital I/O.
AN6				I	Analog	Analog input 6.
RE1/AN7	9	26	26			
RE1				I/O	ST	Digital I/O.
AN7				I	Analog	Analog input 7.
RE2/AN8	10	27	27			
RE2				I/O	ST	Digital I/O.
AN8				I	Analog	Analog input 8.
VSS	12, 31	6, 29	6, 30, 31	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	7, 28	7, 8, 28, 29	P	—	Positive supply for logic and I/O pins.
NC	—	12, 13, 33, 34	13	NC	NC	No connect

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power; OD = Open-Drain (no diode to VDD)

1.3. Các module cơ bản

1.3.1. Power control PWM module

Power Control PWM module đơn giản là tạo ra nhiều xung đồng bộ có độ rộng thay đổi được (PWM : Pulse Width Modulation). Các ngõ ra PWM ứng dụng trong điều khiển động cơ và các ứng dụng chuyển đổi công suất . Module PWM này hỗ trợ điều khiển các ứng dụng sau :

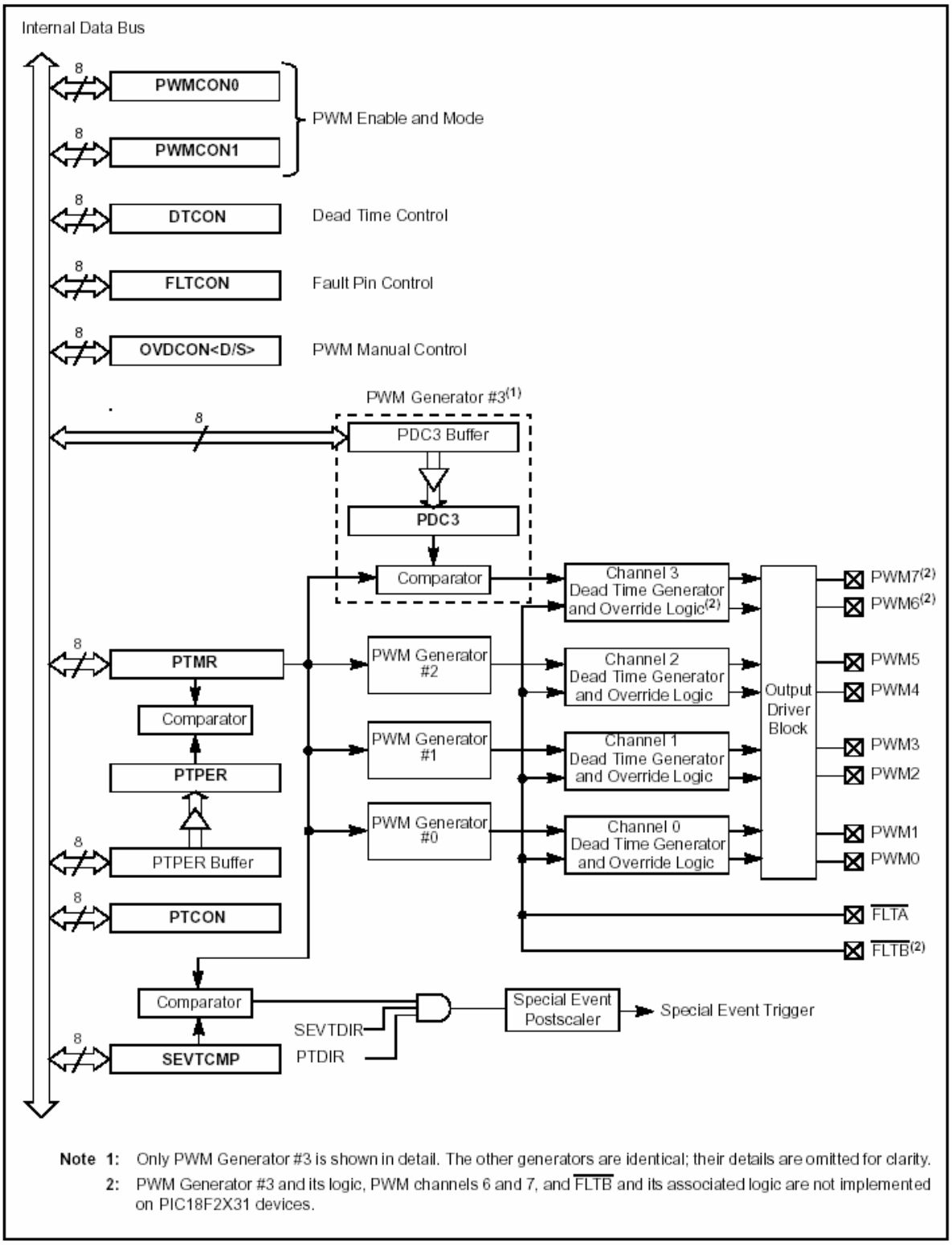
+ Động cơ KĐB 1 pha và 3 pha
+ Swithched Reluctance Motor
+ Động cơ DC không chổi than
+ UPS (Uninterruptible Power Suppliers)
+ Mutiple DC Brush motor

1.3.1.1. Các thông số cơ bản của module PWM

+ Có 8 ngõ I/O PWM với 4 duty cycle khác nhau
+ Độ phân giải 14 bit dựa trên PWM periode
+ Thời gian dead time có thể lập trình (ứng dụng trong trường PWM đối nghịch => chống trùng dẫn)
+ Ngắt hỗ trợ update không đối(asymmertrical update) xứng trong chế độ canh giữa (center aligned mode)

1.3.1.2. Sơ đồ khối của module PWM

FIGURE 17-1: POWER CONTROL PWM MODULE BLOCK DIAGRAM



Note 1: Only PWM Generator #3 is shown in detail. The other generators are identical; their details are omitted for clarity.

2: PWM Generator #3 and its logic, PWM channels 6 and 7, and $\overline{FLTB}$ and its associated logic are not implemented on PIC18F2X31 devices.

Hình 1.3.1.2.1: Sơ đồ khối của module PWM

FIGURE 17-2: PWM MODULE BLOCK DIAGRAM, ONE OUTPUT PAIR, COMPLEMENTARY MODE

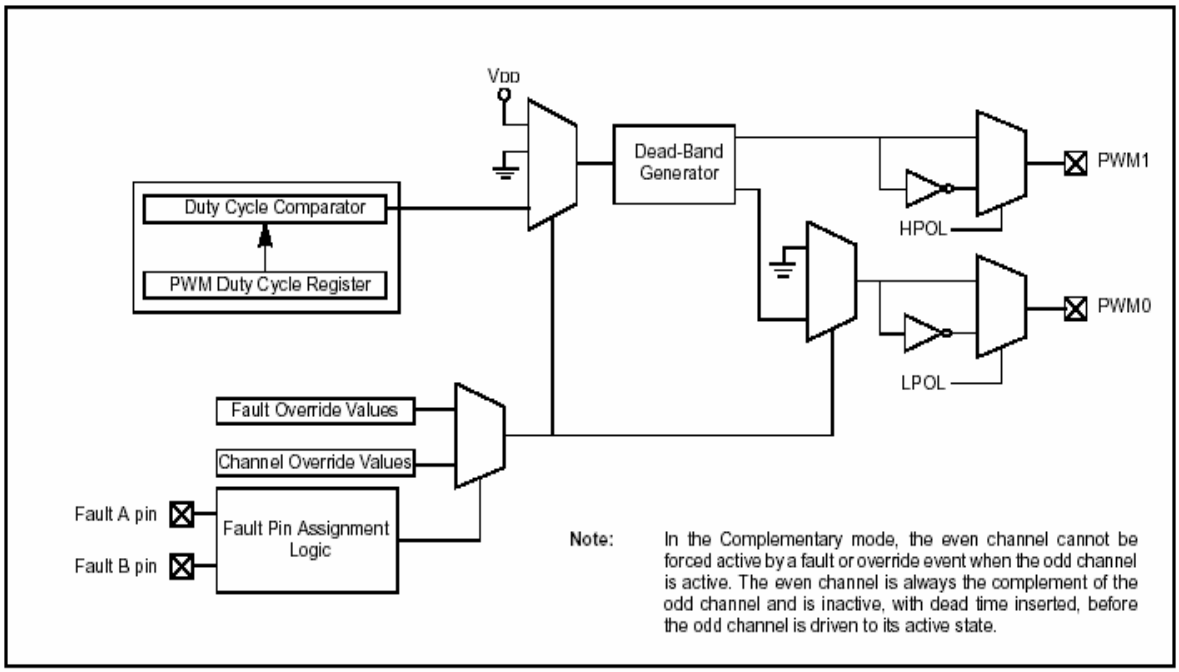
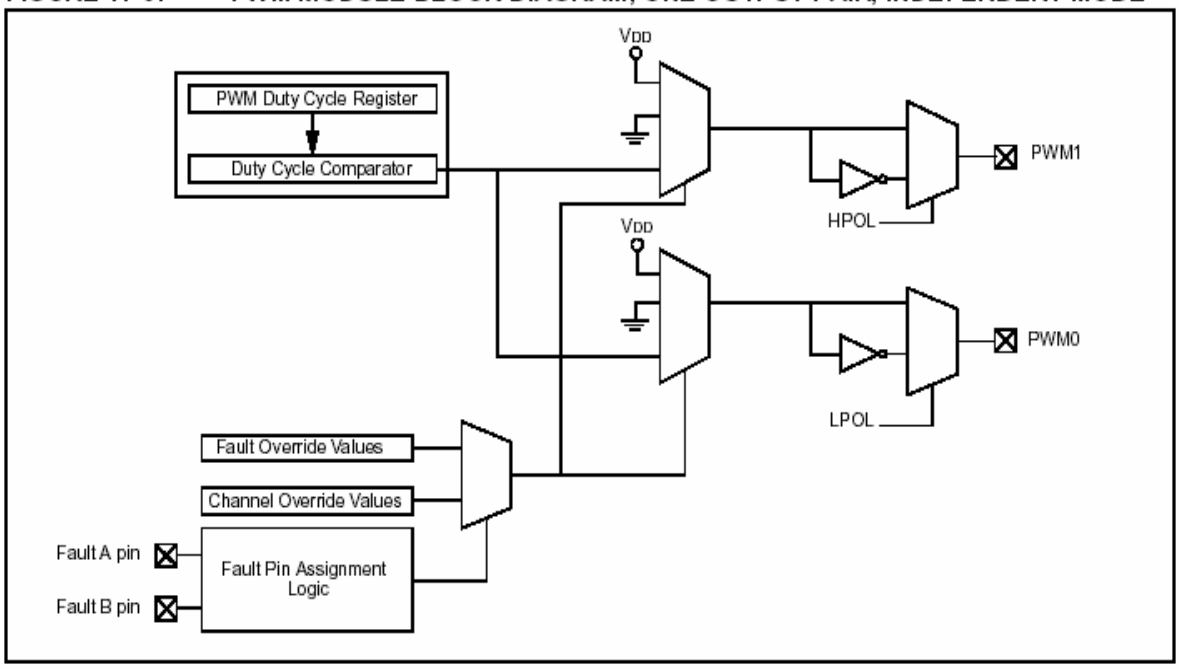


FIGURE 17-3: PWM MODULE BLOCK DIAGRAM, ONE OUTPUT PAIR, INDEPENDENT MODE



Hình 1.3.1.2.2: Sơ đồ khối của module PWM

Trong module PWM có 4 bộ tạo duty cycle riêng biệt, chúng được đánh số từ 0 -> 3. Module này có 8 ngõ ra, được đánh số từ 0->7. Trong chế độ đối nghịch các pin chẵn – pin lẻ là 1 cặp. VD: PWM0 sẽ đối nghịch với PWM1; PWM2 sẽ đối nghịch với PWM3; ….

Bộ tạo dead time sẽ chèn 1 khoản " off " giữa lúc xung PWM của pin này đang cạnh xuống và xung PWM của chân đối nghịch đang ở cạnh lên (trong 1 cặp chân đối nghịch). Điều này ngăn chặn trùng dẫn => các khóa công suất được bảo vệ.

1.3.1.3. Các thanh ghi điều khiển:

Hoạt động của module PWM được điều khiển thông qua 22 thanh ghi khác nhau. 8 trong số đó được dùng để điều chỉnh các thông số của module:

+ PWM timer control register 0 (**PTCON0**)
+ PWM timer control register 1 (**PTCON1**)
+ PWM control register 0 (**PWCON0**)
+ PWM control register 1 (**PWCON1**)
+ Dead time control register (**DTCON**)
+ Output overide register(**OVDCOND**)
+ Output state register (**OVDCONS**)
+ Fault configrration register (**FLTCONFIG**)

7 cặp (14 thanh ghi) còn lại : hiệu chỉnh thông số đặc biệt:

+ PWM time base registers (**PTMRH** and **PTMRL**)
+ PWM periode registers (**PTPERH** and **PTPERL**)
+ PWM special event compare register (**SEVTCMPH** and **SEVTCMPL**)
+ PWM duty cycle #0 register (**PDC0H** and **PDC0L**)
+ PWM duty cycle #1 register (**PDC1H** and **PDC1L**)
+ PWM duty cycle #2 register (**PDC2H** and **PDC2L**)
+ PWM duty cycle #3 register (**PDC3H** and **PDC3**L) Những cặp thanh ghi trên đều double buffers

1.3.1.4. Các module chức năng:

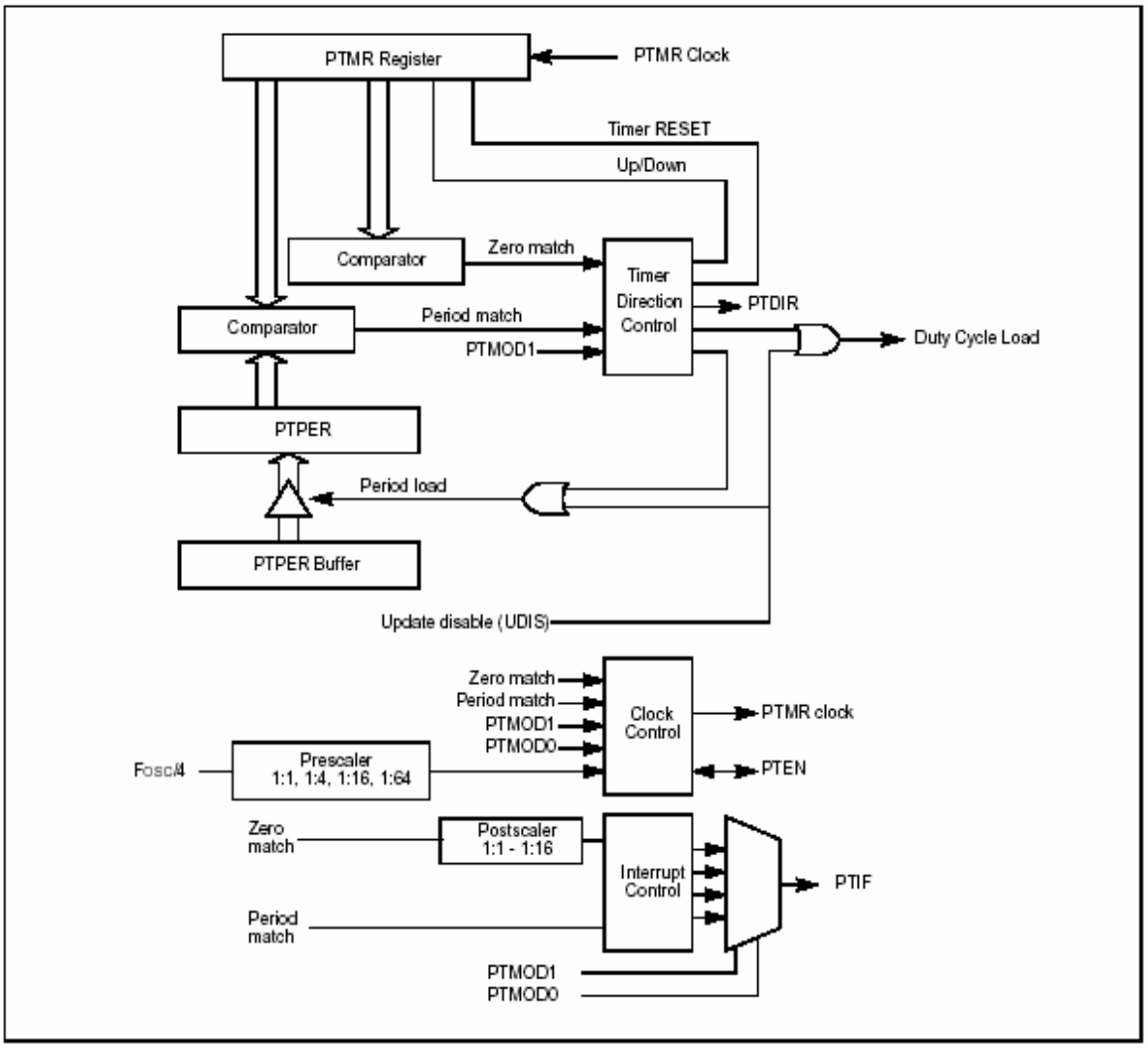
PWM module hỗ trợ nhiều chế độ hoạt động phù hợp cho yêu cầu điều khiển động cơ. PWM module được tổng hợp từ các khối chức năng sau:

+ PWM Time Base
+ PWM Time Base Interrrupts
+ PWM Period
+ PWM Duty Cycle
+ Dead Time Generators
+ PWM Output Overrides
+ PWM Fault Inputs
+ PWM Special Event Trigger

1.3.1.5. PWM Time Base:

PWM time base được cung cấp 12 bit timer với chức năng prescaler and postcaler. Sơ đồ khối đơn giản của PWM time base được trình bày trong hình 17-4.PWM time base được hiệu chỉnh thông qua 2 thanh ghi PTCON0 và PTCON1.Time base được enabled hay disabled bởi set hay clear bit PTEN trong thanh ghi PTCON1 . Chú ý,
cặp thanh ghi PTMR (PTMRH:PTMRL) sẽ không bị clear khi bit PTEN bị clear trong phần mềm !!!

FIGURE 17-4: PWM TIME BASE BLOCK DIAGRAM



Hình 1.3.1.5: Sơ đồ khối PWM Time Base:

PWM time base có 4 chế độ hoạt động như sau
+ Free running mode => edge aligned PWM
+ Single shot mode => center aligned PWM

+ Continous Up/Down count mode => support electronically commtated motors
+ Continous Up/Down count mode with interrupts for double updates

4 chế độ trên được lựa chọn thông qua bit PTMOD1:PTMOD0 trong thanh ghi PTCON0.

REGISTER 17-1: PTCON0: PWM TIMER CONTROL REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0
bit 7							bit 0

bit 7-4 **PTOPS3:PTOPS0:** PWM Time Base Output Postscale Select bits
0000 =1:1 Postscale
0001 =1:2 Postscale
.
.
.
1111 =1:16 Postscale

bit 3-2 **PTCKPS1:PTCKPS0:** PWM Time Base Input Clock Prescale Select bits
00 =PWM time base input clock is Fosc/4 (1:1 prescale)
01 =PWM time base input clock is Fosc/16 (1:4 prescale)
10 =PWM time base input clock is Fosc/64 (1:16 prescale)
11 =PWM time base input clock is Fosc/256 (1:64 prescale)

bit 1-0 **PTMOD1:PTMOD0:** PWM Time Base Mode Select bits
11 =PWM time base operates in a Continuous Up/Down mode with interrupts for double PWM updates.
10 =PWM time base operates in a Continuous Up/Down Counting mode.
01 =PWM time base configured for Single-shot mode.
00 =PWM time base operates in a Free Running mode.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = bit is set	'0' = bit is cleared	x = bit is unknown

REGISTER 17-2: PTCON1: PWM TIMER CONTROL REGISTER 1

R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
PTEN	PTDIR	—	—	—	—	—	—
bit 7							bit 0

bit 7 **PTEN:** PWM Time Base Timer Enable bit
1 = PWM time base is ON
0 = PWM time base is OFF

bit 6 **PTDIR:** PWM Time Base Count Direction Status bit
1 = PWM time base counts down.
0 = PWM time base counts up.

bit 5-0 **Unimplemented:** Read as '0'.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1'= bit is set	'0' = bit is cleared	x = bit is unknown

REGISTER 17-3: PWMCON0: PWM CONTROL REGISTER 0

U-0	R/W-1(1)	R/W-1(1)	R/W-1(1)	R/W-0	R/W-0	R/W-0	R/W-0
—	PWMEN2	PWMEN1	PWMEN0	PMOD3(3)	PMOD2	PMOD1	PMOD0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'.

bit 6-4 **PWMEN2:PWMEN0:** PWM Module Enable bits(1)
111 =All odd PWM I/O pins enabled for PWM output(2).
110 =PWM1, PWM3 pins enabled for PWM output.
101 =All PWM I/O pins enabled for PWM output(2) .
100 =PWM0, PWM1, PWM2, PWM3, PWM4 and PWM5 pins enabled for PWM output.
011 =PWM0, PWM1, PWM2 and PWM3 I/O pins enabled for PWM output.
010 =PWM0 and PWM1 pins enabled for PWM output.
001 =PWM1 pin is enabled for PWM output.
000 =PWM module disabled. All PWM I/O pins are general purpose I/O.

bit 3-0 **PMOD3:PMOD0:** PWM Output Pair Mode bits
For PMOD0:
1 = PWM I/O pin pair (PWM0, PWM1) is in the Independent mode.
0 = PWM I/O pin pair (PWM0, PWM1) is in the Complementary mode.
For PMOD1:
1 = PWM I/O pin pair (PWM2, PWM3) is in the Independent mode.
0 = PWM I/O pin pair (PWM2, PWM3) is in the Complementary mode.
For PMOD2:
1 = PWM I/O pin pair (PWM4, PWM5) is in the Independent mode.
0 = PWM I/O pin pair (PWM4, PWM5) is in the Complementary mode.
For PMOD3(3):
1 = PWM I/O pin pair (PWM6, PWM7) is in the Independent mode.
0 = PWM I/O pin pair (PWM6, PWM7) is in the Complementary mode.

Note 1: Reset condition of PWMEN bits depends on PWMPIN device configuration bit.
2: When PWMEN2:PWMEN0 = 101, PWM[5:0] outputs are enabled for PIC18F2X31 devices; PWM[7:0] outputs are enabled for PIC18F4X31devices. When PWMEN2:PWMEN0 = 111, PWM outputs 1, 3 and 5 are enabled in PIC18F2X31devices; PWM outputs 1, 3, 5 and 7 are enabled in PIC18F4X31 devices.
3: Unimplemented in PIC18F2X31 devices; maintain these bits clear.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = bit is set	'0' = bit is cleared	x = bit is unknown

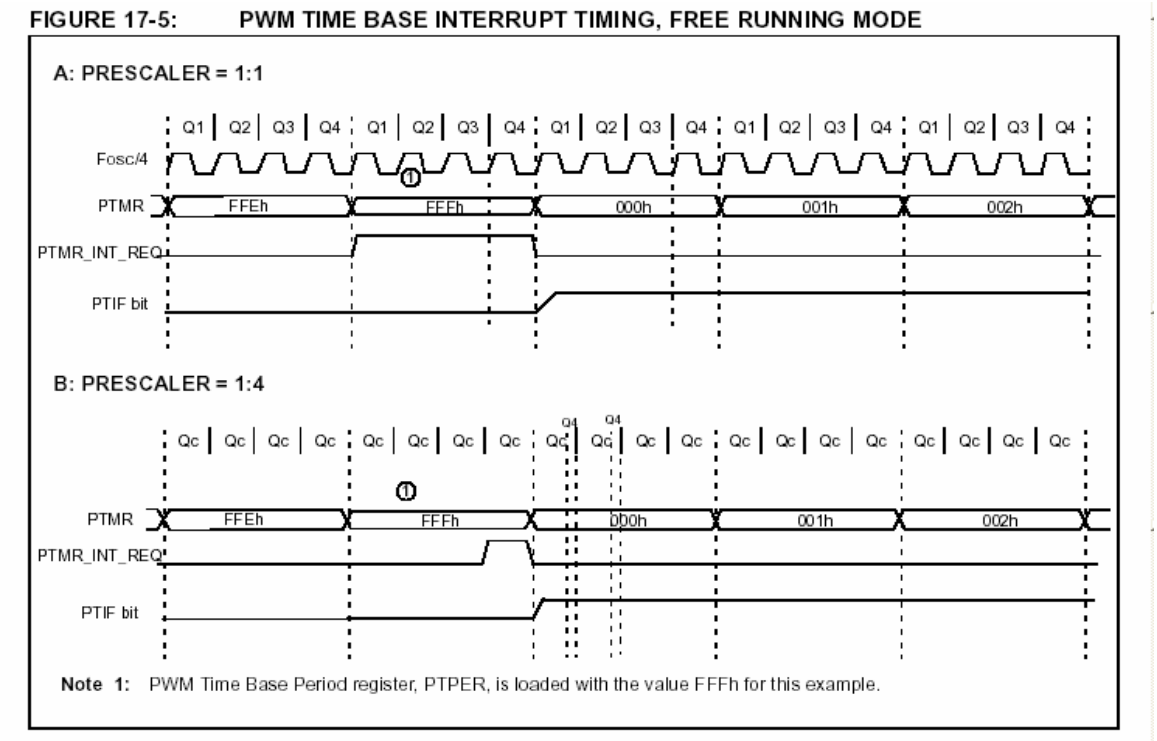
1.3.1.6. PWM Time Base Interrrupts:

PWM timer tạo ra interrupts dựa trên chế độ hoạt động được lựa chọn bởi những bit PTMOD<1:0> và những bit postscaler<3:0>

Interrupts trong chế độ FREE RUNNING:

PWM time base ở chế độ time base (PTMOD<1:0>=00), sự kiện interrupts xảy ra khi giá trị trong thanh ghi PTPER bằng giá trị của thanh ghi PTMR. Giá trị của thanh ghi PTMR sẽ được được đưa về zero ngay xung clock sau đó.

Sử dụng postscaler lớn hơn 1:1 sẽ giảm tần số của các sự kiện interrupts .

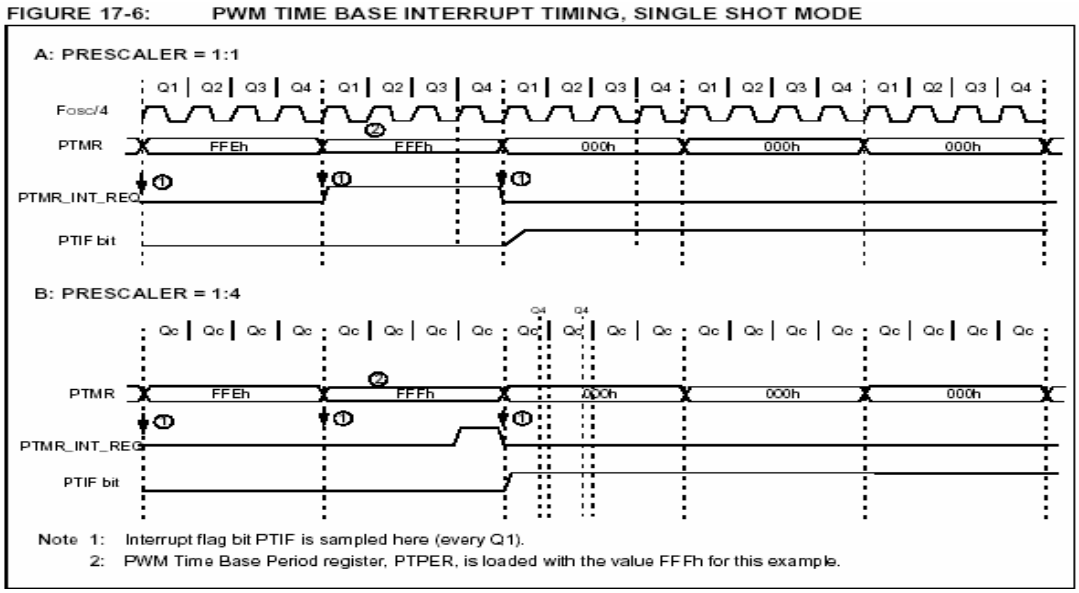


Hình 1.3.1.6.1. Ngắt trong chế độ FREE RUNNING

Interrupts trong chế độ SINGLE SHOT:

Khi bit PTMOD<1:0>=01 =>PWM time base ở chế độ single shot. Sự kiện interrupts xảy ra khi giá trị trong thanh ghi PTPER bằng giá trị của thanh ghi PTMR. Giá trị của thanh ghi PTMR sẽ được đưa về zero ngay xung clock sau đó.

Những bit postscaler không có tác dụng gì khi timer ở chế độ này.



Hình 1.3.1.6.2. Ngắt trong chế độ SINGLE SHOT:

Interrupts trong chế độ DOUBLE UPDATE:

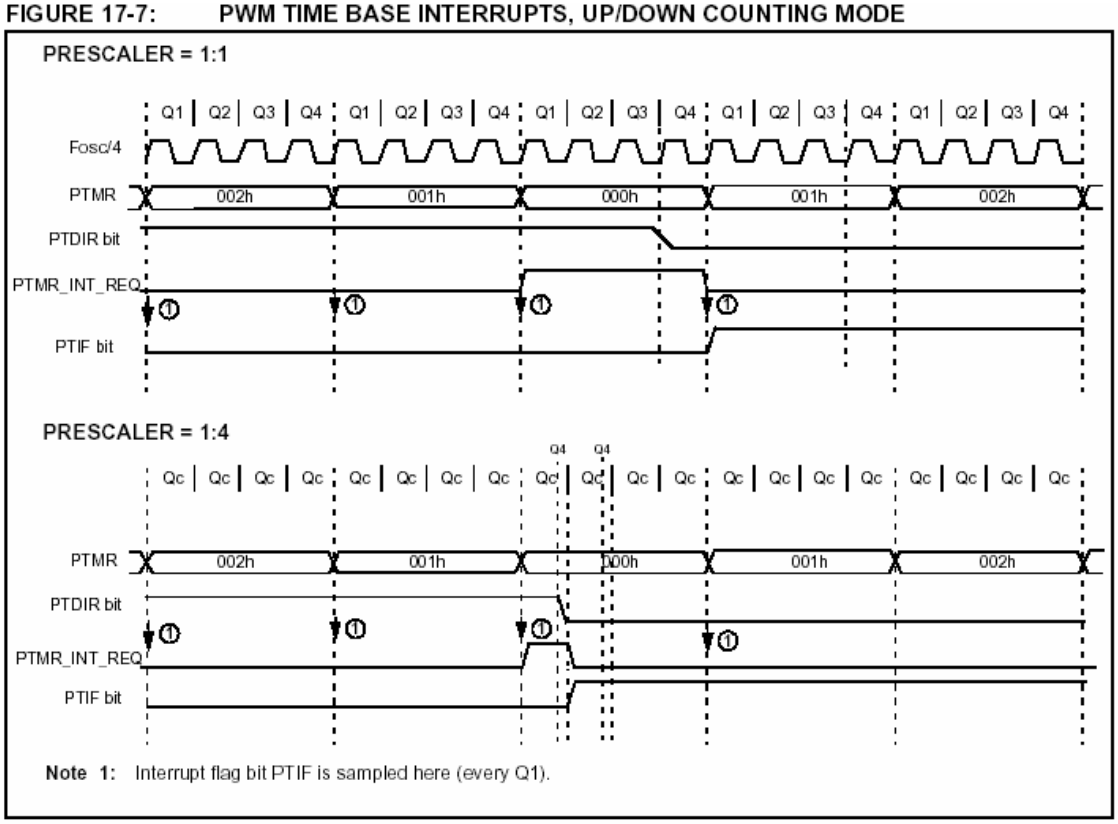
Chế độ này chỉ có trong Up/Down Counting mode (PTMOD<1:0>=11). Sự kiện interrupts xảy ra mỗi khi giá trị thanh ghi PTMR tương đương với zero hay khi giá trị thanh ghi PTMR trùng với giá trị thanh ghi PTPER.

Chế độ double update cung cấp cho người dùng thêm 2 chức năng trong chế độ center-align mode:

+ Bandwidth có độ lớn gấp đôi vì PWM duty cycle được update 2 lần trong mỗi chu kỳ (periode)

+ Có thể tạo ra được dạng sóng PWM center-align không đối xứng, điều này rất hữu dụng trong việc hạn chế tối đa sự méo dạng của dạng sóng ngõ ra trong 1 số ứng dụng điều khiển động cơ .

FIGURE 17-7: PWM TIME BASE INTERRUPTS, UP/DOWN COUNTING MODE



Hình 1.3.1.6.3. Ngắt trong chế độ DOUBLE UPDATE:

1.3.1.7. PWM Period :

PWM periode được định nghĩa bởi cặp thanh ghi PTPER (PTPERH và PTPERL). PWM periode có độ phân giải 12 bit. PTPER là cặp thanh ghi double buffered sử dụng để set chế độ đếm của PWM time base.

Nội dung của PTPER buffer được nạp vào thanh ghi PTPER ở các thờ điểm sau:

+ ***Free running mode và Single shot modes***: thanh ghi PTMR được đưa về zero sau khi trùng giá trị với thanh ghi PTPER

+ ***Up/down counting mode:*** khi PTMR bằng zero. Giá trị được lưu trong PTPER buffer tự động nạp vào thanh ghi PTPER khi PWM time base được disabled (PTEN=0)

FIGURE 17-9: PWM PERIOD BUFFER UPDATES IN FREE RUNNING COUNT MODE

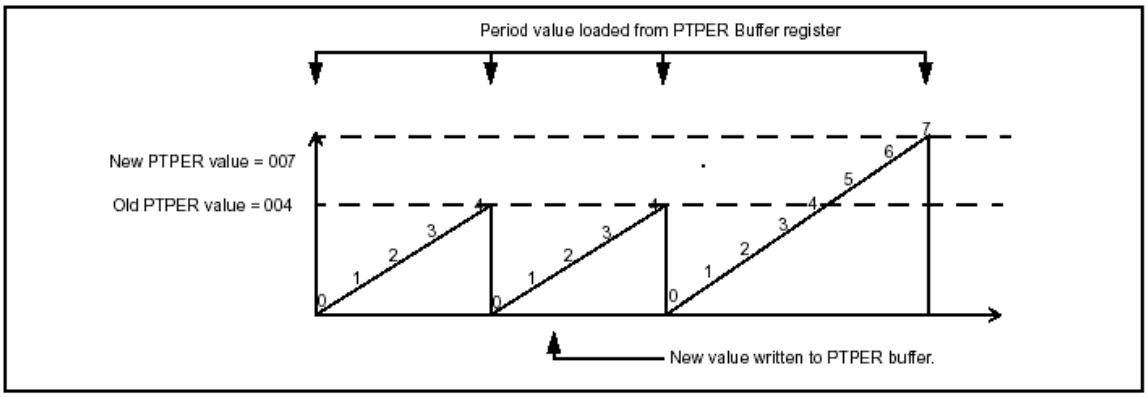
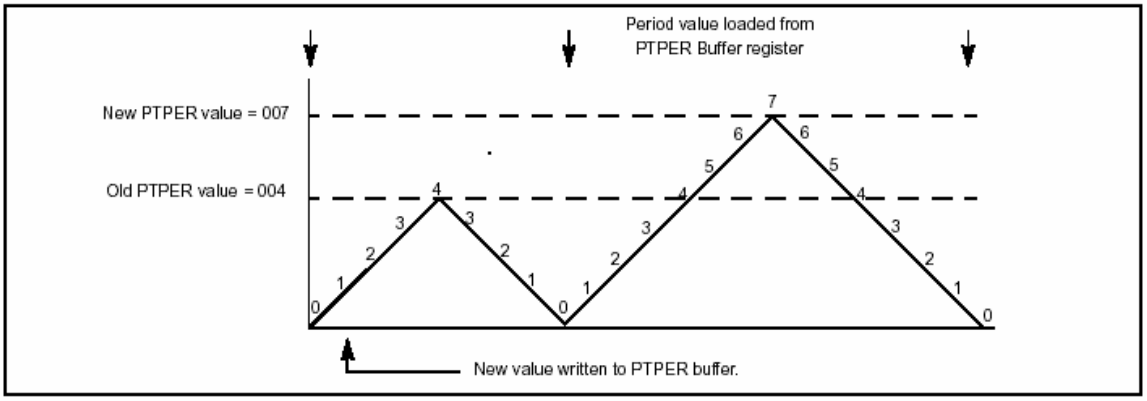


FIGURE 17-10: PWM PERIOD BUFFER UPDATES IN UP/DOWN COUNTING MODES



Hình 1.3.1.7.1: Up/down counting mode:

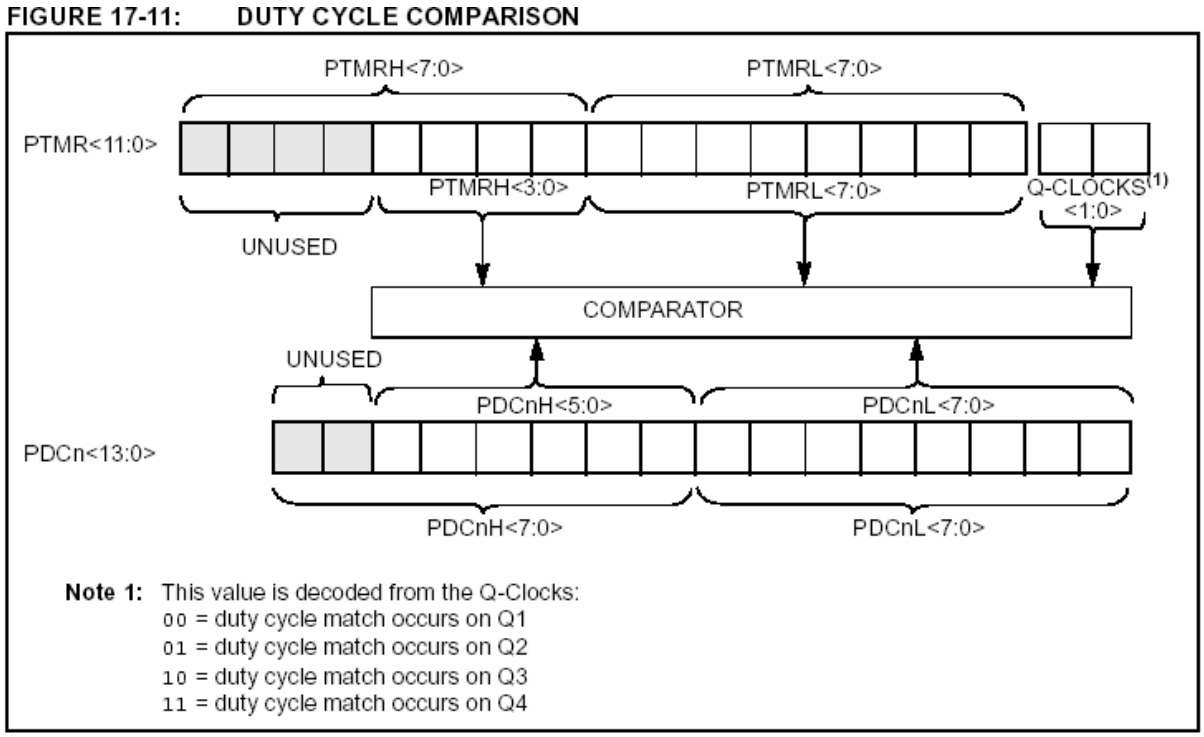
PWM duty cycle:

PWM duty cycle được xác định bởi các thanh ghi PDCx (PDCxH và PDCxL). Có tổng cộng 4 cặp thanh ghi PWM duty cycle cho 4 cặp xung PWM.

