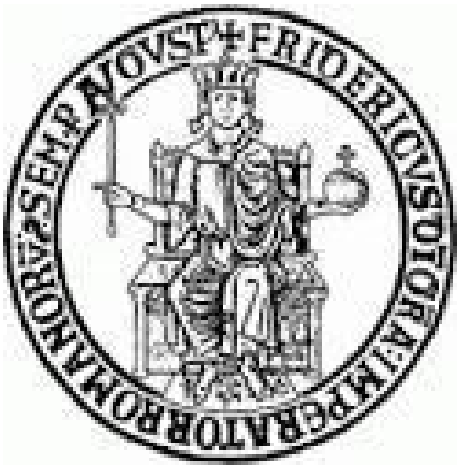

Università degli Studi di Napoli Federico II

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Corso di Computer System Design

Progetto di Sistema Embedded

Autori:
Benfenati Domenico, Camilla Fusco

Anno Accademico 2021-2022

Indice

Elenco delle figure

1 Introduzione

Il progetto sviluppato ha lo scopo di effettuare la trasmissione bidirezionale tra due schede STM32F303VC, che sfruttano il protocollo UART per inviare tra loro una sequenza prestabilita di stringhe.
La scheda infatti, ricevuto il messaggio tramite il protocollo di ricezione UART, si occupa di verificare l'ultimo carattere di tale messaggio:

- Se il carattere è pari a **0**, la scheda ricevente accenderà il LED di colore rosso.
- Se il carattere è pari a **1**, la scheda ricevente accenderà il LED di colore blu.
- Se il carattere è pari a **2**, la scheda ricevente accenderà il LED di colore verde.
- Qualsiasi altro carattere ricevuto, la scheda ricevente accenderà il LED di colore bianco, ottenuto tramite combinazione RGB.

2 Collegamenti

La soluzione prevede la configurazione mostrata in Figura 1.

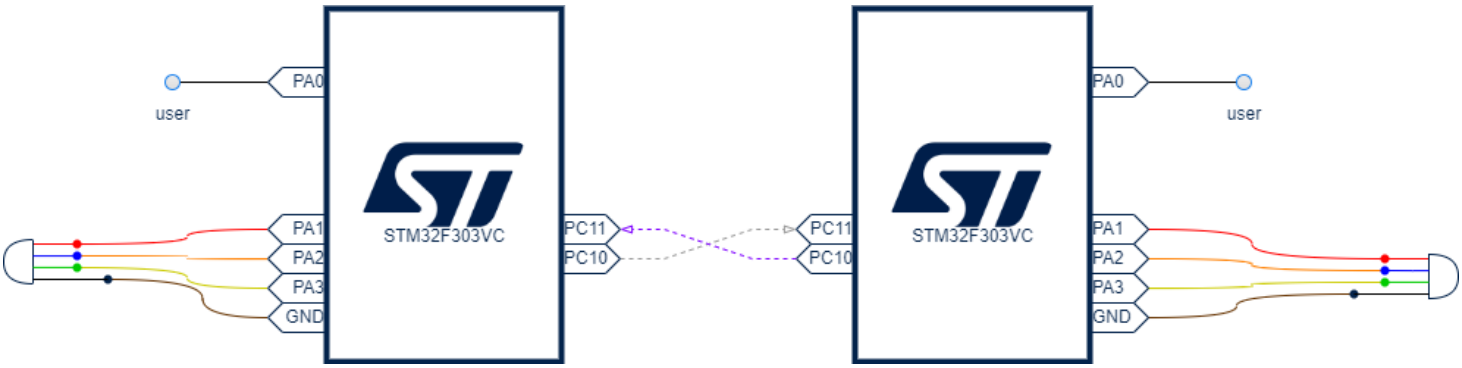

Figura 1: Collegamenti fisici tra le schede

Si elencano quindi di seguito e configurazioni apportate a tali pin:

