基於物聯網之實境應用設計
期末報告

組別：　鴿子襯衫

組長：　廖其忻 **F74045018**

組員：　洪正皇 **C14031162**

尤辰因 **F74046399**

李佳勳 **F74046404**

積分戰應對策略

/演算法

第一戰（LoopSolver）：

廖其忻：

這是分完組別的第一戰，經過討論過後，我們組決定將我們所有的車車改成有同樣的車型，以配合我們過後的策略，在我們都組完車車過後，我嘗試用記錄路徑的方式推算迴圈讓車車可以在迴圈里跑過一圈後（轉了 3 次彎）再找到正確的出口，可是當時的車體設計的不好（太小，圓形，導致與墻壁的距離太大，車車會以為有路而轉彎），所以導致車子經常錯誤偵測轉彎，，而且比賽當天，其他兩個組員的車車突然遇到不會懂的問題，所以改用最簡單的靠左墻法則來寫車車偵測判定的演算法。

檢討：車型很重要，不能太小，不建議圓形車，中心不能太高

洪正皇：

期中考後發現車體形狀過大不太有利，改為小型、圓形之車體，期望對稱的小車型可以增加轉彎靈活性，但離第一積分戰時間間隔不足，以變數紀錄走過多少個彎了，以及在第幾個彎的時候應當往哪個方向轉彎。

檢討：寫太死一次出錯就得整個重來，風險太大，每次上路的結果都很容易不同。

尤辰因：

第一站準備得很匆忙，花了許多時間重新組車，但後來覺得圓形車因為超音波的距離太接近，調整有點困難，最後還是改回以方型車比賽。在 code 的部分，以變數紀錄走到 T 字路口的次數，因為是左右轉優先，因此走到第二次的 T 字就控制出迴圈，把情況寫死了。但第一戰我遇到的最大問題是車子馬達問題 (仍然不確定明確是什麼問題)，車子會一直在原地抽搐，輪子無法正常地轉導致轉彎微調都會不正確。

李佳勳：

期中驗收後，發現 PP 板治的車身有點不太牢固，且又適逢分組，就跟著全組一起將原先的方型車改為圓形車來跑，且加裝穩壓模組並使用行動電源供電，相較於期中時，電壓更穩定，比較不會出現電壓不足導致的誤差。

程式碼方面，因為時間分配不佳，導致時間不夠去準備程式碼的方面，我程式碼方面是設變數偵測轉過幾次彎，轉三次之後就不要再轉了

<檢討>算是寫死的寫法，一掛掉就要重來，而且有時候超音波判斷進去錯誤的條件，就無法正確的轉出來。風險過大。

第二戰（RelayRace）：

廖其忻：

由於第二戰只要跑ㄇ字型，而重點是學會使用 wifi 模組，所依我只是用了能右轉就右轉，遇到死路（3 面為墻）就發送訊息給中控台的演算法。可是發現有裝 WiFi 模組會令車車跟平常走得不太一樣，推測是我們的 delay 問題

在經過第一戰的車型問題，我們決定放棄之前的想法，然後自己再組一次自己認為最適合的車型，而我就使用了長方形焊板作為車身，用焊接的方式減少杜邦線的數量及掉線導致電路不穩定的問題。

洪正皇：

因為無任何岔路，此戰程式碼為極簡風格，唯獨前面無路可走時才轉彎，三面碰壁時即可傳送訊息使下一台車出發，並停在原地。

尤辰因：

使用了 wifi 模組，雖然好像有稍微影響到車子跑的情況，但因為第二戰只要跑ㄇ字型，所以 code 的部分只是簡單的以前方優先，遇到牆就轉彎。

李佳勳：

經過第一次積分戰，發現圓車其實並不如預期的好跑，所以又將車型改為方型車。因為主要這一戰的重點是連線，所以車子我就直接讓他靠牆走。進死路之後傳訊息給下一個人。Wifi 的部分一開始讓他用 broadcast 傳遞訊息，後來發現這樣大家都會一起出發，經過修正才成功讓她接力。

