Sessione ordinaria 2012
Seconda prova scritta
M272.2.1/1

Ministero dell'Istruzione, dell'Università e della Ricerca

M272 – ESAME DI STATO DI ISTITUTO TECNICO INDUSTRIALE

CORSO DI ORDINAMENTO

Indirizzo: ELETTRONICA E TELECOMUNICAZIONI

Tema di: SISTEMI ELETTRONICI AUTOMATICI
(Testo valevole per i corsi di ordinamento e per i corsi del progetto sperimentale "Sirio")

Si vuole realizzare un dispositivo elettronico che sia in grado di misurare la distanza da un oggetto, utilizzando una coppia di capsule ultrasoniche funzionanti da trasmettitore Tx e da ricevitore Rx opportunamente interfacciate (vedi figura Fig. 1).

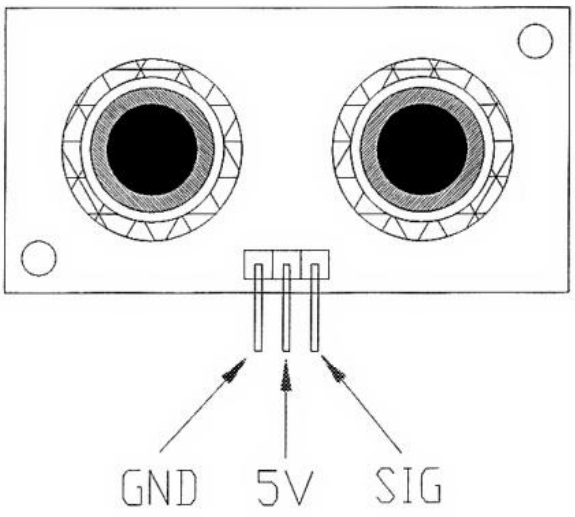


Fig. 1

Il trasduttore ultrasonico contiene un circuito elettronico di controllo che permette di attivare il dispositivo (il segnale SIG del trasduttore ultrasonico è di tipo bidirezionale) e che produce in uscita un impulso (T_3) di durata proporzionale alla distanza (vedi figura Fig. 2).

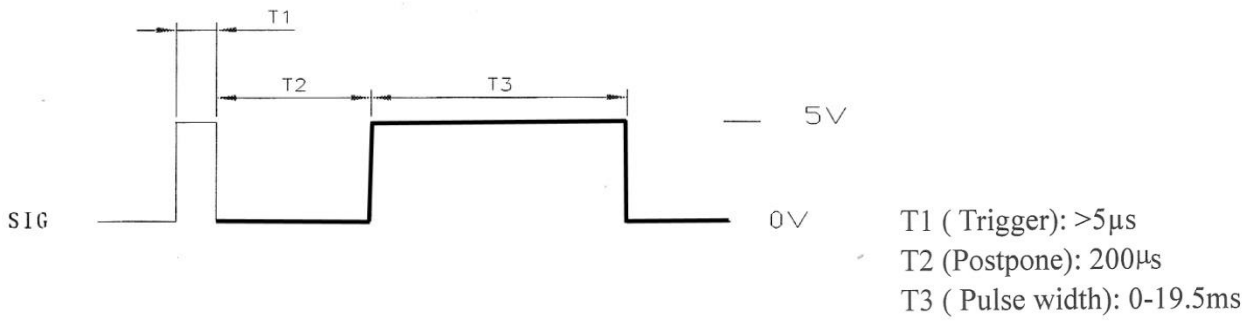


Fig. 2

La distanza D [m] può essere calcolata, conoscendo la durata di T_3 e la velocità V del suono nell'aria attraverso la seguente formula:

$$D = \frac{V\,T_3}{2}$$

Sessione ordinaria 2012
Seconda prova scritta
M272.2.1/1

Ministero dell'Istruzione, dell'Università e della Ricerca

M272 – ESAME DI STATO DI ISTITUTO TECNICO INDUSTRIALE

CORSO DI ORDINAMENTO

Indirizzo: ELETTRONICA E TELECOMUNICAZIONI

Tema di: SISTEMI ELETTRONICI AUTOMATICI
(Testo valevole per i corsi di ordinamento e per i corsi del progetto sperimentale "Sirio")

La velocità del suono dipende dalla temperatura secondo la seguente equazione:

$$V = a + b\ Temp \text{ [m/s]}$$

Dove le costanti a e b valgono rispettivamente: a = 331.5 [m/s] e b = 0.62 [m/s °C]
T_{emp} rappresenta la temperatura in gradi centigradi [°C].
Per misurare la temperatura T_{emp} è utilizzato un sensore con uscita digitale a 9 bit in complemento a due di tipo proporzionale (vedi tabella Tab. 1):

Temperature	Digital Output	
	Binary	Hex
+125°C	0 1111 1010	0FAh
+25°C	0 0011 0010	032h
+0.5°C	0 0000 0001	001h
0°C	0 0000 0000	000h
−0.5°C	1 1111 1111	1FFh
−25°C	1 1100 1110	1CEh
−55°C	1 10010010	192h

Tab. 1

Il candidato, formulate le eventuali ipotesi aggiuntive:

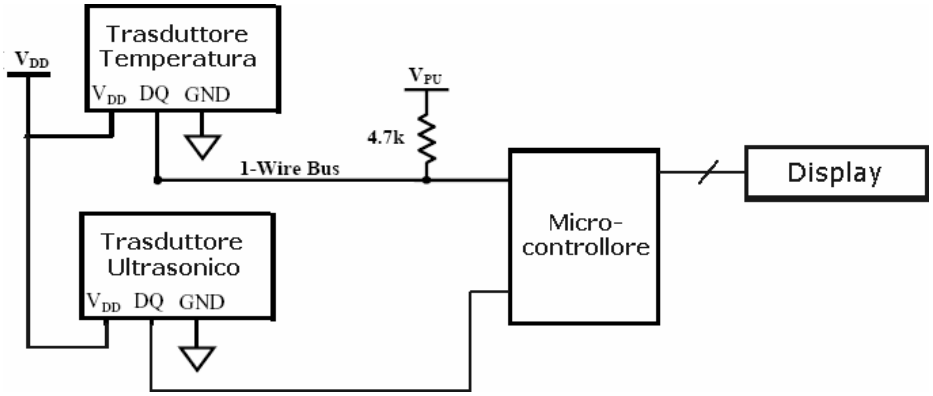
a. descriva attraverso quali periferiche del microcontrollore o microprocessore intende misurare la durata di T_3 e il valore di temperatura T_{emp};
b. disegni il diagramma di flusso per la gestione dei sensori, evidenziando le modalità di interfacciamento se in "polling", in "interrupt" o in modalità "mista";
c. descriva almeno una metodologia software e/o hardware per rilevare il periodo T_3;
d. indichi una metodologia per visualizzare le grandezze rilevate dai sensori;
e. progetti il codice di gestione di almeno uno dei due sensori utilizzati;
f. valuti la distanza massima rilevabile dal sensore ultrasonico supponendo la temperatura T costante e pari a +20°C.

