

國立虎尾科技大學

機械設計工程系

cd2023 2a3-pj3ag1 分組報告

網際足球對戰場景設計

Web-based Foosball Scene Design

指導教授：嚴家銘老師

班級：四設二甲

學生：王樟皓 (41023114)

 吳政憲 (41023118)

 呂承駁 (41023119) 組長

 呂昕叡 (41023120)

 李彥廷 (41023122)

 李茂廷 (41023124)

 卓桓琮 (41023126)

 林敬燦 (41023138)

中華民國

112 年 6 月

摘要

本課程將採兩人一組、四人一組與八人一組的方式進行協同機電整合產品開發，開發一款能在 web-based CoppeliaSim 場景中雙方或多方玩的遊戲產品。最後在 w16 現場發表八人協同四週後所完成的產品，在 w17 各組採 OBS + Teams 以影片發表所完成的協同產品。

課程一開始讓同學從專案一練習中，了解套件中的諸多功能以及用法，其中包括利用近接感測器偵測障礙物，並透過程式控制機器人雙輪車的移動。為了讓各組學員了解在多人協同模式下，開發機電資產品流程中必須面臨的許多議題。再接續專案一的雙輪車，改用 Python zmqRemoteAPI 進行控制，各分組需完成能在 Visualization Stream 瀏覽器中，跨網路雙方各控制一台雙輪車在足球場中進行對陣，且需設計一組能在雙方瀏覽器中進行計分的系統。最後各組需對雙輪車進行設計改良，以提升行進與對戰效能，各組需採 CAD 進行場景與多輪車零組件設計後，轉入足球場景中以鍵盤 arrow keys 與 wzas 等按鍵進行控制，對陣雙方每組將有四名輪車球員，且每兩人在同一台電腦上操作，完成後各組需在分組網站中提供所有相關檔案下載連結，且提供線上分組簡報與分組 pdf 報告連結。

專案場景必須要有感測器及記分板，讓進球後可以顯示分數在場景上，而記分板除了採用 LED 顯示計分外，也要以建立以機械轉盤傳動計分系統。另外建立計時器讓學員在對戰時得知時間剩多少，並利用程式控制球門使球重置後繼續對戰，最後在 CoppeliaSim 模擬環境中透過埠號及 [http://\[2001:288:6004:17:2023:cda:x:x\]:23020/](http://[2001:288:6004:17:2023:cda:x:x]:23020/) 進行對戰及觀看。

Abstract

This course will involve collaborative development of mechatronic integrated products in teams of two, four, and eight members. The objective is to create a web-based game product using CoppeliaSim, where two or more participants can engage in gameplay within the virtual environment. At the end of the course, during week 16, the eight-member teams will present their completed products in a live demonstration. In week 17, each group will use OBS + Teams to present a video showcasing their collaborative product.

At the beginning of the course, students will practice with Project 1 to familiarize themselves with various features and usage of the package. This includes utilizing proximity sensors to detect obstacles and controlling the movement of a robot two-wheeled car through programming. In order for each group of students to understand the many issues faced in the development of electromechanical products in a multi-user collaborative mode, we will continue with the two-wheeled car from Project 1 and switch to Python zmqRemoteAPI for control. Each group is required to develop a system where two teams can control their respective two-wheeled cars in a soccer field, engaging in a match through cross-network control using Visualization Stream in a web browser. Additionally, each group needs to design a scoring system that can keep track of scores within the web browsers of both teams. Finally, each group must design improvements for the two-wheeled car to enhance its movement and performance during matches. Using CAD software, each group will design the scene and components of the multi-wheeled car. The control will then be transferred to the soccer field using keyboard arrow keys and other designated keys such as 'w', 'z', 'a', and 's'. Each group will have four car players, and two members will operate on the same computer. After completion, each group is required to provide download links for all relevant files on the group website, along with links to online group presentations and PDF reports.

The project scenario requires the presence of sensors and a scoreboard in the simulation environment. The scoreboard should display the score on the scene, indicating goals scored. Apart from using LED displays to show the score, a mechanical rotary-driven scoring system should also be implemented. Additionally, a timer needs to be created to inform the participants about the remaining time during the gameplay. The program should control the goal posts to reset the ball and continue the game. Finally, the participants can engage in the match and observe it through the CoppeliaSim simulation environment using the port address

[http://\[2001:288:6004:17:2023:cda:x:x\]:23020/](http://[2001:288:6004:17:2023:cda:x:x]:23020/). (Note: Please replace "x" in the provided address with the appropriate values or specific information.)

目 錄

摘要.....	i
Abstract	ii
第一章 前言	1
1.1 專案概述與目標	1
1.2 規則.....	1
第二章 專題設計	2
2.1 尺寸規定.....	2
2.2 建立球員.....	2
2.3 建立記分板.....	3
2.4 建立計時器.....	5
2.5 建立球場.....	5
第三章 程式碼說明	6
3.1 控制機器人程式	6
3.2 記分板程式.....	8
第四章 場景模擬	18
4.1 摘要.....	18
4.2 統整場景.....	18
4.3 CoppeliaSim	19

第五章 組員連線	21
5.1 摘要	21
5.2 連線說明-防火牆	21
5.3 連線說明-IPv6	22
第六章 討論與分工	24
6.1 分工	24
6.2 討論紀錄	24
第七章 心得	26
第八章 參考文獻	27

圖 目 錄

圖 1.1 專案目標	1
圖 2.1 第一版車體	2
圖 2.2 車體導入場景	3
圖 2.3 記分板建立	3
圖 2.4 初版記分板	4
圖 2.5 匯入記分板	4
圖 2.6 匯入第二版記分板	4
圖 2.7 匯入記時器	5
圖 2.8 球場繪製	5
圖 2.9 導入球場	6
圖 3.1 控制機器人程式之一	6
圖 3.2 控制機器人程式之二	7
圖 3.3 LED 記分板程式之一	8
圖 3.4 LED 記分板程式之二	9
圖 3.5 LED 記分板程式之三	10
圖 3.6 LED 記分板程式之四	11
圖 3.7 LED 記分板程式之五	12
圖 3.8 LED 記分板程式之六	13

圖 3.9 機械式記分板程式之一	14
圖 3.10 機械式記分板程式之二	15
圖 3.11 計時器程式之一	16
圖 3.12 計時器程式之二	17
圖 3.13 計時器程式之三	18
圖 3.14 計時器程式之四	19
圖 4.1 球場建立	18
圖 4.2 球場全貌	19
圖 5.1 控制台連接埠	21
圖 5.2 網路 IPv6 位置	22
圖 5.3 組長 IPv6	23

第一章 前言

1.1 專案概述與目標

本課程專案目標需要有場景與多輪車零組件設計、控制程式、開會紀錄與逐字稿、各組員任務分配與執行過程影片及分組報告 pdf 檔案，最後在 w16 現場發表八人協同四週後所完成的產品，在 w17 各組採 OBS + Teams 以影片發表所完成的協同產品。

各組交付內容:

專案三場景與多輪車零組件設計 (可使用各種 CAD 系統建立, 但必須提供完整的檔案下載連結)

專案三控制程式 (以 zmqRemoteAPI Python 製作), 除可由各組員分別跨網路控制各自的球員外, 必須建立模擬啟動後, 由雙方的自動控制程式控制各自的球員後, 進行賽局。

專案三開會紀錄與逐字稿 (可利用 jit.si 或 OBS 或其他線上開會系統)

專案三各組員任務分配與執行過程影片 (可置於 Youtube 或 Onedrive)

