Breve descrizione dell'architettura hardware

La scheda a bordo missile (SBM) ha il compito di gestire tutte quelle azioni previste sul razzo prima e dopo il suo decollo, esse si riassumono in tre operazioni fondamentali:

- Commutazione delle telecamere al distacco delle due sezioni: normalmente è la telecamera laterale ad essere connessa al trasmettitore, ma al momento del distacco delle due sezioni del razzo viene abilitata una telecamera posta sul raccordo fra le due sezioni; in questo modo è possibile acquisire quei quattro secondi circa durante i quali le due sezioni si separano ed i paracadute si aprono. Passati i 4 secondi, viene riconnessa al trasmettitore la telecamera laterale, in modo da avere una ripresa panoramica della discesa.
  Questa commutazione viene inoltre effettuata dal comando esterno proveniente dalla SBR, quando si invia una richiesta di check da PC.
- Commutazione dell'alimentazione 12V: quando il razzo è installato sulla rampa, i 12V necessari alla sua alimentazione sono prelevati dalla batteria di strumentazione presente sulla rampa stessa, questo ci consente di risparmiare le batterie interne al razzo facendole intervenire soltanto alla partenza. Quando il razzo si stacca dalla rampa e comincia la sua ascesa, l'alimentazione passa alle batterie interne.
  Anche questa commutazione viene testata dalla SBR su richiesta del PC.
- Acquisizione della accelerazione: è presente un accelerometro integrato che trasduce l'accelerazione assiale in una tensione, questa viene convertita in digitale e memorizzata in una memoria interna al microcontrollore. Alla partenza del razzo, il forte impulso di accelerazione iniziale scatena la memorizzazione dei 512 campioni con un periodo tale da coprire circa 16 secondi di volo, più che sufficenti al momento per ottenere informazioni importanti. Successivamente possiamo prelevare i campioni dalla memoria e scaricarli su un PC per la loro elaborazione.

Scheda a bordo missile SBM

Cominciamo col descrivere la parte più semplice, ossia il commutatore di alimentazione. Si è preferita una soluzione semplicissima che si svincolasse dal programma su microcontrollore, in modo da avere un sistema molto affidabile. Un solo relè si occupa di commutare sui 12V esterni o interni a seconda che l'alimentazione esterna sia presente o meno, poiché è questa stessa ad alimentare la sua bobina:

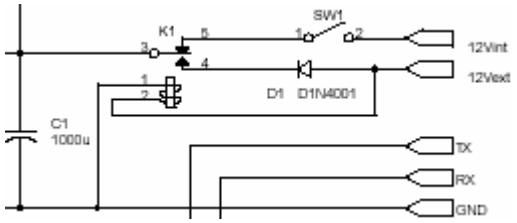

Figura 1: Circuito per la commutazione delle alimentazioni

L'interruttore SW1 è montato sullo sportello della sezione strumentale del razzo e ci permette di collegare l'alimentazione interna al circuito di commutazione, altrimenti alimenterebbe la circuiteria interna durante il trasporto del razzo fino sulla rampa. Il diodo invece permette al commutatore di spostarsi il prima possibile sulla alimentazione interna quando quella esterna non è più presente, l'assenza di questo semplice componente implica un certo tempo di ritardo dovuto alla scarica del condensatore da 1000μF usato per livellare la tensione durante la commutazione e così verrebbe meno la sua funzione di livellamento, disturbando anche la trasmissione.

Il contatto centrale del relè va alla linea di 12V utilizzata per il relè di commutazione telecamere e alla alimentazione del trasmettitore. Dai 12V ricaviamo i 5V per la logica con uno stabilizzatore integrato 7805.

Passiamo al circuito che segnala il distacco delle sezioni, abbiamo utilizzato un fotoaccopppiatore alloggiato in un contenitore ad U, nel quale si inserisce una lamella incollata sulla sezione motori e che interrompe il collegamento ottico quando le due sezioni sono unite. Al distacco, la lamella viene portata via dalla sezione motori ed il collegamento ottico si ripristina portando in conduzione il fototransistore, per cui avremo uno zero logico sul suo collettore. Il segnale 1/0 del fototransistore viene simulato esternamente mediante un BJT BC337 collegato in parallelo al primo ed il segnale che porta in conduzione il BJT proviene dalla SBR su richiesta del PC (fase di check).

