物聯網應用設計與實作 - 期末專題

Battleship IoT

隊名：第五組

組長： 0616231 彭世丞

隊員： 0516098 吳涵毅

0616095 曾敏峰

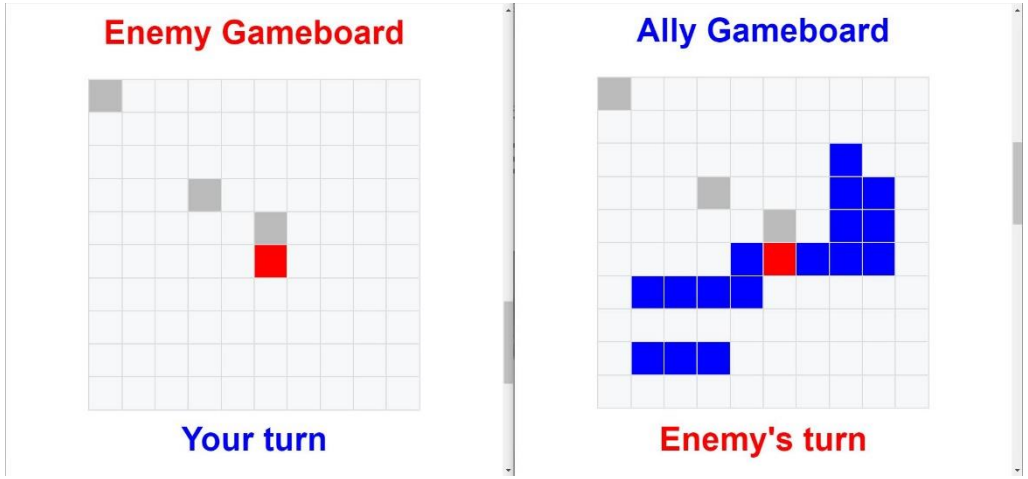
0716031 陳煜凱

0716010 張晏禎

專題簡介

我們這組使用Javascript製作了雙人對戰的battleship遊戲。

接著利用IoTtalk平台自行建立獨特的battleship device model，並互相連線，傳輸雙方的資料來進行遊戲



海戰船（Battleship）規則

- 雙方玩家皆有一個己方的10*10棋盤，棋盤中放有五艘艦船：1*5、1*4、1*3（兩艘）、1*2
- 對手的擺放位置在遊戲最初無法得知
- 玩家輪流猜測對方船艦的擺放位置，一次猜測一格，並顯示猜測的結果
- 較先猜中所有敵方船艦的一方獲勝

Battleship程式設計

主要分成三部分：

- Battleship：包含隨機生成船艦位置等等遊戲本身的架構及邏輯設計（利用網路上單人版本Battleship的code稍作修改）
- IoTtalk：包含與IoTtalk平台連結所需要的部分（自行設計）
- 美化：利用CSS和jQuery美化網頁上呈現的遊玩過程

Battleship程式設計

可更改之參數

```
1   // Parameters
2   var rows = 10;
3   var cols = 10;
4   var squareSize = 8; // size of each square, unit= vmin
5   var ships = [2, 3, 3, 4, 5]; // length of each ship
6   var interval = 1500; // delay between each identical ajax request; unit: ms
7   var scrollAnimationTime = 1500; // duration of scroll animation; unit: ms
8   var colorAnimationTime = 750; // duration of color change animation; unit: ms
9   var dieTime = 20000; // after game over, stop sending ajax requests after this much time has elapsed; unit: ms
10  var allyGameBoard = generateGameboard(ships);
```

