

# 使用大型語言模型進行機器控制指令的自動化生成

## Automated Generation of Machine Control Commands Using Large Language Models

研究生：蔡時富  
指導教授：蘇木春



# 大綱

- 研究動機與目的
- 背景知識與相關研究
- 研究方法
- 實驗設計以及成果
- 結論與未來展望



# 大綱

- 研究動機與目的
  - 研究動機
  - 研究目的
- 背景知識與相關研究
- 研究方法
- 實驗設計以及成果
- 結論與未來展望



# 研究動機

- 在現今蓬勃發展的人工智慧領域中，隨著高效能運算技術 (High Performance Computing, HPC) 的快速發展，許多大型語言模型 (Large Language Model, LLM) 的問世，至今已經深刻改變了我們對於人工智慧認知。
- 對於這些模型的需求不斷增長，也促使了各大科技公司不斷投入資源，提供更加強大且多樣化的大型語言模型，使的人工智慧發展方向更加明朗。



# 研究動機

- 目前許多研究集中於微調大型語言模型本身，以提升其在各個任務中的準確度與效能。
- 本研究看到了大型語言模型在自動程式碼生成 (Code Generation) 領域的可能性，希望以此為出發點，讓使用者說出的自然語言指令透過語言模型轉換為程式碼，進而去控制其他硬體設備，使得使用者能夠以更加直觀和自然的方式與機器進行互動。



# 研究目的

- 探索大型語言模型在自動程式碼生成中的應用，並達到使用自然語言控制機器的目的。
- 驗證將自然語言指令轉換為程式碼的可行性和實際效果。
- 自行組裝設計機器，並透過 3D 列印技術實現更為客製化的機型製作。
- 分析和評估所開發的系統在不同應用場景下的實際效果和應用價值。
- 提供相關技術和方法的研究成果，為自然語言控制機器的相關研究和應用提供實證基礎和技術範本。



# 大綱

- 研究動機與目的
- 背景知識與相關研究
  - 背景知識
  - 文獻回顧
- 研究方法
- 實驗設計以及成果
- 結論與未來展望





# 大綱

- 研究動機與目的
- 背景知識與相關研究
  - 背景知識
    - 大型語言模型的研究現況
    - 智慧機器與人工智慧物聯網的應用場景
    - 3D 列印技術的發展現況
  - 文獻回顧
- 研究方法
  - 實驗設計以及成果
  - 結論與未來展望



CHILab

Computational Intelligence and Human-Computer Interaction Lab.



# 大型語言模型的研究現況

- 早期階段：統計模型為當時主要的方法，但這些模型在處理複雜的語言時存在一定的局限性，特別是對於語境理解和生成的能力有限。
- 神經語言模型（Neural Language Model, NLM）：自然語言處理技術轉為使用類神經網路作為模型架構，但由於當時硬體計算能力和儲存空間的限制，這些模型的規模和性能十分有限。
- 變換器模型（Transformer Model）：變換器模型的提出是自然語言處理領域的一個重要突破，它引入了自注意機制（Self-Attention），使模型能夠更好的捕捉序列中的長程依賴性，從而取得了顯著的性能提升。



# 大型語言模型的研究現況

- BERT (Bidirectional Encoder Representations from Transformers) : BERT 是一個雙向的 Transformer 模型，通過預訓練和微調，在多項自然語言處理任務上表現卓越。
- 大型語言模型 (Large Language Model, LLM) : 在過去的幾年中，大型語言模型取得了巨大的進展，這些模型擅長理解和生成類人語言。
- 多模態模型 (Large Multimodal Models, LMM) : 現今的人工智慧技術不僅能夠處理文本資訊，還可以處理如圖像、音訊、影片等多種模式。



# 智慧機器與人工智慧物聯網的應用場景

- 工業 4.0(Industry 4.0)：人工智慧物聯網技術是智慧工廠實現生產過程的自動化和智能化的關鍵，這有助於提高生產效率並減少人力成本，從而提高了整個製造業的競爭力。
- 物流和倉儲：無人搬運車 (Automated Guided Vehicle, AGV) 在物流和倉儲管理中扮演著關鍵角色。它們可以自動運送物品，根據預定路徑進行導航。
- 醫療保健：手術機器人是其中一個典型的例子，它可以幫助外科醫生不受時間與地點的影響進行精確的手術操作，同時兼顧了手術的準確性、安全性與急迫性三個重要需求。





# 智慧機器與人工智慧物聯網的應用場景



Tesla factory



Da Vinci xi



Amazon warehouse robot



# 3D 列印技術的發展現況

- 受惠於電腦輔助設計（CAD）、電腦輔助製造（CAM）和電腦數值控制加工（CNC）等技術的蓬勃發展。使快速成形技術（RP）在現在的工業設計與製造的研究領域中，已越來越成熟。此技術時常用於快速生成零件模型的製造技術，它通過電腦控制，將材料進行堆疊加工，生成立體實品，因此又稱為積層製造技術（AM）。
- 現今的 3D 列印技術已被廣泛應用於建築、工業設計、汽車、航太、醫療生技、服飾、飾品、地理資訊和食品等產業。一些劃時代的成品包括 3D 列印的房屋、無人機、汽車、人工血管和各式食品都已在近年陸續問世。





# 3D 列印技術的發展現況



3D列印：房子



3D列印：糖果



3D列印：無人機



# 大綱

- 研究動機與目的
- 背景知識與相關研究
  - 背景知識
  - 文獻回顧
    - 大型語言模型及其在程式碼生成與機器控制上的應用
    - 運動學研究與機器人控制
    - 3D 列印應用於機器人製作的相關文獻
- 研究方法
  - 實驗設計以及成果
  - 結論與未來展望





# 大型語言模型及其在程式碼生成與機器控制上的應用

- Vaithilingam 等人探討大型語言模型作為程式碼生成工具的可用性
- 招募了 24 名擁有不同程式設計經驗的參與者，使用Copilot執行一系列不同難度的程式設計任務，而研究人員則從旁紀錄。
- 大多數參與者都喜歡在進程式設計時使用 Copilot，因為它能夠提供一個有用的起點，讓使用者能節省許多思考與搜索的時間。
- 以文字形式將任務賦予大型語言模型，通常需要人為修改。



# 大型語言模型及其在程式碼生成與機器控制上的應用

- Sai 等人探討了 ChatGPT 在機器人應用中的實驗研究
- 提出了一種策略，ChatGPT能透過與人對話、解析XML標記以及合成程式碼，完成機器人領域的一系列任務。
- 研究中提供了一個名為 PromptCraft 的開源研究工具，其中包含一個平台，研究人員可以共同上傳並投票選出最優的對話策略。
- 將大型語言模型與機器人領域結合，由於重點在於研究對話策略，實驗中需要較長的對話過程逐步修正。



