TẬP ĐOÀN ĐIỆN LỰC VIỆT NAM
TRƯỜNG ĐẠI HỌC ĐIỆN LỰC
KHOA CÔNG NGHỆ TỰ ĐỘNG

ĐẠI HỌC ĐIỆN LỰC
ELECTRIC POWER UNIVERSITY

ĐỒ ÁN VI XỬ LÝ

ĐỀ TÀI : NGHIÊN CỨU THIẾT KẾ MẠCH ĐO KHOẢNG CÁCH SỬ DỤNG VI ĐIỀU KHIỂN PIC16F887

Giảng viên hướng dẫn :**Ths.** Bùi Thị Duyên
Lớp : Đ6LT – ĐCN
Sinh viên thực hiện : Ngô Ngọc Hà

Hà Nội, Ngày 14 Tháng 2 Năm 2013

LỜI NÓI ĐẦU

Ngày nay kỹ thuật vi điều khiển đã trở nên quen thuộc trong các ngành kỹ thuật và trong dân dụng. Các bộ vi điều khiển có khả năng xử lý nhiều hoạt động phức tạp mà chỉ cần một chip vi mạch nhỏ, nó đã thay thế các tủ điều khiển lớn và phức tạp bằng những mạch điện gọn nhẹ, dễ dàng thao tác sử dụng.

Vi điều khiển không những góp phần vào kỹ thuật điều khiển mà còn góp phần to lớn vào việc phát triển thông tin. Chính vì các lý do trên, việc tìm hiểu, khảo sát vi điều khiển là điều mà các sinh viên ngành điện mà đặc biệt là chuyên ngành kỹ thuật điện-điện tử phải hết sức quan tâm. Đó chính là một nhu cầu cần thiết và cấp bách đối với mỗi sinh viên, đề tài này được thực hiện chính là đáp ứng nhu cầu đó.

Các bộ điều khiển sử dụng vi điều khiển tuy đơn giản nhưng để vận hành và sử dụng đươc lại là một điều rất phức tạp. Phần công việc xử lý chính vẫn phụ thuộc vào con người, đó chính là chương trình hay phần mềm. Nếu không có sự tham gia của con người thì hệ thống vi điều khiển cũng chỉ là một vật vô tri. Do vậy khi nói đến vi điều khiển cũng giống như máy tính bao gồm 2 phần là phần cứng và phần mềm.

Từ yêu cầu của môn học kĩ thuật vi xử lý trong đo lường điều khiển và thực tiễn như trên, chúng em quyết định chọn đề tài cho đồ án môn học là: "Nghiên cứu thiết kế mạch đo khoảng cách dùng vi điều khiển PIC16F887"

Dưới đây chúng em xin trình bày toàn bộ nội dung đồ án: "**Nghiên cứu thiết kế mạch đo khoảng cách dùng vi điều khiển PIC16F887**" do cô Ths.Bùi Thị Duyên giảng viên Trường Đại Học Điện Lực hướng dẫn.

Trong quá trình thực hiện đề tài vẫn còn nhiều sai sót, mong nhận được nhiều ý kiến đóng góp từ cô và các bạn.

Sinh viên thực hiện: Ngô Ngọc Hà

NHIỆM VỤ THIẾT KẾ

- Sử dụng cảm biến siêu âm SRF05 để đo khoảng cách.
- Sử dụng RealTime DS1307 lấy thời gian lúc đo.
- Hiện thị kết quả đo được và thời gian đo lên LCD.
- Nguồn cung cấp sử dụng DC Adaptor 7 ÷ 12VDC

MỤC LỤC

CHƯƠNG I
ĐẶT VẤN ĐỀ VÀ NHIỆM VỤ THỬ

I.ĐẶT VẤN ĐỀ

Ngày nay, những ứng dụng của vi điều khiển đã đi sâu vào đời sống tinh sinh hoạt và sản xuất của con người. Thực tế hiện nay là hầu hết các thiết bi dân dụng điều có sự góp mặt của Vi điều khiển và vi xử lý. Ứng dụng vi điều khiển trong thiết kế hệ thống làm giảm chi phí thiết kế và hạ giá thành sản phẩm đồng thời nâng cao tính ổn định của thiết bị cũng như hệ thống. Trên thị trường có rất nhiều họ vi điều khiển như 8051 của hãng Intell, PIC của hãng Microchip, H8 của hãng Hitachi,vv….

Việc phát triển ứng dụng các hệ thống vi điều khiển đòi hỏi những hiểu biết về cả phần cứng và phần mềm, nhưng cũng chính vì vậy mà các hệ thống vi xử lý được sử dụng đẻ giải quyết các bài toán khác nhau. Tính đa dạng của các ứng dụng phụ thuộc vào việc lựa chọn các hệ thống vi xử lý cụ thể cũng như vào kỹ thuật lập trình.

Ngày nay các bộ vi xử lý có mặt trong các thiết bị điện tử hiện đại như máy thu hình, máy ghi hình dàn âm thanh, các bộ điều khiển cho lò sưởi và hệ thống điều hòa đến các thiết bị điều khiển dùng trong công nghiệp.Lĩnh vực ứng dụng của hệ thống vi xử lý như nghiên cứu khoa học, cũng như trong y tế giao thông đến công nghiệp, năng lượng…Chúng ta có thể sử dụng các ngôn ngữ khác như lập trình C, C++, Visual,…

II.NHIỆM VỤ THỬ

-Tìm tài liệu liên quan đến đề tài mà mình nghiên cứu, đưa ra các giải pháp tối ưu cho việc thiết kế chế tạo sản phẩm thực tế.

-Thiết kế chế tạo board mạch gồm các khối như: khối xử lý trung tâm dung họ vi điều khiển PIC16F887, khối cảm biến dùng SRF05 để đo khoảng cách, khối hiện thị dùng LCD.

-Thiết kế khối nguồn để cung cấp điện áp và dòng điện ổn định cho board hoạt động tốt.

-Tiến hành viết chương trình phần mềm phối hợp hoạt động các khối với nhau dưới sự điều khiển của khối mạch chính chứa PIC16F887.

CHƯƠNG II
TỔNG QUAN VỀ PIC VÀ CÁC PHƯƠNG PHÁP ĐO KHOẢNG CÁCH VÀ CẢM BIẾN

I.Giới thiệu vi điều khiển PIC16F887

- Sơ lược về vi điều khiển PIC16F887.
- Khảo sát vi điều khiển PIC16F887 của hãng Microchip.
+ Sơ đồ chân linh kiện.
+ Sơ đồ khối của PIC16F887.
+ Các ứng dụng của PIC16F887.

1.Sơ lược về vi điều khiển PIC 16F887

+ Sử dụng công nghệ tích hợp cao RISC CPU.
+ Người dùng có thể lập trình với 35 câu lệnh đơn giản.
+ Tất cả các câu lệnh thự hiện trong một chu kỳ, lệnh ngoại trừ một số câu lệnh riêng rẽ nhánh thực hiện trong hai chu kỳ lệnh.
+ Tốc độ hoạt động : - Xung đồng hồ là DC -20MHz.
- Chu kì lệnh thực hiện trong 200ns.
+ Bộ nhớ chương trình Flash 8Kx14 words.
+ Bộ nhớ SRam 368x8 bytes.
+ Bộ nhớ EFPROM 256x8 bytes.
+ Số port I/O 35 port

*Khả năng của PIC

+ Khả năng ngắt
+ Ngăn nhớ Stack được phân chia làm 8 mức
+ Truy cập bộ nhớ bằng địa chỉ trực tiếp hoặc gián tiếp
+ Nguồn khởi động lại (POR)
+ Bộ tạo thời gian PWRT, bộ tạo dao động OST
+ Bộ đếm xung thời gian WDT với nguồn dao động trên chip(nguồn dao động RC) đáng tin cậy
+ Có mã chương trình bảo vệ
+ Phương thức cất giữ Sleep
+ Thiết kế toàn tĩnh
+ Dải điện thế hoạt động 2V÷5,5V
+ Dòng điện sử dụng 25mA

-Các tính năng nổi bật của thiết bị ngoại vi trên chíp

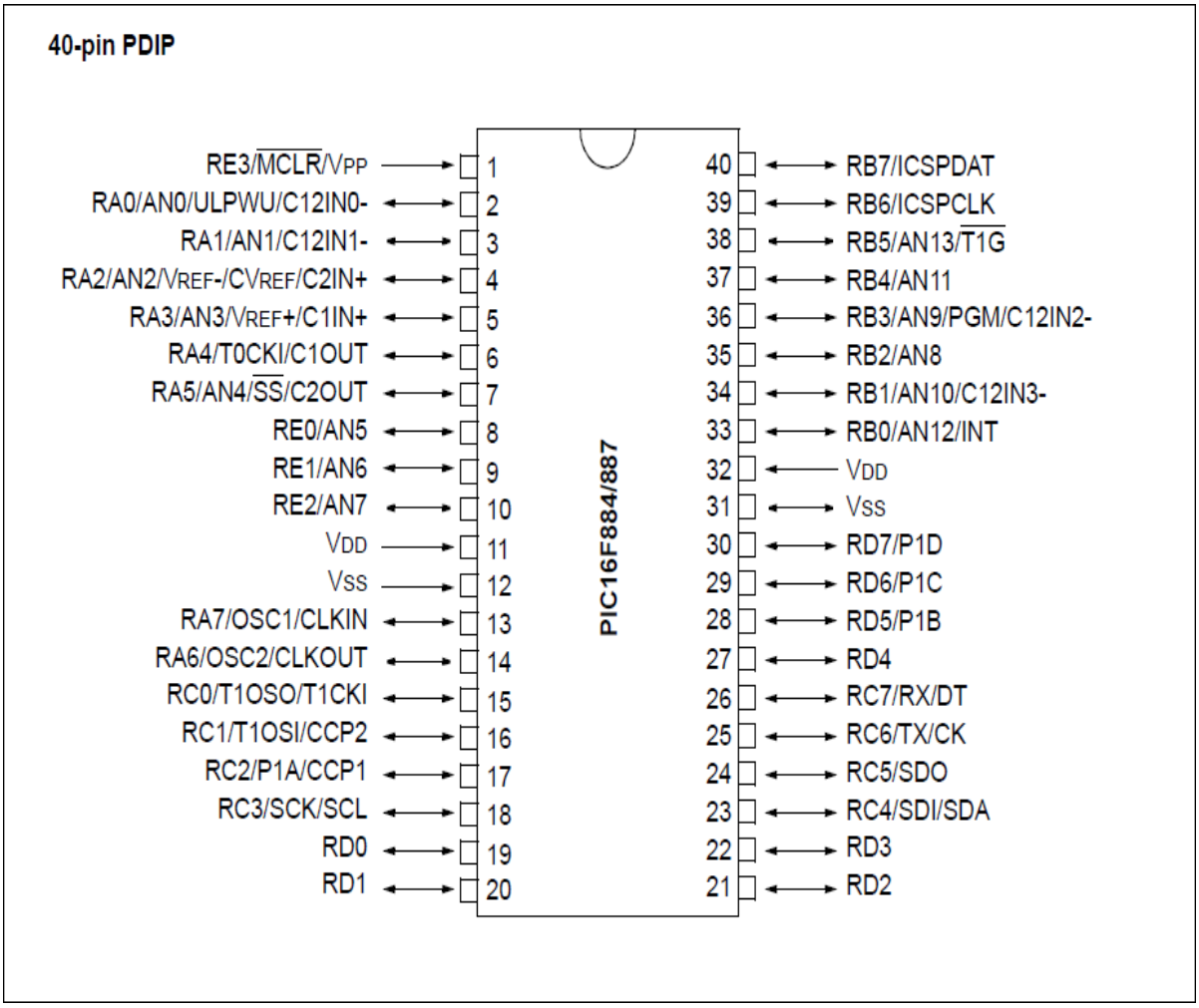
+ TIMER0: 8 bit của bộ định thời, bộ đếm với hệ số tỉ lệ trước.

+ TIMER1: 16 bit của bộ định thời, bộ đếm với tỉ số tỉ lệ trước, có khả năng tăng trong khi ở chế độ Sleep qua xung đồng hồ cung cấp bên ngoài.
+ TIMER2: 8 bit của bộ định thời, bộ đếm với 8 bit của hệ số tỉ lệ trước, hệ số tỉ lệ sau
+ Bộ chuyển đổi tín hiệu số sang tín hiệu tương tự với 10 bit
+ Cổng truyền thông tin nối tiếp SSP với SPI phương thức chủ

2.Khảo sát vi điều khiển PIC16F887 của hãng Microchip

2.1 Sơ đồ chân của PIC16F887

a.Sơ đồ chân



Hình 1.1 Sơ đồ chân của PIC16F887

b.Chức năng chân của vi điều khiển PIC16F887

Port A: PortA(RA0 ÷ RA5) có số chân từ chân số 2 đến chân số 7.

PortA (RPA) bao gồm 6 I/O pin. Đây là các chân "hai chiều" (bidirectional pin),
nghĩa là có thể xuất và nhập được. Chức năng I/O này được điều khiển bởi thanh ghi TRISA (địa chỉ 85h)

Port B: PortB(RB0 ÷ RB7) có số chân từ chân số 33 đến chân số 40.

PortB (RPB) gồm 8 pin I/O. Thanh ghi điều khiển xuất nhập tương ứng là TRISB. Bên cạnh đó một số chân của PORTB còn được sử dụng trong quá trình nạp chương trình cho vi điều khiển với các chế độ nạp khác nhau. PORTB còn liên quan đến ngắt ngoại vi và bộ Timer0. PORTB còn được tích hợp chức năng điện trở kéo lên được điều khiển bởi chương trình.

Port C: PortC(RC0 ÷ RC7) có số chân từ chân số 15 đến chân số 18 và chân số 23 đến chân số 26.

PortC (RPC) gồm 8 pin I/O. Thanh ghi điều khiển xuất nhập tương ứng là TRISC. Bên cạnh đó PORTC còn chứa các chân chức năng của bộ so sánh, bộ Timer1, bộ PWM và các chuẩn giao tiếp nối tiếp I2C, SPI, SSP, USART

Port D: PortD(RD0 ÷ RD7) có số chân từ chân số 33 đến chân số 40.

PortD (RPD) gồm 8 chân I/O, thanh ghi điều khiển xuất nhập tương ứng là TRISD. PORTD còn là cổng xuất dữ liệu của chuẩn giao tiếp PSP (Parallel Slave Port).

Port E: PortE(RE0 ÷ RE2) có số chân từ chân số 19 đến chân số 22 và chân số 27 đến chân 30.