+ PDC0 (PDC0L và PDC0H)
+ PDC1 (PDC1L và PDC1H)
+ PDC2 (PDC2L và PDC2H)
+ PDC3 (PDC3L và PDC3H)

Giá trị trong mỗi thanh ghi xác định khoản thời gian mà ngõ ra PWM đó tích cực.

Trong chế độ Edge-aligned, PWM periode bắt đầu tại Q1 và kết thúc khi thanh ghi duty cycle trùng với giá trị PTMR.

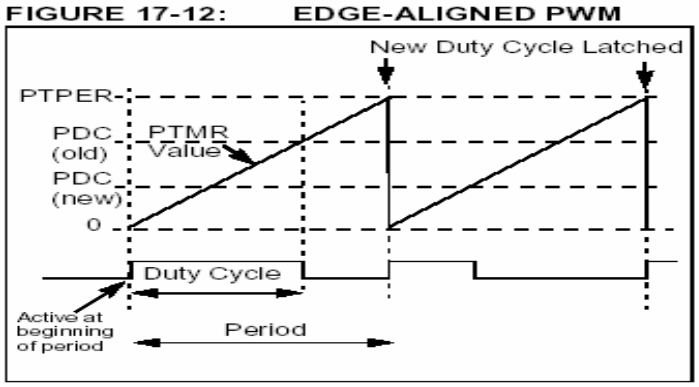


Hình 1.3.1.7.2: PWM duty cycle

Duty cycle register buffer:

4 thanh ghi PWM duty cycle đều được double buffered. Mỗi duty cycle block, đều có thanh ghi duty clycle buffer mà có thể truy xuất bởi người dùng. Thang ghi duty cycle buffer thứ hai sẽ giữ giá trị so sánh với PWM periode hiện tại.

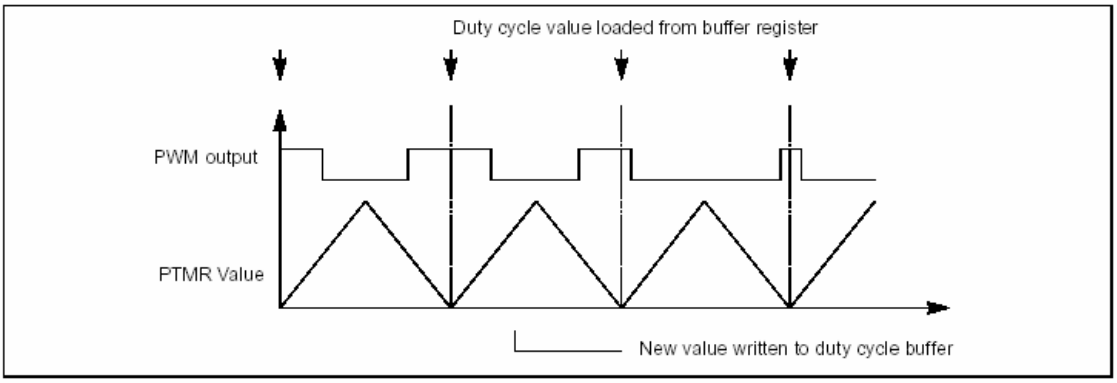
Trong chế độ edge-aligned PWM output, giá trị duty cycle mới sẽ được update mỗi khi giá trị thai thanh ghi PTMR và PTPER trùng nhau. Sau đó PTMR sẽ được reset như trong hình 17-12. Nội dung của duty cycle buffer sẽ tự động cập nhật vào thanh ghi duty cycle khi PWM time base bị disable (PTEN=0)



Hình 1.3.1.7.3: chế độ edge-aligned PWM

Khi PWM time base ở chế độ Up/Down couting, giá trị duty cycle mới sẽ được update khi giá trị thanh ghi PTMR bằng zero và PWM time base bắt đầu đếm lên. Nội dung của duty cycle buffer sẽ tự động cập nhật vào thanh ghi duty cycle khi PWM time base bị disable (PTEN=0). Hình 17-13 trình bày giản đồ thời gian khi duty cycle được update ở chế độ Up/Down counting . Trong chế độ này PWM periode phải được sẵn sàng để nạp và tính toán trước PWM duty cycle mới trước khi các thay đổi có hiệu lực.

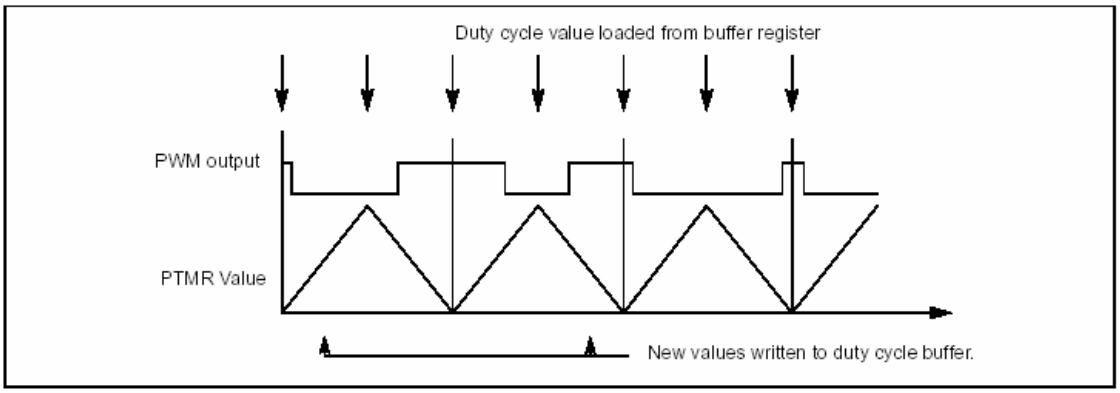
FIGURE 17-13: DUTY CYCLE UPDATE TIMES IN UP/DOWN COUNTING MODE



Hình 1.3.1.7.4:Dạng xung PWM time base ở chế độ Up/Down couting

Khi PWM time base ở chế độ Up/Down couting vơi double update mode, giá trị duty cycle mới sẽ được update khi giá trị thanh ghi PTMR bằng zero và khi giá trị hai thanh ghi PTMR và PTPER trùng nhau. Nội dung của duty cycle buffer sẽ tự động được nạp vào thanh ghi duty cycle khi một trong hai điều kiện trên xảy ra.

FIGURE 17-14: DUTY CYCLE UPDATE TIMES IN UP/DOWN COUNTING MODE WITH DOUBLE UPDATES



Hình 1.3.1.7.5: PWM time base ở chế độ Up/Down couting vơi double update mode

Bộ tạo thời gian dead time:

Trong bộ biến tần , khi các xung PWM ở chế độ đối nghịch để điều khiển các khóa công suất phía cao; phía thấp trong cùng 1 nhánh, phải chèn 1 khoản thời gian dead time. Khoảng thời gian dead time đó làm cho ngõ ra PWM đối nghịch đều ở trạng

thái không tác động trong 1 khoảng thời gian ngắn=> tránh trùng dẫn khi khóa này đang ON , khóa kia đang OFF

Mỗi cặp xung PWM đối nghịch đều có một counter 6 bit đếm xuống, để chèn khoảng dead time vào xung PWM. Mỗi bộ tạo dead time có bộ phát hiện cạnh lên và cạnh xuống được kết nối với bộ so sánh duty cycle. Dead time được nạp vào timer khi phát hiện PWM ở cạnh lên hay cạnh xuống. Tùy vào xung PWM đang ở cạnh lên hay cạnh xuống, mà 1 khoảng thời gian chuyển tiếp được làm trễ cho đến khi timer đếm về zero.

FIGURE 17-17: DEAD TIME CONTROL UNIT BLOCK DIAGRAM FOR ONE PWM OUTPUT PAIR

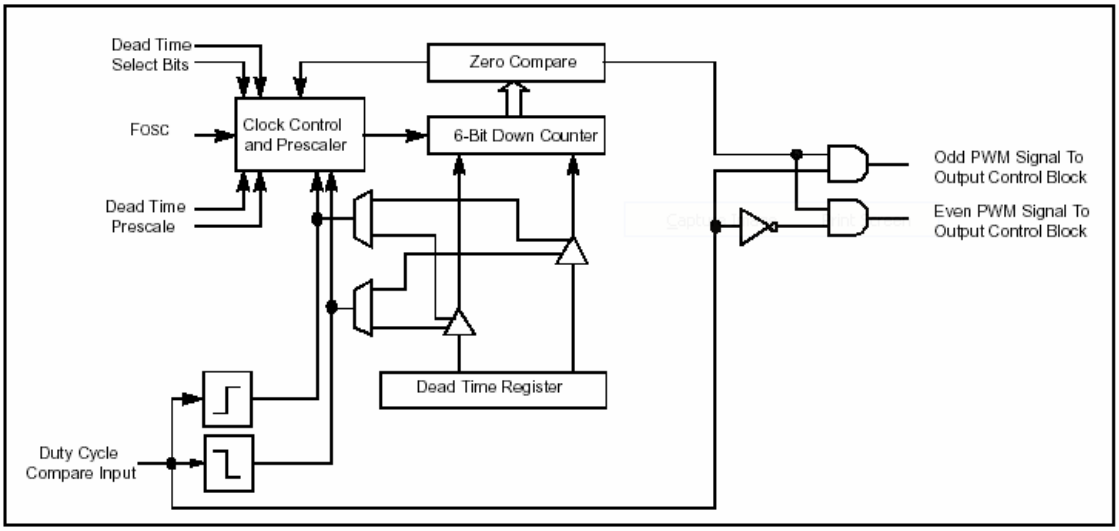
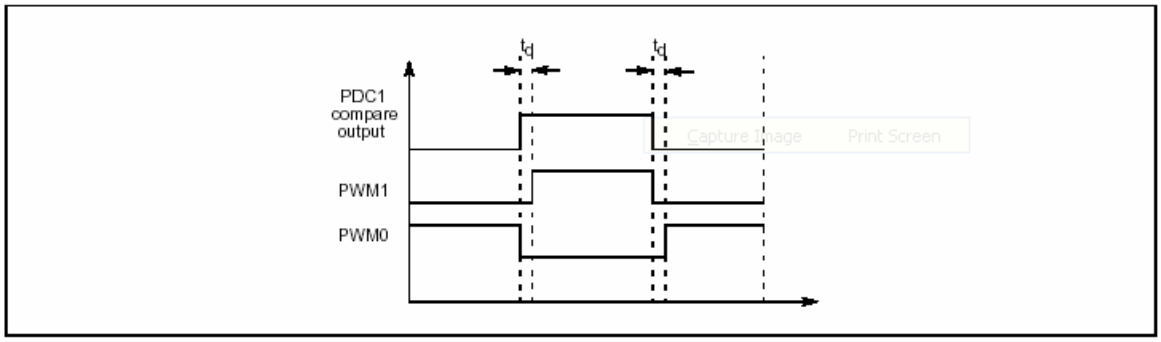


FIGURE 17-18: DEAD TIME INSERTION FOR COMPLEMENTARY PWM



Hình 1.3.1.7.6: Bộ tạo thời gian dead time

Thanh ghi DTCON:

REGISTER 17-5: DTCON – DEAD TIME CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTPS1	DTPS0	DT5	DT4	DT3	DT2	DT1	DT0
bit 7							bit 0

bit 7-6 **DTPS1:DTPS0:** Dead Time Unit A Prescale Select bits
11 = Clock source for Dead Time Unit is Fosc/16.
10 = Clock source for Dead Time Unit is Fosc/8.
01 = Clock source for Dead Time Unit is Fosc/4.
00 = Clock source for Dead Time Unit is Fosc/2.

bit 5-0 **DT5:DT0:** Unsigned 6-bit dead time value bits for Dead Time Unit.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = bit is set	'0' = bit is cleared	x = bit is unknown

Bảng tóm tắt các thanh ghi có liên quan của POWER CONTROL PWM MODULE :

TABLE 17-6: REGISTERS ASSOCIATED WITH THE POWER CONTROL PWM MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
IPR3	—	—	—	PTIP	IC3DRIP	IC2QEIP	IC1IP	TMR5IP	---1 1111	---1 1111
PIE3	—	—	—	PTIE	IC3DRIE	IC2QEIE	IC1IE	TMR5IE	---0 0000	---0 0000
PIR3	—	—	—	PTIF	IC3DRIF	IC2QEIF	IC1IF	TMR5IF	---0 0000	---0 0000
PTCON0	PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0	0000 0000	0000 0000
PTCON1	PTEN	PTDIR	—	—	—	—	—	—	00-- ----	00-- ----
PTMRL(1)	PWM Time Base (lower 8 bits)								0000 0000	0000 0000
PTMRH(1)	—	—	—	—	PWM Time Base (upper 4 bits)				---- 0000	---- 0000
PTPERL(1)	PWM Time Base Period (lower 8 bits)								1111 1111	1111 1111
PTPERH(1)	—	—	—	—	PWM Time Base Period (upper 4 bits)				---- 1111	---- 1111
SEVTCMPL(1)	PWM Special Event Compare (lower 8 bits)								0000 0000	0000 0000
SEVTCMPH(1)	—	—	—	—	PWM Special Event Compare (upper 4 bits)				---- 0000	---- 0000
PWMCON0	—	PWMEN2	PWMEN1	PWMEN0	PMOD3(2)	PMOD2	PMOD1	PMOD0	-101 0000	-101 0000
PWMCON1	SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC	0000 0-00	0000 0-00
DTCON	DTPS1	DTPS0	Dead Time A Value register						0000 0000	0000 0000
FLTCONFIG	BRFEN	FLTBS(2)	FLTBMOD(2)	FLTBEN(2)	FLTCON	FLTAS	FLTAMOD	FLTAEN	0000 0000	0000 0000
OVDCOND	POVD7(2)	POVD6(2)	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0	1111 1111	1111 1111
OVDCONS	POUT7(2)	POUT6(2)	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0	0000 0000	0000 0000
PDC0L(1)	PWM Duty Cycle #0L register (lower 8 bits)								--00 0000	--00 0000
PDC0H(1)	—	—	PWM Duty Cycle #0H register (upper 6 bits)						0000 0000	0000 0000
PDC1L(1)	PWM Duty Cycle #1L register (lower 8 bits)								0000 0000	0000 0000
PDC1H(1)	—	—	PWM Duty Cycle #1H register (upper 6 bits)						--00 0000	--00 0000
PDC2L(1)	PWM Duty Cycle #2L register (Lower 8 bits)								0000 0000	0000 0000
PDC2H(1)	—	—	PWM Duty Cycle #2H register (Upper 6 bits)						--00 0000	--00 0000
PDC3L(1,2)	PWM Duty Cycle #3L register (Lower 8 bits)								0000 0000	0000 0000
PDC3H(1,2)	—	—	PWM Duty Cycle #3H register (Upper 6 bits)						--00 0000	--00 0000

Legend: - = Unimplemented, u = Unchanged. Shaded cells are not used with the power control PWM.
Note 1: Double-buffered register pairs. Refer to text for explanation of how these registers are read and written to.
2: Unimplemented in PIC18F2X31 devices; maintain these bits clear. Reset values shown are for PIC18F4X31 devices.

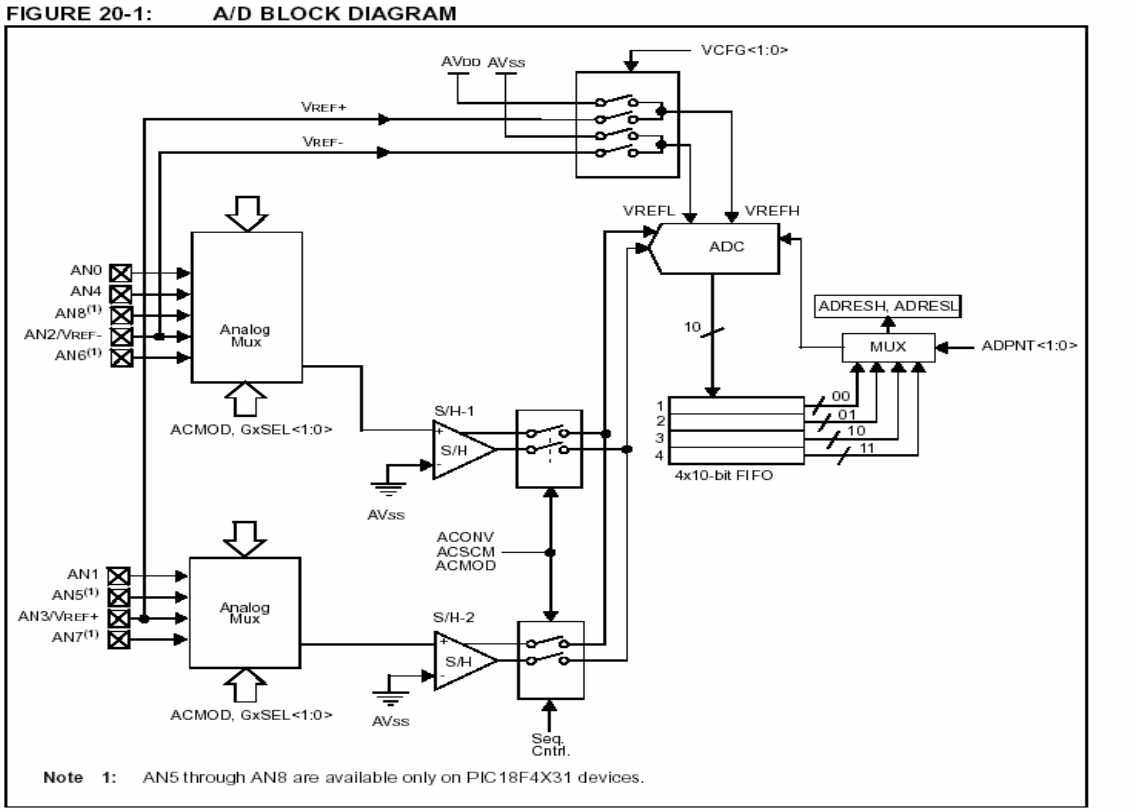
1.3.2. Analog to digital converter module (A/D):

Bộ A/D có 5 ngõ vào cho PIC 28 chân và 8 cho các PIC khác . Tín hiệu analog được lấy mẫu và giữ bởi tụ điện , sau đó đưa vào bộ chuyển đổi . Bộ này tạo ra 1 kết quả số tương ứng . Giá trị này là 1 số 10 bit.

Bộ A /D có ngõ vào so sánh áp cao và thấp ,và có thể lựa chọn thông qua kết hợp Vdd , Vss , RA2 hay RA3. Bộ A/D có điểm đặc biệt là có thể hoạt động trong khi vi điều khiển ở trạng thái SLEEP . Để làm được điều này , xung clock A/D phải được nhận từ bộ dao động RC nội của bộ A/D.

Module A/D có 9 thanh ghi :

+ A/D Result High Register (ADRESH)
+ A/D Result Low Register (ADRESL)
+ A/D Control Register0 (ADCON0)
+ A/D Control Register1 (ADCON1)
+ A/D Control Register2 (ADCON2)
+ A/D Control Register3 (ADCON3)
+ A/D chennel Select Register (ADCHS)
+ Analog I/O Select Register 0 (ANSEL0)
+ Analog I/O Select Register 1 (ANSEL1) Sơ đồ khối bộ A/D :



Hình 1.3.2.1: Cấu trúc Analog to digital converter module (A/D):

Các bước sau để làm việc với bộ A/D :

1_Thiết lập bộ A/D :

+ Thiết lập các chân analog / so sánh áp và I/O số (ADCON1) .
+ Chọn kênh ngõ vào A/D (ADCONO).
+ Chọn xung clock bộ A/D (ADCONO).
+ Kích hoạt A/D (ADCONO).

2_Thiết lập ngắt A/D nếu sử dụng

+ xoá bit ADIF.
+ Set bit ADIE.
+ set bit PEIE
+ set bit GIE

3_Chờ thời gian đáp ứng cần thiết.

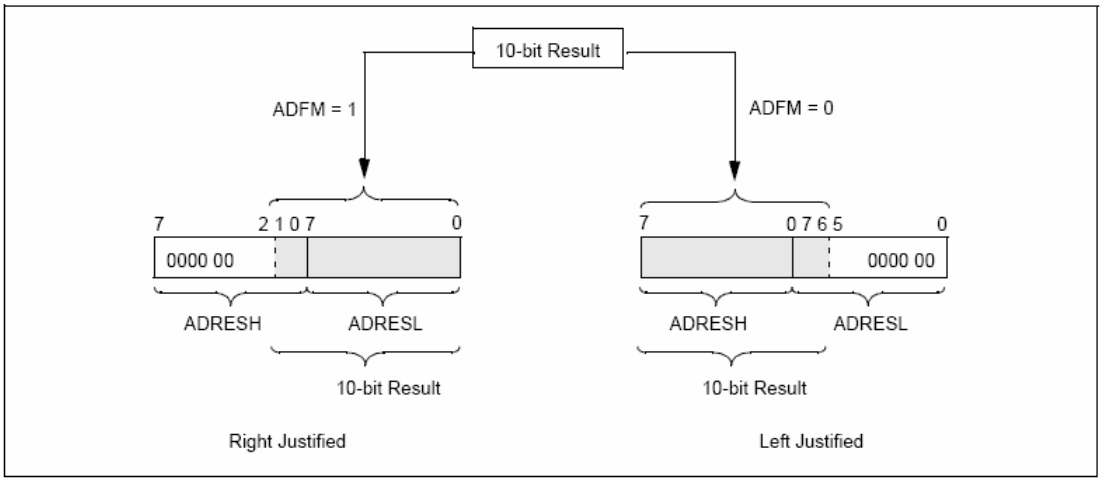
4_Bắt đầu chuyển đổi : set bit ADCONO<2>.

5_Chờ chuyển đổi A/D hoàn thành bằng cách hỏi vòng bit ADCONO<2> có bị xoá chưa hay chờ ngắt A/D

6_Đọc kết quả từ cặp thanh ghi ADRESH : ADRESL , xoá bit ADIF nếu cần .

7_Lặp lại từ bước 1 hay 2 nếu có yêu cầu. Thời gian chuyển đổi A/D mỗi bit gọi là T_{AD} .

Một khoảng chờ tối thiểu $2T_{AD}$ được yêu cầu trước khi lần đáp ứng kế tiếp bắt đầu.



Hình 1.3.2.2: Thời gian chuyển đổi A/D

Các thanh ghi ADRESH : ADRESL chứa 10 bit kết quả của chuyển đổi A/D . Khi sự chuyển đổi A/D hoàn tất , kết quả đưa vào cặp thanh ghi này , bit ADCON0 <2> bị xoá và cờ ngắt ADIF được set. Cặp thanh ghi này rộng 16 bit . Do đó nếu bit ADFM =1 :lấy 10 bit bean phải và ADFM = 0 thì lấy 10 bit bên trái , các bit còn lại bằng 0. Nếu A/D bị vô hiệu , các thanh ghi này có thể dùng như 2 thanh ghi đa mục đích.

CHƯƠNG 2:
GIAO TIẾP VI ĐIỀU KHIỂN VÀ MÁY TÍNH

2.1. Giới thiệu

Chuẩn giao tiếp nối tiếp RS232 là chuẩn phần cứng, qui định các chân nối và mức điện áp được sử dụng bởi một số các thiết bị giao tiếp nối tiếp với nhau. Ngày nay, chuẩn này trở nên thông dụng trong việc ghép nối máy tính với thiết bị ngoại vi.

Các loại vi điều khiển, vi xử lý cũng sử dụng chuẩn này trong việc truyền thông mạng vi điều khiển hoặc giữa vi điều khiển với máy tính.

Vì vậy, chuẩn RS232 thường được dùng để máy tính điều khiển và giám sát hệ thống vi điều khiển trong các ứng dụng công nghiệp.

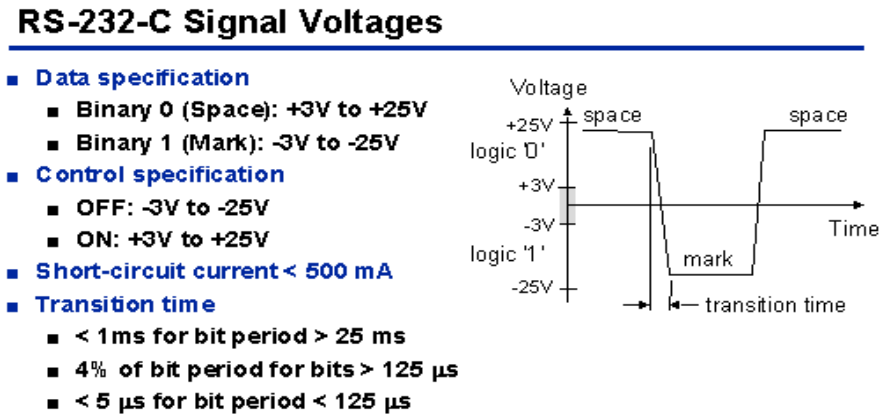
Một số ưu điểm của chuẩn:

+ Sơ đồ kết nối đơn giản
+ Tiết kiệm được dây dẫn
+ Có thể dùng để kết nối với vi điều khiển và PLC
+ Tính năng Plug-in Plug-out, nghĩa là có thể lắp đặt thiết bị truyền khi máy tính đang hoạt động.

2.2. Chuẩn RS232

2.2.1. Chuẩn điện áp:

+ Bit có 2 logic 0 và 1.
+ Theo chuẩn RS232.
+ Logic 0 (SPACE) tương ứng với điện áp -25 đến -3V
+ Logic 1 (Mark) tương ứng với điện áp 3 đến 25V
+ Khoảng điện áp từ -3 đến 3 là không xác định

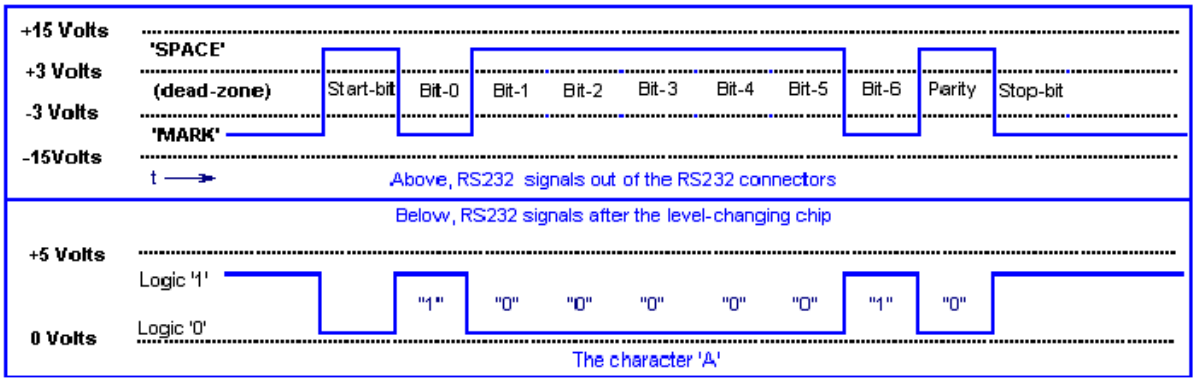


Hình 2.2.1:Chuẩn điện áp RS232

2.2.2. Chuẩn giao thức:

+ Giao thức RS232 qui định truyền 1 frame tại một thời điểm.

+ Một Frame truyền bao gồm: 1 bit start, 7 hoặc 8 bit dữ liệu ASCII của kí tự, bit Parity (kiểm tra chẵn lẻ) và kết thúc bởi 1 bit Stop.
Ví dụ sau đây là frame truyền 1 kí tự ‘A’, mã ASCII của ‘A’ là 41h, nếu truyền trong 7 bit dữ liệu sẽ là 100 0001.



Hình 2.2.2:Chuẩn giao thức RS232

2.2.3. Các qui định khác của chuẩn RS232:

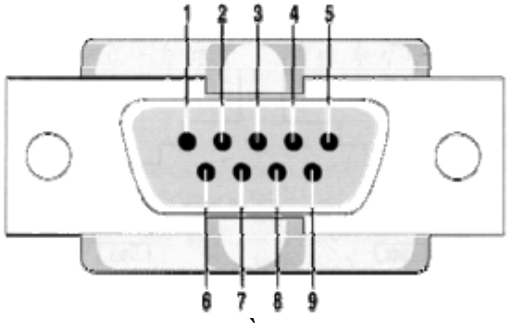
Chiều dài cable cực đại	15m
Tốc độ dữ liệu cực đại	20 Kbps
Điện áp ngõ ra cực đại	± 25V
Điện áp ngõ ra có tải	± 5V đến ± 15V
Trở kháng tải	3K đến 7K
Điện áp ngõ vào	± 15V
Độ nhạy ngõ vào	± 3V
Trở kháng ngõ vào	3K đến 7K

2.2.4. Tốc độ truyền:

Các tốc độ truyền thông dụng là: 1200bps, 4800bps, 9600bps, 19200bps

2.2.5. Sơ đồ chân:

Chuẩn RS232 qui định sử dụng 2 loại đầu nối chuẩn DB25 (25 chân) và DB9 (9 chân) Vi điều khiển thường dùng đầu nối 9 chân DB9

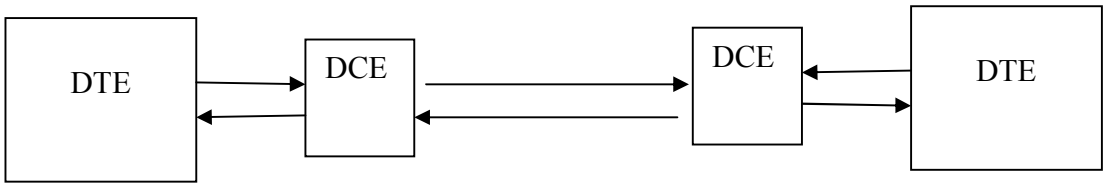


Hình 2.2.5:Sơ đồ chân chân DB9

D9	Tín hiệu	Hướng truyền	Mô tả
-	-	-	Protected ground: nối đất bảo vệ
3	TxD	DTE→DCE	Transmitted data: dữ liệu truyền
2	RxD	DCE→DTE	Received data: dữ liệu nhận
7	RTS	DTE→DCE	Request to send: DTE yêu cầu truyền dữ liệu
8	CTS	DCE→DTE	Clear to send: DCE sẵn sàng nhận dữ liệu
6	DSR	DCE→DTE	Data set ready: DCE sẵn sàng làm việc
5	GND	-	Ground: nối đất (0V)
1	DCD	DCE→DTE	Data carier detect: DCE phát hiện sóng mang
4	DTR	DTE→DCE	Data terminal ready: DTE sẵn sàng làm việc
9	RI	DCE→DTE	Ring indicator: báo chuông
-	DSRD	DCE→DTE	Data signal rate detector: dò tốc độ truyền

2.2.6. Phương thức truyền

2.2.6.1. Sơ đồ kết nối qua modem:



Hình 2.2.6.1:Sơ đồ kết nối qua modem

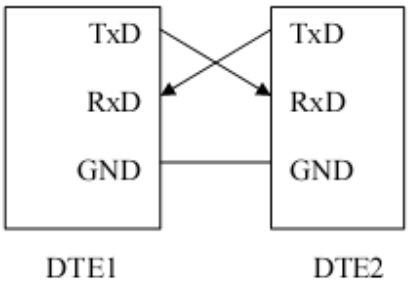
DTE- Data Terminal Equipment: Thiết bị đầu cuối dữ liệu, có thể là máy tính, vi điều khiển, PLC.

DCE- Data Communication Equipment: Thiết bị truyền dữ liệu, ví dụ Modem

2.2.6.2. Sơ đồ kết nối không qua modem:

Đây là sơ đồ hay dùng cho việc truyền dữ liệu giữa máy tính và vi điều khiển

+ Kết nối không bắt tay:

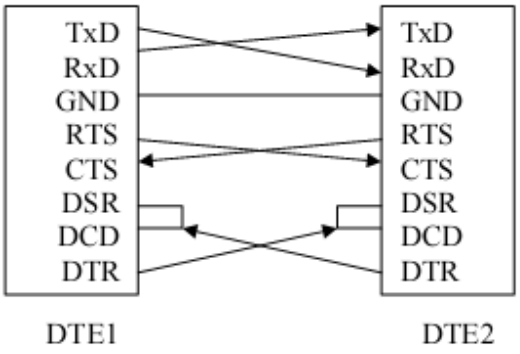


Hình 2.2.6.2.1:Sơ đồ kết nối không bắt tay

Trường hợp này phải đảm bảo cài đặt cho cả hai DTE cùng tốc độ truyền, số bit dữ liệu, dạng kiểm tra lỗi Parity và Số bit STOP.

Chú ý: Do chuẩn RS232 trên máy tính qui định mức điện áp cho logic 0 là 3 đến 25V, logic 1 là -25 đến -3V, trong khi mức điện áp tương ứng trên vi điều khiển là 0 và 5V nên cần có bộ chuyển đổi đồng bộ điện áp là MAX232 làm trung gian cho kết nối này.

+ Kết nối có bắt tay:



Hình 2.2.6.2.2:Sơ đồ kết nối bắt tay

Đây là hình thức truyền có bắt tay, có quá trình thăm dò hỏi đáp trước khi bắt đầu truyền.

Giả sử DTE1 muốn truyền dữ liệu đến DTE2. Trước hết DTE1 phải đưa chân RTS (Request to send) lên mức tích cực và bắt đầu thăm dò tín hiệu từ chân CTS (Clear to Send) của mình. Nếu DTE2 đã sẵn sàng để nhận dữ liệu, DTE2 sẽ đưa chân RTS của mình lên mức tích cực. Chân này nối với chân CTS của DTE1. Vì vậy, DTE1 sẽ nhận được tín hiệu tích cực thông qua chân CTS và biết rằng DTE2 đã sẵn sàng nhận dữ liệu. DTE1 bắt đầu truyền dữ liệu.

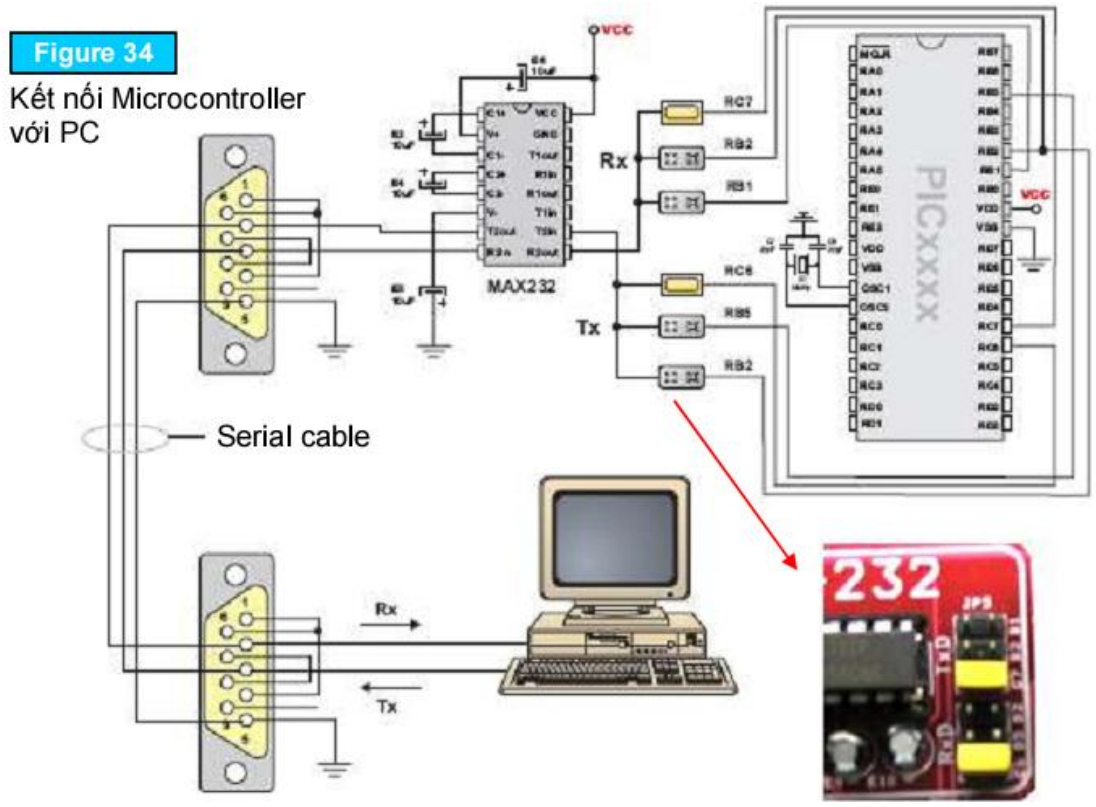
Nếu DTE1 chủ động muốn nhận dữ liệu từ DTE2. DTE1 đưa chân DTR (Data terminal Ready) lên mức tích cực để báo với với DTE2 rằng nó đã sẵn sàng và muốn nhận dữ liệu. Nếu DTE2 cũng sẵn sàng để truyền dữ liệu, nó sẽ đưa chân DSR (Data Set Ready) để báo với DTE1 là nó cũng đã chuẩn bị dữ liệu sẵn sàng để gửi đến DTE1. Và sau đó, DTE2 bắt đầu truyền dữ liệu cho DTE1.

2.3. Kết nối giao tiếp máy tính và vi điều khiển PIC qua chuẩn RS232:

2.3.1. Sơ đồ phần cứng:

Vi điều khiển PIC hỗ trợ giao tiếp nối tiếp theo chuẩn RS232.

