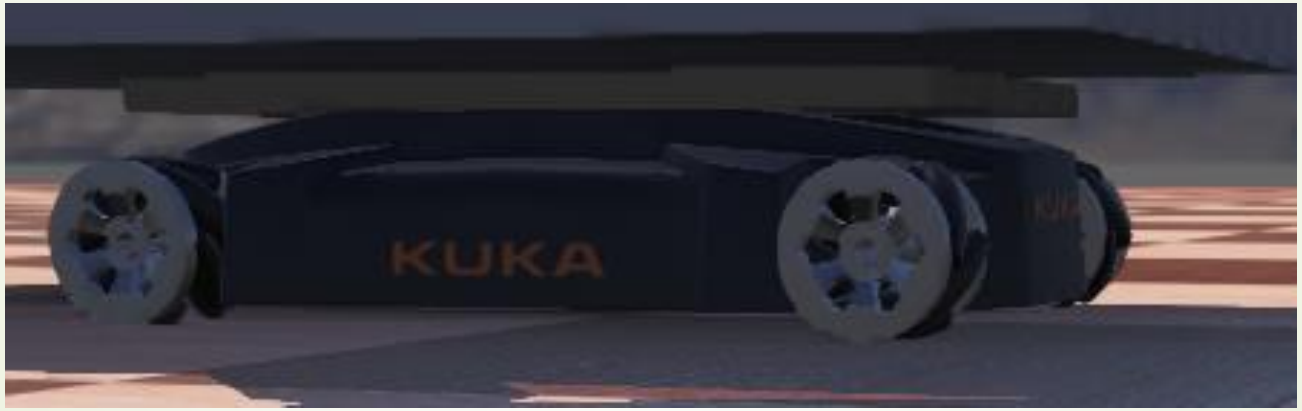
協同產品設計實習
期末報告

組長:41023213張義聖
組員:41023215許嘉祐
41023216郭宥辰
41071202陳依平
41071203侯昀熙
41071204黃雅萱

車子

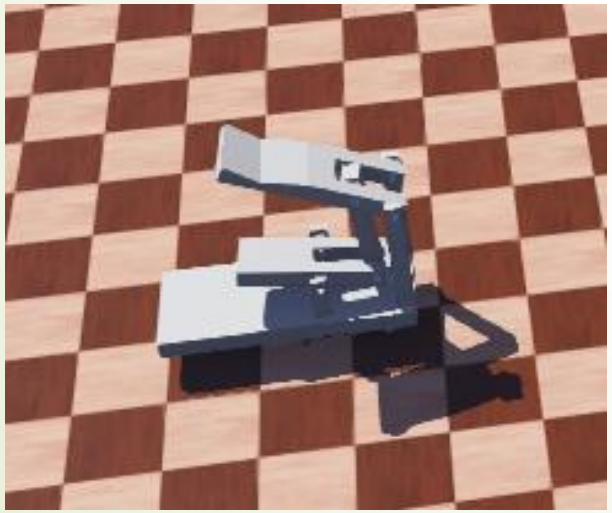
使用老師給的車子，讓我們可以減少製作過程，車子上新增一個板子來放投籃機



籃球機

如同作業3的籃球是一樣的，可以去看作業3的影片

只是在下面新增車子，可以使其移動



程式說明

1.Feed _ball

首先定義球的大小、顏色及位置，再來設定O鍵生成一顆靜止的球。

2.fourbar_controller

首先定義籃球機擊出的角度，再來設定J鍵回復投籃機的動作、 M則是將球擊出；上下左右鍵控制籃球機移動。

3.stand_controller

首先定義感測器，用來進行判斷球是否有投入籃框內，再來設定W鍵為籃框前進、A鍵為籃框左轉、 S鍵為籃框後退、 D鍵為籃框右轉。

emiter sensor跟reciver就是先透過程式發出分數訊息 再由reciver接收 然後透過程式轉圜為七段顯示器的顯示

4.counter_supervisor

利用7段顯示器來進行計分，進一顆得兩分，上限999。

Feed _ball

```
from controller import Supervisor, Keyboard
import time
import random
import numpy as np
import re
# ---------------- 參數區 ----------------
HOOP_CENTER = [0.622, -0.103, 0.742838]
BALL_DEF_PATTERN = re.compile(r"Sphere_\d+")
supervisor = Supervisor()
timestep = int(supervisor.getBasicTimeStep())
keyboard = Keyboard()
keyboard.enable(timestep)

sphere_radius = 0.1
TRAJECTORY_POINT_RADIUS = 0.03     # 軌跡小球半徑
TRAJECTORY_POINT_STEP = 0.12       # 軌跡點間最小距離
TRAJECTORY_MAX_POINTS = 5          # 只保留5個軌跡點
waiting_ball_def = None
waiting_ball_info = None
last_key_time = 0
debounce_time = 0.5
default_feed_pos = (-0.3, 0.0, 1.3)
PRINT_INTERVAL = 0.2