# 大型語言模型及其在程式碼生成與機器控制上的應用

- Liang 等人使用大型語言模型直接生成可以作為控制機器人策略的程式。
- 提出了名為「程式即為策略」（Code as Policies）的方法，省去了重複對話的過程。
- 使用大型語言模型生成的程式碼，在某些實驗中的確可以成功地控制機器人執行各種指令，不過隨著任務難度的複雜化，實驗中隨之降低的準確率也展示出了，現今大型語言模型在語意推理和空間幾何推理領域的發揮是有限的。
- 實驗中使用大型語言模型輸出的程式碼直接控制機器，但功能有限。



# 運動學研究與機器人控制

- 順向運動學 (Forward Kinematics)：即為順著機械臂的每個關節，透過計算每個關節角度來得出機械臂末端的位置。
- 逆向運動學 (Inverse Kinematics)：是順向運動學的反向問題。能根據機械臂末端的位置，計算出每個關節應該轉動的角度。



# 運動學研究與機器人控制

- 順向運動學中，Denavit-Hartenberg 方法為本研究使用的計算方式。

$$\begin{aligned}
 {}^{i-1}_i T &= R_x(\alpha_{i-1}) D_x(a_{i-1}) R_z(\theta_i) Q_i(d_i) \\
 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_{i-1} & -\sin \alpha_{i-1} & 0 \\ 0 & \sin \alpha_{i-1} & \cos \alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & a_{i-1} \\ \sin \theta_i \cos \alpha_{i-1} & \cos \theta_i \cos \alpha_{i-1} & -\sin \alpha_{i-1} & -\sin \alpha_{i-1} d_i \\ \sin \theta_i \sin \alpha_{i-1} & \cos \theta_i \sin \alpha_{i-1} & \cos \alpha_{i-1} & \cos \alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

$${}^{base}_{end} T = {}^0_1 T {}^1_2 T \dots {}^{n-1}_n T$$



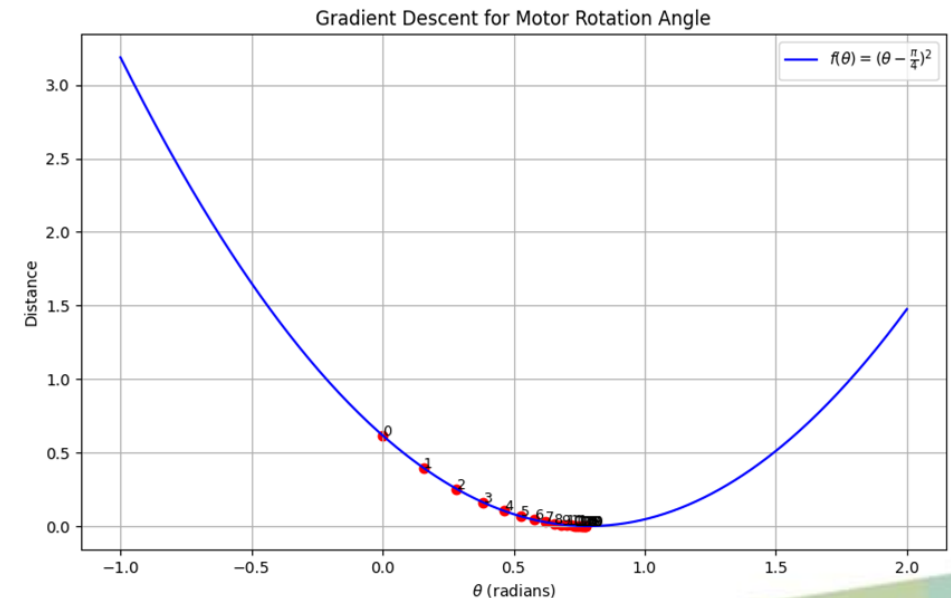


# 運動學研究與機器人控制

- 逆向運動學問題，實驗中使用了幾何求解法 (Geometric solution approach)，與梯度下降法 (Gradient descent solution approach) 來開發機械臂的逆向控制。

$$\theta_1 = A \tan 2 \left( \pm \sqrt{1 - \left( \frac{p_x (l_1 + l_2 \cos \theta_2) + p_y l_2 \sin \theta_2}{p_x^2 + p_y^2} \right)^2}, \frac{p_x (l_1 + l_2 \cos \theta_2) + p_y l_2 \sin \theta_2}{p_x^2 + p_y^2} \right)$$

$$\theta_2 = A \tan 2 \left( \pm \sqrt{1 - \left( \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2} \right)^2}, \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2} \right)$$



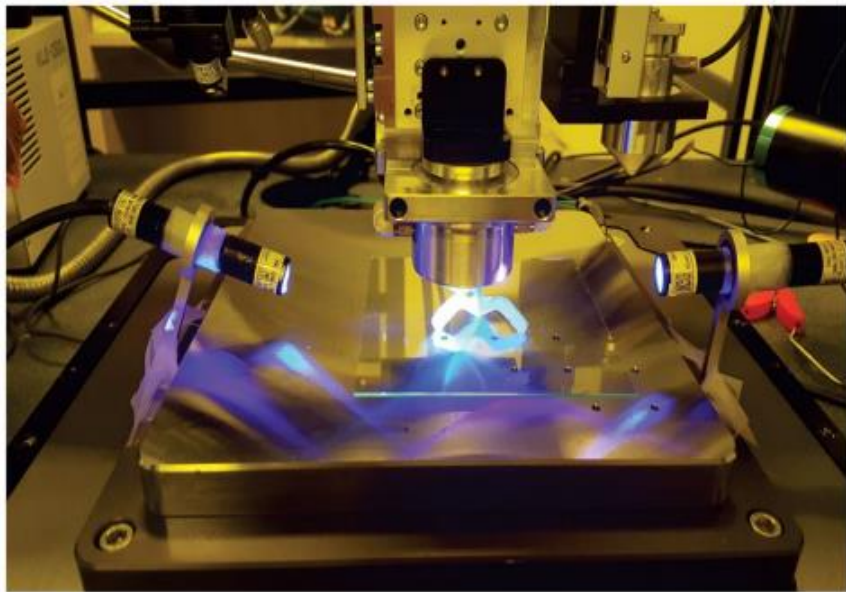
# 3D 列印應用於機器人製作的相關文獻

- Siemasz等人介紹了如何利用3D列印技術製造機器手臂，受惠於3D列印技術方便，且快速成型的特性，在實驗團隊在實驗中才得以快速、低成本的製造複雜的機器手臂零件。
- Gul等人探討了3D列印在軟體機器人中的應用，軟體機器人是一種由柔性材料製成的機器人，具備高柔韌性和高適應性的特點能模仿生物的動作和行為。
- Zhang等人介紹了一種創新的大規模3D列印方法，通過多個移動機器人的同步作業來完成。傳統的大規模 3D 列印設備通常尺寸龐大且笨重，難以應對複雜的列印需求。

