Versione N° 1.0c

LaurTec

Metro ad Ultrasuoni

Autore : *Mauro Laurenti*

email: info.laurtec@gmail.com

ID: PJ6001-IT

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.

AVVERTENZE

I progetti presentati non hanno la certificazione CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.

Si fa inoltre presente che quanto riportato viene fornito cosi com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Introduzione

Si è soliti dire che i pipistrelli siano ciechi...ma i pipistrelli vedono anche di notte! In questo progetto si sfruttano gli ultrasuoni per poter ricavare la distanza tra il sensore e un ostacolo. Questo permette l'utilizzo del progetto in ambito dei sistemi automatici quali la robotica o domotica. L'applicazione mostra inoltre la versatilità del sistema Freedom al quale non bisogna far altro che connettere un sensore ad ultrasuoni e un LCD, per i quali è già predisposto.

Il sensore ad ultrasuoni

Il sensore ad ultrasuoni di cui si è fatto uso è SRF05 del tipo Trig/Echo. In Figura 1 è riportata la foto del sensore. I pin sulla destra sono utilizzati solo in fase di programmazione del dispositivo stesso ovvero per l'installazione del firmware all'interno del PIC che controlla il sensore stesso. Questo significa che non sono utilizzati per nessuna applicazione da parte del progettista.

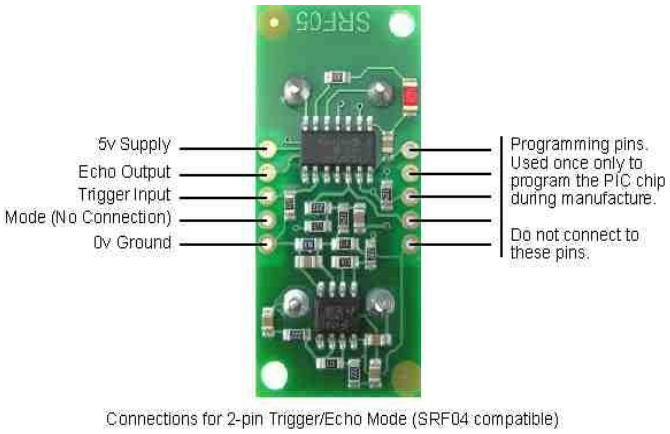

Figura 1: *Sensore ad Ultrasuoni SRF05 (modalità 2-pinTrig/Echo)*

Le caratteristiche tecniche di questo sensore sono riportate in Tabella 1.

Caratteristiche Tecniche	
Tensione Operativa	5V
Corrente Operativa Tipica	4mA
Frequenza	40 Khz
Portata	1cm - 4m
Impulso di ritorno	Segnale TTL positivo, di durata proporzionale alla distanza rilevata.
Trigger di Input	Impulso TTL di durata minima di 10 uS.
Modalita' di funzionamento	Pin singolo per trig/echo o 2 Pin SRF04 compatibile.
Dimensioni	43 x 20 x H 17 mm

Tabella 1: *Caratteristiche tecniche del sensore SRF05*

Il principio di funzionamento del sensore ad ultrasuoni si potrebbe dire che è molto semplice, ma se madre natura ha impiegato milioni di anni per sfruttarlo e l'uomo non ha potuto che copiare...forse non

è poi tanto semplice. Certo è che comprenderne il principio al livello d'astrazione che ci interessa non sara' complicato.

Gli ultrasuoni sono frequenze che il nostro orecchio non è in grado di percepire. Il sensore in questione lavora alla frequenza di 40KHz mentre un buon orecchio umano percepisce suoni fino a circa 22KHz. Questo significa che il nostro orecchio non è in grado di percepire il "suono" emesso da questo sensore. Il segnale emesso dal sensore è paragonabile ad una pallina da biliardo che viene lanciata contro una sponda del tavolo da gioco. Quando il segnale, ovvero la pallina, raggiunge un ostacolo, ovvero la sponda del tavolo, viene riflesso. Dal momento che la velocità con cui il suono viaggia nello spazio libero, è nota[1], misurando il tempo che percorre tra l'emissione del segnale e il suo ritorno è possibile risalire alla distanza dell'oggetto che ha causato la riflessione. Da questo si capisce che se non c'è nessuno ostacolo non verrà rilevato nessun eco.

Il sensore SRF05 possiede due sensori ad ultrasuoni uno utilizzato come sorgente per generare la nota a 40KHz e uno utilizzato come orecchio per rilevare l'eventuale segnale di eco derivante dalla presenza di un ostacolo. La presenza dei due sensori non è in generale obbligatoria, sono infatti presenti sistemi con un solo sensore che funziona prima come sorgente e poi come orecchio per captare un eventuale eco.