專案三網站包括所有協同設計流程所衍生的檔案下載連結, 各檔案必須設法壓縮在 30 MB 內並置於網站downloads 目錄中。

專案三線上簡報檔案

專案三分組報告 pdf 檔案

圖. 1.1: 專案目標

1.2 規則

本專題設計理想為一款足球遊戲，比賽一開始球會置於場中央，遊戲開始後雙方即可鍵盤操控機器人，透過隊友間的傳球並將球送至球門即可得分。

遊戲規則如下：

1. 球送至敵方球門即得一分。
2. 時間內進球數多的一方獲勝。
3. 球進入球框後會回到原位。
4. 球出界後會回到原位。

第二章 專題設計

2.1 尺寸規定

在 <https://mde.tw/cd2023/content/pj3.html> 中規定球場及球員的大小及重量。

1. 足球規格：球為白色、直徑 0.1m、重量 0.5kg
2. 足球場地：長 4m x 寬 2.5m
3. 球門規格：長 0.6m, 高 0.3m, 寬 0.1m
4. 球員尺寸範圍：長寬高各 0.2m, 重量 5kg。

2.2 建立球員

本組使用 CoppeliaSim 內的 primitive shape 來製作車子，之所以使用簡單的形狀來製作車子是因為在模擬時車子細節太複雜會導致模擬運行速度變慢。由於初始的機器人為球型會導致在碰撞時容易翻倒，所以經過本組討論後在後續車體改良中更改為長方體使球員不容易翻倒。

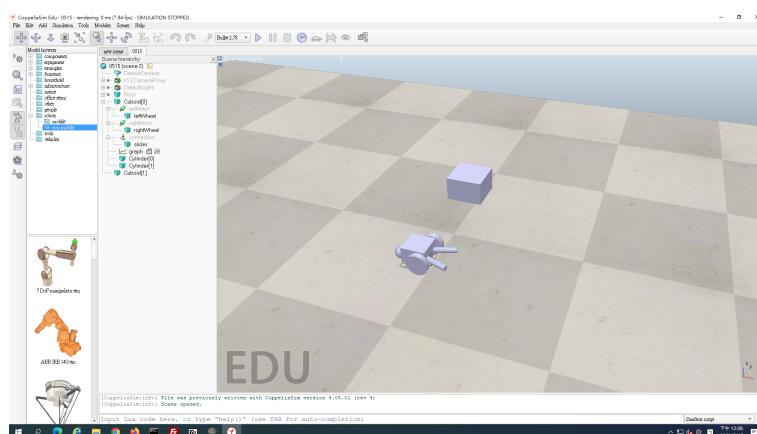


圖. 2.1: 第一版車體

在執行控制車子程式時在移動左右轉彎時馬達產生偏移，後來發現是馬達座標設定錯誤直接而修改了定位，並且加上顏色及背號。

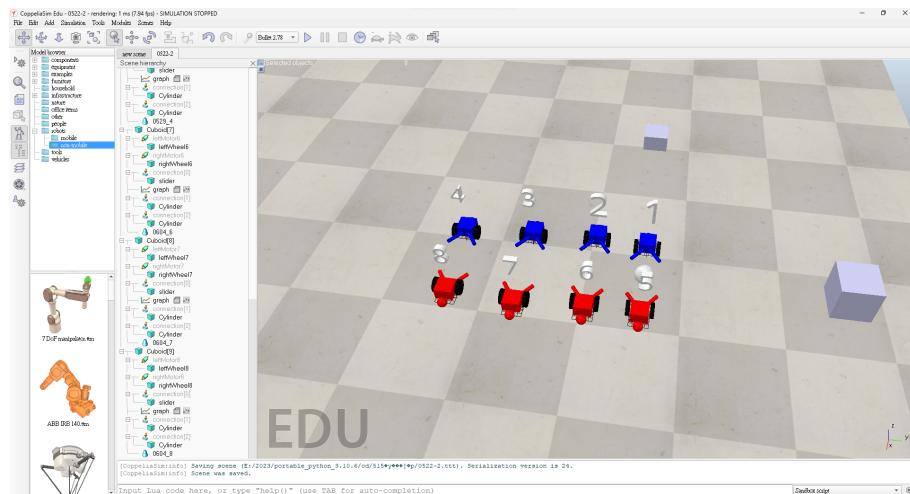


圖. 2.2: 車體導入場景

2.3 建立記分板

本課程規定除了採用 LED 顯示計分外，也要建立機械轉盤傳動計分系統。

LED 顯示計分是採用 NX 繪製出 stl 檔再導入 CoppeliaSim 中而程式部分後續會說明。

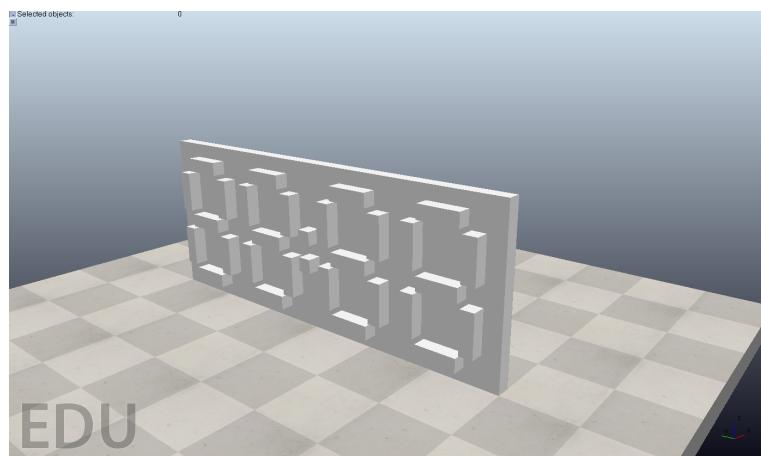


圖. 2.3: 記分板建立

機械轉盤傳動計分版是利用 onshape 繪製而成，並更改了顏色，但後續程式無法編譯出來而參考 pj3ag4 的機械式記分板。

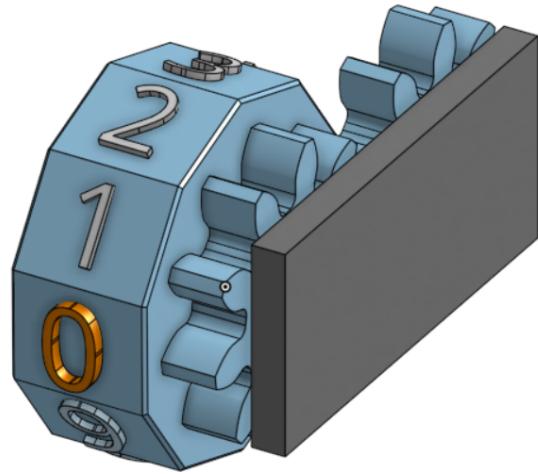


圖. 2.4: 初版記分板
初版記分板概念是由馬達帶動齒輪 10 齒輪傳動

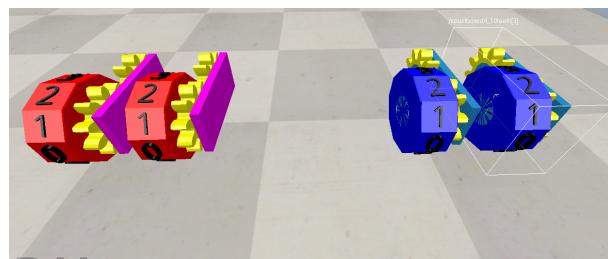


圖. 2.5: 匯入記分板

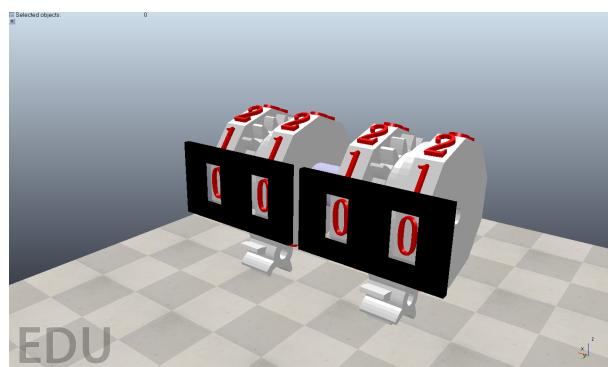


圖. 2.6: 匯入第二版記分板
最終版記分板是由齒輪組傳動 20 齒及 10 齒和二階傳動輪組成

詳細的記分板組成及製作過程可以參考本組網頁：[輪盤記分板繪製](#)