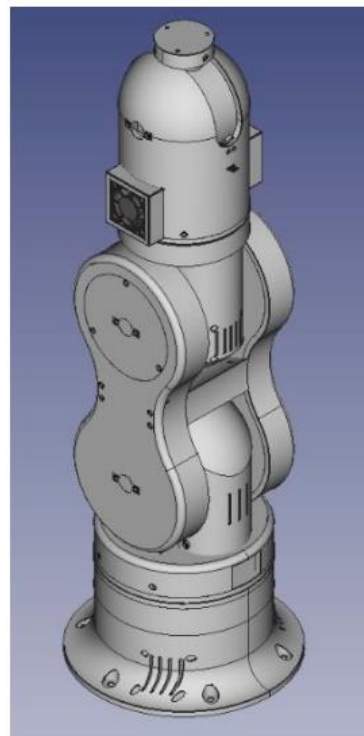




# 3D 列印應用於機器人製作的相關文獻



軟體機器人



機械臂模型



大規模3D列印



# 大綱

- 背景知識與相關研究
- 研究方法
  - 硬體設計流程
  - 運動學開發
  - 大型語言模型開發
  - 系統架構
- 實驗設計以及成果
- 結論與未來展望



# 大綱

- 背景知識與相關研究
- 研究方法
  - 硬體設計流程
    - 模型設計軟體：Autodesk Fusion 360
    - 檔案輸出格式：STL (Stereolithography)
    - 3D 列印機：Crealty K1 MAX
    - 馬達與開發版介紹
  - 運動學開發
  - 大型語言模型開發
  - 系統架構



# 模型設計軟體：Autodesk Fusion 360

- Autodesk Fusion 360 是一款集合了電腦輔助設計 (CAD)、電腦輔助製造 (CAM)、電腦輔助工程 (CAE) 及印刷電路板設計 (PCB) 的多功能設計軟體，可以使用此軟體完成3D建模、配裝設計、開發版組裝測試、機械結構碰撞測試、設計圖生成等一系列開發工作。



## Fusion 360



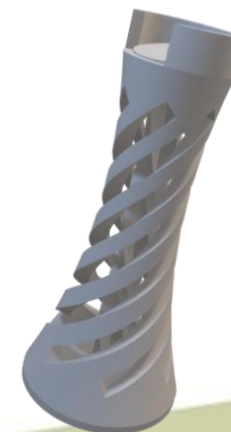


# 檔案輸出格式：STL (Stereolithography)

- STL (Stereolithography) 是一種常用的 3D 模型文件格式，特別是在3D列印領域中廣泛應用，因為不包含材質、顏色等其他屬性，格式結構較為單純，所以在3D列印領域被廣泛應用。



**3D SYSTEMS®**



# 3D 列印機：Creality K1 MAX

- 大尺寸列印：使用 CoreXY 運動結構，300x300x300mm
- 解析度：0.1mm 列印解析度，可以實現細膩的表面細節和精確的尺寸控制。
- 高速列印：能達到  $600\text{mm/s}$ 、 $20000\text{mm/s}^2$  的加速度與  $32\text{mm}^3/\text{s}$  的流量。
- 耗材多樣性：支援多種列印耗材，包括聚乳酸 (PLA)、丙烯腈-丁二烯-苯乙烯 (ABS)、熱塑性聚氨酯 (TPU) 等。



# 馬達與開發版介紹

- 馬達選擇：SG90、MG90S、MG996R
- 特點：輕、成本低、控制方便



SG90



MG90S



MG996R





# 馬達與開發版介紹

- 開發版選擇：ESP32 Doit-Devkit、ESP32-S3-Devkit、Raspberry Pi 4
- 特點：輕、功能足夠、支援無線連接



ESP32 Doit-Devkit



ESP32-S3-Devkit



Raspberry Pi 4



# 馬達與開發版介紹

- 驅動模組選擇：PCA9685、L298N
- 特點：操作電壓範圍廣、適合用於控制馬達



PCA9685



L298N



# 大綱

- 背景知識與相關研究
- 研究方法
  - 硬體設計流程
  - 運動學開發
    - 運動模擬環境
    - 順向運動學
    - 逆向運動學
  - 大型語言模型開發
  - 系統架構



實驗設計以及成果

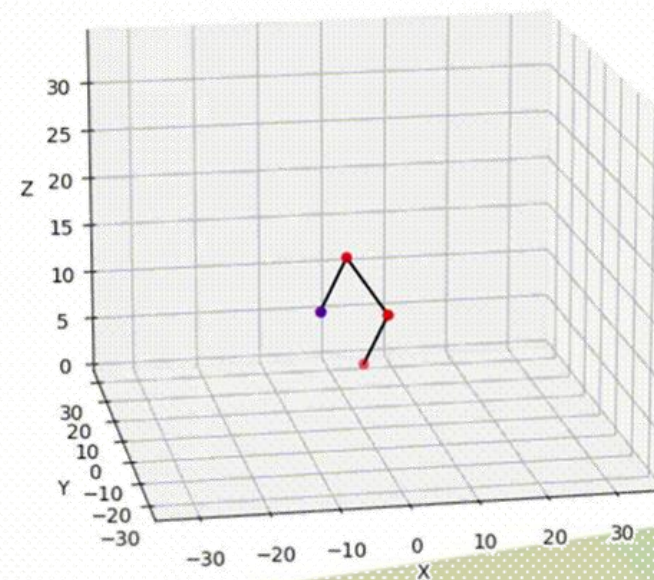
結論與未來展望

Computational Intelligence and Human-Computer Interaction Lab.