Inoltre il candidato ipotizzando una variazione di temperatura da +20°C a +40°C, valuti la variazione relativa della distanza rilevata.

Durata massima della prova: 6 ore.
È consentito l'uso di manuali tecnici e di calcolatrici non programmabili.
Non è consentito lasciare l'Istituto prima che siano trascorse 3 ore dalla dettatura del tema.

Svolgimento

(a)
Il sensore di temperatura ha un'uscita digitale a 9 bit, emettendo un codice binario proporzionale al valore Temp richiesto. Possiamo pensare si tratti di un dispositivo 1-Wire. Un sistema di comunicazione 1-Wire (progettato e sviluppato da Dallas Semiconductors) si basa su un dispositivo, detto master (il microcontrollore), che gestisce la comunicazione, scambiando informazioni e fornendo alimentazione a più dispositivi, detti slave (il sensore di temperatura), mediante un singolo cavo twisted-pair: un singolo filo (oltre quello di massa) permette la comunicazione tra più dispositivi e contemporaneamente fornisce l'alimentazione necessaria al loro funzionamento.
Il master ha una uscita open-drain o tri-state, con un resistore di pull-up a V_{PU}, gli slave hanno un'uscita open-drain con cui possono portare a 0 il segnale sul bus.
I dati sulla rete 1-Wire vengono trasferiti in "time slots", cioè porzioni di tempo di lunghezza prestabilita, tra 60 e 120µs, in cui è possibile scambiare informazioni sul bus. Il sistema non richiede clock, poiché ogni dispositivo slave 1-Wire contiene un oscillatore interno che si sincronizza durante i fronti di discesa del clock del master.

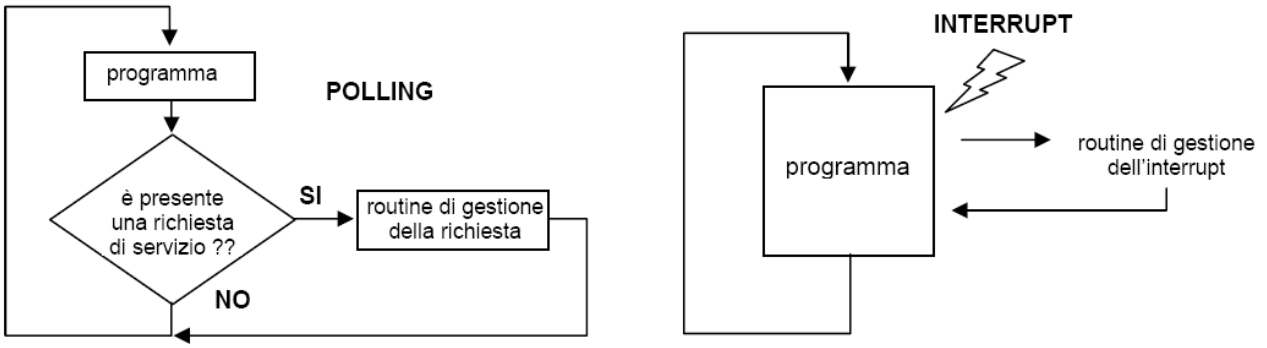


Il dato proveniente dal traduttore di temperatura viene acquisito mediante opportuna routine di acquisizione sul pin RB1 e memorizzato in questo modo:

- MSB nel file register tempsegno,
- I rimanenti 8 bit in tempvalore

La forma d'onda proveniente dal trasduttore ultrasonico trattandosi di segnale digitale 0-5V viene mandata ad un pin RB0 in ingresso al PIC.

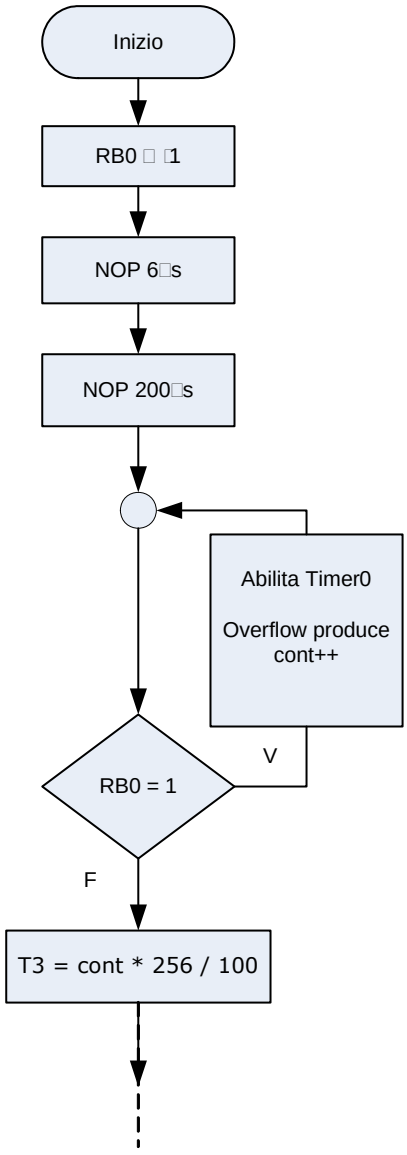
(b)
Un microcontrollore che ha in carico l'esecuzione di un programma, svolge in sequenza le istruzioni programmate secondo un ordine predeterminato e inoltre deve anche rispondere a richieste che gli giungono in modo casuale dai dispositivi con i quali interagisce.



A fronte di una richiesta di servizio casuale, il microprocessore deve abbandonare il regime corrente e avviare una procedura di gestione della richiesta di servizio. Due sono le modalità

con le quali il microcontrollore si dispone ad accogliere la richiesta di servizio proveniente da un dispositivo esterno:

a. con il Polling, che significa interrogazione, il controllo delle richieste segue uno schema prefissato, ed è lo stesso programma in esecuzione sul microcontrollore a prevedere un controllo periodico delle periferiche per intercettare quelle con richieste di servizio pendenti. Il polling e quindi un metodo software: il controllo è tutto a carico del programma in esecuzione che periodicamente deve interrogare le periferiche per vedere se è presente una richiesta di servizio.
b. Con l'interrupt è il dispositivo a lanciare la richiesta di servizio, la quale viene ricevuta dall'hardware del microcontrollore (mentre il software in esecuzione non ha il compito di sondare le richieste di servizio, come nel polling). Non appena un dispositivo lancia sulla linea una richiesta di interrupt (mediante un opportuno segnale: settando un pin al livello logico alto, oppure effettuando un cambio di stato su un pin), il microcontrollore sospende il programma corrente e passa ad eseguire la routine di gestione dell'interrupt memorizzata all'interno del programma stesso. Dopodichè il microcontrollore torna a eseguire il programma corrente dal punto dove era stato interrotto.