棋盤大小（rows=行數，cols=列數）

格子的大小

各艘艦船的長度

重複傳送相同訊息的時間差

選完格子後，畫面scrolling的時間

選完格子後，格子變色的時間

自動生成己方棋盤（船艦配置初始狀態）。

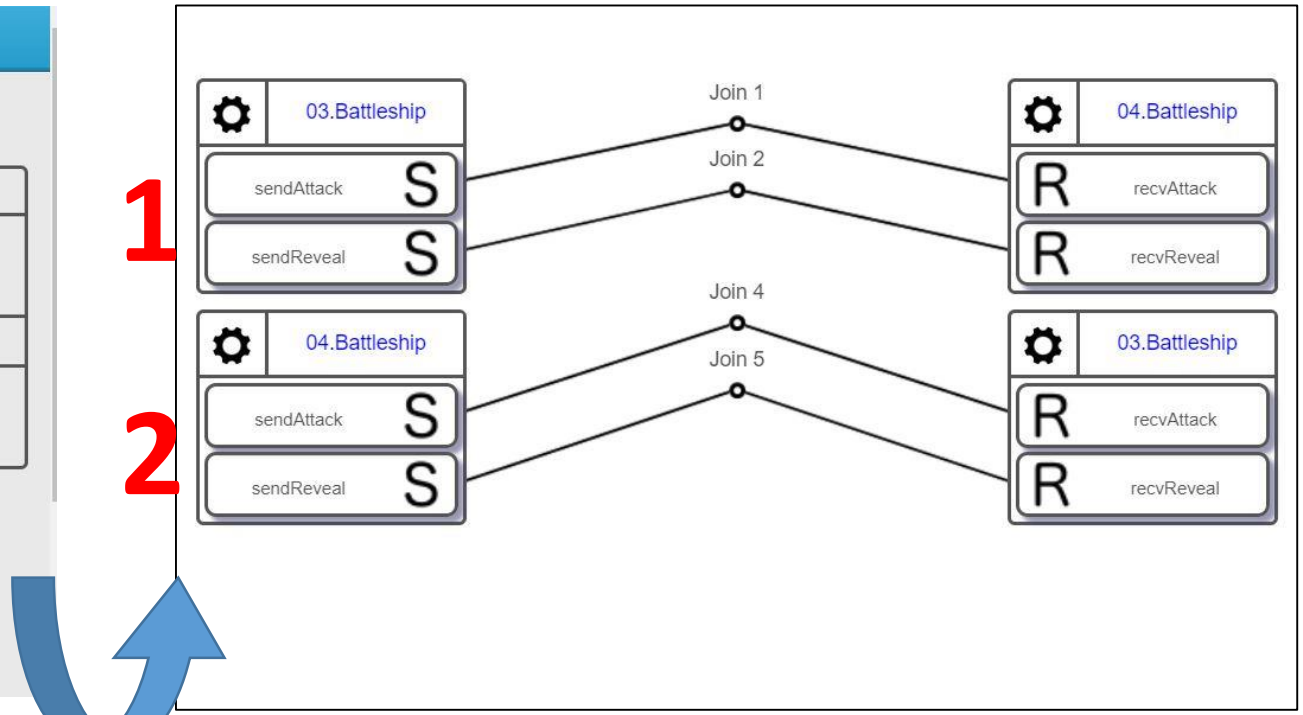
遊戲結束後，至停止發送訊息的時間。
可視網路狀況調整，例如若傳一次訊息需要5秒，則可設定10秒後停止發送，以(大致)確保對方知道遊戲已經結束。

若要自行設計棋盤，可用二維陣列代替，
惟棋盤大小需與上面rows, cols一致，而ships參數則不採用。

遊戲流程：前置工作（IoT）

在IoTtalk放上兩組包含input和output的device“Battleship”

