

**Extra point #1
Max 2 points**



Expected delivery of extapoint1.zip must include:

- zipped project folder
- this document compiled in pdf
- A **4 minutes video** with audio (.mp4 or .avi) explaining how the project works; the video has to show a software debug session with all significant peripheral windows opened; the audio must be a voice recording of you describing (in Italian or English) the behavior of the running system.

Purpose of Part 1: to acquire full confidence in the usage of the KEIL **software debug** environment to emulate the behaviour of the LPC1768 and the LANDTIGER Board.

This part is evaluated to assign a maximum of 2 extra-points for qualified students taking the exam with a mark ≥ 18

Quoridor

Quoridor is an abstract board game published in 1997 by Gigamic.



1) Rules of the game

Each player is equipped with a **token** and **8 barriers**. The game board is a 7x7 wooden square, with the peculiar feature that the lines dividing and forming the squares are grooved, allowing walls to be inserted.

Each player starts from the centre of their perimeter line (the 4th square), and the goal is to get their token to the opposite perimeter line. The player who achieves this objective first wins.

On their turn, a player has two choices:

1. He/she can choose to move their token horizontally or vertically;
2. He/she can choose to place a wall. The wall cannot be jumped over but must be navigated around.

At every step of the game, there are always two players: the moving player (the one that is doing the choice), and the opponent. Their roles alternate at every step.

- If two tokens (the moving player and the opponent) are facing each other, the moving player can jump over the opponent and position themselves behind them if there is no barrier behind the opponent.
- It is not allowed to "trap" a player with the help of walls; you must always leave them a way to reach the goal.
- A move must be chosen within 20 seconds, or else it loses the turn.

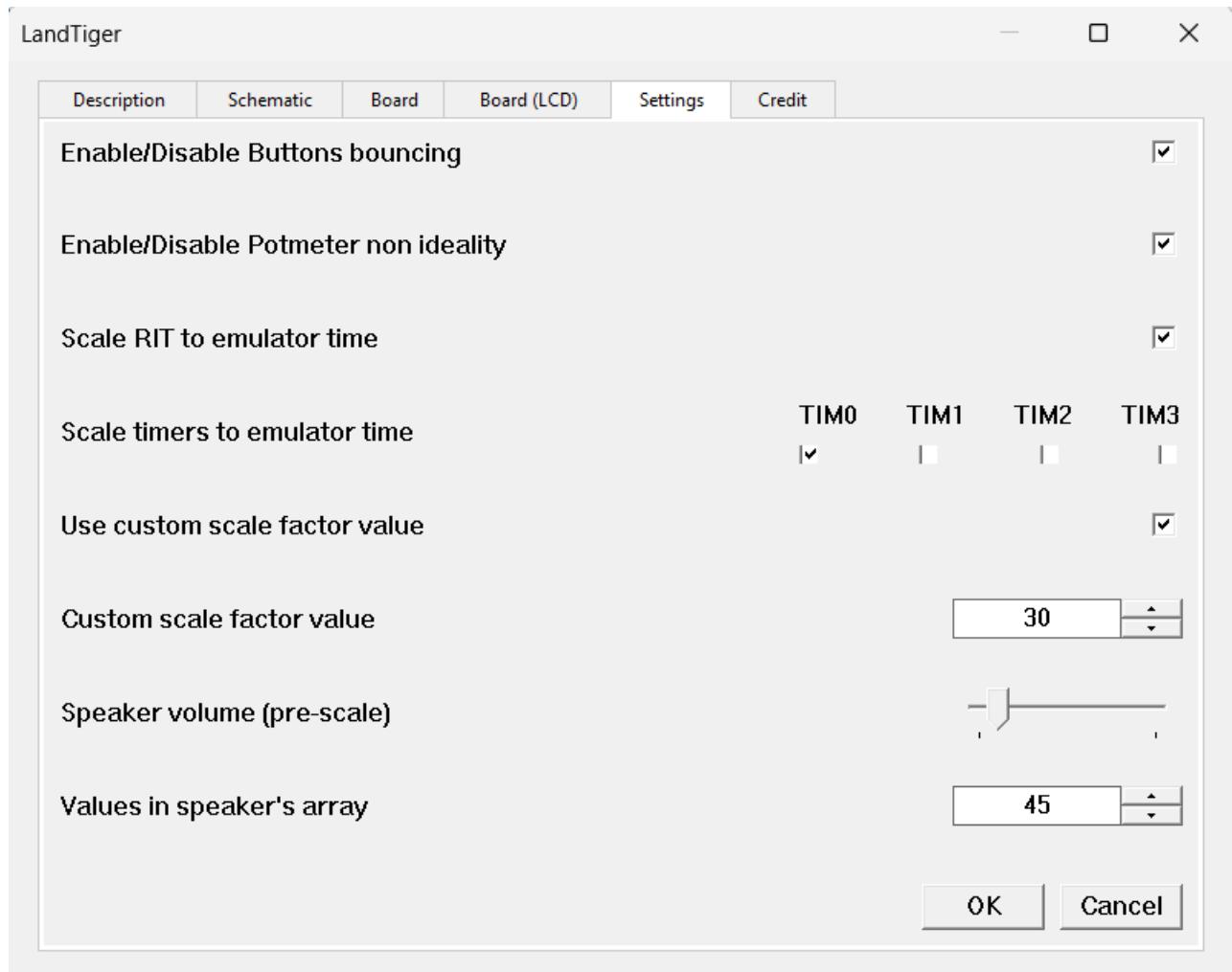
2) Implementation on the LandTiger Board or Emulator

