Il candidato progetti il software per l'applicazione qui specificata, e ne implementi una parte significativa. Il progetto sia scritto in linguaggio naturale integrato da diagrammi UML ed eventualmente altre notazioni che si rendano necessarie.

1 Descrizione del sistema

Il Digital Sound Recorder è un dispositivo, controllato da un microprocessore, per la registrazione e la riproduzione di messaggi audio in formato digitale. È dotato di uno schermo LCD, di un microfono, di un altoparlante, e di una tastiera con due tasti di navigazione (*Up* e *Down*) e quattro tasti funzione (*Play*, *Record*, *Stop*, *Delete*).

I messaggi registrati vengono immagazzinati in una memoria non-volatile.

1.1 Hardware

1.1.1 Schermo

Lo schermo, in bianco e nero, è formato da 10 righe di 15 caratteri.

1.1.2 Microfono

L'interfaccia del microfono verso il microprocessore è costituita da un registro di 8 bit. Ogni operazione di lettura sul registro causa il campionamento del segnale analogico, la sua memorizzazione nel registro e la sua lettura da parte del microprocessore.

Ogni lettura causa l'aggiornamento del registro.

1.1.3 Altoparlante

L'interfaccia dell'altoparlante verso il microprocessore è costituita da un registro di 8 bit. Ogni operazione di scrittura sul registro causa la generazione del segnale analogico corrispondente.

Il segnale analogico mantiene il valore impostato dall'ultima operazione di scrittura, fino all'operazione successiva.

Il registro è inizializzato automaticamente a zero.

1.1.4 Tastiera

La tastiera assegna un codice a ciascun tasto: *Up* (1), *Down* (2), *Play* (3), *Record* (4), *Stop* (5), *Delete* (6).

Quando viene premuto un tasto, il codice corrispondente viene inserito in una coda.

1.1.5 Memorie

I messaggi vengono memorizzati in una memoria di massa a stato solido di 20 MB. Si suppone che questo spazio di memoria sia interamente disponibile per i messaggi (si trascurino quindi gli overhead per la gestione del filesystem).

Si suppone che la memoria RAM per il sistema operativo e il codice dell'applicazione abbia una dimensione sufficiente.

1.1.6 Timer

Il dispositivo ha un timer/contatore che genera un certo numero di interruzioni, a intervalli prefissati.

1.2 Formato del segnale audio

Il segnale audio viene memorizzato nella memoria dell'apparecchio in questo formato:

1. ogni messaggio consiste in una intestazione lunga 4 byte, seguita dalle codifiche dei campioni audio;
2. i primi 21 bit dell'intestazione contengono il numero di campioni formanti il messaggio;
3. ciascun campione del segnale audio è codificato su 8 bit.

La frequenza di campionamento del segnale audio è pari a 6 KHz.

1.3 Software di sistema

Si suppone che il dispositivo sia controllato da un sistema operativo monoprogrammato accessibile attraverso l'interfaccia di programmazione descritta di seguito. A ciascuna sottosezione corrisponde una classe in C++ o in Java.

Gli ambienti di sviluppo (v. oltre) permettono di integrare il codice applicativo al sistema operativo.

All'accensione del dispositivo il controllo passa all'applicazione.

1.3.1 Display

print(in row:int, in text: string)
Scrive il testo text alla riga numero row, cancellando il messaggio eventualmente già presente. Se il testo supera la lunghezza della riga, viene troncato. Il numero di riga va da 0 a 9; se l'argomento non rientra in questo intervallo, l'operazione non ha effetto.

highlight(in row:int)
Evidenzia il testo della riga row.

1.3.2 Microphone

get(): unsigned char
Preleva il valore attuale del segnale analogico e restituisce il valore della sua conversione digitale su 8 bit.

1.3.3 Speaker

put(in val: unsigned char)
Scrive nel registro dell'altoparlante il valore digitale del segnale, che viene convertito in analogico.

1.3.4 Keyboard

get(): int
Estrae dalla coda il primo codice e lo restituisce al chiamante. Restituisce zero se la coda è vuota.