（利用Device Feature Management自行設計）

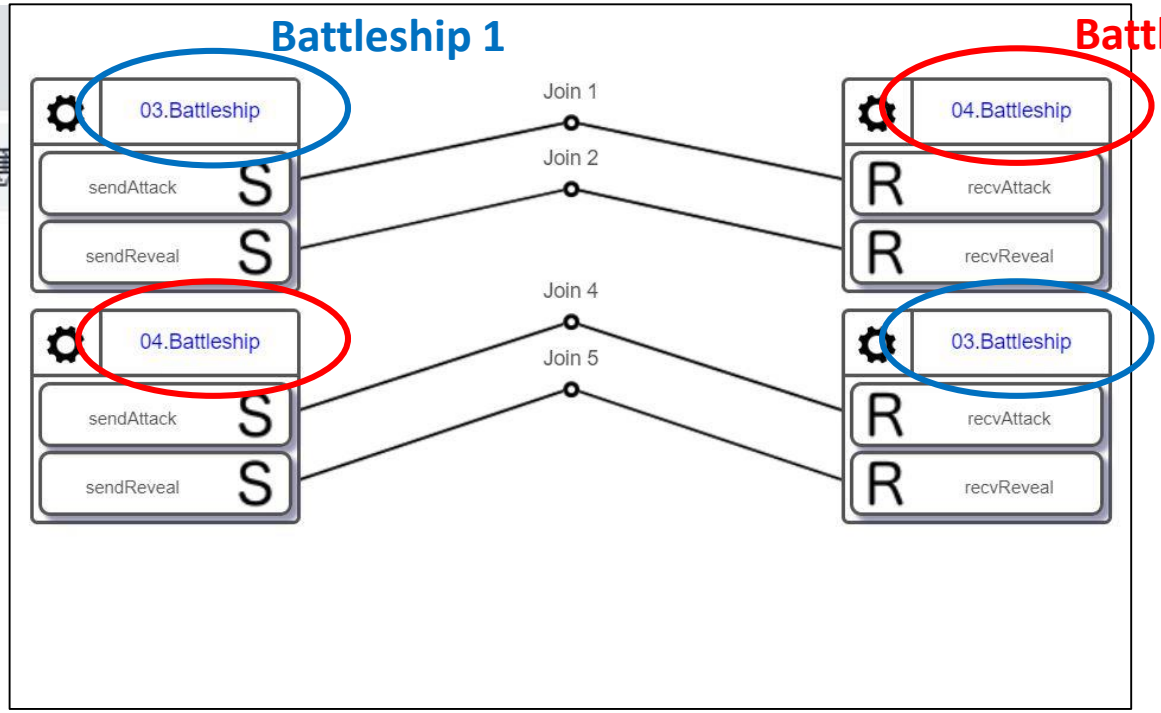


遊戲流程：前置工作（IoT）

開啟兩個Battleship網頁（分別代表兩個玩家自己的棋盤），並與DM之間做關聯

注意！！

兩個網頁的數字需要1奇1偶
奇數號先攻，偶數號後攻



Device Feature

- **sendAttack** ：當輪到你回合時，會把你所選的座標位置傳給敵人的網頁，並由敵人的網頁判斷你是否有擊中他的軍艦。
- **sendReveal** ：回傳給敵人他所選擇的格子座標以及此格的狀態（state），state表示此格是否有船艦。