2.4 建立計時器

在進行比賽時需要有計時器讓參賽者得知比賽還多久結束，而計時器模型則是沿用記分板之檔案。

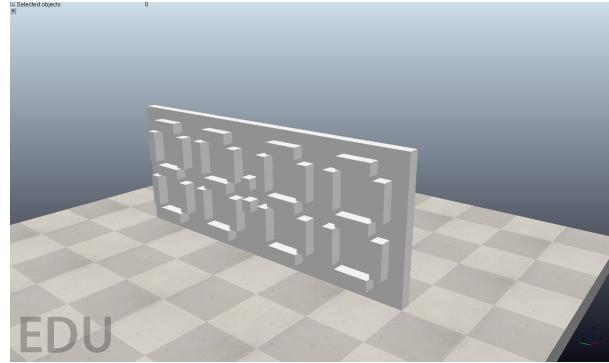


圖. 2.7: 匯入計時器

2.5 建立球場

我們使用 Onshape 繪製了球場底板及球門，匯入 CoppeliaSim 後更改了顏色。

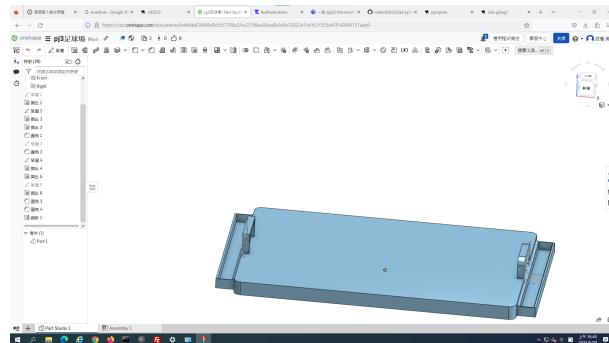


圖. 2.8: 球場繪製

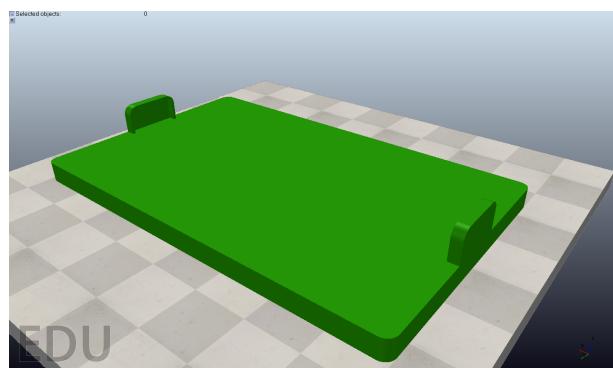


圖. 2.9: 導入球場

第三章 程式碼說明

3.1 控制機器人程式

此控制機器人程式利用 Python 語言。

使用 pip install pyzmq cbor keyboard 安裝了所需的套件，其中：

```
# pip install pyzmq cbor keyboard
from zmqRemoteApi import RemoteAPIClient
from zmqRemoteApi_IPv6 import RemoteAPIClient
import keyboard

client = RemoteAPIClient('192.168.56.1', 19997)

print('Program started')
sim = client.getObject('sim')

sim.startSimulation()
print('Simulation started')
```

圖. 3.1: 控制機器人程式之一

- pyzmq 用於建立 ZeroMQ 連線。
- cbor 用於將資料序列化和反序列化。
- keyboard 用於操控鍵盤事件。

使用 zmqRemoteApi 中的 IPv6 導入了用於建立與 CoppeliaSim 之間通訊的 Remote API 相關程式庫，使得可以通過程式碼控制仿真場景和物件，建立了一個 RemoteAPIClient 物件 client，並將 IP 和埠號分別作為 CoppeliaSim 的 IP 地址和連接埠。

這樣就建立了與 CoppeliaSim 的連線，使用 client.getObject('sim') 獲取了 CoppeliaSim 中的 sim 物件，該物件代表了整個仿真環境。透過這個物件，可以執行相關的仿真操作，再來透過 sim.startSimulation() 開始模擬。

```

# use keyboard to move BubbleRob
while True:
    if keyboard.is_pressed('up'):
        setBubbleRobVelocity(1.0, 1.0)
    elif keyboard.is_pressed('down'):
        setBubbleRobVelocity(-1.0, -1.0)
    elif keyboard.is_pressed('left'):
        setBubbleRobVelocity(-1.0, 1.0)
    elif keyboard.is_pressed('right'):
        setBubbleRobVelocity(1.0, -1.0)
    elif keyboard.is_pressed('q'):
        # stop simulation
        sim.stopSimulation()
    else:
        setBubbleRobVelocity(0.0, 0.0)

```

圖. 3.2: 控制機器人程式之二

這段程式碼是一個無窮迴圈，用於持續感測鍵盤並根據按鍵的狀態來控制機器人的運動。

如果按下'up' 鍵，則呼叫 setBubbleRobVelocity(1.0, 1.0)，將機器人的速度設定為正向。

如果按下'down' 鍵，則呼叫 setBubbleRobVelocity(-1.0, -1.0)，將機器人的速度設定為反向。

如果按下'left' 鍵，則呼叫 setBubbleRobVelocity(-1.0, 1.0)，將機器人的速度設定為左轉。

如果按下'right' 鍵，則呼叫 setBubbleRobVelocity(1.0, -1.0)，將機器人的速度設定為右轉。

如果按下'q' 鍵，則停止仿真。若沒有按下上述任何按鍵，則呼叫 setBubbleRobVelocity(0.0, 0.0)，將機器人的速度設定為零，即停止移動。

3.2 記分板程式

此程式是使用 Lua 語言。

設定名稱為 sysCall init 的函數，在程式初始化時被呼叫。該函數的主要

```
function sysCall_init()
    scorewallb = 0
    scorewalla = 0
    scorewalld = 0
    scorewallc = 0
    sensor = sim.getObject('./sensor')
    sensor2 = sim.getObject('./sensor2')
    sensor3 = sim.getObject('./sensor3')
    sensor4 = sim.getObject('./sensor4')
    sensor5 = sim.getObject('./sensor5')
    sensor6 = sim.getObject('./sensor6')
    ball = sim.getObject('/shape')
    initialballPosition = sim.getObjectPosition(ball, -1)
    initialballOrientation = sim.getObjectOrientation(ball, -1)
```

