Integrazione di Sistemi Embedded
Lab 07 Report

Matteo Perotti (251453)
Giuseppe Puletto (251437)
Luca Romani (255244)
Giuseppe Sarda (255648)

January 4, 2019

Abstract

E' stato creato un sistema che si compone di due parti: un programma Python che viene eseguito da PC e un programma C caricato e fatto girare su scheda NUCLEO-F401RE. La comunicazione tra i due dispositivi avviene mediante un cavo USB che emula una seriale. Il debug stato eseguito grazie alla modalit di SEMIHOSTING che ha permesso il reindirizzamento dello std-output dal processore ARM Cortex M4 passando per il debugger/loader ST-LINK fino al PC.

1 Descrizione dell'algoritmo

Le specifiche sono disponibili nel documento "lab07.pdf". Di seguito verranno chiarite le specifiche non definite:

1. Qualora venga inviato un comando di accensione LED mentre esso gi acceso, il comando viene considerato non valido e semplicemente ignorato. Stessa cosa se il comando chiede di spegnere un LED gi spento.
2. E' necessario che dal PC i comandi arrivino nel formato corretto. Non devono essere inviati comandi pi lunghi di due caratteri.
3. Per ottenere un risultato corretto indispensabile non attendere per pi di 65 secondi dopo l'accensione/spegnimento del LED prima di premere il pulsante.

2 Programma Python

2.1 Graphical user interface GUI

L'interfaccia grafica stata creata usando uno script in python e adoperando il modulo "tkinter". La GUI consente all'utente di mandare i comandi "L0" e "L1" mediante l'utilizzo di due tasti presenti a schermo. Per quel che concerne l'utilizzo di funzioni e l'inizializzazione della porta seriale, l'interfaccia grafica si appoggia allo script lib.py.

2.1.1 Widgets

L'interfaccia grafica ha una finestra principale chiamata "root" che viene creata mediante l'inizializzazione dell'interprete "Tk()" del modulo di tkinter e al suo interno contiene gli oggetti:

- main_Frame
- led_on
- led_off
- quit_button
- text_score_lab
- t_score_lab

mainFrame un frame che funziona da contenitore per i widget dell'interfaccia grafica. È stato istanziato per gestire in maniera separata il comportamento della finestra principale e dei widget. Gli altri oggetti sopra elencati sono i widget dell'interfaccia grafica e in particolare:

led_on un bottone verde che se premuto richiama la funzione ¡da definire¿ per inviare il comando "L1" tramite la seriale alla scheda Nucleo.

led_off un bottone grigio che consente all'utente, come il widget led_on, di trasmettere il comando "L0" tramite seriale per spegnere il LED della Nucleo.

quit_button un bottone rosso che consente all'utente di uscire dal programma. Se premuto invoca la funzione "quit()" dell'oggetto root che fa uscire lo script dal mainloop() dell'interfaccia grafica.

text_score_lab un'etichetta che contiene del testo (Your time score is).

t_score_lab un'etichetta che mostra il valore della variabile t_score aggiornato in tempo reale.

2.1.2 Struttura

Per quello che riguarda il geometry manager si deciso di adoperare "pack()" per gestire mainFrame e "grid()" per i vari widget. Dato che root ha un solo frame stato usato pack(), in questo modo stato pi facile da configurare l'opzione di riempimento totale di mainFrame nella root e il suo ridimensionamento sempre in funzione della finestra padre. Al fine di ottenere questo risultato stato necessario impostare in pack() i flag di "expand" e "fill" rispettivamente a "BOTH" e "1". Inoltre bisogna anche specificare quali righe e colonne del frame dovranno ridimensionarsi, per questo motivo sono stati inseriti due cicli for annidati che per ogni riga e ogni colonna vanno a mettere la variabile "weight" delle funzioni "rowconfigure()" e "columnconfigure()" a 1.
Avendo in totale cinque widget, si scelto grid() per avere un'organizzazione di tipo matriciale della GUI, in seguito viene riportata la matrice dell'interfaccia:

	1	2	3
1	led_on		led_off
2	text_score_lab	t_score_lab	
3		quit_button	

Anche per il manager grid necessario cambiare delle variabili per ottenere un ridimensionamento automatico di ogni widget in funzione della finestra padre, di conseguenza stata richiamata la funzione "grid_rowconfigure()" e "grid_columnconfigure" per andare a specificare quale riga e colonna dovessero essere ridimensionate dinamicamente. Per capire il fattore di scalamento del widget bisogna assegnare un valore alla variabile "weight" delle due funzioni sopra citate, in questo caso per tutti i widget, tranne il quit button, stato scelto il valore 1.