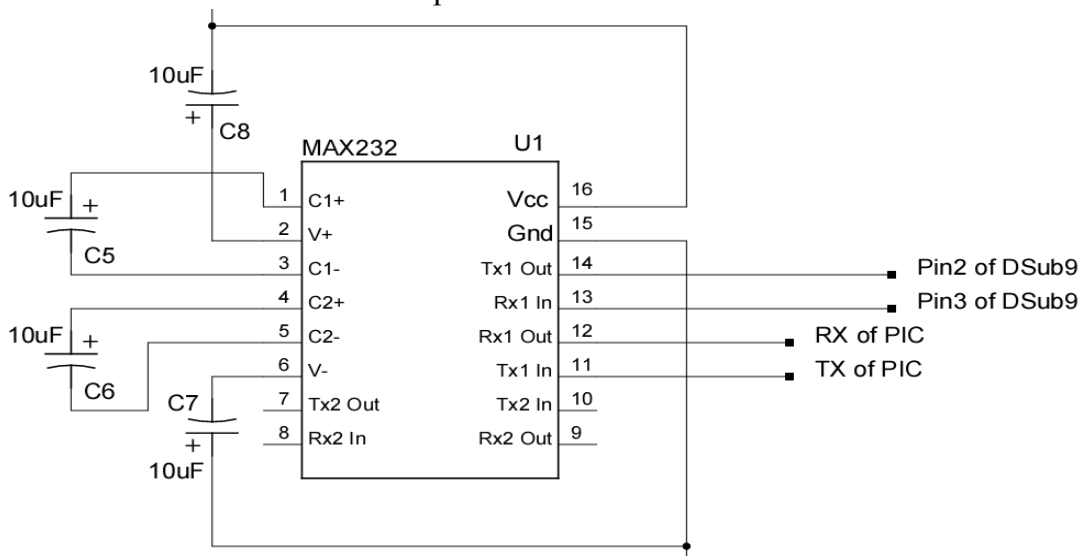
Công thức truyền theo *kiểu không bắt tay*: Gồm 2 đường truyền, đường truyền dữ liệu từ PIC đến máy tính thông qua chân RC6, đường nhận dữ liệu từ máy tính truyền đến thông qua chân RC7. Do qui định mức điện áp cho các logic 0 và 1 trên PIC là 0 và 5V, trong khi mức điện áp cho 2 logic đó ở máy tính là 3 đến 25V và -25 đến -3V, nên cần có một bộ chuyển đổi điện áp cho đồng bộ. Người ta hay sử dụng chip MAX232 để làm nhiệm vụ này.



Hình 2.3.1.1:Sơ đồ kết nối máy tinh với PIC

Ở 2 đầu của dây nối ta sử dụng 2 đầu nối DB9. Có chân RX(chân số 2) của đầu nối này nối với TX (chân số 3) của đầu nối kia và ngược lại. 2 GND của 2 đầu nối được nối với nhau.

Sơ đồ nối từ vi điều khiển PIC ra chip MAX232 như sau:



Hình 2.3.1.2:Sơ đồ khối giao tiếp

+ Chân nhận dữ liệu trên vi điều khiển PIC RC6 được nối với chân 12 trên MAX232.

+ Chân truyền dữ liệu trên vi điều khiển PIC RC7 được nối với chân 12 trên MAX232.

+ Chân 13,14 trên MAX232 được nối với chân 3 (truyền dữ liệu đến máy tinh) và chân 2 (nhận dữ liệu từ máy tính).

Giái trị các tụ dùng là theo datasheet.

2.3.2. Phần mềm trên máy tính:

Có thể sử dụng Visual Basic, Visual C, MatLab, LabView, Delphi v.v

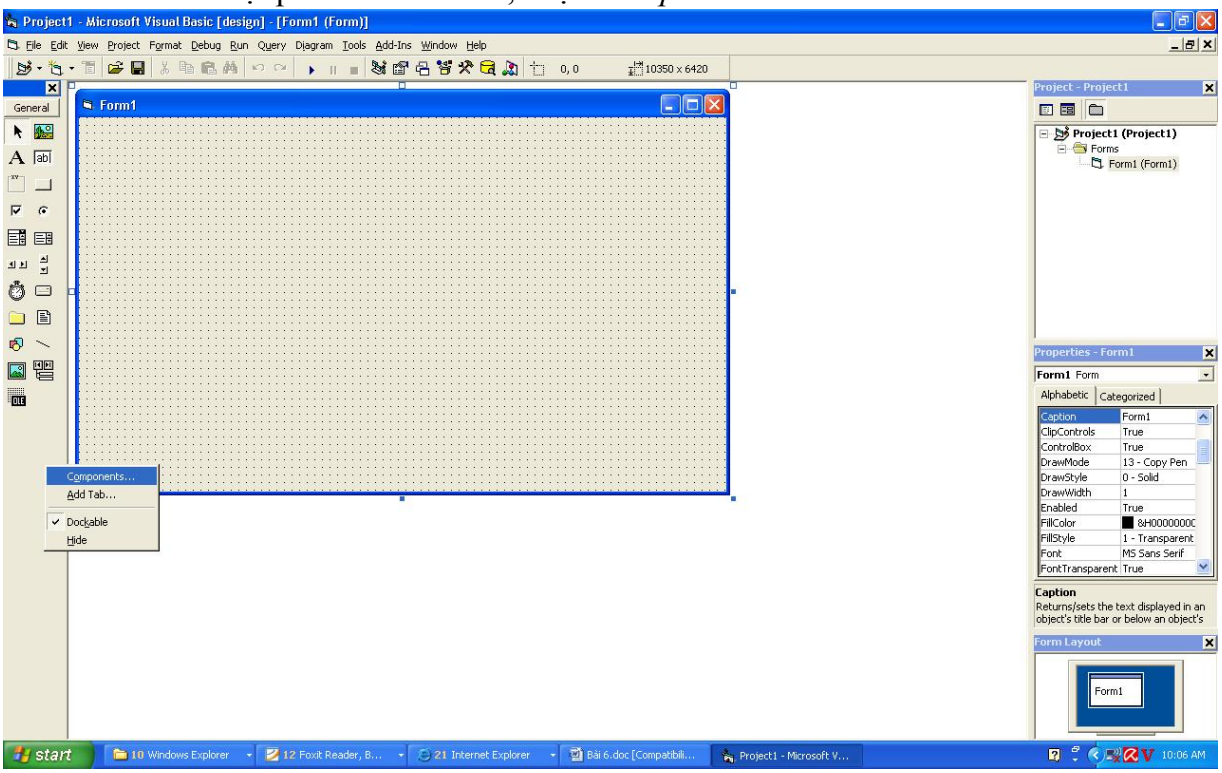
Phần mềm dùng Visual Basic:

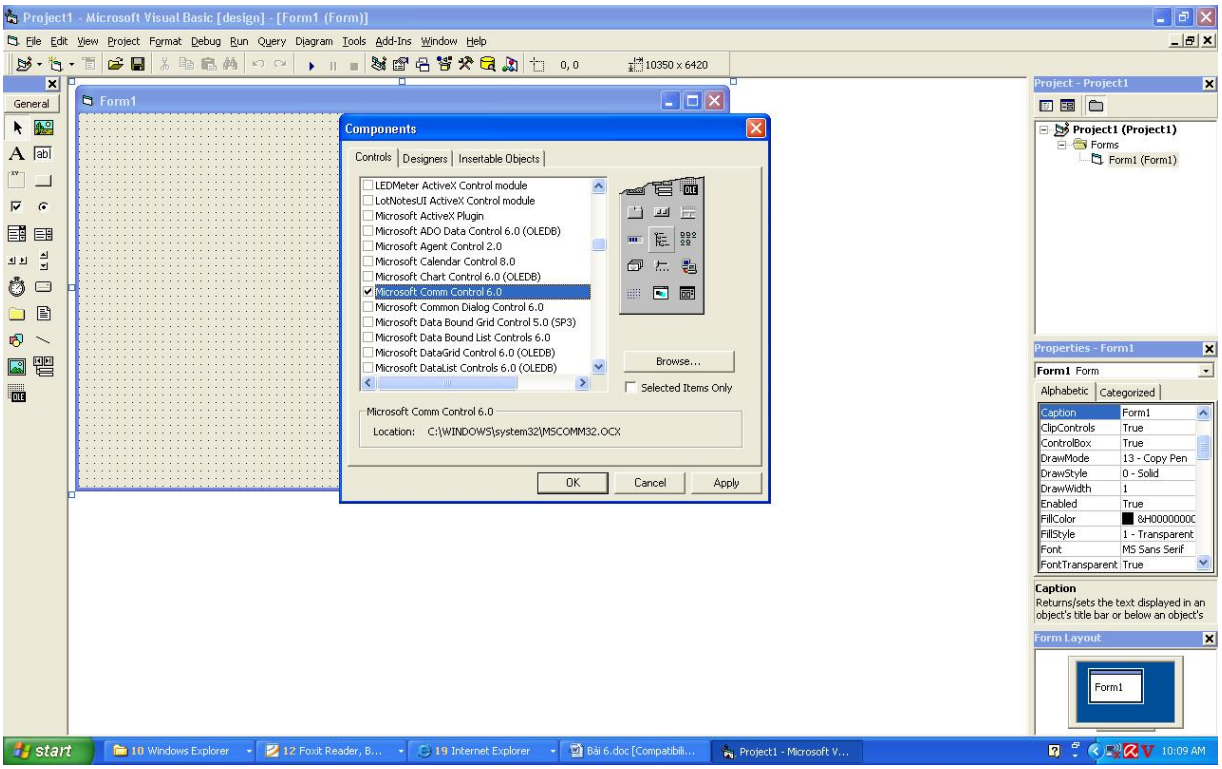
Visual Basic sử dụng một ActiveX MSCOMM, là một component trong thư viện Visual Basic để dùng để giúp máy tính giao tiếp với thiết bị ngoại vi.

2.3.2.1. Các bước để sử dụng MSCOM

Lấy component MSCOMM từ thư viện ra ToolBox

Bước 1: Kích chuột phải lên *ToolBox*, chọn *Components*





Bước 2: Hiện ra thư viện Components, chọn *Microsoft Comm Control 6.0* , nhấn Apply, Hình chiếc điện thoại sẽ hiện ra trên thanh ToolBox. Đây chính là MSCOMM, Cách để đưa đối tượng này vào ứng dụng tương tự như các đối tượng khác trong ToolBox.

2.3.2.2. Các thuộc tính quan trọng của MSCOMM:

Setting:

Cú pháp: MSCOMM1.Setting= "BBBB,P,D,S"

BBBB: Tốc độ truyền, chọn 9600 hoặc 19200

P: Dạng của bit kiểm tra lỗi chẵn lẻ

Giá trị	Mô tả
O	Odd (kiểm tra lẻ)
E	Even (kiểm tra chẵn)
M	Mark (luôn bằng 1)
S	Space (luôn bằng 0)
N	Không kiểm tra

D: Số bit dữ liệu 7 hoặc 8, mặc định là 8.

S: số bit STOP, 1,1.5 hoặc 2

Ví dụ: MSCOMM1.Setting= "9600,O,8,1"

+ CommPort: dạng object.CommPort = value. Value là chỉ số của cổng Com có giá trị từ 1 -> 16 và mặc định có giá trị =1. Các bạn cần phải thiết lập thông số này trước khi mở cổng. Sẽ có lỗi error 68 (Device unavailable) nếu như không mở được cổng này.

+ InBuferSize: thiết lập hoặc trả lại kích thước của bộ đệm nhận, tính = byte. Mặc định là

1024 byte. Các bạn không được nhầm lẫn với đặc tính InBufferCount là số byte đang chờ trong bộ đệm nhận.
+ InputLen : object.InputLen [= value] thiết lập hoặc trả lại số byte mỗi lần thuộc tính Input đọc trong bộ đệm nhận. Mặc định giá trị Value=0 tức là thuộc tính Input sẽ đọc hết nội dung của bộ đệm nhận khi thuộc tính này được gọi. Nếu số kí tự trong bộ đệm nhận không = InputLen thì thuộc tính Input sẽ trả lại kí tự rỗng "". Ví thế bạn cần phải chọn cách kiểm tra InBufferCount để chắc chắn số kí tự yêu cầu đã có đủ trước khi dùng lệnh .Input. Tính chất này rất là có ích khi đọc dữ liệu một máy mà dữ liệu ra được định dạng bằng các khối có kích thước cố định.
+ InputMode: object.InputMode [= value] .Value = 0 hay = comInputModeText dữ liệu nhận được dạng văn bản kiểu kí tự theo chuẩn ANSI. Dữ liệu nhận được sẽ là một sâu. Value=1 hay = comInputModeBinary dùng nhận mọi kiểu dữ liệu như kí tự điều khiển nhúng, kí tự NULL,.. Giá trị nhận được từ Input sẽ là một mảng kiểu Byte.
+ OutBuferSize: giống như InBuferSize, mặc định là 512.
+ ParityReplace: thiết lập và trả lại kí tự thay thế kí tự không đúng trong lỗi giống nhau.
+ PortOpen: thiết lập và trả lại tính trạng của cổng(đóng hoặc mở).
object.PortOpen [= value]. value = true cổng mở. value =false cổng đóng và xóa toàn bộ dữ liệu trong bộ đệm nhận và truyền. Cần phải thiết lập thuộc tính CommPort đúng với tên của cổng trước khi mở cổng giao tiếp. Thêm vào đó, cổng giao tiếp của thiết bị của bạn phải hỗ trợ giá trị trong thuộc tính Setting thì thiết bị của bạn mới hoạt động đúng, còn không thì nó sẽ hoạt động rất dở hơi nếu không nói là nó chạy không tốt. Đường DTR và RTS luôn giữ lại trạng thái của cổng.
+ RthresHold: object.Rthreshold [= value] value kiểu số nguyên. Thiết lập số kí tự nhận được trước khi gây lên sự kiện comEvReceive. Mặc định = 0 tức là không có sự kiện OnComm khi nhận được dữ liệu. Thiết lập = 1 tức là sự kiện OnComm xảy ra khi bất kì kí tự nào được chuyển đến bộ đệm nhận.
+ Settings: object.Settings [= value] thiết lập hoặc trả lại các thông số tần số baud, bít dữ liệu, bít chẵn lẻ, bít stop. Nếu Value không có giá trị khi mở sẽ gây ra lỗi 380 (Invalid property value).
+ SThreshold: thiết lập và và trả lại số kí tự nhỏ nhất được cho phép trong bộ đệm gửi để xảy ra sự kiện OnComm = comEvSend . Theo mặc định giá trị này = 0 tức là khi truyền sẽ không gây ra sự kiện OnComm. Nếu thiết lập thông số này =1 thì sự kiện OnComm xảy ra khi bộ đệm truyền rỗng. Sự kiện OnComm = comEvSend chỉ xảy ra khi mà số kí tự trong bộ đệm truyền nhỏ hơn hoặc = Sthreshold. Nếu số kí tự trong bộ đệm này luôn lớn hơn Sthreshold thì sự kiện này không thể xảy ra.
Truyền nhận dữ liệu:
+ CommEvent: trả lại phần lớn sự kiện giao tiếp hoặc có lỗi. CommEvent xảy ra khi có lỗi hoặc khi xảy ra sự kiện nào đó. Sau đây là một số hằng số lỗi:

Sự kiện	Giá trị	Miêu tả sự kiện
comEventBreak	1001	Xảy ra khi nhận được một tín hiệu Break.
comEventFrame	1004	Lỗi hệ thống. Phần cứng phát hiện ra một lỗi hệ thống
comEventOverrun	1006	Xảy ra khi cổng tự tràn(Overrun). Một kí tự không được

đọc từ phần cứng trước khi kí tự tiếp theo tới và do đó kí tự
này bị mất.
comEventRxOver 1008 Xảy ra khi bộ đệm nhận bị tràn. Không có đủ chỗ cho dữ liệu trong bộ đệm nhận.
comEventRxParity 1009 Lỗi Parity. Phần cứng phát hiện ra một lỗi Parity.
comEventTxFull 1010 xảy ra khi bộ đệm truyền bị đầy. Bộ đệm truyền bị đầy trong khi ghi dữ liệu lớn vào bộ đệm
comEventDCB 1011 Một lỗi không mong muốn khi đang khôi phục lại lỗi điều khiển thiết bị(DCB – Device Control Block) cho cổng
Một số sự kiện :
Sự kiện Giá trị Miêu tả sự kiện
comEvSend 1 Xảy ra khi số kí tự trong bộ đệm truyền nhỏ hơn giá trị SthresHold.
comEvReceive 2 Xảy ra khi bộ đệm nhận được số kí tự bằng giá trị RthresHold. Sự kiện này được tạo ra liên tục cho tới khi bạn dùng thuộc tính Input để lấy hết dữ liệu từ trong bộ đệm nhận.
RcomEvCTS 3 Xảy ra khi có thay đổi trong đường CTS(Clear To Send)
comEvDSR 4 Xảy ra khi thay đổi trong đường DSR(Data Set Ready). Sự kiện này chỉ xảy ra khi đường DSR thay đổi từ 1 -> 0.
comEvCD 5 Xảy ra khi có thay đổi trong đường CD(Carrier Detect)
comEvRing 6 Phát hiện chuông (Ring).Một số UART không hỗ trợ sự kiện này.
comEvEOF 7 Xảy ra khi nhận được kí tự kết thúc file (kí tự 26 trong bảng mã ASCII)
+ EOFEnable : object.EOFEnable [= value] quyết định các hành động nếu MSComm tìm thấy kí tự kết thúc file. Nếu value=true khi tìm thấy kí tự kết thúc file thì sẽ gây lên sự kiện comEvEOF trong OnCommEvent. Nếu value= false thì sẽ không gây lên sự kiện này.
+ InBufferCout: trả lại số kí tự đang có trong bộ đệm nhận Bạn có thể xoá bộ đệm nhận bằng cách đặt thuộc tính này =0 . Không nhầm với thuộc tính InBufferSize là tổng kích thước của bộ đệm nhận.
+ Input: nhận và xoá dữ liệu trong bộ đệm nhận.
Nếu InputMode là comInputModeText thì giá trị trả về sẽ là một xâu tức có kiểu String , dữ liệu dạng text trong một biến kiểu Variant. Nếu InputMode = comInputModeBinary thì thuộc tính này sẽ trả lại dữ liệu dạng nhị phân dưới dạng một mảng kiểu byte trong một biến Variant.
+ OutBufferCount: trả lại số kí tự trong bộ đệm truyền.
+ Output: ghi dữ liệu vào bộ đệm truyền. có thể truyền kiểu text hoặc kiểu nhị phân. Nếu truyền bằng kiểu text thì cho một biến Variant = kiểu String, nếu truyền kiểu nhị phân thì cho cho Output= variant = một mảng kiểu Byte.

2.3.2.3. Các bước cơ bản để thực hiện việc truyền và nhận từ máy tính dùng VB:

Cài đặt thuộc tính Setting qui định tốc độ truyền và khung truyền

Ví dụ: MSCOMM1.Setting= “9600,N,8,1”

Qui định thuộc tính RThreshold xác định số kí tự nhận được trong bộ đệm nhận làm phát sinh sự kiện Oncomm.

Ở đây nói thêm về sự kiện Oncomm: Đây có thể là sự kiện lỗi, sự kiện bộ đệm nhận nhận được N kí tự (N= RThreshold), sự kiện bộ đệm truyền truyền đi N kí tự (N=SThreshold) v.v

Vì vậy, nếu ta muốn sau khi bộ đệm nhận nhận đúng N kí tự thì tiến hành lấy dữ liệu vào biến nào đó để xử lý ta phải cài đặt trước thuộc tính RThreshold bằng N:

MSCOMM1.RThreshold=N

Và khi có N kí tự đến bộ đệm nhận, sự kiện OnComm sẽ xảy ra và lập tức chương trình VB sẽ nhảy vào hàm

Private Sub MSComm1_OnComm()

Thân hàm

End Sub

Trong thân hàm, ta sẽ xử lý dữ liệu. Ví dụ đọc dữ liệu vào biến:

Biến= MSCOMM1.INPUT

Qui định dạng dữ liệu nhận được dạng Text hay Binary bằng thuộc tính InputMode

Ví dụ: Nhận dạng dữ liệu dạng Binary

mscomm1.InputMode= comInputModeBinary

Nhận dữ liệu dạng Text

MSComm1.InputMode=comInputModeText

Muốn truyền dữ liệu đi dùng thuộc tính Output

MSCOMM1.Output= chuỗi cần truyền

2.3.3. Phần mềm trên vi điều khiển:

Sử dụng phần mềm CCS để lập trình cho vi điều khiển PIC

Các phần cần thiết khi muốn giao tiếp với máy tính :

Khai báo đúng tốc độ thạch anh đang dùng, nếu sai sẽ không truyền được

#use delay(clock=4000000) // Dùng thạch anh 4MHZ

Khai báo cài đặt truyền thông cho vi điều khiển theo chuẩn RS 232 sử dụng

#use rs232 (tham số cài đặt 1, tham số cài đặt 2,…)

Các tham số cài đặt có rất nhiều, chúng ta chỉ quan tâm đến một số các tham số cơ bản sau:

BAUD=x: Tốc độ truyền, hay dùng 9600 hoặc 19200. Lưu ý là tốc độ truyền phải giống như cài đặt trên máy tính

Ví dụ: BAUD=9600

XMIT= Pin: Qui định chân truyền dữ liệu là chân chức năng nào trên vi điều khiển, ví dụ: vi điều khiển 16F là RC6

Ví dụ: XMIT=PIN_C6

RCV= Pin: Qui định chân nhận dữ liệu là chân chức năng nào trên vi điều khiển, ví dụ: vi điều khiển 16F là RC7

Ví dụ: RCV=PIN_C7

Parity=x: Khai báo dạng kiểm tra chẵn lẻ, x= N, E hoặc O. Lưu ý là phải giống như cài đặt Parity của phần mềm trên máy tính

Ví dụ: Parity=N

Bits= n: Số bit dữ liệu, có thể là 7 hoặc 8

Stop=n: Số bit Stop, mặc định là 1

Hàm nhận dữ liệu:

KBhit(): cho biết trạng thái bộ đệm nhận có dữ liệu hay không

Bằng 0 nếu chưa có dữ liệu đến

Bằng 1 nếu đã có dữ liệu đến

Cú pháp: value= KBhit()

Getc():

Cú pháp: value=getc()

Hàm đợi cho đến khi vi điều khiển nhận được 1 kí tự và đọc kí tự vào biến value.

Vì hàm đợi nhận kí tự nên nếu không muốn mất thời gian đợi ta nên dùng hàm Kbhit() để xem vi điều khiển đã nhận kí tự hay không trước khi dùng hàm này để đọc kí tự.

Hàm truyền chuỗi kí tự: Printf()

Cú pháp: Printf(Chuỗi kí tự cần truyền)

Hàm này truyền đi một chuỗi kí tự từ vi điều khiển

Có thể định dạng chuỗi kí tự cần truyền bằng %nt

n là số kí tự cần truyền ví dụ: %2.3 nghĩa là truyền 2 kí tự phần nguyên và 3 kí tự thập phân

t là dạng truyền, có thể là một trong các dạng sau:

c	Character
s	String or character
u	Unsigned int
d	Signed int
w	Unsigned int with decimal place inserted. Specify two numbers for n. The first is a total field width. The second is the desired number of decimal places.

Hàm truyền 1 kí tự: Putc()

Cú pháp: Putc(kí tự cần truyền)

CHƯƠNG 3 :
MODULE THU PHÁT SÓNG RF

3.1. Khái niệm RF:

+ ***RF*** (Radio frequence): tần số sóng radio (sóng vô tuyến) Radio thực chất là sóng điện từ (bức xạ điện từ). Sóng điện từ có tần số từ 3Hz - 3GHz, sinh ra do một điện tích điểm dao động.

+ Sóng điện từ có đầy đủ các tính chất như sóng cơ học, nhưng sóng cơ học truyền đi trong những môi trường đàn hồi, còn sóng điện từ thì tự nó truyền đi mà không cần nhờ đến sự biến dạng của một môi trường đàn hồi nào cả,vì vậy nó truyền được cả trongchânkhông.

+Sóng điện từ bản chất là sóng nên nó thừa hưởng các đặc tính như giao thoa, cộng hưởng, triệt tiêu, nhiễu,... và có thể lan truyền đi xa trong môi trường nước, chân không, không khí.

+ Ứng dụng của sóng điện từ:

Sóng điện từ hiện nay được sử dụng rất rộng rãi trong thông tin vô tuyến truyền thanh và truyền hình, cũng như trong một số lĩnh vực khác như vô tuyến định vị (rađa), thiên văn vô tuyến, điều khiển bằng vô tuyến… Sóng điện từ được đặc trưng bằng tần số hoặc bằng bước sóng. Giữa bước sóng (đo bằng mét) và tần số (đo bằng hec) của sóng điện từ có hệ thức:

$$\lambda = \frac{c}{f} = \frac{3.10^8}{f}$$

Những dao động điện từ có tần số hàng chục và hàng trăm hec bức xạ rất yếu. Sóng điện từ của chúng không có khả năng truyền đi xa. Trong thông tin vô tuyến, người ta sử dụng những sóng có tần số từ hàng nghìn hec trở lên, gọi là sóng vô tuyến (sóng Radio)

3.2. Cơ bản về sóng vô tuyến :

Trong một phiên truyền thông, vì tận cùng bản chất của dữ liệu là bao gồm các bit 0 và 1, bên phát dữ liệu cần có một cách thức để gửi các bit 0 và 1 để gửi cho bên nhận. Một tín hiệu xoay chiều hay một chiều tự nó sẽ không thực hiện tác vụ này. Tuy nhiên, nếu một tín hiệu có thay đổi và dao động, dù chỉ một ít, sự thay đổi này sẽ giúp phân biệt bit 0 và bit 1. Lúc đó, dữ liệu cần truyền sẽ có thể gửi và nhận thành công dựa vào chính sự thay đổi của tín hiệu. Dạng tín hiệu đã điều chế này còn được gọi là sóng mang (carrier signal).

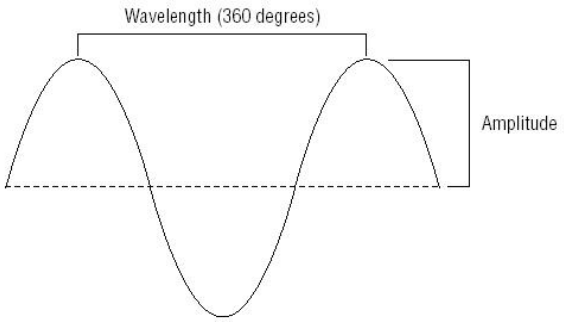
Có ba thành phần của dạng sóng có thể thay đổi để tạo ra sóng mang, đó là biên độ, tần số và pha. Tất cả các dạng truyền thông dùng sóng vô tuyến đều dùng vài dạng điều chế để truyền dữ liệu. Để mã hóa dữ liệu vào trong một tín hiệu gửi qua sóng AM/FM, điện

thoại di động, truyền hình vệ tinh, ta phải thực hiện một vài kiểu điều chế trong sóng vô tuyến đang truyền.

Sóng và truyền sóng :

+ Biên độ và bước sóng

Truyền thông vô tuyến bắt đầu khi các sóng vô tuyến được tạo ra từ một máy phát và gửi đến máy nhận ở một vị trí khác. Sóng vô tuyến tương tự như các cơn sóng mà bạn hay gặp ở biển hay hồ. Sóng có hai thành phần chính: biên độ và bước sóng.



Hình 3.2.1:Biên độ và bước sóng

Biên độ là chiều cao, độ mạnh hoặc công suất của sóng. Nếu bạn đang đứng trước biển khi các cơn sóng đi vào bờ, bạn có thể cảm nhận sức mạnh của những con sóng lớn so với những cơn sóng nhỏ. Thiết bị ăng-ten cũng thực hiện một chức năng tương tự nhưng với sóng vô tuyến. Các sóng lớn thường tạo ra nhiều tín hiệu điện trong một ăng-ten, giúp cho tín hiệu dễ nhận dàng nhận ra hơn.

Bước sóng là khoảng cách giữa hai điểm tương tự trên hai đỉnh sóng liên tiếp. Biên độ và tần số cả hai đều là các thuộc tính của sóng.

+ Bức xạ điện từ

Đầu tiên ta xét đến sóng điện từ. Bức xạ điện từ bao gồm sóng radio, vi ba, hồng ngoại, ánh sáng khả kiến, tia cực tím, tia X, và tia gamma. Tất cả chúng đều truyền đi với vận tốc ánh sáng là c = 3x108 m/s và tạo ra phổ điện từ. Sự khác nhau giữa các loại sóng điện từ này phụ thuộc vào bước sóng của mỗi thứ và chính cái gọi là bước sóng này liên quan trực tiếp đến năng lượng của sóng (bước sóng càng nhỏ thì năng lượng càng cao).

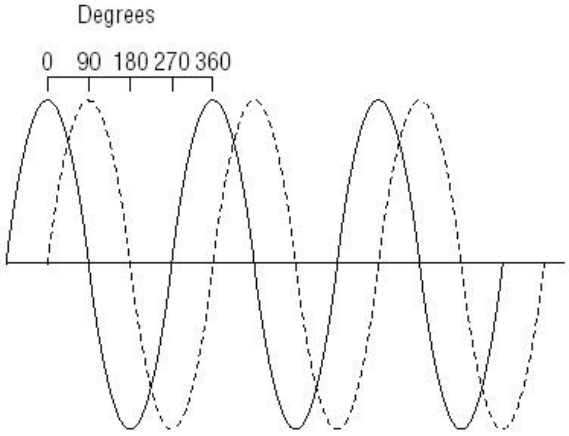
Khi chúng ta cung cấp một dòng điện xoay chiều có tần số cao vào một dây dẫn thì dây dẫn đó sẽ tạo ra sóng điện từ (radio wave) và truyền ra không gian theo phương thẳng về mọi hướng. Ví dụ dễ hình dung nhất là sự phát ánh sáng của mặt trời.

Nếu lấy một viên đá thả xuống hồ nước, ta sẽ tạo ra những gợn sóng nhấp nhô lên xuống và di chuyển dần ra xa, sóng điện từ cũng bức xạ ra ngoài với hình dáng y hệt như thế và phát năng lượng ra môi trường xung quanh.

+ Pha (Phase)

Pha là một thuật ngữ mang tính tương đối. Nó chỉ ra mối quan hệ giữa hai sóng có cùng tần số. Để xác định pha, bước sóng được chia thành 360 phần, được gọi là độ. Nếu bạn nghĩ thông số độ này tựa như thời gian bắt đầu thì nếu có một sóng bắt đầu từ điểm 0 độ và một sóng khác bắt đầu lúc 90 độ, hai sóng này được xem là lệch pha nhau 90 độ.

Trong một môi trường lý tưởng, sóng được tạo ra và truyền từ một máy này và nhận một cách hoàn hảo bên máy kia. Tuy nhiên, truyền thông vô tuyến không xảy ra trong môi trường lý tưởng. Có nhiều nguồn gây nhiễu và nhiều vật cả ảnh hưởng sóng khi nó đang di chuyển. Hình vẽ dưới đây vẽ hai sóng đang lệch pha nhau 90 độ.



Hình 3.2.2:Biên độ và pha sóng

+ Thời gian và pha

Giả sử bạn có hai đồng hồ đang đứng yên và cả hai cùng chỉnh về 12 giờ. Lúc 12 giờ, bạn khởi động đồng hồ đầu tiên và sau đó một giờ, bạn khởi động đồng hồ thứ hai. Đồng hồ thứ hai được xem là đi chậm hơn đồng hồ thứ nhất một giờ. Khi thời gian trôi đi, đồng hồ thứ hai cũng vẫn đi sau đồng hồ thứ nhất 1 giờ. Cả hai đồng hồ cùng duy trì một ngày 24 giờ nhưng cả hai không đồng bộ với nhau. Các sóng lệch pha nhau thì cũng là hai sóng xuất phát ở những thời gian khác nhau. Cả hai sóng sẽ hoàn thành chu kỳ 360 độ nhưng nó sẽ lệch pha với nhau.

+ Các phương thức điều chế

Để dữ liệu có thể được truyền, tín hiệu phải được xử lý sao cho bên máy nhận có cách để phân biệt bit 0 và 1. Phương pháp xử lý tín hiệu sao cho nó tượng trưng cho nhiều mẫu dữ liệu được gọi là điều chế. Phương thức này sẽ biến tín hiệu vào trong sóng mang. Phương thức này mã hóa dữ liệu sao cho nó có thể truyền. Có ba kiểu điều chế: điều biên (Amplitude Shift Keying - ASK), điều tần – (Frequency Shift Keying - FSK), và điều pha (Phase Shift Keying - PSK).

Ý tưởng cho phần điều chế tín hiệu trong WLAN là để chứa càng nhiều dữ liệu càng tốt vào trong tín hiệu và để giảm thiểu lượng dữ liệu có thể bị mất do nhiễu. Khi dữ liệu bị mất, nó phải được truyền lại, và vì vậy làm tốn tài nguyên của mạng không dây.

Có hai kỹ thuật khác nhau được dùng để mô tả dữ liệu:

Trạng thái hiện hành (current state):

Với kỹ thuật này, giá trị hiện hành của tín hiệu được dùng để phân biệt giá trị 0 và 1. Kỹ thuật này sẽ gán một giá trị cụ thể để chỉ ra giá trị nhị phân là 0 hay là 1. Ở một thời điểm

cụ thể, giá trị của tín hiệu sẽ xác định giá trị nhị phân. Ví dụ, bạn có thể mô tả giá trị 0 và 1 bằng một cánh cửa bình thường. Mỗi một phút, bạn kiểm tra xem cửa là đóng hay mở. Nếu cửa là đang mở, nó tượng trưng cho giá trị 0. Nếu cửa đang đóng, nó tượng trưng cho giá trị 1. Tình trạng hiện thời của cánh cửa, đóng hay mở, sẽ xác định giá trị 0 hay 1.

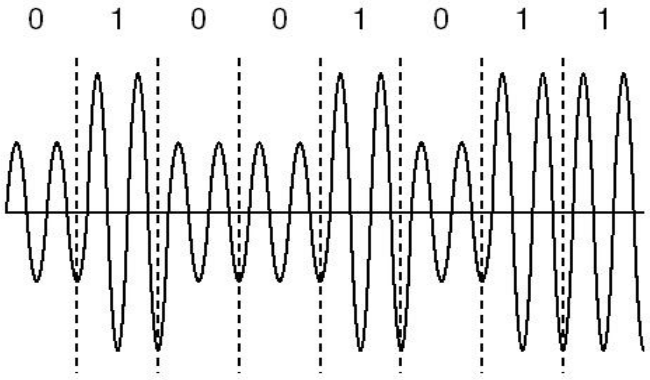
Chuyển trạng thái (state transition):

Với kỹ thuật này, sự thay đổi hay chuyển tín mức tín hiệu sẽ được dùng để phân biệt 0 và 1. Kỹ thuật này có thể mô tả giá trị 0 bằng cách thay đổi pha của sóng ở một thời điểm cụ thể, trong khi giá trị 1 sẽ đặc trưng bằng việc giữ nguyên pha. Ở một thời điểm cụ thể, yếu tố có sự thay đổi hay không có sự thay đổi trong pha của tín hiệu sẽ được dùng để xác định giá trị nhị phân. Ví dụ cánh cửa bên trên có thể được dùng lại một lần nữa để minh họa. Cứ mỗi một phút, nếu cánh cửa đang di chuyển (dù để mở hay để đóng), nó tượng trưng cho giá trị 0. Nếu cánh cửa đứng yên (dù đang mở hay đóng), nó tượng trưng giá trị 1. Trong ví dụ này, trạng thái chuyển đổi (di chuyển hay không di chuyển) sẽ xác định giá trị 0 hay 1.

Điều biên

Điều biên thay đổi biên độ/ độ cao của tín hiệu để mô tả dữ liệu nhị phân. Điều biên dùng kỹ thuật trạng thái hiện hàng, trong đó một mức biên độ được dùng để tượng trưng mức 0 và một mức được dùng để tượng trưng mức 1. Hình bên dưới mô tả làm thế nào một dạng sóng có thể điều chế một mã ASCII, ký tự K dùng phương pháp điều biên. Biên độ lớn tượng trưng cho bit 1, trong khi biên độ nhỏ tượng trưng cho mức 0.

Chính biên độ của sóng sẽ xác định dữ liệu đang được truyền. Bên máy nhận, đầu tiên máy nhận sẽ chia tín hiệu nhận được ra thành những khoản thời gian được gọi là thời gian lấy mẫu. Máy nhận sau đó sẽ kiểm tra sóng để tìm ra biên độ. Tùy thuộc vào giá trị biên được truyền độ của sóng, máy nhận sẽ xác định giá trị nhị phân đang.



Hình 3.2.3:Phương pháp điều biên sóng

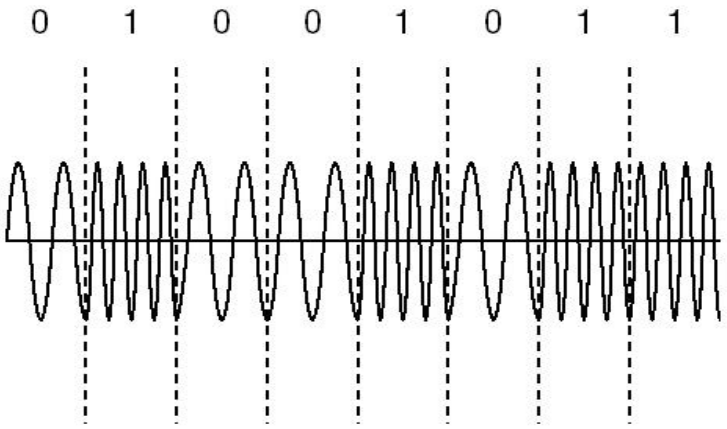
Như bạn cũng có thể biết, các tín hiệu không dây thì có thể khó dự đoán và cũng có thể bị nhiễu từ nhiều nguồn. Khi nhiễu xảy ra, nó thường ảnh hưởng đến biên độ của tín hiệu. Vì khi có một sự thay đổi trong biên độ có thể làm cho máy nhận diễn dịch sai giá trị dữ liệu, kỹ thuật này phải được dùng một cách cẩn thận.

Điều tần

Điều tần thay đổi tần số tín hiệu để mô tả dữ liệu nhị phân. Điều biên dùng kỹ thuật trạng thái hiện hành, trong đó một mức tần số có thể tượng trưng cho bit 0 và một tần số khác tượng trưng cho bit 1. Sự thay đổi tần số sẽ xác định dữ liệu đang được truyền. Bên máy nhận lấy mẫu của tín hiệu, nó sẽ xác định tần số của sóng, và tuỳ thuộc vào giá trị tần số, máy nhận sẽ xác định giá trị nhị phân.

Hình dưới đây sẽ mô tả làm thế nào một dạng sóng có thể mô tả ký tự K của bảng mã ASCII dùng kỹ thuật điều tần. Mức tần số nhanh hơn được diễn dịch như mức 1 và mức tần số thấp hơn được diễn tả như mức 0.

Trong các chuẩn 802.11 ban đầu, kỹ thuật điều tần FSK được dùng. Khi yêu cầu truyền thông nhanh hơn, kỹ thuật FSK sẽ đòi hỏi nhiều kỹ thuật đắt hơn để hỗ trợ tốc độ nhanh hơn. Điều này làm cho nó không còn thực tế.