# 運動模擬環境

- 本專案使用 Python 語言開發了一個簡易的機械臂模擬環境。這個模擬環境的基本原理，是將機械臂定義為一個擁有多個可調整參數的獨立物件，其中包括力臂長度、扭轉角、節點偏移、節點角度、目標頂點、和目前頂點等。

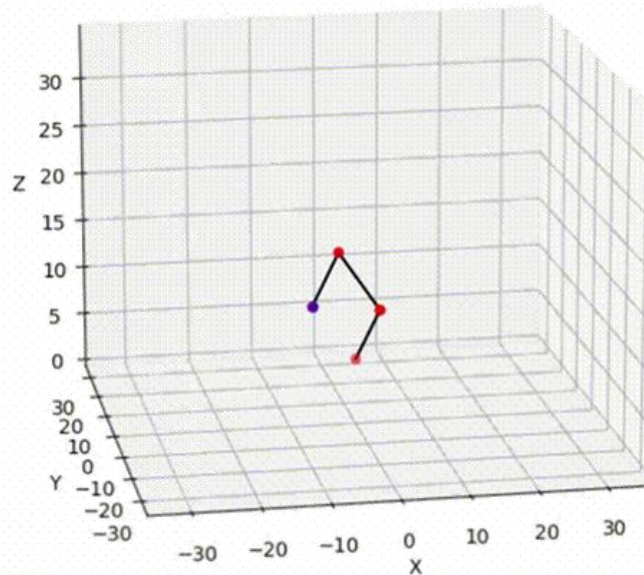
```
1 class ArmEnv_3D:
2     def __init__(self, lList, vision=True):
3         self.Link_length = lList
4         self.Link_offset=[0]*len(lList)
5         self.Twist_angle=[0]*len(lList)
6         self.Joint_angle = [0]*len(lList)
7         self.Target_point=[0, 0, 0]
8         self.Current_point=[0, 0, sum(lList)]
9
10
11     def Forward_kinematics(self, joint_angles):...
12
13     def Geometric_solution(self):...
14
15     def Gradient_descent(self, times=100, points=10, speed=5):...
16
17     #visualization
18     def initPicture(self):...
19
20     def updatePicture(self):...
```





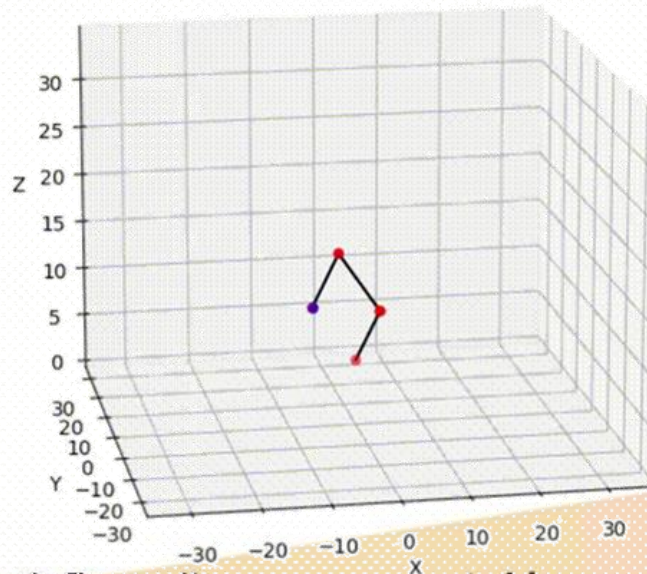
# 順向運動學

- 在順向運動學部分，本專案採用了 Denavit-Hartenberg 方法作為主要的計算方式。這種方法能夠系統地描述和計算機械臂的運動行為。模擬環境中的相關函式可以直接被呼叫，利用即時的機械臂參數來計算操作點的空間座標。



# 逆向運動學

- 在逆向運動學方面，本專案結合了幾何求解法(Geometric solution approach)和梯度下降法(Gradient descent solution approach)來計算。
- 幾何求解方法主要用於計算運動模式較為簡單的機械臂(二軸路徑規劃)，而梯度下降法則用於運動較為複雜的機械臂(三軸以上的路徑規劃)。



# 大綱

- 背景知識與相關研究
- 研究方法
  - 硬體設計流程
  - 運動學開發
  - 大型語言模型開發
    - OpenAI 與 GPT 模型
    - 模型使用流程
  - 系統架構



實驗設計以及成果

結論與未來展望

Computational Intelligence and Human-Computer Interaction Lab.



# OpenAI 與 GPT 模型

- 本實驗使用的大型語言模型為 OpenAI 所推出的Generative Pre-trained Transformer(GPT)模型，GPT 模型是一種基於Transformer架構的生成式預訓練模型，用於應對自然語言處理相關的任務。



# 模型使用流程

- 在對 GPT 模型下達指令時，需要將預計上傳的文字資訊封裝成特定格式

```
{  
  "model": "model_type",  
  "prompt": " 介紹一下 OpenAI 和 GPT 模型。",  
  "max_tokens": 100,  
  "temperature": 0.7,  
  "top_p": 1.0,  
  "n": 1,  
  "stop": null  
}
```

JSON request 範本

```
curl https://api.openai.com/v1/completions \  
-H "Content-Type: application/json" \  
-H "Authorization: Bearer API 密鑰" \  
-d '{  
  "model": "model_type",  
  "prompt": " 介紹一下 OpenAI 和 GPT 模型。",  
  "max_tokens": 100,  
  "temperature": 0.7,  
  "top_p": 1.0,  
  "n": 1,  
  "stop": null  
}'
```

HTTP request 範本



# 模型使用流程

- model: 指定使用的GPT模型，例如 text-davinci-004、GPT-4o、GPT-3.5 等。
- prompt: 下達的指令。
- max\_tokens: 回傳文字的最大長度。
- temperature: 控制生成文字的隨機性。此值越高會生成更具創造性的文字，反之值越低則生成越保守的文字。
- top\_p: 取樣方法，決定生成文本的多樣性。
- n: 回傳生成文章的數量。
- stop: 用於指定生成文字停止的條件，可設定為空值。



# 模型使用流程

- model: GPT-3.5 Turbo、GPT-4o
- prompt:
- max\_tokens: 1024
- temperature: 0.7
- top\_p: 1.0
- n: 1
- stop: null