PortE (RPE) gồm 3 chân I/O. Thanh ghi điều khiển xuất nhập tương ứng là TRISE. Các chân của PORTE có ngõ vào analog. Bên cạnh đó PORTE còn là các chân điều khiển của chuẩn giao tiếp PSP

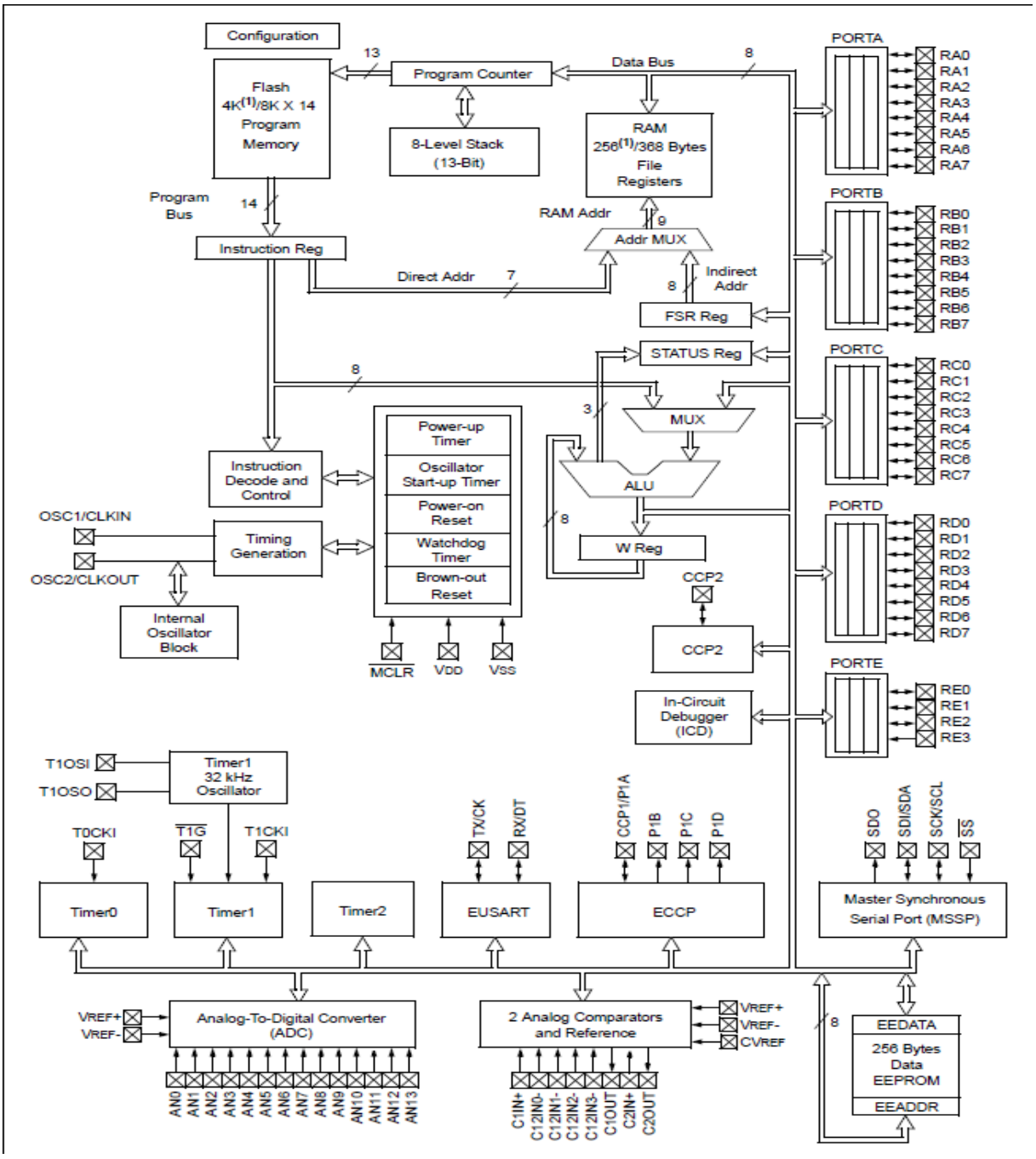
Chân 11,12,31,32 là các chân cung cấp nguồn cho vi điều khiển.

Chân 13,14 là chân được đấu nối thạch anh với bộ dao động xung clock bên ngoài cung cấp xung clock cho chip hoạt động.

Chân 1 là chân RET: Là tín hiệu cho phép thiết lập lại trạng thái ban đầu cho hệ thống, và là tín hiệu nhập là mức tích cục cao.

2.2 Sơ đồ khối

2.2.1 Sơ đồ



Hình 1.2 Sơ đồ khối

2.2.2 Tổ chức các bộ nhớ.

Cấu trúc của bộ nhớ vi điều khiển PIC16F887 bao gồm 2 bộ nhớ:

+ Bộ nhớ chương trình (Programmemory).
+ Bộ nhớ dữ liệu (Data memory).

a.Bộ nhớ chương trình (Programmemory).

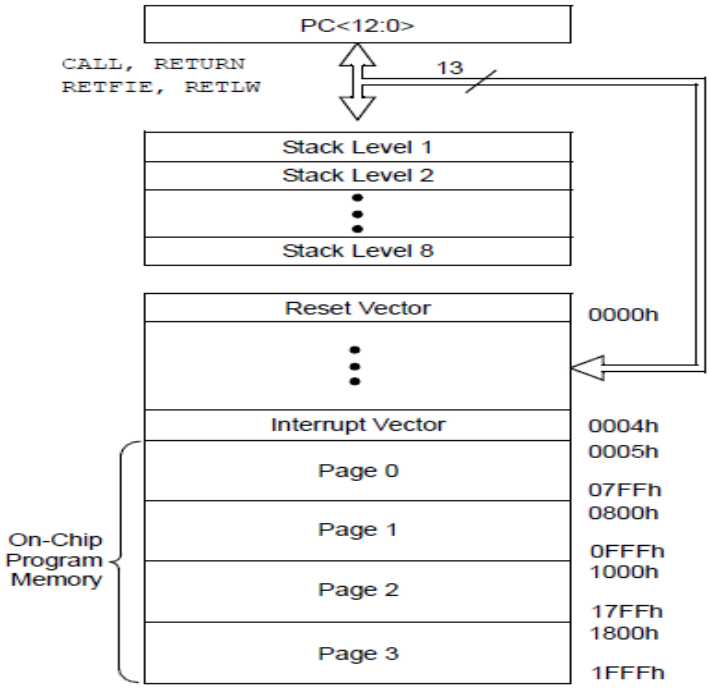
Bộ nhớ chương trình của vi điều khiển PIC16F887 là bộ nhớ Flash dung lượng bộ nhớ 8K được phân chia thành nhiều trang(từ 0÷ 3). Như vậy bộ nhớ chương trình có khả năng chứa được 8*1024=8192 câu lệnh.

Để mã hóa được địa chỉ của 8K bộ nhớ chương trình, bộ đếm chương trình có dung lượng 13bit.

Khi vi điều khiển được Reset, bộ đếm chương trình chỉ đến địa chỉ 0004h(Interrupt vector).

Bộ nhớ chương trình không bao gồm bộ nhớ stach và không được địa chỉ hóa bởi bộ đếm chương trình.

Bảng bộ nhớ chương trình và các ngăn xếp



Hình 1.3 Bộ nhớ chương trình 16F8887

b.Bộ nhớ dữ liệu (Data memory).

Bộ nhớ dữ liệu của PIC là bộ nhớ EEPROM được chia thành nhiều bank. Đối với PIC16F887 chia thành 4 bank. Mỗi bank có dung lượng chứa 128 byte, bao gồm các thanh ghi có chức năng đặc biệt SFG (Spencial Function Register) nằm ở các vùng địa chỉ thấp và các thanh ghi mục đích chung GPR(General Purpose Register) nằm ở các vùng địa chỉ còn lại trong back. Các thanh ghi SFG thường xuyên được sử dụng sẽ được đặt ở tất cả các bank của bộ nhớ dữ liệu giúp truy suất và làm giảm bớt lệnh của chương trình.

Bộ nhớ dữ liệu của PIC 16F887

	File Address		File Address		File Address		File Address
Indirect addr. (1)	00h	Indirect addr. (1)	80h	Indirect addr. (1)	100h	Indirect addr. (1)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	WDTCON	105h	SRCON	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	CM1CON0	107h	BAUDCTL	187h
PORTD(2)	08h	TRISD(2)	88h	CM2CON0	108h	ANSEL	188h
PORTE	09h	TRISE	89h	CM2CON1	109h	ANSELH	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2(1)	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved	18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh	Reserved	18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUB	95h		115h		195h
CCPR1H	16h	IOCB	96h		116h		196h
CCP1CON	17h	VRCON	97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah	SPBRGH	9Ah		11Ah		19Ah
CCPR2L	1Bh	PWM1CON	9Bh		11Bh		19Bh
CCPR2H	1Ch	ECCPAS	9Ch		11Ch		19Ch
CCP2CON	1Dh	PSTRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Registers 96 Bytes	20h – 7Fh	General Purpose Registers 80 Bytes	A0h – EFh	General Purpose Registers 80 Bytes	120h – 16Fh		1A0h – 1EFh
		accesses 70h-7Fh	F0h – FFh	accesses 70h-7Fh	170h – 17Fh	accesses 70h-7Fh	1F0h – 1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

Hình 1.4 Bộ nhớ data 16F887

2.2.3 Các thanh ghi đặc biệt

Các thanh ghi được sử dụng bởi CPU hoặc dùng đẻ thiết lập điều khiển các khối chức năng tích hợp trong vi điều khiển. Phân chia thanh SFR làm hai loại: thanh ghi SFR liên quan đến các chức năng bên trong CPU và thanh ghi SFR dùng để thiết lập và điều khiển các khối chức năng bên ngoài.

a.Các thanh ghi liên quan đến bên trong:

-Thanh ghi SATUS (03h, 83h, 103h, 183h): thanh ghi chứa kết quả thực hiện phép toán của khối ALU, trạng thái RESET và cấc bit chọn bank cùng truy suất trong bộ nhớ dữ liệu.

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC(1)	C(1)
bit 7							bit 0

-Thanh ghi OPTION_REG (81h, 181h): thanh ghi này cho phép đọc và ghi, cho phép điều khiển các chức năng puled ma trận-up của các chân PORTB, xác lập các tham soosxung tác động, cạnh tác động của ngắt ngoại vi và bộ đếm Time0

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

-Thanh ghi INTCON (0Bh, 8Bh, 10Bh, 18Bh): thanh ghi cho phép đọc và ghi, chứa các bit điều khiển và các bít cờ hiệu khi Time0 tràn, ngắt ngoại vi RB0/INT và ngắt interrupt-on-change tại các chân của PORTB.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE(1)	T0IF(2)	INTF	RBIF
bit 7							bit 0

-Thanh ghi PIE1 (8Ch): chứa các bit điều khiển chi tiết các ngắt của khối chức năng ngoại vi

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

-Thanh ghi PIR1 (0Ch) chứa cờ ngắt của các khối chức năng ngọai vi,các ngắt này được cho phép bởi các bit điều khiển chứa trong thanh ghi PEI1.

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

-Thanh ghi PIE2 (8Dh): chứa các bit điều khiển các ngắt của các khối chức năng CCP2, SSP bus, ngắt của bộ so sánh và ngắt ghi vào bộ nhớ EEPROM

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
OSFIE	C2IE	C1IE	EEIE	BCLIE	ULPWUIE	—	CCP2IE
bit 7							bit 0

-Thanh ghi PIR2 (0Dh): chứa các cờ ngắn của của các khối chức năng ngoại vi các ngắt này được cho phép bởi các bit điều khiển chứa trong thanh ghi PIE2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
OSFIF	C2IF	C1IF	EEIF	BCLIF	ULPWUIF	—	CCP2IF
bit 7							bit 0

-Thanh ghi PCON (8Eh): chứa các cờ hiệu cho biết trạng thái các chế độ Reset của vi điều khiển.

U-0	U-0	R/W-0	R/W-1	U-0	U-0	R/W-0	R/W-x
—	—	ULPWUE	SBOREN(1)	—	—	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

b.Thanh ghi mục đích chung GPR

Các thanh ghi này có thể truy suất trực tiếp hoặc gián tiếp thông qua thanh ghi FSG (File Select Rersister). Đây là các thanh ghi dữ liệu thông thường, người sử dụng có thể tùy theo mục đích chương trình mà có thể dùng các thanh ghi này để chứa các biến số, hằng số, kết quả hoặc các tham số phục vụ cho chương trình.

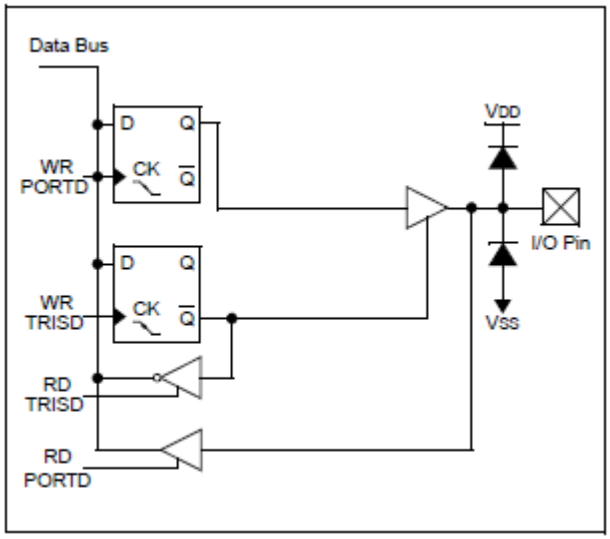
2.2.4 Các cổng xuất nhập của PIC(I/O)

PIC16F887 tất cả có 35 chân I/O mục đích thông thường(GPIO: GeneralPurpose Input Ouput) có thể được sủ dụng. Tùy theo những thiết bi ngoại vi được chọn mà một vài chân không thể sủ dụng ở chức năng GPIO. Thông thường, khi một thiết bị ngoai vi được chọn, những chân liên quan của thiết bị ngoại vi không được sủ dụng ở chức năng GPIO.35 chân được chia thành 5 port:

+ PortA chia làm 8 chân.
+ PortB chia làm 8 chân.
+ PortC chia làm 8 chân.
+ PortD chia làm 8 chân.
+ PortE chia làm 3 chân.

Mỗi port được điều khiển bởi 2 thanh ghi 8-bit, thanh ghi Port và thanh ghi Tris. Thanh ghi Tris được sử dụng để điều khiển port nhập hay xuất. Mỗi bit của Tris sẽ điều khiển mỗi chân của port đó, nếu giá trị bit là 1 thì chân liên quan là nhập, ngược lại nếu giá trị bit là 0 thì chân liên quan là xuất. Thanh ghi Port được suwe\r dụng để chứa các giá trị của port liên quan. Mỗi bit của thanh ghi Port chứa giá trị của chân liên quan.