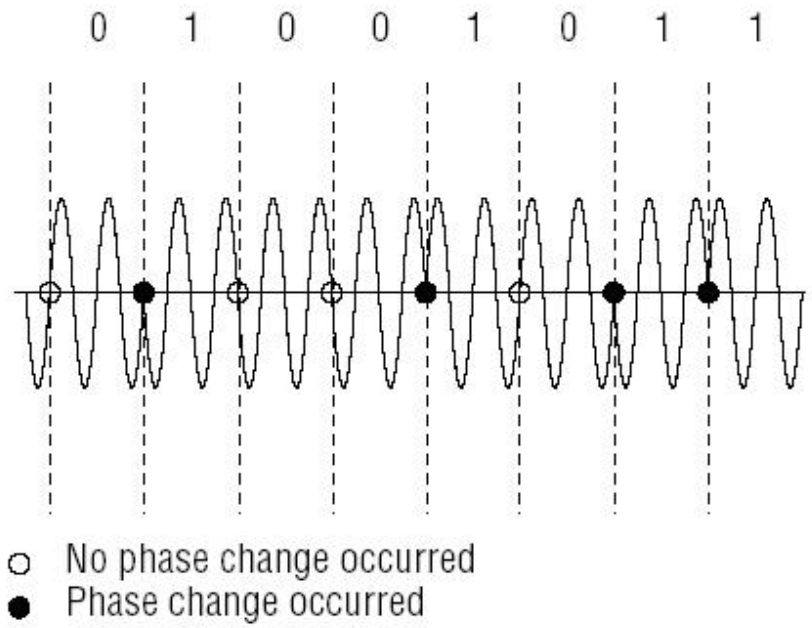


Hình 3.2.4:Phương pháp điều tần sóng

Điều pha

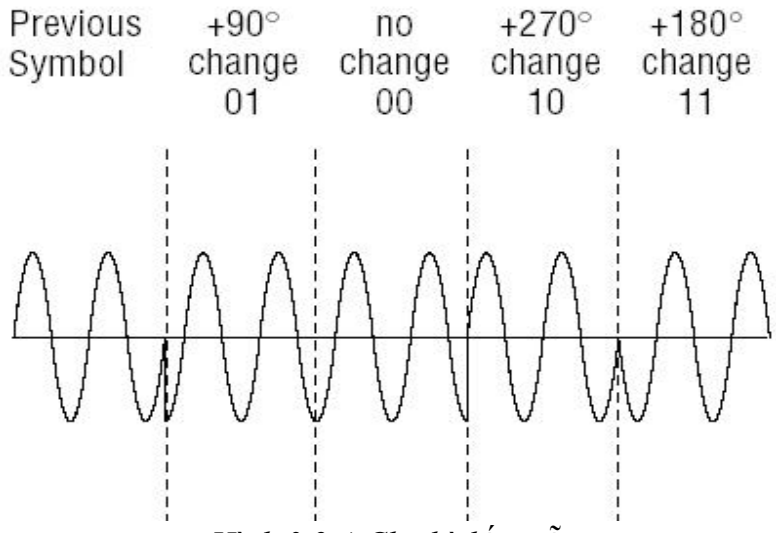
Kỹ thuật này sẽ thay đổi pha của tín hiệu để mô tả dữ liệu nhị phân. Điều pha dùng kỹ thuật thay đổi trạng thái, trong đó một pha dùng để mô tả bit 0 và một pha khác dùng để mô tả mức 1. Sự thay đổi trạng thái của pha sẽ xác định dữ liệu đang được truyền. Khi máy nhận lấy mẫu tín hiệu, nó sẽ xác định pha và trạng thái của bit.

Hình dưới đây mô tả làm thế nào một dạng sóng có thể mô tả ký tụ K dùng kỹ thuật điều pha. Pha thay đổi đầu chu kỳ dùng để mô tả giá trị nhị phân là 1. Nếu pha không thay đổi ở đầu chu kỳ thì mô tả giá trị 0. Điều pha dùng nhiều trong các chuẩn 802.11. Một cách tiêu biểu, bên máy nhận sẽ lấy mẫu tín hiệu và so sánh pha của mẫu hiện hành với pha trước đó và xác định sự khác nhau. Sự khác nhau trong pha (lệch pha) sẽ được dùng để xác định giá trị bit.



Hình 3.2.3:Phương pháp điều pha sóng

Các phiên bản cao cấp của điều pha có thể mã hóa nhiều bit. Thay vì dùng hai pha để xác định giá trị nhị phân, bốn pha có thể được dùng. Mỗi pha trong bộ bốn này có thể dùng để mô tả hai giá trị nhị phân (00,01,10,11) thay vì chỉ 1 (0 hoặc 1), như vậy sẽ giúp ngắn thời gian truyền. Hình dưới đây mô tả ký tự K trong bảng mã ASCII có thể được mã hóa và truyền dùng kỹ thuật này. Bạn chú ý là có ít chu kỳ lấy mẫu hơn hình vẽ trước.

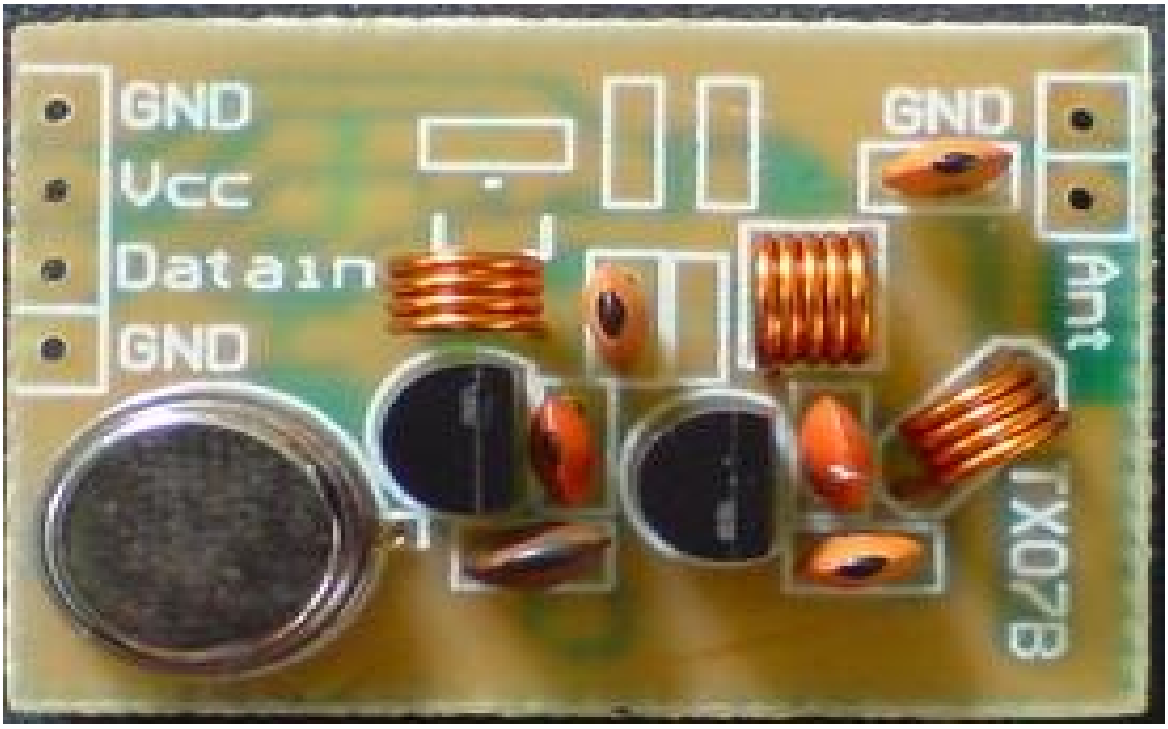


Hình 3.2.4:Chu kỳ lấy mẫu

Hiện nay, việc truyền dữ liệu số được sử dụng rrất rộng rãi, nhất là trong lĩnh vực điều khiển thông tin số. nhiều vi mạch hỗ trợ xử lí tín hiệu không dây như PT2248,PT2249,PT9148,PT9149,HT12D… vấn đề đặt ra là các vi mạch này truyền dữ liệu chỉ dành cho mục đích riêng là điều khiển thiết bị, thôn tin truyền đi đã được mã hóa sẵn, số bít truyền dữ liệu đi thấp, khôn phù hợp với nhu cầu truyền dữ liệu hàng loạt và liên tục.

Giải quyết vấn đề này, ta tận dụng khả năng của VDK về truyền nhận dữ liệu nối tiếp nhờ vào bộ UART trong chip. VDK có khả năng truyền thong đa xử lí rất thích hợp cho việc truyền dữ liệu trong mộ hệ thống mạng không dây gồm nhiều bộ xử lý tớ.

3.3. Modul phát RF TX07B :



Hình 3.3.1 : Modul phát RF TX07B

Modul phát RF TX07B là modul OOK dùng để truyền dữ liệu digital chưa có thành phần mã hóa, được thiết kế với 4 chân chính đó là chân VCC(chân cấp nguồn), chân GND(chân ground của mạch), chân data in(dữ liệu truyền đi được đưa vào chân này), chân ant(dùng để nối anten phát).

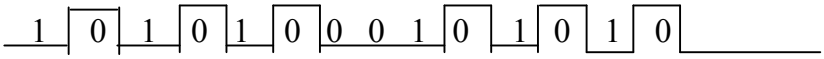
Chức năng của các chân

+ Chân VCC và chân ground : là dùng để cấp nguồn 1 chiều vào cho mạch (chân VCC nối vào nguồn +9V đến +12V, chân gnd nối ground của nguồn).

+ Chân data in : là chân đưa các tín hiệu 0 và 1 vào đây(mức tín hiệu chuẩn của TTL 0 là 0V và 1 là 5V).Khi ta đưa tín hiệu 0 vào thì mạch vẫn chưa có gì, khi đưa tín

hiệu 1 vào thì mạch phát được kích hoạt và phát ra sóng điện từ.Cứ liên tục đưa xung 0,1 vào thì mạch sẽ ngừng, phát...(ON OFF KEYING).

Hình dưới đây mô tả việc phát dữ liệu 0,1 của mạch phát RF :



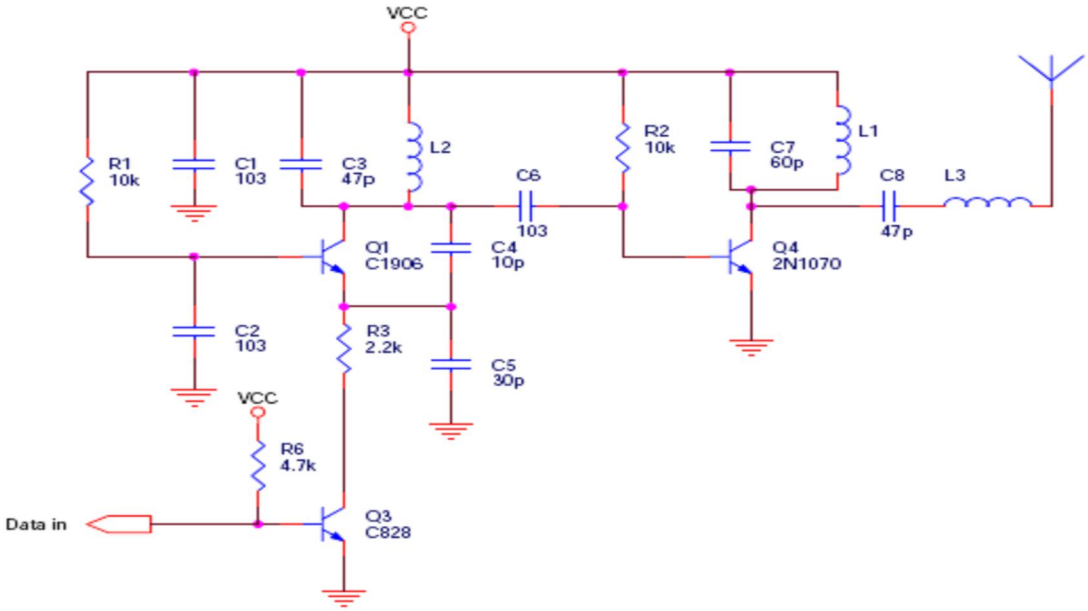
Hình 3.3.2:Phát dữ liệu

+ Chân ant : Dùng để nối anten vào đây, để mạch phát có thể phát được dữ liệu ra ngoài thì cần hàn nối anten vào chân này.

3.3.1. Sơ đồ khối :

Hình 3.3.3:Sơ đồ khối mạch RF

3.3.2. Sơ đồ mạch nguyên lí của modul phát RF :



Hình 3.3.4 : Sơ đồ mạch nguyên lí của mudul phát RF

3.3.3. Modul thu RF R05C :

Hình 3.3.3.1: *Modul phát RF R05C*

Đây là modul thu được chế tạo đơn giản với đầu ra data out, VCC, GND, và ant.

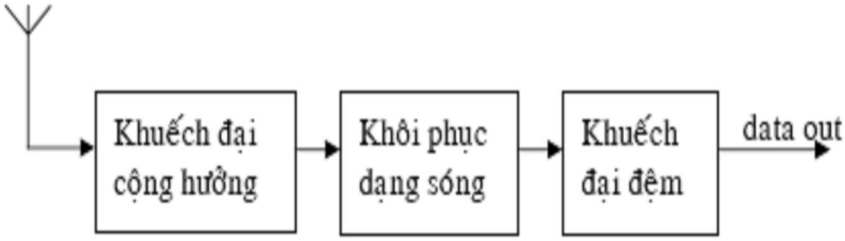
+ Chân ant : chân này dùng để gắn anten vào, mạch thu chỉ thu được tốt với khoảng cách xa khi được hàn nối anten đúng quy định.

+ VCC và GND : Cấp nguồn 1 chiều vào chân này(VCC nối 5V, GND nối ground của nguồn).

+ Chân data out : Chân này sẽ đưa ra các xung 0,1 khi máy phát phat tín hiệu 0,1.

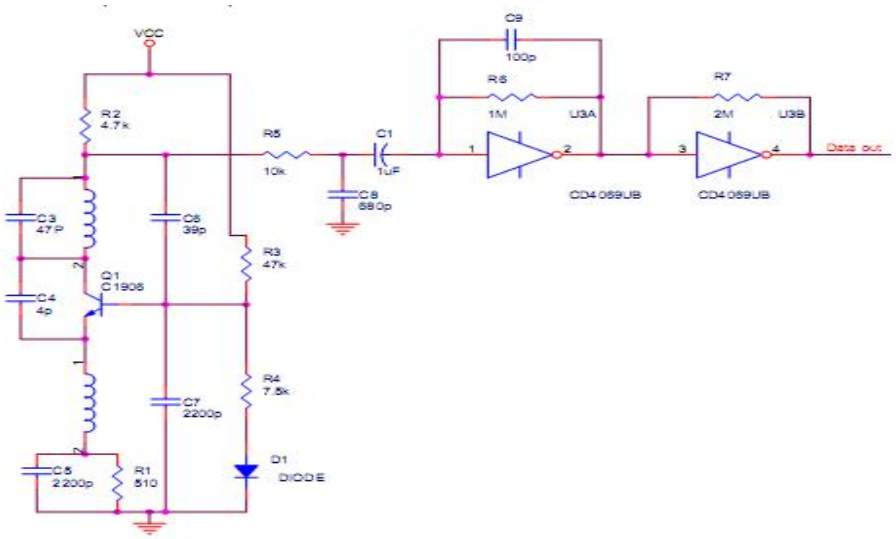
* Đặc điểm của modul thu : Bình thường khi cấp nguồn cho modul thu thì đầu ra data out có mức 0,1 ngẫu nhiên không xác định. Mức 0,1 này là do yếu tố môi trường tác động, bởi thế khi lập trình giải mã cho modul thu ta không dùng ngắt nhận ở đầu và mà dùng phương pháp hỏi vòng, loại trừ. Nếu dùng ngắt thì sẽ bị nhảy liên tục vào ngắt và kết quả ta sẽ chẳng nhận được gì.

3.3.4. *Sơ đồ khối mạch thu :*



Hình 3.3.4.1: Sơ đồ khối mạch thu

3.3.5. Sơ đồ nguyên lí mạch thu :



Hình 3.3.5.1 : Sơ đồ nguyên lí mạch thu RF

3.4. Lập trình mã hóa và giải mã tín hiệu thu phát RF :

3.4.1. Lập trình mã hóa :

Tốc độ baud : Là số bít truyền được trong 1 giây. Mỗi modul thu phát có thể hỗ trợ tốc độ thu phát nhanh nhất và chậm nhất (do nhà sản xuất làm ra) . Muốn cho modul làm việc chính xác, hiệu quả thì cần phải lập trình truyền nhận ở tốc độ đó.

+ Nếu ta phát tín hiệu 1 quá dài thì modul thu có lên mức 1 tuy vậy nó sẽ giảm theo thời gian và ở thời gian quá dài thì đầu ra của modul thu không còn là mức 1 nữa.

Nếu ta phát tín hiệu 0 quá dài : modul thu cũng đưa ra mức 0 tuy nhiên theo thời gian nó sẽ thu nhiễu của môi trường và đầu ra của modul thu không còn là mức 0 nữa.

+ Nếu ta phát tín hiệu 0 hoặc 1 quá ngắn(tốc độ cao hơn giá trị của nhà sản xuất quy định) thì có thể modul thu sẽ không thể bắt kịp được tốc độ đó và không thể thu được dữ liệu.

+ Phát tín hiệu để báo hiệu cuộc trao đổi dữ liệu bắt đầu: Khi ở trạng thái bình thường đầu ra của modul thu có mức tín hiệu không xác định(0 hoặc 1) liên tục. Lúc này bộ thu có độ lợi(gain) điện áp tín hiệu là rất lớn. Nếu ta phát ngay tín hiệu(dữ liệu) lúc này thì có thể bộ thu không kịp nhận biết là ta đang gởi dữ liệu cho nó. Bởi thế ta cần làm một khâu gọi là chuẩn bị (tức là phát một chuỗi xung 0,1) từ 20 đến 40ms. Việc phát chuỗi xung này nhằm để điều chỉnh lại độ lợi của của mạch thu (báo hiệu cho nó) đưa nó sẵn sàng bước vào phiên giao tiếp.

Mã hóa Manchester :

+ Đặc điểm của mã hóa : Khi ta phát một dữ liệu 0 hay 1 quá dài thì có thể làm mạch thu hoạt động sai. Mã Manchester có nhiệm vụ phân đều các khoảng thời gian dài thành những khoảng thời gian nhỏ phù hợp với modul phát và thu.

+ Mã Manchester sẽ mã hóa tín hiệu 1 thành 10 và tín hiệu 0 thành 01. VD ta mã hóa tín hiệu 10110100 tín hiệu này được mã hóa thành 1001101001100101. Độ rộng của bit 1 bằng bit 0.

+ Phương pháp lập trình mã hóa như sau :

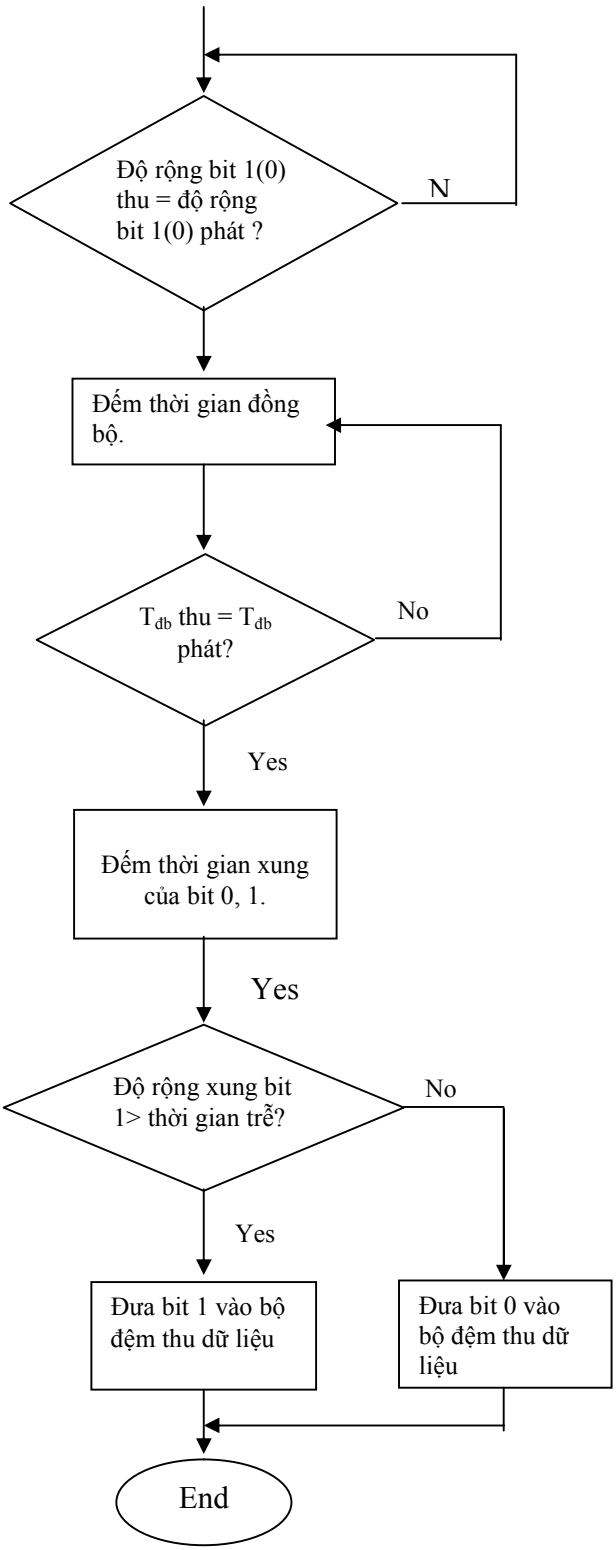
Phát một chuỗi xung trong khoảng 30s (bước chuẩn bị).

Nghĩ một khoảng thời gian(thời gian này để đồng bộ với máy thu khoảng 1200us).

Đưa bit dữ liệu cần phát vào bộ đệm biến chúng thành 01 nếu là bit 0 hoặc thành 10 nếu là bit 1.

Phát một xung kết thúc(xung này để đảm bảo dữ liệu không bị bỏ sót) kết thúc phiên giao tiếp.

3.4.2. .Lập trình giải mã :


Độ rộng bit 1(0) thu = độ rộng bit 1(0) phát ?
N
Đếm thời gian đồng bộ.
T_{db} thu = T_{db} phát?
No
Yes
Đếm thời gian xung của bit 0, 1.
Yes
Độ rộng xung bit 1> thời gian trễ?
No
Yes
Đưa bit 1 vào bộ đệm thu dữ liệu
Đưa bit 0 vào bộ đệm thu dữ liệu
End

CHƯƠNG 4:
CẤU TRÚC VÀ THIẾT KẾ PHẦN CỨNG

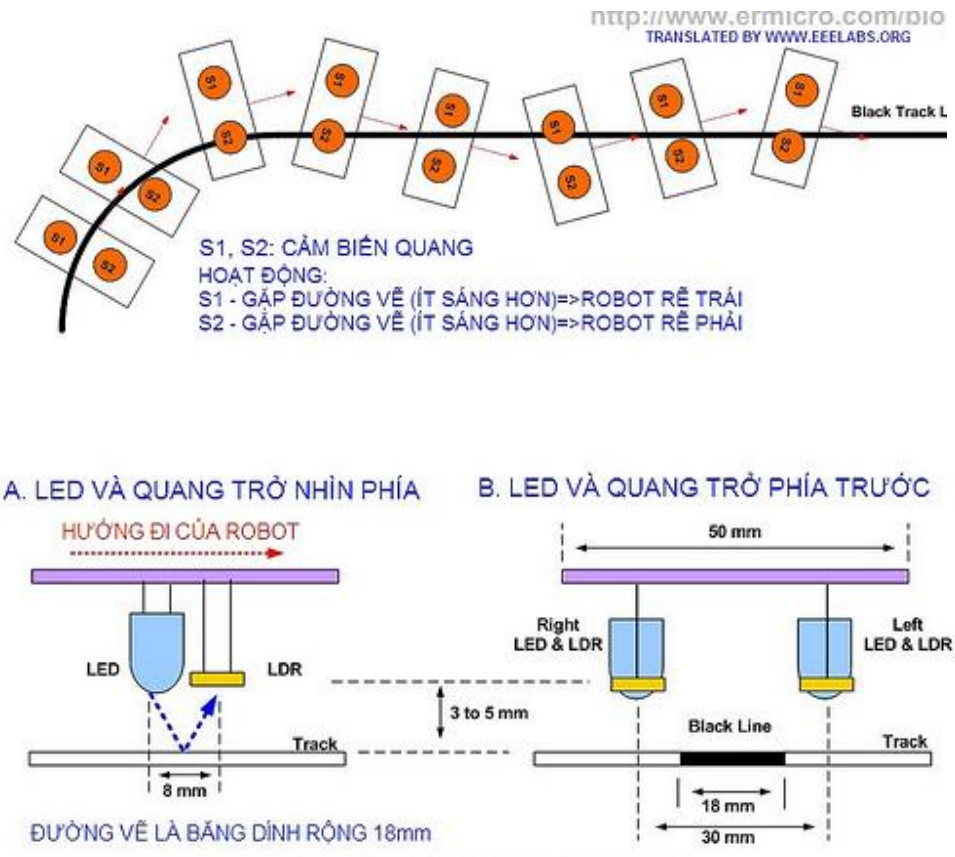
4.1. Thiết kế module điều khiển robocon.

4.1.1. Thiết kế module sensor dò line dùng quang trở.

4.1.1.1. Các loại cảm biến có thể dùng cho robot tự động.

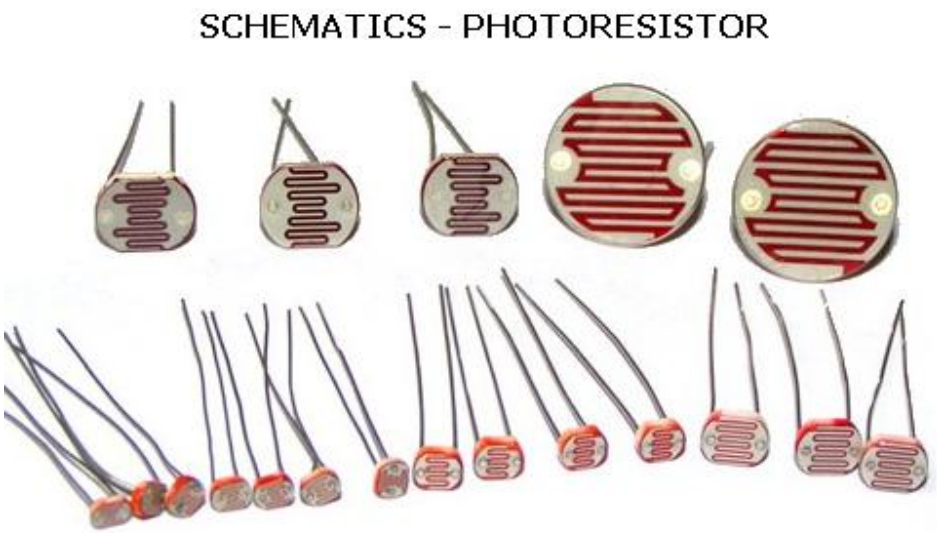
+ Nguyên lý hoạt động của robot dò đường. Thật ra, robot dò đường là 1 biến thể đặc biệt của robot hướng sáng. Sở dĩ nói như vậy là do chúng có cùng nguyên tắc hoạt động là sử dụng cảm biến quang điện (quang trở hoặc diode hồng ngoại) để so sánh cường độ sáng từ đó điều chỉnh hướng đi thích hợp.Tuy nhiên, ở robot dò đường, cảm biến được bố trí gần mặt đường và nguồn sáng để so sánh lúc này do chính robot tạo ra. Nhưng do đâu lại có sự sai lệch về cường độ sáng. Câu trả lời nằm ở đường vẽ, đường vẽ này có tính chất khác với xung quanh, thường thì nó có màu đen để hấp thụ ánh sáng. Khi robot đi lệch vào vùng có vạch vẽ, ánh sáng phát ra từ robot không phản xạ lại như bình thường mà bị đường kẻ hấp thụ 1 phần làm sai lệch độ sáng giữa 2 cảm biến. Việc còn lại là thiết kế sao cho robot có hành vi khắc phục sự sai lệch đó và ta có được loại robot đi theo đường vẽ.

+ Nhiệm vụ của các cảm biến quang được mô tả như hình sau:



Hình 4.1.1.1 :Vị trí lắp đặt cảm biến

4.1.1.2. Quang trở:



Hình 4.1.1.2 :Hình dáng bên ngoài quang trở

+ Quang trở là điện trở mà hoạt động của nó dựa trên hiệu ứng quang dẫn.

+ Cấu tạo: Quang trở được làm từ chất bán dẫn nhạy quang (có thể là Cadmium Sulfide – CdS, Cadmium Selenide – CdSe).

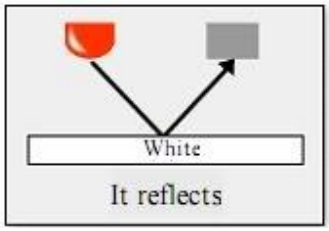
+ Nguyên lý làm việc của quang trở là khi có bức xạ chiếu vào, chất bán dẫn hấp thụ năng lượng làm phát sinh các điện tử tự do và lỗ trống, tức sự dẫn điện tăng lên và làm giảm điện trở của chất bán dẫn. Các đặc tính điện và độ nhạy của quang trở dĩ nhiên tùy thuộc vào vật liệu dùng trong chế tạo.

4.1.1.3. Thiết kế board sensor:

+ Phần phát dùng 8 con led siêu sáng, phần thu là 8 quang trở theo từng cặp bố trí ở mặt sau của board như hình dưới:

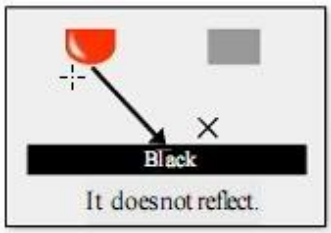
+ Led siêu sáng chiếu ánh sáng xuống Line và phản xạ ngược trở lại quang trở tương ứng.

Nếu Line là Line trắng thì ánh sáng phản xạ lại quang trở với cường độ mạnh hơn, ứng với mức Logic 1.



Hình 4.1.1.3.1 :Phản xạ trên nền trắng của cảm biến

Nếu Line là Line đen hoặc màu xanh xẩm tương ứng thì ánh sáng gần như bị hấp thụ và phản xạ lại quang trở rất ít, ứng với mức Logic 0.



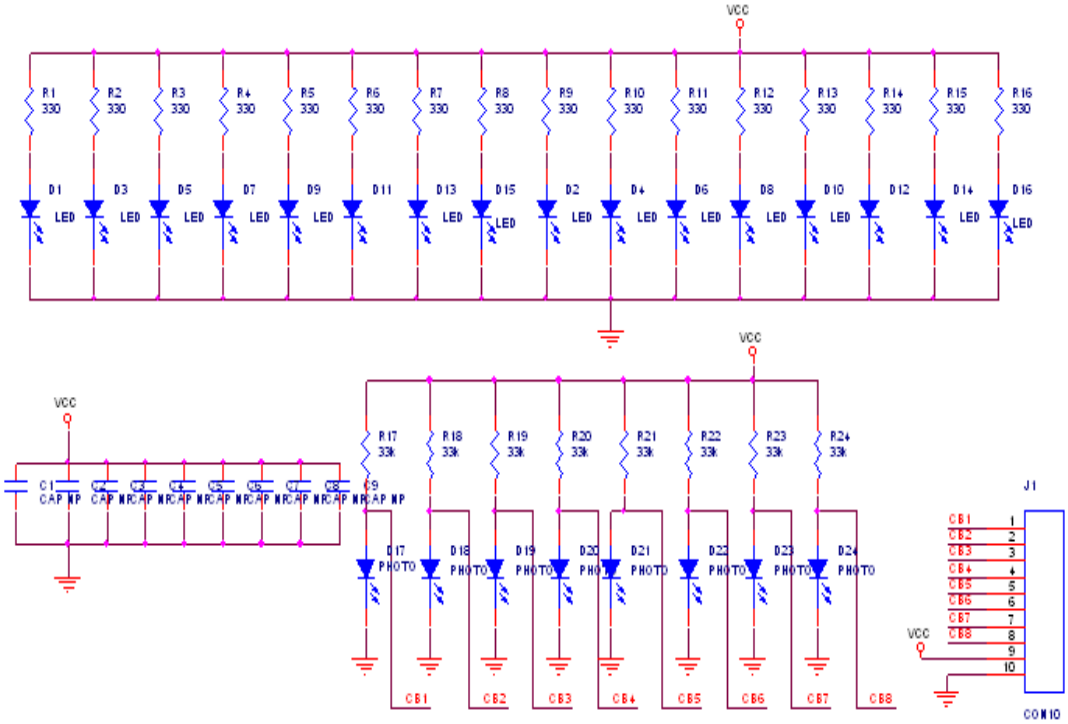
Hình 4.1.1.3.2 :Phản xạ trên nền đen của cảm biến

+ Cường độ ánh sáng hấp thụ vào quang trở có thể được chỉnh bằng biến trở chỉnh (gắn ở mặt trên của board sensor). Bằng việc chỉnh biến trở ứng với từng cặp Led phát – quang trở ta chỉnh được độ nhạy của quang trở.

+ Cần lưu ý vấn đề chống nhiễu cho sensor, phải che chắn cho các quang trở và leds phát.

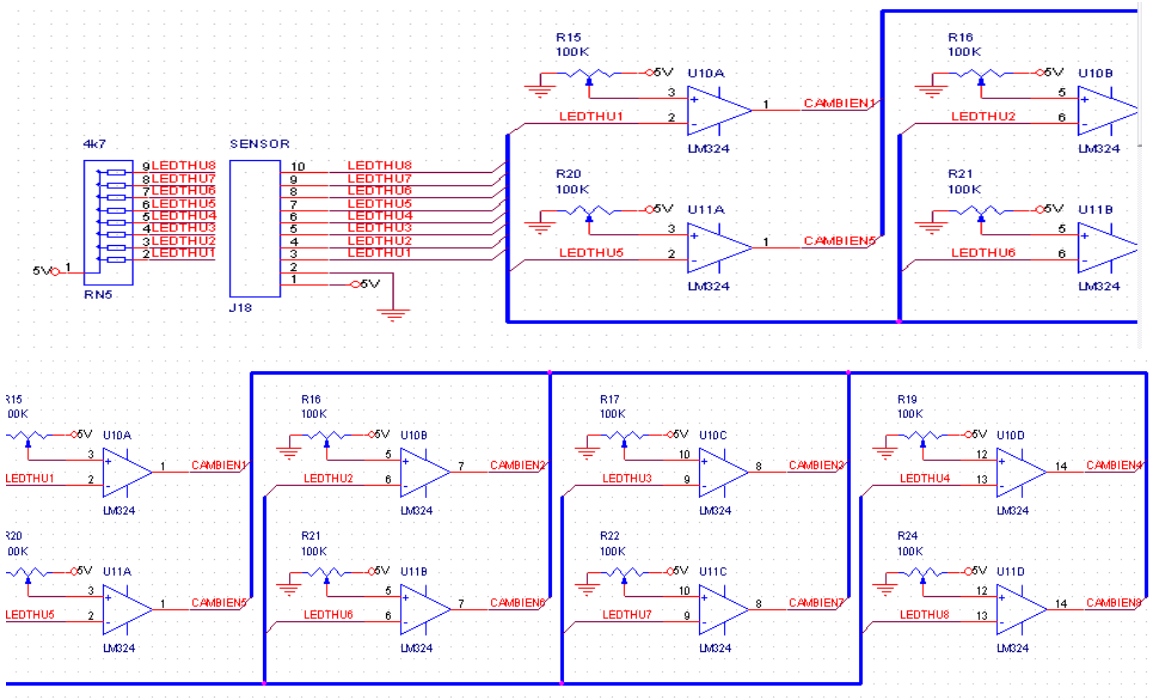
- Sơ đồ nguyên lý:

Mạch sensor:



Hình 4.1.1.3.3:Sơ đồ nguyên lý mạch sensor

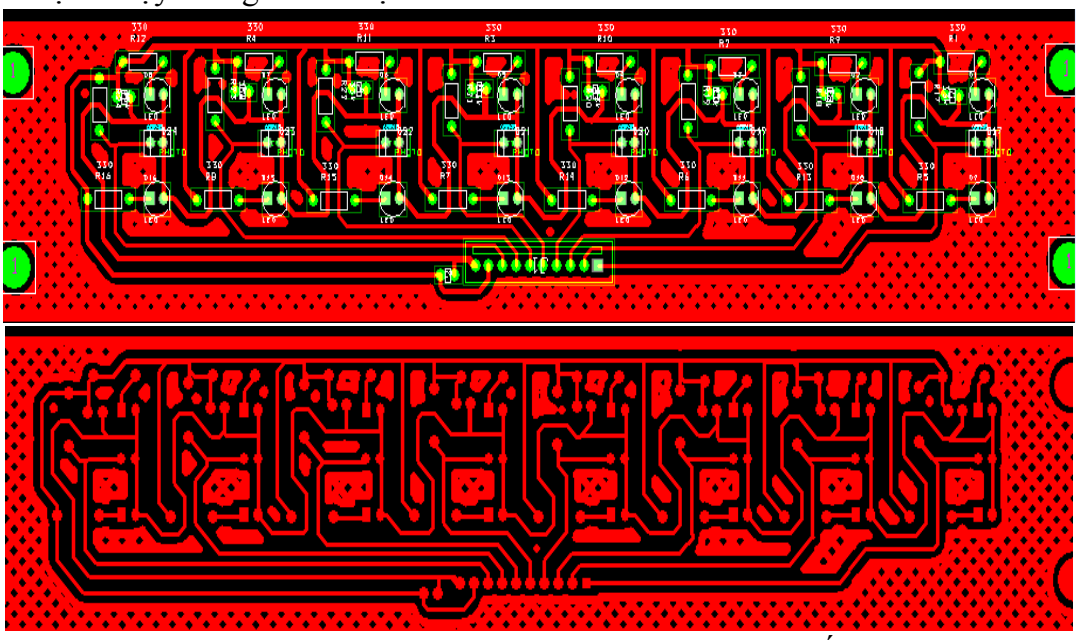
Mạch so sánh:



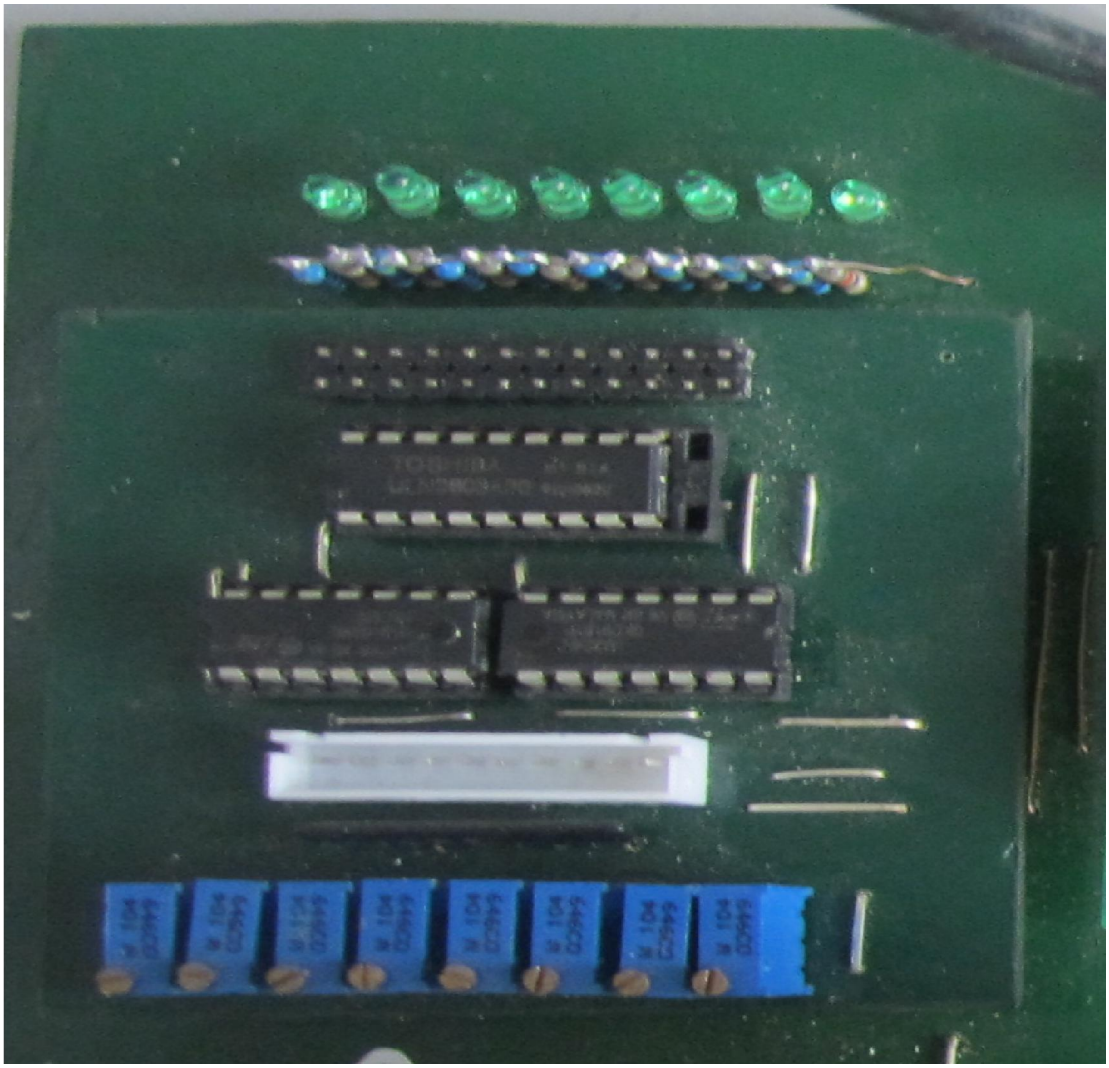
Hình 4.1.1.3.4:Sơ đồ nguyên lý mạch so sánh

4.1.1.4. Kết quả.

Mạch chạy tương đối ổn định



Hình 4.1.1.4.1:Mạch lay_out cảm biến

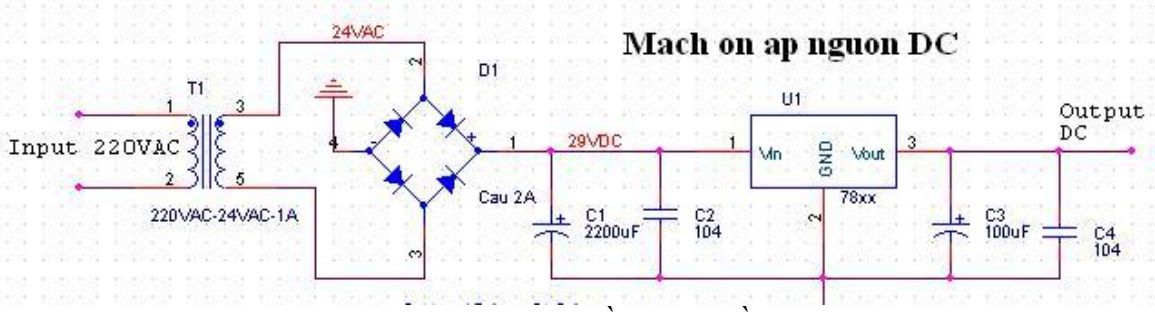


Hình 4.1.1.4.2:Mạch so sánh

4.1.2. Thiết kế khối mạch nguồn

Nguồn điện là thành phần không thể thiếu được trong các mạch điện tử và nó đóng thành phần quan trọng ảnh hưởng tới hoạt động của mạch. Việc cung cấp nguồn điện 1 chiều có điện áp 5V, 6V, 9V, 12V, 15V,18V, 24V để cung cấp cho các thiết bị điện tử thông dụng chạy điện 1 chiều. Do ngoài thực tế do nguồn điện chúng ta không ổn định so với giá trị yêu cầu ở lý thuyết nên ổn áp lại 1 giá trị điện áp đầu ra không thay đổi để cho mạch điện chúng ta hoạt động ổn định và chính xác. Ở đây sử dụng bộ nguồn 5V và 12V. Vì các linh kiện dùng nguồn đầu vào là 5V chuẩn .Thông thường có 2 phương pháp ổn áp. Ổn áp dùng IC số và ổn áp dùng transitor kết hợn với zenner ổn áp. Ở đây dùng mạch ổn áp đơn giản dùng họ 78xx với dòng điện <=1A.