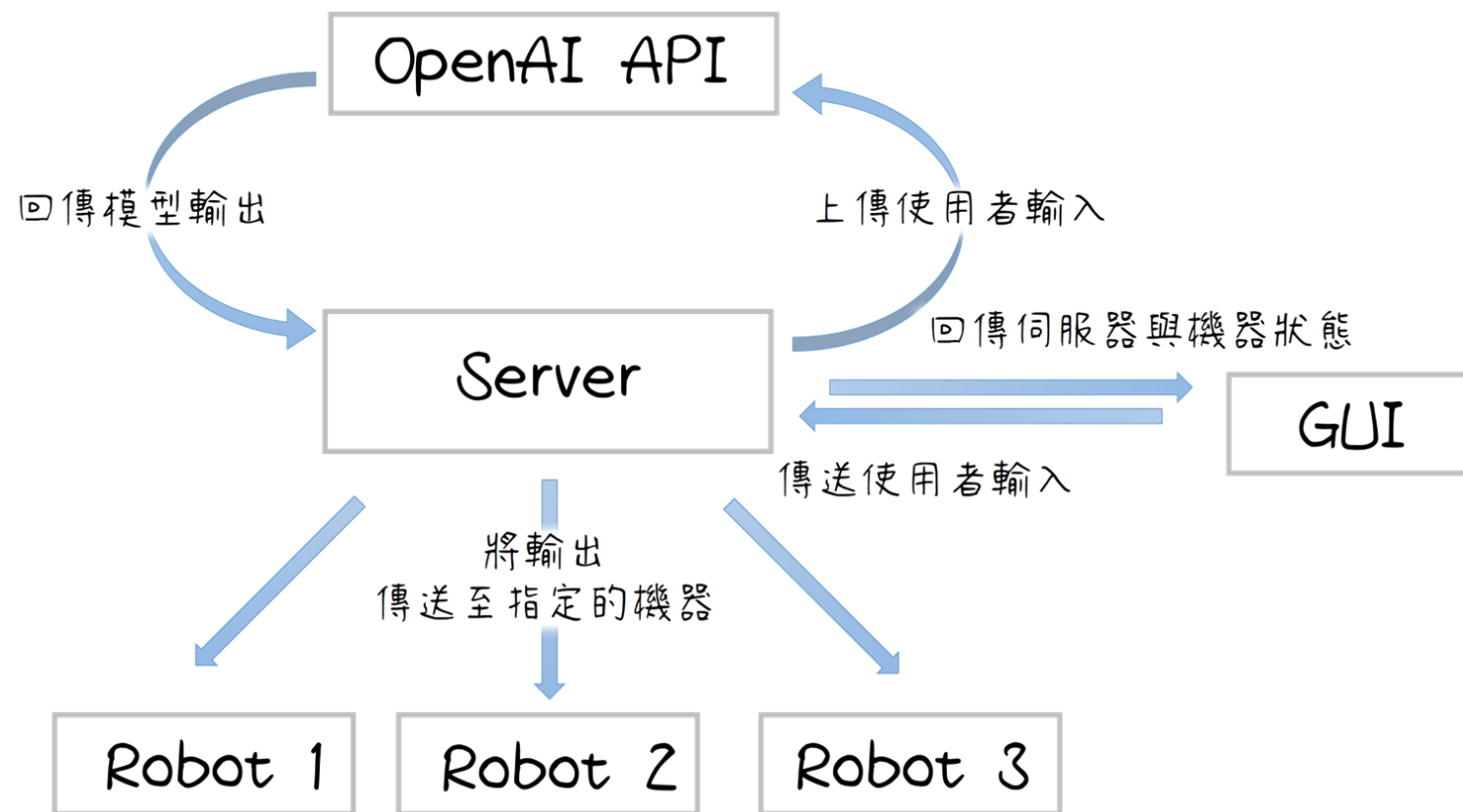
# 大綱

- 背景知識與相關研究
- 研究方法
  - 硬體設計流程
  - 運動學開發
  - 大型語言模型開發
  - 系統架構
    - 系統架構與流程
- 實驗設計以及成果  
結論與未來展望



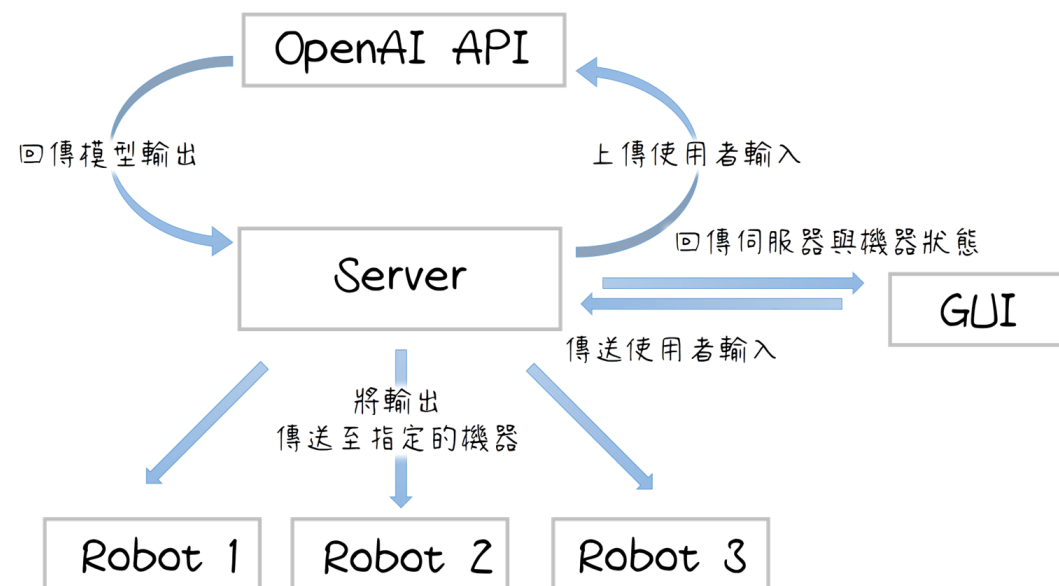


# 系統架構與流程



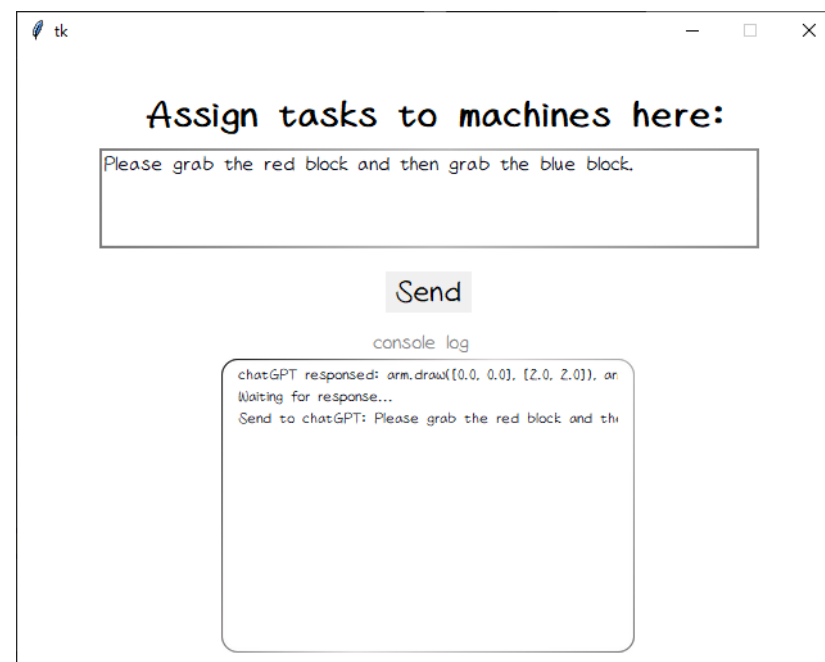
# 系統架構與流程

1. GUI 上傳使用者輸入至 Server
2. Server 上傳使用者輸入至 OpenAI API
3. OpenAI API 回傳模型輸出至 Server
4. Server 將輸出傳送至指定的機器
5. Server 向 GUI 回傳伺服器與機器狀態