current_tracked_def = None
last_print_time = time.time()

# 軌跡資料
trajectory_points = []  # [(pos, def_name)] 最多五個
```

```
def axis_angle_to_rotation_matrix(axis, angle):
    x, y, z = axis
    c = np.cos(angle)
    s = np.sin(angle)
    C = 1 – c
    return np.array([
        [x*x*C + c,   x*y*C - z*s, x*z*C + y*s],
        [y*x*C + z*s, y*y*C + c,   y*z*C - x*s],
        [z*x*C - y*s, z*y*C + x*s, z*z*C + c]
    ])

def generate_valid_def_name(base_name="Sphere"):
  timestamp = int(supervisor.getTime() * 500)
    return f"{base_name}_{timestamp}_{random.randint(0, 10000)}"

def generate_random_color():
  return 1, 1, 1

def youbot_local_to_world(local_pos):
    youbot_node = supervisor.getFromDef('youbot')
    if youbot_node is None:
        raise RuntimeError("找不到 DEF 為 youbot 的 Robot 物件")
    youbot_translation = np.array(youbot_node.getField('translation').getSFVec3f())
    youbot_rotation = youbot_node.getField('rotation').getSFRotation()
    youbot_axis = youbot_rotation[:3]
    youbot_angle = youbot_rotation[3]
    youbot_rot_mat = axis_angle_to_rotation_matrix(youbot_axis, youbot_angle)
    rotated = youbot_rot_mat @ np.array(local_pos)
    world_pos = youbot_translation + rotated
    return tuple(world_pos)
```

