Roboitalia
www.roboitalia.com

Domo GPS

I progetti di Roboitalia

Hai deciso di costruire un robot?
Bene!
Iniziamo dalle brutte notizie…
Non e' facile!
Ora le buone notizie…
E' possibile, a patto di applicarsi e studiare.

di Marco Fabbri

Licenza di questo documento

Indice

Premessa

Quando si pensa ad un robot mobile, una delle prime considerazioni che viene spontaneo fare è relativa alla consapevolezza della posizione del robot; in altre parole sono rari i casi dove il robot può svolgere azioni senza sapere dov'è.
Alla domanda "dove sono" i robot, o i loro creatori, devono quindi saper rispondere, esistono in merito alcuni documenti reperibili anche in rete attraverso i quali è possibile sviluppare alcuni ragionamenti.
A meno di avere un campo di gara o di prova con dimensioni e caratteristiche ben precise, scopriamo in breve che un'applicazione ad uso generico deve tener presente molte incognite e anche imparare a muoversi mappando il territorio.
Il fascino del GPS a mio avviso è dato proprio dalla possibilità di avere sempre un riferimento assoluto indipendente dall'ambiente (a patto di ricevere il segnale).
Per un uso domestico evidentemente non è possibile usare la stessa tecnologia, ma si possono usare gli stessi concetti che sono:

1) Triangolazione: si tratta di calcolare un punto avendo come riferimento la distanza da almeno due punti noti, il calcolo consiste appunto nel trovare il triangolo che ha come vertici i due punti noti più quello incognito conoscendo la misura dei tre lati
2) Cielo: nel sistema GPS si intende con "cielo" la posizione precisa dei satelliti ai quali ci riferiamo per costruire la triangolazione, è vero che i satelliti sono geostazionari ma quelli visibili e quindi utilizzabili per triangolare cambiano in funzione della posizione del punto incognito
3) Tempo di volo: conoscendo la velocità di propagazione in un determinato mezzo di un'onda (sonora o elettromagnetica) posso ricavare dal tempo che essa impiega a percorrerla la distanza tra il punto di partenza e il punto di arrivo.

Vediamo ora come mettere in pratica un sistema simile.

Il tempo di volo.

Ad uso hobbistico è piuttosto diffuso l'utilizzo degli ultrasuoni per misurare una distanza e questo in quanto è piuttosto semplice generarli e la velocità del suono nell'aria è tale da poter produrre dei tempi di volo facilmente misurabili anche per distanze piuttosto piccole. Supponendo la velocità del suono nell'aria di 330m/s abbiamo che, volendo discriminare una distanza minima di 1mm dobbiamo riuscire a misurare un tempo di 0,001/330=3uS (microsecondi) ora per quanto possa sembrare un tempo molto piccolo, non è così assurdo pensare di misurarlo, in particolare se usiamo MCU con clock a qualche decina di MHz.
Nelle applicazioni comuni o commerciali questo sistema si usa per rilevare la distanza di un ostacolo misurando "l'eco" come nei sonar, in questo caso la distanza è doppia (andata e ritorno) ma soprattutto abbiamo un problema, non sappiamo quale sia il punto che ha prodotto l'eco e quindi non possiamo utilizzare questo punto come "punto noto".
Occorre pertanto fare in modo che l'onda sonora sia generata in un punto ben preciso e sia rilevata dal robot determinando quindi la distanza che separa il punto (satellite) dal robot. Fino a qui nulla di particolare, si tratta di posizionare la capsula trasmittente sul satellite e quella ricevente sul robot.
In questo modo ho una misura diretta tra due punti e il tempo di volo sarà senz'altro funzione della distanza (S=V*T); sarà vero inoltre che eventuali rimbalzi o percorsi effettuati dall'onda in modo non diretto arriveranno successivamente all'arrivo dell'onda diretta.

Cielo.