Sơ đồ mạch điện đơn giản



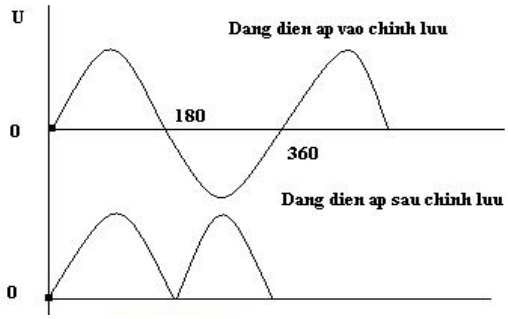
Hình 4.1.2.1:Mạch ổn áp nguồn DC

Phân tích mạch điện .

Mạch điện gồm những phần sau : Hạ áp, chỉnh lưu, lọc, biến đổi (78xx). Nguồn điện xoay chiều 220VAC-50Hz qua biến áp là hạ áp xuống còn 24VAC - 1A và được qua bộ chỉnh lưu nhằm biến đổi xoay chiều thành 1 chiều. Thành phần 1 chiều này có độ gợn nên phải qua bộ lọc C để san phẳng điện áp gợn đó cho ra điện áp 1 chiều. Sau đó điện áp 1 chiều này qua bộ ổn áp 78xx cho ra điện áp ổn áp mà mình cần.

- *Hạ áp* : Ở đây chúng ta biến đổi điện áp lưới 220VAC-50Hz xuống còn 24VAC - 1A. Mục đích là cấp đầy vào cho bộ biến đổi và bộ lọc để có điện áp một chiều mong muốn.
- *Chỉnh lưu*: Thành phần chỉnh lưu là biến đổi tín hiệu xoay chiều thành tín hiệu 1 chiều thông qua 4 condiode chỉnh lưu.

Đây là sơ đồ chỉnh lưu cả chu kì với dạng sóng đầu vào và đầu ra sau chỉnh lưu như sau:



Hình 4.1.2.2:Dạng sóng điện áp

+ Điện áp đầu vào của bộ chỉnh lưu : Uv = 24sqrt2 = 34VDC

+ Điện áp sụt áp trên cầu là : 34VDC - 1.5VDC = 32.5VDC (Do đi qua 2 diode nên mỗi đioe nó bị sụt áp mất 0.7V)

+ Điện áp sau chỉnh lưu là : Ucl = 32.5 * 0.9 = 29VDC (0.9 là hệ số chỉnh lưu của chỉnh luu cầu)

Dạng điện áp sau chỉnh lưu nó vẫn còn các sóng nhấp nhô như ngọn núi và dạng điện áp này vẫn được coi là điện áp 1 chiều nhưng chưa ổn định

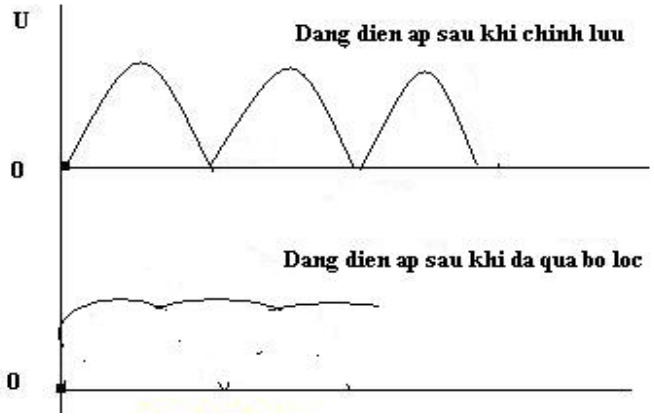
- *Thành phần lọc :*

Mạch này dùng lọc C cho đơn giản. Không có thể dùng mạch lọc RC, CRC.

+ Tụ C1 và C3 lọc các thành phần điện áp nhấp nhô sau chỉnh lưu cho nó bằng phẳng.

+ Tụ C2 và C4 lọc các thành phần cao tần

Dạng điện áp sau khi qua bộ lọc

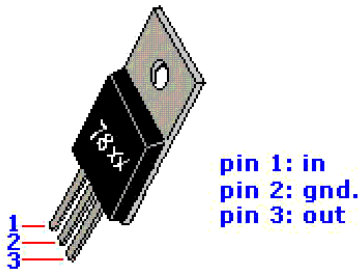


Hình 4.1.2.3:Dạng sóng điện áp sau khi lọc

Dựa vào nguyên tắc phóng nạp của tụ điện mà nó cho ra dòng điện 1 chiều thằng như trên hình vẽ. Tụ càng lớn thì độ gợn điện áp càng giảm. Những sóng có tần số cao tần phải được lọc đi nhờ 2 tụ kẹo C2 và C4 vì trong mạch dùng IC nếu tồn tại những thành phần này thì sẽ gây ra những sai sót khó phát hiện làm cho mạch hoạt động không bình thường.

Qua bộ lọc là ta đã tạo được điện áp 1 Chiều cấp vào cho bộ biến đổi đổi hay là bộ ổn áp

- Bộ ổn áp



Hình 4.1.2.4:Hình dáng bên ngoài của 78xx

+ Dòng họ 78xx cho ra nhiều loại ổn áp điện khác nhau : như 7805 nó ổn áp 5V, 7806 cho ổn áp 6V...

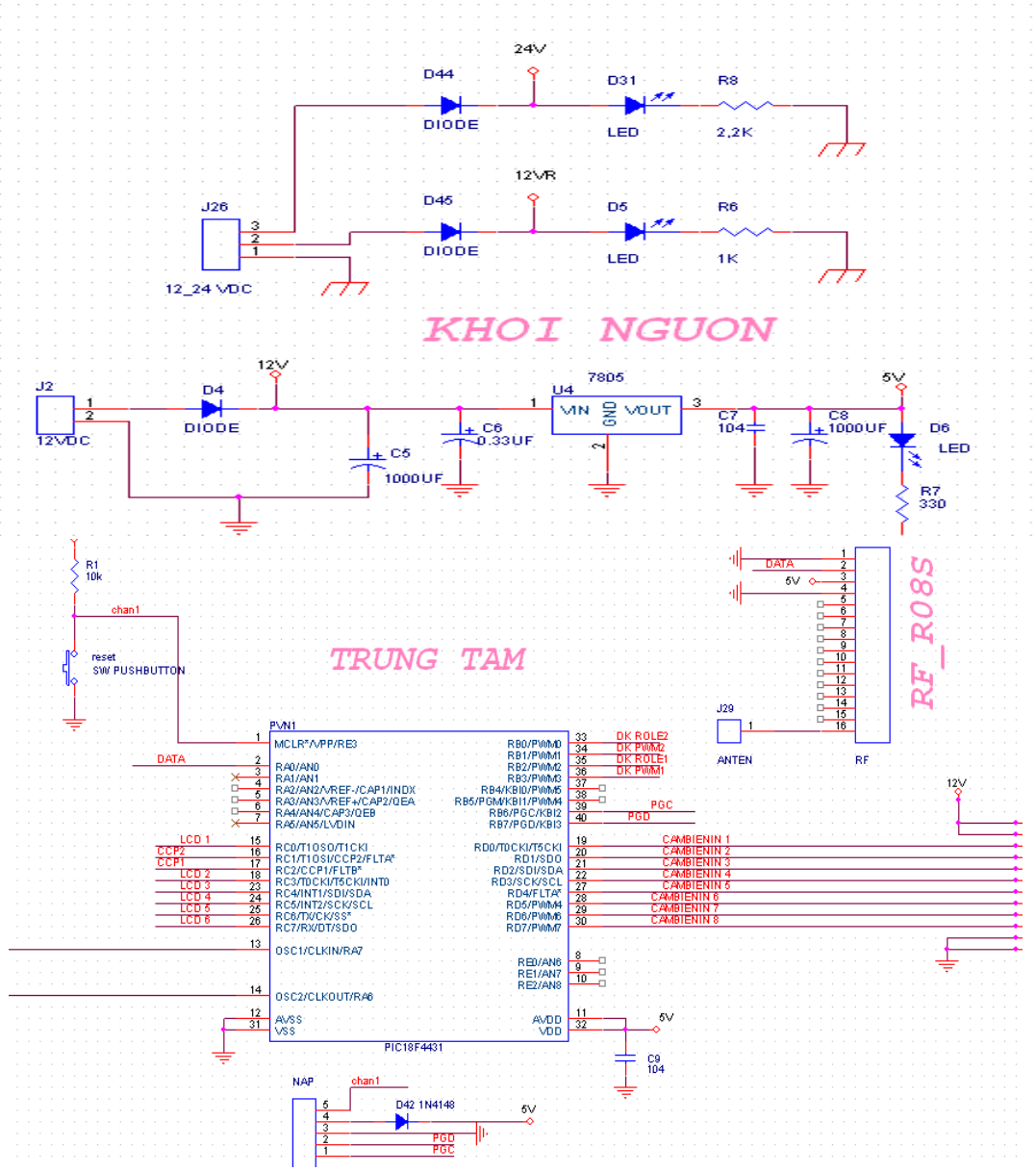
+ Điện áp đầu vào của họ 78xx là điện áp 1 chiều và max <=40V. Dòng điện không vượt quá 1A

+ Đảm bảo thông số là : Vi - V0 = 2V đến 3V (lúc đó mạch mới hoạt động ổn áp được)

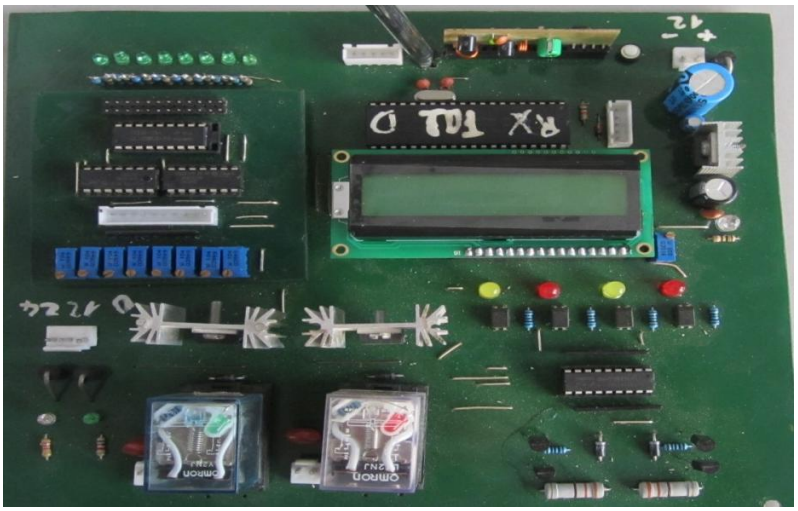
+ Tản nhiệt tốt cho 78xx. Khi hoạt động với tải thì 78xx rất nóng . Đối với cấp

điện áp là 29V thì 78xx nóng khi có tải và chú ý tản nhiệt tốt cho nó.

4.1.3. Thiết kế khối điều khiển



Hình 4.1.3.1:Sơ đồ nguyên lý khối điều khiển



Hình 4.1.3.2:Mạch điều khiển

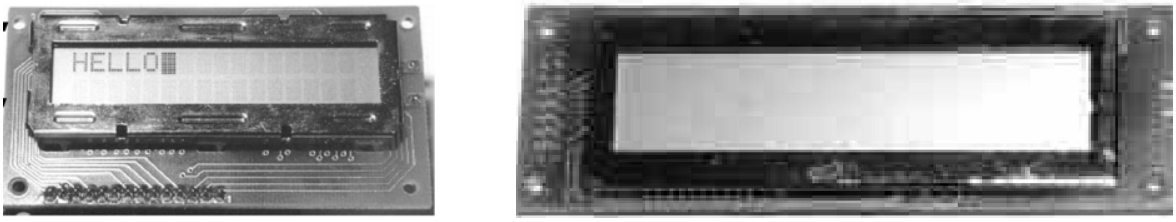
Để điều khiển và xử lý mọi thông tin và đưa lệnh cho cơ cấu chấp hành làm việc thì dùng vi điều khiển để lập trình xử lý.Ở đây em dùng PIC để xử lý.Sơ đồ mạch được thiết kế như sau.

4.1.4. Thiết kế khối hiển thị

Để hiện thị các thông tin điều khiển thì ta có thể sử dụng nhiều phương án để hiển thị có thể dùng led 7seg hay LCD,led Matrix….Ở đây em dùng LCD để hiển thị thông tin điều khiển.

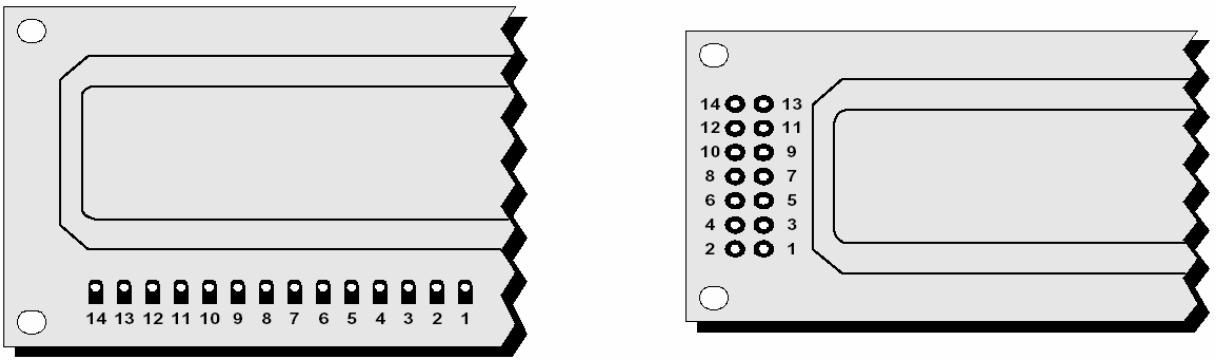
Ngày nay, thiết bị hiển thị LCD (Liquid Crystal Display) được sử dụng trong rất nhiều các ứng dụng của VĐK. LCD có rất nhiều ưu điểm so với các dạng hiển thị khác: Nó có khả năng hiển thị kí tự đa dạng, trực quan (chữ, số và kí tự đồ họa), dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tốn rất ít tài nguyên hệ thống và giá thành rẽ …

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



Hình 4.1.4.1: Hình dáng của hai 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 *4.1.4.2* :



Hình 4.1.4.2 : Sơ đồ chân của LCD

Chức năng các chân

Chân	Tên	Chức năng
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	Chân này dùng để 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. + Logic "0": Bus DB0-DB7 sẽ nối với thanh ghi lệnh IR của LCD (ở chế độ "ghi" - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ "đọc" - read) + Logic "1": Bus DB0-DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD.
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.

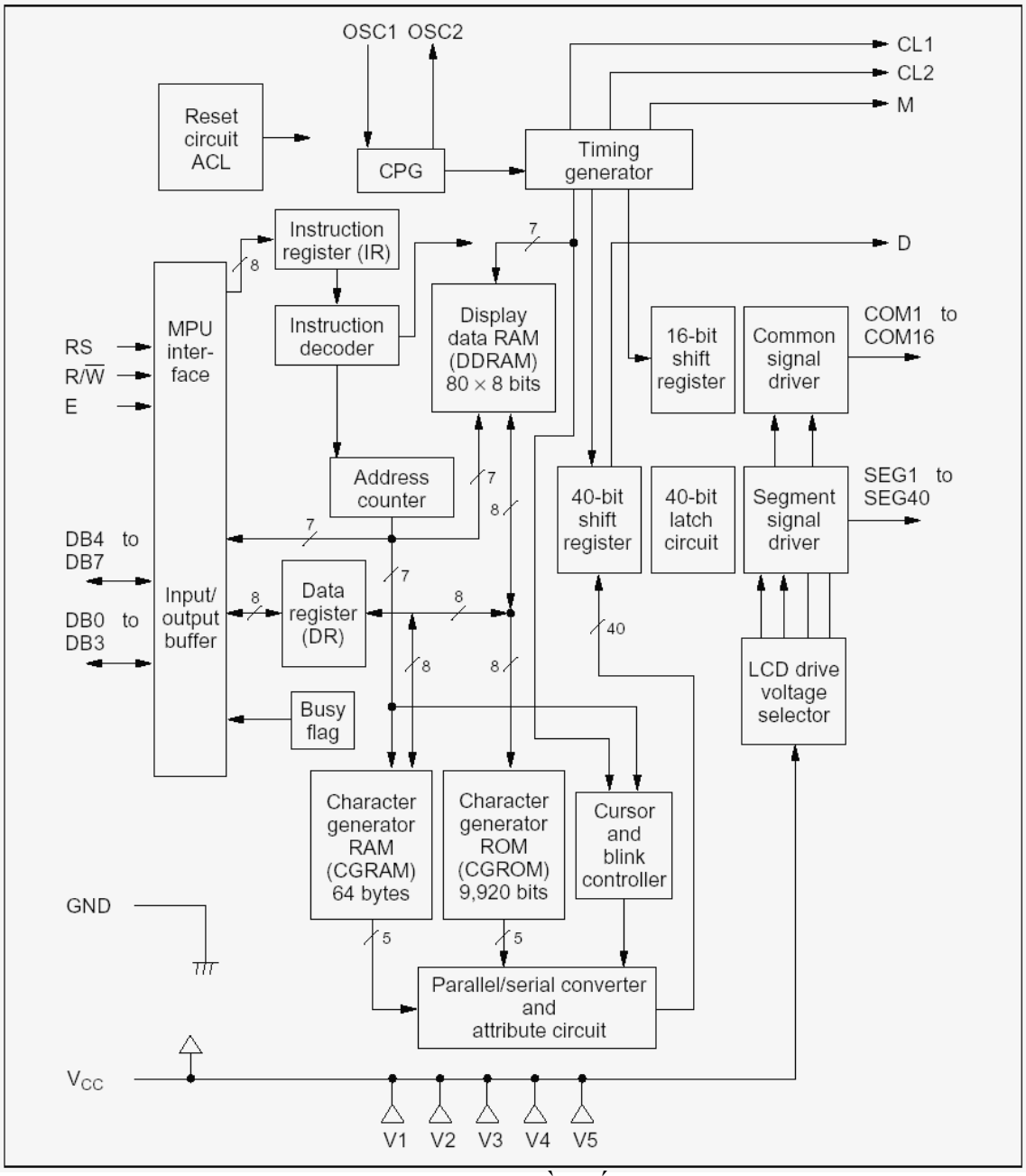
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E. + Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào(chấp nhận) thanh ghi bên trong nó khi phát hiện một xung (high-to-low transition) của tín hiệu chân E. + Ở chế 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.
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 : + Chế độ 8 bit : Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + Chế độ 4 bit : Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7

Bảng 4.1.4.1 : Chức năng các chân của LCD

* 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.

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 4.1.4.3 : Sơ đồ khối của HD44780

Các thanh ghi

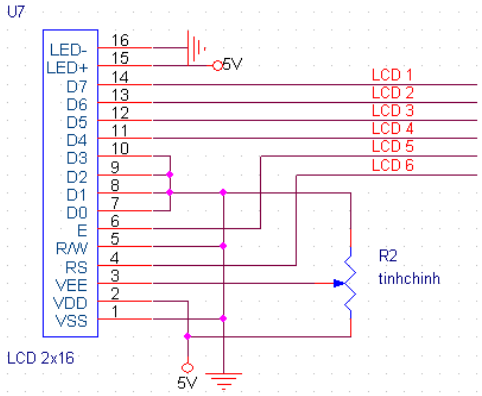
Chíp HD44780 có 2 thanh ghi 8 bit quan trọng : Thanh ghi lệnh IR (Instructor Register) và thanh ghi dữ liệu DR (Data Register)

- 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 địachỉ 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

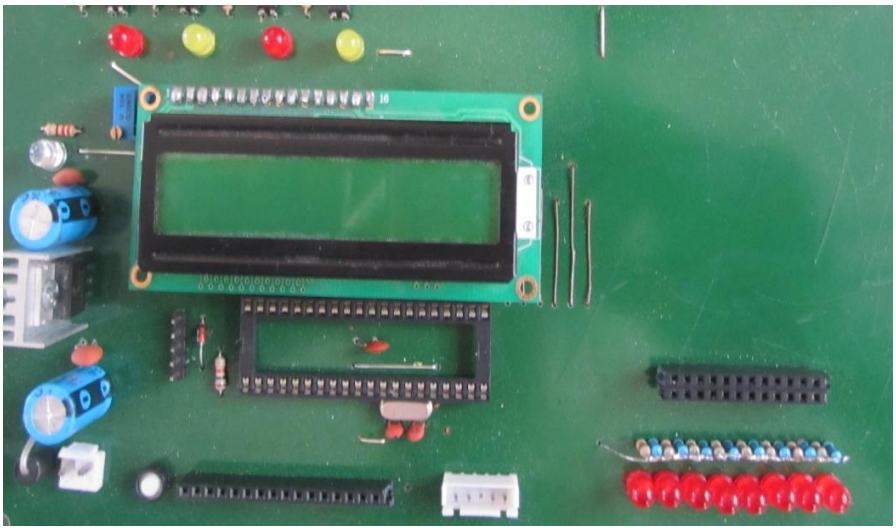
cấp và thực hiện lệnh đó.
VD : Lệnh "hiển thị màn hình" có địa chỉ lệnh là 00001100 (DB7…DB
Lệnh "hiển thị màn hình và con trỏ" có mã lệnh là 00001110
- Thanh ghi DR : Thanh ghi DR dùng để chứa dữ liệu 8 bit để ghi vào vùng RAM DDRAM hoặc CGRAM (ở chế độ ghi) hoặc dùng để chứa dữ liệu từ 2 vùng RAM này 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.
□Bằng cách điều khiển chân RS và R/W chúng ta có thể chuyển qua lại giữ 2 thanh ghi này khi giao tiếp với MPU. Bảng sau đây tóm tắt lại các thiết lập đối với hai chân RS và R/W theo mục đích giao tiếp.

RS	R/W	Khi cần
0	**0**	Ghi vào thanh ghi IR để ra lệnh cho LCD (VD: cần display clear,…)
0	**1**	Đọc cờ bận ở DB7 và giá trị của bộ đếm địa chỉ ở DB0-
1	**0**	Ghi vào thanh ghi DR
1	**1**	Đọc dữ liệu từ DR

Bảng 4.1.4.2 : Chức năng chân RS và R/W theo mục đích sử dụng



Hình 4.1.4.4 : Sơ đồ mạch hiển th



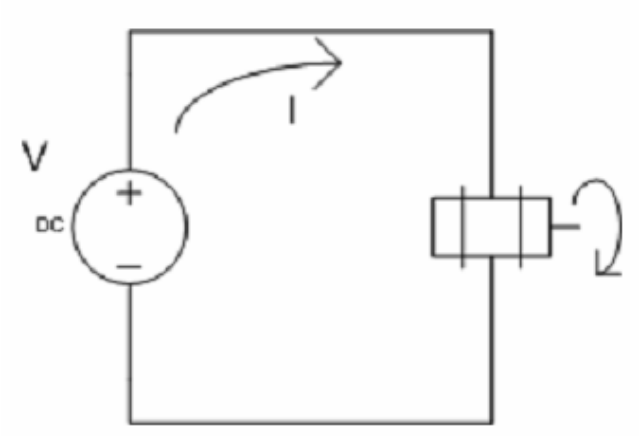
Hình 4.1.4.5 : Mạch hiển thị LCD

4.1.5. Thiết kế khối công suất

Hệ thống truyền động là hệ thống rất quan trọng, roobot di chuyển được nhờ hệ thống truyền động này, mà chủ yếu là hệ thống truyền động cho động cơ DC.

Động cơ DC có rất nhiều loại, mỗi loại có một đặc tính riêng về tần số cũng như tốc độ dẫn đến việc điều khiển chúng cũng khác nhau.Mặc dầu khác nhau về kích thước,kiểu dáng, màu sắc nhưng nhìn chung có 3 loại cơ bản:DC motor, Stepper Motor, Servo Motor.

Nhưng ở đây chỉ dùng động cơ DC .Cấu tạo về cơ bản gồm bộ phận đứng yên có nam châm vĩnh cửu được gọi là stator,bộ phận chuyển động được gọi là roto.



Hình 4.1.5.1 : Sơ đồ khối mạch DC motor

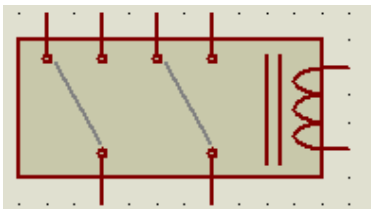
Đặc tính kỹ thuật chủ yếu là moment ngẫu lực thấp và tốc độ quay cao,nhưng mà hầu hết robot chạy với tốc độ thấp và moment ngẫu lực cao.Do đó trên động cơ DC ta lắp bộ

phận để giảm tốc độ,bộ phận này được gọi là hộp số,nhằm giảm tốc độ quay và tăng moment ngẫu lực.

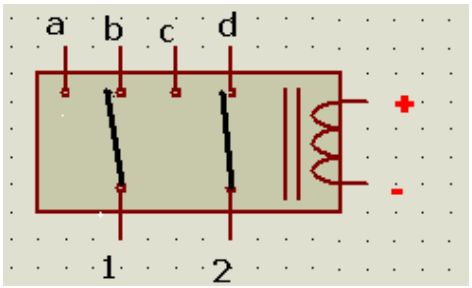
Để điều khiển tốc độ DC thì có nhiều phương pháp điều khiển như dùng cầu H,hay kết hợp rơle và feet.Ở đây em dùng feet kết hợp với rơle để điều khiển chiều quay và tốc độ động cơ DC.

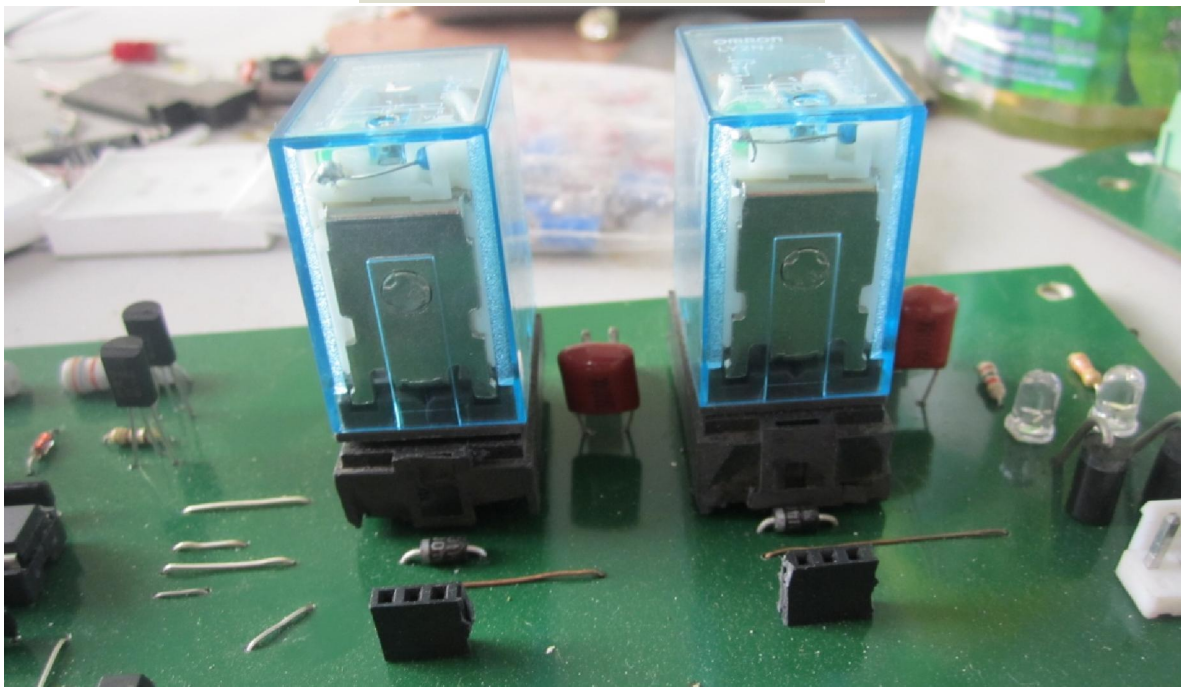
Role là thiết bị điện chuyên đóng cắt mạch điện,gồm có cuộn dây và hệ thống tiếp điểm thường kín và hở.Và em chọn role ORMROM để sử dụng trong việc đảo chiều động cơ.

Sơ đồ nguyên lý rơle.



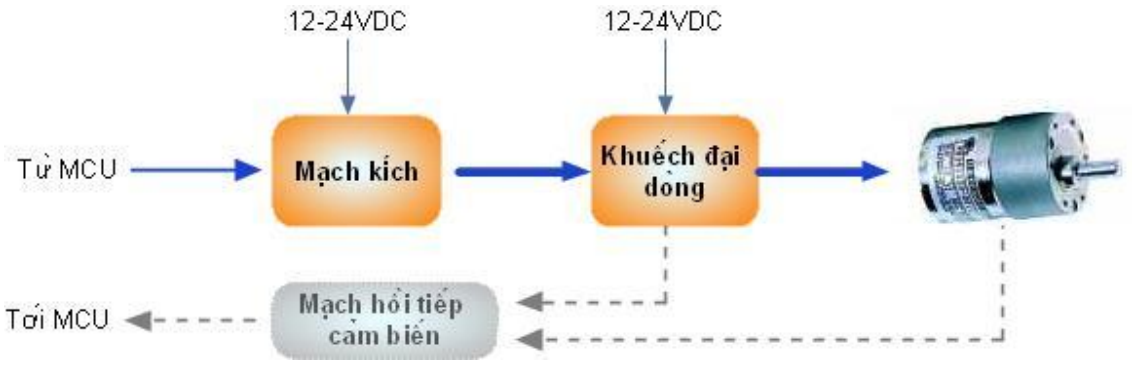
Nguyên lý hoạt động:Khi có dòng điện chạy trong cuộn dây sẽ sinh ra lực điện động đủ lớn để hút các tiếp của role.



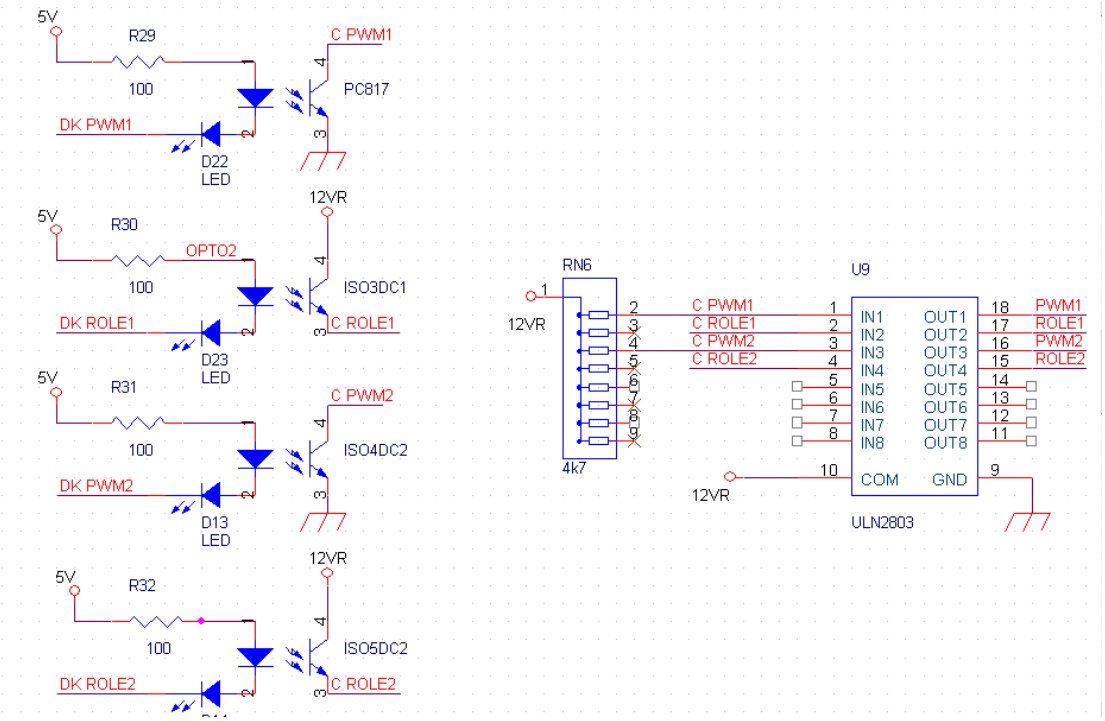


Hình 4.1.5.2 : Khối role

Thiết kế mạch điều khiển động cơ:



Hình 4.1.5.3 : Sơ đồ khối mạch điều khiển



Hình 4.1.5.4 : Sơ đồ nguyên lý mạch cách ly

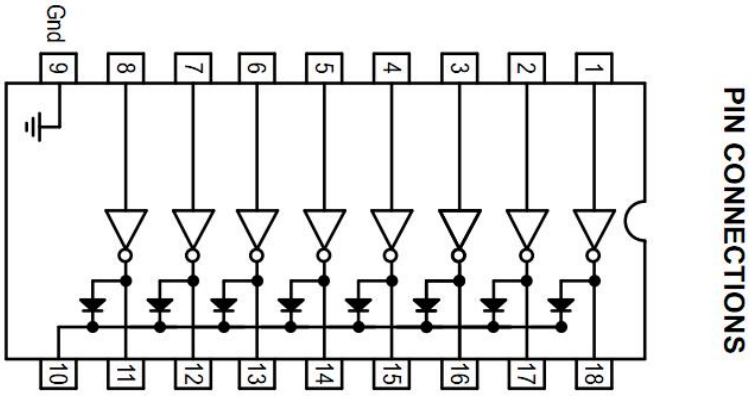
Tính toán thông số trong mạch:

+ Dòng đi vào PIC (chân DKPWM) thường là 10mA,để led D22 sáng bình thường thì áp 2-3 V cộng với áp trên optoPC817 là 1.2-1.4 V => tính được giá trị điện trở R29 như sau:

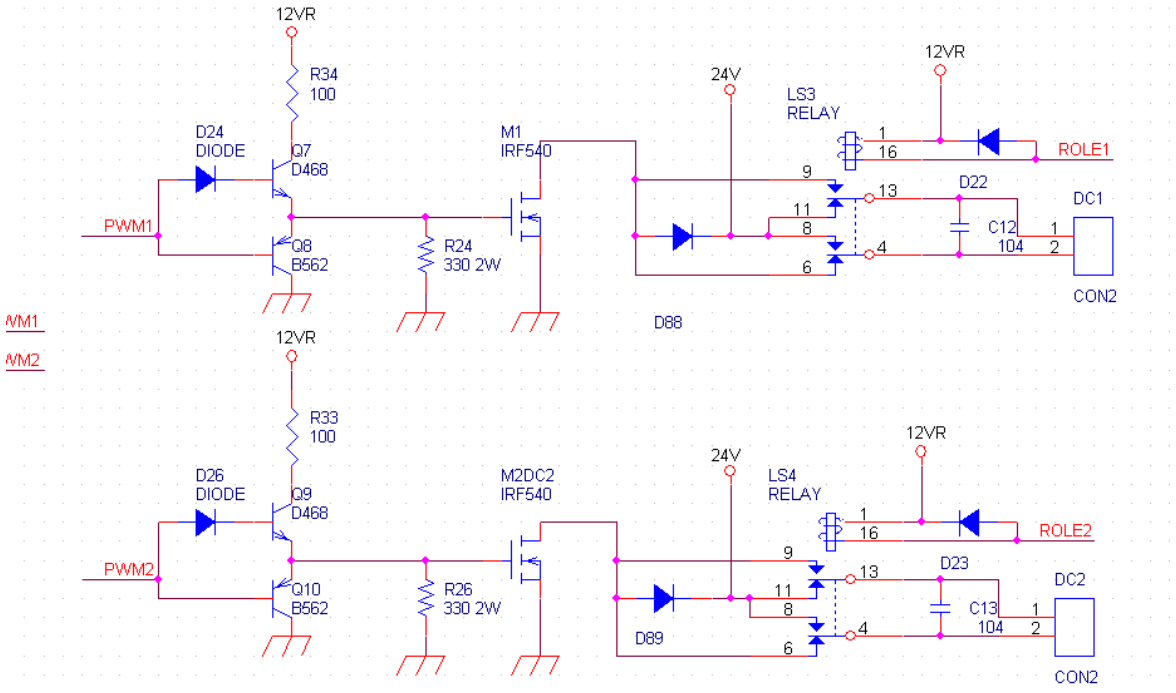
$$R22 = \frac{5-1,2-2,8}{10}.1000 = 100\ \Omega$$

+ Opto PC817 có nhiệm vụ cách ly mạch động lực và mạch điều khiển.ULN2803 là cổng đảo, có nhiệm vụ đảo tín hiệu cho phù hợp để điều khiển mạch lực.Sơ đồ được mắc theo datasheet .Cấu trúc bên trong như hình dưới.