Cấu trúc của GPIO:



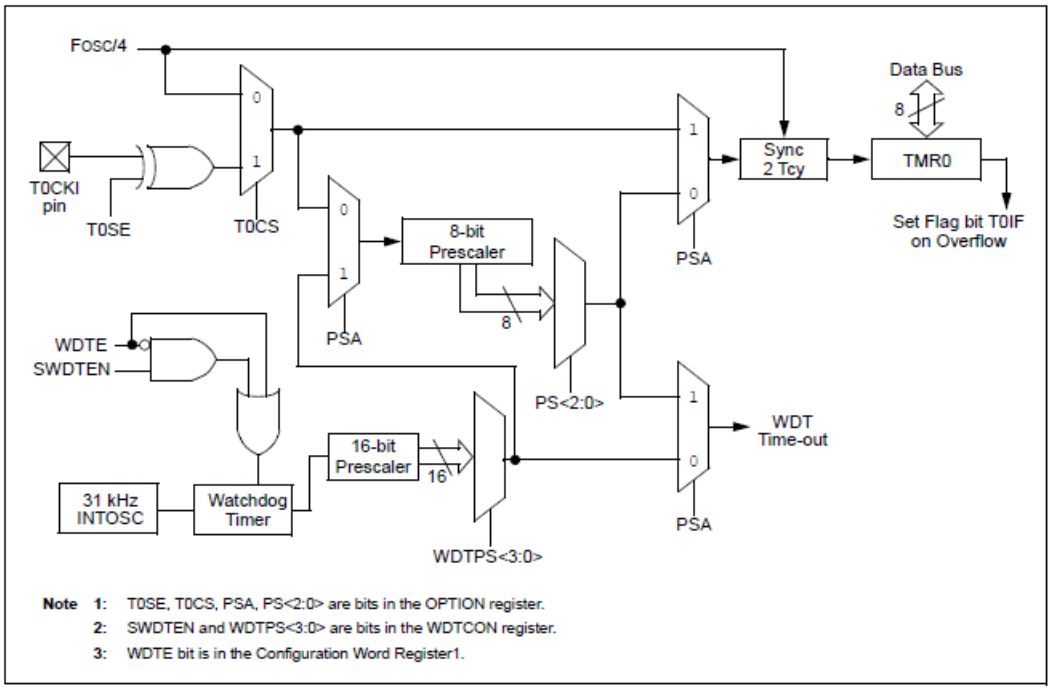
Hình 1.5 Cấu trúc của GPIO

2.2.5 Các bộ định thời của chip

Bộ vi điều khiển PIC16F887 có 3 bộ định thời Timer đó là Tmer0, Timer1, Timer2

a.Bộ Timer0

Đây là một trong 3 bộ đếm hoặc bộ định thời của vi điều khiển PIC16F887. Timer0 là bộ đếm 8 bit được kết nối với bộ chia tần 8bit. Cấu trúc của Time0 cho phép ta lựa chọn xung clock tác động và cạnh tích cực của xung clock. Ngắt Timer0 sẽ xuất hiện khi Timer0 bị tràn. Bit TMR0IE (INTCON<5>) là bit điều khiển của Timer0. TMR0IE=1 cho phép ngắt Timer0 tác động, TMR0IE=0 không cho phép ngắt Timer0 tác động



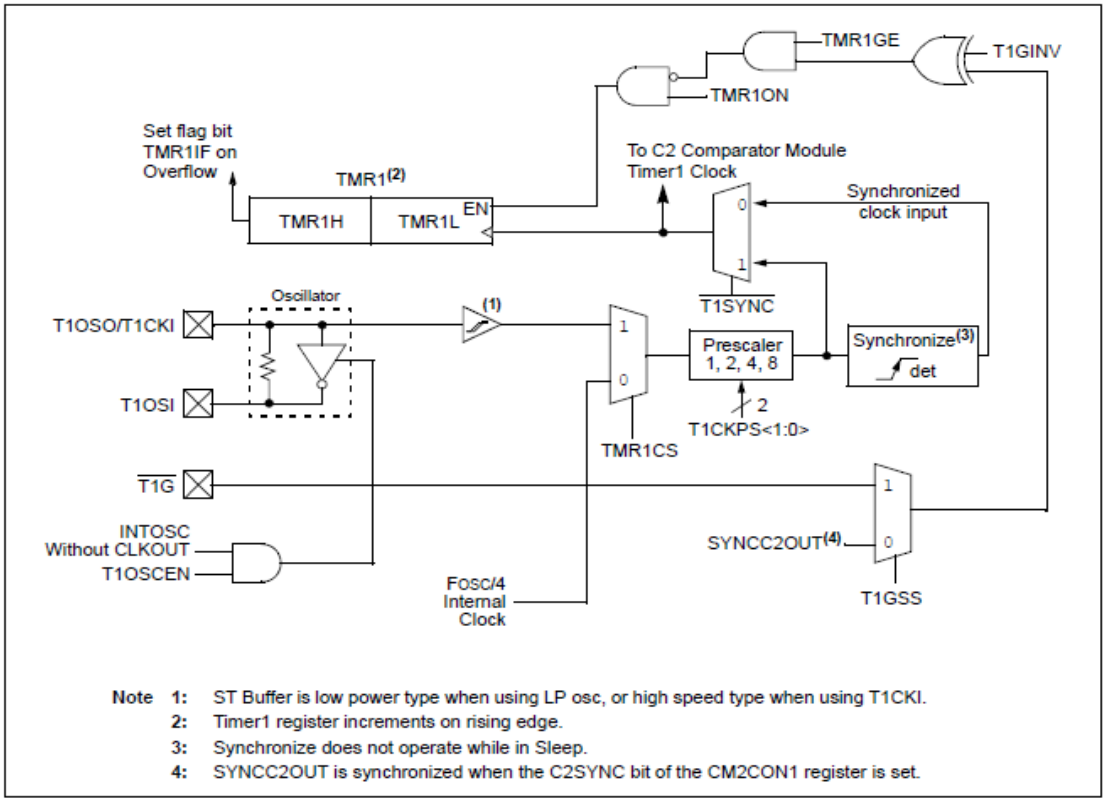
Hình 1.6 Sơ đồ khối của Timer0

b.Bộ Time1

Bộ Timer1 là bộ định thời 16bit, giá trị của Timer1 sẽ được lưu trong thanh ghi (TMR1H:TMR1L). Cờ ngắt của Timrer1 là bit TMR1IF. Bit điều khiển của Timer1 là TMR1IE

Tương tự như Timer0, Timer1 cũng có 2 chế độ hoạt động: chế độ định thời và chế độ xung kích là xung clock của osciled ma trậnator (tần số Timer bằng $^1/_4$ tần số của osciled ma trậnator và chế độ đếm (counter) với xung kích là xung phản ánh các sự kiện cần đếm lấy từ bên ngoài thông qua chân RCO/T1OSO/T1CKI (cạnh tác động là cạnh bên). Việc lựa chọn chế độ hoạt động của Timer được điều khiển bởi bit TMR1CS.

FIGURE 6-1: TIMER1 BLOCK DIAGRAM

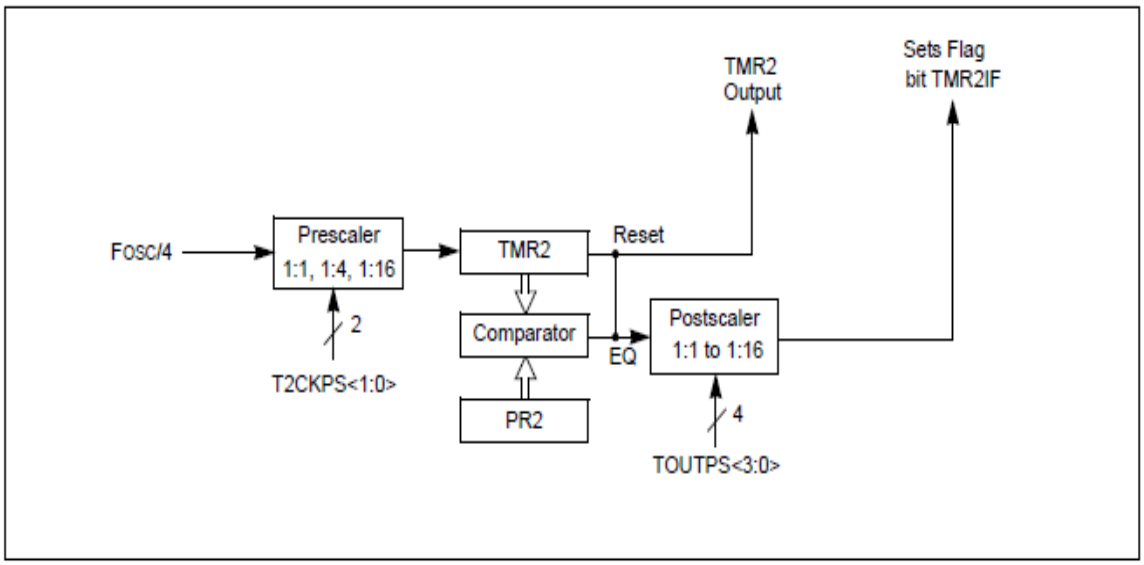


Hình 1.7 Sơ đồ khối của Timer1

c.Bộ Timer2

Bộ Timer2 là bộ định thời 8 bit và được hỗ trợ hai bộ chia tần prescaler và postscaler. Thanh ghi chứa giá tị đếm của Timer2 lafTMR2. Bit cho phép ngắt Timer2 tác động là TMR2ON. Cờ ngắt của Timer2 là bit TMR2IF. Xung ngõ vào được đưa qua bộ chia tần số prescaler 4 bit(với các tỉ số chia tần 1:1, 1:4 hoặc 1:6) và được điều khiển bởi các bit T2CKPS1:T2CKPS0

FIGURE 7-1: TIMER2 BLOCK DIAGRAM

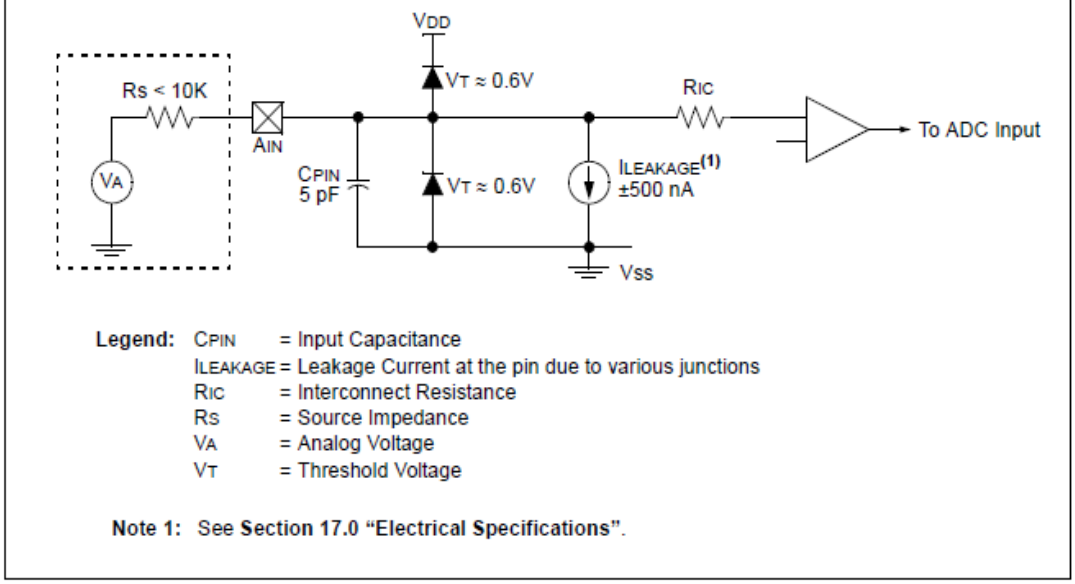


Hình 1.8 Sơ đồ khối của Timer2

d.Bộ biến đổi ADC

ADC (Analog to Digital Converter) là bộ chuyển đổi tín hiệu giữa hai dạng tương tự và số.PIC16F887 có 14 ngõ vào analog (RA5:RA0, RE2:RE0và RB5:RB0). Hiệu điện thế chuẩn V_{REF} có thể được chọn là V_{dd}, V_{ss} hay hiệu điện thế chuẩn được xác lập trên 2 chân RA2 và RA3. Kết quả chuyển đổi từ tin hiệu tương tự sang tín hiệu số là 10bit tương ứng và được lưu trong hai thanh ghi ADRESH:ADRESL. Khi không sử dụng bộ chuyển đổi ADC các thanh ghi này có thể sử dụng các thanh ghi thông thường khác. Khi quá trình chuyển đổi hoàn tất, kết quả sẽ được lưu vào 2 thanh ghi ADRESH:ADRESL.

FIGURE 8-6: ANALOG INPUT MODEL



Hình 1.9 Sơ đồ khối bộ chuyển đổi Analog

2.3 Các ứng dụng của PIC16F887

2.3.1 Giao tiếp với máy tính

PIC kết nối với máy tính thông qua cổng nối tiếp IC Max232. Ghép nối qua cổng RS232 là một trong những kỹ thuật sử đụng rộng rãi nhất để ghép nối với thiết bị ngoại vi với máy tính.