Feed _ball

```
def create_static_ball(def_name, world_pos, r, g, b):
sphere_string = f
    root = supervisor.getRoot()
    children_field = root.getField("children")
children_field.importMFNodeFromString(-1, sphere_string)
def create_dynamic_ball(def_name, world_pos, r, g, b):
sphere_string = f
root = supervisor.getRoot()
    children_field = root.getField("children")
children_field.importMFNodeFromString(-1, sphere_string)
def create_trajectory_point(pos):
def_name = generate_valid_def_name("TrajectoryPt")
sphere_string = f
root = supervisor.getRoot()
    children_field = root.getField("children")
children_field.importMFNodeFromString(-1, sphere_string)
return def_name
def delete_trajectory_points():
  """刪除所有軌跡點"""
global trajectory_points
    for _, def_name in trajectory_points:
        node = supervisor.getFromDef(def_name)
        if node:
            node.remove()
    trajectory_points.clear()
def create_static_sphere(supervisor, x, y, z):
   global waiting_ball_def, waiting_ball_info
    def_name = generate_valid_def_name()
waiting_ball_def = def_name
    r, g, b = generate_random_color()
    world_pos = youbot_local_to_world((x, y, z))
waiting_ball_info = (world_pos, r, g, b)
create_static_ball(def_name, world_pos, r, g, b)
def activate_dynamic_ball():
   global waiting_ball_def, waiting_ball_info
    if waiting_ball_def is None or waiting_ball_info is None:
        return
    ball_node = supervisor.getFromDef(waiting_ball_def)
    if ball_node is not None:
      ball_node.remove()
        supervisor.step(int(supervisor.getBasicTimeStep()))
    world_pos, r, g, b = waiting_ball_info
    create_dynamic_ball(waiting_ball_def, world_pos, r, g, b)
    waiting_ball_def = None
    waiting_ball_info = None
def is_ball_landed(pos, threshold_z=0.13):
    """當球z接近地面時視為落地"""
  return pos[2] < threshold_zprint("按 O 產生一顆靜止球，按 M 讓球變 dynamic 可
擊出（最多只有5個軌跡點跟著球跑，球落地後軌跡自動消失）")
print("球總共有20顆")
while supervisor.step(timestep) != -1:
    key = keyboard.getKey()
    current_time = time.time()
    # 產生球
    if key == ord('O') and (current_time - last_key_time >= debounce_time):
        if waiting_ball_def is None:
            create_static_sphere(supervisor, *default_feed_pos)
            current_tracked_def = waiting_ball_def
           delete_trajectory_points()  # 新球產生時清除舊軌跡
        else:
          print("還有一顆球等待擊出，請先擊出再產生新球。")
        last_key_time = current_time
    # 讓球變動態
    if key == ord('M') and (current_time - last_key_time >= debounce_time):
activate_dynamic_ball()
       last_key_time = current_time
```

Feed _ball

```
# 拋物線軌跡追蹤
    if current_tracked_def is not None:
        ball_node = supervisor.getFromDef(current_tracked_def)
        if ball_node is not None:
            pos = ball_node.getPosition()
            # 每 PRINT_INTERVAL 印座標
            if current_time - last_print_time >= PRINT_INTERVAL:
                #print(f"球 {current_tracked_def} 絕對座標:
[{ pos[0]:.4f}, {pos[1]:.4f}, {pos[2]:.4f}]")
                last_print_time = current_time
            # 軌跡點：每隔一段距離才加一個，僅保留5個點
            if (not trajectory_points) or
np.linalg.norm(np.array(pos) - np.array(trajectory_points[-
1][0])) > TRAJECTORY_POINT_STEP:
                def_name = create_trajectory_point(pos)
trajectory_points.append((pos, def_name))
                if len(trajectory_points) > TRAJECTORY_MAX_POINTS:
# 移除最舊的點
                  _, old_def = trajectory_points.pop(0)
                    node = supervisor.getFromDef(old_def)
                    if node:
                        node.remove()
            # 若球落地，自動清除軌跡
            if is_ball_landed(pos):
              delete_trajectory_points()
        else:
# 球消失，停止追蹤並清除軌跡
            delete_trajectory_points()
            current_tracked_def = None
```