Nel sistema GPS la posizione dei satelliti è nota e tutti insieme trasmettono il loro segnale che tra l'altro contiene l'identificativo del satellite stesso.
Nel nostro caso la cosa non sarebbe così semplice, inoltre non avendo orologi atomici di riferimento non conosciamo il tempo nel quale è partito il segnale e questo ci precluderebbe la possibilità di calcolare il delta tempo.
Si tratta quindi di risolvere due punti fondamentali, riconoscere il satellite che sta trasmettendo e conoscere quando ha cominciato a trasmettere.
Per fare questo doteremo il robot di una trasmittente in radio frequenza e il satellite di un ricevitore.
Cosa otteniamo in questo modo? Il robot avrà la possibilità di chiamare un satellite via radio e lo farà inviando via radio un messaggio che corrisponde al nome di un satellite ben preciso, tutti i satelliti ricevono il

messaggio ma solo quello con il nome chiamato risponderà con un treno di impulsi ad ultrasuoni. Ci troviamo quindi nella seguente situazione: sappiamo qual è il satellite che ci risponde (di cui conosciamo le coordinate) conosciamo inoltre quando ha cominciato a lanciare gli ultrasuoni, in realtà prendiamo come tempo zero il momento in cui il robot termina la sua chiamata radio introducendo due errori. Il primo di questi errori è il tempo di volo del segnale radio che non è proprio zero ma direi sicuramente trascurabile, il secondo è il tempo di ritardo di elaborazione del satellite che deve riconoscere il suo nome e iniziare a trasmettere; questo tempo è fortunatamente una costante e vedremo in seguito come considerarlo.

Triangolazione

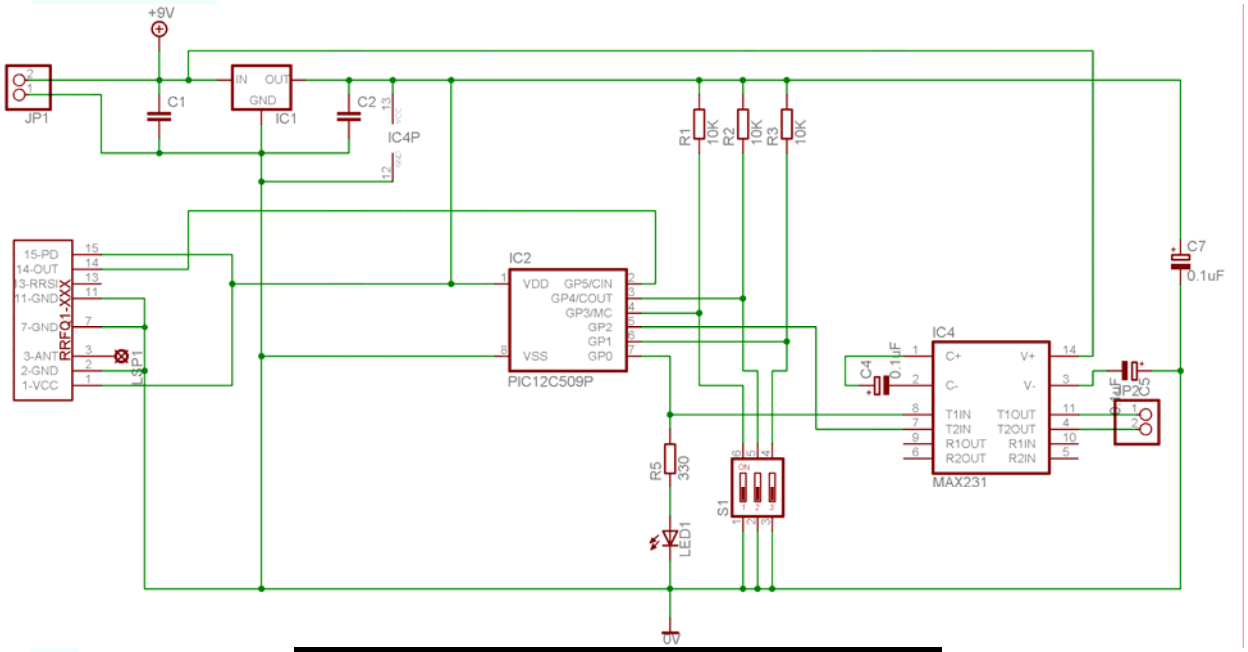
Qui il problema è esclusivamente matematico e in realtà non è nemmeno un problema visto che è gia stato risolto da Carnot qualche anno fa.
Per avere una triangolazione ci serve la lettura da almeno due satelliti (posizione in 2D, due dimensioni, valida se il robot non vola)
In pratica ci troviamo nella situazione di conoscere i tre lati di un triangolo, non mi dilungo su questo dandolo per noto o eventualmente rimandando alla consultazione di www.math.it
Unica nota, se abbiamo solo 2 satelliti saranno possibili 2 soluzioni mentre con tre satelliti solo una soluzione alla equazione, inoltre con 3 satelliti possiamo eseguire a coppie 3 diverse triangolazioni ottenendo così una media del punto che cerchiamo e riducendo l'errore (come accade per il GPS).