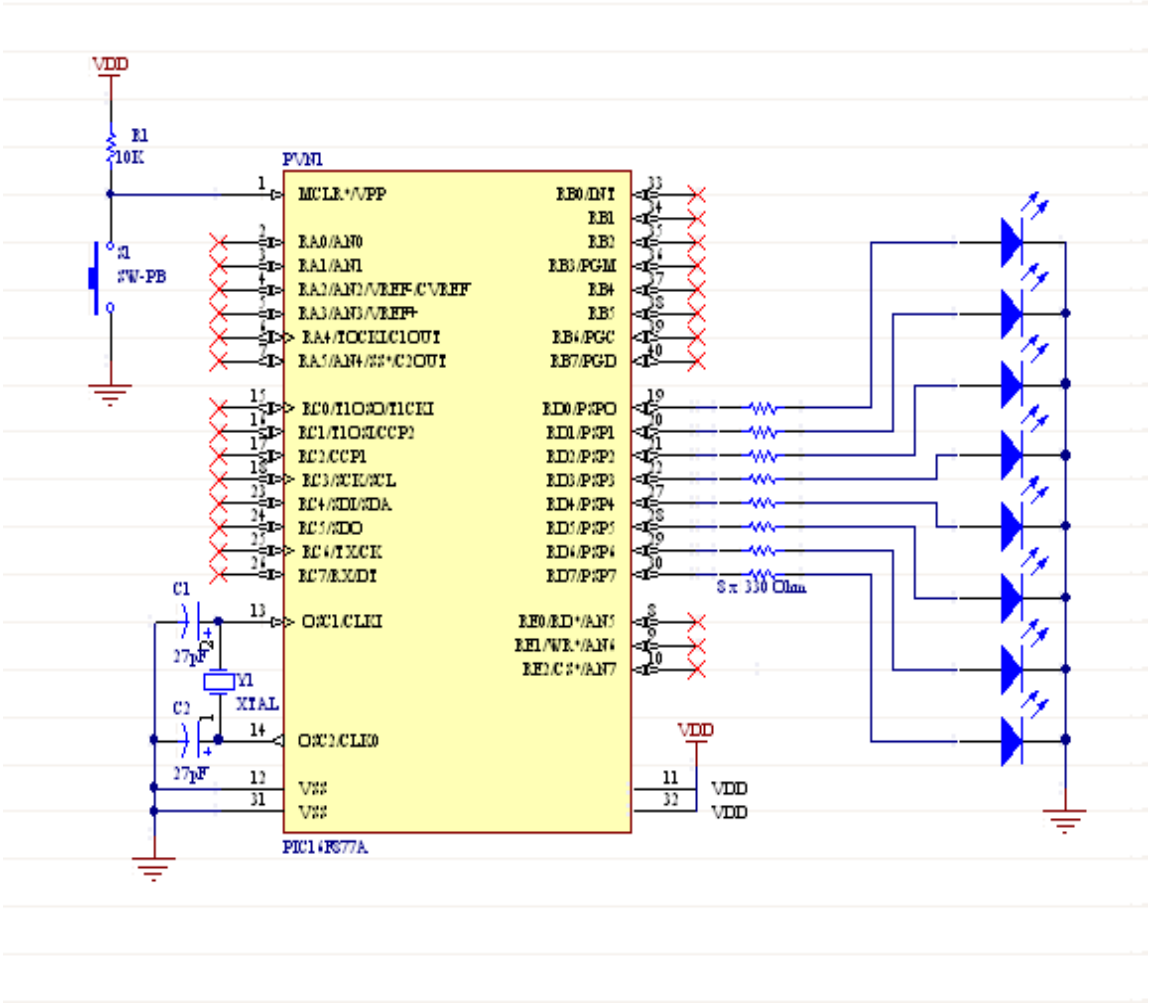
Ưu điểm:

+ Khả năng chống nhiễu của các cổng nối tiếp cao

+ Thiết bị ngoại vi có thể lắp ráp ngay khi máy tính đang cấp điện

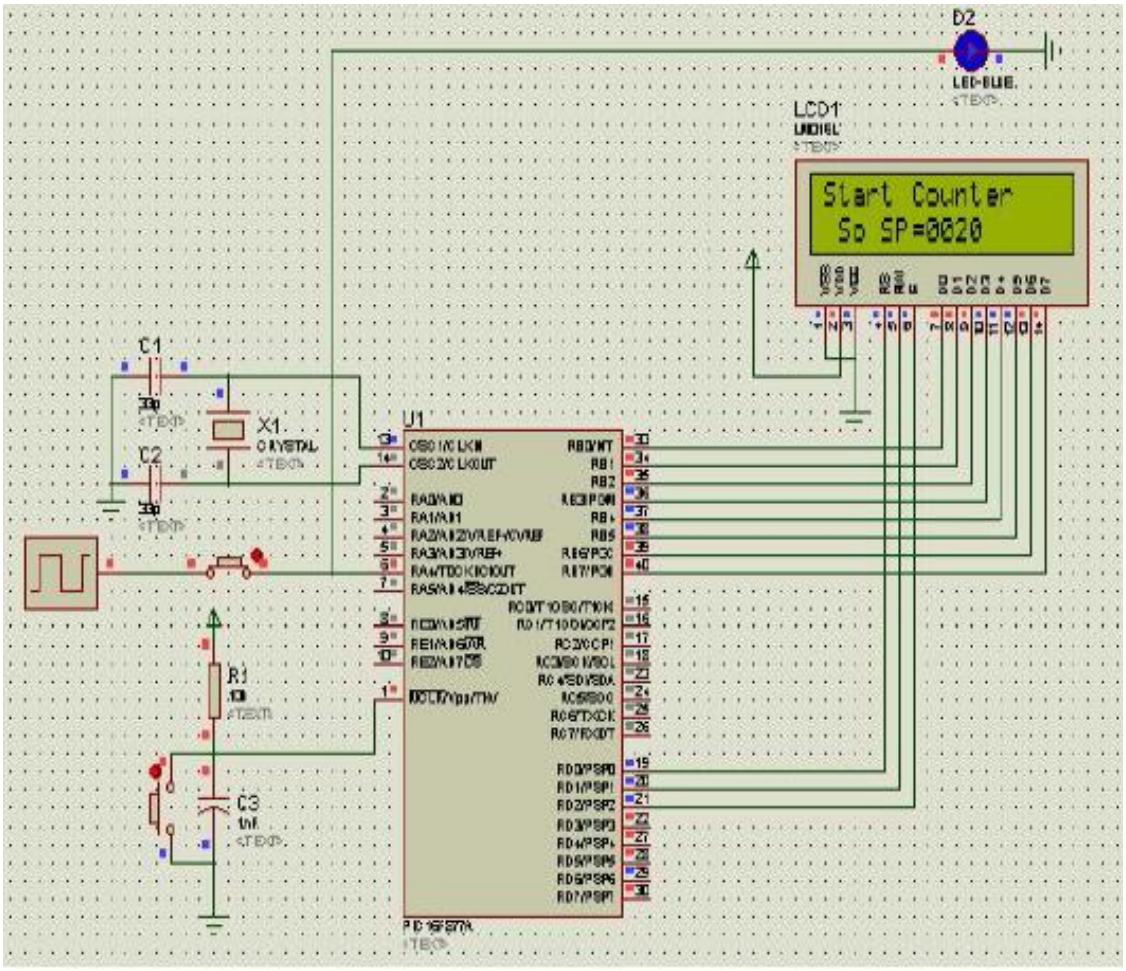
+ Các mạch điện đơn giản có thể nhận được điện áp nguồn nuôi qua cổng nối tiếp

2.3.2 Giao tiếp với led



Hình 1.10 Sơ đồ giao tiếp với Led

2.3.3.Giao tiếp với LCD



Hình 1.11 Sơ đồ giao tiếp với LCD

II.CÁC PHƯƠNG PHÁP ĐO KHOẢNG CÁCH

1.Đo thủ công

Đo thủ công bằng các loại thước đo đơn giản, độ chính xác khá cao nhưng phụ thuộc nhiều vào người đo.

2.Sử dụng Lase để đo khoảng cách

Đo khoảng cách dựa trên nguyên lý điều biến pha.

Nguyên lý chung của phương pháp điều biến pha là đo độ khác biệt giữa pha của ánh sáng phát ra với ánh sáng nhận được sau khi phản hồi từ vật. Tuy nhiên không có một loại photodetector nào có thể đáp ứng được với sự thay đổi của tần số trực tiếp của ánh sáng lên đến 100THz. Vì thế phương pháp điều biến pha sử dụng tần số trực tiếp của ánh sáng không thể nào tạo ra được. Do đó để có thể dễ dàng đo được thì ta phải điều biến tần số ánh sáng theo một tần số thấp hơn mà các linh kiện thu quang học và mạch điện tử còn có thể đáp ứng được. Vì thế tôi dùng phương pháp điều biến sóng sin để điều khiển laser diot và sử dụng photodetector để thu ánh sáng laser phản hồi.

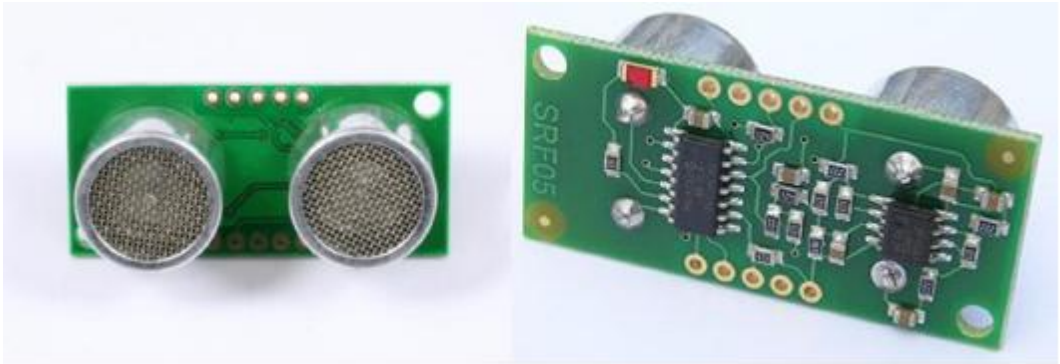
3. Phương pháp đo khoảng cách bằng sóng siêu âm bằng cảm biến SRF05

Siêu âm là dạng sóng âm được ứng dụng rộng rãi trong việc đo khoảng cách và định vị vật thể. Báo cáo giới thiệu một phương pháp đo khoảng cách và xác định vị trí vật thể bằng sóng siêu âm với sự kết hợp phương pháp xác suất Bayes trong xử lý tín hiệu. Phương pháp sử dụng công thức xác suất toàn phần Bayes để đánh giá khả năng không gian bị chiếm bởi vật thể, và tỉ lệ chiếm giữa các ô lưới trên bản đồ nhằm xác định vị trí xác suất cao nhất có vật thể.

Sóng siêu âm được truyền đi trong không khí với vận tốc khoảng 343m/s. Nếu một cảm biến phát ra sóng siêu âm và thu về các sóng phản xạ đồng thời, đo được khoảng thời gian từ lúc phát đi tới lúc thu về, thì máy tính có thể xác định được quãng đường mà sóng đã di chuyển trong không gian. Quãng đường di chuyển của sóng sẽ bằng 2 lần khoảng cách từ cảm biến tới chướng ngoại vật, theo hướng phát của sóng siêu âm.

3.1Cảm biến SRF05

SRF05 là bước phát triển từ SRF04, được thiết kế đẻ làm tính năng linh hoạt, tăng ngoại vi, ngoài ra còn giảm chi phí. SRF05 là hoàn toàn tương thích với SRF04. Khoảng cách tăng từ 3m đến 4m. Một chế độ hoạt động mới SRF05 cho phép sử dụng một chân duy nhất cho cả kích hoạt và phản hồi, do đó sẽ tiết kiệm có giá tri trên chân điều khiển. Khi chân chế độ không kết nối, SRF05 hoạt động riêng biệt chân kích hoạt và chân hồi tiếp. SRF05 bao gồm một thời gian trễ trước khi xung phản hồi đẻ mang lại điều khiển chậm hơn

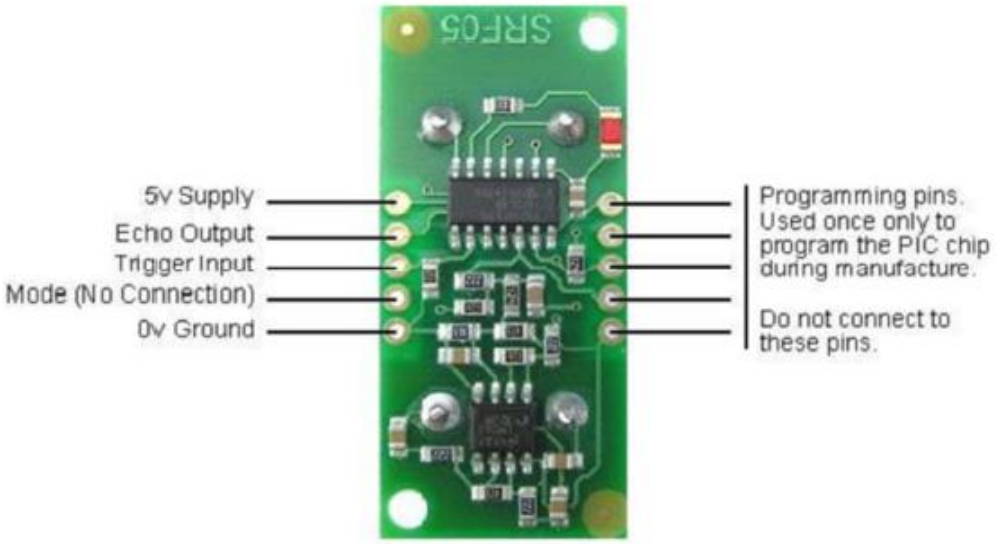


Hình 2.1 Cảm biến SRF05

Cảm biến SRF05 thiết lập 2 mode haotj đọng khac nhau thông qua các chang điều khiển MODE.Nối hoặc không nối chân MODE xuộng MASS cho phép cảm biến thông qua giao tiếp dùng một chân hay 2 chân I/O.