1.3.5 Filesystem

create(): string
Crea un messaggio vuoto sulla memoria di massa e restituisce una stringa di 15 caratteri stampabili che lo identifica.

write(in id: string, in size: int, in msg: string): int
Scrive la stringa di byte (non necessariamente caratteri di testo) msg, lunga size byte, nel messaggio vuoto identificato dalla stringa id.

Il comportamento dell'operazione nel caso che il messaggio non sia vuoto, o che la dimensione effettiva di msg sia diversa da size, non è definito.

Restituisce 0 se eseguita correttamente, 1 se nessun messaggio corrisponde all'identificatore, 2 se lo spazio nella memoria di massa non è sufficiente.

read(in id: string, in size: int, out msg: string): int
Legge il messaggio identificato da id e lo copia in msg, allocando la memoria necessaria.

Restituisce la dimensione del messaggio se eseguita correttamente, 0 se nessun messaggio corrisponde all'identificatore.

remove(in id: string): int
Cancella il messaggio identificato da id, rendendo disponibile la memoria occupata.

Restituisce 0 se eseguita correttamente, 1 se nessun messaggio corrisponde all'identificatore.

list(out n: int): string[]
Restituisce gli identificatori dei messaggi memorizzati, come un array di string, e scrive nel parametro di uscita n il numero di messaggi memorizzati. Se nessun messaggio è memorizzato, n vale 0 ed il valore restituito è indefinito.

1.3.6 Timer

set(in n: unsigned, in period: unsigned)
Inizializza il timer impostando il numero n di interruzioni che devono essere generate, a intervalli di period microsecondi.

Dopo l'ultima interruzione, il timer si arresta e torna allo stato non inizializzato.

start()
Avvia il timer. Non ha effetto se il timer non è inizializzato, o è già stato avviato.

stop()
Arresta il timer, riportandolo allo stato non inizializzato. Non ha effetto se il timer non è inizializzato, o non è stato ancora avviato.

{abstract} on_tick()
Questa operazione deve essere implementata nel codice applicativo (in una classe derivata), e viene invocata ad ogni scatto del timer.

1.4 Ambienti di sviluppo

L'interfaccia di programmazione sú esposta è disponibile sotto forma di librerie in C++ e in Java. In queste librerie è definito un package (o namespace) **uOS** (*micro Operating System*) contenente una classe per ciascuna sottosezione della sezione precedente (**Display**, **Keyboard**...).

Il tipo string citato nella sezione precedente viene implementato com `const char` nella libreria C++ e con `String` nella libreria Java.

Gli ambienti di programmazione (C++ e Java) girano su un PC e permettono di caricare il software completo (applicazione e libreria di sistema) sul dispositivo.

2 Requisiti

2.1 Accesso diretto ai messaggi

Il sistema deve fornire all'utente un metodo di accesso ai messaggi, presentando su schermo l'elenco dei messaggi (ogni messaggio è identificato da una sequenza di caratteri generata automaticamente) e permettendo la selezione dei messaggi con i tasti di navigazione.

2.2 Registrazione di un messaggio

L'utente inizia la registrazione premendo pulsante *Record*. Il sistema procede con la registrazione del segnale audio attraverso il microfono finché non si verifica una delle seguenti condizioni: (a) l'utente preme il pulsante *Stop*; (b) si raggiunge la la capacità massima del messaggio; oppure (c) si esaurisce la memoria disponibile.

2.3 Riproduzione di un messaggio

L'utente seleziona un messaggio dall'elenco e preme il pulsante *Play*. Il messaggio viene riprodotto attraverso l'altoparlante fino al termine del messaggio o finché l'utente preme il pulsante *Stop*.

2.4 Cancellazione di un messaggio

L'utente seleziona un messaggio dall'elenco e preme il pulsante *Delete*. Il messaggio viene cancellato in maniera permanente dalla memoria del sistema.

3 Requisiti temporali

Si ignorino i requisiti temporali, supponendo che il microprocessore sia abbastanza veloce da eseguire tutte le operazioni richieste nell'intervallo fra un campione audio e l'altro.