Battleship遊戲介面

網頁包含兩個部分：Ally Gameboard 以及 Enemy Gameboard

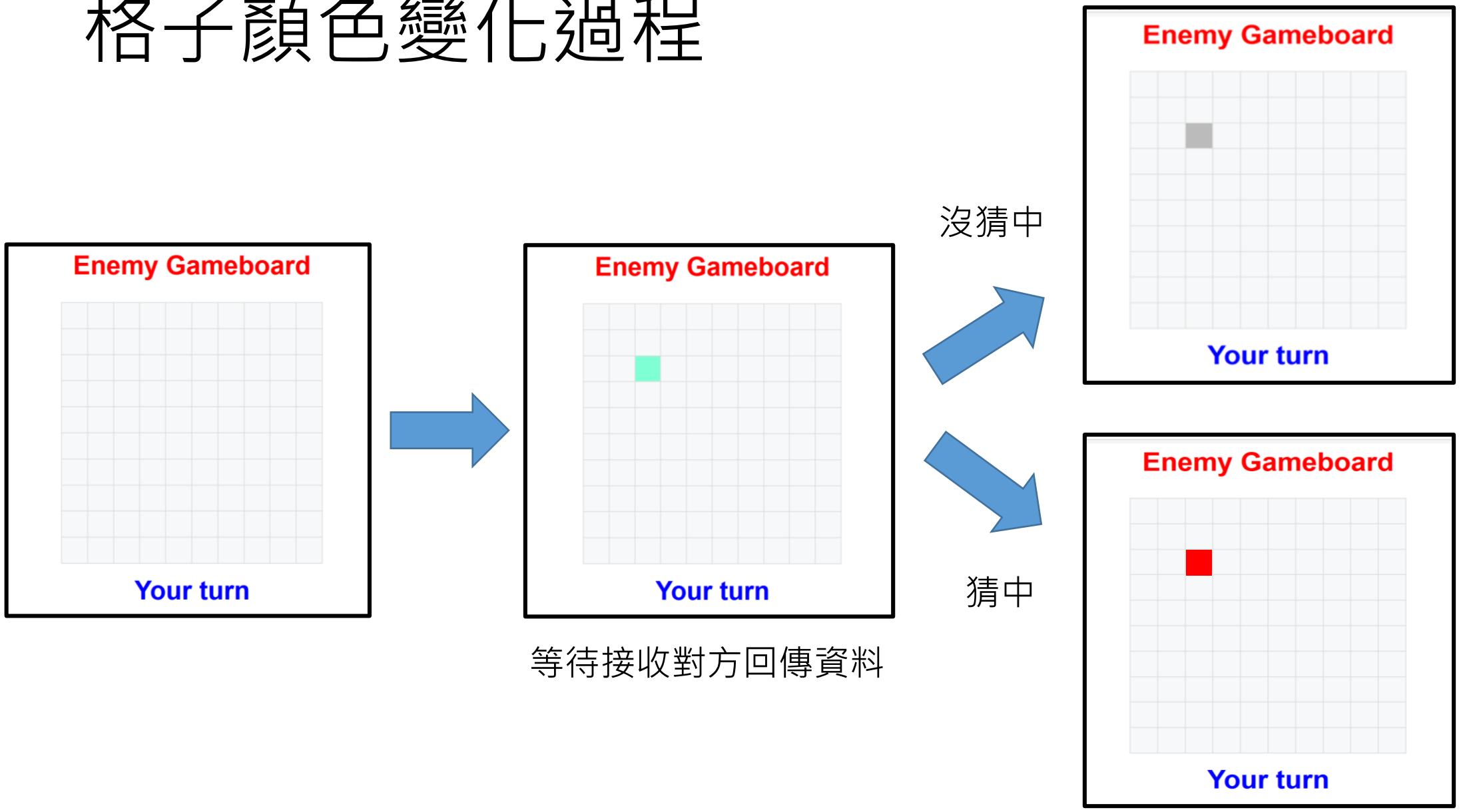
Ally Gameboard 顯示己方艦船位置以及目前已被對方猜過的格子

Enemy Gameboard 顯示己方猜測對方的情況

遊戲流程

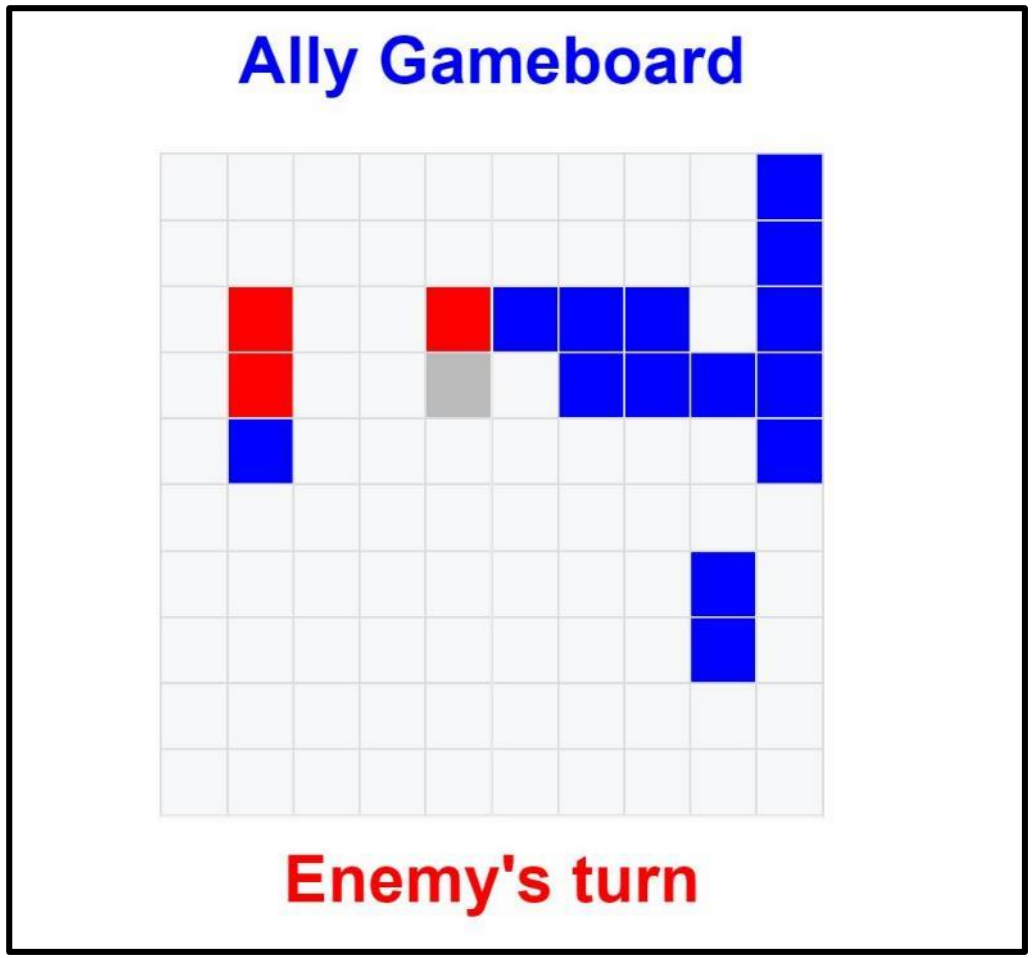
- 由奇數編號的玩家開始作戰
- 輪到自己的回合時，可以點擊Enemy Gameboard上任意一格
- 若此格為敵方船艦位置，則顏色變為紅色，表示猜中
- 若此格不是地方船艦位置，則顏色變為深灰色，表示沒猜中
- 與此同時，對方的Ally Gameboard也會發生相同變化