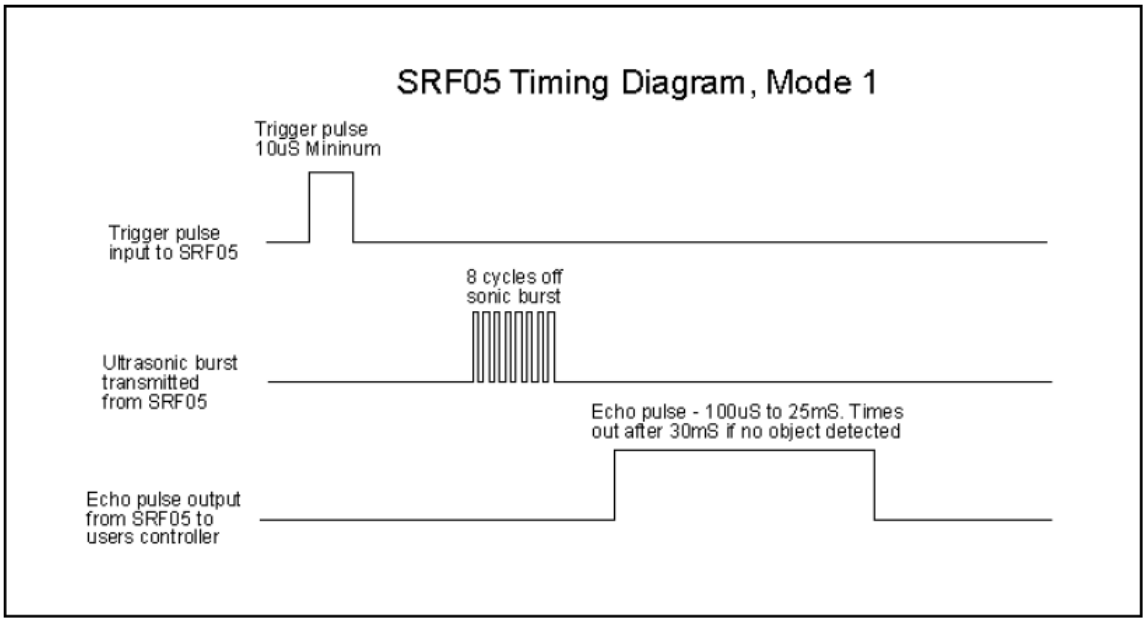
-Mode1: Tách chân **TRIGGER & ECHO** dùng riêng:

Trong mode này SRF05 sử dụng cả hai chân TRIGGER và ECHO cho việc giao tiếp với CPU. Để sử dụng mode này ta chỉ cần đẻ chống chân Mode của module, điện trở bên trong module sẽ kéo chân pin lên mức 1



Hình 2.2 Cấu hình SRF05 mode1

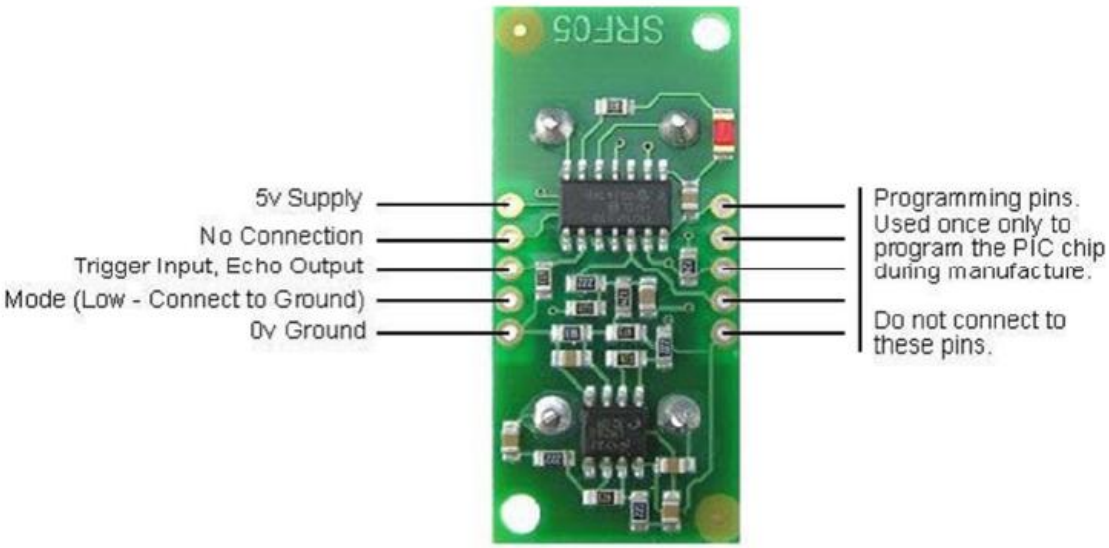
Để điều khiển SRF05, ta chỉ cần cấp cho chân TRIGGER một xung điều khiển với độ rộng tối thiểu 10uS. Sau đó một klhoangr thời gian, đầu phát song siêu âm sẽ phát ra song siêu âm, vi xử lý tích hợp trên module sẽ xác định thời điểm phát song siêu âm và thu song siêu âm. Vi xử lý tích hợp này sẽ đưa kết quả thu được ra chân ECHO. Độ rộng xung vông tại chân ECHO tỉ lệ với khoảng cách từ cảm biến đến vật thể.



Hình 2.3 Nguyên lý hoạt động SRF05 ở mode1

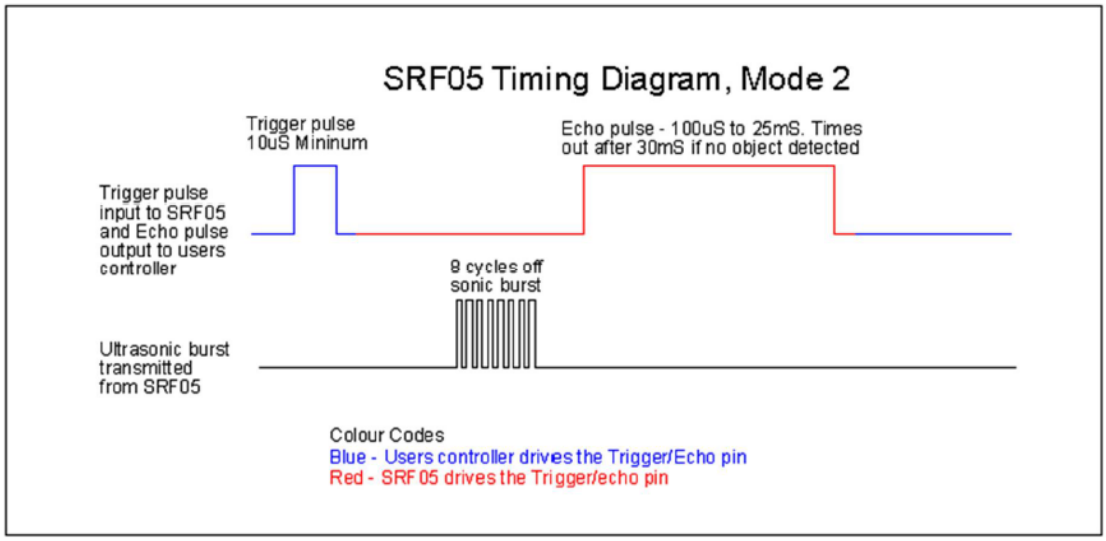
-Mode2 Chân TRIGGER & ECHO dùng chung

Được thiết kế nhằm cho mục đích tiết kiệm chân pin cho MCU nên trong mode này, SRF05 chỉ sử dụng chân pin cho 2 chức năng TRIGGER và ECHO. Để sử dụng mode này, ta kết nối chân MODE xuong MASS.Đây cũng chính là mode sẽ được sử dụng trong demo.



Hình 2.4 Cấu hình SRF05 mode2

Để điều khiển SRF05, đầu tiên xuất một xung với độ xung tối thiểu 10uS vào chân TRIGGER – ECHO(chân số 3) của cảm biến. Sau đó vi sử lý tích hợp trên cảm biến sẽ phát ra tín hiệu điều khiển phát siêu âm. Sau 700uS kể từ lúc kết thúc tín hiệu điều khiển từ chân TRIGGER – ECHO có thể đọc ra một xung mà độ rộng tỉ lệ với khoảng cách từ cảm biến tới vật thể.



Hình 2.5 Nguyên lý hoạt động SRF05 ở mode 2

3.2 Tính toán khoảng cách

Giản đồ định thời SRF05 thể hiện trên đây cho mỗi chế độ. Bạn chỉ cần cung

cấp một đoạn xung ngắn 10uS kích hoạt đầu vào để bắt đầu đo khoảng cách. Các SRF05 sẽ gửi cho ra một chu kỳ 8 burst của siêu âm ở 40khz và tăng cao dòng phản hồi của nó (hoặc kích hoạt chế độ dòng 2). Sau đó chờ phản hồi, và ngay sau khi phát hiện nó giảm các dòng phản hồi lại. Dòng phản hồi là một xung có chiều rộng là tỷ lệ với khoảng cách đến đối tượng. Bằng cách đo xung, ta hoàn toàn có thể để tính toán khoảng cách ttheo inch / centimét hoặc bất cứ điều gì khác. Nếu không phát hiện gì cả SRF05 giảm thấp hơn dòng phản hồi của nó sau khoảng 30mS.

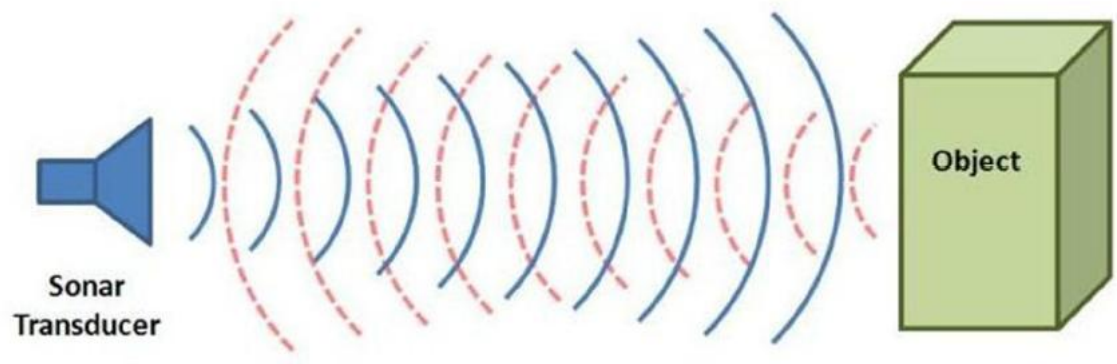
SRF04 cung cáp một xung phản hồi tỷ lệ với khoang cách. Nếu độ rộng của pulse được đo trong hệ uS, sau đó chia cho 58 sẽ cho khoảng cách theo cm, hoặc chia cho 148 sẽ cho khoảng cách theo inch. uS/58 = cm hay uS/148 = inch.

SRF05 có thể được kích hoạt nhanh chóng với mọi 50mS, hoặc 20 lần mỗi giây. Bạn nên chờ 50ms trước khi kích hoạt kế tiếp, ngay cả khi SRF05 phát hiện một đối tượng gần và xung phản hồi ngắn hơn

Có 4 chân ra khỏi các môđun: VCC, Trig, Echo, GND. Vì vậy đó là một giao diện rất dễ dàng cho bộ điều khiển sử dụng nó khác nhau. Quá trình tất cả là: kéo pin của Tring mức độ hơn 10us xung, mô đun khác nhâu , bắt đầu , kết thúc khác nhau. Nếu thấy một đối tượng ở phía trước, Echo pin cao cấp, và dựa vào khoảng khác nhau nó sẽ có thời gian khác nhau.

3.3 Hoạt động và nhận phản hồi sóng cơ bản của SRF05

+Nguyên tắc cơ bản của sonar: là tạo ra một xung âm thanh sau đó lắng nghe tiếng vọng tạo ra khi các làn xóng âm thanh truy cập một đối tượng và được phản xạ trở lại. Để tính thời gian phẩn hồi trở về, một ước tính chính xác có thể làm được bằng khoảng cách tới đối tượng. Xung âm thanh tạo ra bởi SRF05 là siêu âm, nghĩa là nó ở trên phạm vi nhận xét của côn người. Trong khi tần số thấp hơn có thể được sủ dụng trong các loại ứng dụng, tần số cao hơn thực hiện tốt hơn cho phạm vi ngắn, nhu cầu độ chính xác cao.



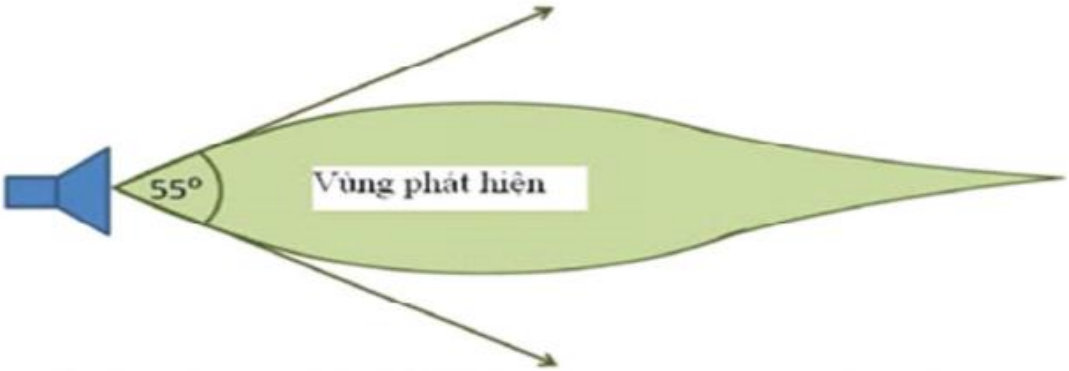
Hình 2.6 Nguyên lý hoạt động của SRF05

3.4 Một số đặc điểm khác của cảm biến siêu âm.

+Mức độ của sóng âm hồi tiếp phụ thuộc vào cấu tạo của đối tượng và góc phản xạ của nó.

Hình 2.7 Mức độ của sóng âm hồi tiếp

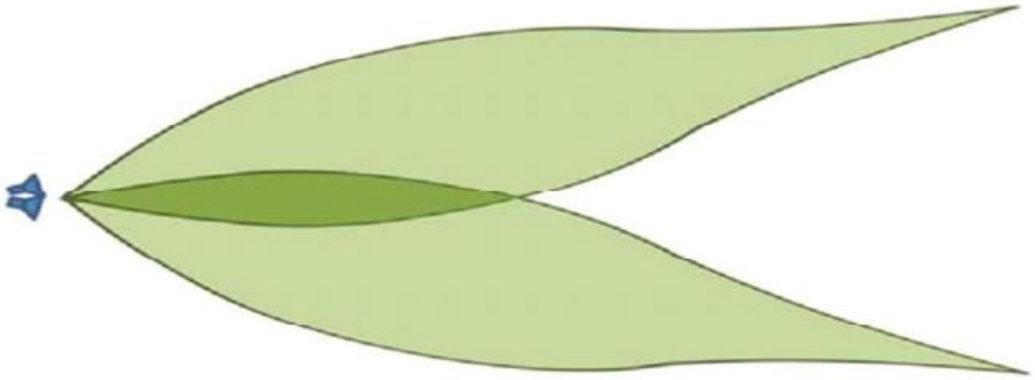
+Một số đối tượng mềm có thể cho ra tín hiệu phản hồi yếu hoặc không có phản hồi. Một số đối tượng ở một góc cân đối thì mới có thể chuyển thành tín hiệu phản chiếu một chiều cho cảm biến.



Hình 2.8 Vùng phát hiện của cảm biến

+Các vùng cảm biến của SRF05 nằm trong khoảng 1m chiều rộng từ bên này sang bên kia và hông quá 5m chiều dài.

+Một số kỹ thuật để làm giảm các điểm mù và đạt được phát hiện chiều rộng lớn hơn ở cụ ly gần là them một cải tiến bằng cách thêm một cảm biến bổ xung và gắn kết hai đơn vị hướng về phía trước. Thiết lập như vậy thì có một khu vực mà là hai khu vực chồng chéo lên nhau.