圖. 3.3: LED 記分板程式之一

目的是初始化一些變數並取得物件的句柄。

1. scorewallb = 0、scorewalla = 0、scorewalld = 0、scorewallc = 0 用於初始化四個得分牆的變數，將初始值設為 0。
2. sim.getObject 獲取了各感測器及球的句柄。
3. initialballPosition = sim.getObjectPosition(ball, -1)：用於取得球體的初始位置。
4. initialballOrientation = sim.getObjectOrientation(ball, -1)：用於取得球體的初始方向。

```
score0={1,1,1,0,1,1,1}
score1={0,0,1,0,0,1,0}
score2={1,0,1,1,1,0,1}
score3={1,0,1,1,0,1,1}
score4={0,1,1,1,0,1,0}
score5={1,1,0,1,0,1,1}
score6={1,1,0,1,1,1,1}
score7={1,0,1,0,0,1,0}
score8={1,1,1,1,1,1,1}
score9={1,1,1,1,0,1,1}
score={score0,score1,score2,score3,score4,score5,score6,score7,score8,score9}
```

圖. 3.4: LED 記分板程式之二

定義了一個包含十個元素的表格，每個元素都是由七個二進制數字(0 或 1)組成的列表。每個列表代表一個數字(0 到 9)的數字模式，而最後一行程式碼將這些數字模式放入一個名為 score 的表格中，以便在程式中進行使用。

```

for j = 0,6,1 do
    local scorea = sim.getObject('./scorea[..j..]')
    if (score[1][j+1]==1) then
        sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
    else
        sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
    end
end

for j = 0,6,1 do
    local scoreb = sim.getObject('./scoreb[..j..]')
    if (score[1][j+1]==1) then
        sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
    else
        sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
    end
end

for j = 0,6,1 do
    local scorec = sim.getObject('./scorec[..j..]')
    if (score[1][j+1]==1) then
        sim.setShapeColor(scorec, nil, sim.colorcomponent_ambient_diffuse, {0, 0, 1})
    else
        sim.setShapeColor(scorec, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
    end
end

for j = 0,6,1 do
    local scored = sim.getObject('./scored[..j..]')
    if (score[1][j+1]==1) then
        sim.setShapeColor(scored, nil, sim.colorcomponent_ambient_diffuse, {0, 0, 1})
    else
        sim.setShapeColor(scored, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
    end
end
end

```

圖. 3.5: LED 記分板程式之三

這段程式碼是一個迴圈，用於對記分板進行顏色設定，根據事先定義的數字模式進行，迴圈的運行範圍是從 0 到 6 每次遞增 1，在迴圈的每次迭代中，會執行以下操作：

1. scorewallb = 0、scorewalla = 0、scorewalld = 0、scorewallc = 0 用於初始化四個得分牆的變數，將初始值設為 0。
2. 創建一個指向記分板的參考是由迴圈變量 j 指定。
3. 檢查表格中的特定位置如果該位置的值等於 1，則使用 sim.setShapeColor 函數設定記分板數字的顏色為紅色。
4. 如果該位置的值不等於 1，使用 sim.setShapeColor 函數設定記分板數字的顏色為白色。

```

function sysCall_actuation()
    result = sim.readProximitySensor(sensor)
    if scorewallb < 10 then
        if result > 0 then
            score2 = scorewallb + 1
            for i = 0, 9, 1 do
                if score2 == i then
                    for j = 0, 6, 1 do
                        local scoreb = sim.getObject('./scoreb[' .. j .. ']')
                        if score[i + 1][j + 1] == 1 then
                            sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
                        else
                            sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
                        end
                    end
                end
            end
            scorewallb = score2
        sim.setObjectPosition(ball, -1, initialballPosition)
        sim.setObjectOrientation(ball, -1, initialballOrientation)
    end
end

```

圖 . 3.6: LED 記分板程式之四

在此函式中，首先使用 `sim.readProximitySensor` 函式讀取 sensor 的近接傳感器數值，並將結果存儲在 `result` 變數中，如果 `scorewallb` 小於 10，且如果 `result` 大於 0，表示與某個物體接觸。則執行以下操作：

1. 將 `score2` 設定為 `scorewallb` 加 1。
2. 進行一個迴圈從 0 到 9，每次遞增 1。
3. 如果 `score2` 等於 `i` 進行另一個迴圈從 0 到 6 每次遞增 1，創建一個指向 `scoreb` 的參考由迴圈 `j` 變量檢查 `score` 表格中的特定位置，如果該位置的值等於 1 使用 `sim.setShapeColor` 函式設定 `scoreb` 物體的顏色為紅色，如果該位置的值不等於 1 使用 `sim.setShapeColor` 函式設定 `scoreb` 物體的顏色為白色。
4. 更新 `scorewallb` 的值為 `score2`。
5. 使用 `sim.setObjectPosition` 函式將 `ball` 物體的位置設定為初始位置。
6. 使用 `sim.setObjectOrientation` 函式將 `ball` 物體的方向設定為初始方向。

```

scorewallb = 0
for j = 0, 6, 1 do
    local scoreb = sim.getObject('./scoreb[' .. j .. ']')
    if score0[j + 1] == 1 then
        sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
    else
        sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
    end
end
score3 = scorewalla + 1 -- scorewalla?1
for i = 0, 9, 1 do
    if score3 == i then
        for j = 0, 6, 1 do
            local scorea = sim.getObject('./scorea[' .. j .. ']')
            if score[i + 1][j + 1] == 1 then
                sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
            else
                sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
            end
        end
    end
    scorewalla = score3
    sim.setObjectPosition(ball, -1, initialballPosition)
    sim.setObjectOrientation(ball, -1, initialballOrientation)
end
if scorewallb == 9 and scorewalla == 9 then
    sim.pauseSimulation()
end

```

圖. 3.7: LED 記分板程式之五

反之如果 scorewallb 大於 9 則執行以下操作：

1. 將 scorewallb 的值設定為 0。
2. 進行一個迴圈從 0 到 6，每次遞增 1 創建一個指向 scoreb 物體的參考，該物體是由迴圈變量 j 指定的物體，如果該位置的值等於 1 使用 sim.setShapeColor 函式設定 scoreb 物體的顏色為紅色，如果該位置的值不等於 1 使用 sim.setShapeColor 函式設定 scoreb 物體的顏色為白色。
3. 將 score3 設定為 scorewalla 加 1。
4. 更新 scorewallb 的值為 score2。
5. 進行一個迴圈從 0 到 9，每次遞增 1，如果 score3 等於 i 進行一個迴圈從 0 到 6，每次遞增 1 創建一個指向 scoreb 物體的參考，該物體是由迴圈變量 j 指定的物體，如果該位置的值等於 1 使用 sim.setShapeColor 函式設定 scoreb 物體的顏色為紅色，如果該位置的值不等於 1 使用 sim.setShapeColor 函式設定 scoreb 物體的顏色為白色。

6. 更新 scorewalla 的值為 score3。
7. 使用 sim.setObjectPosition 函式將 ball 物體的位置設定為初始位置。
8. 使用 sim.setObjectOrientation 函式將 ball 物體的方向設定為初始方向。
9. 程式碼檢查兩個變數 scorewallb 和 scorewalla 是否都等於 9。如果兩個變數的值都等於 9 用 sim.pauseSimulation() 函式暫停仿真。