Dato che per attivare il trasduttore ultrasonico e quindi per produrre in uscita un impulso di durata $T_2 + T_3$ è necessario aver inviato prima un impulso > 5μs si preferisce un interfacciamento in polling (a meno di non dover far altro con il microcontrollore durante il tempo T_2, usando in tal caso l'interrupt per non tenerlo impegnato durante quei 200μs).
Un diagramma di flusso per rilevare il tempo T3 è dato ponendo RB0 in uscita, emettendo un segnale alto verso il SIG del trasduttore della durata di 6μs, ottenuto con 6 No Operation (NOP). Ponendo quindi RB0 in ingresso per ricevere un segnale dal trasduttore, dopo avere aspettato un tempo pari a 200μs è noto che il livello del segnale si porti alto, cosi, mendiante un ciclo e finché il segnale rimane alto, si valuta il tempo T3 usando il timer del microcontrollore e una variabile di conteggio.

(c) (e)
La codifica del programma avviene in PICC per PIC16. Per misurare il tempo T3 si collega il SIG del trasduttore ultrasonico al piedino RB0 del microcontrollore (si considera un PIC16) il quale viene dapprima settato in uscita (RB0 = 0) in modo da inviare sul pin SIG verso il trasduttore un segnale di 5V della durata T1 > 5μs usando una funzione da noi realizzata DelayUs(n) e definita in una libreria.h importata nel programma. Utilizziamo quindi DelayUs(6) per realizzare un ritardo di 6μs.
Successivamente, posto in ingresso RB0 (RB0 = 1) si attende 200μs usando DelayUs(200), quindi si controlla che dall'esterno il trasduttore ultrasonico di prossimità abbia portato RB0 a livello logico alto. A questo punto si avvia un conteggio con timer, finché il segnale su RB0 non si sia portato sul livello basso. La durata dell'impulso T3 va da 0 a 19,5ms in base alla distanza rilevata. Per il conteggio viene impiegato il Timer0 del microcontrollore, che fa capo al registro TMR0.
Il Timer è un semplice contatore ad 8 bit e ad ogni quarto di periodo del clock esso incrementa il suo valore (da 0 a 255). Raggiunto il valore di 255, al clock successivo, il registro effettua un overflow e riprende il suo conteggio da 0. L'evento di overflow influisce sul bit T0IF del registro INTCON, il quale subisce una transizione da 0 a 1.
Il clock del PIC16 si considera impostato a f_{osc} = 4MHz, così che la frequenza del ritardo base del registro TMR0 vale: $f_{osc}/4$ = 1MHz (ossia il ritardo base vale 1μs). Si farà incrementare una variabile di conteggio cont ad ogni overflow:

$t_{ritardo\ overflow} = 1\mu s \cdot 256 = 256\mu s$

Il tempo di ritardo può essere eventualmente amplificato mediante il prescaler che è un divisore di frequenza: riceve in ingresso la frequenza fosc/4 e la restituisce divisa per 2, 4, 8, ecc. fino a 256. Il fattore di divisione può essere programmato tramite la combinazione di tre bit PS2, PS1, PS0 del registro OPTION_REG. Settiamo il prescaler come di default a 1:2.

```
OPTION=0b10000000;
// Impostazione Registro Opzioni
// bit 0 -> PS0 Prescaler Rate Select bit 0
// bit 1 -> PS1 Prescaler Rate Select bit 1
// bit 2 -> PS2 Prescaler Rate Select bit 2
// bit 3 -> PSA Prescaler assegnato al Timer0 (1=al Watchdog Timer)
// bit 4 -> T0SE Incremento Timer0 su transizione: 0=low->high 1=high->low
// bit 5 -> T0CS Clock Timer0 da clock interno (1=su transizione pin T0CKI)
// bit 6 -> INTEDG Interrupt (INT) edge (1=salita 0=discesa)
// bit 7 -> RBPU Resistenze pull-up su porta B (1=off 0=on)
```

Il valore di partenza di conteggio del Timer0 si imposta a zero:

```
TMR0=0
```

Noto che T3 vale al massimo 19,5ms, si avrà in luogo di tale valore un conteggio massimo pari a:

cont = 19,5ms/256µs = 76

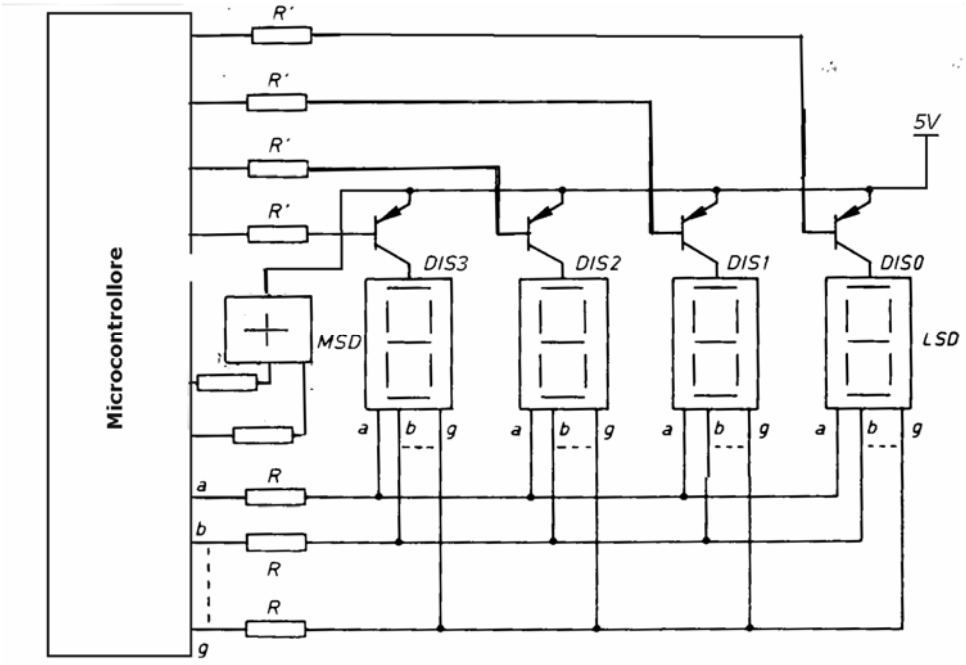
e in ogni caso un conteggio variabile tra 0 e 76 per ampiezze di impulso T3 variabili tra 0 a 19,5ms.

```
int cont;       // Contatore per il Timer
long T3, Temp, tempsegno, tempvalore, V, D;
// T3:          Tempo di misura T3 espresso in decine di ms
// Temp:        Temperatura
// tempsegno: MSB del codice a 9 bit fornito dal sensore di temperatura (bit di segno)
// tempvalore: Rimanenti 8 bit della misura di tempartura
// V:           Velocità del suono
// D:   Distanza espressa in cm

void main (void) {
  TRISB=0b00000000;             // RB0 in uscita
  RB0 = 1;                      // RB0 a 5V
  DelayUs(6);                   // Attende 6µs, produce impulso T1
  RB0 = 0;                      // RB0 a 0V, fine dell'impulso di attivazione del trasduttore
  TRISB = 0b00000001;           // RB0 in ingresso
  DelayUs(200);                 // Attende 200µs
  while (RB0 == 1) {
      TMR0 = 0;             // Timer0 = 0
      if (T0IF) {           // Overflow del timer0
          cont++;           // Incremento il contatore
          T0IF = 0;         // Resetta il flag del timer0
      }
  }
  T3 = cont * 256 / 100;    // Calcolo il tempo T3 espresso in decine di ms: T3 * 10^-4
```