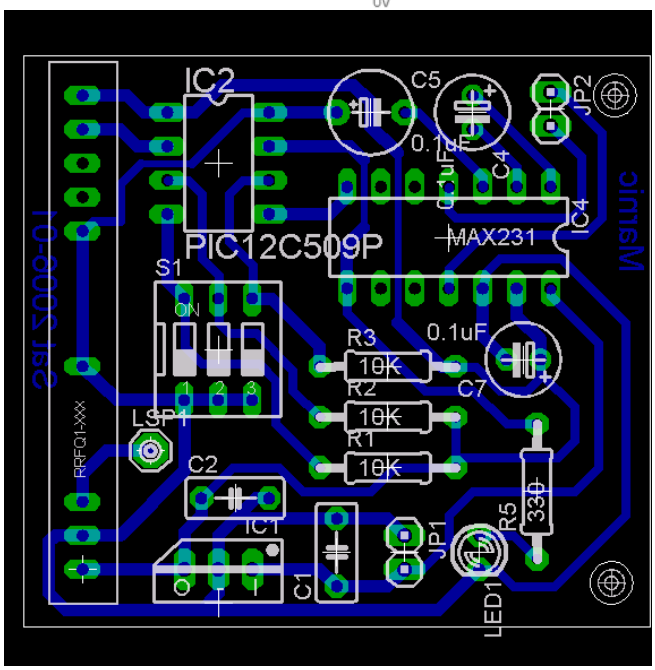
Il progetto

Fatte le premesse del caso andiamo ad analizzare l'aspetto pratico e il progetto di un sistema DomoGPS.

Satellite

Partiamo dai satelliti, come sappiamo questi devono essere in grado di ricevere un segnale radio, decifrare il messaggio, confrontarlo e in caso positivo emettere un segnale ad ultrasuoni.
Vediamo lo schema:
Cuore del sistema un PIC a 8 pin, più che sufficiente per il nostro scopo. Si occupa infatti di ricevere il codice trasmesso dal robot attraverso un modulo RX, leggere l'indirizzo al quale deve rispondere e che sarà impostato sui dip switch. I rimanenti due pin andranno a pilotare un MAX231, quest'ultimo viene utilizzato solo per elevare la tensione dai 5V del circuito ai 12V con i quali far oscillare la capsula ad ultrasuoni e ottenere una portata maggiore.
Il tutto trova posto in un piccolo circuito stampato, l'alimentazione può essere fornita con una batteria da 9V o con un piccolo alimentatore vista comunque la presenza di uno stabilizzatore a 5V.
Come è facile intuire dal fatto che abbiamo 3 pin per l'indirizzo, questo potrebbe sembrare limitato ai valori che vanno da 000 (0 in decimale) a 111 (7 in decimale). In realtà possiamo, qualora serva fare n satelliti per coprire varie aree, lavorare sul firmware del PIC e suddividerli in gruppi e programmando un valore base come indirizzo al quale sarà poi sommato quello letto sui pin, ad esempio potremmo avere 8 PIC con indirizzo base 10 (in decimale) e altri otto con 20 come indirizzo base e così via.
Per quanto riguarda il firmware niente di particolare, il PIC rimarrà costantemente in ascolto sul pin collegato al ricevitore radio fino a che non riceverà qualcosa, a quel punto confronterà il dato ricevuto con il proprio indirizzo, nel caso coincidano farà cambiare di stato i pin collegati al MAX231 con una frequenza di 40KHz per un tempo (non critico) di circa mezzo secondo, fatto ciò si rimetterà in ascolto della radio.
Questo in linea di massima, vedremo poi analizzando il codice che ci sono altri piccoli accorgimenti, sia di utilità che legati al funzionamento.
Due parole vanno spese per il modulo radio, quello nello schema è il modello RRFQ1-433 della Telecontrolli potete trovare il datasheet su www.telecontrolli.com ovviamente possono essere usati altri moduli di altre marche, uniche modifiche sono riferite al pinout per il resto mi sembra siano tutti simili come funzionamento, il pin OUT va alto quando si riceve la portante per poi seguire la sequenza 0-1 del segnale ricevuto. Questo modulo ha il limite di 4800baud, non critico per questa applicazione. E' interessante e può essere implementata la possibilità di mandare in SLEEP il pic e risvegliarlo mandando la portante, questo diminuisce i consumi e allunga la vita della batteria qualora si sia optato per questa forma di alimentazione.