而另一隊記分板也是使用一樣的程式只是更改名稱。

```

result3 = sim.readProximitySensor(sensor3)
if result3 > 0 then
    sim.setObjectPosition(ball, -1, initialballPosition)
    sim.setObjectOrientation(ball, -1, initialballOrientation)
end

result4 = sim.readProximitySensor(sensor4)
if result4 > 0 then
    sim.setObjectPosition(ball, -1, initialballPosition)
    sim.setObjectOrientation(ball, -1, initialballOrientation)
end

result5 = sim.readProximitySensor(sensor5)
if result5 > 0 then
    sim.setObjectPosition(ball, -1, initialballPosition)
    sim.setObjectOrientation(ball, -1, initialballOrientation)
end

result6 = sim.readProximitySensor(sensor6)
if result6 > 0 then
    sim.setObjectPosition(ball, -1, initialballPosition)
    sim.setObjectOrientation(ball, -1, initialballOrientation)
end

```

圖. 3.8: LED 記分板程式之六

如果感測器感測到物體則執行以下操作：

1. 使用 sim.setObjectPosition 函式將 ball 物體的位置設定為初始位置。
2. 使用 sim.setObjectOrientation 函式將 ball 物體的方向設定為初始方向。

```
function sysCall_init()

    sensor1 = sim.getObject('/sensor')
    sensor2 = sim.getObject('/sensor2')
    a1=sim.getObject('/scorebord/board_a/joint[1]')
    a2=sim.getJointTargetPosition(a1)
    b1=sim.getObject('/scorebord/board_b/joint[1]')
    b2=sim.getJointTargetPosition(b1)

end
```

圖 . 3.9: 機械式記分板程式之一

這段程式碼是命名 sysCall init 為函數，在程式初始化時，該函數被呼叫。

1. sim.getObject：獲取的物件的句柄並將其存儲在變數中。
2. sim.getJointTargetPosition：獲取變數的目標位置並將其存儲在另一個變數中。

```

function sysCall_actuation()
    sensor1 = sim.getObject('/sensor1')
    sensor2 = sim.getObject('/sensor2')
    result1=sim.readProximitySensor(sensor1)
    result2=sim.readProximitySensor(sensor2)

    if (result1>0)then
        a3=a2+36*math.pi/180
        sim.setJointTargetPosition(a1,a3)
        a2=a3
    elseif (result2>0)then
        b3=b2+36*math.pi/180
        sim.setJointTargetPosition(b1,b3)
        b2=b3
    end

end

```

圖. 3.10: 機械式記分板程式之二

這段程式碼將函數用於控制行為。

1. `sim.getObject`：獲取的物件的句柄並將其存儲在變數中。
2. `sim.readProximitySensor`：讀取感測器的數據，並將結果存儲在變數中。
3. 如果 `result1` 大於 0，表示感測器偵測到物體，則執行以下操作：
4. 計算新的目標位置 `a3`，為 `a2` 加上 36 度 (`36*math.pi/180`) 的角度。
5. 使用 `sim.setJointTargetPosition` 將 `a1` 的目標位置設置為 `a3`。
6. 更新變數 `a2` 為新的目標位置 `a3`。

另一個記分板程式相同只是換了個名稱。

```
function sysCall_init()  
  
    score0={1,1,1,0,1,1,1}  
    score1={0,0,1,0,0,1,0}  
    score2={1,0,1,1,1,0,1}  
    score3={1,0,1,1,0,1,1}  
    score4={0,1,1,1,0,1,0}  
    score5={1,1,0,1,0,1,1}  
    score6={1,1,0,1,1,1,1}  
    score7={1,0,1,0,0,1,0}  
    score8={1,1,1,1,1,1,1}  
    score9={1,1,1,1,0,1,1}  
  
    score={score0,score1,score2,score3,score4,score5,score6,score7,score8,score9}
```

圖. 3.11: 計時器程式之一

沿用了記分板的程式：

設定名稱為 sysCall init 的函數，在程式初始化時被呼叫。該函數的主要目的是初始化一些變數並取得物件的句柄。

定義了一個包含十個元素的表格，每個元素都是由七個二進制數字 (0 或 1) 組成的列表。每個列表代表一個數字 (0 到 9) 的數字模式，而最後一行程式碼將這些數字模式放入一個名為 score 的表格中，以便在程式中進行使用。

```

for j = 0..6..1 do
    local scorea = sim.getObject('./scorea[..j..]')
    if (score[1][j+1]==1) then
        sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
    else
        sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
    end

    for j = 0..6..1 do
        local scoreb = sim.getObject('./scoreb[..j..]')
        if (score[1][j+1]==1) then
            sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
        else
            sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
        end

        for j = 0..6..1 do
            local scorec = sim.getObject('./scorec[..j..]')
            if (score[1][j+1]==1) then
                sim.setShapeColor(scorec, nil, sim.colorcomponent_ambient_diffuse, {0, 0, 1})
            else
                sim.setShapeColor(scorec, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
            end

            for j = 0..6..1 do
                local scored = sim.getObject('./scored[..j..]')
                if (score[1][j+1]==1) then
                    sim.setShapeColor(scored, nil, sim.colorcomponent_ambient_diffuse, {0, 0, 1})
                else
                    sim.setShapeColor(scored, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
                end
            end
        end
    end
end

```

圖. 3.12: 計時器程式之二

沿用了記分板的程式：

這段程式碼是一個迴圈，用於對記分板進行顏色設定，根據事先定義的數字模式進行，迴圈的運行範圍是從 0 到 6 每次遞增 1，在迴圈的每次迭代中，會執行以下操作：

1. scorewallb = 0、scorewalla = 0、scorewalld = 0、scorewallc = 0 用於初始化四個得分牆的變數，將初始值設為 0。
2. 創建一個指向記分板的參考是由迴圈變量 j 指定。
3. 檢查表格中的特定位置如果該位置的值等於 1，則使用 sim.setShapeColor 函數設定記分板數字的顏色為紅色。
4. 如果該位置的值不等於 1，使用 sim.setShapeColor 函數設定記分板數字的顏色為白色。

```

function sysCall_actuation()
    totaltime = sim.getSimulationTime()/4
    time = 1200 - totaltime

    timemtd = math.floor((time/600)%10)
    for i = 0, 9, 1 do
        if timemtd == i then
            for j = 0, 6, 1 do
                local scorea = sim.getObject('./scorea[' .. j .. ']')
                if score[i + 1][j + 1] == 1 then
                    sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
                else
                    sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
                end
            end
        end
        timemud = math.floor((time/60)%10)
        for i = 0, 9, 1 do
            if timemud == i then
                for j = 0, 6, 1 do
                    local scoreb = sim.getObject('./scoreb[' .. j .. ']')
                    if score[i + 1][j + 1] == 1 then
                        sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
                    else
                        sim.setShapeColor(scoreb, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
                    end
                end
            end
        end
    end
    timestd = math.floor((time/10)%6)

```

圖. 3.13: 計時器程式之三

1. `totaltime = sim.getSimulationTime()/4`：計算從模擬開始到現在的總時間，並除以 4(為了調整時間的比例)。
2. `time = 1200 - totaltime`：根據總時間計算剩餘時間，1200 這部分可依照自己需求去修改。
3. `timemtd = math.floor((time/600)%10)`：根據剩餘時間計算十位數分鐘數字，`math.floor` 函數用於將計算結果向下取整，% 表示取模運算符號，這裡是將時間除以 600 取其餘數，再取該餘數的整數部分。
4. `for i = 0, 9, 1 do`：進行一個迴圈從 0 到 9，每次遞增 1。
5. `if timemtd == i then`：檢查十位數分鐘是否等於當前迴圈數字。
6. 在內部循環 `for j = 0, 6, 1 do` 中，進行顯示相關的操作。
7. `local scorea = sim.getObject('./scorea[' .. j .. ']')`：獲取記分板十位數分鐘的句柄。
8. `if score[i + 1][j + 1] == 1 then`：檢查特定位置上的數字是否為 1 (即該位置需要亮起)。
9. `sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient diffuse, 1, 0, 0)`：設置該顯示元素的顏色為紅色 (表示亮起)。
10. `sim.setShapeColor(scorea, nil, sim.colorcomponent_ambient diffuse, 1, 1, 1)`