# 系統架構與流程

1. GUI 上傳使用者輸入至 Server
2. Server 上傳使用者輸入至 OpenAI API
3. OpenAI API 回傳模型輸出至 Server
4. Server 將輸出傳送至指定的機器
5. Server 向 GUI 回傳伺服器與機器狀態



# 大綱

- 研究方法
- 實驗設計以及成果
  - 實驗一：機械臂的基本控制
  - 實驗二：將機械臂用於畫圖
  - 實驗三：機械臂在自動運輸車上的應用
- 結論與未來展望



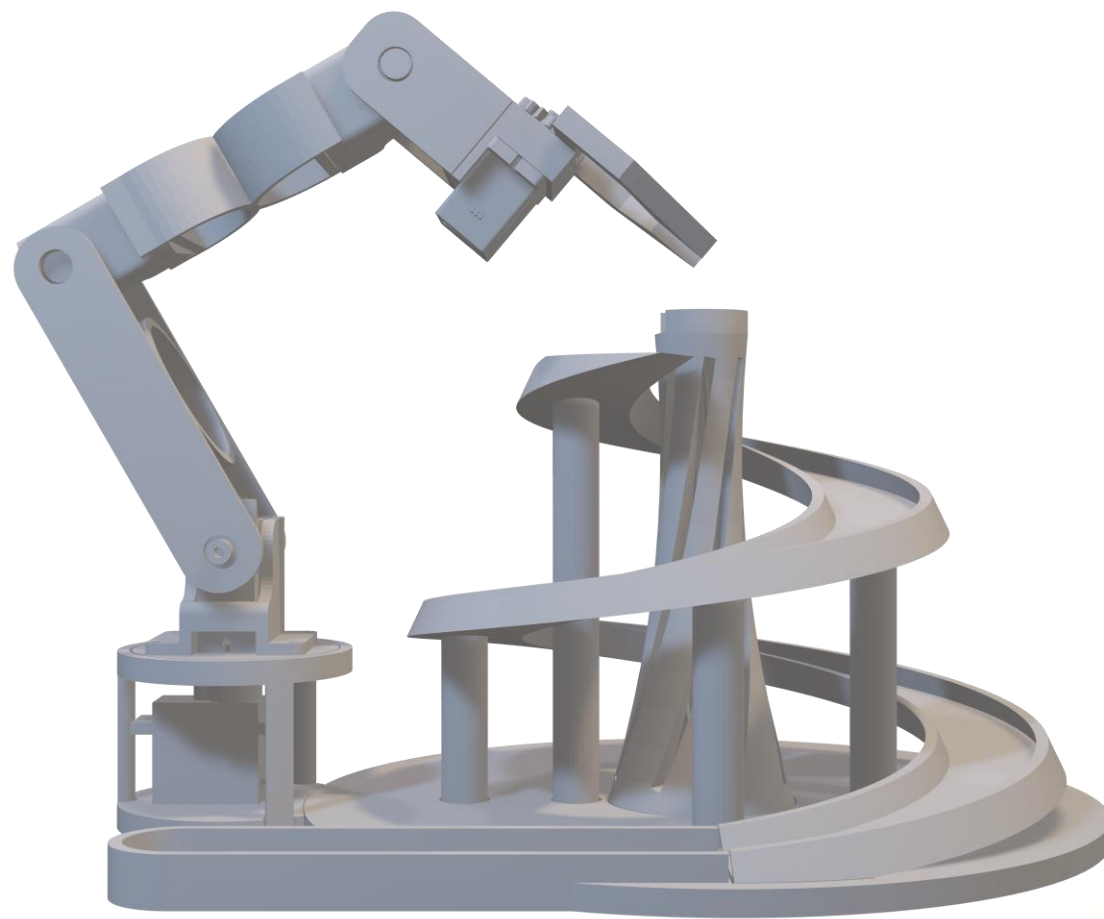


# 大綱

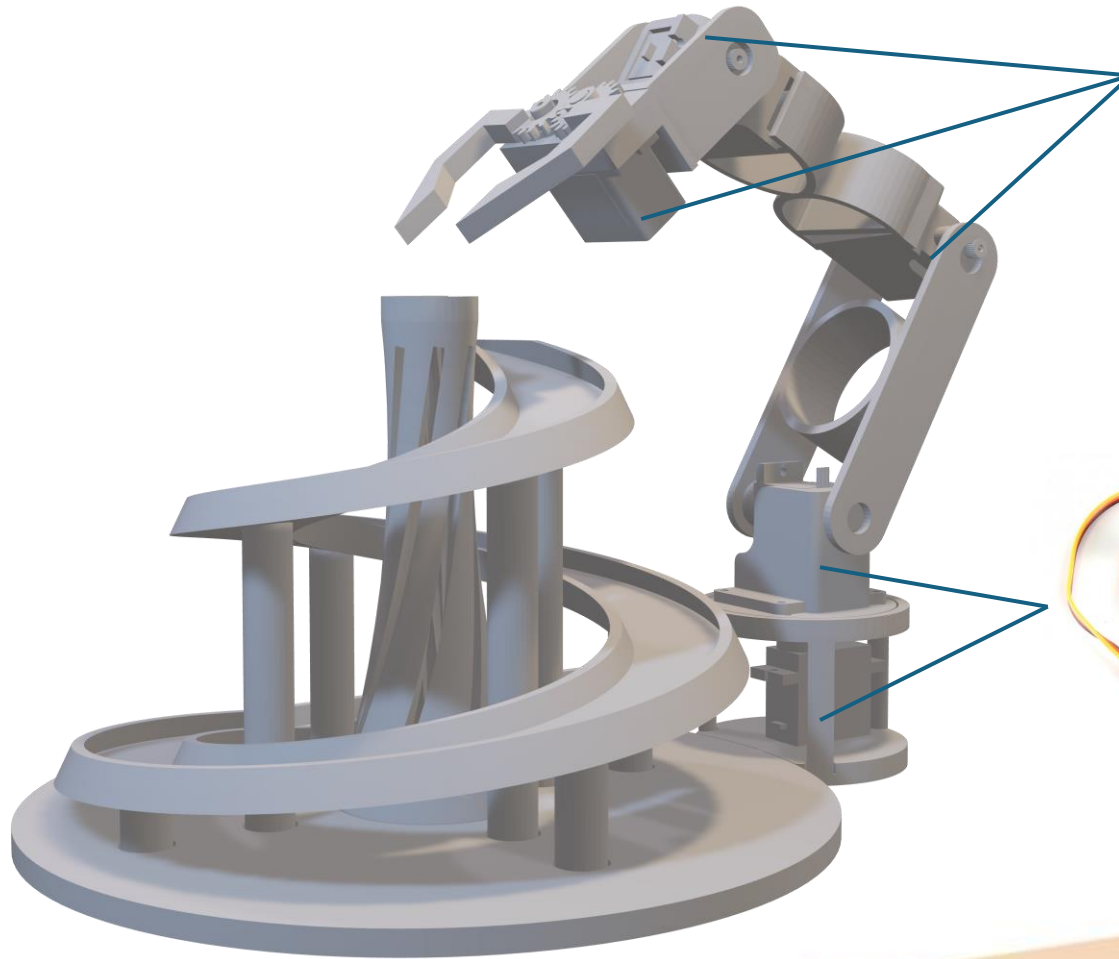
- 研究方法
- 實驗設計以及成果
  - 實驗一：機械臂的基本控制
    - 機械結構設計圖
    - 函數設計
    - 下達指令的格式範例
    - 實驗結果
  - 實驗二：將機械臂用於畫圖
  - 實驗三：機械臂在自動運輸車上的應用