- Il pin `PC11` è relativo all'utilizzo del protocollo UART. In particolare viene usata l'**UART4** presente all'interno della scheda STM32. Nello specifico, al pin PC11 è stato collegato il porto di **ricezione (UART4_RX)** dell'UART citata.
- Il pin `PC10` è relativo all'utilizzo del protocollo UART. Nello specifico, al pin PC10 è stato collegato il porto di **trasmissione (UART4_TX)** dell'UART citata.
- Il pin `PA0` è relativo al button **USER** presente all'interno della scheda. Tale pulsante è stato configurato come interrompente, settando tale pin come **GPIO_EXTI_0**, indicando che tale bottone aziona l'interrupt 0 all'interno del vettore delle interruzioni **NVIC**. Da tale vettore infatti è stata abilitata l'interruzione sulla posizione **EXTI0**.
- I pin `PA1,PA2,PA3` sono stati collegati ai piedini di un LED esterno RGB, rispettivamente PA1 al colore Rosso(R), PA2 al colore Blu(B) e PA3 al colore Verde(G). Tali pin infatti configureranno un differente colore che il LED deve assumere, a seconda di quale pin viene attivato. Tutti e tre i pin sono quindi stati settati come **GPIO_output**.
- Il pin `GND` è relativo alla messa a massa, e vi è stato collegato il piedino del LED corrispondente al ground, nel disegno indicato come piedino più lungo.

L'elenco dei pin utilizzabili è reperibile all'interno del manuale della scheda utilizzata *STM32F303xC Reference Manual* 2017.

3 Codice

Si riporta quindi di seguito il codice corrispondente alle varie funzionalità della scheda. Visto che entrambe le schede svolgono sia la fase di trasmissione che quella di ricezione, il codice montato all'interno delle due STM32 è il medesimo. Infatti, si è fatto in modo che ogni scheda potesse comunicare con l'altra tramite il proprio tasto USER.

3.1 Definizioni di variabili globali

```
UART_HandleTypeDef huart4;

uint8_t N = 5; //Ogni messaggio contiene N caratteri
uint8_t M = 8; //Vengono inviati M messaggi
uint8_t counter = 0;
uint8_t countersend = 0;
// Caratteri da trasmettere
char* bufftrasmit[] = {"luci2","cant0", "p1gn4", "fior1",
                       "c0cc0", "babb1", "fest4", "neve2"};
// Caratteri in ricezione
char* buffreceive[] = {"00000","00000", "00000", "00000",
                       "00000", "00000", "00000", "00000"};
```

3.2 Interrupt Pressione del Tasto USER

```
void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin){
    // Trasmissione tramite UART4 del carattere espresso dal *counter*
    HAL_UART_Transmit_IT(&huart4, (uint8_t*) bufftrasmit + N*counter, N);
    // Incremento del contatore in trasmissione
    countersend ++ ;
    if(countersend == M) countersend = 0;
    //il LED sul pin P15 interno alla scheda comunica che è avvenuta la trasmissione
    HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_15);

}
```

3.3 Controllo in Ricezione

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    // Salvataggio ultimo carattere ricevuto
    char last_byte = buffreceive[counter][N-1];
    // al primo messaggio ricevuto vengono resettati tutti i pin GPIO
    if (counter == 0)
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_All , GPIO_PIN_RESET);
    // ---ATTIVAZIONE COLORE ROSSO
    if (last_byte == '0'){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);    //LED rosso
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_RESET);
    }
    // ---ATTIVAZIONE COLORE BLU
    else if(last_byte == '1'){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET);    //LED blu
```