Hình 2.9 Hai cảm biến hoạt động

+Các vùng hoạt động của 2 cảm biến tạo góc chung 30 độ. Vùng chung thì được phân biệt bởi 2 phần tín hiệu trí phải và phần cản ở giữa.

III.Ứng dụng của ngôn ngữ lập trình Assembler, C điều khiển

1 Ngôn ngữ lập trình Assembler

Ngôn ngữ lập trình Assembler là một ngôn ngữ bậc thấp dùng trong việc viết các chương trình máy tính. Ngôn ngữ Assembler sử dụng các từ có thính chất gọi nhớ, các từ viết tắt giúp chúng ta ghi nhớ các chỉ thị phức tạp và làm cho việc lập trình Assembler được dễ dàng hơn. Mục đích việc dùng các từ gợi nhớ là nhằm thay thế việc lập trình trực tiếp bằng ngôn ngữ máy được sử dụng trong các máy tính đầu tiên thường gặp nhiều lỗi và tốn thời gian. Việc sử dụng ngôn ngữ Assembler vào lập trình PIC rất đơn giản,

-Ưu điểm.

+ Chạy nhanh

+ Tiết kiệm bộ nhớ.

+ Có thể lập trình truy cập qua các giao diện vào ra nhưng hiện nay các ngôn ngữ bậc cao cũng có thể làm được.

-Nhược điểm

+ Khó viết bởi yêu cầu người lập trình rất am hiểu về phần cứng

+ Khó tìm sai bao gồm cả cú pháp và sai về thuật toán

+ Không chuyển chương trình Assembler cho các máy tính có cấu trúc khác nhau.

2 Ngôn ngữ lập trình C

Ngôn ngữ lập trình C là ngôn ngữ lập trình tương đối nhỏ gọn vận hành gần với phần cứng và nó giống với ngôn ngữ Assember. C còn được đánh giá như là có khả năng di động, cho thấy sự khác nhau quan trọng giữ nó với ngôn ngữ bậc thấp như là Assember. C được tạo ra với một mục tiêu làm cho nó thuận tiện đẻ viết các chương trình lớn hơn với số lỗi ít hơn

-Ưu diểm

+ Tiết kiệm bộ nhớ

+ Câu lệnh được thực hiện nhanh

+ Cho phép người lập trình dễ dàng kiểm soát được những gì mà chương trình thực thi.

-Nhược điểm

+ Điều chỉnh bằng tay chậm hơn Assembler

CHƯƠNG III
THIẾT KẾ PHẦN CỨNG

Đề tài :"**Nghiên cứu thiết kế mạch đo khoảng cách dùng vi điều khiển PIC16F887**" bao gồm những phần chính sau :

-Sử dụng cảm biến siêu âm SRF05 để đo khoảng cách.

-Sử dụng RealTime DS1307 lấy thời gian lúc đo.

-Hiện thị kết quả đo được và thời gian đo lên LCD.

-Nguồn cung cấp sử dụng DC Adaptor 7 ÷ 12VDC.

I Các linh kiện trong đề tài.

1 Điện trở

Điện trở là đại lượng vật lý đặc trưng cho tính chất cản trở dòng điện của một vật dẫn điện. Nó được định nghĩa là tỉ số của hiệu điện thế giữa hai đầu vật thể với cường độ dòng điện đi qua nó.

Nó được xác định bởi công thức $R=U/I$

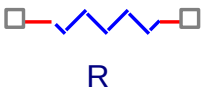
Trong đó:

U : là hiệu điện thế giữa hai đầu vật dẫn điện, đo bằng Vôn (V)

I : là cường độ dòng điện đi qua vật dẫn điện, đo bằng Ampe(A)

R : là điện trở của vật dẫn điện, đo bằng Ohm(Ω)

Ký hiệu:



Hình 3.1 Ký hiệu và hình dạng của điện trở

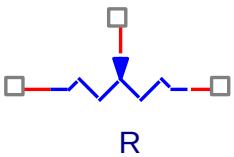
Điện trở là linh kiện thu động có tác dụng cản trở điện áp và dòng điện.

Điện trở được sử dụng rất nhiều trong mạch điện tử.

2 Biến trở.

Biến trở là điện trở thay đổi được, có tác dụng là thay đổi điện áp theo yêu cầu người sử dụng.

Ký hiệu:

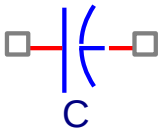


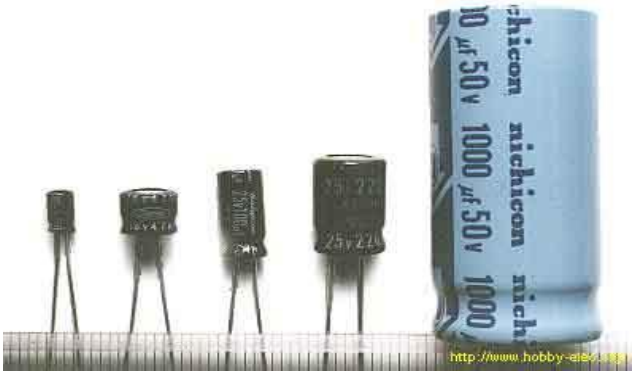
Hình 3.2 Ký hiệu và hình dạng của biến trở

3 Tụ điện

Tụ điện là linh kiện thụ động cấu tạo của tụ điện là hai bnr cực bằng kim loại ghép cách nhau một khoảng d ở giữa hai bản tụ là dung dịch hay chất điện môi cách điện có điện dung. Đặc điểm của tụ là cho dòng xoay chiều đi qua, ngăn cnar dòng điện một chiều.

Ký hiệu :

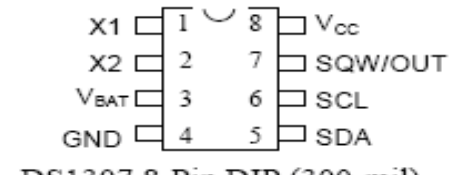




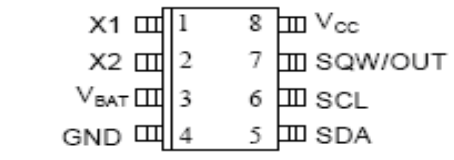
Hình 3.3 Ký hiệu và hình dạng của tụ điện

4 Bộ tạo xung chuẩn(xung clock).

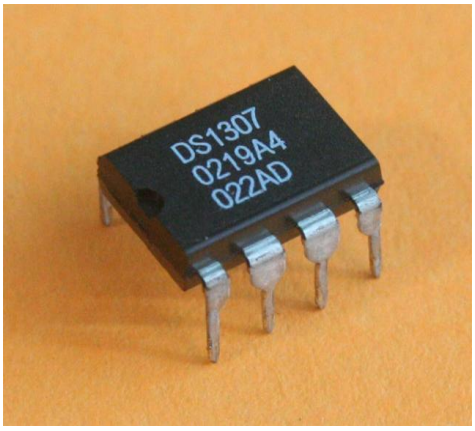
DS1307 là chip thời gian thực hay RTC (Read time clock). Đây là một IC tích hợp cho thời gian bởi vì tính chính xác về thời gian tuyệt đối cho thời gian : Thứ, ngày,tháng, năm, giờ, phút, giây. DS1307 là chế tạo bởi Dallas. Chip này có 7 thanh ghi 8 bit mỗi thanh ghi này chứa : Thứ , ngày, tháng, năm, giờ , phút, giây. Ngoài ra DS1307 còn chứa 1 thanh ghi điều khiển ngõ ra phụ và 56 thanh ghi trống các thanh ghi này có thể dùng như là RAM. DS1307 được đọc thông qua chuẩn truyền thông I2C nên do đó để đọc được và ghi từ DS1307 thông qua chuẩn truyền thông này. Do nó được giao tiếp chuẩn I2C nên cấu tạo bên ngoài nó rất đơn giản.



DS1307 8-Pin DIP (300-mil)



DS1307 8-Pin SOIC (150-mil)



Hình 3.4 Ký hiệu và hình dạng của DS1307

Trên là hai dạng cấu tạo của DS1307. Chip này có 8 chân và chúng ta hay dùng là dạng Dip và các chân nó được mô tả như sau :

+ **X1 và X2** là đầu vào dao động cho DS1307. Cần dao động thạch anh 32.768Khz.

+ **Vbat** là nguồn nuôi cho chip. Nguồn này từ (2V- 3.5V) ta lấy pin có nguồn 3V. Đây là nguồn cho chip hoạt động liên tục khi không có nguồn Vcc mà DS1307 vẫn hoạt động theo thời gian.

+ **Vcc** là nguồn cho giao tiếp I2C. Điện áp cung cấp là 5V chuẩn và được dùng chung với vi xử lý. Nếu mà Vcc không có mà Vbat có thì DS1307 vẫn hoạt động bình thường nhưng mà không ghi và đọc được dữ liệu.

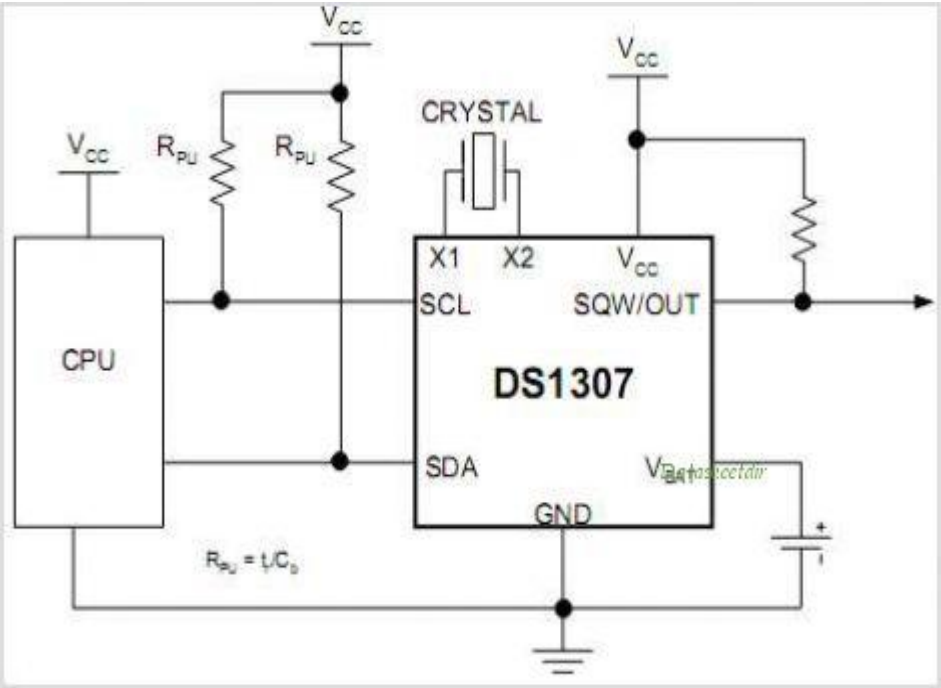
+ **GND** là nguồn Mass chung cho cả Vcc và Vbat.

+ **SQW/OUT** là một ngõ ra phụ tạo xung dao động (xung vuông). Chân này tôi nghĩ không ảnh hưởng đến thời gian thực nên chúng ta không sử dụng chân này trong thời gian thực và bỏ trống chân này!

+ **SCL và SDA** là hai bus dữ liệu của DS1307. Thông tin truyền và ghi đều được truyền qua 2 đường truyền này theo chuẩn I2C.

***Ghép nối DS1307 với vi điều khiển**

Do DS1307 giao tiếp chuẩn I2C nên việc ghép nối nó với vi điều khiển khá là đơn giản và theo sơ đồ sau :



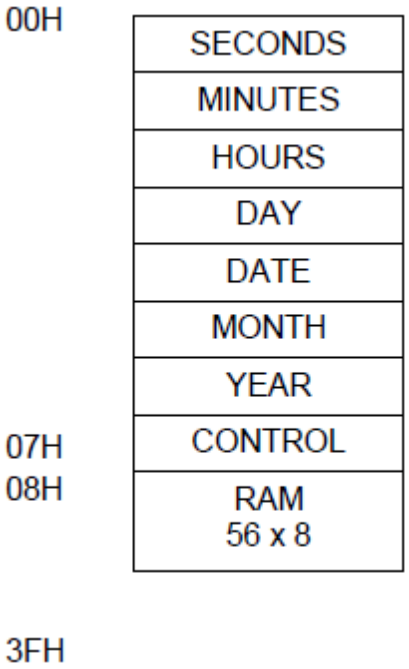
Hình 3.5 Ghép nối DS1307 với vi điều khiển

Ds1307 nó chỉ giao tiếp với vi điều khiển với 2 đường truyền SCL và SDA nên do đó trên vi xử lý cần phải xác định chân nào trên vi xử lý nó có SCL và SDA để nối với DS1307 cái này đối với dòng PIC, AVR.

***Thanh ghi trong DS1307**

Cấu tạo bên trong của DS1307 bao gồm mạch nguồn, dao động, logic và con trỏ ,thanh ghi thực hiện việc ghi đọc. Do trong các bài toán chúng ta thường sử dụng DS1307 cho đồng hồ thời gian thực nên do đó chúng ta chỉ quan tâm đến việc ghi đọc các thanh ghi cần thiết (sec, min, hour…) thông qua chuẩn truyền thông I2C còn các thanh ghi khác thì chúng ta có thể tìm hiểu kỹ trong datasheet! Vì các thanh ghi đó được coi như là RAM lưu trữ. Nên do đó tôi chỉ giới thiệu các thanh ghi có chức năng thời gian thực phục vụ cho bài toán thời gian.