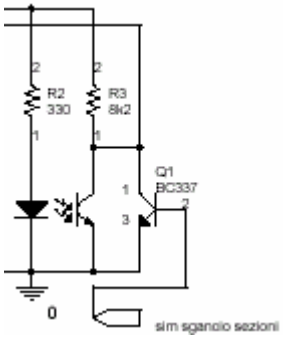

Figura 2: Circuito di segnalazione del distacco sezioni

Il distacco sezioni provoca la commutazione del segnale video sulla telecamera di fondo per 4 secondi, ciò avviene mediante un relè che smista le uscite video delle due telecamere all'ingresso

del trasmettitore ed inoltre provvede alla alimentazione delle telecamere stesse quando è richiesta la trasmissione del loro video.

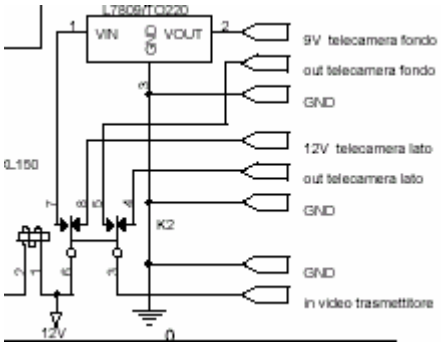

Figura 3: Commutazione delle telecamere

Passando alla parte di acquisizione della accelerazione, intanto impostiamo il programma in scrittura quando deve registrare i dati di volo ed in lettura quando , giunti a terra, si deve scaricare la memoria sul PC. Le due impostazioni sono eseguite con un ponticello su scheda:

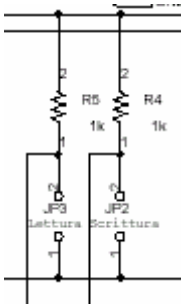

Figura 4: Ponticelli per impostare il programma

Si passa poi all'accelerometro, ossia un sensore di accelerazione integrato ADXL150 che sfrutta la tecnologia di microfabbricazione su silicio assieme all'elettronica di condizionamento (MEMS) per convertire un'accelerazione in una tensione. Il principio fondamentale è una variazione di capacità causata dallo spostamento di un'armatura mobile del condensatore integrato quando è sottoposta ad una accelerazione. La tensione in uscita al sensore di accelerazione viene filtrata LP con una squadra RC avente una frequenza di taglio di 600Hz e poi inviata al convertitore AD integrato nel microcontrollore:

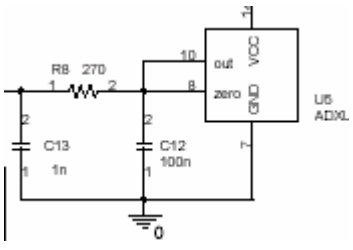

Figura 5: Accelerometro

Con questo semplice sistema si è calcolata (risparmiamo i calcoli in questo manuale) una misura digitale di accelerazione con rumore efficace totale pari a 78,43 mg, per cui possiamo avere una registrazione sufficientemente accurata.
Il sensore di accelerazione collegato come in figura ci consente di trasdurre l'accelerazione assiale con la legge seguente:

$$V_{out} = V_{CC}\left(0.5 + 0.2 \cdot S_{v\_a} \cdot a\right)$$

dove S_{v_a} è pari a 76 mV/g, dunque abbiamo una dipendenza diretta dalla tensione di alimentazione; questo inconveniente scompare con la quantizzazione della Vout, infatti il convertitore AD prende come riferimento proprio la V_{CC} (ingresso AVCC del microcontrollore) ed il valore convertito in digitale è:

$$C = \frac{V_{CC}\left(0.5 + 0.2 \cdot S_{v\_a} \cdot a\right)}{V_{CC} / 256} = 128 + 3.891 \cdot a$$

dove a è espressa in g.
Prendendo una VCC di 5V, possiamo calcolare la differenza minima registrabile dall'ADC, ossia il quanto di tensione:

$$q_v = \frac{5}{256} = 19.2\text{mV}$$

questo valore in tensione corrisponderà ad una differenza minima di accelerazione, ossia il quanto di accelerazione che è una misura della precisione del campionamento:

$$q_a = \frac{q_v}{S_{v\_a}} = 0.25\text{g}$$

I dati registrati nella memoria EEPROM del microcontrollore possono essere scaricati collegando la seriale RS232 del PC alla scheda, il circuito di conversione di livelli MAX232 si occupa di fornire le tensioni adeguate per questa connessione:

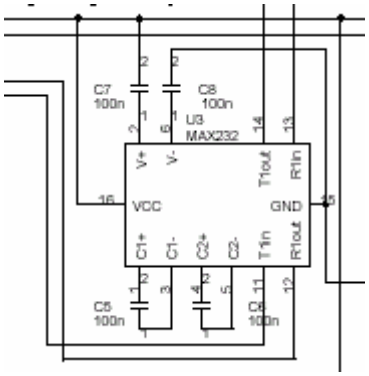

Figura 6: Convertitore di livelli per seriale RS232

Descrizione del software

Tutta la logica che gestisce le operazioni già elencate precedentemente è implementata nel programma ASSEMBLER AVR caricato nel microcontrollore Atmega8, in questa descrizione non ci dilungheremo troppo nei dettagli già descritti nel manuale della SBR ma concentreremo l'attenzione nella logica del programma di gestione.