```
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_RESET);
    }
    // ---ATTIVAZIONE COLORE VERDE
    else if(last_byte == '2'){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);    //LED verde
    }
    // ---ATTIVAZIONE COLORE BIANCO
    else{
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);    //LED rosso
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET);    //LED blu
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);    //LED verde
    }
    counter++;
    // se sono terminati i messaggio si riparte dal primo messaggio
    if (counter == M)
        counter = 0;
    // Ricezione successivo messaggio
    HAL_UART_Receive_IT(&huart4, (uint8_t*) buffreceive + N*counter, N);
}
```

Le funzionalità e le relative configurazioni della scheda sono reperibili all'interno del manuale completo *STM32F3 HAL and LL Drivers Manual* 2020

4 Funzionamento

Tramite l'IDE **STM32Cube** è stato effettuato l'assemblaggio di tale programma, cliccando su Project » Build all, e successivamente, a valle del collegamento tramite USB della scheda, sfruttando la porta di programmazione **USB ST-LINK** è stato caricato il codice su entrambe le schede, cliccando su Run » Run as... » STM32 Cortex-M C/C++ Application. Tale operazione effettua quindi la programmazione della scheda.
A valle della programmazione, è stato quindi testato il funzionamento di tale comunicazione. Si riportano di seguito alcune immagini dell'esecuzione della comunicazione, e si rimanda al video allegato per la dimostrazione pratica.

Figura 2: Invio e ricezione del messaggio che termina con "0"

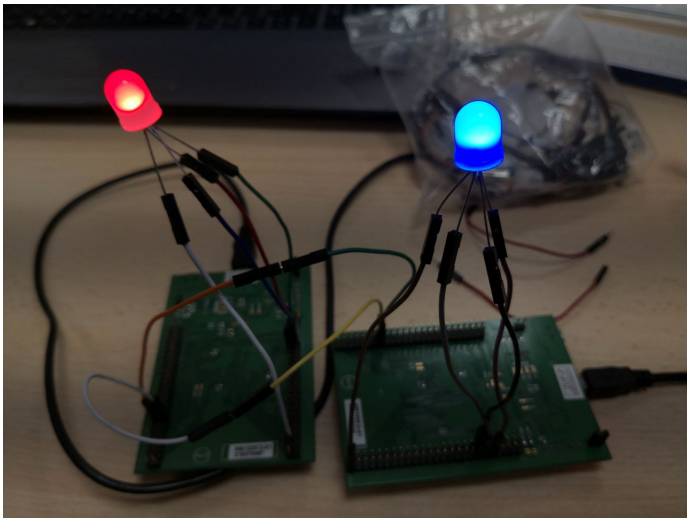

Figura 3: Invio e ricezione del messaggio che termina con "1"

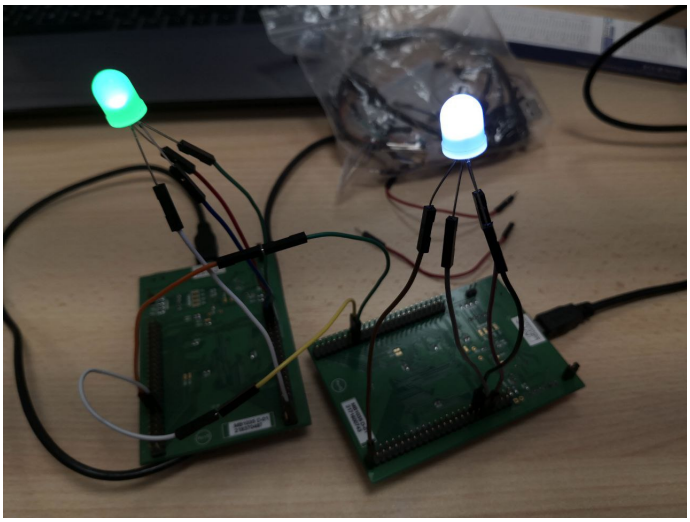

Figura 4: Invio e ricezione del messaggio che termina con "2"

Riferimenti bibliografici

STM32F3 HAL and LL Drivers Manual (2020). URL: https://www.st.com/resource/en/user_manual/um1786-description-of-stm32f3-hal-and-lowlayer-drivers-stmicroelectronics.pdf.

STM32F303xC Reference Manual (2017). URL: https://www.st.com/resource/en/reference_manual/dm00043574-stm32f303xb-c-d-e-stm32f303x6-8-stm32f328x8-stm32f358xc-stm32f398xe-advanced-arm-based-mcus-stmicroelectronics.pdf.