La velocità con cui il suono viaggia nello spazio libero viene a dipendere dall'umidità dell'aria dalla temperatura e anche dalla pressione atmosferica. Da questo si capisce che se lo strumento non è opportunamente calibrato ad ogni utilizzo le misure saranno soggette ad un certo errore. In questo progetto non è prevista nessuna autocalibrazione ma è possibile comunque ottenere una precisione intorno al cm. Un altro fattore che può causare un deterioramento della precisione del sensore è legata alla forma dell'oggetto che causa la riflessione stessa. Forme complesse o troppo grandi causano riflessioni multiple che degradano la precisione del sensore stesso[2]. Anche in questi casi la precisione è comunque più che sufficiente in molte applicazioni. Il sensore è utilizzabile secondo le specifiche riportate in Tabella 1 in un range compreso tra 1cm e 4m anche se l'accuratezza in questi due estremi non è elevata.

Vediamo ora in maggior dettaglio come comandare il sensore SRF05. Le linee di controllo sono quelle sulla sinistra di Figura 1. Queste possono essere impostate per due modalità di controllo differenti nominate 2-pin Trig/Echo e 1-pin Trig/Echo. La prima modalità è riportata in Figura 1; è possibile osservare che partendo dall'alto bisogna collegare il pin 1 a Vcc, il pin 2 è per il segnale di output per l'echo, il pin 3 è il segnale di input per il Trig, il pin 4 è non connesso[3] mentre il pin 5 è collegato a massa. Questa piedinatura è compatibile con il connettore per i sensori ad ultrasuoni presente sul sistema Freedom. In particolare il pin 1 del connettore per ultrasuoni sulla scheda Freedom corrisponde a Vcc (+5V) della scheda del sensore. La modalità ora descritta è detta 2-pin Trig/Echo poiché come descritto è presente una linea per il segnale di Trig e una linea per il segnale di Echo. Per comandare il sensore in questa modalità bisogna inviare un impulso sulla linea di Trig di almeno 10uS. Quando il sensore riceve questo impulso trasmette il segnale alla frequenza di 40KHz in 8 piccoli "colpetti", e si mette mette poi in attesa del segnale di eco. Il tempo che impiega il segnale d'eco a tornare al sensore è misurabile dalla durata dell'impulso che è presente in uscita alla linea Echo. Se la durata è più' di 30ms vuol dire che l'oggetto è oltre i 4 metri della portata del sensore. Quanto appena descritto è riassunto in Figura 2.

La seconda modalità permette di fare misure utilizzando un solo pin. Questo viene ottenuto mettendo a massa il pin 4 e non utilizzando il pin 2. Il pin 3 viene dunque utilizzato in maniera alternativa come linea di Trig e come linea di Echo. Questa modalità può risultare un po' più complicata per quanto riguarda il software di gestione del sensore ma permette di risparmiare un pin. Il grafico riassuntivo è lo stesso di Figura 2 ma con la linea 1 e 3 sovrapposte.

---

1 La velocità del suono nello spazio libero e alla temperatura di 25°C è circa 340m/s, quindi se gridiamo la nostra voce giungerà a 340m di distanza dopo un secondo.

2 Se si mettono troppo vicini un oggetto piccolo e uno grande il sensore tenderà a rilevare solo l'oggetto grande.

3 Non va collegato nè a massa nè a Vcc poiché sulla scheda del sensore è gia' presente un resistore di pull-up.

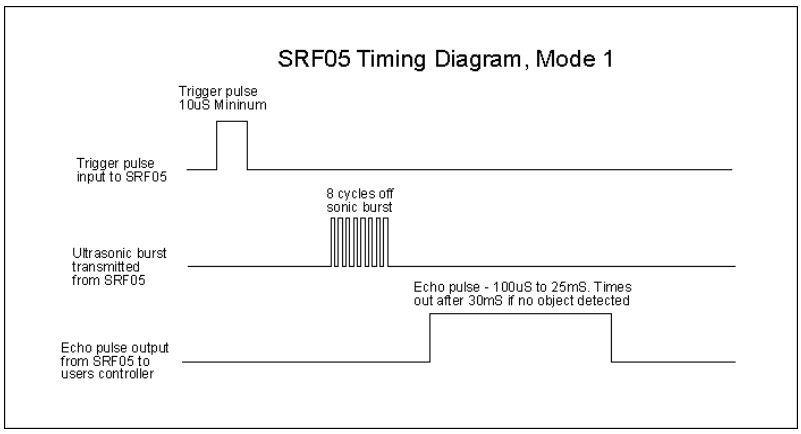

Figura 2: *Modalità 2-pin Trig/Echo*

Analisi del progetto

Il programma di gestione è stato scritto in C18[4] per il PIC18F4580 montato sulla scheda Freedom. Questa scheda è utilizzata come supporto principale ma nulla vieta di utilizzare una scheda personale, montata anche su mille fori, in cui montare solo l'hardware strettamente necessario per l'applicazione; in Figura 3 è riportato il montaggio del sistema ottenuto collegando direttamente un display LCD 16x2 e il sensore SRF05 sul sistema Freedom.