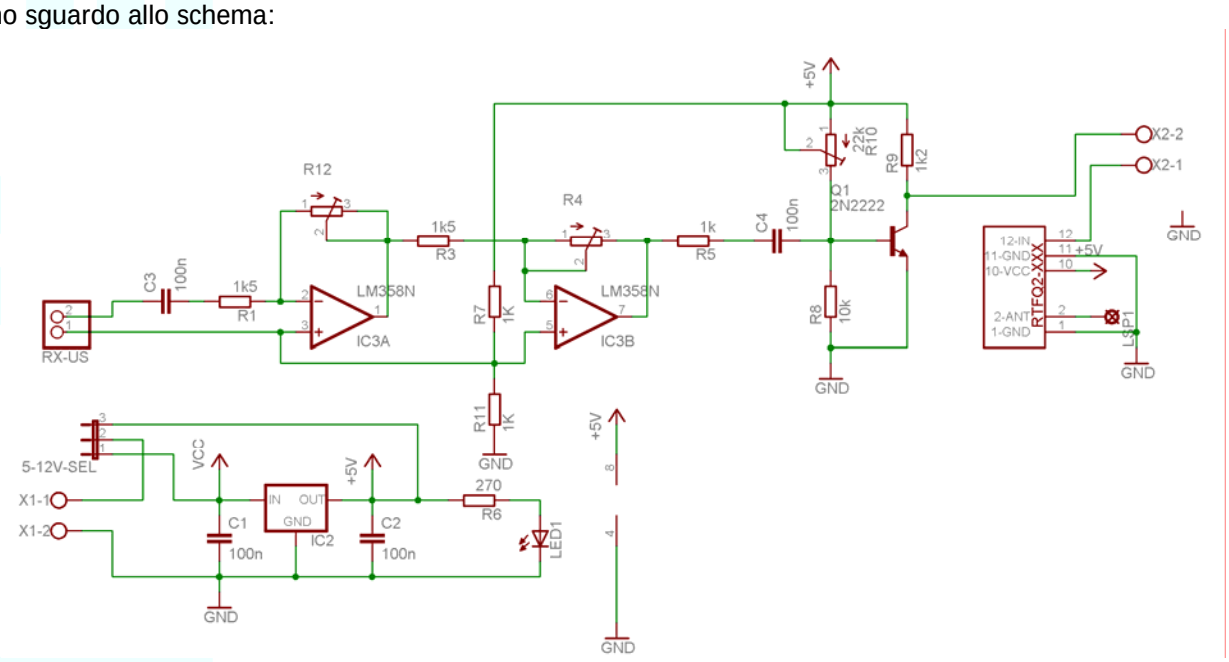
Roboboard:

La roboboard a sua volta dovrà essere in grado ti trasmettere un segnale radio e ricevere un segnale ad ultrasuoni.
Analizziamo adesso lo schema della scheda che sarà montata sul robot.
Per realizzare qualcosa che si potesse adattare un po' a tutte le realizzazioni lo schema proposto possiamo immaginarlo come una scatola nera con un ingresso e una uscita.
Sull'ingresso manderemo un segnale seriale codificato 4800/8/N/1 dove scriveremo semplicemente l'indirizzo del satellite che vogliamo interrogare.
Sull'ingresso avremo un impulso a 1 (5V).
Il ritardo con cui riceveremo questo impulso sarà proporzionale alla distanza che stiamo cercando di misurare.
In questo modo si può collegare il sistema a qualsiasi scheda "robotica" unici requisiti sono:

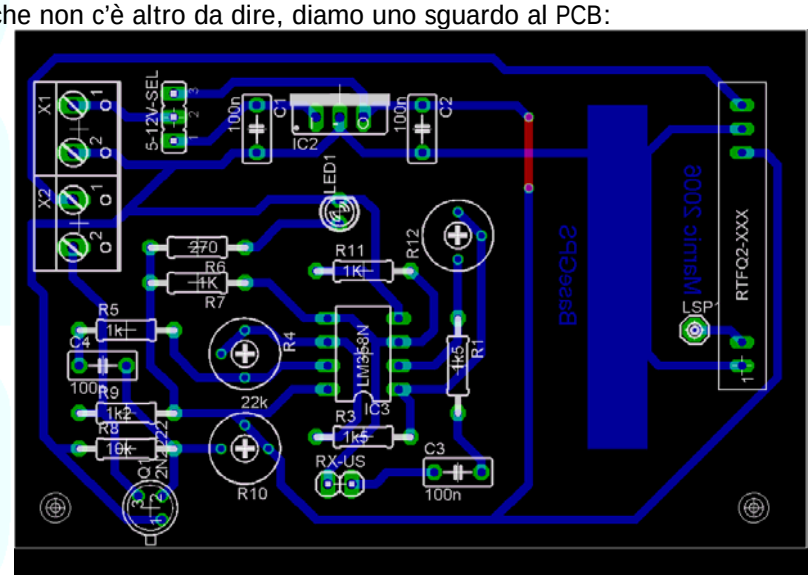
1) possibilità di modulare un segnale seriale
2) capacità di misura di un tempo tra uno start software e uno stop legato allo stato di un pin
3) capacità di calcolo

Il terzo punto è molto legato al tipo di MCU usata, ci potrebbe stare il caso in cui il robot non esegue nessun calcolo ma manda semplicemente i dati ad un PC vero cervello "remoto" del robot.

Per i primi due punti direi che sono caratteristiche ormai comuni se non indispensabili.
Uno sguardo allo schema:



In basso vediamo la sezione di alimentazione dove è stato predisposto un ponticello, il circuito funziona a 5V per cui è presente un regolatore tipo 7805, questo verrà utilizzato inserendo un jumper tra i pin 1 e 2 qualora sul robot sia disponibile una tensione maggiore.
Nel caso sul robot abbiamo i 5V dovremo cortocircuitare i pin 2 e 3 bypassando così il regolatore che non riuscirebbe a funzionare avendo un dropout maggiore di zero.
In alto a destra il modulo TX radio, sempre della Telecontrolli modello RTFQ2-433, è solo alimentato e collegato al pin della MCU per ricevere il segnale seriale.
Tutto il resto è un amplificatore realizzato con due operazionali (il modello non è critico), i trimmer R4 e R12 (47K) ci permettono di regolare il guadagno dei due stadi.
L'uscita dell'amplificatore va poi ad un transistor usato come comparatore e manda a zero il morsetto X2-2 quando viene ricevuto il segnale ad ultrasuoni, questo segnale sarà rilevato dal robot per il calcolo del tempo.
Sullo schema direi che non c'è altro da dire, diamo uno sguardo al PCB:



Tenete presente che i PCB in questo file non sono in formato 1:1 ma adattati alle esigenze grafiche della pagina.
Anche qui piste di spessore tale da rendere facilmente realizzabile lo stampato da tutti, un solo ponticello e disposizione piuttosto pulita.
Non dimenticatevi di collegare un'antenna al pad LSP di opportune dimensioni in base alla frequenza del vostro modulo radio.