Abbiamo un iniziale elenco di impostazioni riguardanti l'interfaccia seriale UART (per la comunicazione RS232 col PC) ed il convertitore AD:

```
;IMPOSTAZIONI DELLA SERIALE RS232
; costanti per la comunicazione seriale asincrona
; velocità a 9600 bit/sec,abilitazione a trasmettere e ricevere

.EQU h_baud_rate=0
.EQU l_baud_rate=28
.EQU UART_en=0x18
.EQU impostazioni=0x86

; IMPOSTAZIONI DELL'ADC
; configurazione di ADMUX
; riferimento su AVCC 5V,allineamento dato a sinistra (otto bit),
; canale PC0 all'ADC

.EQU CH0=0x60

;configurazione di ADSCR
;clock del convertitore 115KHz con un fattore di divisione 32,
;interruzione disabilitata,abilitazione al convertitore e start

.EQU AD_en=0xC5
```

```
; impostazione della velocità della seriale

LDI R16,h_baud_rate
OUT UBRRH,R16
LDI R16,l_baud_rate
OUT UBRRL,R16

; abilitazione seriale RS232 a trasmettere e ricevere

LDI R16,UART_en
OUT UCSRB,R16
LDI R16,impostazioni
OUT UCSRC,R16

; svuotamento buffer seriale UART

IN R19,UDR

; impostazioni del convertitore

LDI R16,CH0
OUT ADMUX,R16
```

A questo punto definiamo gli ingressi e le uscite

```
; definizione delle porte di IO,tutte uscite tranne:
; PC0 ingresso analogico sensore di accelerazione
; PC3 uscita LED di segnalazione stato
; PD2 ingresso da fototransistore distacco sezioni
; PD6 ingresso selezione download accelerazione campionata
; PD7 ingresso selezione acquisizione su scheda

SER R16
OUT DDRB,R16
LDI R16,0x3B
OUT DDRD,R16
LDI R16,0xFE
OUT DDRC,R16
```

Definiamo adesso un registro nel quale caricare la soglia ed un altro che funge da flag per far partire la registrazione. Nel registro di soglia carichiamo poi il valore 2, in questo modo faremo partire la registrazione quando si osserva un impulso di accelerazione superiore al valore precedente di circa 0.5g:

```
; definizione della soglia di tensione per l'inizio campionamento
; e del registro di start al ciclo di campionamento

.DEF soglia=R14
.DEF start_AD=R19

; ogni LSB corrisponde a circa 0.25g di accelerazione, soglia impostata a circa
g/2

LDI R16,2
MOV soglia,R16
```

Si parte adesso col programma principale; una segnalazione importante da parte del LED di stato (montato sullo sportello della sezione strumentale) consiste nell'avvio programma a LED spento, nell'avvio ciclo di campionamento a LED acceso, e nell'avvio registrazione a LED spento. Quando daremo alimentazione alla sezione, dovremo osservare con un breve ritardo l'accensione del LED che tornerà spento a terra se l'acquisizione è andata a buon fine.

```
; PROGRAMMA PRINCIPALE

; spegne il LED di programma in attesa

CBI PORTC, PC3

; attende 0.33 sec per stabilizzare le tensioni di alimentazione e
; di segnale del sensore accelerometro

CALL delay_33ms
CALL delay_33ms
CALL delay_33ms
CALL delay_33ms
CALL delay_33ms
CALL delay_33ms
CALL delay_33ms
```

```
CALL delay_33ms
CALL delay_33ms
CALL delay_33ms
```