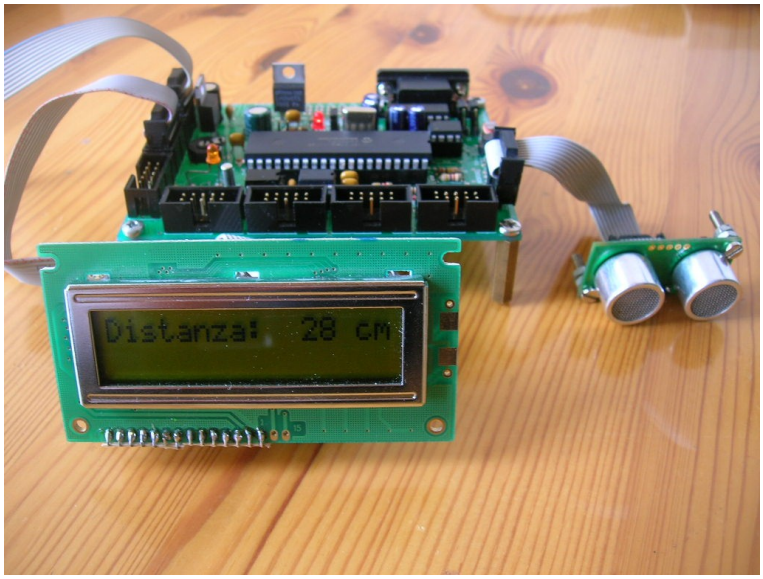

Figura 3: *Sistema Metro ad Ultrasuoni montato*

Le prime 21 righe di programma rappresentano l'inizializzazione classica per il PIC18F4580 che può essere utilizzata in molte applicazioni. In particolare si sono incluse le librerie per il controllo dell'LCD e alcune librerie standard del C18. Nelle righe 23 e 24 si sono definite delle etichette per

[4] Per ulteriori informazioni su come programmare in C18 si rimanda al Tutorial "C18 step by step".

individuare i pin del PIC che sono rispettivamente utilizzati per la linea Echo e Trig. In questo modo risulta più facile l'utilizzo di questi pin. Tra le righe 28 e 30 sono dichiarate le variabili globali utilizzate all'interno del programma principale per la misura dello spazio. In particolare la variabile `total_us` contiene il numero di us che intercorrono tra l'invio dell'impulso e il suo eco. In realtà il valore è approssimato visto che la funzione utilizzata per il conteggio non è gestita in assembly e per la calibrazione si è fatto uso di un centimetro.

La variabile `distance` contiene invece il valore dello spazio tra il sensore e l'ostacolo, mentre l'array di caratteri `distanceLCD [3]` contiene lo stesso valore della variabile `distance` ma in formato stringa[5], pronta per la visualizzazione sull'LCD. In particolare `distanceLCD [2]` contiene la cifra meno significativa della distanza espressa in cm, mentre `distanceLCD [0]` contiene la cifra più significativa.

Il programma è stato concepito per poter essere integrato in un sistema più complesso senza dover modificare nulla. Per questa ragione la struttura del programma potrà sembrare più complessa del necessario.

Per permettere l'integrazione con altre parti di programma che possono essere aggiunte a quanto segue si è fatto in modo che l'impulso sia inviato ad intervalli regolari usufruendo l'interruzione generata dal TMR0. Gli intervalli in cui bisogna inviare l'impulso sono decisi dalla variabile num all'interno della funzione per la gestione dell'interupt. In particolare la variabile num viene incrementata ad ogni interruzione del TMR0 e per mezzo dell'operatore if si controlla se la variabile num è uguale a un multiplo di 20. Se questa condizione viene verificata viene inviato un impulso di Trig al sensore ad ultrasuoni in modo da inviare il segnale. Quanto spiegato viene gestito indipendentemente dal programma principale che può quindi svolgere qualunque altra cosa.