In Keil µVision, use the **LANDTIGER emulator** or the actual **board** for implementing a Quoridor game.

Please deliver a zip folder with all the files of your project. At the end of the Keil project folder, please add “EMULATOR” if you developed it using the emulator or “BOARD” if you had the opportunity of developing the project on the actual board.

Example: extrapoint1_emulator.zip **or** extrapoint1_board.zip

Please attach any useful comments and your emulator configurations (substitute the image below).
Your project will be evaluated according to your specification!!



Additional Comments (C-variable/defines defined in your code, e.g., scaling factors) :

I valori del Rit e del timer0 sono i valori calcolati su 100MHz e 25MHz per 50ms e 1s divisi per 30. Nella foto dell'emulatore ho mostrato lo scaling attivo e con il valore di 30, ma la realtà è che non scalava e i valori li ho cambiati a mano, spero non sia un problema perché se non ricordo male in aula era stato detto che sarebbe potuto capitare.

Ho definito due nuovi tipi di variabile con typedef struct: la variabile di tipo pos, che è stata usata per tutti quegli oggetti che hanno una posizione nella mappa e sono caratterizzati da coordinate (int x, int y) un verso (int v) usato in modo diverso a seconda dei casi (per definire su, giù, destra e sinistra oppure verticale o orizzontale ad esempio), una variabile n (usata principalmente dalle pedine per registrare il numero dei muri) e un colore, e la variabile di tipo G, che contiene 4 campi pos g1, g2 per le pedine, mossa_sospesa, utile per la fase di decisione di movimento di una pedina, e muro_sospeso, utile per la fase di posizionamento di un muro, un int turno che tiene il conto di quale giocatore sia di turno, int game_mode, che rappresenta la modalità di movimento (game_mode = 0) o di posizionamento del muro (game_mode = 1). Se game_mode == 2 la partita è finita e dunque si arresta tutto. Infine è presente anche una matrice int i[13][13] che ha registrati tutti i valori di ogni oggetto: 1 se muro, 2 se pedina, 0 se libero. Ne deriva da questo che una pedina potrà stare solo in coordinate pari e che il centro di un muro potrà trovarsi solo in coordinate dispari. La prima posizione è usata per le x e descrive la scacchiera lateralmente, la seconda posizione è usata per le y e descrive la tastiera verticalmente; ciò significa che se g1.x = 6, g1.y = 0 e i[g1.x][g1.y] = 2, anche se effettivamente il numero è registrato nella prima cella della settima riga sulla scacchiera ci si riferisce alla settima cella della prima riga. Inizialmente potrà sembrare confusionario o controproducente ma rimanendo coerenti con i dati del tipo i[x][y] ci si ritrova nella familiarità del mondo cartesiano (con asse positivo delle y verso il basso) che agevola non di poco le astrazioni mentali.

La funzione LCD_draw_game disegna il gioco, setup_turn cambia turno, riazzera muro_sospeso, eguaglia le coordinate di mossa_sospesa con quelle della pedina che sarà di turno e imposta v a -1. Se il giocatore muoverà il joystick la mossa sospesa registrerà nella v la direzione (0 su, 1 giù, 2 sinistra, 3 destra, 4 secondo quadrato in su e così via) e si colorerà il bordo della cella. La cella si colora solo se la direzione scelta fa parte delle Hmoves, cioè delle mosse ipotetiche, che sono calcolate all'inizio del turno da un'apposita funzione. Se poi il giocatore ne seleziona un'altra, la cella con il bordo colorato torna del colore iniziale e si colora quella selezionata. Se cambia modalità ritorna tutto come prima e appare un muro o al centro o, se è occupato, in una posizione definita dalla funzione search_free_place. Muro_sospeso acquisisce le coordinate del punto dove vuole essere collocato il muro (non ancora confermato) e viene stampato a schermo. Se si muove, si cancella dal vecchio posto colorandolo del colore dello sfondo, si cambiano le sue coordinate e si ricolora del colore del muro sospeso. Se lo si indirizza in un posto dove non può andare diventa rosso per un istante, se lo si conferma viene stampato con un colore più scuro. Prima di muoversi un muro controlla che la prossima postazione sia libera, altrimenti cerca quella successiva. Se non ce n'è una libera diventa rosso per un istante. Se si ritrova bloccato con nessuna direzione in cui potersi muovere (la casistica in cui il muro è in verticale ma se si ruotasse potrebbe spostarsi non fa parte di questo scenario; lo scenario descritto, come mostrato anche nel video, è quello in cui anche ruotando si finirebbe in errore in tutti e 4 le direzioni) viene spostato dalla funzione search_free_place.