Il codice

Premetto che il codice è sviluppato in PicBasic magari non simpatico a tutti ma la relativa semplicità e i commenti che seguiranno vi permetteranno di adattarlo ad altri linguaggi o controllori.

```
INCLUDE "modedefs.bas"
ASM
   bsf         STATUS, RP0
   call        0x3ff
   movwf   OSCCAL
   bcf STATUS, RP0
ENDASM
DEFINE NO_CLRWDT 1
DEFINE OSC 4
CMCON = 7   ' Setta GPIO.1 in digitale
GRUPPO con 10    'Gruppo sensori di appartenenza da sommare all'indirizzo
INDIRIZZO var byte
RXIND VAR BYTE
N VAR BYTE
TRISIO = %00111010
GPIO.0=0
GPIO.2=0
INDx1 var GPIO.3
INDx2 var GPIO.4
INDx4 var GPIO.1
RADIO var GPIO.5
INDIRIZZO=GRUPPO
pause 1000
IF INDx1=0 THEN INDIRIZZO=INDIRIZZO+1
IF INDx2=0 THEN INDIRIZZO=INDIRIZZO+2
IF INDx4=0 THEN INDIRIZZO=INDIRIZZO+4
FOR N=1 TO INDIRIZZO
        GPIO.0=1
        PAUSE 500
        GPIO.0=0
        PAUSE 500
NEXT N
GPIO.2=0
loop:
SERIN RADIO,N2400,["M"],RXIND
IF RXIND=INDIRIZZO THEN urla
GOTO loop
urla:
GPIO.0=1
PAUSE 1
for N=1 to 10
ASM
  bsf GPIO, 0 ; Set bit 0 on GPIO.0
  bcf GPIO, 2 ; Clear bit 0 on GPIO.2
  nop
  nop
  nop
  nop
  nop
  nop
  bcf GPIO, 0 ; Clear bit 0 on GPIO.0
  bsf GPIO, 2 ; Set bit 0 on GPIO.2
  nop
  nop
  nop
  nop
  nop
  nop
  nop
ENDASM
NEXT N
GOTO loop
```

Quello che vedete sopra è il firmware dei satelliti. Nelle prime righe oltre all'include ci sono quattro righe racchiuse tra i tag ASM e ENDASM, sono righe in assembler che dicono al PIC 12F629 di usare il clock interno e di usare come valore di calibrazione quello memorizzato nella locazione di memoria 0x3ff, vedremo che questo valore è piuttosto importante.
Proseguendo abbiamo il settaggio dell'HW del PIC e la dichiarazione delle variabili tra cui la costante GRUPPO che come accennato analizzando lo schema del satellite permette di avere più degli 8 indirizzi disponibili con i soli 3 dip switch.
Incontriamo poi tre IF che vanno a controllare lo stato dei dip switch e determina l'ID del dispositivo, questa parte del codice è eseguita una sola volta per cui è necessario impostare l'ID prima di accendere il satellite.
Segue un ciclo FOR-NEXT che farà lampeggiare il led un numero di volte pari all'ID del satellite, questa parte non è necessaria ma può essere utile per controllare visivamente l'ID impostato e soprattutto il gruppo e visto che c'è spazio in memoria è stata prevista.
A questo punto inizia il LOOP del firmware composto solo da 3 righe che vanno a leggere la porta collegata al ricevitore radio, qualora venga ricevuto il carattere di controllo "M" allora si controlla l'ID che segue, in caso coincida con il proprio il controllo passa alla routine "urla" diversamente si rimane in ascolto.
Perché un carattere di controllo? Uno dei motivi è che il trasmettitore radio visto che c'è potrebbe essere usato dal robot per comunicare anche con altri dispositivi e in questo caso il satellite non deve intervenire anche se viene trasmesso un numero che coincide con il suo ID, stessa cosa per trasmissioni che potrebbero giungere da altre fonti.
La routine urla altro non fa se non commutare in opposizione due porte collegate al MAX, per avere i 40KHz ho usato direttamente l'assembler giocando con i nop il tutto dentro un ciclo FOR-NEXT che determina la lunghezza del burst.
Prima del ciclo notate un "PAUSE 1" questa istruzione genera una pausa di 1 millisecondo prima di avviare la trasmissione degli ultrasuoni, è una riga molto dipendente dal controllore e dal linguaggio che userete sul robot, vediamo perché:
Sulla scheda del robot nelle prove ho usato un 16F877 programmato sempre in Picbasic, come vedremo il robot dopo la trasmissione dell'ID deve mettersi in ascolto sulla porta che riceve gli ultrasuoni avviando un timer, ebbene queste operazioni non sono a tempo zero ma, in gran parte per colpa del compilatore, c'è un tempo tecnico che per distanze brevi (30-40cm) fa si che gli ultrasuoni arrivino prima che il robot sia pronto a riceverli invalidando la misura. Per questo motivo è stata introdotta una pausa (costante) che permetta al sistema di stabilizzarsi, vedremo in seguito come gestire questo delta T.
Per quanto il ragionamento con la pausa da 1 sia corretto, in fase di test allineando le capsule TX di 3 satelliti uno dei tre mi generava una misura diversa, solo per caso notai che anche i 40KHz non erano così precisi come per gli altri due, la causa di questo è da attribuire alla tolleranza dell'oscillatore interno, ecco come promesso che ritorniamo al valore di calibrazione del clock interno, se vogliamo aumentare la precisione del sistema è necessario ridurre al minimo le differenze di clock tra i vari PIC, per fare questo io ho usato un SW di microchip "OSCCAL" fornito insieme al PicKit1 ma scaricabile dal sito che prova diversi valori di calibrazione confrontandoli con una frequenza generata dal programmatore.
La lunghezza del burst di ultrasuoni non incide sulla misura, noi misureremo infatti il primo fronte o almeno ci proviamo, vedremo che questo particolare sarà una fonte di errore.
Da notare che quando il satellite "urla" il led emette un piccolo lampeggio.
Non avendo altro da aggiungere propongo un esempio di codice sul lato robot in modo da analizzare le problematiche accennate e rimaste in sospeso.

```
N var byte
VELUS var word
TARA var word  ' è il tempo di ritardo in incrementi contatore (2uS a 20MHz)
TEMPO var word ' è il "tempo" in incrementi del contatore 2uS ogni digit a 20MHz
TEMPOV var word ' tempo vero in mS
DIST var word    'Spazio in mm
RADIO var PORTB.5
RXUS var PORTD.0
GRUPPO var byte

VELUS = 33
TARA=110
GRUPPO = 10

'I/O
TRISA = %11111111
TRISB = %01000011
TRISD = %11111111

loop:
For N=13 to 15
serout RADIO,N2400,["M",N]
RCTIME RXUS,0,TEMPO
TEMPOV=(TEMPO-TARA)   'tempo vero in uS
DIST=(TEMPOV/5*VELUS)/10       'Distanza in mm
HSEROUT ["Timer *",dec N,".. *",dec TEMPO,"  TempoVOLO...","<*",dec TEMPOV,"us> Spazio... <*",DEC DIST,"mm>",13,10]
pause 2000
next N
HSEROUT [13,10]
goto loop
```