fourbar_controller

```
from controller import Robot, Keyboard
# Constants
TIME_STEP = 32
MAX_VELOCITY = 10.0
ANGLE_STEP = 25 * 3.14159 / 180
# 40 degrees in radians
POSITION_M = ANGLE_STEP          # +40 deg
POSITION_K = 0.0       # 0 deg
# Initialize robot and keyboard
robot = Robot()
timestep = int(robot.getBasicTimeStep())
keyboard = Keyboard()
keyboard.enable(timestep)
# Get devices
try:
    motor = robot.getDevice('motor1')
    sensor = robot.getDevice('motor1_sensor')
    sensor.enable(timestep)
    mechanism_enabled = True
except Exception:
    mechanism_enabled = False
# Wheel setup (if available)
try:
   wheels = [robot.getDevice(f"wheel{i+1}") for i in range(4)]
    for wheel in wheels:
        wheel.setPosition(float('inf'))  # Infinite position to enable velocity control
        wheel.setVelocity(0)  # Start with zero velocity
    platform_enabled = True
except Exception:
    platform_enabled = False
# State machine: which key is allowed to trigger
current_state = "allow_m"  # Start by allowing 'm'
# Key debounce
key_pressed = {
    'j': False,    'm': False}
```

```
while robot.step(timestep) != -1:
    key = keyboard.getKey()
        # Platform control
    if platform_enabled:
        if key == Keyboard.UP:
            for wheel in wheels:
                wheel.setVelocity(MAX_VELOCITY)
        elif key == Keyboard.DOWN:
            for wheel in wheels:
                wheel.setVelocity(-MAX_VELOCITY)
        elif key == Keyboard.LEFT:
            wheels[0].setVelocity(MAX_VELOCITY)
            wheels[1].setVelocity(-MAX_VELOCITY)
            wheels[2].setVelocity(MAX_VELOCITY)
            wheels[3].setVelocity(-MAX_VELOCITY)
        elif key == Keyboard.RIGHT:
            wheels[0].setVelocity(-MAX_VELOCITY)
            wheels[1].setVelocity(MAX_VELOCITY)
            wheels[2].setVelocity(-MAX_VELOCITY)
            wheels[3].setVelocity(MAX_VELOCITY)
        elif key == ord('Q') or key == ord('q'):
            print("Exiting...")
            break
        else:
            for wheel in wheels:
                wheel.setVelocity(0)
```

fourbar_controller

```
# Motor key control
  if mechanism_enabled:
      # Read current motor position (not for relative, but for debug)
_current_motor_position = sensor.getValue()
      # M key: only take action if allowed and not held
      if key == ord('M') or key == ord('m'):
         if not key_pressed['m'] and current_state == "allow_m":
            motor.setPosition(POSITION_M)
            current_state = "allow_j“
         key_pressed['m'] = True
      else:
         key_pressed['m'] = False
      # K key: only take action if allowed and not held
      if key == ord('J') or key == ord('j'):
         if not key_pressed['j'] and current_state == "allow_j":
            motor.setPosition(POSITION_K)
            current_state = "allow_m“
         key_pressed['j'] = True
      else:
         key_pressed['j'] = False
```

stand_controller

```
from controller import Robot, Keyboard, DistanceSensor
# Constants
#TIME_STEP = 32
# Simulation time step in milliseconds
WHEEL_RADIUS = 0.1  # Radius of the wheels in meters (10cm)
L = 0.471  # Half of the robot's length in meters
W = 0.376  # Half of the robot's width in meters
MAX_VELOCITY = 10.0  # Maximum velocity allowed for the
wheels
# Initialize the robot
robot = Robot()
# Get simulation time step
timestep = int(robot.getBasicTimeStep())
# Get the DistanceSensor device
sensor = robot.getDevice('sensor')
emitter = robot.getDevice("score_emitter")
sensor.enable(timestep)
previous_detected = False
score = 0score_to_send = 2
# Initialize the keyboard
keyboard = Keyboard()
#keyboard.enable(TIME_STEP)
keyboard.enable(timestep)
# Get motor devices
wheel5 = robot.getDevice("wheel5")  # Front-right
wheelwheel6 = robot.getDevice("wheel6")  # Front-left
wheelwheel7 = robot.getDevice("wheel7")  # Rear-right
wheelwheel8 = robot.getDevice("wheel8")  # Rear-left wheel
# Set motors to velocity control mode
for wheel in [wheel5, wheel6, wheel7, wheel8]:
wheel.setPosition(float('inf'))  # Enable velocity control
wheel.setVelocity(0)  # Set initial velocity to 0
```

```
def set_wheel_velocity(v1, v2, v3, v4):
    wheel5.setVelocity(v1)
    wheel6.setVelocity(v2)
    wheel7.setVelocity(v3)
    wheel8.setVelocity(v4)
# lookupTable 轉成程式用的格式
lookup_table = [
    (1000, 0.00),
    (800, 10),
    (400, 10),
    (0, 0.00)]
def ad_to_distance(ad_value):
    # 假設AD值遞減，距離遞增
    for i in range(len(lookup_table)-1):
        a0, d0 = lookup_table[i]
        a1, d1 = lookup_table[i+1]
        if a1 <= ad_value <= a0:
            # 線性插值
            return d0 + (d1 - d0) * (ad_value - a0) / (a1 - a0)
    # 超出範圍時回傳極值
    if ad_value > lookup_table[0][0]:
        return lookup_table[0][1]
    return lookup_table[-1][1]
# Main loop
print("Use 'W', 'A', 'S', 'D' keys to control the robot.")
print("S: Move forward, W: Move backward, A: Turn left, D: Turn right.")
print("Press 'Q' to quit.")
#while robot.step(TIME_STEP) != -1:
while robot.step(timestep) != -1:
```