1, 1, 1) : 設置該顯示元素的顏色為白色 (表示熄滅)。

11. timemtd = math.floor ((time/60)%600) : 分別用於計算十位數分鐘數字，並根據結果進行相應的顯示操作。

個位數分鐘與上面程式相同唯一不同的是 timemud = math.floor((time/10)%60) 用於計算個位數分鐘數字，並根據結果進行相應的顯示操作。

```
for i = 0, 9, 1 do
    if timestd == i then
        for j = 0, 6, 1 do
            local scorec = sim.getObject('./scorec[' .. j .. ']')
            if score[i + 1][j + 1] == 1 then
                sim.setShapeColor(scorec, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
            else
                sim.setShapeColor(scorec, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
            end
        end
    end
end
timesud = math.floor(time%10)
for i = 0, 9, 1 do
    if timesud == i then
        for j = 0, 6, 1 do
            local scored = sim.getObject('./scored[' .. j .. ']')
            if score[i + 1][j + 1] == 1 then
                sim.setShapeColor(scored, nil, sim.colorcomponent_ambient_diffuse, {1, 0, 0})
            else
                sim.setShapeColor(scored, nil, sim.colorcomponent_ambient_diffuse, {1, 1, 1})
            end
        end
    end
end
if time < 1 then
    sim.pauseSimulation()
end
end
```

圖. 3.14: 計時器程式之四

程式與十位、個位數分鐘相同唯一不同的是。

1. timestd = math.floor((time/10)%10) 用於計算十位數秒鐘數字，並根據結果進行相應的顯示操作。
2. timesud = math.floor(time/10) 用於計算個位數秒鐘數字，並根據結果進行相應的顯示操作。
3. 當時間小於 1 時停止模擬。

