20/10/2017 Consegna n. 1 Simone Magnani - 754082

Follow the Light

Premessa

Lo studente si impegna ad utilizzare tutte le funzionalità apprese durante le lezioni frontali, in modo tale da realizzare un elaborato robusto, completo e funzionante. Sono state prese certe scelte durante la programmazione che potrebbero essere discutibili; per evitare ambiguità, nelle prossime sezioni verranno evidenziate e spiegate le scelte più significative.

Analisi della Consegna

Viene richiesto di realizzare un sistema embedded su Arduino che implementi il gioco *Follow the Light* servendosi di 3 led verdi, 1 led rosso, 3 pulsanti tattili e 1 potenziometro. Prima ancora di iniziare a scrivere codice si può intuire il numero di pin da utilizzare: 3(led verdi) + 1(led rosso) +3(pulsanti) + 1(potenziometro) + 1(ground) + 1(5V) = 10 pin (minimo).
Riguardo alla realizzazione fisica del circuito, non vi sono speciali vincoli, in quanto ognuno è libero di riprodurlo a proprio piacimento a patto che si aggiornino i valori dei pin utilizzati nel programma.

Analisi dello schema Fritzing

Nello schema fritzing proposto nel file *SchemaFritzing.pdf* ho fornito una personale interpretazione del circuito, cercando di realizzare uno schema pulito e facilmente comprensibile (purtroppo per motivi di spazio e dimensioni componenti su Fritzing, ho utilizzato la breadbord grande al posto della versione ridotta che avrei preferito).

Scelte Progettuali

Suddivisione file (è sottointeso che per ogni file di tipo .cpp vi è un file .h)

- *follow_the_Light.ino*: programma Arduino;
- *Compute.cpp*: file contenente le funzioni di puro calcolo;
- *Evaluations.cpp*: file contenente la funzione che valuta la sequenza inserita dall'utente;
- *Lights*.cpp: file contenete le funzioni relative ai led (accensione/spegnimento, ecc.);
- *Map*.cpp: file contenente le funzioni di mapping tra led, pulsanti e i numeri della sequenza;
- State.cpp: file contente le funzioni di interrupt relative al cambio di "stato" di gioco iniziale;
- Settings.h: header file in cui vi sono tutte le istruzioni #define.

La scelta di creare cosi tanti file è data dall'intento di distinguere le varie funzioni cercando di isolare i vari compiti, in maniera tale da realizzare codice leggibile, facilmente comprensibile e riusabile.

Analisi Interrupt & Sleep Mode

Il led rosso che lampeggia regolarmente non appena si avvia il programma è pilotato dall' interrupt relativo al Timer1, appositamente inizializzato.
Al bottone1 è attaccato l'interrupt relativo all'incremento di stato, ovvero alla richiesta di iniziare la partita; al contrario, il bottone2 ha un apposito interrupt per decrementare lo stato, tornando alla fase iniziale di scelta della difficoltà tramite potenziometro.
Una volta finita la partita, Arduino va il Power Down mode, una tra le varie modalità di risparmio energia presentate a lezione. L'unico modo per risvegliare Arduino e, quindi, iniziare una nuova partita, è tramite un differente interrupt associato al bottone1 che permette di procedere all'interno del ciclo di super loop, ritornando quindi alla funzione di inizio gioco.
Ho preso tali scelte non solo perché a volte sono risultate più comode, ma anche per cercare di scrivere codice non ripetitivo e monotono, servendomi in questo modo di funzionalità che richiedono un'implementazione diversa dal semplice ciclo while di polling.
Da notare che tali cicli di polling non sono assenti, non essendo in grado di evitarli e non volendo attivare ogni volta una modalità di risparmio energetico ho deciso di inserire dei controlli senza istruzioni interni. Essi servono quindi solamente per aspettare che lo stato di gioco cambi.

Analisi #define e "magic number"

Nel file Settings.h ho messo tutte le *define* utili per la realizzazione di codice riusabile. Vi sono infatti le definizioni dei pin utilizzati, del tempo di bouncing dei pulsanti, del tempo concesso al giocatore per premere un singolo bottone, degli stati di gioco e del numero massimo di livelli.
Con ciò riconosco che il codice non è 100% riutilizzabile, perché ho volutamente lasciato dei magic number all'interno del programma, ad esempio nelle funzioni di calcolo, nelle funzioni di delay e nell'uso di alcune variabili. Se in un futuro si volesse modificare tale caratteristica, basterebbe riorganizzare il codice con alcune define (e appositi controlli) e utilizzare queste dichiarazioni al posto dei magic number. Personalmente ritengo che a parte questi dettagli (che danno anche un tocco personale al gioco), io abbia cercato di sviluppare un codice leggibile e ben progettato.

Analisi variabili globali

Come si può notare dal main file *follow_The_Light.ino*, le variabili globali utilizzate sono per la prevalenza di tipo *short int*, in quanto dopo una valutazione riguardo al loro utilizzo ho deciso che non mi sarebbero serviti tutti i 32 bit del normale *int*, e quindi potevo benissimo risparmiarmene la metà (16 bit). In aggiunta, la variabile *state* è *volatile* per evitare le varie ottimizzazioni (nel mio caso inutili) che compie Arduino discusse a lezione.
Le variabili relative al tempo sono di tipo *unsigned long* per il semplice motivo che ho utilizzato la funzione *micros()* che ritorna un valore *unsigned long* (invece di *millis()*).

Conclusioni

Mi ritengo molto soddisfatto del lavoro svolto, gli obiettivi che mi ero prefissato li ho raggiunti e di sicuro utilizzando le varie funzionalità mostrate a lezione, penso di aver consolidato tali argomenti. Ho molto apprezzato la libertà decisionale lasciataci dal professore e i vari confronti con gli altri studenti nel forum riguardo problematiche riscontrate da più persone.