# 機械結構設計圖



# 機械結構設計圖



SG90



ESP32-S3-Devkit



MG90S



# 函數設計

- 目標是讓機械臂能夠成功夾取特定物體，並放置於指定位置。這個過程涉及機械臂路徑的規劃和精確的抓取動作。
- 為了實現這一目標，我們撰寫了一個簡易的路徑規劃程式，利用語言模型在填空方面的優勢，使其生成和編輯機械臂的運動指令。

```
1  from ArmEnv_3D_ori import ArmEnv_3D
2  import math
3
4  Arm = ArmEnv_3D([5.942, 6.81, 8.747, 7.523], False)
5  path = []
6  #Please planning path form hear
7  #path+=Arm.moveto([0, 0, 25], 20) explain: Move to [0, 0, 25] in 20 steps
8  #path+=[[ -91, 0, 0, 0]] explain: Close claw
9  #path+=[[ 91, 0, 0, 0]] explain: Open claw
10
11 #reset
12 end=path[-2].copy()
13 while not sum(end)==0:
14     for i in range(len(end)):
15         if end[i]>0: end[i]-=1
16         elif end[i]<0: end[i]+=1
17     path.append(end.copy())
```

} 運動學模擬環境程式

} LLM填空部分(機械臂路徑規劃)

} 讓機械臂回到初始位置





# 下達指令的格式範例

- 架構使用上述JSON request範本。
- 指令：Please catch the ball to the top of the tower.

```
{  
  "model": "model_type",  
  "prompt": "  
  "max_tokens": 100,  
  "temperature": 0.7,  
  "top_p": 1.0,  
  "n": 1,  
  "stop": null  
}
```

```
{  
  role: "user",  
  content :  
  "  
  from ArmEnv_3D_ori import ArmEnv_3D\  
  import math\  
  \  
  Arm = ArmEnv_3D([5.942, 6.81, 8.747, 7.523], False)\  
  path = []\  
  #Please planning path form hear\  
  #path+=Arm.moveto([0, 0, 25], 20) explain: Move to [0, 0, 25] in 20 steps\  
  #path+=[-91, 0, 0, 0]] explain: Close claw\  
  #path+=[[91, 0, 0, 0]] explain: Open claw\  
  \  
  #reset\  
  end=path[-2].copy()\  
  while not sum(end)==0:\  
    for i in range(len(end)):\  
      if end[i]>0: end[i]-=1\  
      elif end[i]<0: end[i]+=1\  
      path.append(end.copy())\  
  "  
  },  
  {  
    role: "user",  
    content: "This is a sample program for controlling a robotic arm.\\  
    Please modify it to help me complete the following requirements: \  
    the ball is at [10.5, 12, 2], and the top of the tower is at [9.5, 0, 14.5]. \  
    Please help me catch the ball to the top of the tower. \  
    During the process, please do not get too close to the side of the tower \  
    (keep outside the radius 7).\  
    return only code.\\  
    "  
  }  
}
```

程式碼部分

指令要求



# 下達指令的格式範例

- 架構使用上述JSON request範本。
- 指令：Please catch the ball to the top of the tower.

```
{
  role="assistant",
  content=
  "\
from ArmEnv_3D_ori import ArmEnv_3D\
import math\
\
Arm = ArmEnv_3D([5.942, 6.81, 8.747, 7.523], False)\
path = []\
path += Arm.moveto([0, 0, 25], 20)\
path += Arm.moveto([10.5, 12, 2], 20)\
path += [[-91, 0, 0, 0]]\
path += Arm.moveto([10, 13, 15], 20)\
path += Arm.moveto([9.5, 0, 14.5], 20)\
path += [[91, 0, 0, 0]]\
\
#reset\
end=path[-2].copy()\
while not sum(end)==0:\
    for i in range(len(end)):\
        if end[i]>0: end[i]-=1\
        elif end[i]<0: end[i]+=1\
    path.append(end.copy())\
"
```

} LLM補全部分

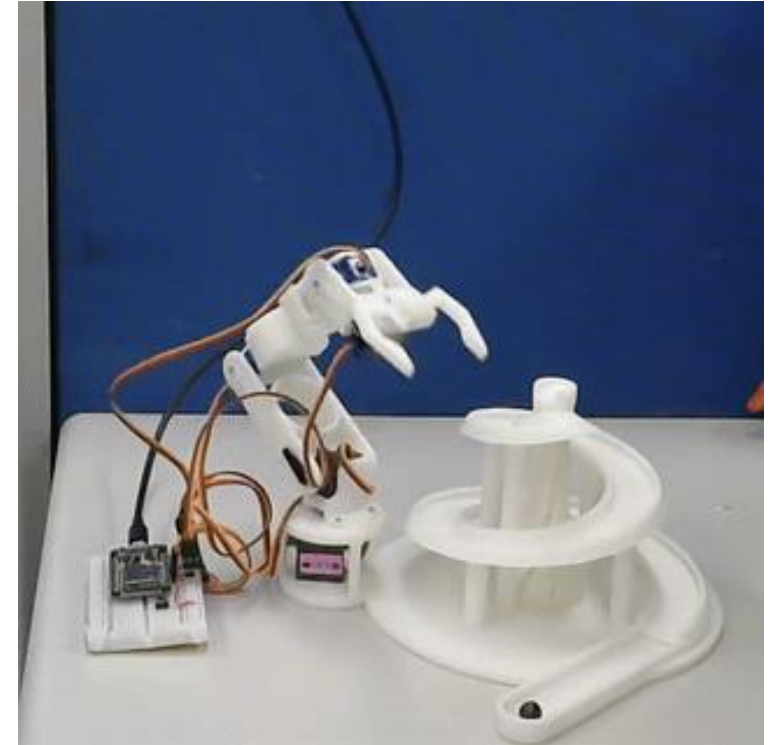


# 實驗結果

- 效果相當不錯，並且隨著語言模型技術的發展，路徑規劃的準確率也有顯著的提升。
- 需要將整個程式碼放入語言模型的輸入中以生成控制程式碼，這種方式過於繁瑣。
- 需要提供大量的前置條件，使用不便。

表 4.1: 實驗結果：Please catch the ball to the top of the tower.