Il timer conta un secondo e decrementa una variabile int secondi[2] sempre contenuta in G g. Se arriva a zero resetta tutto ciò che è sospeso e cambia turno.

Il joystick e i bottoni si appoggiano sul rit, il joystick per necessità mentre i bottoni per garantire il debouncing.

Non si può selezionare nulla se non si è fatta nessuna mossa, cioè se si preme select senza mai aver mosso il joystick non si "conferma di rimanere fermi" e non si perde il turno.

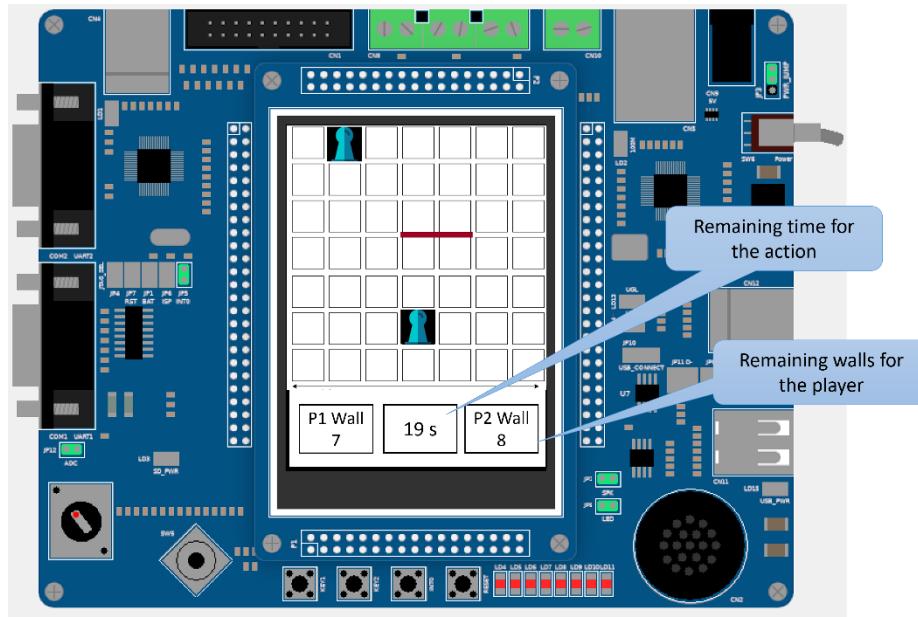
La funzione mossa32bit si occupa di tradurre su 32 bit la mossa che è stata effettuata.

Specifications

Note that images are only for explanation purposes.

You must implement for the LandTiger board the Quoridor game.

An implementation example could be the following.



Using the DISPLAY library, draw your game board as a **7x7 wooden square**.

Draw the two **tokens** using two colours: white (PLAYER1) and red (PLAYER2).

Remember that each player has 8 available **barriers (or walls)**.

The tokens must move using squares (vertically or horizontally), Figure 1.

The barriers must be placed between squares, Figure 2.

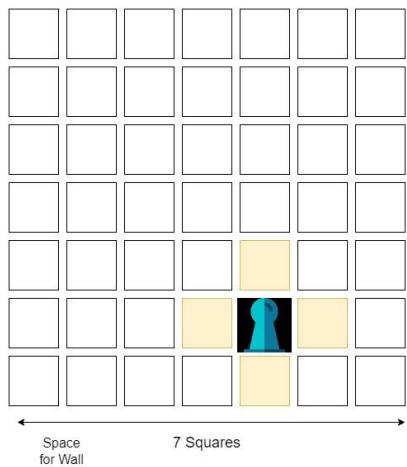


Figure 1

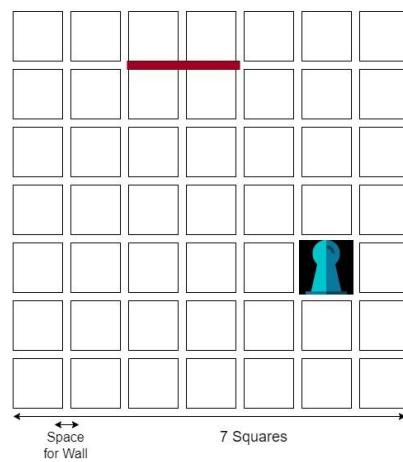


Figure 2

When the system starts, to start a match, you must press **INT0** (otherwise you remain in wait mode).