格子顏色變化過程

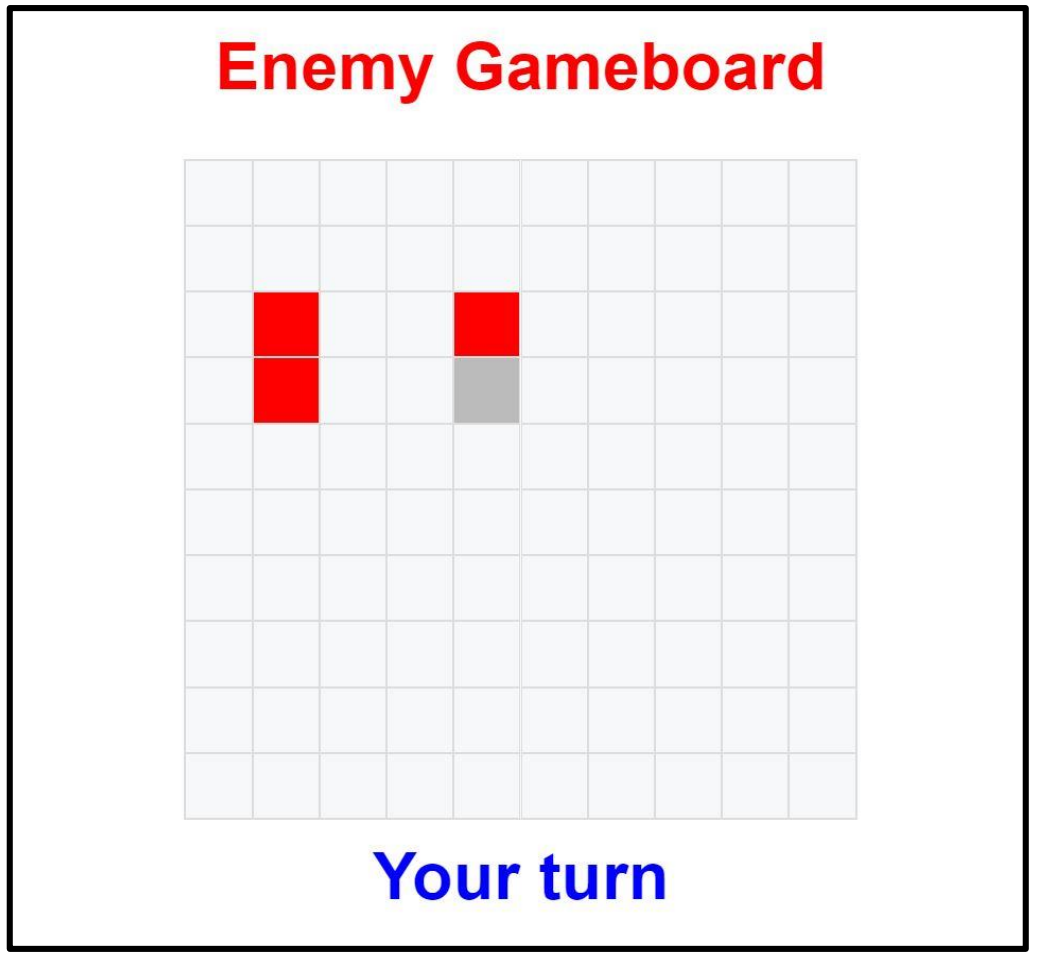

Enemy Gameboard
Your turn
Enemy Gameboard
Your turn
等待接收對方回傳資料
沒猜中
猜中
Enemy Gameboard
Your turn
Enemy Gameboard
Your turn