Continuando il programma, per ricavare il valore della distanza, programmiamo la parte inerente la temperatura. Il bit di segno della temperatura è il MSB fornito in seriale dal trasduttore che abbiamo memorizzato nel LSB del file register tempsegno.
Se tempsegno = 0 allora consideriamo i rimanenti 8 bit, convertendoli in decimale e ottenendo un numero pari al valore della temperatura moltiplicato per due (es: 125 °C -> 250) quindi dividiamo per 2. Se tempsegno = 1 allora per recuperare il valore di temperatura in decimale dobbiamo effettuare prima l'operazione inversa del complemento a 2, ossia: 256 – tempvalore e poi dividere per 2. L'equazione che useremo per conoscere la velocità del suono in quest'ultimo caso sarà sottrattiva. In ultimo ricaviamo la distanza D espressa in centimetri. Successivamente mediante una opportuna funzione sarà possibile rappresentare il valore ottenuto su display.

```
if (tempsegno == 1) {
      Temp = (256 - tempvalore) / 2;
      V = (33150 - 62 * Temp) / 100 ;
}
else
{
      Temp = tempvalore / 2;
      V = (33150 + 62 * Temp) / 100 ;
}

D = V * T3 / 2 / 100; // risultato in centimetri
```

(d)
Per visualizzare le grandezze rilevate dai sensori è possibile utilizzare un display LCD oppure realizzarlo mediante 4 display a 7 segmenti con dot pitch (necessario per separare le cifre decimali) e un display di segno.
Per visualizzare la distanza fino all'ordine del centimetro, basterebbero 3 display 7 segmenti, mentre per quanto riguarda la temperatura, necessita di un display di segno ± e per quanto riguarda le cifre, di dover rappresentare le centinaia ma anche i decimi di grado centigrado.



I 4 display 7 segmenti con dot pitch, del tipo anodo comune, sono collegati in serie (con i rispettivi catodi collegati tra loro) e vengono pilotati usando gli stessi pin del PIC, il quale mediante un altro pin agisce di volta in volta abilitando l'anodo del display destinato a ricevere il numero da visualizzare. Così facendo si sfrutta la persistenza dell'immagine sulla retina dell'occhio, per creare l'illusione che i

numeri riportati sui display siano fissi, mentre in realtà vengono aggiornati rapidamente in sequenza. Usiamo 8 piedini di PORTC vanno collegati ai catodi dei display: RC0 con dp, RC1 con c, RC2 con d, RC3 con e, RC4 con b, RC5 con a, RC6 con f, RC7 con g.
Per illuminare di volta in volta un display il PIC dovrà mandare il rispettivo anodo a 5V. La corrente inviata dal microcontrollore però non è sufficiente ad accendere nitidamente i LED dei 3 display, pertanto è conveniente pilotarli mediante tre transistor PNP in configurazione ON/OFF, che separano il microcontrollore dai display; si è scelto di utilizzare i BC307, di tipo general purpose. I piedini del PIC che pilotano gli anodi dei display sono RB4, RB5, RB6 e RB7 e sono collegati in base ai rispettivi transistor, mediante una resistenza da 2,2 kΩ. Gli emettitori dei PNP vanno all'alimentazione di 5V, i collettori agli anodi dei display a LED.
Il display di segno è alimentato da 5V sull'anodo e pilotato da RB3 e RB2.

(f)
Nel caso T = 20°C:
$V = a + (b \cdot Temp) \quad \rightarrow \quad V_{20} = 331,5 + (0,62 \cdot 20) = 343,9$ m/s
$D = V \cdot T_3 / 2 \quad \rightarrow \quad D_{20} = 343,9 \cdot 19,5 \cdot 10^{-3} / 2 = 3,35$ m

Nel caso T = 40°C
$V_{40} = 331,5 + (0,62 \cdot 40) = 356,3$ m/s
$D_{40} = 343,9 \cdot 19,5 \cdot 10^{-3} / 2 = 3,47$ m

Nel passaggio della temperatura da 20 °C a 40 °C la misura della distanza subisce una variazione:
$\Delta D = D_{40} - D_{20} = 12$ cm

Soluzione tema di maturità tecnica industriale in "Sistemi Elettronici Automatici"

indirizzo "Elettronica e Telecomunicazioni" a.s. 2011/2012

prof. Giuseppe SPALIERNO – docente di "Sistemi Elettronci Automatici"
I.T.T. "M. PANETTI" - BARI

Descrizione generale e ipotesi aggiuntive

Si tratta di un sistema di acquisizione dati a due canale: distanza e temperatura.