<檢討>因為是算時間，所以車子要儘量跑快跑穩，但我們將時間都用於改車以及 wifi 方面，時間方面分配不均。

第三戰（RFID Finder）：

廖其忻：

在第三戰中，由於迷宮再也不是精簡版的，而且要偵測的有效的 RFID 都設定在死路的條件下，所以我是用了一套比較完整的演算法，判定所有要走非直線的所有情況，由於有 3 個方向（前左右），所以有 2 個 3 次方種可能性的情況：

- 遇到一個彎時就轉彎
- 遇到兩個彎時就判斷兩邊的距離，假定比較短的那一個方向有死路（因為先偵測到牆壁）
- 因為迷宮的條件限定，所以不會遇到三個彎（十字路口）

可是我認為是 delay 的問題及車車的馬達轉動都是用 delay 來控制，所以導致 RFID 沒有辦法在每一個時間點都能有效地偵測到

洪正皇：

每次 LOOP 中都掃描一次 RFID，有左右彎時優先轉彎，遇到 T 字型岔路時，優先進入距離較短那邊(死路通常較短)，從死路迴轉出來時忽略先前走過的那條路，例如：右轉進死路，出來時忽略左邊的路。

尤辰因：

加入 RFID 一整個混亂，一開始還沒加 RFID，花了時間許多時間調整車子(因為使用電池，所以常常因為電池耗損程度不一樣，需一直調整)，但之後加入 RFID 後車子跑的狀況和原本完全不一樣，因為 RFID 多了一些 delay，轉彎也整個亂掉。後來決定當走入死路才偵測 RFID 來減少時間還有車子跑的問題。而為了提高完成度，採左右優先最後才是前方，出死路的部分分成 L 型叉路和 T 字型叉路，偵測若左右兩邊距離皆大於 30 則為 T 字型，紀錄 cross = T_cross 出叉路的方式則為先倒車使車子回到進叉路前的位置再往非死路的另一方向，而 L 型則是先倒車若剛剛是右轉入則左轉，左轉入則右轉。

李佳勳：

沿用積分第二戰的車，將叉路分為 T 字(前方沒路，左右有路)以及 L 字(前面有路，左或右有路)叉路，T 形叉路選短邊走，L 形叉路左轉進去的話就出來也左轉(反之)，RFID 感應原本是邊走邊感應，但後來他的 delay 嚴重影響到我的車子的跑的狀況，所以改成進叉路之後迴轉才會開始感應(至出叉路)。

期末戰：

廖其忻：

沿用第三戰的演算法，WIFI 模組一直測試都一切正常，可是在比賽的當下突然連不上中控台，助教說是供電問題，可是過後檢查過供電是沒有問題的，可是比第二次的時候就突然好了，不知道什麼問題。

遇到的迷宮問題，如果 4 台車車同時出發會有行走在相同路徑而相撞的問題，所以會用一片墻隔開。

洪正皇：

以第三戰的程式碼，在 20 分鐘初期先探索迷宮，探索到停車點位置後再以目測判斷左牆法則或右牆法則何者較快到達停車點位置。

檢討：沒有良好的探索地圖的演算法

尤辰因：

期末戰因為無法先得知地圖，所以 code 的部分採簡單得靠左走或靠右走，來提高地圖探索的完整度。我覺得在不知道停靠點的情況去探索地圖真的有點難，可能有時候走進死路但是不確定是 RFID 沒有偵測到還是不是停靠點，因此利用把一個動作拆成多段做，並在中間插入多個偵測來提高 RFID 偵測成功率。