Il programma principale consiste semplicemente nel controllare la linea Echo per eventuali impulsi, i quali sono indice della presenza di ostacoli; questo controllo è effettuato all'interno del ciclo infinito. In particolare quando la linea Echo viene trovata alta viene eseguito il seguente loop

```
while (Echo == 1)
{
  total_us++;

  _asm
  nop
  _endasm
}
```

questo termina al termine dell'impulso di Echo. Durante questo ciclo viene incrementata la variabile `total_us` e viene anche eseguita un'istruzione assembly nop (nop, ovvero nessuna operazione). Il nop è stato inserito semplicemente per facilitare la conversione del conteggio in centimetri. Dal momento che la variabile `total_us` non viene incrementata per mezzo di un controllo in assembly si perde il controllo del tempo[6].

Quando la linea Echo è nuovamente a livello logico basso il ciclo termina e inizia la parte del programma per la conversione tempo-centimetri e relativa visualizzazione su LCD. Per la conversione tempo-centimetri è necessario dividere il tempo misurato in microsecondi per 58. Questo è il valore riportato nel datasheet del sensore. Dal momento che il tempo da noi preso è in realtà stato preso più lentamente è sufficiente dividere per 32. In particolare il nop nel ciclo while precedentemente descritto è stato inserito proprio per fare in modo che la conversione tempo-centimetri potesse essere effettuata dividendo per 32. Infatti la divisione è un'operazione che richiede molto tempo di calcolo e molta memoria di programma. Se però la divisione è una potenza di due, è possibile ottenerla semplicemente

[5] Per stringa si intende semplicemente un Array di caratteri.

[6] Sono presenti anche Tool per mezzo dei quali è possibile calcolare il tempo di esecuzione del ciclo while in questione ma in ogni modo il controllo totale del microcontrollore lo si ottiene in assembly.

con uno shift verso destra per un numero di volte pari all'esponente. Ragionamento analogo, ma con shift a sinistra, è valido per moltiplicazioni per potenze di due. Nel nostro caso dovendo dividere per 32 ovvero per 2^5 bisogna fare 5 shift a destra. Il risultato della divisione è posto all'interno della variabile globale `distance` che può essere utilizzata da altre parti del pogramma per il controllo della distanza del sensore ed eventuali oggetti.

Come detto l'Array `distanceLCD` contiene sempre lo stesso valore di `distance` ma convertito in stringa, in modo da facilitarne la visualizzazione sull'LCD e la sua manipolazione. Se la distanza risulta maggiore di 400cm ovvero di 4m la variabile `distanceLCD` viene caricata con i caratteri <-> come riportato dal seguente segmento di codice:

```
if (distance>400)
{
  distanceLCD[2]='>';
  distanceLCD[1]='-';
  distanceLCD[0]='<';
}
```

se la distanza è invece inferiore o uguale a 400cm viene effettuata la conversione da intero a stringa per mezzo della funzione `itoa (distance,distanceLCD)`.

Dopo questa conversione la stringa sarebbe formalmente pronta per essere scritta per mezzo della funzione void WriteVarLCD(char *buffer). In realtà la funzione itoa() non mantiene il valore posizionale all'interno della stringa `distanceLCD` dunque è necessario manipolare la stringa prima della visualizzazione in modo da spostare le cifre all'interno della stringa stessa e mantenere sempre lo stesso valore posizionale all'interno della stessa.

Dopo la manipolazione della stringa è eseguita per tre volte la funzione `ShiftCursorLCD (LEFT);` che sposta il cursore del display di tre posizioni verso sinistra. In questo modo ad ogni nuova lettura della misura della distanza sensore ostacolo non è necessario riscrivere tutto il testo ma solo le tre cifre della distanza.

Da quanto spiegato si capisce che se si volesse utilizzare questo software per altre applicazioni è possibile far uso delle variabile globale `distance` e scrivere il proprio codice a partire dal testo

```
// ulteriori controlli nel programma vanno inseriti dopo questo testo
```