+ Tín hiệu sau khi qua ULN2803 sẽ kích cho mở transtor và cấp điện cho cuộn dây của role.



Hình 4.1.5.5 : Sơ đồ khối ULN2803



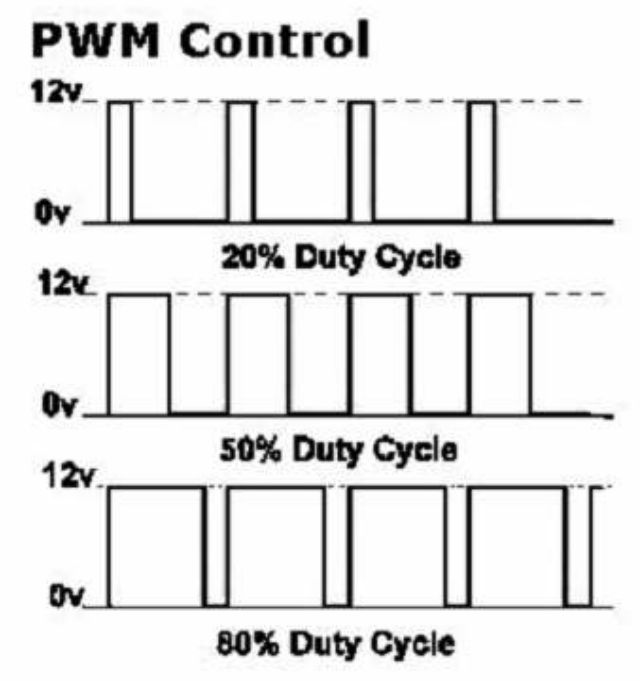
Hình 4.1.5.6 : Sơ đồ nguyên lý mạch công suất

Mạch kích dùng hai transitor pnp và npn mắc E chung cùng với một số linh kiện phụ để làm cho mạch ổn định hơn.

Diot D24 và D26 là hai con diot mắc như vậy để tránh hiện tượng trùng dẫn, con điện trở công suất 2W R24,R26 330Ω là điện trở dùng để xả điện áp tàn dư khi kích giúp Feet đóng cắt tốt hơn vì trong khi đóng cắt với tần số cao.

D88 và D89 là hai con diot chóng dòng ngược cho feet .

Để điều khiển tốc độ của động cơ thì ta chỉ cần thay đổi điện áp của DC,sử dụng phương pháp thay đổi độ rộng xung,điều xung PWM .

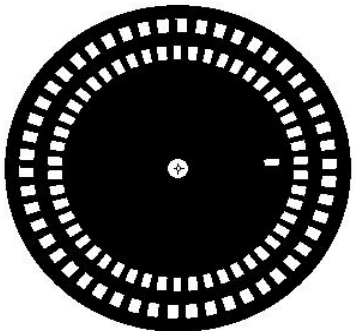


Hình 4.1.5.7 : Sóng ra PWM

Ứng với một giá trị của Duty sẽ có một tốc độ nhất định,giá trị duty này được thay đổi ở trong chương trình.

4.1.6. Encoder và ứng dụng

Encoder mục đích dùng để quản lý vị trí góc của một đĩa quay, đĩa quay có thể l à bánh xe, trục động cơ, hoặc bất kỳ thiết bị quay nào cần xác định vị trí góc. Đĩa quay được chia theo 1/4 vòng,1/8 vòng, hoặc 1/n vòng, tùy theo số lỗ nằm trên encoder



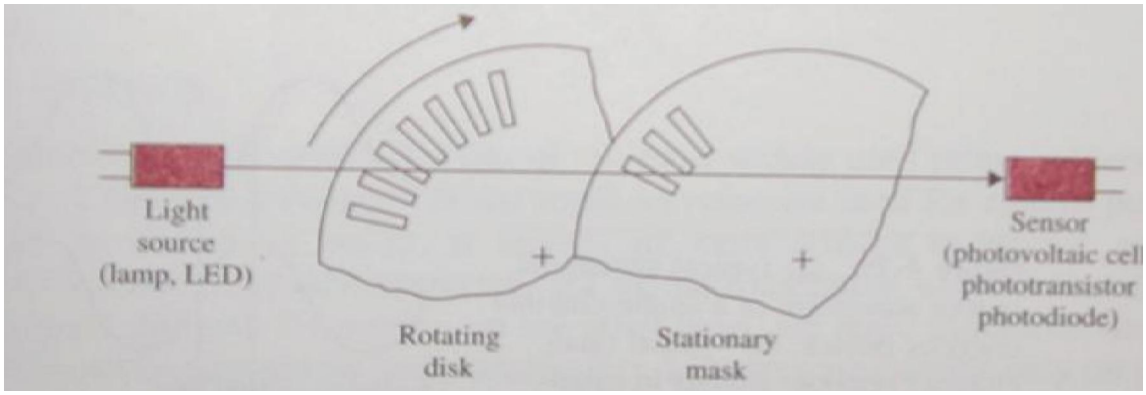
Hình 4.1.6.1 : Cấu tạo đĩa encoder

Nguyên lý hoạt động cơ bản của encoder, Led và lỗ.

Nguyên lý cơ bản của encoder, đó là một đĩa tròn xoay,quay quanh trục.Trên đĩa có các lỗ (rãnh).Người ta dùng một đèn led để chiếu lên mặt đĩa.Khi đĩa quay, chỗ không có lỗ (rãnh),đèn led không chiếu xuyên qua được,chỗ có lỗ(rãnh),đèn led sẽ chiếu xuyên qua. Khi đó, phía mặt bên kia của đĩa, người ta đặt một con mắt thu.Với các tín hiệu có, hoặc không có ánh sáng chiếu qua,người ta ghi nhận được đèn led có chiếu qua lỗ hay không.

Khi trục quay,giả sử trên đĩa chỉ có một lỗ duy nhất, cứ mỗi lần con mắt thu nhận được tín hiệu đèn led, thì có nghĩa là đĩa đã quay được một vòng.Đây là nguyên lý rất cơ bản của encoder.

Tuy nhiên,những vấn đề được đặt ra là,làm sao để xác định chính xác hơn vị trí của đĩa quay (mịn hơn) và làm thế nào để xác định được đĩa đang quay theo chiều nào?Đó chính là vấn đề để chúng ta tìm hiểu về encoder. Hình sau sẽ minh họa nguyên lý cơ bản của hoạt động encoder.

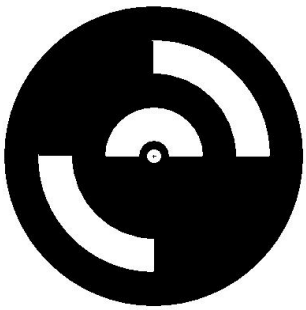


Hình 4.1.6.2: Nguyên lý hoạt động của encoder

Ta thấy trong hình trên,có một đĩa mask, không quay, đó là đĩa cố định, thực ra là để che khe hẹp ánh sáng đi qua, giúp cho việc đọc encoder được chính xác hơn mà thôi.Chúng ta không đề cập đến đĩa mặt nạ này ở đây.

Vấn đề chúng ta sẽ quan tâm ở đây,chính là vấn đề về độ mịn của encoder, có nghĩa là làm thế nào biết đĩa đã quay 1/2 vòng, 1/4 vòng, 1/8 vòng hay 1/n vòng, chứ không phải chỉ biết đĩa đã quay được một vòng.

Quay lại bài toán cơ bản về bit và số bit, chúng ta xem xét vấn đề theo một cách hoàn toàn toán học .Với một số nhị phân có 2 chữ số, chúng ta sẽ có 00, 01, 10, 11, tức là 4 trạng thái. Điều đó có nghĩa là với 2 chữ số, chúng ta có thể chia đĩa encoder thành 4 phần bằng nhau.Và khi quay,chúng ta sẽ xác định được độ chính xác đến 1/4 vòng.Tương tự như vậy, nếu với một số có n chữ số, chúng ta sẽ xác định được độ chính xác đến 1/(2^n) vòng.



Hình 4.1.6.3 : đĩa encoder

Ở đây, em đưa ra ví dụ với đĩa encoder có 2 vòng đĩa. Chúng ta sẽ thấy rằng, ở vòng trong cùng, có một rãnh rộng bằng 1/2 đĩa. Vòng phía ngoài, sẽ có 2 rãnh nằm đối diện nhau.Như vậy, chúng ta cần 2 đèn led để phát xuyên qua 2 vòng lỗ, và 2 đèn thu. Giả sử ở vòng lỗ thứ nhất (trong cùng), đèn đọc đang nằm ở vị trí có lỗ hở, thì tín hiệu nhận được từ con mắt thu sẽ là 1. Và ở vòng lỗ thứ hai, thì chúng ta đang ở vị trí không có lỗ, như vậy con mắt thu vòng 2 sẽ đọc được giá trị 0.Và như vậy, với số 10, chúng ta xác định được encoder đang nằm ở góc phần tư nào, cũng có nghĩa là chúng ta quản lý được độ chính xác của đĩa quay đến 1/4 vòng.Trong ví dụ trên, nếu đèn Led đọc được 10, thì vị trí của LED phải nằm trong góc phần tư thứ hai,phía trên,bên trái.Kết quả, nếu đĩa encoder có đến 10 vòng lỗ, thì chúng ta sẽ quản lý được đến 1/(2^10) tức là đến 1/1024 vòng.Hay người ta nói là độ phân giải của encoder là 1024 xung trên vòng (pulse per revolution - ppr).

Ứng dụng encoder trong trong việc đo khoảng cách đường đi của Robot.

Hình 4.1.6.4 : Hình dáng bên ngoài của encoder

Trong Robocon, ta thường sử dụng incremental encoder (encoder tương đối) hay còn gọi là rotary encoder. Mục đích của việc sử dụng encoder trong robot là đếm số vòng quay để tính số vòng quay của động cơ (bánh xe), từ đó suy ra quãng đường di chuyển và và có hướng điều chỉnh tốc độ của robot.

Sử dụng PIC để nhận và đếm xung từ encoder

Để nhận xung từ encoder, ta có thể sử dụng ngắt ngoài, ngắt timer hoặc đơn giản là tham dò mức logic của các chân vi điều khiển một cách liên tục. Phần sau đây giới thiệu cách nhận và đếm xung của PIC16F877A dùng ngắt ngoài B0 (nối với kênh A của encoder) và chân B1 (nối với kênh B của encoder). Ta có thể làm tương tự đối với các cách nhận xung khác.

Khởi tạo ngắt ngoài theo cạnh lên tại chân B0:

Code:

```
ext_int_edge(0,L_TO_H);    // Ngắt cạnh lên tại RB0
enable_interrupts(INT_EXT); // Cho phép ngắt ngoài
enable_interrupts(GLOBAL);   // Cho phép ngắt toàn cục
```

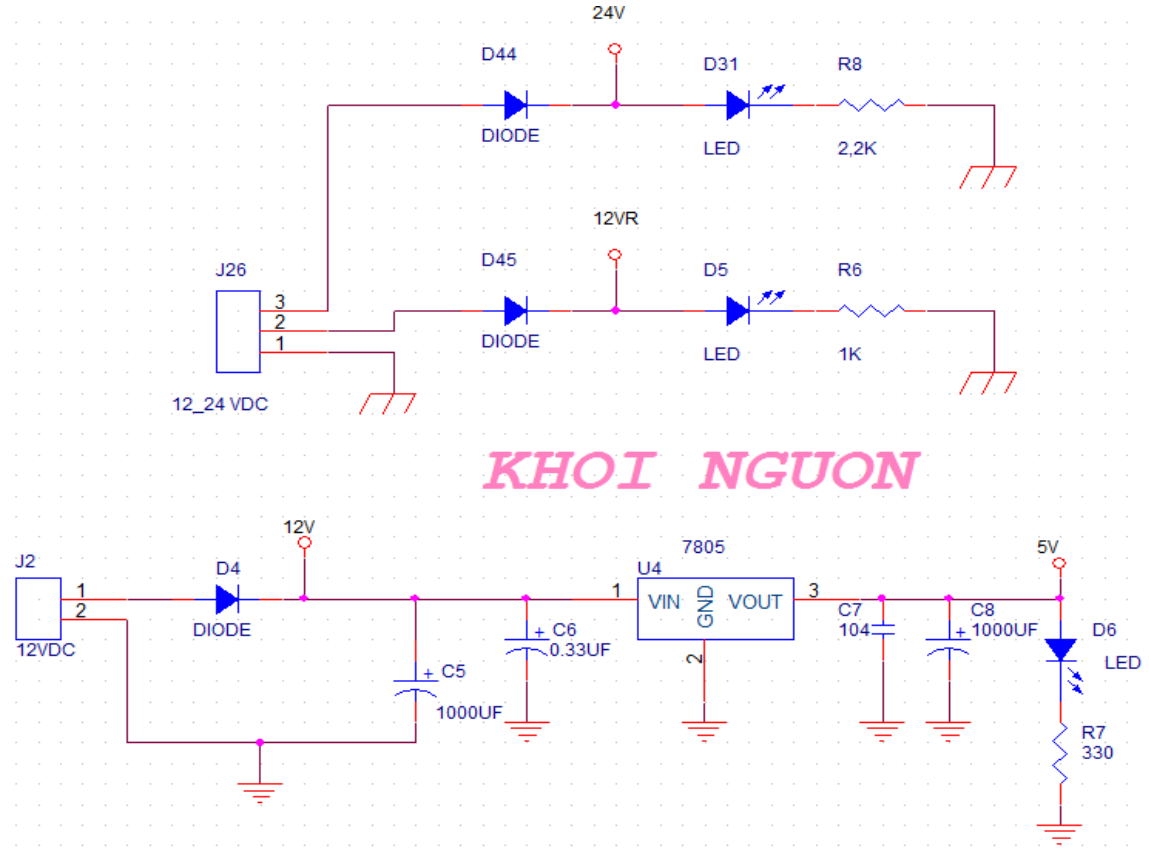
Chương trình con phục vụ ngắt:

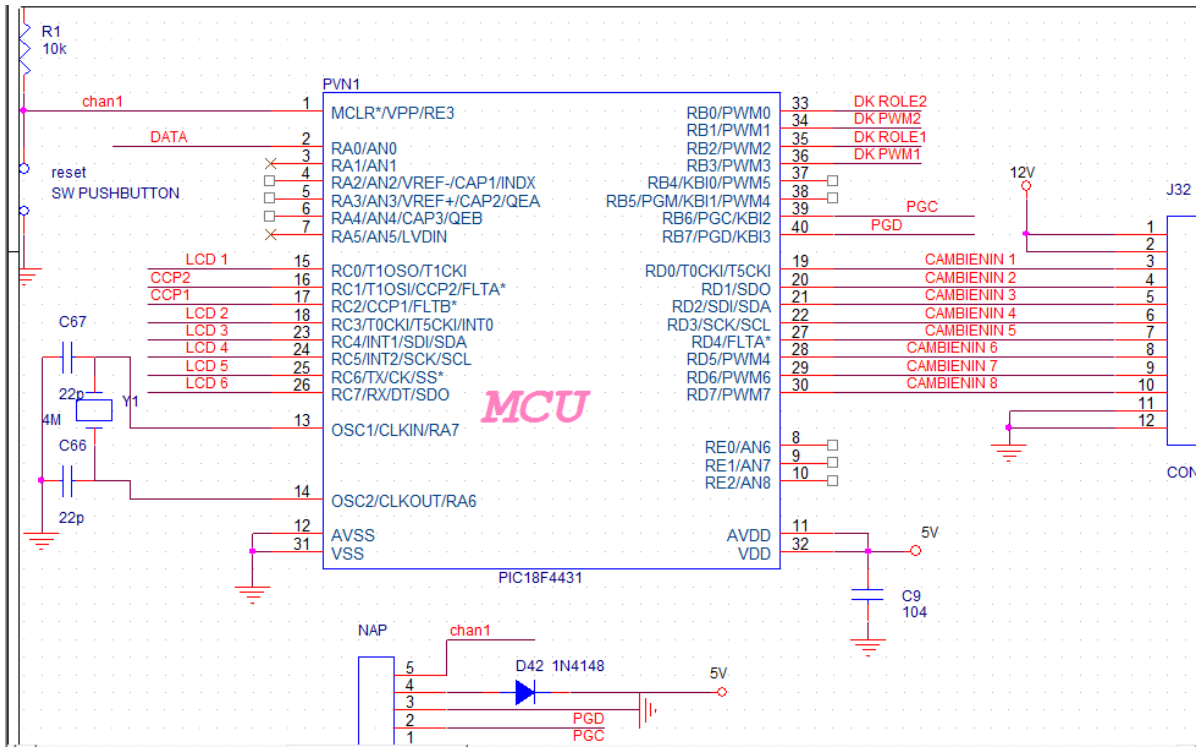
Code:

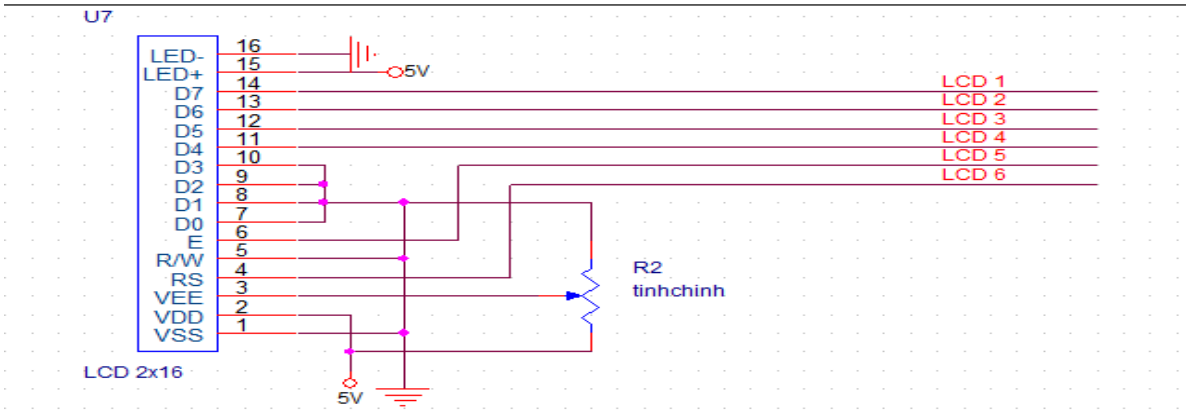
```
#int_EXT
void  EXT_isr(void) //Chương trình được gọi khi có tác động cạnh lên tại chân B0
{
if (RB1==1) pulse++;                    // Nếu kênh B mức cao thì tăng giá trị xung thêm 1
else pulse--;                           // Nếu kênh B mức cao thì giảm giá trị xung xuống
1
}
```

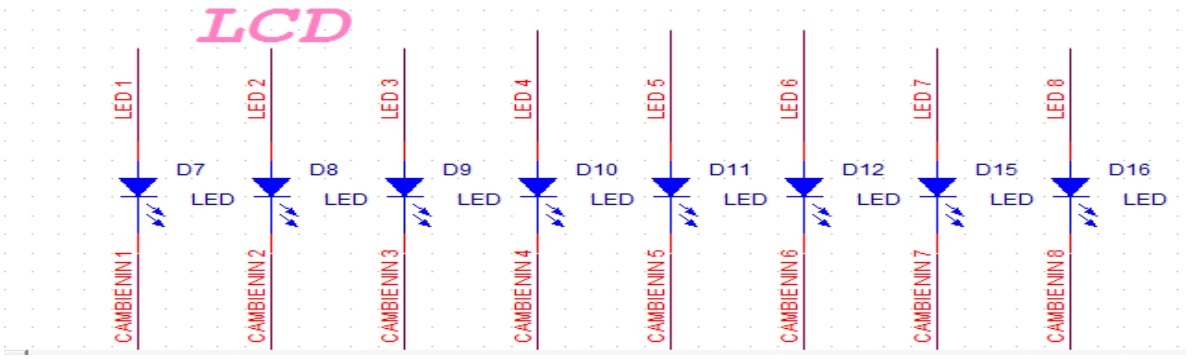
Từ giá trị xung tính được tại các thời điểm ta có thể tính ra các thông số mong muốn.

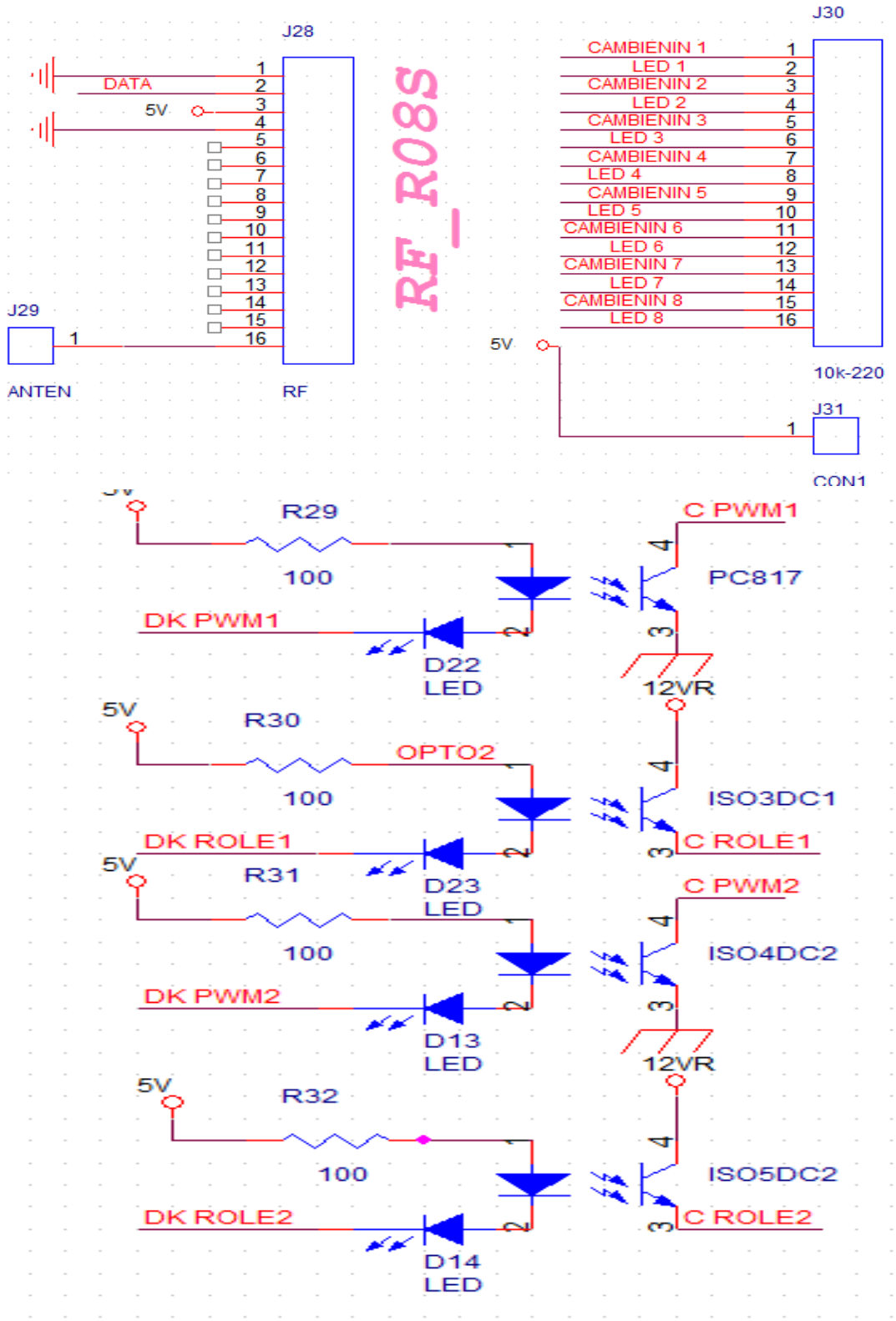
4.1.7. Kết quả đạt được.

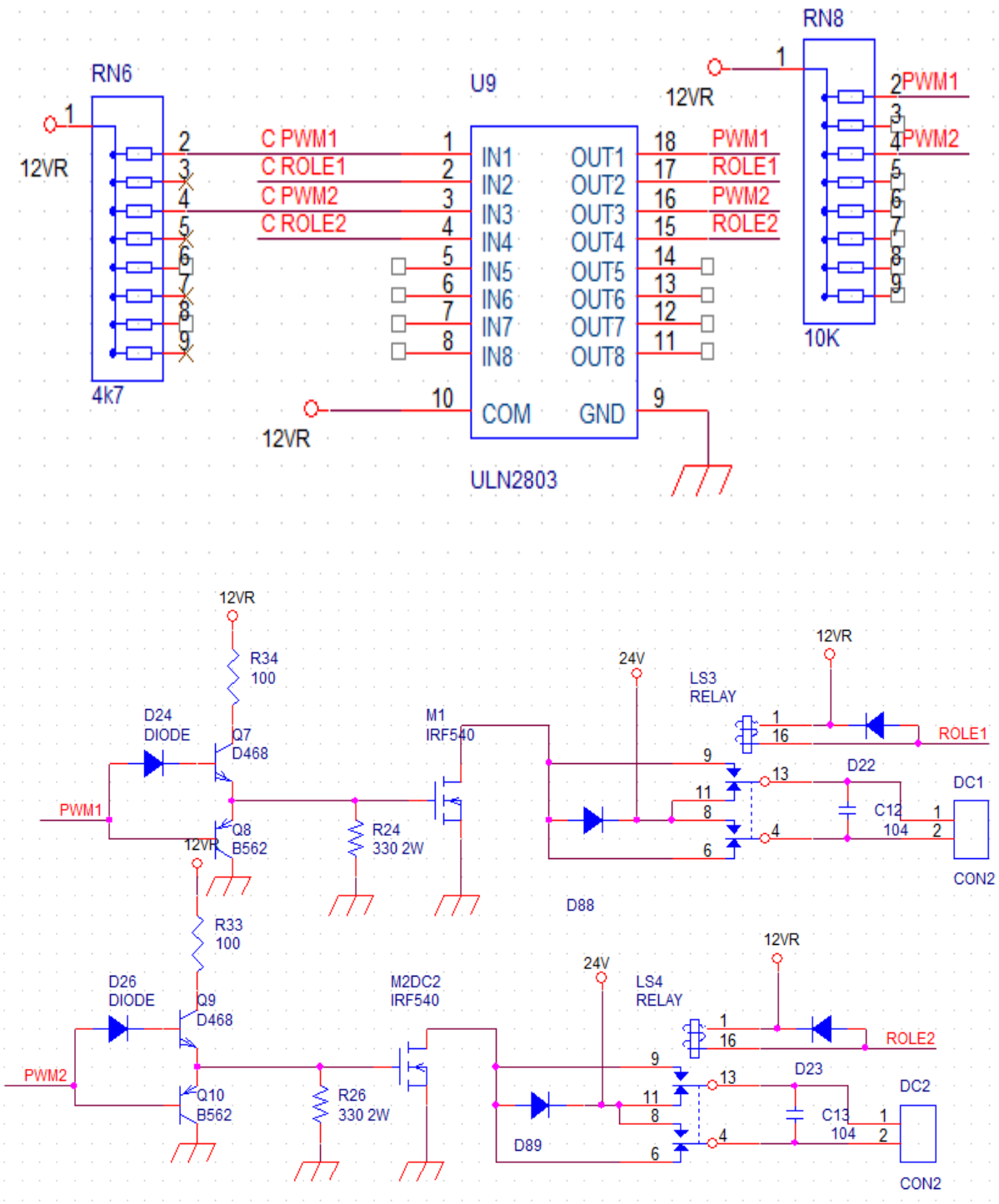



R1
10k
PVN1
chan1
DATA
reset
SW PUSHBUTTON
MCLR*/VPP/RE3
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CAP1/INDX
RA3/AN3/VREF+/CAP2/QEA
RA4/AN4/CAP3/QEB
RA5/AN5/LVDIN
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2/FLTA*
RC2/CCP1/FLTB*
RC3/T0CKI/T5CKI/INT0
RC4/INT1/SDI/SDA
RC5/INT2/SCK/SCL
RC6/TX/CK/SS*
RC7/RX/DT/SDO
OSC1/CLKIN/RA7
OSC2/CLKOUT/RA6
AVSS
VSS
RB0/PWM0
RB1/PWM1
RB2/PWM2
RB3/PWM3
RB4/KBI0/PWM5
RB5/PGM/KBI1/PWM4
RB6/PGC/KBI2
RB7/PGD/KBI3
RD0/T0CKI/T5CKI
RD1/SDO
RD2/SDI/SDA
RD3/SCK/SCL
RD4/FLTA*
RD5/PWM4
RD6/PWM6
RD7/PWM7
RE0/AN6
RE1/AN7
RE2/AN8
AVDD
VDD
PIC18F4431
MCU
LCD 1
CCP2
CCP1
LCD 2
LCD 3
LCD 4
LCD 5
LCD 6
C67
22p
Y1
4M
C66
22p
DK ROLE2
DK PWM2
DK ROLE1
DK PWM1
PGC
PGD
12V
J32
CAMBIENIN 1
CAMBIENIN 2
CAMBIENIN 3
CAMBIENIN 4
CAMBIENIN 5
CAMBIENIN 6
CAMBIENIN 7
CAMBIENIN 8
5V
C9
104
NAP
chan1
D42 1N4148
5V
PGD
PGC

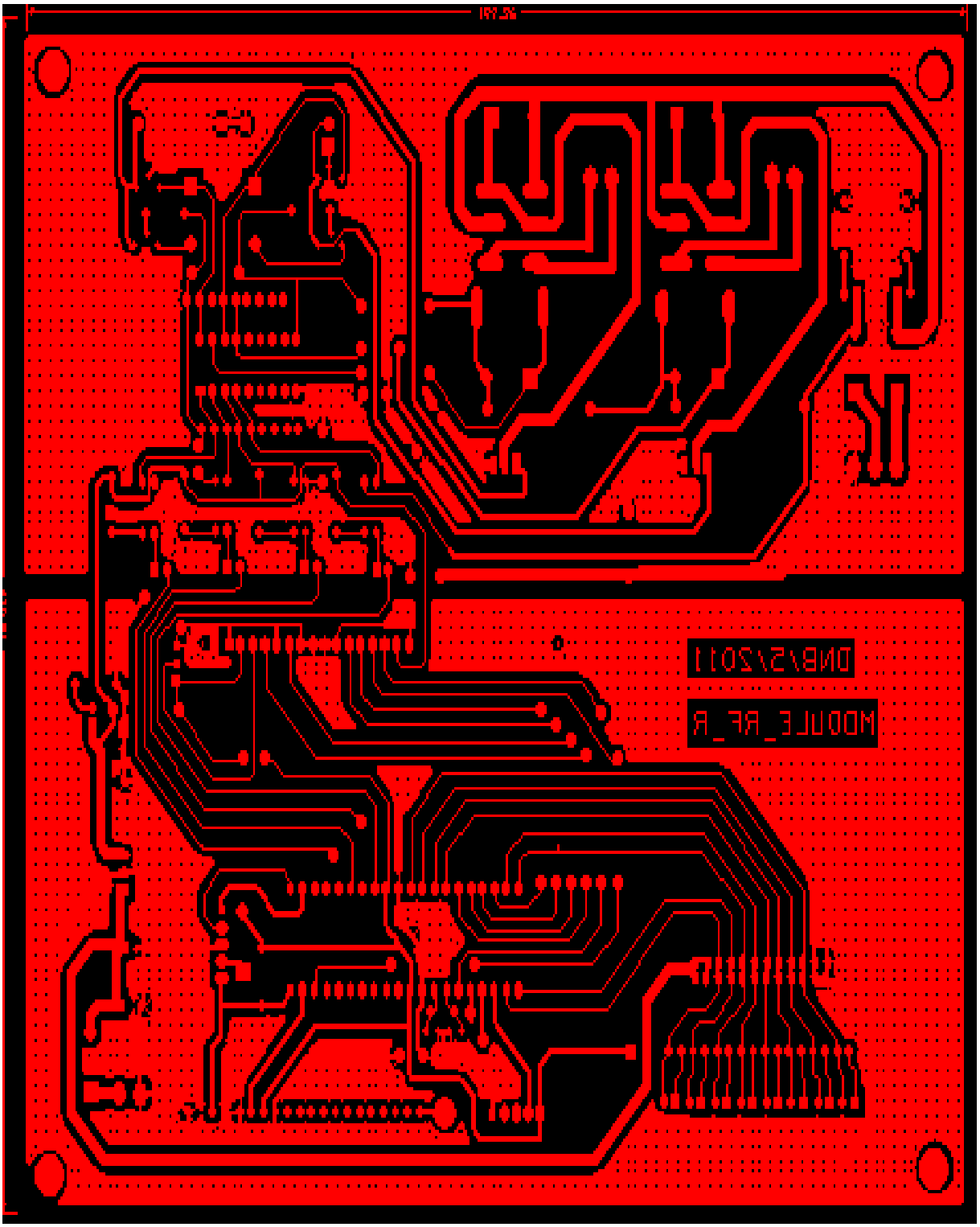

U7
LED-
LED+
D7
D6
D5
D4
D3
D2
D1
D0
E
R/W
RS
VEE
VDD
VSS
5V
LCD 1
LCD 2
LCD 3
LCD 4
LCD 5
LCD 6
R2
tinhchinh
LCD 2x16
5V


LCD
LED 1
LED 2
LED 3
LED 4
LED 5
LED 6
LED 7
LED 8
D7 LED
D8 LED
D9 LED
D10 LED
D11 LED
D12 LED
D15 LED
D16 LED
CAMBIENIN 1
CAMBIENIN 2
CAMBIENIN 3
CAMBIENIN 4
CAMBIENIN 5
CAMBIENIN 6
CAMBIENIN 7
CAMBIENIN 8

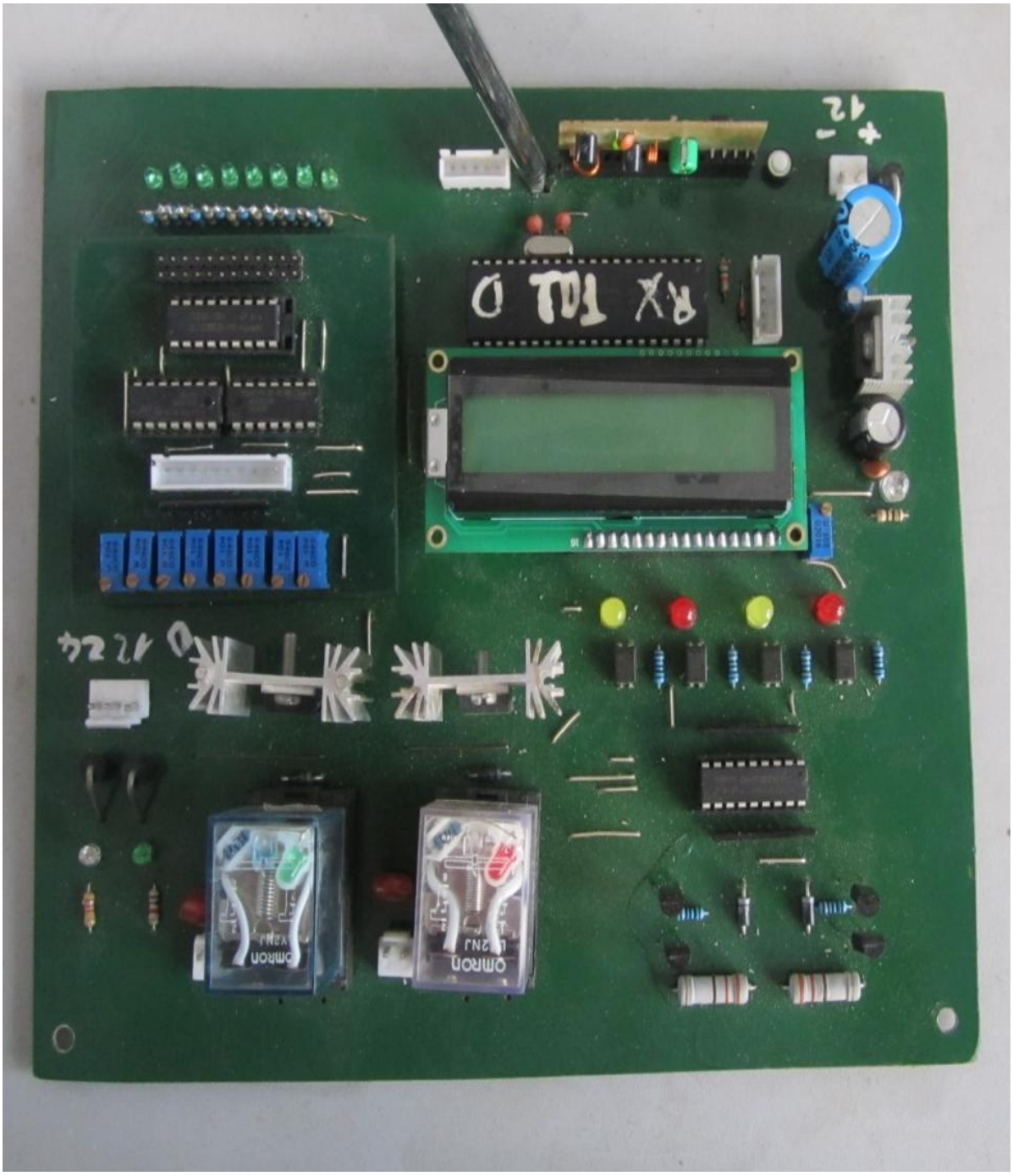

J28
DATA
5V
J29
ANTEN
RF
RF_R08S
J30
CAMBIENIN 1
LED 1
CAMBIENIN 2
LED 2
CAMBIENIN 3
LED 3
CAMBIENIN 4
LED 4
CAMBIENIN 5
LED 5
CAMBIENIN 6
LED 6
CAMBIENIN 7
LED 7
CAMBIENIN 8
LED 8
5V
10k-220
J31
CON1
R29
100
DK PWM1
D22
LED
C PWM1
PC817
12VR
R30
OPTO2
ISO3DC1
DK ROLE1
C ROLE1
R31
D23
LED
C PWM2
ISO4DC2
DK PWM2
D13
LED
R32
ISO5DC2
DK ROLE2
C ROLE2
D14
LED