Ciascuno dei due trasduttori utilizzati fornisce un'uscita digitale:

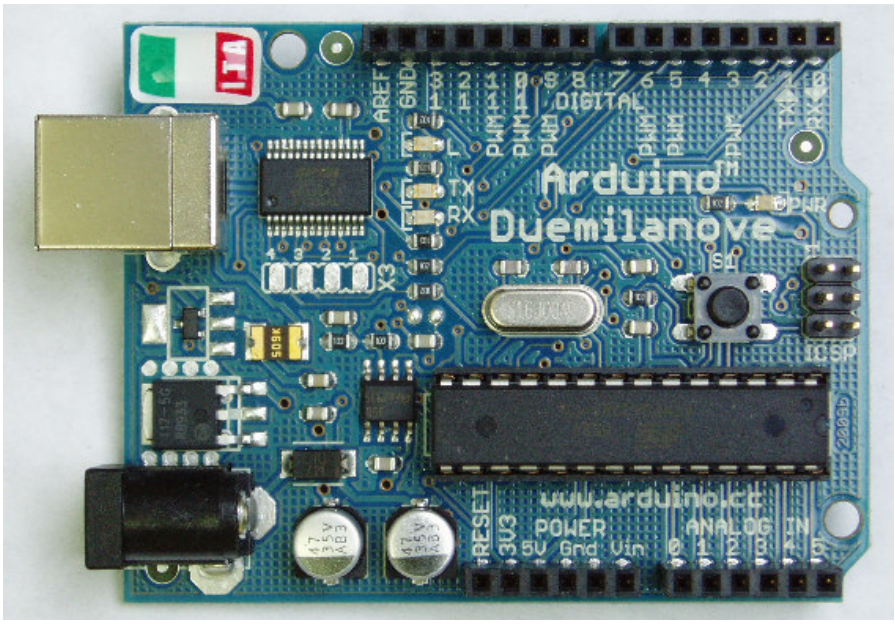
1) il trasduttore di distanza presenta il pin denominato SIG (signal) di tipo bidirezionale. Il trigger di durata T_1 è applicato dall'esterno e consente l'attivazione della capsula trasmettitore che genera onde ultrasonore dopo un tempo T_2. SIG si porta al livello logico alto (inizio dell'impulso T_3) e ritorna al livello logico basso dopo la prima ricezione dell'onda ultrasonora riflessa dall'ostacolo posto alla distanza D. Poiché il percorso effettivamente compiuto dall'onda è un'andata ed un ritorno, indicando con V la velocità dell'ultrasuono nel mezzo, il prodotto $V \cdot T_3$ fornisce il percorso totale dell'onda per cui la distanza dell'ostacolo è la metà e vale:

$$\mathbf{D = V \cdot T_3 / 2}$$

2) Il trasduttore di temperatura fornisce un codice binario in complemento a due per la rappresentazione dei numeri negativi. Il quanto è 0,5°C con andamento proporzionale. Infatti alla temperatura T_{emp} = 0°C il codice numerico di uscita, che in decimale indicheremo con "n", vale: n = 0. Se T_{emp}=0,5°C: n=1, a T_{emp}=25°C: n=50 (infatti $0\ 0011\ 0010_2 = 32 + 16 + 2 = 50$), e così via. Per temperature negative il bit più significativo vale 1 per cui occorre eseguire il complemento a 2 dei restanti 8 bit: complemento a 1 di ciascun bit scambiando gli 0 con gli 1 ed aggiunta di 1 al risultato ottenuto oppure, ed è la stessa cosa, basta svolgere la seguente sottrazione: codice decimale corrispondente degli ultimi 8 bit meno 256. Ad esempio, nel caso della temperatura -25°C, si dovrà ottenere un codice binario corrispondente a -50; infatti: $11001110_2 = 128+64+8+4+2 = 206$ quindi:
206 – 256 = -50.
Poiché la traccia chiede di risolvere il problema utilizzando un microprocessore o un microcontrollore, conviene orientarsi subito su una soluzione sbilanciata sul versante software. Il sistema da realizzare può essere risolto con un computer attraverso un opportuno circuito di interfaccia o con una scheda elettronica autonoma contenente un microprocessore o un microcontrollore. Nel caso dell'impiego di un PC si ha il vantaggio dell'implementabilità di numerosi linguaggi di programmazione sia a basso che ad alto livello e lo svantaggio di dover tenere il PC sempre acceso per la gestione del sistema. Nel caso di soluzioni autonome l'elaborazione dei dati è svolta dal microprocessore o microcontrollore residente sulla scheda elettronica che dovrà essere dotata di opportuni circuiti di interfaccia per l'I/O, di un sistema di alimentazione, di un sistema di comunicazione con un PC per la programmazione della ROM nel caso dei sistemi a microprocessore, e della ROM all'interno del PIC nel caso dei microcontrollori.

Recentemente si sono rese disponibili numerose tipologie di schede elettroniche autonome espandibili contenenti un microcontrollore e tutte le periferiche necessarie. Un microcontrollore contiene al suo interno un microprocessore, un'area di memoria RAM per i dati ed un'area ROM per i programmi. Contiene, inoltre,

convertitori A/D, contatori, interfacce, timer, ecc. Una di queste schede, ormai da anni famosa in tutto il mondo, è l'italianissima Arduino, disponibile in oltre 10 versioni. Le più famose sono "Arduino 2009" e "Arduino UNO" che montano il potente microcontrollore ATMega 328. La scheda è dotata di quarzo a 16MHz per il clock del microcontrollore, un'interfaccia USB, un regolatore di tensione integrato che consente di ottenere, dal sistema di alimentazione utilizzato, +5V e +3.3V necessari per l'hardware esterno purché non particolarmente esoso di corrente elettrica. E' disponibile gratuitamente un ambiente di sviluppo semplice ma nel contempo molto potente denominato Arduino 1.0 che utilizza un linguaggio di programmazione derivato dal C/C++. Si mostra in fig. 1 la scheda Arduino 2009.



La scheda presenta 14 linee digitali (strip superiore numerati da 0 a 13) ciascuna delle quali può essere programmata indipendentemente dalle altre come linea di ingresso o di uscita. Di queste linee, 6 possono essere pilotate in PWM per cui è possibile prelevare un segnale analogico tra 0 e 5V modificando a software il duty- cycle del PWM da 0 al 100%. La quota di PWM deve essere introdotta con un valore a 8 bit cioè con un valore compreso tra 0 (0%) e 255 (100%). Si mostrano in figura 2 tre forme d'onda con duty-cycle D%, rispettivamente, del 25%, 50% e 75%.

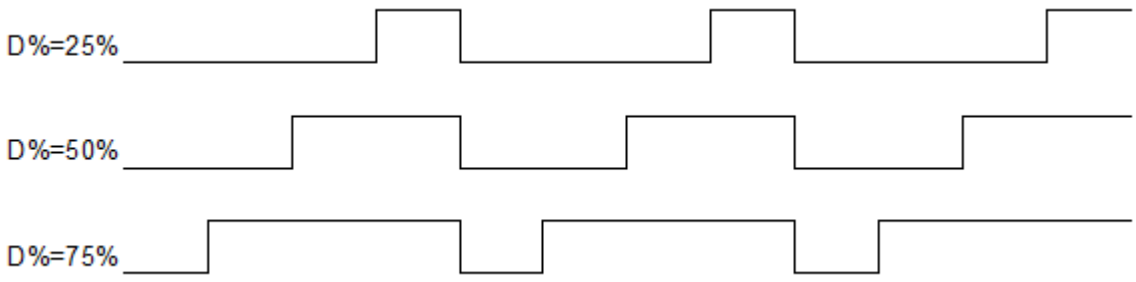


Fig. 2

Presenta ulteriori 6 piedini per l'input analogico (strip inferiore da 0 a 5) poiché sfrutta i 6 ADC interni al

microcontrollore ATMega 328. Il range di ingresso va da 0 a 5V ed il codice generato è a 10 bit quindi è possibile far acquisire alla scheda 6 valori tra 0 e 5V sotto forma di codice numerico compreso tra 0 e 1023.