Si effettua adesso un primo campionamento dell'accelerazione, in questo modo possiamo prepararci le soglie di innesco superiore ed inferiore sommando e sottraendo al registro di campionamento R16 la soglia in R14 e mettendo il risultato in un registro di soglia superiore R15 e di soglia inferiore R13. Abbiamo così creato una fascia di 0.5g attorno al valore attuale di accelerazione che, se superata, farà partire la registrazione. Se invece il valore che campioneremo successivamente non supera tale fascia aggiorneremo le nuove soglie con tale valore di accelerazione e così inseguiamo le eventuali lente derive termiche o della VCC senza rischiare una falsa partenza della registrazione.

```
; effettua un primo campionamento dell'accelerazione e prepara il valore di soglia

CALL AD_converti
MOV R15,R16
MOV R13,R16
ADD R15,soglia
SBC R13,soglia
```

Prepariamo adesso l'indirizzo iniziale della memoria EEPROM del microcntrollore nella quale andremo a scrivere e resettiamo il registro di flag in modo che non indichi l'avvio alla registrazione:

```
; prepara gli indirizzi iniziali per la eeprom

CLR R24
CLR R25

; prepara il registro di segnalazione superamento soglia,
; col quale diamo il via alla scrittura dei campioni

CLR start_AD
```

Mediante i jumper di selezione su scheda, decidiamo se far girare il programma di scrittura (durante il lancio) o di lettura (a lancio avvenuto, per scaricare i campioni su PC):

```
; controlla i jumper di selezione

start:
SBIS PIND,PD6
JMP leggi
SBIS PIND,PD7
JMP scrivi
JMP start
```

Il programma di scrittura comincia accendendo il LED di stato e memorzzando il valore precedentemente campionato in un registro di appoggio R17:

```
; accende il LED, di attesa acquisizione
SBI PORTC,PC3
; prepara il valore precedentemente campionato
MOV R17,R16
```

Il registro R16 viene poi sovrascritto con il nuovo valore di accelerazione

```
; conversione di un nuovo valore
CALL AD_converti
```

Controlliamo a questo punto se abbiamo superato la fascia di soglia, la comparazione verrà eseguita dopo ma è a questo punto che decidiamo il da farsi andando a controllare il contenuto del registro di flag: se 1 significa che la fascia è stata superata e possiamo registrare i campioni senza più preoccuparci di controllare il superamento della fascia stessa (salto a sample).
Se abbiamo superato la fascia invece saltiamo a rec e scriviamo 1 nel registro di flag, questo segnalerà al programma che dobbiamo cominciare la registrazione, come visto:

```
; se è stata raggiunta la soglia, passa alla scrittura
CPI start_AD,1
BREQ sample
; se è stata raggiunta la soglia superiore, segnalalo in start_AD
CP R16,R15
BRGE rec
; se è stata raggiunta la soglia inferiore, segnalalo in start_AD
CP R16,R13
BRLT rec
```

```
      ; negli altri casi continua a campionare senza memorizzare
      JMP salta
      ; flag di soglie superate
rec:  LDI start_AD,1
```

Le istruzioni sotto l'etichetta sample effettuano la registrazione del campione precedentemente acquisito e memorizzato in R17 nella memoria EEPROM, in questo modo possiamo anche registrare il campione precedente al superamento della fascia, che fungerà da livello zero per il grafico dell'accelerazione.

```
        ; scrive in memoria il valore campionato precedentemente
        ; in questo modo memorizza anche il valore all'istante zero
sample: CALL e2prom_scrivi
        ADIW R25:R24,1
        ; spegne il LED, acquisizione in corso
        CBI PORTC,PC3
```

Le operazioni di scrittura nella EEPROM sono dunque un aggiunta al normale ciclo di programma, il quale prevede in tutti i casi le istruzioni che elenchiamo di seguito:
Creiamo una nuova fascia di soglia col campione acquisito in R16

```
salta:  ; prepara una nuova soglia superiore ed inferiore
        MOV R15,R16
        MOV R13,R16
        ADD R15,soglia
        SBC R13,soglia
```

Attendiamo 33 ms, che sarà il nostro periodo di campionamento; questo valore è stato calcolato in maniera tale da registrare circa 17 secondi di volo nella EEPROM da 512 byte. Il pregio del programma è quello di non variare significativamente il valore del periodo di campionamento anche quando si avvia la commutazione telecamere ed in questo modo non disturbiamo le tempistiche dell'acquisizione.

```
; attesa per il campionamento
CALL delay_33ms
```

Controllando lo stato del pin PD2, possiamo segnalare l'avvenuto distacco delle sezioni scrivendo 1 nel registro di flag R21 dedicato a questa fase; come vedremo, questo registro non è un semplice flag come quello per la registrazione dei campioni, ma serve anche ad avviare il conteggio del timer

```
; controlla se è avvenuto il distacco sezioni, segnalandolo in R21
SBIS PIND,PD2
LDI R21,1
```