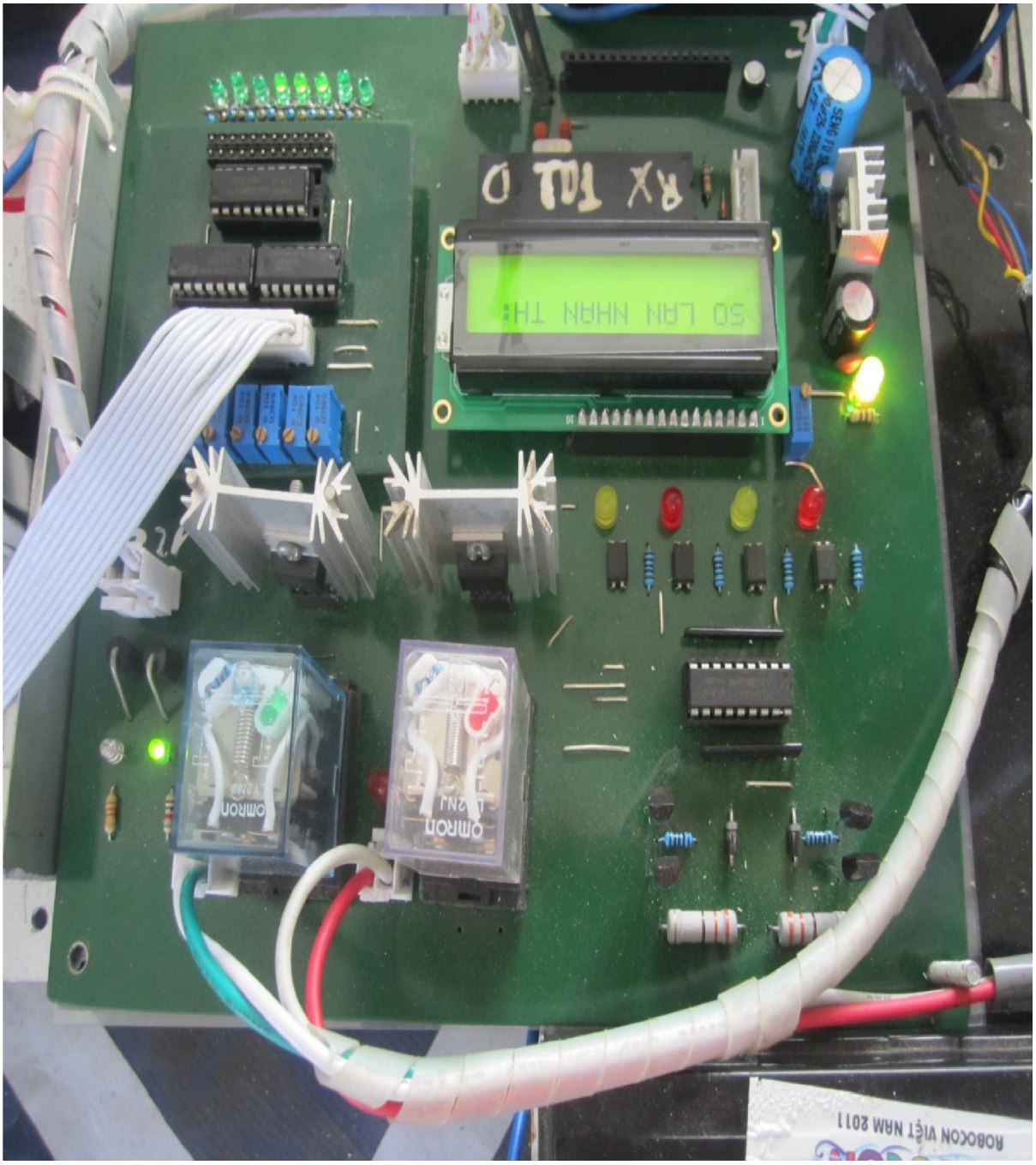
Hình 4.1.7.1 : Sơ đồ nguyên lý mạch điều khiển robot



Hình 4.1.7.2 : Sơ đồ lay_out mạch điều khiển robot



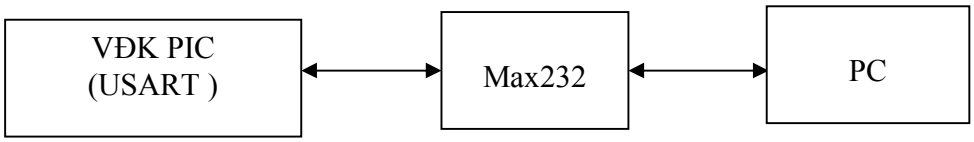
Hình 4.1.7.3 : Mạch điều khiển robo



Hình 4.1.7.4 : Mạch điều khiển lắp trên robo

4.2. Thiết kế module giao tiếp máy tính.

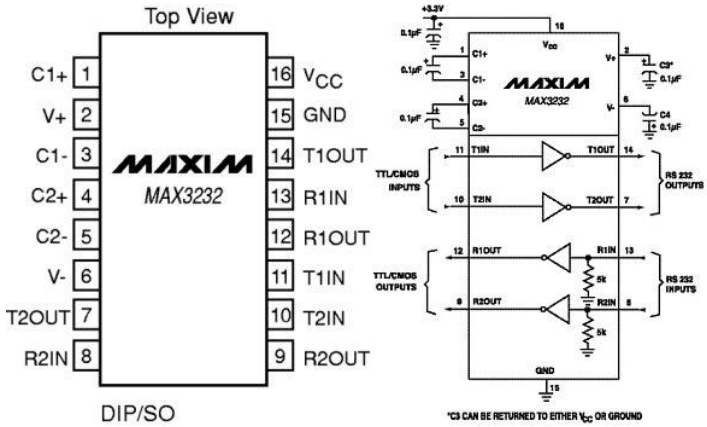
4.2.1. Sơ đồ khối của mạch giao tiếp :



Hình 4.2.1.1: Sơ đồ khối mạch giao tiếp

Max232 dùng để chuyển đổi mức điện áp của vi điều khiển cho tương thích với điện áp của PC.

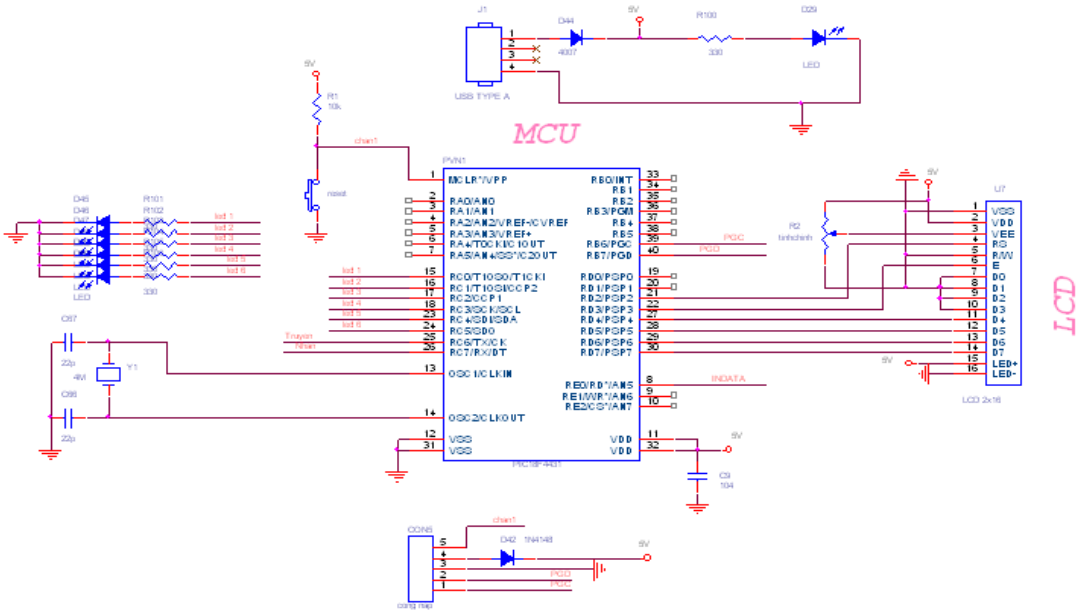
Sơ đồ chân của max232 :



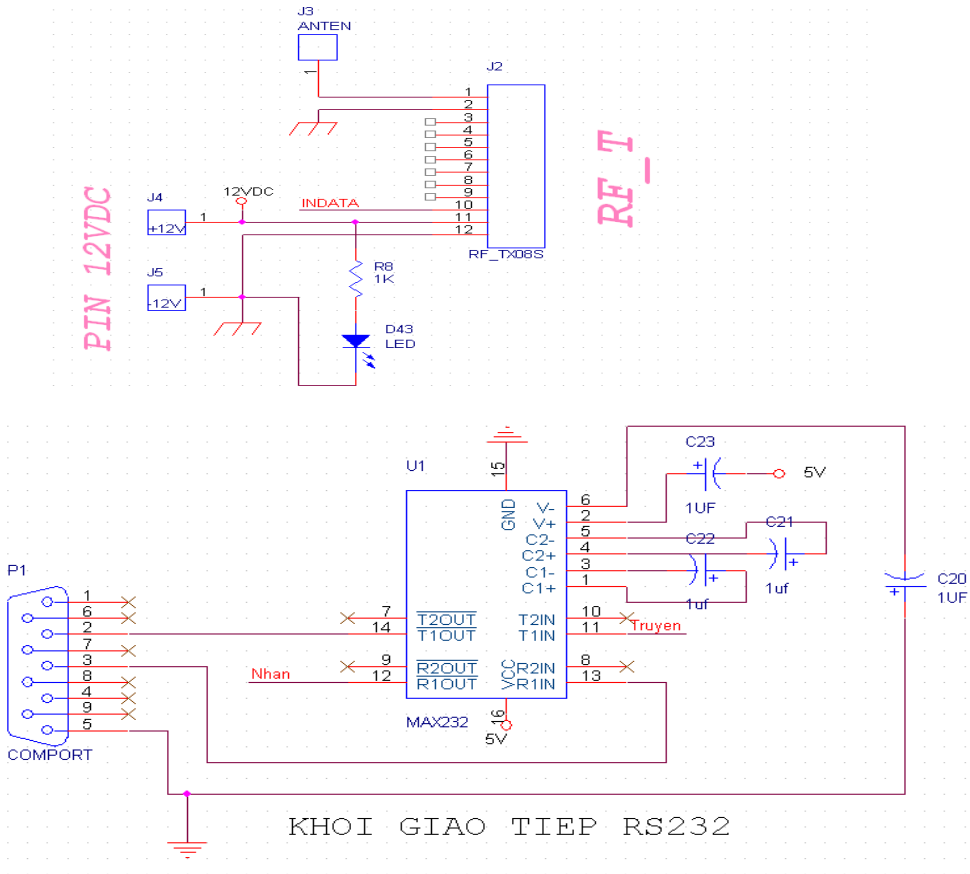
Hình 4.2.1.2: Sơ đồ chân của ic Max232

Vi mạch max232 chuyển đổi mức TTL ở ngỏ vào thành mức +10V hoặc -10V ở phía truyền và các mức +3V…+15V hoặc -3V…-15V thành mức TTL ở phía nhận.

4.2.2. Sơ đồ mạch nguyên lí :



Hình 4.2.2.1: Sơ đồ nguyên lý module giao tiếp



Hình 4.2.2.2 : Sơ đồ nguyên lí khối tiếp máy tính

4.2.3. Quá trình truyền nhận của PIC với PC :

Trong PIC16F877A có nhiều chế độ truyền nhận khác nhau: chế độ truyền nhận dùng bit 9 để định chẵn lẻ, hay cho mạng VĐK 1 master và nhiều slave. Và chế độ chuyền đồng bộ ứng dụng cho việc giao tiếp với A/D, D/A hay với các EEPRO. Và tất nhiên các chế dộ này sẽ được thiết lập bởi các bit trong thanh ghi TXSTA, và TCSTA. Tuy nhiên trong tutorial này chỉ xin giới thiệu chế độ truyền và nhận 8 bit giao tiếp với PC một trong các chức năng của bộ AUSART của PIC.

4.2.3.1. Quá trình truyền dữ liệu:

Trong PIC16F877A để nhận biết được dữ liệu truyền tới người ta dùng bit cờ RCIF trong thanh ghi PIR1. Như vậy khi thanh ghi đệm dữ liệu chứa dữ liệu thì RCIF sẽ được đưa lên 1. Và chính cờ này cho phép PIC16F877A có hai phương thức để nhận biết lúc nào có dữ liệu truyền tới. Sử dụng ngắt và sử dụng kiểu Polling (quay vòng).

Kiểu Polling: liên tục kiểm tra cờ RCIF nếu =1 thì đọc dữ liệu: Phương thức này có ưu điểm dễ lập trình , phù hợp với những ứng dụng nhỏ.

Kiểu dùng ngăt: được thiết lập bằng cách cho RCIE= 1 để cho phép ngắt. Tức là mỗi khi có dữ liệu truyền tới RCREG thì sinh ra một ngắt và PIC sẽ tạm dừng chương trình hiện thời để xử lý dữ liệu vừa nhận được. Cách này chủ yếu được sử dụng .Như vậy các bước cho quá trình nhận dữ liệu của quá trình sử dụng INTERUPT bao gồm:

1. Khởi tạo tốc độ baud: ở thanh ghi SPBRG. Cho SPBRG = 25, BRGH =1 ứng với tốc độ 9600 (thạch anh 4M).

2. Cho phép quá trình truyền không đồng bộ bắng cách thiết lập SPEN = 1, SYNC= 0;

3. Cho phép ngắt quá trình nhận dữ liệu: RCIE=1

4 Cho phép nhận dữ liệu : CREN = 1

5. Cho phép ngắt toàn cục băng việc GIE =1, PEIE = 1 (GIE, PEIE trong thanh ghi INTCON)

6. Xử lý các phần khác chương trình khi có ngắt xảy ra thì xử lý dữ liệu.

4.2.3.2. Quá trình nhận dữ liệu :

Trong quá trình truyền dữ liệu cũng có hai chế độ sử dụng ngắt và Polling , tuy nhiên quá trình truyền dữ liệu lên PC không phải là thời diểm bất kỳ mà dặt dưới sự kiểm soát của chương trình cho nên người ta thường sử dụng kiểu polling khi cần truyền thì truyền đi. Như vậy các bước cho quá trình thiết lập ngắt:

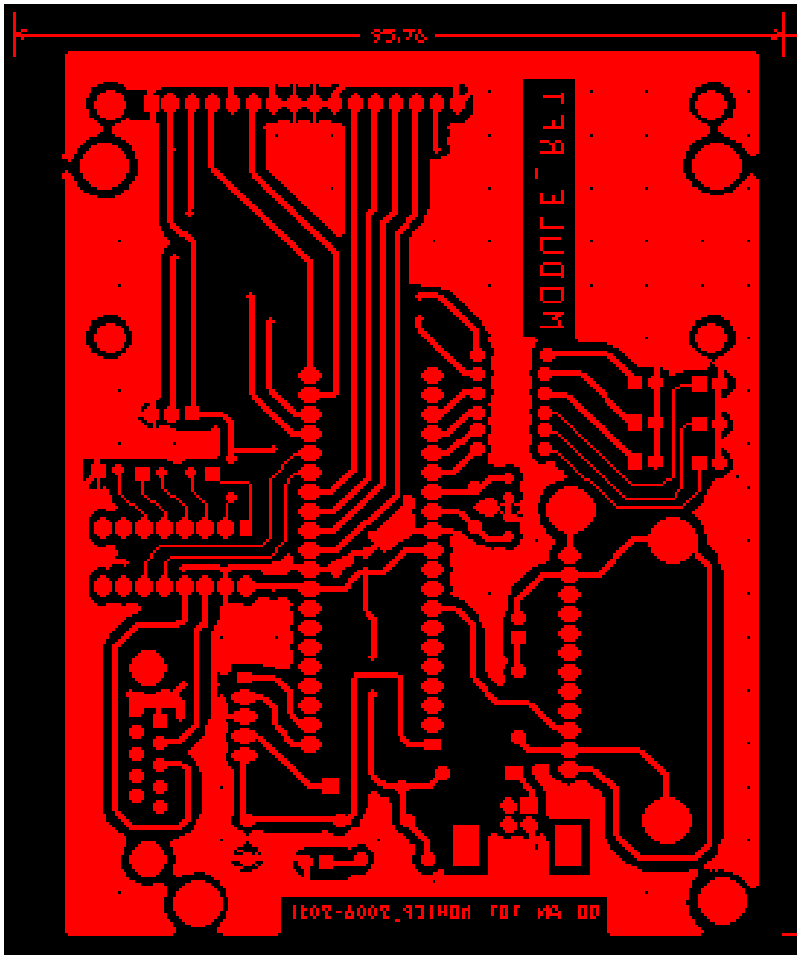
1. Khởi tạo tốc độ baud: ở thanh ghi SPBRG. Cho SPBRG = 25, BRGH =1 ứng với tốc độ 9600 (thạch anh 4M)

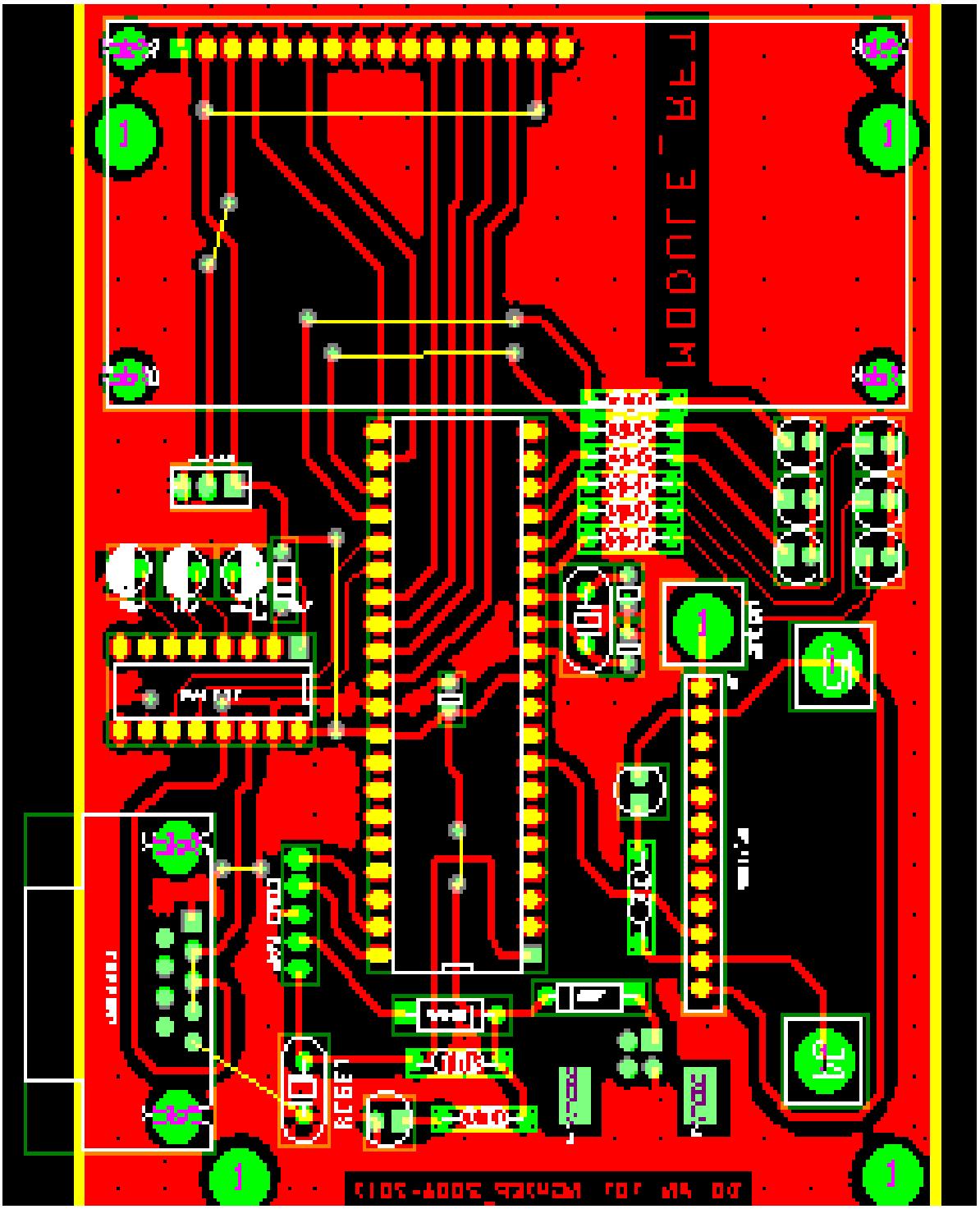
2. Cho phép quá trình truyền không đồng bộ bằng cách thiết lập SPEN = 1, SYNC= 0;

3. Cho phép truyền dữ liệu bằng cách thiết lập bít TXEN = 1;

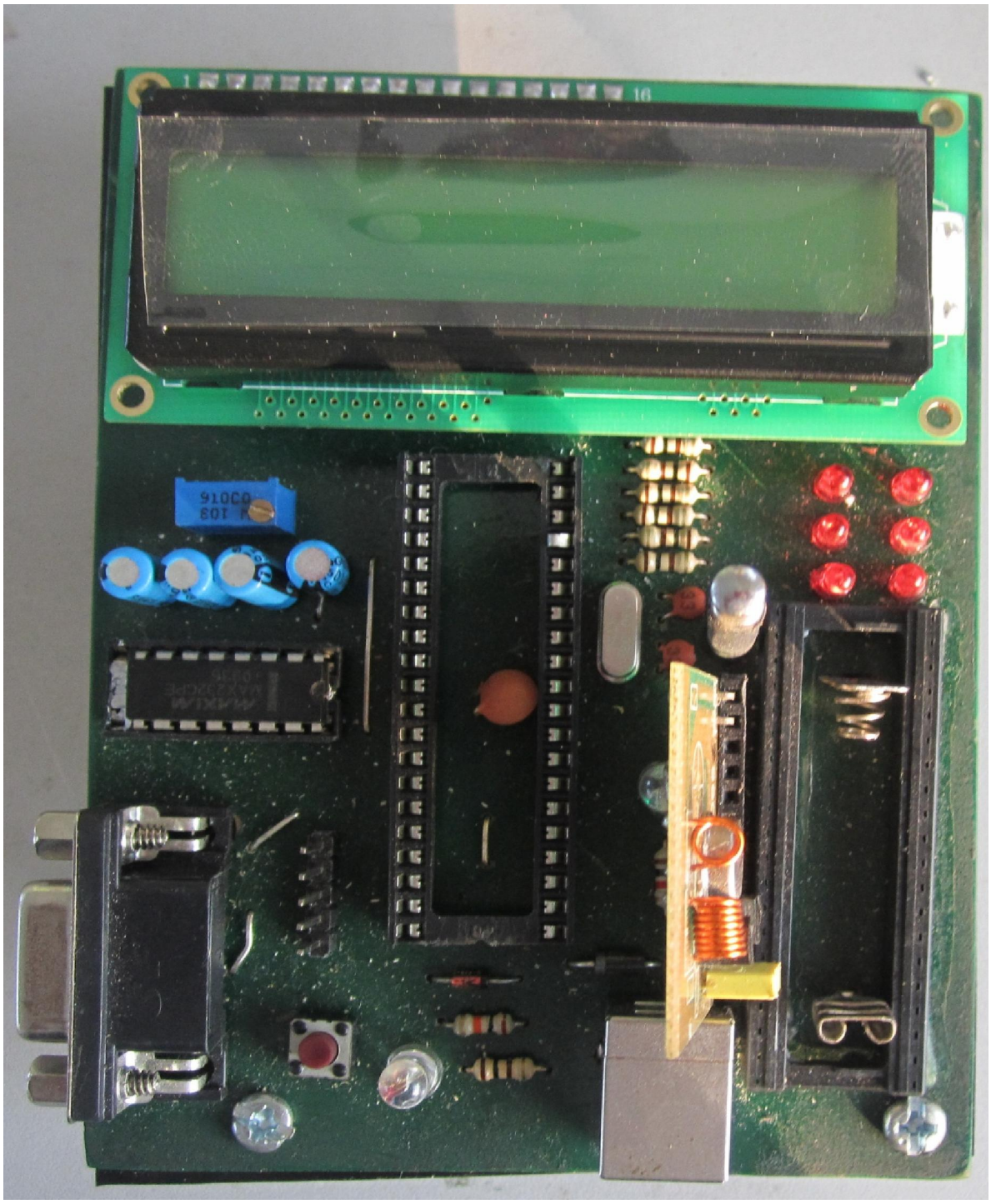
4. Khi cần truyền dữ liệu chỉ cần Load dữ liệu đó lên TXREG.

4.2.4. Kết quả đạt được.





Hình 4.3.1 : Sơ đồi mạch layout khối tiếp máy tính



Hình 4.3.2 : Sơ đồi mạch lay_out khối tiếp máy tính

CHƯƠNG 5:
CẤU TRÚC VÀ THIẾT KẾ PHẦN MỀM

5.1. Chương trình giao diện PC dùng phần mềm Visual Basic:

5.1.1. Chương trình Visual Basic.

```
Public Class frmThihanh

    Dim count As Byte = 0
    Dim count_2 As Byte = 0
    Dim count_1 As Byte = 0
    'Dim array_2(20) As Byte
    Dim array_1(10) As Byte
    Dim tn As Byte = 0
    Private Sub FormThihanh_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        If SerialPort1.IsOpen Then SerialPort1.Close()
        SerialPort1.Open()
    End Sub

    Private Sub btn10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn10.Click
        count = count + 1
        btn10.Text = count
        lblToadoX.Text = "X = " & 0
        lblToadoY.Text = "Y = " & 1
        txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "01; "
        btn10.BackColor = Color.Green
        'array_1(count - 1) = "10"
        'txtNhaptoado.Text = array_1(count - 1)
    End Sub

    Private Sub btn00_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn00.Click
        count = count + 1
        btn00.Text = count
        lblToadoX.Text = "X = " & 0
        lblToadoY.Text = "Y = " & 0
        txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "00; "
        btn00.BackColor = Color.Green
        'array_1(count - 1) = "00"
    End Sub

    Private Sub btn01_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn01.Click
        count = count + 1
        btn01.Text = count
        lblToadoX.Text = "X = " & 1
        lblToadoY.Text = "Y = " & 0
        txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "10; "
        btn01.BackColor = Color.Green
        'array_1(count - 1) = "01"
    End Sub

    Private Sub btn02_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn02.Click
        count = count + 1
        btn02.Text = count
        lblToadoX.Text = "X = " & 2
```

```
    lblToadoY.Text = "Y = " & 0
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "20; "
    btn02.BackColor = Color.Green
    'array_1(count - 1) = "02"
End Sub

Private Sub btn03_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn03.Click
    count = count + 1
    btn03.Text = count
    lblToadoX.Text = "X = " & 3
    lblToadoY.Text = "Y = " & 0
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "30; "
    btn03.BackColor = Color.Green
    'array_1(count - 1) = "03"
End Sub

Private Sub btn04_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn04.Click
    count = count + 1
    btn04.Text = count
    lblToadoX.Text = "X = " & 4
    lblToadoY.Text = "Y = " & 0
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "40; "
    btn04.BackColor = Color.Green
    'array_1(count - 1) = "04"
End Sub

Private Sub btn11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn11.Click
    count = count + 1
    btn11.Text = count
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 1
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "11; "
    btn11.BackColor = Color.Green
    'array_1(count - 1) = "11"
End Sub

Private Sub btn12_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn12.Click
    count = count + 1
    btn12.Text = count
    lblToadoX.Text = "X = " & 2
    lblToadoY.Text = "Y = " & 1
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "21; "
    btn12.BackColor = Color.Green
    'array_1(count - 1) = "12"
End Sub

Private Sub btn13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn13.Click
    count = count + 1
    btn13.Text = count
    lblToadoX.Text = "X = " & 3
    lblToadoY.Text = "Y = " & 1
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "31; "
    btn13.BackColor = Color.Green
    'array_1(count - 1) = "13"
End Sub

Private Sub btn14_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn14.Click
    count = count + 1
```

```
    btn14.Text = count
    lblToadoX.Text = "X = " & 4
    lblToadoY.Text = "Y = " & 1
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "41; "
    btn14.BackColor = Color.Green
    'array_1(count - 1) = "14"
End Sub

Private Sub btn20_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn20.Click
    count = count + 1
    btn20.Text = count
    lblToadoX.Text = "X = " & 0
    lblToadoY.Text = "Y = " & 2
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "02; "
    btn20.BackColor = Color.Green
    'array_1(count - 1) = "20"
End Sub

Private Sub btn21_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn21.Click
    count = count + 1
    btn21.Text = count
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 2
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "12; "
    btn21.BackColor = Color.Green
    'array_1(count - 1) = "21"
End Sub

Private Sub btn22_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn22.Click
    count = count + 1
    btn22.Text = count
    lblToadoX.Text = "X = " & 2
    lblToadoY.Text = "Y = " & 2
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "22; "
    btn22.BackColor = Color.Green
    'array_1(count - 1) = "22"
End Sub

Private Sub btn23_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn23.Click
    count = count + 1
    btn23.Text = count
    lblToadoX.Text = "X = " & 3
    lblToadoY.Text = "Y = " & 2
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "32; "
    btn23.BackColor = Color.Green
    'array_1(count - 1) = "23"
End Sub

Private Sub btn24_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn24.Click
    count = count + 1
    btn24.Text = count
    lblToadoX.Text = "X = " & 4
    lblToadoY.Text = "Y = " & 2
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "42; "
    btn24.BackColor = Color.Green
    'array_1(count - 1) = "24"
End Sub
```

```
Private Sub btn30_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn30.Click
    count = count + 1
    btn30.Text = count
    lblToadoX.Text = "X = " & 0
    lblToadoY.Text = "Y = " & 3
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "03; "
    btn30.BackColor = Color.Green
    'array_1(count - 1) = "30"
End Sub

Private Sub btn31_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn31.Click
    count = count + 1
    btn31.Text = count
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 3
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "13; "
    btn31.BackColor = Color.Green
    'array_1(count - 1) = "31"
End Sub

Private Sub btn32_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn32.Click
    count = count + 1
    btn32.Text = count
    lblToadoX.Text = "X = " & 2
    lblToadoY.Text = "Y = " & 3
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "23; "
    btn32.BackColor = Color.Green
    'array_1(count - 1) = "32"
End Sub

Private Sub btn33_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn33.Click
    count = count + 1
    btn33.Text = count
    lblToadoX.Text = "X = " & 3
    lblToadoY.Text = "Y = " & 3
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "33; "
    btn33.BackColor = Color.Green
    'array_1(count - 1) = "33"
End Sub

Private Sub btn34_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn34.Click
    count = count + 1
    btn34.Text = count
    lblToadoX.Text = "X = " & 4
    lblToadoY.Text = "Y = " & 3
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "43; "
    btn34.BackColor = Color.Green
    'array_1(count - 1) = "34"
End Sub

Private Sub btn40_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn40.Click
    count = count + 1
    btn40.Text = count
    lblToadoX.Text = "X = " & 0
    lblToadoY.Text = "Y = " & 4
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "04; "
    btn40.BackColor = Color.Green
End Sub
```

```
Private Sub btn41_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn41.Click
    count = count + 1
    btn41.Text = count
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 4
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "14; "
    btn41.BackColor = Color.Green
    'array_1(count - 1) = "41"
End Sub

Private Sub btn42_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn42.Click
    count = count + 1
    btn42.Text = count
    lblToadoX.Text = "X = " & 2
    lblToadoY.Text = "Y = " & 4
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "24; "
    btn42.BackColor = Color.Green
    'array_1(count - 1) = "42"
End Sub

Private Sub btn43_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn43.Click
    count = count + 1
    btn43.Text = count
    lblToadoX.Text = "X = " & 3
    lblToadoY.Text = "Y = " & 4
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "34; "
    btn43.BackColor = Color.Green
    'array_1(count - 1) = "43"
End Sub

Private Sub btn44_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn44.Click
    count = count + 1
    btn44.Text = count
    lblToadoX.Text = "X = " & 4
    lblToadoY.Text = "Y = " & 4
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "44; "
    btn44.BackColor = Color.Green
    'array_1(count - 1) = "44"
End Sub

Private Sub btn50_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn50.Click
    count = count + 1
    btn50.Text = count
    lblToadoX.Text = "X = " & 0
    lblToadoY.Text = "Y = " & 5
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "05; "
    btn50.BackColor = Color.Green
    'array_1(count - 1) = "50"
End Sub

Private Sub btn51_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn51.Click
    count = count + 1
    btn51.Text = count
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 5
    txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "15; "
    btn51.BackColor = Color.Green
```

```
        'array_1(count - 1) = "51"
    End Sub

    Private Sub btn52_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn52.Click
        count = count + 1
        btn52.Text = count
        lblToadoX.Text = "X = " & 2
        lblToadoY.Text = "Y = " & 5
        txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "25; "
        btn52.BackColor = Color.Green
        'array_1(count - 1) = "52"
    End Sub

    Private Sub btn53_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn53.Click
        count = count + 1
        btn53.Text = count
        lblToadoX.Text = "X = " & 3
        lblToadoY.Text = "Y = " & 5
        txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "35; "
        btn53.BackColor = Color.Green
        'array_1(count - 1) = "53"
    End Sub

    Private Sub btn54_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn54.Click
        count = count + 1
        btn54.Text =
        lblToadoX.Text = "X = " & 4
        lblToadoY.Text = "Y = " & 5
        txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & "45; "
        btn54.BackColor = Color.Green
        'array_1(count - 1) = "54"
    End Sub

    Private Sub txtNhaptoado_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles txtNhaptoado.TextChanged
        count_1 = count_1 + 1
        If count_1 = 1 Then
            'txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & txtNhaptoado.Text
        End If

        If count_1 = 2 Then
            txtCactoadovuanhap.Text = txtCactoadovuanhap.Text & txtNhaptoado.Text & "; "
            'array_1(count_2) = txtNhaptoado.Text
            'count_2 = count_2 + 1
            array_1(0) = txtNhaptoado.Text
            txtNhaptoado.Clear()
            count_1 = 0
            count_2 = count_2 + 1
            Select Case array_1(0)
                Case 0
                    btn00.Text = count_2
                    lblToadoX.Text = "X = " & 0
                    lblToadoY.Text = "Y = " & 0
                    btn00.BackColor = Color.Green
                Case 1
                    btn10.Text = count_2
                    lblToadoX.Text = "X = " & 0
                    lblToadoY.Text = "Y = " & 1
```

