Il progetto si sviluppa essenzialmente su 4 file.

IRQ_timer.c al cui interno vi è l'interrupt handler di TIMER0, che viene usato per gestire in polling il touchscreen (1 lettura ogni 500us). Una volta lette le coordinate del tocco, queste vengono usate per capire l'area in cui questo è avvenuto. Se il gioco non è ancora inziato, e un tocco nell'area del labirinto viene rilevato, allora si chiama la funzione play() per avviarlo. Analogamente per le aree relative ai pulsanti clear e restart, richiamando le apposite funzioni.

IRQ_RIT.c al cui interno vi è l'interrupt handler del RIT, che viene usato per gestire, ancora una volta in polling, il joystick. Innanzitutto, tutto il pezzo di codice dell'handler è bloccato da una if che controlla che sia stato avviato il gioco o non sia finito (vittoria), in modo da bloccare il joystick se non è stata avviata ancora una partita. Il RIT handler, che scatta ogni 50ms si occupa di controllare per ogni direzione e per il tasto select se questo sia stato premuto e di scatenare le opportune azioni: il cambio di modalità Explore/Move per il tasto select (e successivo aggiornamento su display dell'indicazione della modalità), ed il cambio di direzione (subito dopo i 50 ms) per le quattro direzioni. Inoltre, se i tasti direzionali vengono letti come premuti costantemente per multipli di 20 scatti del RIT handler (quindi multipli di 1 secondo di pressione) e se in modalità Move, viene effettuato il movimento del robottino di una casella al secondo nella direzione opportuna, attraverso la funzione move(), aggiornando opportunamente le variabili y,x di coordinata del robot.

La restante logica del gioco risiede nel file blind_labyrinth.c

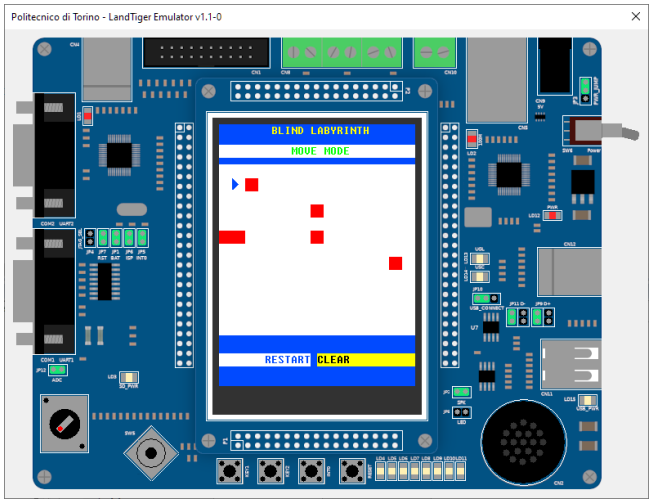
Sia la rotazione che il moviemento del robot, vengono riportati a display, prima cancellando il vecchio simbolo del robottino (tramite clear_robot()) e dopo ridisegnando il nuovo simbolo con la nuova posizione/nuovo orientamento (tramite draw_robot()).

Ad ogni movimento/rotazione viene inoltre richiamata la funzione look() che si occupa di guardare fino a 5 caselle avanti al robot e di disegnare un ostacolo rosso se rilevato. Sia la funzione move() che look() ovviamente implementano tutti i controlli per non muoversi sopra un ostacolo e non uscire dalle pareti del labirinto.

Inoltre, ad ogni movimento, viene anche richiamata la funzione win_check(), che si occupa di controllare se si è giunti su una uscita, segnalando la vittoria, bloccando joystick e tasto clear (solo "Restart" rimane cliccabile), rendendo l'area del labirinto a sfondo verde e comunicando il messaggio "You win!"

La funzione restart_button() viene chiamata al primo avvio del gioco e ogni qualvolta viene premuto attraverso il touchscreen il tasto "Restart". Questa funzione ripristina il robot nella sua posizione e orientamento iniziali, ripulisce il labirinto ed essenzialmente riporta il sistema come al primo avvio.

La funzione clear_button() si occupa solo di ripulire l'area dagli ostacoli scoperti, lasciando invariate la posizione e l'orientamento del robottino, all'ultimo movimento dell'utente. Questa effettua il refresh di tutta l'area del labirinto, per cui la sua esecuzione è un po' lenta.



L'area del labirinto sul display è un rettangolo di dimensioni 208x240 e mappandola sulla matrice del labirinto 13x15, ogni casella è un quadrato di 16x16 pixel.

Per cui, per poter individuare il punto corretto in cui disegnare i simbolini di robot ed ostacoli, tutte le coordinate di posizione x e y del robot nella matrice vengono moltiplicate per il valore 16 per posizionarsi correttamente sul display.

Una funzione LCD_DrawTriangle aggiunta nella libreria GLCD.c si occupa, dato un vertice, la base del triangolo, la dimensione, l'orientamento e il colore voluti di disegnare il triangolo del robot. Questa funzione è utilizzata anche da clear_robot() per cancellare il robot prima di disegnare il nuovo, disegnando un triangolo con le stesse caratteristiche ma di colore bianco (uguale allo sfondo del labirinto).

*All'avvio del gioco è necessario effettuare la calibrazione del touchscreen
*Per via della simulazione il refresh dell'area del labirinto è molto lenta
*Gli interrupt su button sono stati disabilitati all'avvio (eccetto il reset). Il lampeggio dei led non è in alcun modo dovuto alla logica del gioco ma alle funzioni di touchscreen e disegno elementi sul display.
*Anziché il doppio colore del robot per le modalità move/explore è stata utilizzata una dicitura più chiara e con colori diversi in alto al labirinto
*Si è evitato di disegnare la griglia del labirinto, altrimenti ad ogni movimento il refresh sarebbe stato molto lento.