（範例）此時為Player2的回合

紅色：Player2過去猜中Player1的船艦位置
灰色：Player2過去沒猜中Player1的船艦位置
藍色：Player1的船艦位置



Player1視角



Player2視角

勝利畫面

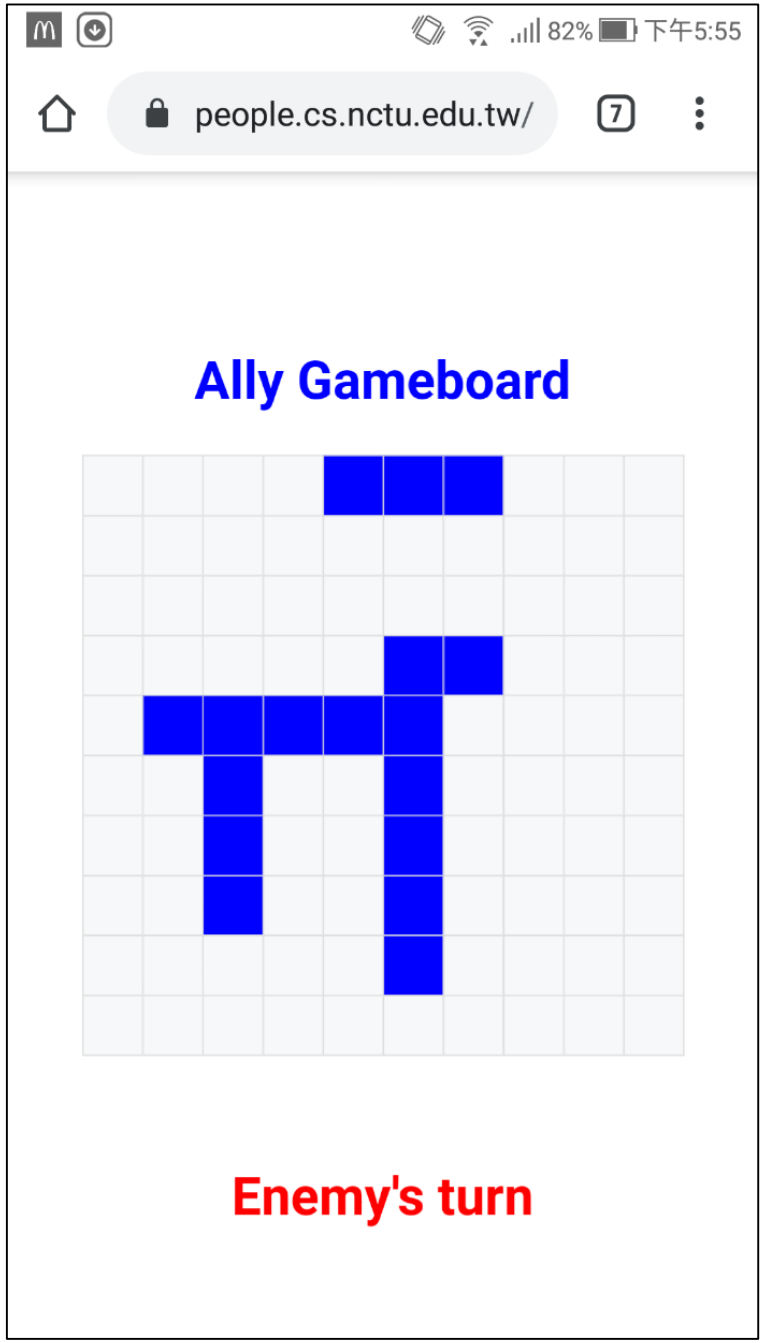

people.cs.nctu.edu.tw says
All enemy battleships have been defeated! You win!
OK
Game Over. You Win!

落敗畫面


people.cs.nctu.edu.tw says
All ally battleships have been defeated! You lose.
OK
Game Over. You Lose.

因為是網頁遊戲
在手機上也可以玩喔!

https://people.cs.nctu.edu.tw/~hywu1337/battleship/



結論

由於IoTtalk在傳輸資料時的延遲長達一秒之多，因此很難實現即時性、節奏上較為快速的遊戲。例如我們的前一個主題：小朋友下樓梯，雖然製作成功，但延遲導致操作十分不順手，遊玩過程無法感受到樂趣，讓我們不得不放棄real-time類型的遊戲，另尋他路。

而像Battleship這種非即時性的遊戲就可以很完整的還原出遊戲的原貌，能透過IoTtalk將單人玩法改成多人對戰，並且絲毫不減樂趣。