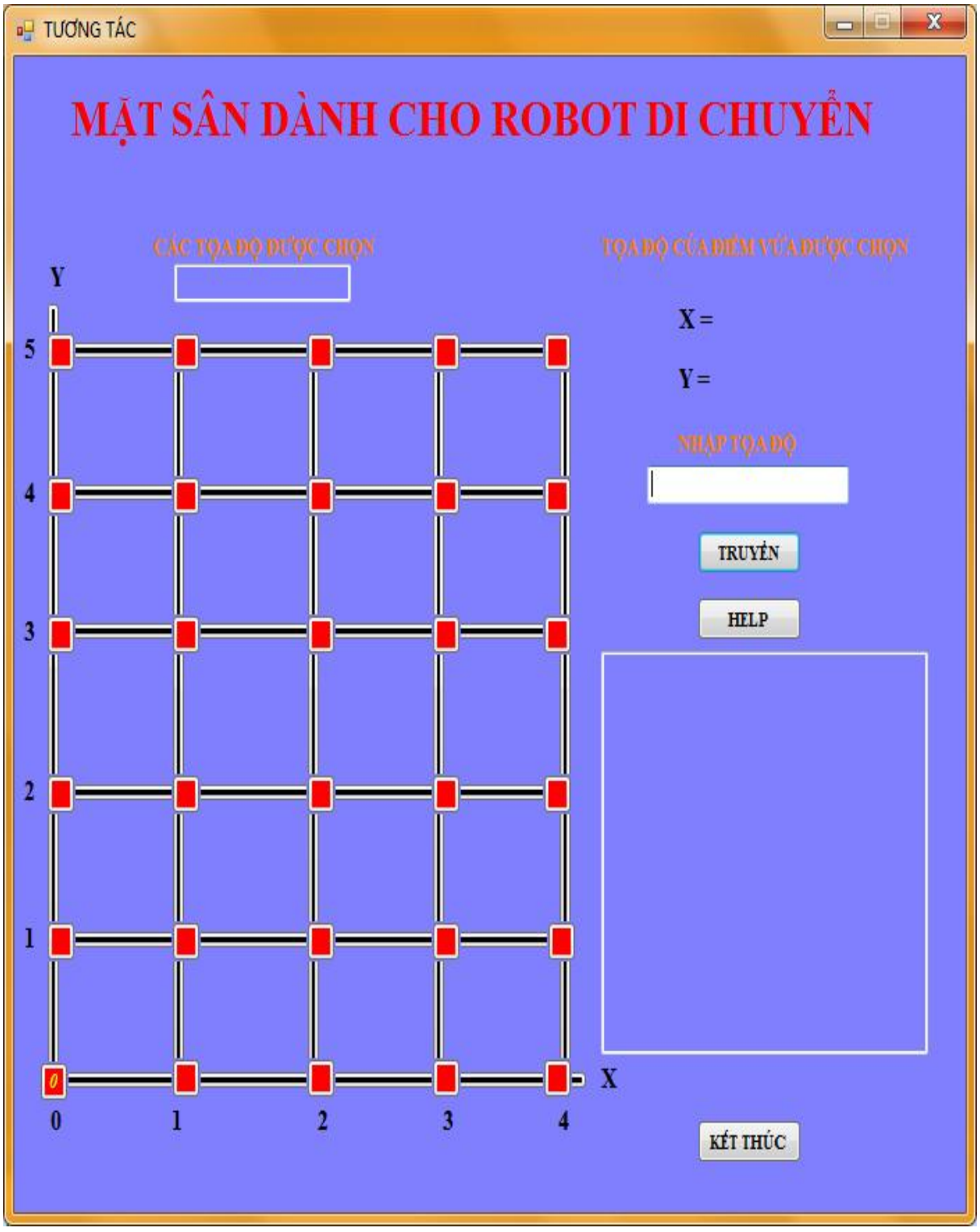
```
    btn10.BackColor = Color.Green
Case 2
    btn20.Text = count_2
    lblToadoX.Text = "X = " & 0
    lblToadoY.Text = "Y = " & 2
    btn20.BackColor = Color.Green
Case 3
    btn30.Text = count_2
    lblToadoX.Text = "X = " & 0
    lblToadoY.Text = "Y = " & 3
    btn30.BackColor = Color.Green
Case 4
    btn40.Text = count_2
    lblToadoX.Text = "X = " & 0
    lblToadoY.Text = "Y = " & 4
    btn40.BackColor = Color.Green
Case 5
    btn50.Text = count_2
    lblToadoX.Text = "X = " & 0
    lblToadoY.Text = "Y = " & 5
    btn50.BackColor = Color.Green
Case 10
    btn01.Text = count_2
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 0
    btn01.BackColor = Color.Green
Case 11
    btn11.Text = count_2
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 1
    btn11.BackColor = Color.Green
Case 12
    btn21.Text = count_2
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 2
    btn21.BackColor = Color.Green
Case 13
    btn31.Text = count_2
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 3
    btn31.BackColor = Color.Green
Case 14
    btn41.Text = count_2
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 4
    btn41.BackColor = Color.Green
Case 15
    btn51.Text = count_2
    lblToadoX.Text = "X = " & 1
    lblToadoY.Text = "Y = " & 5
    btn51.BackColor = Color.Green
Case 20
    btn02.Text = count_2
    lblToadoX.Text = "X = " & 2
    lblToadoY.Text = "Y = " & 0
    btn02.BackColor = Color.Green
Case 21
    btn12.Text = count_2
```

```
        lblToadoX.Text = "X = " & 2
        lblToadoY.Text = "Y = " & 1
        btn12.BackColor = Color.Green
    Case 22
        btn22.Text = count_2
        lblToadoX.Text = "X = " & 2
        lblToadoY.Text = "Y = " & 2
        btn22.BackColor = Color.Green
    Case 23
        btn32.Text = count_2
        lblToadoX.Text = "X = " & 2
        lblToadoY.Text = "Y = " & 3
        btn32.BackColor = Color.Green
    Case 24
        btn42.Text = count_2
        lblToadoX.Text = "X = " & 2
        lblToadoY.Text = "Y = " & 4
        btn42.BackColor = Color.Green
    Case 25
        btn52.Text = count_2
        lblToadoX.Text = "X = " & 2
        lblToadoY.Text = "Y = " & 5
        btn52.BackColor = Color.Green
    Case 30
        btn03.Text = count_2
        lblToadoX.Text = "X = " & 3
        lblToadoY.Text = "Y = " & 0
        btn03.BackColor = Color.Green
    Case 31
        btn13.Text = count_2
        lblToadoX.Text = "X = " & 3
        lblToadoY.Text = "Y = " & 1
        btn13.BackColor = Color.Green
    Case 32
        btn23.Text = count_2
        lblToadoX.Text = "X = " & 3
        lblToadoY.Text = "Y = " & 2
        btn23.BackColor = Color.Green
    Case 33
        btn33.Text = count_2
        lblToadoX.Text = "X = " & 3
        lblToadoY.Text = "Y = " & 3
        btn33.BackColor = Color.Green
    Case 34
        btn43.Text = count_2
        lblToadoX.Text = "X = " & 3
        lblToadoY.Text = "Y = " & 4
        btn43.BackColor = Color.Green
    Case 35
        btn53.Text = count_2
        lblToadoX.Text = "X = " & 3
        lblToadoY.Text = "Y = " & 5
        btn53.BackColor = Color.Green
    Case 40
        btn04.Text = count_2
        lblToadoX.Text = "X = " & 4
        lblToadoY.Text = "Y = " & 0
        btn04.BackColor = Color.Green
```

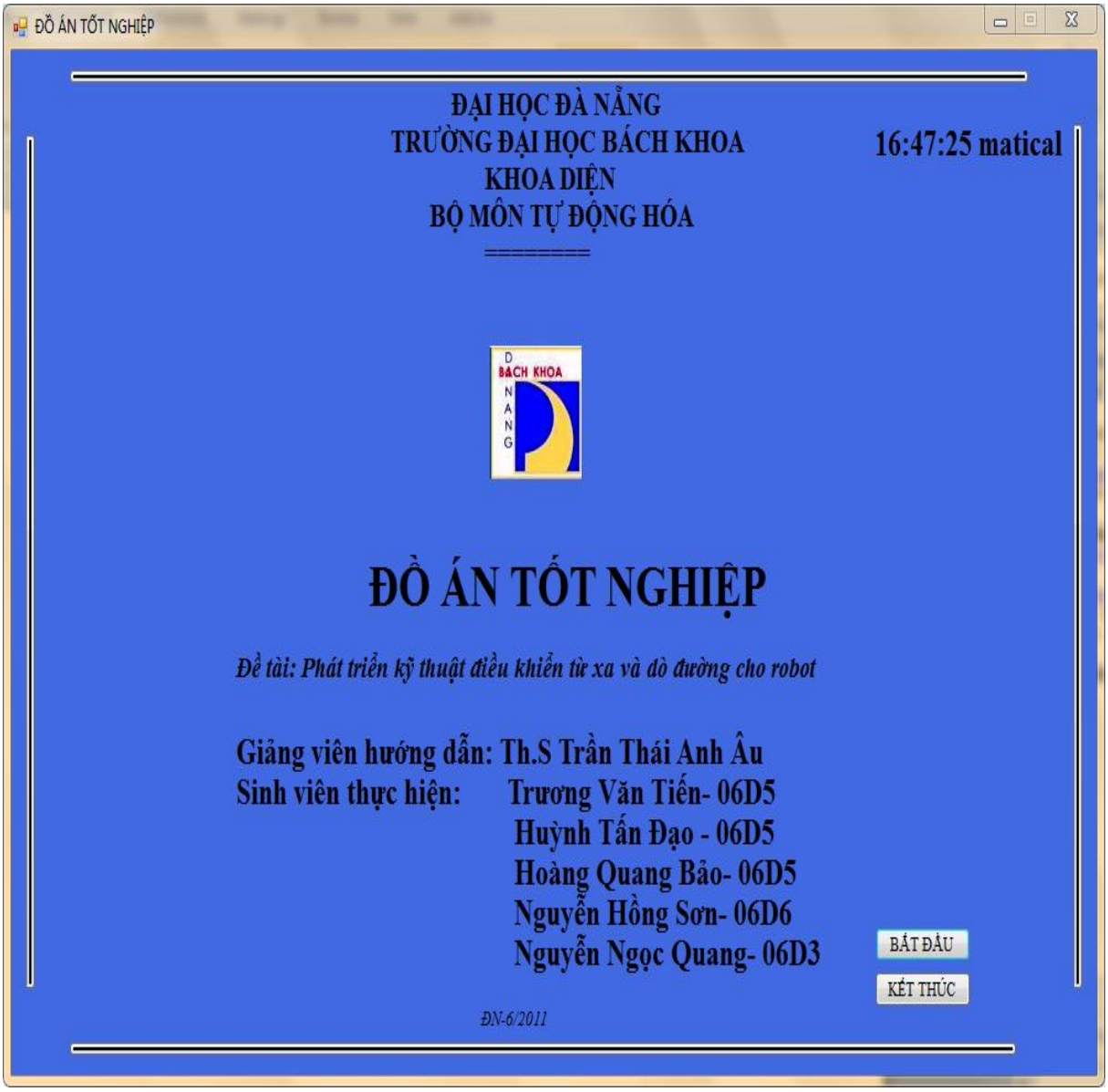
```
            Case 41
                btn14.Text = count_2
                lblToadoX.Text = "X = " & 4
                lblToadoY.Text = "Y = " & 1
                btn14.BackColor = Color.Green
            Case 42
                btn24.Text = count_2
                lblToadoX.Text = "X = " & 4
                lblToadoY.Text = "Y = " & 2
                btn24.BackColor = Color.Green
            Case 43
                btn34.Text = count_2
                lblToadoX.Text = "X = " & 4
                lblToadoY.Text = "Y = " & 3
                btn34.BackColor = Color.Green
            Case 44
                btn44.Text = count_2
                lblToadoX.Text = "X = " & 4
                lblToadoY.Text = "Y = " & 4
                btn44.BackColor = Color.Green
            Case 45
                btn54.Text = count_2
                lblToadoX.Text = "X = " & 4
                lblToadoY.Text = "Y = " & 5
                btn54.BackColor = Color.Green
        End Select

    End If
End Sub

Private Sub btnKetthuc_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnKetthuc.Click
    Me.Close()
    Me.Dispose()
End Sub

Private Sub btnTruyen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnTruyen.Click
    SerialPort1.Write(txtCactoadovuanhap.Text & "@")
    'txtCactoadovuanhap.Clear()
    count = 0
    count_1 = 0
    count_2 = 0
End Sub

Private Sub txtCactoadovuanhap_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles txtCactoadovuanhap.TextChanged

End Sub

Private Sub btnHelp_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnHelp.Click
    'Dim tn As Byte = 0
    tn = tn + 1
    If (tn = 1) Then
        txtTN.Text = "TRỢ GIÚP:                    Có 2 cách để đưa tọa độ từ PC xuống PIC.
1> Nhập các số có 2 chữ số vào ô 'NHẬP TỌA ĐỘ'. Ví dụ: Bạn nhập điểm có tọa độ là 23, như vậy điểm này có
hoành độ là 2 và tung độ là 3.                    2> Click chuột trực tiếp lên các nút trên sân. "
```

```
            End If
            If (tn = 2) Then
                txtTN.Text = ""
                tn = 0
            End If
        End Sub
    End Class
```

5.1.2. Giao diện Visual Basic.



Hình 5.1.2.1:Giao diện sân dành cho ROBOT di chuyển



Hình 5.1.2.2:Giao diện Visual Basic.

5.2. Chương trình nhận dữ liệu từ PC và truyền sóng RF:

5.2.1. Chương trình nhận dữ liệu và xử lý tọa độ của PIC1 trên module phát :

```
#include <18F4431.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232 (baud=9600, xmit=PIN_C6,rcv=PIN_C7)
#include <lcd_lib_4bit_transmit_tn.c>
#bit tx= 0xf84.0   //re0
#define addr 0xaa
int buff[14];// dem cho rf
int count = 0;
int dem = 0;
int nhan = 0, chuyen = 0;
int8 buff_pc_pic[40];
int8 buff_tran_rf[10];
int buff_pc[];  // dem cho pc, so fan tu cua buff va buff_pc bang nhau
void send_data(unsigned char *data);
void transmit_data();
 #int_rda
void get_data_cp()
  {
    nhan = getc();
    if(nhan !=64)
       {
         nhan =nhan -48;
         buff_pc_pic[dem]= nhan;
         dem++;
       }
       else
```

```
        {
          chuyen =1;
        }}
void send_data(unsigned char *data)
{
  unsigned char i,j;
  for(i=0;i<35;i++)
  {
    tx=1;
    delay_us(416);
    tx=0;
    delay_us(416);
  }
  delay_us(1248);// doi dong bo
  tx=1;          // start bit
  delay_us(416);
  tx=0;
  delay_us(416);
  for(i=0;i<(dem/4+4);i++)
  {
    for(j=0;j<8;j++)
    {
      if((data[i]&0x80)==0x80)
        {
          tx=1;
          delay_us(416);
          tx=0;
```

```
            delay_us(416);
          }
        else
          {
           tx=0;
           delay_us(416);
           tx=1;
           delay_us(416);
          }
        data[i]=data[i]<<1;
    }}
  tx=1;
  delay_us(416);
  tx=0;
  delay_us(416);
  delay_ms(2);
}
void transmit_data()
{
  int i;
  {
   buff[0]=addr;
   buff[1]=0x01&addr;
   buff[2]=dem/4;
  for(i=0;i<dem/4;i++)
    {
      buff[i+3]= buff_tran_rf[i];
```

```
        }
       buff[dem/4+3]= 0x99;
      send_data(buff);
      }
      delay_ms(1000);
}
void main()
{
   char k,m=0,n=0;
   set_tris_d(0x00);
   set_tris_a(0x01);
   set_tris_b(0x00);
   set_tris_e(0x00);
   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);
   LCD_init();
   LCD_putcmd(0x01);
   LCD_putcmd(0x80);
   LCD_putchar("BAT DAU");
   while(1)
   {
   if(chuyen ==1)
   {
         LCD_init();
         LCD_putcmd(0x01);
         LCD_putcmd(0x80);
         printf(LCD_putchar,"%d",dem);
```

```
    delay_ms(1000);
  for(k=0;k<(dem/4);k++)
   {
    buff_tran_rf[k]=(buff_pc_pic[n])*10;
    buff_tran_rf[k]=buff_tran_rf[k]+ buff_pc_pic[n+1];
    n=n+4;
    LCD_putcmd(0x01);
    LCD_putcmd(0x80);
    printf(LCD_putchar,"%d",buff_tran_rf[k]);
    delay_ms(500);
   }
  chuyen =0;
  n=0;
  for(m=0;m<4;m++)
  {
  transmit_data();
  output_high(pin_c0);
  delay_ms(500);
  output_low(pin_c0);
  delay_ms(500);
  }
  dem=0;
}}}
```

5.2.2. Chương trình nhận dữ liệu và xử lý tọa độ của PIC2 trên module thu :

```
#include <18F4431.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
```

```
#include <lcd_lib_4bit.c>
#bit rx = 0xf80.0
#define addr 0xaa
#priority ext1
#bit i2c_off=0xfc6.5
#byte fltconfig=0xf6c
 #define role_phai pin_b0
 #define role_trai pin_b2
 #define thuan 1
 #define nghich 0
int getdata[14];
int1 rx_bit;
int count=0,count1=0,huong=1;
int k=0;
int1 get_data(unsigned char *data);
int16 soxung=0,sovach=0;
int rx_toado[10]={0,0,0,0,0,0,0,0,0,0};
 int hoanhdo[10]={0,0,0,0,0,0,0,0,0,0};
 int tungdo[10]={0,0,0,0,0,0,0,0,0,0};
int vtt =75;
int vtp =74;
int quayxong=0;
byte sensor;
int demvach=0,i=0;
int demvach_dau=0;
int demvachquay=0;
int line_status=0;
```

```
void doduong(int vtt,int vtp);
void left_motor_forward(int value)
{
output_high(pin_b2);   // role trai = 1 : chay thuan
set_pwm1_duty(value);
}
void right_motor_forward(int value)
{
output_high(pin_b0);   // role phai = 1 : chay thuan
set_pwm2_duty(value);
}
void left_motor_reverse(int value)
{
output_low(pin_b2); // role trai = 0 : chay nghich
set_pwm1_duty(value);
}
void right_motor_reverse(int value)
{
output_low(pin_b0); // ro le phai = 0 : chay nghich
set_pwm2_duty(value);
}
void left_motor_stop()
{
set_pwm1_duty(83);
}
void right_motor_stop()
{
```

```
set_pwm2_duty(83);
}
 // chuong trinh xu ly toc do 2 dong co
 // 0: stop , " - " : chay nguoc
void speed (int left_motor_speed, int right_motor_speed,int
huong_banh_trai,int huong_banh_phai)
{
int left_pwm_value=0,right_pwm_value=0;
/* Left motor */
if( huong_banh_trai==thuan )
     {
       left_motor_forward(left_motor_speed);
     }
     else
      left_motor_reverse(left_motor_speed);
if( huong_banh_phai==thuan)
     {
       right_motor_forward(right_motor_speed);
     }
     else
      right_motor_reverse(right_motor_speed);
     }
int dovachngang(byte sensor)
{
  int i,soled=0;
  for (i=0;i<8;i++)
  {
```

```
    if (bit_test(sensor,i)==0)
    soled++;
   }
   if (soled>3)
   return 1;
   else
   return 0;
}
int dovachquay(byte sensor)
{
   int i,soled=0;
   for (i=0;i<8;i++)
   {
    if (bit_test(sensor,i)==0)
    soled++;
   }
   if (soled>1)
   return 1;
   else
   return 0;
}
void cuaphai()
{
speed(vtt,vtp,thuan,thuan);
 delay_ms(100) ;
soxung=0;
while(soxung<=200)
```

```
{
doduong(vtt,vtp);
}
 speed(83,83,thuan,thuan);
 delay_ms(200) ;
        //  giua vach
 soxung=0;
speed(83,vtp,thuan,nghich);
  while(soxung<=(int16)120);
  if ((dovachquay(input_d())==0))
  {speed(83,vtp,thuan,nghich);
  while((dovachquay(input_d())==0));
  }
  soxung=0;
quayxong=1;
}
void cuatrai()
{
speed(vtt,vtp,thuan,thuan);
 delay_ms(100) ;
soxung=0;
while(soxung<=200)
{
doduong(vtt,vtp);
}
 speed(83,83,thuan,thuan);
 delay_ms(200) ;
```

```
        //  giua vach
 soxung=0;
speed(vtt,83,nghich,thuan);
  while(soxung<=(int16)130);
  if ((dovachquay(input_d())==0))
  {speed(vtt,83,nghich,thuan);
  while((dovachquay(input_d())==0));
  }
quayxong=1;
soxung=0;
}
void quay180()
{
speed(vtt,vtp,thuan,thuan);
 delay_ms(100) ;
soxung=0;
while(soxung<=40)
{
doduong(vtt,vtp);
}
 speed(83,83,thuan,thuan);
 delay_ms(200) ;
        //  giua vach
 soxung=0;
speed(vtt,83,thuan,thuan);
  while(soxung<=(int16)120);
  if ((dovachquay(input_d())==0))
```

```
  {speed(vtt,83,thuan,thuan);
  while((dovachquay(input_d())==0));
  }
speed(83,vtp,thuan,nghich);
soxung=0;
  while(soxung<=(int16)120);
  if ((dovachquay(input_d())==0))
  {speed(83,vtp,thuan,nghich);
  while((dovachquay(input_d())==0));
  }
  soxung=0;
quayxong=1;
if(huong==0) huong=2;
if(huong==1) huong=3;
if(huong==2) huong=0;
if(huong==3) huong=1;
}
void tien()
{
if (dovachngang(input_d())==1)
{
while(dovachngang(input_d())==1);
  sovach--;
   if(sovach==1) {vtt=76;vtp=75;}
  if(sovach==0)
  {
   speed(83,83,thuan,thuan);   // lay qua 1
```

```
 delay_ms(500);
 vtt=75;
 vtp=74;
  }}}
void tientoi(int sv)
{
sovach=sv;
while(sovach)
{
tien();
doduong(vtt,vtp);
}}
void nhantoado(buff_toado_trunggian)
{
   int buff_toado_trunggian;
   hoanhdo[k]=buff_toado_trunggian/10;
   tungdo[k]=buff_toado_trunggian%10;
   k++;
}
void xulytoado3()
{
for(i=0;i<getdata[2];i++)
{
if(i==0)
   {
   if(hoanhdo[1]==0) tientoi(tungdo[1]);
   else
```

```
    {
    if(tungdo[1]==0) {cuaphai();tientoi(hoanhdo[1]);}
      else
      {
      tientoi(tungdo[1]);
      cuaphai();
      tientoi(hoanhdo[1]);
      } }}
  if(i!=0)
    {
    if(huong==1)
      {
      if(tungdo[i+1]>tungdo[i])
      {
      if(hoanhdo[i+1]>hoanhdo[i])
        {
        tientoi(tungdo[i+1]-tungdo[i]);
         cuaphai();
        tientoi(hoanhdo[i+1]-hoanhdo[i]);
        }
      if(hoanhdo[i]==hoanhdo[i+1])
        {
        tientoi(tungdo[i+1]-tungdo[i]);
        }
      if(hoanhdo[i+1]<hoanhdo[i])
        {
        tientoi(tungdo[i+1]-tungdo[i]);
```

```
      cuatrai();
     tientoi(hoanhdo[i]-hoanhdo[i+1]);
    }}
 if(tungdo[i+1]==tungdo[i])
 {
 if(hoanhdo[i+1]>hoanhdo[i])
    {
     cuaphai();
     tientoi(hoanhdo[i+1]-hoanhdo[i]);
    }
 if(hoanhdo[i+1]<hoanhdo[i])
    {
     cuatrai();
     tientoi(hoanhdo[i]-hoanhdo[i+1]);
    }}
 if(tungdo[i+1]<tungdo[i])
{
 if(hoanhdo[i+1]>hoanhdo[i])
    {
     cuaphai();
     tientoi(hoanhdo[i+1]-hoanhdo[i]);
     cuaphai();
     tientoi(tungdo[i]-tungdo[i+1]);
    }
 if(hoanhdo[i+1]==hoanhdo[i])
    {
     quay180();
```

```
        huong=3;
        tientoi(tungdo[i]-tungdo[i+1]);
       }
    if(hoanhdo[i+1]<hoanhdo[i])
       {
        cuatrai();
        tientoi(hoanhdo[i]-hoanhdo[i+1]);
        cuatrai();
        tientoi(tungdo[i]-tungdo[i+1]);
       }}}
if(huong==3)
   {
      i++;
if(tungdo[i+1]<tungdo[i])
   {
    if(hoanhdo[i+1]<hoanhdo[i])
       {
        tientoi(tungdo[i]-tungdo[i+1]);
        cuaphai();
        tientoi(hoanhdo[i]-hoanhdo[i+1]);
       }
    if(hoanhdo[i]==hoanhdo[i+1])
       {
        tientoi(tungdo[i]-tungdo[i+1]);
       }
    if(hoanhdo[i+1]>hoanhdo[i])
       {
```

```
        tientoi(tungdo[i]-tungdo[i+1]);
         cuatrai();
        tientoi(hoanhdo[i+1]-hoanhdo[i]);
       } }
if(tungdo[i+1]==tungdo[i])
 {
  if(hoanhdo[i+1]<hoanhdo[i])
     {
      cuaphai();
      tientoi(hoanhdo[i]-hoanhdo[i+1]);
     }
  if(hoanhdo[i+1]>hoanhdo[i])
     {
      cuatrai();
      tientoi(hoanhdo[i+1]-hoanhdo[i]);
     }}
if(tungdo[i+1]>tungdo[i])
 {
  if(hoanhdo[i+1]<hoanhdo[i])
     {
      cuaphai();
      tientoi(hoanhdo[i]-hoanhdo[i+1]);
      cuaphai();
      tientoi(tungdo[i+1]-tungdo[i]);
     }
  if(hoanhdo[i+1]==hoanhdo[i])
     {
```

```
        quay180();
        huong=1;
        tientoi(tungdo[i+1]-tungdo[i]);
       }
    if(hoanhdo[i+1]>hoanhdo[i])
       {
        cuatrai();
        tientoi(hoanhdo[i+1]-hoanhdo[i]);
        cuatrai();
        tientoi(tungdo[i+1]-tungdo[i]);
       } }}
if(huong==2)
{
if(hoanhdo[i+1]>hoanhdo[i])
     {
    if(tungdo[i+1]<tungdo[i])
       {
        tientoi(hoanhdo[i+1]-hoanhdo[i]);
        cuaphai();
        tientoi(tungdo[i]-tungdo[i+1]);
       }
    if(tungdo[i]==tungdo[i+1])
       {
        tientoi(hoanhdo[i]-hoanhdo[i+1]);
       }
    if(tungdo[i+1]>tungdo[i])
       {
```

```
        tientoi(hoanhdo[i+1]-hoanhdo[i]);
        cuatrai();
        tientoi(tungdo[i+1]-tungdo[i]);
       } }
if(hoanhdo[i+1]==hoanhdo[i])
 {
  if(tungdo[i+1]<tungdo[i])
     {
      cuaphai();
      tientoi(tungdo[i]-tungdo[i+1]);
     }
  if(tungdo[i+1]>tungdo[i])
     {
      cuatrai();
      tientoi(tungdo[i+1]-tungdo[i]);
     }}
if(hoanhdo[i+1]<hoanhdo[i])
 {
  if(tungdo[i+1]<tungdo[i])
     {
     cuaphai();
     tientoi(tungdo[i]-tungdo[i+1]);
     cuaphai();
     tientoi(hoanhdo[i]-hoanhdo[i+1]);
     }
  if(tungdo[i+1]>tungdo[i])
     {
```

```
        cuatrai();
        tientoi(tungdo[i+1]-tungdo[i]);
        cuatrai();
        tientoi(hoanhdo[i]-hoanhdo[i+1]);
        }
      if(tungdo[i+1]==tungdo[i])
        {
         quay180();
         huong=0;
         tientoi(hoanhdo[i]-hoanhdo[i+1]);
        }}}
  if(huong==0)
  {
i++;
  if(hoanhdo[i+1]<hoanhdo[i])
     {
      if(tungdo[i+1]<tungdo[i])
        {
         tientoi(hoanhdo[i]-hoanhdo[i+1]);
         cuatrai();
         tientoi(tungdo[i]-tungdo[i+1]);
        }
      if(tungdo[i]==tungdo[i+1])
        {
         tientoi(hoanhdo[i]-hoanhdo[i+1]);
        }
      if(tungdo[i+1]>tungdo[i])
```

```
        {
         tientoi(hoanhdo[i]-hoanhdo[i+1]);
         cuaphai();
         tientoi(tungdo[i+1]-tungdo[i]);
        }}
      if(hoanhdo[i+1]==hoanhdo[i])
     {
       if(tungdo[i+1]<tungdo[i])
         {
         cuatrai();
         tientoi(tungdo[i]-tungdo[i+1]);
         }
       if(tungdo[i+1]>tungdo[i])
         {
         cuaphai();
         tientoi(tungdo[i+1]-tungdo[i]);
         }
       }
    if(hoanhdo[i+1]>hoanhdo[i])
     {
       if(tungdo[i+1]<tungdo[i])
         {
        cuatrai();
        tientoi(tungdo[i]-tungdo[i+1]);
        cuatrai();
        tientoi(hoanhdo[i+1]-hoanhdo[i]);
         }
```

```
      if(tungdo[i+1]>tungdo[i])
        {
        cuaphai();
        tientoi(tungdo[i+1]-tungdo[i]);
        cuaphai();
        tientoi(hoanhdo[i+1]-hoanhdo[i]);
        }
      if(tungdo[i+1]==tungdo[i])
        {
         quay180();
         huong=2;
         tientoi(hoanhdo[i+1]-hoanhdo[i]);
        }} } }}}
void doduong(int vtt, int vtp)
{sensor=input_d();
switch (sensor)
{
   case 0b11100111: speed(vtt,vtp,thuan,thuan);line_status=0;  break;
   case 0b11110111: speed(vtt+1,vtp-1,thuan,thuan);line_status=1;break;
   case 0b11110011: speed(vtt+2,vtp-2,thuan,thuan);line_status=1; break;
   case 0b11111011: speed(vtt+3,vtp-3,thuan,thuan);line_status=1; break;
   case 0b11111001: speed(vtt+4,vtp-4,thuan,thuan);line_status=1; break;
   case 0b11111101: speed(vtt+5,vtp-5,thuan,thuan);line_status=1; break;
   case 0b11111100: speed(vtt+6,vtp-6,thuan,thuan);line_status=1; break;
   case 0b11111110: speed(vtt+7,vtp-7,thuan,thuan);line_status=1; break;
   case 0b00000000: speed(vtt,vtp,thuan,thuan);line_status=0;      break;
   case 0b11101111: speed(vtt-1,vtp+1,thuan,thuan); line_status=2;break;
```

```
    case 0b11001111: speed(vtt-2,vtp+2,thuan,thuan); line_status=2;break;
    case 0b11011111: speed(vtt-3,vtp+3,thuan,thuan); line_status=2; break;
    case 0b10011111: speed(vtt-4,vtp+4,thuan,thuan); line_status=2; break;
    case 0b10111111: speed(vtt-5,vtp+5,thuan,thuan); line_status=2; break;
    case 0b00111111: speed(vtt-6,vtp+6,thuan,thuan);  line_status=2;break;
    case 0b01111111: speed(vtt-7,vtp+7,thuan,thuan);  line_status=2;break;
    case 0b11111111:
    {
        if (line_status==2)
        {
        speed(vtt,83,thuan,thuan);
         break;
        }
        if (line_status==1)
        {
        speed(83,vtp,thuan,thuan);
         break;
        } }}}
 #INT_EXT
 void doc_encoder()
 {
 soxung++;
 }
void main()
 {
      char selection;
   byte value;
```

```
int duty1;
int duty2;
//int16 count=0;
fltconfig=0x80;//DE DUNG POWER HOAC CCP
set_tris_a(0xFF);
set_tris_c(0x08);
set_tris_b(0x00);
set_tris_d(0xff);
set_tris_e(0xff);
setup_ccp1(CCP_pwm);
setup_ccp2(CCP_pwm);
setup_timer_2(T2_DIV_BY_4, 83, 1);
enable_interrupts(GLOBAL);
i2c_off=0;
output_b(0xff);
ext_int_edge( 1, h_TO_l);
output_c(0xff);
clear_interrupt(int_ext1);
enable_interrupts(INT_ext);
speed(83,83,thuan,thuan);
delay_ms(1000);
i=0;
  LCD_init();
  LCD_putcmd(0x01);
  LCD_putcmd(0x80);
  LCD_putchar("SO LAN NHAN TH:");
  while(1)
```

```
{
 // if(rx==1) output_d(0xff);
  while(get_data(getdata));
     LCD_putcmd(0x01);
     LCD_putcmd(0x80);
     LCD_putchar("BAT DAU");
     delay_ms(500);
  if(getdata[0]==addr)
  {
    if(getdata[1]==(addr&0x01))
    {
      if(getdata[getdata[2]+3] == 0x99)// getdata[2] chua so diem co toa do
can xu ly
     //output_b(getdata[2]);
     LCD_putcmd(0x01);
     LCD_putcmd(0x80);
     LCD_putchar("NHAN XONG");
     speed(83,83,thuan,thuan);
     delay_ms(1000);
     for(i=3;i<getdata[2]+3;i++)
        {
         rx_toado[i-3] = getdata[i];
         nhantoado(rx_toado[i-3]);
         LCD_putcmd(0x01);
        // count++;
         LCD_putcmd(0x80);
         printf(LCD_putchar,"%d",hoanhdo[i-3]);
```

```
      LCD_putcmd(0xc2);
      printf(LCD_putchar,"%d",tungdo[i-3]);
      delay_ms(1000);
     }
k=0;
i=0;
huong=1;
xulytoado3();
  output_high(pin_c2);
  output_high(pin_c3);
  while(1);
delay_ms(2000);
}
if(getdata[1]==(addr&0x02))
{
output_b(getdata[2]);

LCD_putcmd(0x80);
LCD_putchar("NHAN SAI:");
delay_ms(2000);
output_b(0xff);
}
if(getdata[1]==(addr&0x03))
{
output_b(getdata[2]);
delay_ms(2000);
output_b(0xff);
```

```
        }
      }
      else
       {
        LCD_putcmd(0x01);
         LCD_putcmd(0x80);
         LCD_putchar("THAT BAI");
       } }
}
int1 get_data(unsigned char *data)
{
 int8 i,j,count=0,count1=0;
 setup_timer_0 (RTCC_DIV_16|RTCC_8_BIT);
 i=35;
 while(i--)
 {
 delay_us(1);
 while(rx);
 set_timer0(0);
  delay_us(1);
while(!rx);
count1=get_timer0();
if(count1==255)
return 1;
if((count1>=124)&&(count1<=135))// 384us va 432us
  {
  count++;
```

```
 }
}
if(count==0)
{
 return 1;
}
if(count>=1)
  {
  count1=0;
  setup_timer_0 (RTCC_DIV_64|RTCC_8_BIT);
  while(!((count1>=118)&&(count1<=143)))// dong bo 1662us 1504/1824
  {
    delay_us(1);
    while(rx);
    set_timer0(0);
    delay_us(1);
    while(!rx);
    count1=get_timer0();
  }
  while(rx);// start bit go slow
  delay_us(624);// 416 us start bit slow + 208us( 1/2 bit dau tien
              // neu la 1 thi bit do la1, neu la 0 thi bit do la 0)
  for(i=0;i<14;i++)
  {
   data[i]=0;
   for(j=0;j<8;j++)
   {
```

```
      data[i]=data[i]<<1;
      rx_bit=rx;
      data[i]=data[i]|rx_bit;
      if(rx_bit==0)
        while(!rx);
      else
        while(rx);
        delay_us(624);// doi rx len 1, khi len 1 thi o bo truyen delay 416us
                    // 624 =416+1/2 thoi jan delay_cua bit ma hoa dau
     }}
   if(rx_bit==0)
     if(!rx) return 1;// ko co bit stop
   else
     if(!rx) return 1;
   return 0;
}}
```

CHƯƠNG 6:
ĐÁNH GIÁ KẾT QUẢ ĐẠT ĐƯỢC VÀ ỨNG DỤNG THỰC TIỄN

6.1. Đánh giá kết quả đạt được:

6.1.1. Ưu điểm :

- Giao diện PC với người điều khiển thân thiện, dễ sử dụng, trực quan và có tính thẩm mỹ.

- Kết nối giữa PC với PIC1 ổn định, dữ liệu truyền đảm bảo được tính chính xác cao và liên tục.

- Đã giảm được tối đa độ nhiễu trong môi trường truyền sóng RF bằng thuật toán và phần mềm, dữ liệu nhận được tương đối ổn định và chính xác, quá trình mã hóa và giải mã được thực hiện đồng bộ và tương đối ổn định.

- Robot dò đường theo vạch trắng ổn định, cảm biến đọc vạch trắng kô bị nhiễu trong nhiều môi trường khác nhau, xử lý tọa độ chính xác, di chuyển đến vị trí mà người điều khiển chọn một cách chính xác, đảm bảo được quãng đường đi là ngắn nhất.

6.1.2. Khuyết điểm :

- Tính ổn định và liên tục của việc truyền, nhận sóng RF chưa đạt đến độ hoàn hảo, đôi lúc còn bị nhiễu bởi môi trường chung quanh.

- Mạch công suất của động cơ vẫn còn nóng khi hoạt động với thời gian dài.

- Mạch cảm biến hoạt động ổn định, tuy nhiên vẫn bị ảnh hưởng nhiễu trong môi trường quá nhiều ánh sáng, dẫn đến việc đọc vạch trắng không chính xác, robot di chuyển lệch vạch.

6.1.3. Phát triển :

- Quá trình truyền nhận sóng RF sử dụng bằng các module chuyên dụng, nâng cao tính ổn định và chống nhiễu tốt hơn, qua đó đảm bảo cho robot nhận và xử lý thông tin một cách chính xác

- Thay đổi mạch dò đường kiểu truyền thống cho robot, thay các bộ so sánh điện áp trong nguyên lý mạch cảm biến bằng cách sử dụng trực tiếp chức năng đọc dữ liệu

ADC của PIC. Qua đó, mạch cảm biến sẽ đơn giản, ổn định, thời gian tinh chỉnh cảm biến nhanh và đảm bảo tính chính xác trong mọi môi trường di chuyển của robot.

- Cần thêm một module thu phát sóng RF làm chức năng phản hồi trong quá trình di chuyển của robot. Từ đó người điều khiển có thể giám sát được vị trí hoặc tốc độ của robot một cách liên tục trong quá trình di chuyển.

- Trang bị thêm cho robot thiết bị ghi hình, như camera, phát hình ảnh quay được trong quá trình di chuyển về PC, và thêm một module điều khiển từ xa bằng tay, để khi độ nhiễu trong môi trường truyền sóng RF tăng lên, người điều khiển vẫn có thể điều khiển robot bằng tay, thông qua hình ảnh mà robot truyền về, nâng cao được độ linh hoạt của robot.

- Nghiên cứu và phát triển thêm một hệ thống đo khoảng cách bằng sóng siêu âm trên robot, để trong quá trình di chuyển, robot phát hiện vật cản, tự động đổi hướng và di chuyển đến điểm mà người điều khiển đã chọn.

6.2. Ứng dụng thực tiễn :

Đề tài nghiên cứu thiết kế hệ thống điều khiển từ xa cho robot có thể áp dụng cho một số công việc ở những địa hình nguy hiểm mà con người không thể đến được, như vào hầm mỏ sâu, ghi lại tình trạng hiện trường của 1 số công trình được xây dựng ở các địa thế hiểm trở, phụ vụ cho công tác cứu hộ.

Đề tài này nếu phát triển được thuật toán dò đường sử dụng chức năng ADC của PIC như đã đề cập ở trên, thì có thể tăng tính ổn định và hiệu quả cho robot các đội tham dự cuộc thi Sáng tạo Robocon Châu Á_Thái Bình Dương diễn ra hằng năm.

Tài liệu tham khảo

Tài liệu sách tham khảo .

[số thứ tự] Tên tác giả (năm xuất bản). *Tên sách*. Nhà xuất bản.

[1]- Đỗ Xuân Thụ (1999) .*Kỷ thuật điện tử* . Nhà Xuất Bản Giáo Dục
[2]-Lê Văn Doanh(1999).Điện tử công suất(tập 1). Nhà Xuất Bản Khoa Học và Kỹ Thuật Hà Nội
[3]-Bùi Quốc Khánh(2001). Điều chỉnh tự động truyền động điện. Nhà Xuất Bản Khoa Học và Kỹ Thuật Hà Nội.
[4] Datasheet PIC 18f4431

.....

Tài liệu tham khảo trên mạng INTERNET.

Đối với loại tài liệu này chủ yếu là các diễn đàn và trang chủ của các công ty điện tử.

[1]- http://www.ccsinfo.com/forum/viewtopic.php?t=21541
[2]- http://www.ccsinfo.com/forum/viewtopic.php?t=42426
[3]- http://www.dientuvietnam.net/forums/
[4]- http://www.robotshop.com/
[5]- http://ngoinhachungnet.com/forum/phan-mem-chuyen-nganh-dien-dien-tu/7228-phan-mem-lap-trinh-vdk-8051-80c51.html
[6]- http://ngoinhachungnet.com/forum/
[7]- http://kysubachkhoa.com/forum/index.php
[9]- http://www.hiendaihoa.com/forum
[10]- http://www.hocnghetructuyen.vn/

.....