Si programma da PC e si trasferisce il programma sulla scheda attraverso l'interfaccia USB. Successivamente si può scollegare la scheda dal PC purché la si alimenti con una sorgente di f.e.m. Il programma compilato in linguaggio macchina eseguibile per il microcontrollore risiede nell'area FLASH ROM del microcontrollore per cui se si stacca l'alimentazione il programma non si perde.

I vantaggi dell'utilizzo di tale scheda Arduino consiste nel basso costo della scheda (meno di 20 euro), nell'estrema diffusione di tale scheda nel mondo, dall'esistenza dell'interfaccia del micro sia verso il PC che verso l'hardware esterno, il software di programmazione gratuito, un'immensa documentazione disponibile sulla rete e nell'espandibilità attraverso altre schede (shield) che si innestano in due secondi sulla scheda base. Le schede di espansione consentono di essere operativi senza dover implementare altro hardware e software aggiuntivi. Come è logico prevedere, lo sviluppo del tema sarà svolto facendo riferimento alla scheda Arduino. Rispondiamo in modo puntuale alle richieste del problema.

a. Per l'acquisizione della durata di T_3 e del valore di temperatura T_{emp} si utilizzeranno 10 piedini di interfaccia digitale della scheda Arduino. In particolare il pin di Arduino da collegare al pin SIG del trasduttore di distanza dovrà essere definito di uscita per la generazione del trigger T_1 da inviare al trasduttore e successivamente di ingresso per la determinazione della durata dell'impulso T_3 proveniente dal trasduttore. (Fig.3).

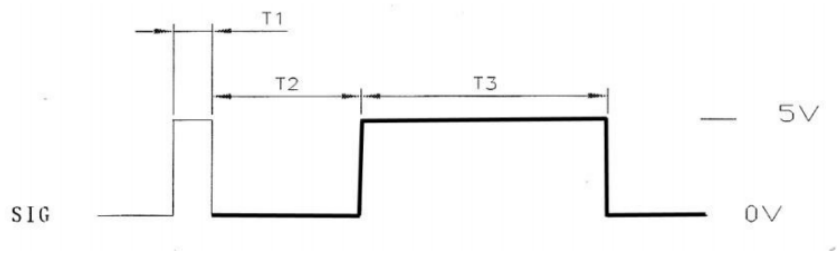


Fig. 3

I nove bit del trasduttore di temperatura saranno applicati ad altrettanti bit della scheda Arduino definiti di ingresso. Vedi lo schema a blocchi di fig.4.

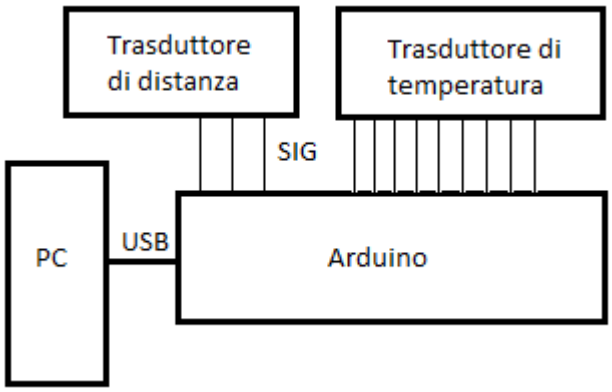


Fig. 4

b. Si decide di scegliere la modalità polling consistente nell'interrogazione ciclica. Il software sviluppato da Arduino, infatti, si basa su due principali routine: "setup" per le impostazioni iniziali di costanti e variabili e "loop" che conterrà le istruzioni di programma da eseguire. Dopo aver eseguito l'ultima istruzione, la routine "loop" viene ripetuta all'infinito.

Il diagramma di flusso di prima approssimazione prevede le seguenti operazioni.

1) trasmissione dell'impulso di durata T_1 sul pin definito di uscita di Arduino collegato a SIG

2) dopo la trasmissione il pin precedente sarà definito di ingresso e ci si pone in attesa del livello alto T_3. Per far questo si ripete continuamente la lettura di SIG finché questo è basso (intervallo T_2). Appena SIG diventa alto (inizio di T_3) si incrementa una variabile contatore della quantità di microsecondi pari a quanti si concedono di pausa prima di ripetere una successiva lettura di SIG. Quando SIG passa a 0 la procedura di interrogazione della linea SIG ha termine e la variabile contatore conterrà sicuramente la durata di T_3 espressa in microsecondi.

3) si passa alla lettura dei nove bit della temperatura e si applica l'algoritmo che consente di ottenere nella variabile "Temp" il valore numerico con segno della temperatura tenendo conto della rappresentazione dei numeri negativi in complemento a due.

4) Con le due formule assegnate si calcola la velocità dell'ultrasuono in funzione della temperatura e successivamente la distanza rilevata.

5) se si ipotizza il collegamento ad un PC, questa quinta fase potrà prevedere la stampa su monitor dei valori T_3 e T_{emp} acquisiti e della velocità V e della distanza D applicando le formule fornite dalla traccia.

6) Attivazione del ritardo tra una fase di acquisizione e la successiva che potrà essere di alcuni millisecondi o di durata superiore e successivo riavvio dell'esecuzione della routine "loop" saltando alla fase 1) di questo elenco.

c. Per rilevare il tempo T_3 si può impiegare il metodo descritto al punto b2. Più piccolo è il tempo di ritardo tra una acquisizione di SIG e la successiva più precisa è la determinazione dello stesso T_3. Poiché T_3 può assumere un valore compreso tra 0 e 19.5ms. in funzione della distanza dell'ostacolo, per ottenere un errore dello 0.1% rispetto al caso di massima distanza rilevabile, si deve ripetere la lettura di SIG ogni 19.5ms/1000=19.5us. Poniamo, ad esempio, ritardo = 19us. Ogni ciclo di interrogazione di SIG, in realtà, ha una durata pari a: durata=ritardo+tempo_istruzioni ove "tempo_istruzioni" è il tempo impiegato dal microcontrollore per svolgere il blocco di istruzioni consistenti in:

1) Lettura pin di Arduino collegato a SIG,
2) controllo se tale pin è HIGH,
3) in caso positivo aggiunge alla variabile contatore la quantità "ritardo"
4) salta al punto 1)