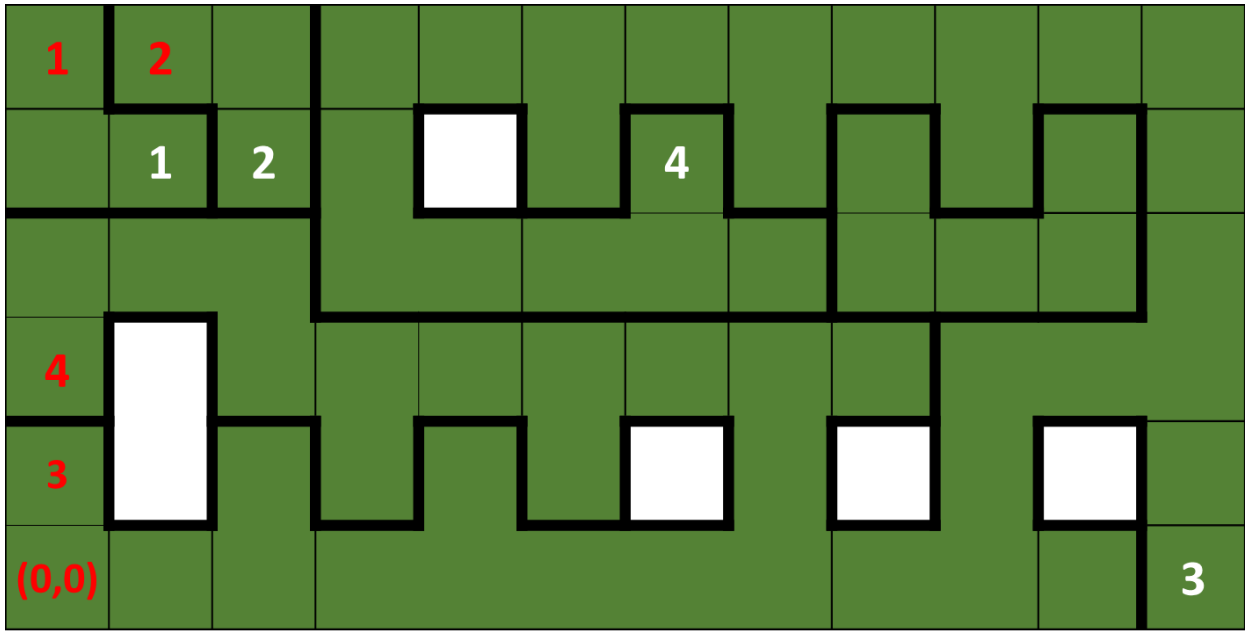
李佳勳：

為了提高迷宮探索率，第三戰我直接寫靠牆走，盡量探索完整個迷宮，找到停靠點之後用肉眼判斷是靠左靠又哪個比較快，就燒哪個 code。

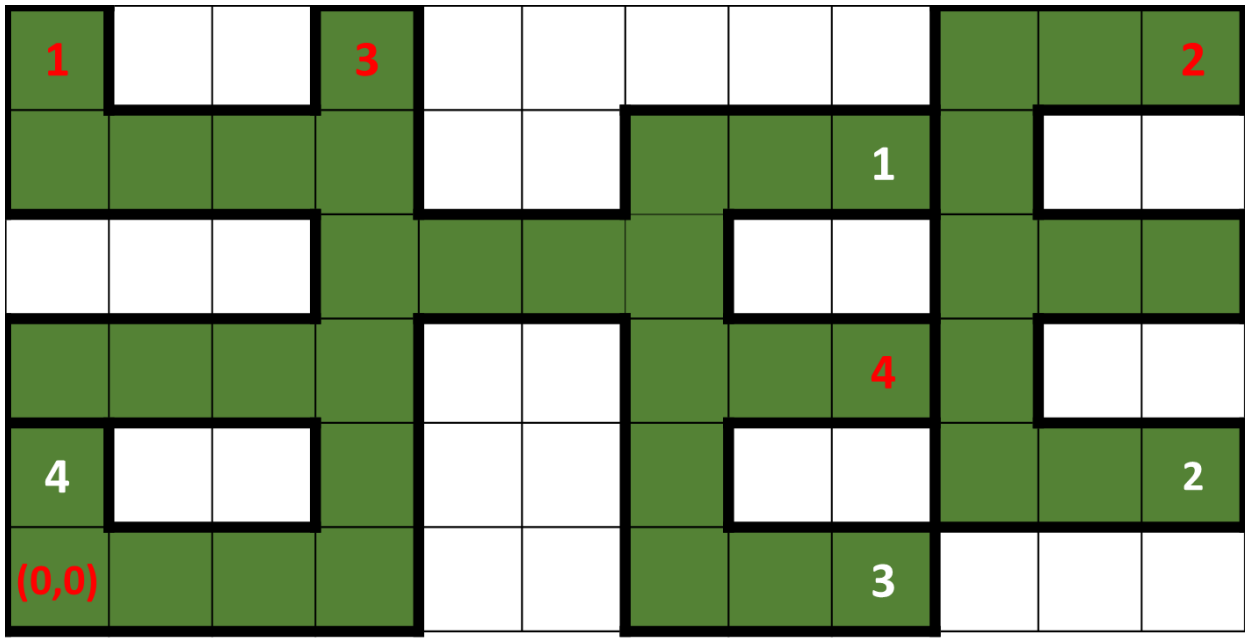
<檢討>因為不知道停靠點，有時候走進叉路不知道是沒有感應到 RFID 還是那裏不是停靠點。所以浪費很多時間，應該要將 RFID 的感應弄得在順一點。