E' un piccolo esempio ma contiene tutte le problematiche incontrate.
Iniziamo dall'etichetta "loop:" vediamo un ciclo da 13 a 15, sono gli ID dei satelliti usati, in questo caso 13, 14 e 15. quindi ciclicamente chiamiamo uno dei tre satelliti, mandando alla radio il segnale seriale contenente il carattere di controllo "M" e l'ID contenuto nella variabile N.
Il comando RCTIME in picbasic in pratica si mette a contare un tempo da quando viene lanciato il comando a quando la porta RXUS non cambia stato.
La riga seguente sottrae la "tara" al tempo misurato per ottenere il tempo vero di volo, vedremo come determinare la tara.
Segue un calcolo molto artificioso per convertire il tempo in distanza, questa riga ero in dubbio se mostrarla così ma visto che è un punto cruciale è visibile in tutta la sua assurdità: il PIC da me usato ha limiti nel calcolo a virgola mobile per cui sono stati scomposti gli operandi in modo da ottenere il minor errore possibile, giocando sia con le unità di misura sia con le scomposizioni, il messaggio qui è solo quello di pensare che se non avete calcoli in virgola mobile e ad esempio il limite di 65535 per le variabili non è così semplice passare dai microsecondi ai metri o centimetri senza perdere parti importanti dei numeri.
L'HSEROUT è semplicemente un comando che mi visualizza sul PC i dati ricevuti ed elaborati.
Bene, passiamo ora alla taratura vera e propria e alla definizione dei tempi ecc.
Se siete degli strumentisti avrete già confidenza con lo zero e lo span, diversamente provate a seguire il ragionamento.
Abbiamo impostato 1mS di ritardo sui satelliti, sicuramente superiore al tempo necessario alla scheda robot per mettersi in ascolto, quindi se noi accendiamo il sistema e facciamo una misura avendo cura di tenere le capsule ultrasuoni a contatto quindi con distanza zero quello che leggeremo sarà un tempo uguale al nostro millisecondo meno il tempo che impiega il PIC del robot per mettersi in ascolto e attivare il conteggio.
Questo tempo è molto importante perché costituisce la nostra "tara", di fatto è il valore che dobbiamo togliere alla lettura affinché questa sia uguale a zero.
A questo punto dobbiamo riprogrammare il nostro sistema inserendo questo valore. Abbiamo tarato lo zero.
Ci resta lo span ovvero dobbiamo regolare la lettura a fondo scala. In realtà il valore di span lo tariamo andando ad agire in modo un po' poco corretto sulla velocità del suono.
Quello che a noi interessa è la misura di una distanza o spazio che come sappiamo è data dalla seguente relazione S=V*T ovvero velocità per il tempo.