From now on, you are in the match/game mode.

Each player has 20 seconds to do the move. If time expires, the player **loses the turn**.

At this point, remember that the player has two choices:

1. He/she moves of 1 step vertically or horizontally. To do this, use the **JOYSTICK**.
 1. **JOY UP**
 2. **JOY DOWN**
 3. **JOY LEFT**
 4. **JOY RIGHT**
 5. **JOY SELECT**: confirm the choice. The game continues with **PLAYER2**.
 6. The 4 squares surrounding the player (as in Figure 1) are highlighted, i.e., the player's possible moves.

2. He/she can insert a wall to block the opposite player. To enter in this mode, the player must press **KEY1**. A wall will appear in the centre (the map itself is 7x7 and the wall is placed between two squares).
 1. After pressing **KEY1**, it appears in the centre of the 7x7 square.
 2. Use the **JOYSTICK** to determine the position of the wall. Once that the position is found, press **JOY SELECT** to confirm the position.
 3. Be careful about the wall position. A wall is sized with two squares plus a space between them.
 4. Each player has only 8 walls to position. If it ends them, a **warning message** will appear on the screen: "No walls available, move the token".
 5. Pressing **KEY2** rotates the wall in the current position of 90 degree.
 6. Pressing **KEY1** again before confirming the position of the wall disable the wall insertion mode and return to player move mode.

Every **move** must be saved in a **32-bit unsigned int variable** to include the following information:

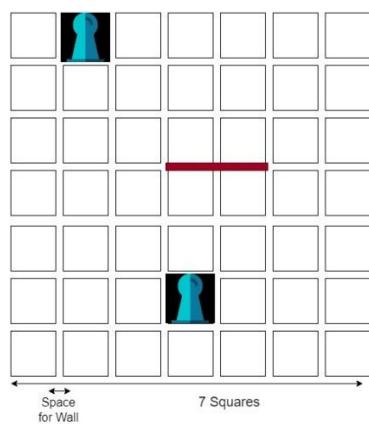
PlayerID	PlayerMove/WallPlacement	Vertical/Horizontal	Y	X
8 bits	4 bits	4 bits	8 bits	8 bits

- PlayerID: 0/1 to identify the player
- PlayerMove/WallPlacement: if 0 you are moving the player, otherwise you are placing a wall
- Vertical/Horizontal: if 0 you are placing a wall in Vertical orientation, otherwise Horizontal orientation. Default to 0 if moving the player.
- Y and X: they contain the coordinates of the **new position** of the player (in player mode) or the coordinates of the centre of the wall to place (in wall placement mode).

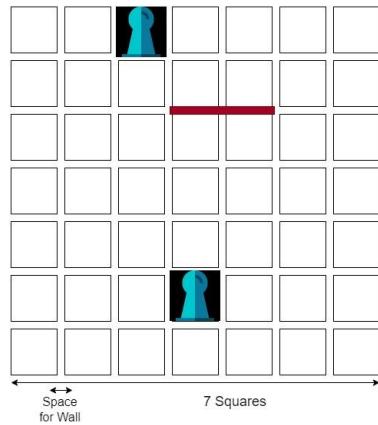
Note that if the player runs out of time the move is not valid and his/her turn is skipped. This is represented by PlayerMove/WallPlacement set to 0 (player move mode) and Vertical/Horizontal set to 1.

An example on how the wall functionality is implemented is shown in the following figures:

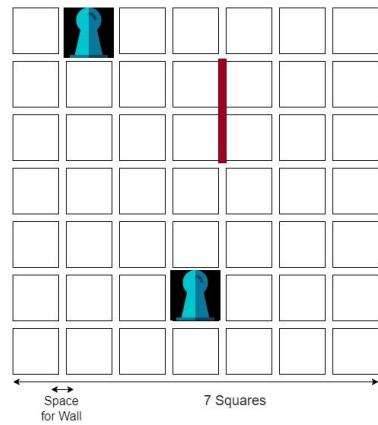
1. After pressing KEY1 a wall appears in a central place on the map. If you want, you can customize its colour to show that the position is not yet finalized.



2. By going up with the Joystick the wall is moved up by one square



3. By clicking **KEY2** the wall is rotated by 90 degrees



4. By clicking **JOY SELECT** you finalize the position of the wall, and your turn is finished.

Remember that the game ends when one of the players reaches the opposite side of the 7x7 square.