Trong bộ nhớ của DS1307 có tất cả 64 thanh ghi địa chỉ từ 0 đến 63 và được bắt đầu từ 0x00 đến 0x3F nhưng trong đó chỉ có 8 thanh ghi đầu là thanh ghi thời gian thực nên chúng ta sẽ đi sâu vào 8 thanh ghi (chức năng và địa chỉ thanh ghi thời gian thực này). Nhìn vào bảng thanh ghi trong datasheet ta sẽ thấy như sau :

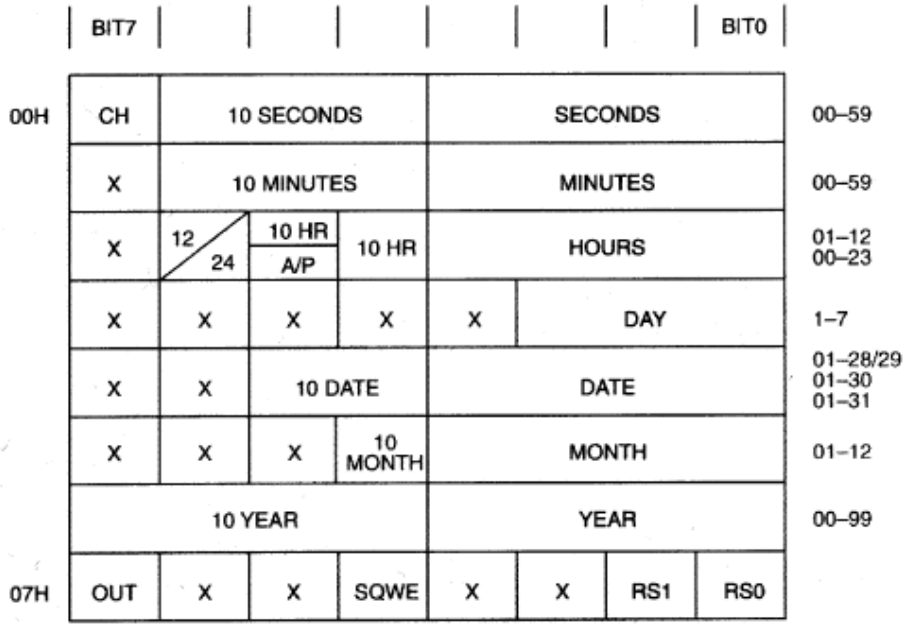


Hình 3.6 Bộ nhớ của DS1307

Nhìn vào bảng trên thấy các thanh ghi thời gian thực nó được sắp sếp theo thứ tự : giây, phút, giờ, thứ, ngày , tháng, năm và bắt đầu từ thanh ghi Giây (0x00) và kết thúc bằng thanh ghi năm (0x06).

Do 7 thanh ghi đầu tiên là khá quan trọng cho thời gian thực và là thanh ghi quan trọng nhất trong con DS1307 nên chúng ta phải hiểu được cách tổ chức thanh ghi này trong DS1307.

Do đó tổ chức thanh ghi như sau.



Hình 3.7 Tổ chức thanh ghi

+ Thanh ghi giây (0x00) : Đây là thanh ghi giây của DS1307. Nhìn trên bảng trên ta thấy được từ bit 0 đến bit 3 là dùng để mã hóa số BCD hàng đơn vị của giây. Tiếp theo từ bit 4 đến bit 6 dùng để mã hóa BCD hàng chục của giây. Tại sao nó chỉ sử dụng có 3 bit này là do giây của chúng ta lớn nhất chỉ đến 59 nên hàng chục lớn nhất là 5 nên chỉ cần 3 thanh ghi này là cũng đủ mã hóa rồi! Còn bit thứ 7 có tên là "CH" theo tôi nó có nghĩa là " Clock Halt – Treo đồng hồ" Do đó nếu mà bit 7 này mà được đưa lên 1 tức là khóa đồng hồ nên do đó nó vô hiệu hóa chip và chip không hoạt động. Nên do vậy lúc nào cũng phải cho bit 7 này luôn xuống 0 từ lúc đầu(cái này sử dụng lệnh end với 0x7F)

+ Thanh ghi phút (0x01) : Đây là thanh ghi phút của DS1307. Cũng nhìn trên bảng thanh ghi này được tổ chức như thanh ghi giây. Cũng là 3 bit thấp dùng để mã hóa BCD chữ số hàng đơn vị và số hàng trục chỉ lớn nhất là 5 nên do đó chỉ cần dùng từ bit 4 đến bit 6 để mã hóa BCD tiếp chữ số hàng chục. Nhưng thanh ghi này có sự khác biệt với thanh ghi giây là bit 7 nó đã mặc định bằng 0 rồi nên do đó chúng ta không phải làm gì với bit 7 mà kệ nó!

+ Thanh ghi giờ (0x02) : Đây là thanh ghi giờ của DS1307 và tôi thấy thanh ghi này được coi là phức tạp nhất vì nó lằng nhà lằng nhằng nhưng mà nhìn bảng thì thấy các tổ chức của nó cũng hợp lý. Trước tiên chúng ta thấy được rằng từ bit 0 đến bit 3 nó dùng để mã hóa BCD của chữ số hàng đơn vị của giờ. Nhưng mà giờ nó còn có chế độ 24h và 12h nên do đó nó phức tạp ở các

bit cao (bit 4 đến bit 7) và sự chọn chế độ 12h và 24h nó lại nằm ở bit 6. Nếu bit 6=0 thì ở chế độ 24h thì do chữ số hàng trục lớn nhất là 2 nên do đó nó chỉ dùng 2 bit (bit 4 và bit 5) để mã hóa BCD chữ số hàng trục của giờ. Nếu bit 6 =1 thì chế độ 12h được chọn nhưng do chữ số của hàng trục của giờ trong chế độ này chỉ lớn nhất là 1 nên do đó bit thứ 4 là đủ để mã hóa BCD chữ số hàng trục của giờ rồi nhưng mà bit thứ 5 nó lại dùng để chỉ buổi sáng hay chiều, nếu mà bit 5 = 0 là AM và bit 5 =1 là PM. Trong cả 2 chế độ 12h và 24h thì bit 7 =0 nên ta ko cần chú ý đến thanh ghi này.

+ **Thanh ghi thứ (0x03):** Đây là thanh ghi thứ trong tuần của DS1307 và thanh ghi này khá là đơn giản trong DS1307. Nó dùng số để chỉ thứ trong tuần nên do đó nó chỉ lấy từ 1 đến 7 tương đương từ thứ hai đến chủ nhật. Nên do đó nó dùng 3 bit thấp (bit 0 đến bit 2) để mã hóa BCD ra thứ trong ngày. Còn các bit từ 3 đến 7 thì nó mặc định bằng 0 và ta không làm gì với các bit này!

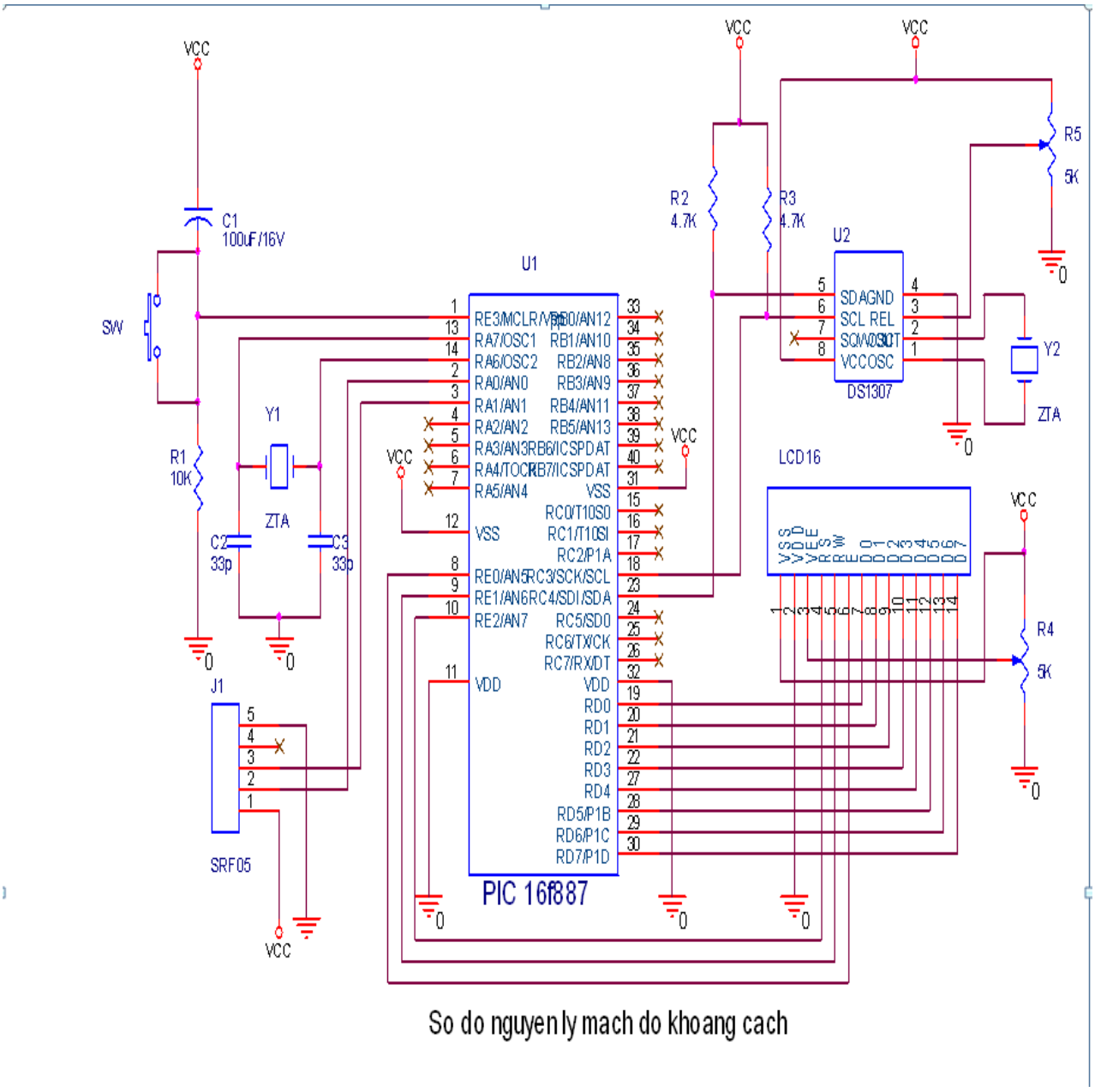
+ **Thanh ghi ngày (0x04)** : Đây là thanh ghi ngày trong tháng của DS1307. Do trong các tháng có số ngày khác nhau nhưng mà nằm trong khoảng từ 1đến 31 ngày. Do đó thanh ghi này các bit được tổ chức khá là đơn giản. Nó dùng 4 bit thấp (bit0 đến bit 3) dùng để mã hóa BCD ra chữ số hàng đơn vị của ngày trong tháng. Nhưng do chữ số hàng trục của ngày trong tháng chỉ lớn nhất là 3 nên chỉ dùng bit 4 và bit 5 là đủ mã hóa BCD rồi. Còn bit 6 và bit 7 chúng ta không làm gì và nó mặc định bằng 0.

+ **Thanh ghi tháng (0x05)** : Đây là thanh ghi tháng trong năm của DS1307. Tháng trong năm chỉ có từ 1 đến 12 tháng nên việc tổ chức trong bit cũng tương tự như ngày trong tháng nên do cũng 4 bit thấp (từ bit 0 đến bit 3) mã hóa BCD hàng đơn vị của tháng. Nhưng do hàng chục chỉ lớn nhất là 1 nên chỉ dùng 1 bit thứ 4 để mã hóa BCD ra chữ số hàng trục và các bit còn lại từ bit 5 đến bit 7 thì bỏ trống và nó mặc định cho xuống mức 0.

+ **Thanh ghi năm (0x06):** Đây là thanh ghi năm trong DS1307. DS1307 chỉ có 100 năm thôi tương đương với 00 đến 99 nên nó dùng tất cả các bit thấp và bit cao để mã hóa BCD ra năm!

+ **Thanh ghi điều khiển (0x07):** Đây là thanh ghi điều khiển quá trình ghi của DS1307 và Quá trình ghi phải được kết thúc bằng địa chỉ 0x93.

II.Sơ đồ nguyên lý của mạch.