Programma Sorgente

```
1 /*
2 Autore : Mauro Laurenti
3 Versione : 1.0
4 Data : 25/5/2006
5 CopyRight 2006
6
7 visita il sito www.LaurTec.com per ulteriore materiale
8
9 */
10
11 #include <p18f4580.h>
12 #include <timers.h>
13 #include <stdlib.h>
14 #include <ctype.h>
15 #include "\Library\LCD_44780_Freedom.h"
16 #include "\Library\Sponsor.h"
17
```

```
18 #pragma config OSC = HS // 20Mhz
19 #pragma config WDT = OFF // disattivo il watchdog timer
20 #pragma config LVP = OFF // disattivo la programmazione LVP
21 #pragma config PBADEN = OFF // disabilito gli ingressi analogici sulla PORTB
22
23 #define Trig PORTCbits.RC0
24 #define Echo PORTCbits.RC5
25
26 void Int_Event (void); // prototipo di funzione
27
28 int total_us ; // variabili globali per la misura dello spazio
29 int distance; // variabile che contiene la distanza come valore intero in cm
30 unsigned char distanceLCD [3]; //come sopra ma il valore è una stringa
visualizzabile su LCD
31
32
33
34 #pragma code low_vector=0x18
35
36 void low_interrupt (void)
37 {
38 _asm GOTO Int_Event _endasm //imposta il salto per la gestione dell'interrupt
39 }
40
41 #pragma code
42
43
44
45 #pragma interruptlow Int_Event
46
47
48 void Int_Event (void)
49 { static unsigned char num =0; // variabile per il conteggio degli interrupt
50
51
52
53 if (INTCONbits.TMR0IF == 1 ) // Controllo che l'interrupt sia stato generato da
TMR0
54    {
55          INTCONbits.TMR0IF = 0;
56          num++;
57
58    if (num == 20 || num == 40 || num == 60 || num == 80 || num
== 100 || num == 120 || num == 140 || num == 160 || num == 180 || num
== 200|| num == 220 || num == 240 || num == 255)
59          {     Trig = 1; //invio dell'impulso di Trig
60                delay (50);
61                Trig = 0;
62          }
63    }
67
68 }
69
70
71
72 void main (void)
73 {
74    TRISA = 0xFF;
75    PORTA = 0x00;
76
77    TRISB = 0x00 ;
78    PORTB = 0x00 ;
```

```
79
80    TRISC = 0xFE; // inizializzo la PORTC per il sensore ad ultrasuoni
81    PORTC = 0x00;
82
83    TRISD = 0x00; // la PORTD è impostata per funzionare  conl'LCD
84    PORTD = 0x00;
85
86    TRISE = 0x00;
87    PORTE = 0x00;
88
89    OpenLCD ();
90
91    WriteSponsor ();
92    WriteStringLCD ("Distanza: --- cm");
93
94
95 //sposto il cursore all'inizio della cifra piu' significativa della misura
96    ShiftCursorLCD (LEFT);
97    ShiftCursorLCD (LEFT);
98    ShiftCursorLCD (LEFT);
99    ShiftCursorLCD (LEFT);
100   ShiftCursorLCD (LEFT);
101   ShiftCursorLCD (LEFT);
102
103
104   OpenTimer0 (TIMER_INT_ON & T0_SOURCE_INT & T0_16BIT );
105
106   INTCONbits.GIE = 1; // Enable global interrupts.
107
108
109   while (1)
110         {
111
112
113         if (Echo == 1)
114               {
115                     total_us = 0;
116
117                     while (Echo == 1)
118                           {
119                           total_us++;
120
121                           _asm
122                                 nop //il nop rallenta il ciclo di lettura
tarando correttamente la lettura
123                           _endasm
124
125
126                           }
127
128   distance = total_us >>5; //divisione per 32
129
130 // oltre i 4 metri la misura perde di significato e viene visualizzato <->
131
132         if (distance>400)
133         {
134               distanceLCD[2]='>';
135               distanceLCD[1]='-';
136               distanceLCD[0]='<';
137         }
138         else
139         {
```

```
140
141         itoa (distance,distanceLCD); // converto la distanza in una stringa
142
143 //sposto le cifre in modo da mantenere lo stesso valore posizionale
144         if (!isdigit(distanceLCD[1]))
145         {
146               distanceLCD[2]=distanceLCD[0];
147               distanceLCD[1]=' ';
148               distanceLCD[0]=' ';
149         }
150
151         if (!isdigit(distanceLCD[2]))
152         {
153               distanceLCD[2]=distanceLCD[1];
154               distanceLCD[1]= distanceLCD[0];
155               distanceLCD[0]=' ';
156
157         }
158         } //fine else
159
160   WriteCharLCD (distanceLCD[0]); //scrivo la distanza sul Display
161   WriteCharLCD (distanceLCD[1]);
162   WriteCharLCD (distanceLCD[2]);
163
164   ShiftCursorLCD (LEFT); //riposiziono il cursore in modo da aggiornare
165   ShiftCursorLCD (LEFT); //solo le cifre di interesse
166   ShiftCursorLCD (LEFT);
167
168
169
170   }     //fine if (Echo ==1)
171
172 // ulteriori controlli nel programma vanno inseriti dopo questo testo
173
174   }     //fine ciclo infinito while
175 }       //fine main
```

Bibliografia

www.LaurTec.com : sito di elettronica dove poter scaricare gli altri articoli menzionati, aggiornamenti e progetti.