stand_controller

```
key = keyboard.getKey()  # Read the key pressed
    # Read DistanceSensor value
    sensor_value = sensor.getValue()
    #print(sensor_value)
    distance = ad_to_distance(sensor_value)
    current_time = robot.getTime()
    #print(sensor_value)
    # Check if the ball blocks the sensor (you may need to
adjust the threshold based on your sensor's range)
currently_detected = distance > 5
    if distance != 0:
        print(distance)
    if currently_detected and not previous_detected:
        score += score_to_send
        previous_detected = True
        emitter.send(str(score_to_send).encode('utf-8'))
        print(f"得分!當前分數{score}")
 if key == ord('L') or key == ord('l'):
        print(sensor_value)
    if key == ord('M') or key == ord('m'):
        print("m")
    if key == ord('J') or key == ord('k'):
        print("j")
    if key == ord('O') or key == ord('o'):
        previous_detected = False
 if key == ord('S') or key == ord('s'):
        # Move forward
        velocity = MAX_VELOCITY
        set_wheel_velocity(velocity, velocity, velocity, velocity)
    elif key == ord('W') or key == ord('w'):
        # Move backward
        velocity = -MAX_VELOCITY
        set_wheel_velocity(velocity, velocity, velocity, velocity)
    elif key == ord('D') or key == ord('d'):
        # Turn right
        velocity = MAX_VELOCITY
        set_wheel_velocity(-velocity, velocity, -velocity, velocity)
    elif key == ord('A') or key == ord('a'):
        # Turn left
        velocity = MAX_VELOCITY
        set_wheel_velocity(velocity, -velocity, velocity, -velocity)
    elif key == ord('Q') or key == ord('q'):
        # Quit the program
        print("Exiting...")
        break
    else:
        # Stop the wheels when no key is pressed
        set_wheel_velocity(0, 0, 0, 0)
if __name__ == "__main__":
    run_robot()
```