La gestione delle telecamere impegna un certo numero di registri per la sua segnalazione e per il timer da 4 secondi, questi registri sono resettati dalla routine clear_reg se non è avvenuto il distacco:

```
; se non arriva il segnale di distacco attendi
SBIC PIND,PD2
CALL clear_reg
```

Il registro R20 conta quanti passaggi con R21 = 1 sono avvenuti, ogni passaggio dura circa 33 ms e quindi quando R20 = 128 sono passati circa 4 secondi. Questo semplice calcolo ci consente di far azzerare R21, riportando la trasmissione video sulla telecamera laterale ed arrestando il conteggio, quando il bit 7 di R20 è alto:

```
; a fine conteggio riporta sulla telecamera laterale e blocca il conteggio
SBRC R20,7
CLR R21
```

Il conteggio delle passate con R21 alto si effettua con la semplice somma R20 + R21; questo sistema permette quindi di avviare il timer al distacco sezioni osservando il segnale del fotoaccoppiatore e sarà poi il timer stesso a prendere il comando ed a terminare le operazioni riportando tutti i registri al valore di riposo

```
; conteggio temporale se è avvenuto il distacco
ADD R20,R21
```

Segue la fase operativa sulle telecamere, ossia si commuta il relè solo se R21 = 1:

```
; commuta sulla telecamera di fondo mentre effettua il conteggio
SBRC R21,0
```

```
SBI PORTD,PD5
; in tutti gli altri casi lascia la telecamera laterale
SBRS R21,0
CBI PORTD,PD5
```

Se stiamo registrando nella memoria, gli indirizzi sono incrementati ad ogni fase di scrittura e quando R25 = 1 la memoria è piena, per cui a questo punto si salta la fase di registrazione e si esegue solo il gruppo di istruzioni per la gestione delle telecamere, altrimenti torna all'inizio del programma per campionare un successivo valore di accelerazione.

```
; a fine memorizzazione non campionare più l'ingresso ma continua
; ad eseguire il ciclo di controllo distacco sezioni
SBRC R25,1
JMP salta
JMP scrivi
```

Il programma ha una struttura ovviamente ciclica e le varie operazioni sono eseguite o meno a seconda delle segnalazioni sui flag, non abbiamo comunque salti all'indietro se non i due obbligatori alla fine. Il programma allora si può leggere facilmente e le istruzioni aggiunte o tolte non modificano molto il periodo di campionamento come detto, questo parametro è molto importante perché con esso poi calcoleremo la velocità e la posizione del razzo, è quindi necessario che non sia modificato troppo dal software.
La fase di lettura è molto semplice, il programma non fa altro che attendere il comando di invio dato dal PC, poi va a leggere la memoria EEPROM ed il campione letto lo spedisce alla seriale, fino all'esaurimento della memoria stessa.

```
leggi: ; prepara gli indirizzi iniziali
       CLR R24
       CLR R25
       ; trasferimento al PC con handshake
read:  CALL PC_ricevi
       CALL e2prom_leggi
       CALL PC_invia
       ADIW R25:R24,1
       ; a fine lettura entra in attesa
       SBRS R25,1
       JMP read
fine:  JMP fine
```

Passando alle routine, le novità rispetto al software della SBR sono e2prom_leggi ed e2prom_scrivi, utilizzate per le operazioni sulla EEPROM del microcontrollore. La scrittura è molto semplice, si controlla in EECR se il bit EERE è basso, al che la memoria è pronta alla scrittura, dopodichè si scrive l'indirizzo in EEARH ed EEARL. Il dato da scrivere viene caricato in EEDR ed infine si abilita la scrittura settando i bit EEMWE ed EEWE del registro EECR.

```
; routine per la scrittura nella memoria

e2prom_scrivi: SBIC EECR,EERE
               JMP e2prom_scrivi
               OUT EEARH,R25
               OUT EEARL,R24
               OUT EEDR,R17
               SBI EECR,EEMWE
               SBI EECR,EEWE
               RET
```

La lettura dalla memoria è analoga, si setta il bit di lettura EERE in EECR e si carica il registro del dato EEDR, la gestione dei bit EERE, EEWE ed EEMWE è avviata dal software e chiusa dall'hardware per cui non dobbiamo preoccuparci di resettare questi bit di abilitazione

```
; routine per la lettura dalla memoria

e2prom_leggi: SBIC EECR,EEWE
              JMP e2prom_leggi
              OUT EEARH,R25
              OUT EEARL,R24
              SBI EECR,EERE
              IN R16,EEDR
              RET
```