Ogni istruzione in linguaggio ad alto livello è in realtà costituita da più istruzioni. Supponendo che per i 4 passi siano necessari mediamente 4 istruzioni in linguaggio macchina la quantità tempo_istruzioni avrà valore 1µs poiché l'oscillatore al quarzo funziona a 16MHz ed ogni istruzione è svolta in un ciclo di clock. In tal caso si ha:

$$\text{durata} = 19+1 = 20\mu s$$

d. Per visualizzare le grandezze rilevate dai sensori si possono applicare due soluzioni: software ed hardware. Per la soluzione software, di immediata implementazione, si può impiegare il monitor del PC purché Arduino sia collegato al PC via USB. In tal caso per aprire un canale di comunicazione tra Arduino ed il PC durante l'esecuzione del programma, si deve inizializzare tale canale con l'istruzione Serial.begin(9600); dove 9600 è la velocità di trasferimento dati espressa in bit/secondo. Se i dati acquisiti dai due trasduttori sono T_3 e T_{emp}, si devono inserire alla fine del programma le istruzioni di stampa su monitor:

```
Serial.println(T3);
Serial.println(Temp);
```

La soluzione hardware è più complessa. Quella proposta parte dalla considerazione che i dati che vogliamo visualizzare siano quelli corrispondenti ai valori acquisiti da Arduino. In tal caso posso sfruttare due bit di uscita della scheda Arduino. Avendone utilizzate già 10 me ne rimangono altre 4 e quindi posso tranquillamente utilizzarne 2. Il problema fondamentale è che tali linee sono digitali. Allora dovrò scegliere due linee che consentono l'uscita in PWM. Queste sono: pin 11, 10, 9, 6, 5, 3. L'istruzione da implementare è: analogWrite(pin, valore) ove "pin" è uno dei 6 pin elencati in precedenza e "valore" è un numero compreso tra 0 e 255 che consente di ottenere in uscita un duty-cycle compreso tra 0 e 100%. Inserendo un filtro passa basso RC con costante di tempo sufficientemente più grande della frequenza di oscillazione rilevabile sul pin in PWM si potrà ottenere una tensione costante di valore proporzionale al duty-cycle a sua volta legato al valore di T_3. Per il valore della temperatura sarà necessario implementare, dopo il filtro, un offset che tenga conto della possibilità di avere una temperatura negativa. Ciascuna di queste due grandezze analogiche saranno inviate in un multimetro digitale che mostrerà il valore in volt che, attraverso un fattore di scala, mi permetterà di risalire finalmente al valore acquisito. In figura si mostra lo schema a blocchi per la visualizzazione di una delle due grandezze analogiche emesse in PWM dalla scheda Arduino. L'uscita del filtro passa basso è una tensione elettrica costante legato al duty-cycle % della grandezza emessa in PWM. Il convertitore A/D ed il display sono i principali elementi presenti in un multimetro digitale. In fig. 5 si mostra lo schema di principio di visualizzazione di una delle due grandezze su un display numerico a 3 cifre.

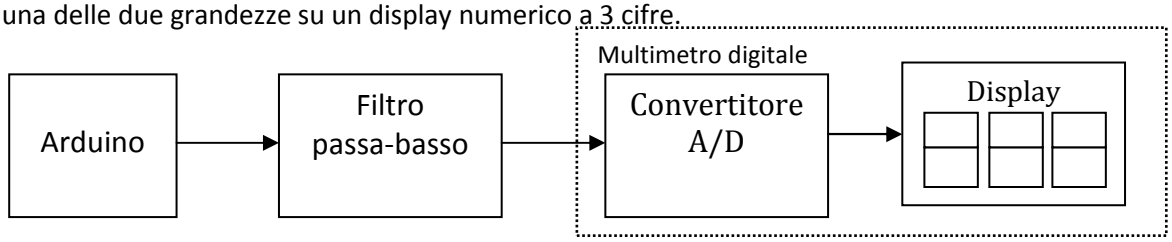


Fig. 5

e. Il programma inserito di seguito stampa sul monitor del PC il valore T_3 (in ms.) della durata del livello logico alto applicato su SIG_Pin (pin 12). Il valore di SIG_Pin è trasmesso su ledPin (pin 13).

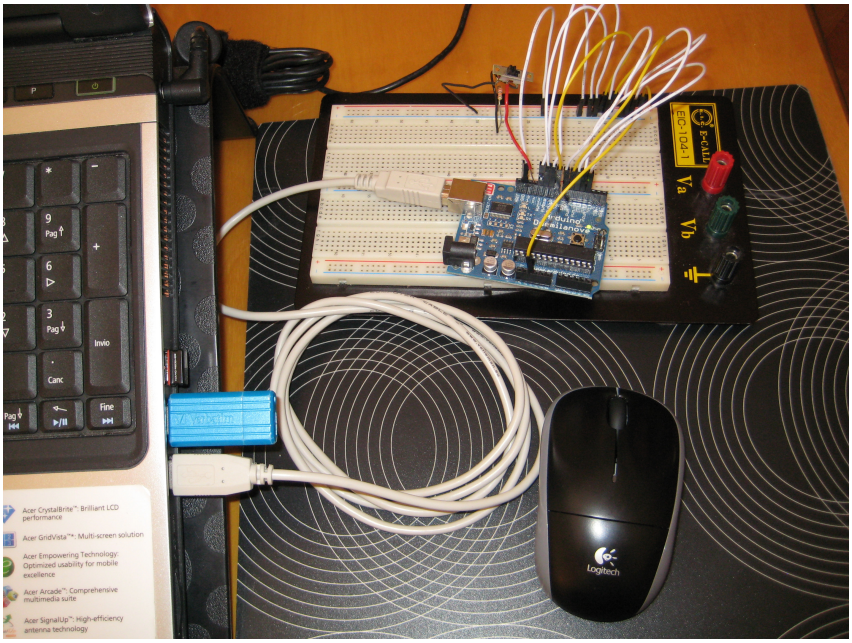
Viene prima generato un impulso T_1 di durata 10µs, dopo aver definito SIG_pin di uscita, e successivamente si definisce SIG_Pin di ingresso e lo si mette in ascolto di T_3 che esordisce sul pin SIG con una transizione dal livello logico basso al livello logico alto. Vedi la precedente fig. 3. Finché SIG_Pin=0 si ripete continuamente la lettura di tale linea ogni "durata" in µs. Quando SIG_Pin viene rilevato al livello alto (inizio di T_3) si incrementa la variabile T_3 della quantità "ritardo +1" (cioè "durata") e si ripete la lettura e l'incremento di T_3 fino a quando SIG_Pin=0 (termine impulso T_3).Successivamente si stampa sul monitor del PC il valore di T_3 che rappresenta il tempo in µs in cui SIG_Pin ha assunto il valore 1.