counter_supervisor

```
from controller import Supervisor

SEGMENTS = [
    [1,1,1,1,1,1,0], # 0
    [0,1,1,0,0,0,0], # 1
    [1,1,0,1,1,0,1], # 2
    [1,1,1,1,0,0,1], # 3
    [0,1,1,0,0,1,1], # 4
    [1,0,1,1,0,1,1], # 5
    [1,0,1,1,1,1,1], # 6
    [1,1,1,0,0,0,0], # 7
    [1,1,1,1,1,1,1], # 8
    [1,1,1,1,0,1,1], # 9
]
DIGIT_MATERIALS = [
    ['a3mat', 'b3mat', 'c3mat', 'd3mat', 'e3mat', 'f3mat', 'g3mat'], # 百
    ['a2mat', 'b2mat', 'c2mat', 'd2mat', 'e2mat', 'f2mat', 'g2mat'], # 十
    ['a1mat', 'b1mat', 'c1mat', 'd1mat', 'e1mat', 'f1mat', 'g1mat'], # 個
]
ON_COLOR = [0, 1, 0]
OFF_COLOR = [0.05, 0.05, 0.05]

def set_digit(supervisor, digit_index, value):
    segs = SEGMENTS[value]
    for i, seg_on in enumerate(segs):
        mat_node = supervisor.getFromDef(DIGIT_MATERIALS[digit_index][i])
        if mat_node:
            mat_node.getField('diffuseColor').setSFColor(ON_COLOR if seg_on else
OFF_COLOR)
        else:
            print(f"找不到 {DIGIT_MATERIALS[digit_index][i]} 這個DEF")

def set_display(supervisor, value):
    value = max(0, min(999, int(value)))
    h = value // 100
    t = (value // 10) % 10
    u = value % 10
    set_digit(supervisor, 0, h)
    set_digit(supervisor, 1, t)
    set_digit(supervisor, 2, u)

supervisor = Supervisor()
timestep = int(supervisor.getBasicTimeStep())

score = 0
receiver = supervisor.getDevice("score_receiver")
receiver.enable(timestep)

while supervisor.step(timestep) != -1:
    while receiver.getQueueLength() > 0:
        data = receiver.getString()
        if data.isdigit():
            try:
                received_score = int(data)
                score += received_score
                print(f"收到得分訊息: +{received_score}, 總分: {score}")
            except Exception as e:
                print("訊息格式錯誤:", e)
        receiver.nextPacket()
    set_display(supervisor, score)
```

問題

- 1.球進去沒有顯示得分
- 2.球在同一個地方投進，卻沒有得分
- 3.紅外線感測器沒有感測到球
- 4.球會穿過籃板

解決方法

- 1.因為不小心把SENSOR刪掉，把它重新加回來就好
- 2.投太快，在第一顆球還未判定之前，後投出
- 3.新增多條紅外線感測器
- 4.重開一次檔案就好

心得

- 40171202 陳依平

期末報告真的很複雜，要把之前做過的東西整合在一起，花費我們好幾個禮拜的時間，還有請程式較擅長的朋友來幫忙，加上AI的建議，才能壓線完成，雖然過程有點辛苦，但學到不少東西，之後說不定能派上場。

- 41071203

這次的final在組裝上面沒有太大的問題，問題最大的還是卡在程式方面，要讓他整個動起來並順利運行花了很長的時間，還是透過網路還有接觸過相關專業的同學幫忙除錯才能順利的做出來，看到可以順利運作時很開心，雖然過程很累但是是值得的

- 41071204黃雅萱

這次的final比之前要複雜及困難許多，期間在研究的時候處處碰壁，也遇到很多問題，嘗試去問AI，但AI給我們的解決方案也有很多問題導致無法正常運作，中間一度卡關很久，慶幸的是身邊有接觸過相關的朋友，所以最後也壓線完成了。3.

- 41023213張義聖

這次的期末專題條件比較困難，尤其是在程式的部分花了很多時間。過程中不斷嘗試不同的方法，但總會遇到一些錯誤。我們有請教AI、上網查資料，還請教了有經驗的朋友協助，才解決問題。雖然花了不少心力，但也因此更加了解整個系統運作的流程，學到了許多之前沒接觸過的知識。

- 41023215

這次的final作業跟之前比起來相當的困難，在製作的過程中遇到了許多的問題，後來經過同學的幫忙才順利解決，透過這次的作業讓我更熟悉程式的撰寫，也讓我增強了程式撰寫的能力，所以對未來我相信有很大的幫助。

- 41023216

這次的FINAL跟之前比起來難了很多，在製作的過中，遇到了許多問題，在程式的部分出現的許多問題，所以花了許多時間 ，去把錯誤的部分找出來，也幸好有組員有認識，了解程式的朋友，幫我找到程式哪裡出現了錯誤，才能順利的完成。