機器人大富翁(Robotic Monopoly)

指導教授：劉惠英 老師　　學生：葉德安、郭士楓、黃昱誠

輔仁大學 電機工程學系 大學部專題生

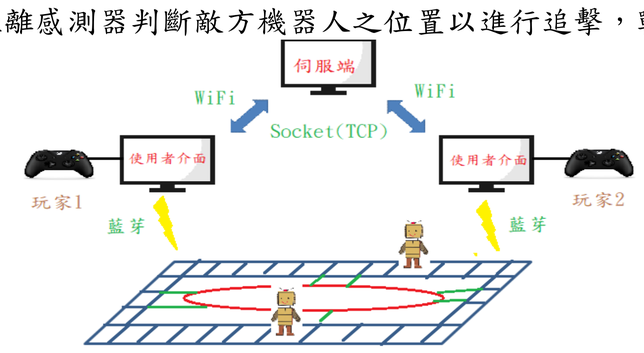
摘要

據中國科學報指出，機器人領域十項最具成長性技術展望，其中醫療、勘探、國防等領域成為重點標的。而其中一項為再造──虛擬現場──虛擬現實機器人技術，該技術可實現操作者對機器人的虛擬控制操作，在維修檢測、娛樂體驗、現場救援、軍事偵察等領域有應用價值。

·我們以娛樂體驗為出發點，但這次有別於以往大富翁，我們做出一款由程式語言與機器人結合大富翁棋盤而成的機器人大富翁遊戲。遊戲中藉由伺服端在使用者介面提供遊戲訊息，並適時在棋盤上控制機器人的行動，也可透過觸發事件來取得獎勵或是讓機器人進行戰鬥，目前遊戲主程式部分已完成，機器人也已部分合成。

系統架構

·本專題的系統架構(圖一)，我們在伺服端運作的的遊戲及使用者介面是由C++以及C#組成，進行遊戲人數為2人，伺服端與玩家是由WiFi來進行連線，再利用socket的方式，在伺服端與使用者介面互相傳遞資料。其中使用者介面是使用C#的windows form來建立，機器人的控制及訊息傳送，均是透過藍芽連線與機器人所提供的API做連結，機器人掛有顏色感測器，能取得棋盤顏色之回傳值，藉由這項功能就可在傳統大富翁棋盤上正確行走並修正路線。另外在戰鬥部分，機器人有裝上距離感測器，我們預期使用距離感測器判斷敵方機器人之位置以進行追擊，戰鬥全程都是由雙方機器人自行判。



圖一、機器人大富翁系統架構圖

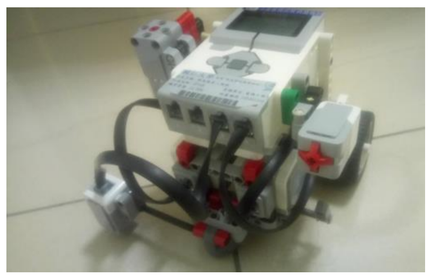
研究內容

在大富翁做為為研究背景的前提下，其本身設計主要內容包括：棋盤設計、棋子挑選及控制設定，以下為我們做過更改的研究內容

>機器人的選擇

最初我們是以樹莓派(Raspberry Pi)(圖二)作為大富翁的棋子，主要原因是程式結合不易與其零件不如EV3好取得，經討論後改為LEGO-EV3(圖三)。

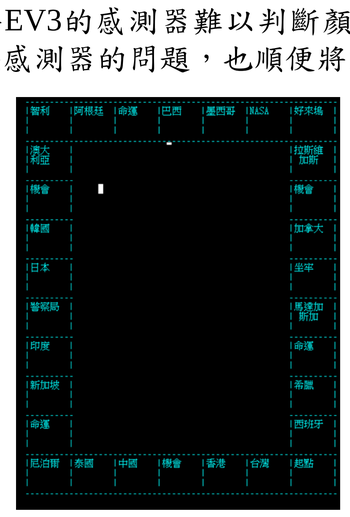
圖二、樹莓派自走車



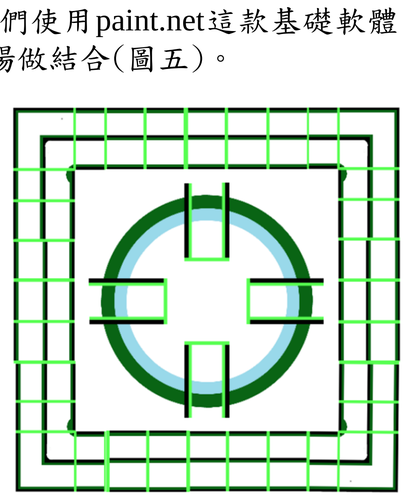
圖三、EV3自走車

>大富翁棋盤的演進

以桌遊最來說，簡單的棋盤大多分為方形，但也有少部分為圓形，甚至是不規則的，我們採用的是方形棋盤(圖四)，但因應遊戲設定，在某些觸發事件會進行戰鬥，接續開發了圓形競技場，在設計初期使用小畫家描繪，由於小畫家是以點陣的形式製作圖片，列印出棋盤後發現成果不佳，圓周上的曲線都因點陣效果變成了馬賽克，使得EV3的感測器難以判斷顏色區塊，隨後我們使用paint.net這款基礎軟體，成功解決感測器的問題，也順便將方形棋盤與競技場做結合(圖五)。