第四章 場景模擬

4.1 摘要

完成球員及程式碼設定後，接著建立模擬場景，添加場地、球員、計時器、LED 計分板和機械式轉盤記分板

4.2 統整場景

在球場周圍設置感測器球碰到會返回中心，放入 8 名球員，以兩色分為兩隊。

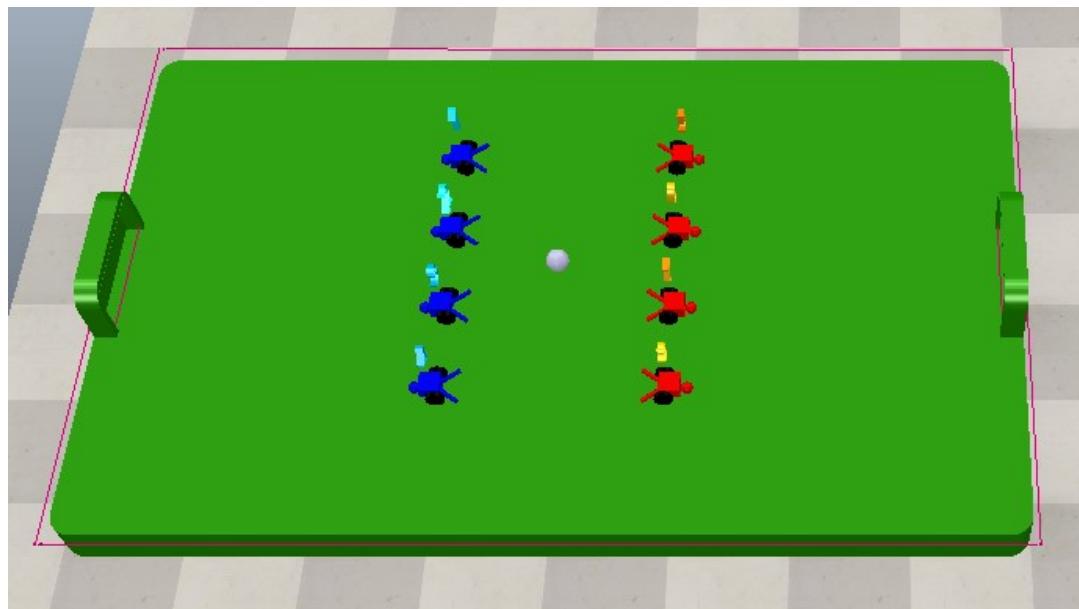


圖. 4.1: 球場建立

和計時器、LED 計分板和機械式記分板。

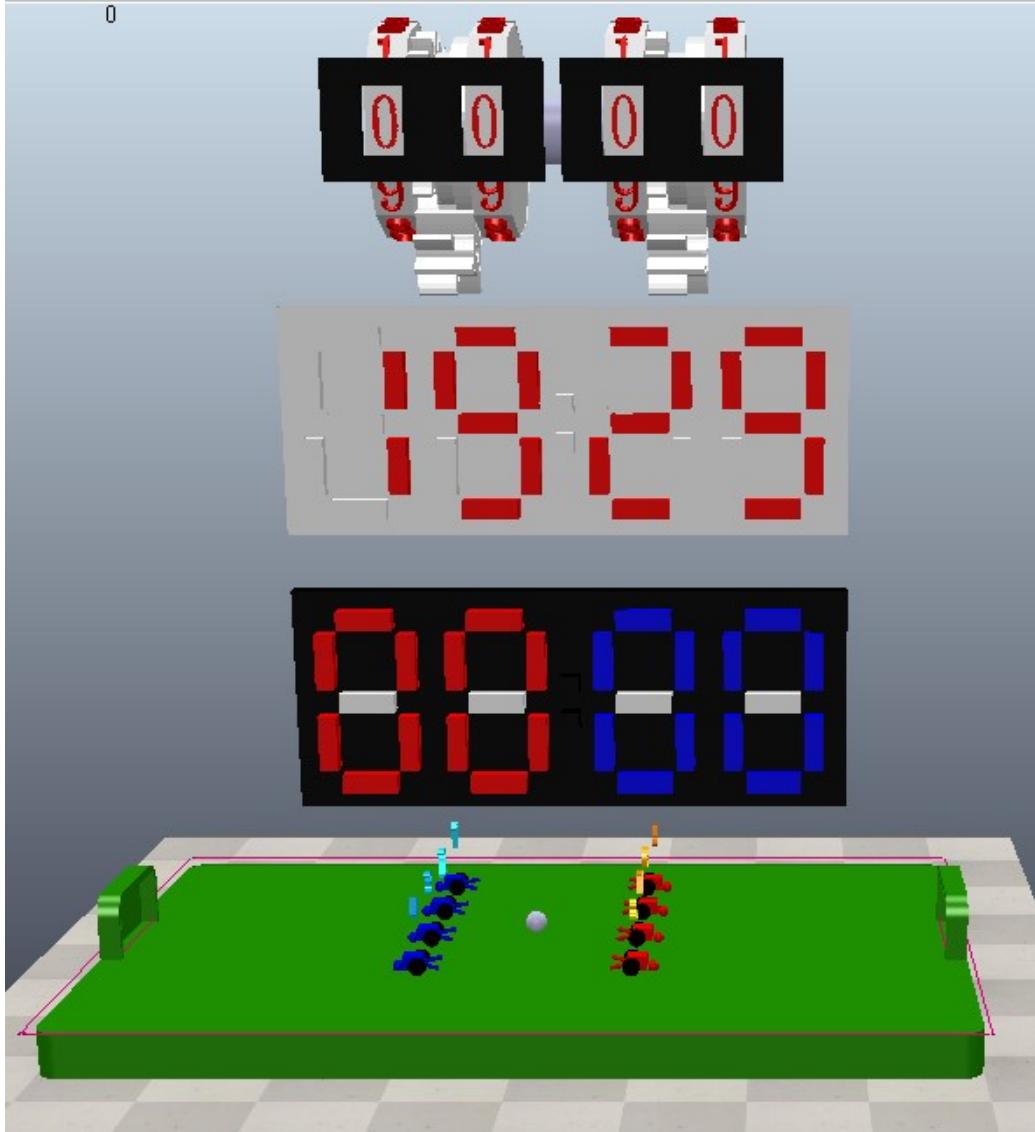


圖. 4.2: 球場全貌

4.3 CoppeliaSim

CoppeliaSim，曾被稱為 V-REP (Virtual Robot Experimentation Platform)，是一個功能強大且多用途的機器人仿真軟體。它允許使用者在虛擬環境中創建、模擬和分析機器人系統。CoppeliaSim 提供各種功能和功能，適用於各種機器人應用、研究和教育目的。機器人建模它提供了一個用戶友好的界面，可以設計和建模具有可自定義屬性（如形狀、大小和運動學）的複雜機器人。用戶可以從頭開始創建機器人，也可以導入現有的機器人模型。仿真環境：CoppeliaSim 提供一個 3D 仿真環

境，用戶可以在其中模擬機器人及其與虛擬世界的交互作用。它包括物理仿真、碰撞檢測和真實動態，以模擬真實世界的情境。傳感器仿真，CoppeliaSim 提供了一個內置的腳本界面，允許使用者使用不同的語言（包括 Lua、Python 和 MATLAB）編寫和控制機器人。這使得使用者可以實現複雜的控制算法並在仿真環境中進行測試。多機器人系統：它支持多機器人系統的仿真，使使用者能夠設計和研究多個機器人之間的協作行為、集群機器人和協調策略。遠程 API：CoppeliaSim 提供一個遠程 API，允許外部應用程序與仿真進行實時通信和控制。這一功能可用於將仿真與外部硬體（如機器人控制器或人工智能算法）

第五章 組員連線

5.1 摘要

完成場景建設後，要實施跨電腦連線對戰，使用 zmqRemoteAPI 寫的程式和下載 CoppeliaSim(4.5.1) 支援 IPv6 版本，zmq 中也需具備 IPv6 環境，然後由組長開起場景，組員跨網路控制各自的編號球員開始對戰。

5.2 連線說明-防火牆

從控制台將防火牆都關閉，點開進階設定，組長設定"輸入規則"、組員設定"輸出規則"。新增規則 / 連接埠 / TCP(傳輸控制協定 Port) / 特定連接埠 23000-23050，選擇允許連線。