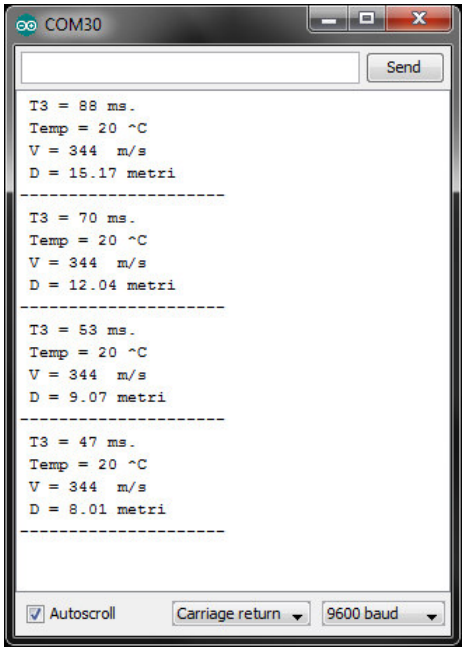
Nella seconda parte del programma si legge il valore del nono bit di temperatura applicato sul pin 3. Se vale 0 la temperatura è positiva, se vale 1 la temperatura è negativa ed è espressa in complemento a 2.

L'algoritmo applicato consente di esprimere sul monitor correttamente il valore n, in decimale con segno, relativo al codice acquisito.

```
int InputPins[] = { 3, 4, 5, 6, 7, 8, 9, 10, 11 }; // pin di Arduino cui applicare i 9 bit de trasduttore di temperatura
const int SIG_Pin=12; // pin di Arduino da collegare al terminale SIG del trasduttore di distanza
const int ledPin=13; //pin di controllo che pilota il LED presente sulla scheda Arduino
const int ritardo=19; // in microsecondi
float T3;

void setup(){
  pinMode(ledPin,OUTPUT); // ledPin definito di uscita
  for (int j = 0; j < 9; j++)  { pinMode(InputPins[j], INPUT); }
  Serial.begin(9600);
}

void loop(){
  T3 = 0;  // inizializzazione durata di T3
  pinMode(SIG_Pin,OUTPUT); // SIG_pin di Arduino definito di uscita per la generazione di T1
 digitalWrite(SIG_Pin, LOW); // SIG_Pin viene forzato al livello logico basso prima di T1
 delayMicroseconds(10); // SIG_Pin tenuto basso per 10µs
 digitalWrite(SIG_Pin, HIGH); // SIG_Pin viene forzato al livello logico alto  (inizio di T1)
 delayMicroseconds(10); // SIG_Pin tenuto alto per 10µs (T1>5µs)
 digitalWrite(SIG_Pin, LOW); // SIG_Pin viene forzato al livello logico basso  (termine di T1)
 pinMode(SIG_Pin,INPUT); // SIG_pin di Arduino definito di ingresso per la lettura della durata di T3
while(digitalRead(SIG_Pin)==LOW) {
    digitalWrite(ledPin,digitalRead(SIG_Pin)); //  ledPin spento
    delayMicroseconds(ritardo); // ritardo in µs
    //delay(ritardo); // ritardo in ms
 }
  while(digitalRead(SIG_Pin)==HIGH) {
    digitalWrite(ledPin,digitalRead(SIG_Pin)); // ledPin acceso
    T3 = T3+ritardo+1; // aggiornamento durata di T3
    delayMicroseconds(ritardo); // ritardo in µs
  }
  Serial.print(" T3 = "); // stampa sul monitor del PC la stringa tra doppie virgolette
  T3=T3/1000; // conversione di T3 da µs in ms.
  Serial.print(round(T3)); // stampa T3 in ms. approssimando
  Serial.println(" ms."); // stampa sul monitor del PC la stringa tra doppie virgolette
  float n = 0; // inizializza la variabile che conterrà la conversione da bin. a dec. della temperatura
  int segno = digitalRead(InputPins[0]); // lettura del bit di segno
  for(int i = 1; i < 9; i++)  {
     int m=digitalRead(InputPins[i]); // lettura del bit i-esimo
     n = n + m*pow(2,(8-i)); // aggiunge a “n” il contributo del bit in posizione 8-i
  }
  if(segno != 0)  n = n-256;
// se la temperature è negativa si calcola il complemento a 2 e si cambia il segno
 float Temp = 0.5*n; // la risoluzione numerica corrisponde a 0.5°C
 Serial.print(" Temp = ");
 Serial.print(round(Temp)); //Stampa sul monitor il valore della temperatura
 Serial.println(" ^C");
// valutazione e stampa della velocità V e della distanza D
  float V = 331.5 + 0.62*Temp; // velocità in m/s
  float D = V*T3/2000; // distanza in metri
  Serial.print(" V = "); // stampa sul monitor del PC la stringa tra doppie virgolette
  Serial.print(round(V)); // stampa V in m/s approssimando
  Serial.println("  m/s"); // stampa sul monitor del PC la stringa tra doppie virgolette
  Serial.print(" D = "); // stampa sul monitor del PC la stringa tra doppie virgolette
  Serial.print(D); // stampa D in m approssimando
  Serial.println(" metri"); // stampa sul monitor del PC la stringa tra doppie virgolette
  Serial.println("----------------------");
}
```

Si riporta in fig.6 l'immagine della sperimentazione realizzata. I cavetti gialli sono collegati a 5V e quei bianchi a massa. Il deviatore consente di simulare l'impulso T_3. Si riporta, infine, la stampa su monitor del PC di 4 tipici casi in cui si è supposta la temperatura Temp=20°C fornendo, all'ingresso di Arduino, 9 bit per la temperatura con una configurazione pari a n=40 (0 0010 1000 i due bit posti a 1 sono quelli col cavetto giallo).
N.B. non avendo il trasduttore di distanza si è generato manualmente l'impulso T_3 con un deviatore non riuscendo ad ottenere tempi inferiori a 47 ms.





f. La traccia fornisce il valore della velocità del suono in funzione della temperatura:

$V = a + b \cdot Temp$ ove: $a = 331.5$ m/s; $b = 0.62$ m/s/°C
$V = 331.5 + 0.62 \cdot 20 = 331.5 + 12.4 = 343.9$ m/s
La distanza massima rilevabile è quella che si ottiene in corrispondenza di $T_3 = 19.5$ ms, quindi:
$D = V \cdot T_3/2 = 343.9 \cdot 19.5 \cdot 10^{-3}/2 = 3.353$ m
$\Delta V = b \cdot \Delta Temp = 0.62 \cdot (40-20) = 12.4$ m/s
$\Delta D = \Delta V \cdot T_3/2 = 12.4 \cdot 19.5 \cdot 10^{-3}/2 = 0.121$ m = 12.1 cm