3 Programma C

Il programma C deve permettere al microcontrollore di ricevere dati da seriale, interpretarli, accendere e spegnere un led, contare intervalli di tempo, formattarli e poi inviarli nuovamente tramite seriale.

3.1 CubeMX

Grazie al software CubeMX stato possibile ottenere fin dall'inizio una struttura consistente del programma in modo molto agevole.

UART2 Dagli schematici possibile vedere che il Cortex-M4 e l'ST-LINK sono collegati tramite seriale. Quest'ultima collegata ai pins PA2 e PA3 del controllore principale. Dal manuale della scheda NUCLEO viene esplicitato come la seriale interessata sia USART2. In CubeMX stato quindi selezionata quest'ultima e le impostazioni sono state regolate da specifiche; in aggiunta stata selezionata anche l'inizializzazione degli interrupts relativi. Questo serve per poter gestire in modo non-blocking la ricezione e la trasmissione.

TIM1 E' necessario contare con una risoluzione del millisecondo, ed il valore massimo esprimibile in tale unit di misura 0xFFFF, ovvero 65,535 secondi. Dal reference manual del microcontrollore possibile vedere che il timer TIM1 collegato ad APB2. Da CubeMX si pu impostare il clock relativo affinch sia di 42 MHz, e il timer affinch il prescaler lo faccia scendere ancora fino ad 1 kHz. In questo modo viene evitata la creazione di una funzione di conversione apposita per ottenere il tempo in millisecondi a partire dal contenuto del registro del timer: con queste impostazioni essi si equivalgono.

GPIO Vengono usati il LED2 a disposizione dell'utente e il blue PushButton. Di quest'ultimo importante attivare l'interrupt corrispondente allo stato di output asserito, ovvero al falling edge (si pu ricavare dallo schematico che il pulsante cortocircuita a gnd il pin del microcontrollore).

Clock settings Come gi spiegato nel precedente paragrafo stato modificato il clock che arriva ad APB2. Il prescaler relativo stato messo a 4. Il main clock 84 MHz, viene quindi diviso per 4 per poi essere moltiplicato per 2 ed arrivare ad APB2.

3.2 Scrittura del programma

Il programma utilizza funzioni messe a disposizione dall'HAL della ST. Gli interrupt di ricezione e trasmissione UART, insieme con quello del push-button, sono stati astratti con tre funzioni callback. Per prima cosa sono definiti il buffer di trasmissione e ricezione (5 e 2 bytes rispettivamente).

Prima del ciclo infinito viene chiamata la funzione $HAL\_UART\_Receive\_IT$, che abilita la ricezione in modalit non blocking con interrupt (disponibile anche in modalit blocking, oppure con DMA). Tutto il resto del programma implementato nelle funzioni di callback: non appena il buffer di ricezione pieno viene eseguita quella di ricezione, che si occupa di fare il parsing del comando ricevuto a condizione che non sia gi in corso una misurazione. Esso viene ignorato se non valido, mentre nel caso contrario viene eseguita l'azione corrispondente sul LED e viene avviato il timer. Un flag viene asserito per indicare che in corso una misurazione, e un altra variabile indica se il LED correntemente acceso o spento. In seguito viene ri-abilitata la ricezione dei dati da seriale.

Quando il pulsante premuto viene immediatamente salvato il valore attuale del counter in una variabile globale ed il timer viene fermato e resettato. Poi viene eseguito un controllo sul flag di misurazione: se non asserito, allora l'azione viene ignorata. Altrimenti il flag di misurazione viene deasserito e il valore della variabile di supporto convertito in esadecimale e formattato tramite la funzione sprintf su un'altra variabile temporanea. Con una funzione che implementa banalmente un ciclo for, questa "stringa" copiata sul buffer di trasmissione senza il null character finale, per essere poi inviata tramite la funzione non-blocking $HAL\_UART\_Transmit\_IT$. Il controllo sul flag eseguito successivamente al salvataggio del valore del counter per non introdurre un errore nel conteggio dovuto alla latenza delle operazioni di controllo.

La funzione callback di fine trasmissione pu non fare nulla ed implementata con una $\_\_NOP()$.

3.3 Test e debug

Ogni periferico stato testato singolarmente prima che venisse assemblato il programma. Il debug stato eseguito mediante l'utilizzo della printf grazie alla modalit di semihosting; essa stata implementata in una macro che riceve in ingresso una stringa che viene passata ad una printf ma con l'aggiunta di un carattere '
n' finale. Come riportato sul sito della ST questa una condizione necessaria per il corretto funzionamento del semihosting stesso.

I comandi sono stati inviati e ricevuti da PC per mezzo del programma minicom.