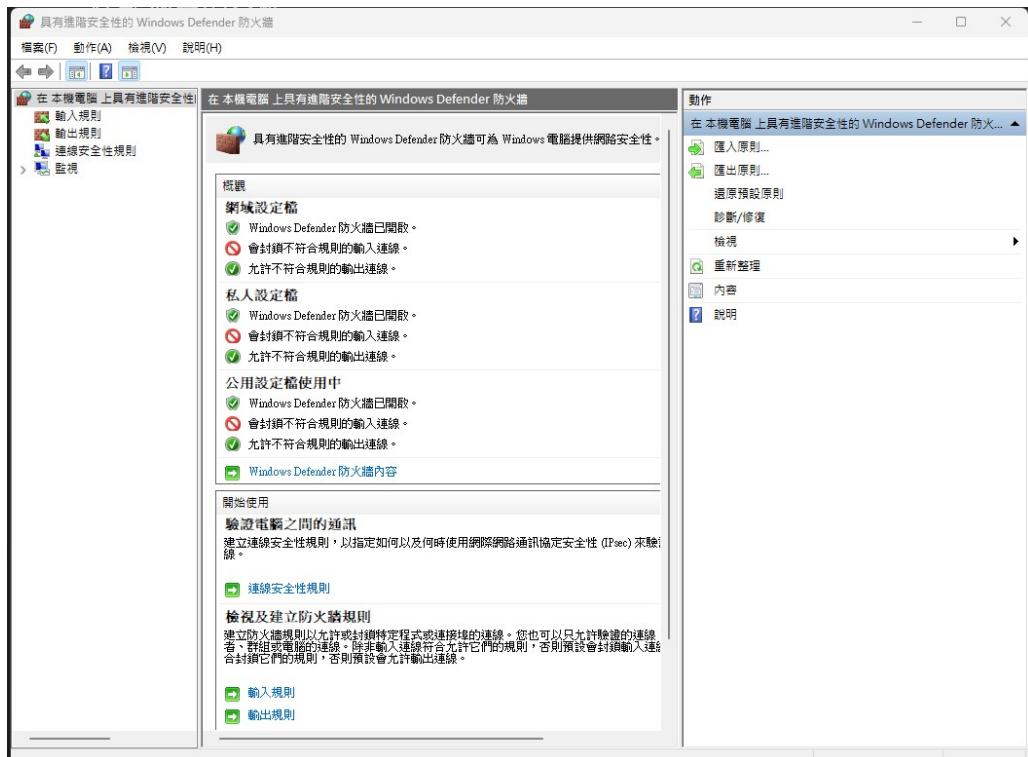


圖. 5.1: 控制台連接埠

5.3 連線說明-IPv6

設定網路 IPv6 位址，而 cda 後方的第一個冒號為各自組別，第二個為各自組員順序

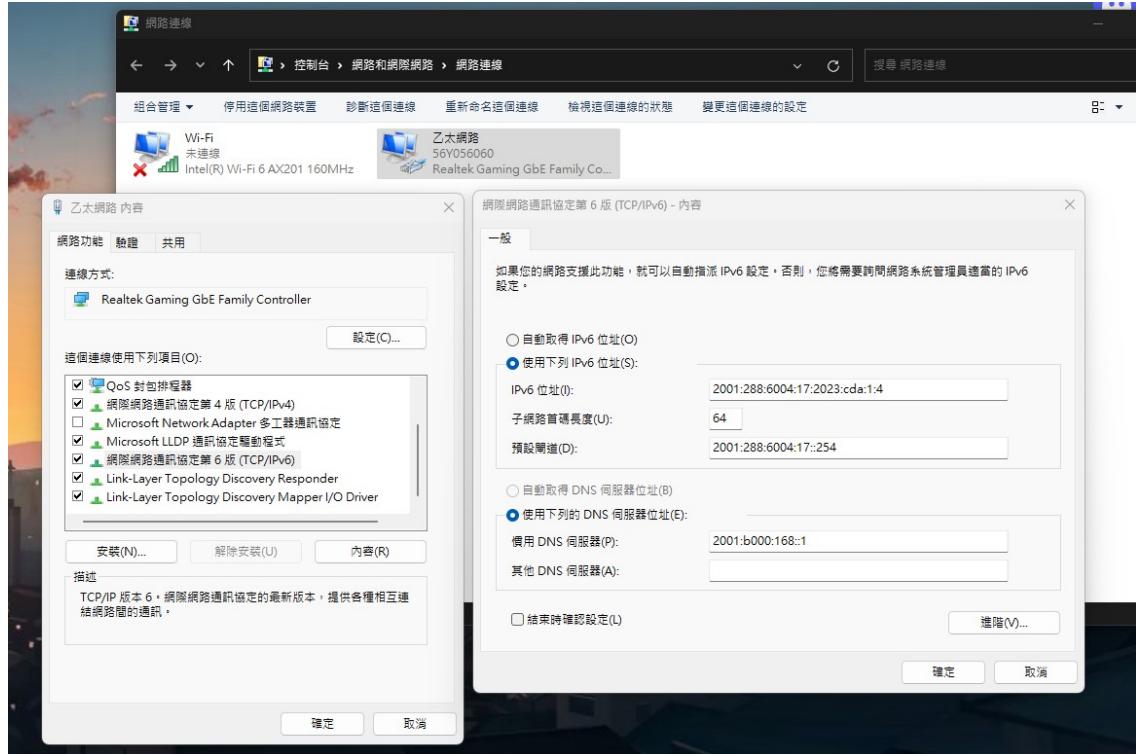
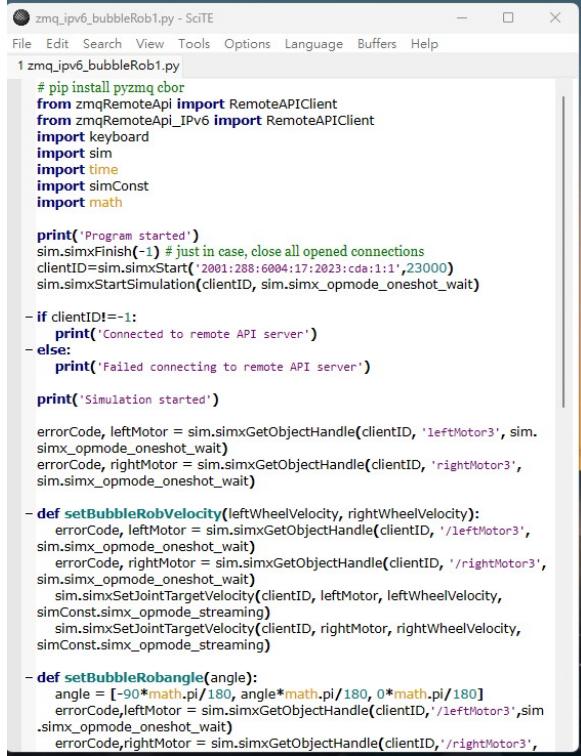


圖. 5.2: 網路 IPv6 位置

接著在 zmq 的 localhost 處打上組長的 IPv6 位置連線，且按下 go 之後得在右邊輸入 pip install keyboard。



```
zmq_ipv6_bubbleRob1.py - ScITE
File Edit Search View Tools Options Language Buffers Help
1 zmq_ipv6_bubbleRob1.py
# pip install pyzmq cbor
from zmqRemoteApi import RemoteAPIClient
from zmqRemoteApi_IPv6 import RemoteAPIClient
import keyboard
import sim
import time
import simConst
import math

print('Program started')
sim.simxFinish(-1) # just in case, close all opened connections
clientID=sim.simxStart('2001:288:6004:17:2023:cda:1:1',23000)
sim.simxStartSimulation(clientID,sim.simx_opmode_oneshot_wait)

if clientID==1:
    print('Connected to remote API server')
else:
    print('Failed connecting to remote API server')

print('Simulation started')

errorCode, leftMotor = sim.simxGetObjectHandle(clientID, 'leftMotor3', sim.
simx_opmode_oneshot_wait)
errorCode, rightMotor = sim.simxGetObjectHandle(clientID, 'rightMotor3',
sim.simx_opmode_oneshot_wait)

def setBubbleRobVelocity(leftWheelVelocity, rightWheelVelocity):
    errorCode, leftMotor = sim.simxGetObjectHandle(clientID, '/leftMotor3',
sim.simx_opmode_oneshot_wait)
    errorCode, rightMotor = sim.simxGetObjectHandle(clientID, '/rightMotor3',
sim.simx_opmode_oneshot_wait)
    sim.simxSetJointTargetVelocity(clientID, leftMotor, leftWheelVelocity,
simConst.simx_opmode_streaming)
    sim.simxSetJointTargetVelocity(clientID, rightMotor, rightWheelVelocity,
simConst.simx_opmode_streaming)

def setBubbleRobAngle(angle):
    angle = [-90*math.pi/180, angle*math.pi/180, 0*math.pi/180]
    errorCode, leftMotor = sim.simxGetObjectHandle(clientID,'/leftMotor3',sim.
simx_opmode_oneshot_wait)
    errorCode,rightMotor = sim.simxGetObjectHandle(clientID,'/rightMotor3',
```

圖. 5.3: 組長 IPv6

最後在瀏覽器輸入 [http://\[組長 IP 位置\]:23020](http://[組長 IP 位置]:23020), 即可看到組長的場景，並且可以移動。

第六章 討論與分工

6.1 分工

分配工作

1. 場地設置 41023118 41023138
2. 車子設計與組裝 41023122 41023124
3. 輪盤記分板繪製 41023126 41023114
4. 輪盤記分板程式 41023119 41023120
5. 車子控制程式與設置 41023119 41023120
6. 會議記錄 41023126
7. 整體組合 41023119 41023120

6.2 討論紀錄

5/18 會議內容：分配工作

1. 場地設置：41023118 41023138
2. 車子設計與組裝：41023122 41023124
3. 輪盤記分板繪製：41023126 41023114
4. 輪盤記分板程式：41023119 41023120
5. 車子控制程式與設置：41023119 41023120

5/22 會議內容：討論分工的作業情況及尚未完成的事項

41023124：車子設計大略完成剩下背號

41023120：程式研究中

41023118：場地大致完成剩下球場線條

41023119：機械式記分板程式正在編寫中球員程式正在編寫中

41023126：機械記分板改第二版

41023138：場地大致完成剩下球場線條

41023122：球員除錯

41023114：機械記分板外觀研究

5/29 會議內容：討論作業情況及模擬場景連線

41023122：車子剩下背號

41023119：球員程式正在修改中

41023138：場景完成

和組長做模擬場景連線

6/5 會議內容：討論作業情況及 latax 報告

41023119：報告 pdf 未完成報告 latex 摘要完成

41023138：正在製作 latex 報告

41023126：繪製機械式計時器 cad 圖

41023124：負責 latex 第六章防火牆與 ipv6 內容設定

41023114：負責 latex 第六章防火牆與 ipv6 內容設定

第七章 心得

在這次團隊作業中，有遇到許多的困難，像是機械式計分板的程式和主機與各組員的連線問題，在解決問題途中問過 chatgpt 也參考過那些已經做出來的組別，慢慢地了解其中的一些原理，然後試著融入我們的作品也檢討了作品中的一些問題，從中讓我們也學習到了很多寶貴的經驗，我們不僅學會了如何實現遠距離連線對戰，還學會了如何組裝並製作實體的記分板，讓遊戲更加有趣，同時也讓我們深刻體會到了團隊合作的重要性，一個人的能力雖然可以做很多東西，但是在團隊中，通過互相溝通討論和分配工作，其他組員可以彌補各自的缺陷，並提高工作的效率，並體驗到了團隊合作的重要性，這將是我們在未來工作和學習中受益良多的經驗。

第八章 參考文獻

<https://mdecd2023.github.io/2a3-pj3ag1/content/index.html>