Poniamo le due capsule ad una distanza ben nota dell'ordine dei 5-6 metri, attenzione, se a questa distanza non riuscite ad essere precisi nel misurarla con il metro è molto meglio ridurre finchè non siete sicuri di poter misurare con la precisione del millimetro o quasi, piccolo particolare, non pensate di fare questa prova appoggiando le capsule in terra o su una parete, in questo caso il suono si propagherebbe attraverso il cemento o il ferro con velocità completamente diverse da quanto ci aspettiamo in aria libera. Bene, fate la misura e calcolate la velocità, avete spazio (misurato da voi) e tempo (misurato dal sistema).
Riprogrammate il PIC variando il valore della velocità.
A questo punto avete fissato due punti abbastanza lontani in un sistema di misura di una grandezza lineare per cui avrete un valore tarato e vero (a meno degli errori) lungo tutto il raggio di azione del sistema, il mio prototipo funziona in un raggio di 6-7 metri.

Fonti di errore note

Come sempre teoria e pratica spesso divergono, le ragioni sono facili, a volte semplifichiamo la teoria, sempre abbiamo a disposizione dispositivi non ideali.

1) Errore di semplificazione della teoria, la velocità del suono nell'aria non è costante ma dipende dalla temperatura in modo importante dall'umidità in modo minore, segue una tabella dove è riportata la velocità a diverse temperature e una solo a titolo didattico sulla velocità in alcuni materiali.

Sostanza	V (m/s)
Aria	344
Anidride Carbonica	259
Alcool Etilico	1207
Acqua	1498
Rame	3750
Ferro	5120
Vetro	5170

Temperatura °C	V (m/s)
0	331.5
+5	334.5
+10	337.5
+15	341.5
+20	343.4
+25	346.3
+30	349.2

Come potete vedere tra 10 e 20 gradi centigradi abbiamo una differenza di 5.9m/s se facciamo due calcoli vediamo che 5 metri a 10 gradi si riducono a 4.91 a 20 gradi un errore di quasi un centimetro. Se l'aria poi è satura di anidride carbonica?
L'idea per correggere questo errore è quella di inserire un sensore di temperatura e visto che la differenza è pressoché lineare variare il valore che abbiamo messo in funzione del delta temperatura rispetto a quello del momento della taratura.

2) Errore dovuto ai componenti, le capsule ad ultrasuoni non hanno una accelerazione infinita per cui le prime 2 oscillazioni non sono al massimo della potenza come le successive, questo comporta che ai limiti della distanza di funzionamento abbiamo il rischio di perderci nella capsula di ricezione il primo o addirittura anche il secondo fronte, la conseguenza è di leggere il terzo periodo che ritarda rispetto al primo di 2*1/40KHz = 50microsecondi che alla velocità di 343.5m/s equivalgono a 1.7centimetri, 8mm se ci perdiamo solo il primo fronte. Una soluzione ci sarebbe ma è piuttosto complessa da implementare, si tratta non solo di misurare il tempo di volo ma anche il numero di fronti del burst, sapendo ad esempio che ne sono stati inviati dieci, se ne leggiamo solo 8 sappiamo cosa aggiungere.

Difficoltà e miglioramenti

La capsula ricevente non è omnidirezionale come ci servirebbe per cui dovrà essere puntata verso l'alto per cercare di captare il più possibile a 360° l'ideale sarebbe riuscire a mettere un cono rovesciato in modo da riflettere il più possibile.
Una alternativa potrebbe essere mettere 4-5 capsule tutte in parallelo e disposte in modo da coprire tutta la circonferenza.
Ho provato diversi circuiti per amplificare gli ultrasuoni sono convinto che si possa migliorare