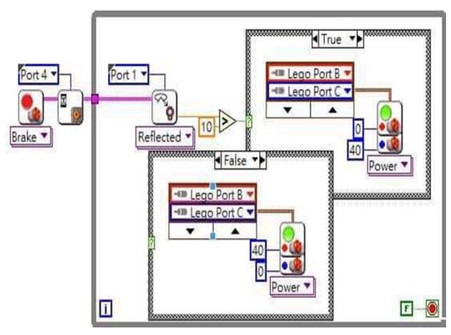
圖四、模擬方形棋盤



圖五、結合後的棋盤

>機器人感測程式

EV3使用顏色感測來取得棋盤顏色的回傳值，這部分的程式撰寫原先是以LabView(圖六)進行測試，經由方便的圖形介面建構出一套基礎程序，但因為我們是第一次接觸這款程式，我們無法解決程式對連上出現的問題，因而轉向使用C#來建立程式間的聯繫，雖然C#的控制程式(圖七)在比起LabView上顯得費工，卻得以讓我們的研究繼續進行。



圖六、LabView的程式介面

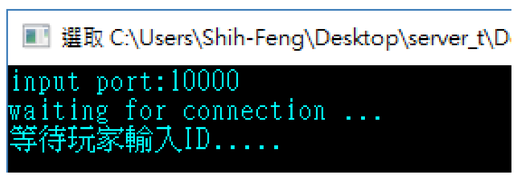
```
if (testgo < 21 && (test2_go<2 || counter_turn_right>0) && counter!=2 && game_start==1) //遊戲開始才走
{
    await _brick.DirectCommand.TurnMotorAtPowerForTimeAsync(OutputPort.B | OutputPort.C, _forward, _time, false); //靠sensor走
}
else if(testgo >= 30 && testgo<=37 && test2_go < 2 && counter != 2 && die_count==die_num)//走到骰子指定點數才停止
{
    await _brick.DirectCommand.TurnMotorAtPowerForTimeAsync(OutputPort.B | OutputPort.C, 0, _time, true); //停止
}
else if (testgo > 85 && (test2_go < 2 || counter_turn_right > 0) && counter != 2)
{
    _brick.BatchCommand.TurnMotorAtPowerForTime(OutputPort.B, _forward, 150, false); //turn left
    _brick.BatchCommand.TurnMotorAtPowerForTime(OutputPort.C, 0, 110, false);

    await _brick.BatchCommand.SendCommandAsync();
}
```

圖七、C#部分程式碼

研究成果

·開啟遊戲後，在伺服端主導遊戲程式，等待玩家進行連線(圖八)，玩家則是輸入姓名與IP位置即可加入遊戲(圖九)，在遊戲中有許多觸發事件，其一為觸發命運如下圖所示(圖十)(圖十一)。



圖八、Server等待連線

圖九、玩家進行連線

·　圖十、進行遊戲後觸發命運

圖十一、玩家獲取命運資訊

·結論

·此次專題研究總體而言只能算是半成功，完成的部分包括遊戲程式碼、使用者介面的撰寫、機器人運作及大富翁棋盤，但程式間的連結仍有問題無法克服。在研究過程中，我們也發現幾項招致失敗的原因，對此探討以下幾點

>機器人會有預測外的行動

在感測棋盤顏色的部分，因感測器太過敏感，只要燈光亮度有稍微的改變，測出來的結果使得機器人有時會提早暫停，或是拐彎越出棋盤的設置。

>對程式語言的熟練度不夠

LabView與C#在此次研究我們算是第一次接觸，在LabView方面，機器人的控制雖然容易完成，但是回傳數碼至C++程式，我們找不到合適的對連方法；而在C#方面，雖然上述兩點都得以解決，但是在建立雙人連線時，建立windows form的程序來進行資料送收，其難度使我們的研究進度嚴重落後先前的規劃。

>無適合的研究場地

因我們設定的棋盤大小約為2.5*2.5平方米，走廊只能擺放一側的棋盤，聖言樓的自由空間也因太多科系借去使用，所以找不到合適的空地進行研究。

>研究所需時間不足

在這尖峰時間點，忙於準備研究所或是報名推甄，鮮少有充分時間投入大量精力進行研究專題，剩餘時間只剩星期日來進行小組研討，以至無法如期完成整份研究。

>學校向廠商訂購的機器人遲遲拿不到

此議題推測為廠商待處理訂單過多，或是廠商將小訂單順位挪至大訂單後面，使得在專題發表前一周才如願拿齊所需零件。

未來展望

先前發現讓機器人做偵測或是照指令行走，在設計方面比起C#，Labview可用來簡化設計概念，節省時間卻又不失功能性，使用上自由度較高，但在關於程式與機器人連結的問題，C#卻比Labview簡單許多，資料傳輸也較為方便，若有時間改進演討，希望這項機器人實驗能如願完成，並在結合各程式的優點的前提下，優化機器人控制系統以達到更高的水平。

另外在機器人在戰鬥的部分，希望能夠採用機械手臂互毆的方式而不是只用相撲決定勝負，但是這次因為時間、硬體設備及對機構設計掌握不夠的關係無法在期限內達成，不過在這次的實驗中，對於機器人的控制及硬體設計已經掌握了初步概念，所以在未來的期望中我相信一定能達成期待的成果。