期末戰迷宮設計想法

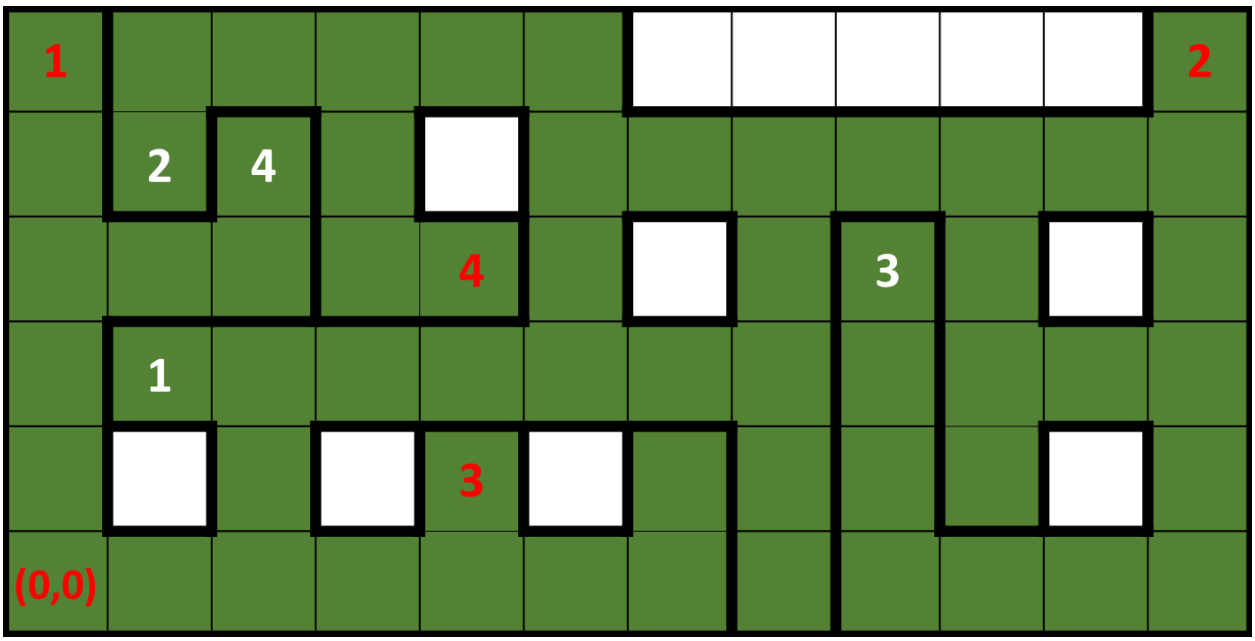
預賽一



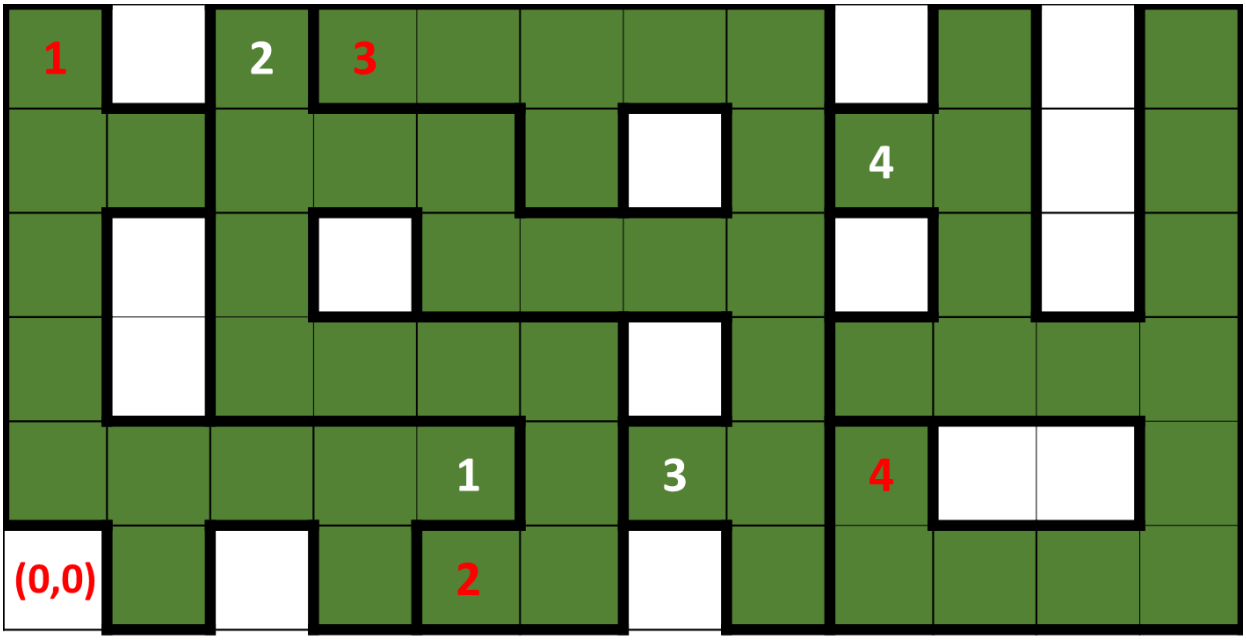
預賽二



決賽一



決賽二



在設計的過程中，

- 我們的迷宮會讓車車在同時出發的前提下有可能相撞。
- 而且我們也有設計不一樣長短的行走路徑，讓每一台車車都有著不一樣的能力下也有機會接觸到不一樣等級的迷宮。
- 迷宮會包含子迷宮
- 4 條路徑會有一些靠左比較快，有一些靠右比較快的方式來跑，並不會全部都要靠左或是全部都要靠右
- 我們盡量會使用盡可能多的格子數來設計迷宮

期末戰應對策略

(或檢討)

在事前知道了賽程表及對手的情況下，我們有做出一些分析：

- 我們車車的性能太不穩定，導致我們花費太多的時間在找出車車的硬體的問題點在哪裡
- 我們車車的演算法不一，沒有一套很好的規劃，打算自己走到自己的終點就好
- 有事前討論好，如果遇到突發狀況是的應對措施，可是因為在現場有點惶恐，所以來不及反應
- 我們遺漏了一些事前檢查車車硬體的檢查工作 導致小狀況發生（有螺絲掉了）