Hình 3.8 Sơ đồ mạch nguyên lý

CHƯƠNG IV THIẾT KẾ PHẦN MỀM

Phần mền viết để điều khiển là phần mềm C.

```
#include "E:\Ha Document\Vi xu ly\ngocha\Main16F887\main.h"
/**
*****************************************************************
*****************
  * Ten Tep      :       main.h
  * Tac Gia      :       Ngo Ngoc Ha
  * Khoa         :       Cong Nghe Tu Dong
  * Lop          :       D6LT-DCN
  * Ngay         :       10-03-2013
```

```
 * Tom Tat     :       Khai bao cac thu vien.
 *                    Cau hinh mot so chuc nang cho trinh bien dich.
 *                    Cau hinh mot so chuc nang cho CHIP.
 *                    Dinh nghia I/O.
 *
 *

*****************************************************************
*****************
 * Chu Y      :
 *
 *

*****************************************************************
*****************
 */
#ifndef _MAIN_
#define _MAIN_
/***************KHAI BAO CAC THU VIEN*****************/
#include <16F877.h>
//#include "string.h"
/**************KHAI BAO CAU HINH FUSE BIT************/
////////////////////////////////////////////////////
//                                                //
// Voi moi muc phai chon mot trong danh sach dua ra  //
//                                                //
////////////////////////////////////////////////////

//_____________________HIGH SPEED OSC_____________________

//#FUSES LP                    // Su dung nguon dao dong tan so thap < 200 khz
//#FUSES XT                     // Dao dong thach anh <= 4mhz voi PCM/PCH ,
3mhz to 10 mhz voi PCD
//#FUSES RC                    // Dao dong RC voi CLKOUT
#FUSES HS                     //Dao dong tan so cao (> 4mhz voi PCM/PCH)
(>10mhz voi PCD)

//_____________________POWER UP TIMER_____________________

#FUSES NOPUT                  //Khong su dung Power Up Timer
//#FUSES PUT                   //Su dung Power Up Timer
//_____________________BROWN OUT_____________________
```

```
#FUSES NOBROWNOUT            //Khong reset chip khi BrownOut
//#FUSES BROWNOUT             //Reset chi khi co BrownOut
//_____________LOW VOLTAGE PROGRAM_____________________

#FUSES NOLVP                 //No low voltage programing, B3(PIC16) or
B5(PIC18) used for I/O
//#FUSES LVP                 //Low Voltage Programming on B3(PIC16) or
B5(PIC18)
//_____________CODE PROTECED EEPROM_____________________

#FUSES NOCPD                 //Khong bao ve du lieu EEPROM
//#FUSES CPD                 //Dung che do bao ve du lieu EEPROM

//_____________PROGRAM WRITE PROTECED_____________________

//#FUSES WRT                 //Program Memory Write Protected
//#FUSES WRT_50%             //Lower half of Program Memory is Write
Protected
//#FUSES WRT_5%              //Lower 255 bytes of Program Memory is
Write Protected
#FUSES NOWRT                 //Program memory not write protected

//_____________ENABLE DEBUG MODE FOR ICD_____________

//#device ICD=TRUE           // Kich hoat chuc nang DEBUG ICD Integrated
Chip Debugging
//_____________DEBUG FOR ICD_____________________

#FUSES NODEBUG               //Khong su dung che do Debug voi ICD
//#FUSES DEBUG               // Su dung che do Debug voi ICD

//_____________CLOCK_____________________

#use delay(clock=24000000)           // Su dung tan so 20Mhz, khong reset
watch dog khi goi den ham delay.
//#use delay(clock=20000000,RESTART_WDT)   // Su dung tan so 20Mhz,
reset watch dog khi goi den ham delay.

//_____________FAST_STANDAR I/O PORTA_____________________

//#use FAST_IO(A)            // Thiet lap che do fast I/O cho PORTA, yeu cau
phai chi ro huong Vao/Ra
```

```
                // cho cac chan I/O. cac ham Input,Output se chi su dung 1 chu ky lenh
//#use STANDARD_IO(A)      // Thiet lap che do chuan I/O cho PORTA, khong yeu cau phai chi ro huong Vao/Ra
                // cho cac chan I/O. cac ham Input,Output se su dung 3-4 chu ky lenh
//#use FIXED_IO(A)      //

//_____________________FAST_STANDAR I/O PORTC_____________________

//#use FAST_IO(C)          // Thiet lap che do fast I/O cho PORTC, yeu cau phai chi ro huong Vao/Ra
                // cho cac chan I/O. cac ham Input,Output se chi su dung 1 chu ky lenh
#use STANDARD_IO(C)     // Thiet lap che do chuan I/O cho PORTC, khong yeu cau phai chi ro huong Vao/Ra
                // cho cac chan I/O. cac ham Input,Output se su dung 3-4 chu ky lenh
//#use FIXED_IO(C)      //

//_____________________FAST_STANDAR                I/O PORTD_____________________

#use FAST_IO(D)         // Thiet lap che do fast I/O cho PORTD, yeu cau phai chi ro huong Vao/Ra
                // cho cac chan I/O. cac ham Input,Output se chi su dung 1 chu ky lenh
//#use STANDARD_IO(D)      // Thiet lap che do chuan I/O cho PORTD, khong yeu cau phai chi ro huong Vao/Ra
                // cho cac chan I/O. cac ham Input,Output se su dung 3-4 chu ky lenh
//#use FIXED_IO(D)      //

//_____________________FAST_STANDAR I/O PORTE_____________________

#use FAST_IO(E)         // Thiet lap che do fast I/O cho PORTE, yeu cau phai chi ro huong Vao/Ra
                // cho cac chan I/O. cac ham Input,Output se chi su dung 1 chu ky lenh
//#use STANDARD_IO(E)      // Thiet lap che do chuan I/O cho PORTE, khong yeu cau phai chi ro huong Vao/Ra
                // cho cac chan I/O. cac ham Input,Output se su dung 3-4 chu ky lenh
```

```
//#use FIXED_IO(E)       //

//_____________________________CASE_____________________________

#CASE               // Phan biet chu hoa va chu thuong trong khi lap trinh code

/************** DINH NGHIA CAC CHAN I/O ************/

////////////////////////////////////////////////////
//   Xoa chu thich cho cac chuc nang I/O su dung       //
//  Dinh nghia lai cac chan cho phu hop voi phan cung //
////////////////////////////////////////////////////

//_____________________________I/O SRF_____________________________
#define SRF05_ECHO PIN_A0
#define SRF05_TRIGGER PIN_A1
//_____________________________I/O DS_____________________________
#define DS1307_SCL PIN_C3
#define DS1307_SDA PIN_C4
//_____________________________I/O LCD_____________________________
#define LCD_RS PIN_E0
#define LCD_RW PIN_E1
#define LCD_EN PIN_E2
#define LCD_D4 PIN_D4
#define LCD_D5 PIN_D5
#define LCD_D6 PIN_D6
#define LCD_D7 PIN_D7
#endif
**

*****************************************************************
*****************
  * Ten Tep      :       lcd_16.h
  * Tac Gia      :       Ngo Ngoc Ha
  * Khoa         :       Cong Nghe Tu Dong
  * Lop          :       D6LT-DCN
  * Ngay         :       10-03-2013
  * Tom Tat   :     Dinh nghia cac ham dieu khien LCD 16x2.
  *
  *
```

```
**********************************************************************
*****************
 * Chu Y      :
 *

**********************************************************************
*****************
 */
//Tao Xung
 void LCD_Enable(void)
{
 output_high(LCD_EN);
 delay_us(3);
 output_low(LCD_EN);
 delay_us(50);
}
//Ham Gui 8 Bit Du Lieu Ra LCD
 void LCD_Send4Bit( unsigned char Data )
{
output_bit(LCD_D0,Data&0x01);
 output_bit(LCD_D1,(Data>>1)&1);
 output_bit(LCD_D2,(Data>>2)&1);
 output_bit(LCD_D3,(Data>>3)&1);
 output_bit(LCD_D4,(Data>>4)&1);
 output_bit(LCD_D5,(Data>>5)&1);
 output_bit(LCD_D6,(Data>>6)&1);
 output_bit(LCD_D7,(Data>>7)&1);
}
// Ham Gui 1 Lenh Cho LCD
 void LCD_SendCommand (unsigned char command )
{
LCD_Send4Bit  ( command >>4 );/* Gui 4 bit cao */
LCD_Enable () ;
LCD_Send4Bit  ( command  );      /* Gui 4 bit thap*/
LCD_Enable () ;
}
// Ham Khoi Tao LCD
 void LCD_Init ( void )
{
output_drive(LCD_D0);
output_drive(LCD_D1);
output_drive(LCD_D2);
```

```
output_drive(LCD_D3);
output_drive(LCD_D4);
output_drive(LCD_D5);
output_drive(LCD_D6);
output_drive(LCD_D7);
output_drive(LCD_EN);
output_drive(LCD_RS);
output_drive(LCD_RW);
LCD_Send4Bit(0x00);
delay_ms(20);
output_low(LCD_RS);
output_low(LCD_RW);
LCD_Send4Bit(0x03);
LCD_Enable();
delay_ms(5);
LCD_Enable();
delay_us(100);
LCD_Enable();
LCD_Send4Bit(0x02);
LCD_Enable();
LCD_SendCommand( 0x28 );      // giao thuc 8 bit, hien thi 4 hang
LCD_SendCommand( 0x0c);    // cho phep hien thi man hinh
LCD_SendCommand( 0x06 );      // tang ID, khong dich khung hinh
LCD_SendCommand( 0x01 ); // xoa toan bo khung hinh
}
void LCD_Gotoxy(unsigned char x, unsigned char y)
{
  unsigned char address;
  if(!y)
  address = (0x80+x);
  else
  address = (0xC0+x);
  delay_us(1000);
  LCD_SendCommand(address);
  delay_ms(10);
}
// Ham Xoa Man Hinh LCD
void LCD_Clear()
{
  LCD_SendCommand(0x01);
  delay_ms(10);
}
// Ham Gui 1 Ki Tu Len LCD
```

```
 void LCD_PutChar ( unsigned char Data )
{
 output_high(LCD_RS);
 LCD_SendCommand( Data );
 output_low(LCD_RS);
}
void LCD_Puts (char *s)
{
   while (*s)
   {
      LCD_PutChar(*s);
      s++;
   }
}
void LCD_PutsDelay (char *s,unsigned int time)
{
   while (*s)
   {
      LCD_PutChar(*s);
      s++;
      delay_ms(time);
   }
}

**

************************************************************************
*****************
  * Ten Tep      :       srf05.h
  * Tac Gia      :       Ngo Ngoc Ha
  * Khoa         :       Cong Nghe Tu Dong
  * Lop          :       D6LT-DCN
  * Ngay         :       10-03-2013
  * Tom Tat   :     Dinh nghia cac ham dieu khien SRF05.
  *
  *

************************************************************************
*****************
  * Chu Y      :
  *
```

```
**********************************************************************
*****************
 */
 #ifndef _SRF05_
#define _SRF05_
unsigned int32 num_pulse=0;
int1 range_ok=0;
#endif
**

**********************************************************************
*****************
 * Ten Tep      :       ds1307.h
 * Tac Gia      :       Ngo Ngoc Ha
 * Khoa         :       Cong Nghe Tu Dong
  * Lop         :       D6LT-DCN
 * Ngay         :       10-03-2013
 * Tom Tat   :     Dinh nghia cac ham dieu khien DS1307.
 *
 *

**********************************************************************
*****************
 * Chu Y      :
 *

**********************************************************************
*****************
 */
#define DS1307_SCL PIN_C3
#define DS1307_SDA PIN_C4
#use i2c(master, sda=DS1307_SDA, scl=DS1307_SCL)
void set_time() ;
void update_time();
int8 giay,phut,gio,thu,ngay,thang,nam;
int l;
int x,y,z;

////////////////////////////////////////////////////////////////////////////////
void int_DS1307()
{
output_float(DS1307_SCL);
```

```
output_float(DS1307_SDA);
}

// ------- Chuyen du lieu m? BINARY cua MASTER -> du lieu m? BCD cho DS1307--------
int DECIMALtoBCD(int data) // MASTER -> DS1307
{
// x -> y trong do x la du lieu cua MASTER, y la du lieu sau khi chuyen doi
// Thuat toan thuc hien bien doi tu ma DECIMAL sang ma BCD he 10

x = data;
if(x<10) // vd: (x=9) -> (y=9) = 0x09
{
y = x;
}
else if(x>=10) // vd: (x=29) -> (y=41) = 0x29
{
y = (x/10 * 6) + x;
}
return y;
}

// ------- Chuyen du lieu m? BCD cua DS1307 -> du lieu m? BINARY cho MASTER-------
int BCDtoDECIMAL(int data) // MASTER <- DS1307
{
// x <- y trong do y la du lieu cua SLAVE, x la du lieu sau khi chuyen doi
// Thuat toan thuc hien bien doi tu ma BCD he 10 sang ma DECIMAL

y = data;
l=0;
if(y<10) // vd: (x=9) <- (y=9) = 0x09
{
x = y;
}
else if(y>=10) // vd: (x=10) <- (y=16) = 0x10
{
do
{
x = y - (6 * l);
z = (x/10 * 6) + x;
l++;
}
```

```
while(z!=y);
}
return x;
}

#int_EXT


void set_time()
{
// --------- Khoi tao hien thi ban dau: SUN, 10-03-2013, 0:00:00 --------
giay = 00; //Giay: 00
phut = 00; //Phut: 00
gio = 00; //Gio: 00 (che do 24h)
thu = CN; //Thu 7: SUN (SUNDAY)
ngay = 10; //Ngay: 10
thang = 03; //Thang: 03
nam = 13; //Nam: 13
#include <main.h>
#include <SRF.h>
#include "lcd.h"
#include "ds1307.h"
#define SRF05_TRIGGER   PIN_A1
#define SRF05_ECHO      PIN_A0
#define NO_OBJECT       0
#INT_EXT
void Ngat_Ngoai(void)
{
  disable_interrupts(GLOBAL);
  num_pulse+=get_timer1();
  range_ok=1;
  enable_interrupts(GLOBAL);
}
#INT_TIMER1
void  Ngat_Timer1(void)
{
  disable_interrupts(GLOBAL);
  num_pulse+=0xffff;
  enable_interrupts(GLOBAL);
}
void  SRF05_StartRange()
{
  while(!range_ok)
```

```
  {
      output_high(SRF05_TRIGGER);
      delay_ms(15);                    // Phai tao 1 xung len co do lon it nhat
10ms
      output_low(SRF05_TRIGGER);       // Bat dau phep do.
      while(!(input(SRF05_ECHO)));     // Doi cho den khi chan ECHO
duoc keo len cao
      set_timer1(0);
      enable_interrupts(GLOBAL);
      delay_ms(50);
  }
}
float32 SRF05_GetDistance()
{
  float32 time_us=0,distance=0;
  SRF05_StartRange();
  disable_interrupts(GLOBAL);
  if(num_pulse>180000)
  {
    num_pulse=0;
    range_ok=0;
  return NO_OBJECT;
  }
  else
  {
    time_us=num_pulse/6;
    distance=time_us/58;
    num_pulse=0;
    range_ok=0;
  return distance;
  }
}
void main()
{
  unsigned char str[20],i=0;
  float32 range;
  output_float(SRF05_ECHO);
  output_drive(SRF05_TRIGGER);
  LCD_Init();
  sprintf(str,"DEMO SRF05");
  delay_ms(10);
  LCD_Puts(str);
  delay_ms(1000);
```

```
  LCD_Clear();
  sprintf(str,"Ngo Ngoc Ha");
  LCD_Gotoxy(1,0);
  LCD_Puts(str);
  port_b_pullups (TRUE);
  ext_int_edge(H_TO_L);                // ngat canh xuong
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);//F_TIMER1=F_OSC/4
  enable_interrupts(INT_TIMER1);
  enable_interrupts(INT_EXT);          // kich hoat ngat ngoai
  disable_interrupts(GLOBAL);
  while(TRUE)
  {
        range=SRF05_GetDistance();
        if(range==NO_OBJECT)
        {
          LCD_Gotoxy(0,1);
          sprintf(str,"Khong Co Vat Can");
          LCD_Puts(str);
          delay_ms(500);
        }
        else
        {
          LCD_Gotoxy(0,1);
          sprintf(str,"Dis : %3.2f Cm  ",range);
          LCD_Puts(str);
        }
    }

}
```

CHƯƠNG IV
KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN

I.KẾT LUẬN

Trong đề tài này chúng em tìm hiểu và nghiên cứu thiết kế mạch đo khoảng cách dùng vi điều khiển PIC16F887.Đây chỉ là ứng dụng nhỏ của PIC trong điều khiển để đo khoảng cách.

+ Hệ thống điều khiển tương đối ổn định, đáp ứng được điều kiện đề tài.

+ Cảm biến đọc thông số khoảng cách tương đối chính xác và hiện thị qua LCD.
+ Phần mềm viết code cũng đơn giản.
Qua đề tài này giúp chúng em thêm kiến thức bổ ích vào trong chuyên ngành học tập của mình, hiểu biết nhiều hơn về môn vi xử lý, kết hợp lý thuyết với thực hành hoàn thành tốt đề tài.

II.HƯỚNG PHÁT TRIỂN

Hướng phát triển của đồ án là: Dùng PIC 16F887 để diều khiển để đo khoảng cách ngoài ra còn SRF05 có khả năng kết nối các vi xử lý khác như họ MC51 và họ AVR…Ngày nay, người ta vẫn nghiên cứu phất triển chế tạo dể gắn các cảm biến khác như laser, camera…và các ứng dụng này để phục vụ con người trong cuộc sống hàng ngày

TÀI LIỆU THAM KHẢO

Datasheet PIC16F887của hãng Microchip,bộ thời gian DS1307.
Ngôn ngữ lập trình C của tác giả Quách Tuấn Ngọc
Giáo trình vi xử lý của DHBKTPHCM