大型語言模型	總次數	成功次數	失敗次數	成功率
GPT-4o	10	9	1	0.9
GPT-3.5 Turbo	10	4	6	0.4



成功的影片

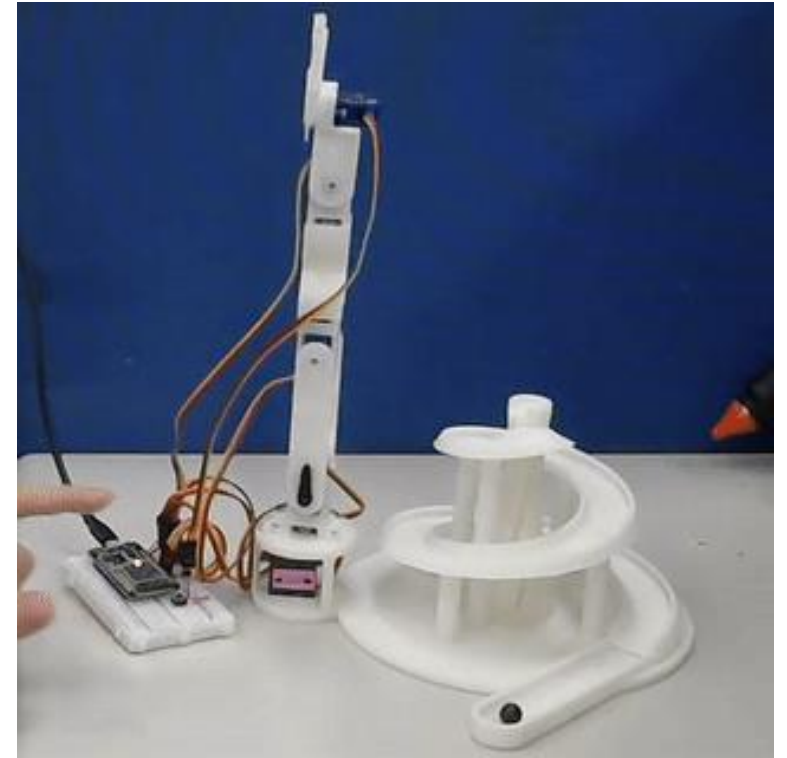


# 實驗結果

- 效果相當不錯，並且隨著語言模型技術的發展，路徑規劃的準確率也有顯著的提升。
- 需要將整個程式碼放入語言模型的輸入中以生成控制程式碼，這種方式過於繁瑣。
- 需要提供大量的前置條件，使用不便。

表 4.1: 實驗結果：Please catch the ball to the top of the tower.

大型語言模型	總次數	成功次數	失敗次數	成功率
GPT-4o	10	9	1	0.9
GPT-3.5 Turbo	10	4	6	0.4



失敗的影片



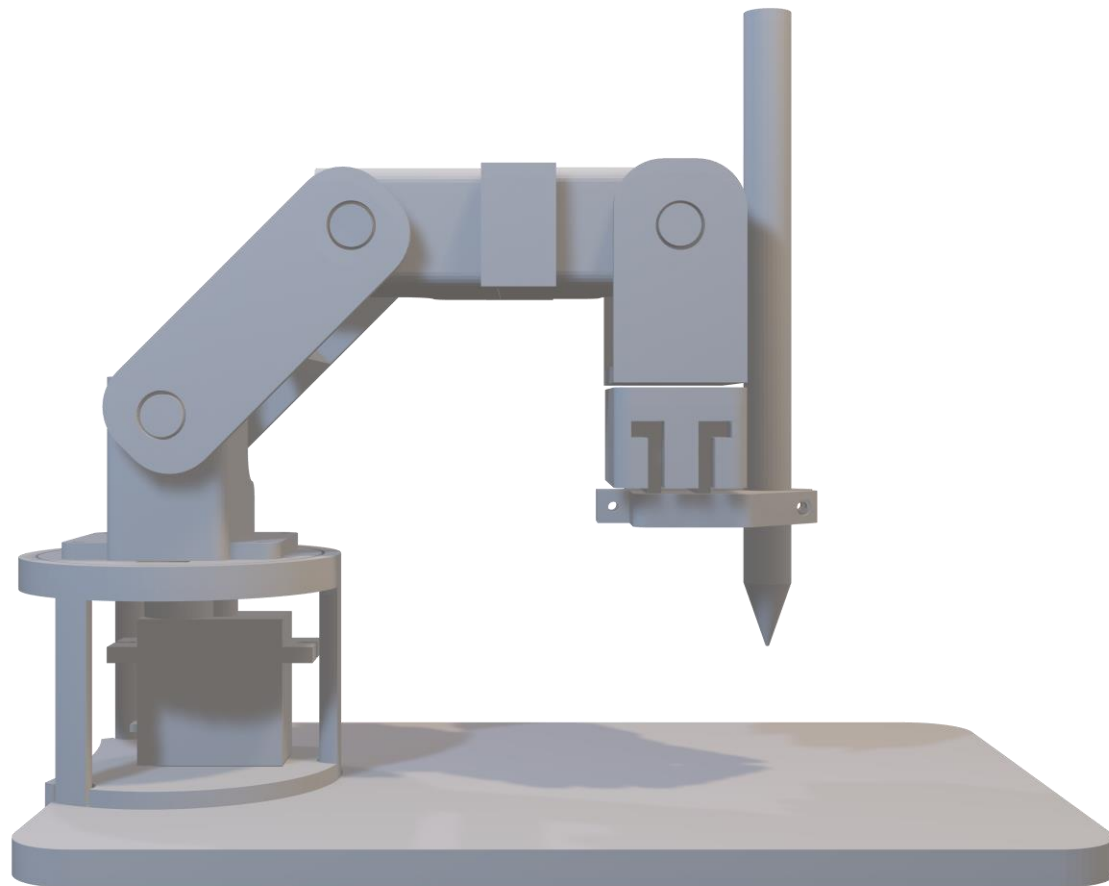


# 大綱

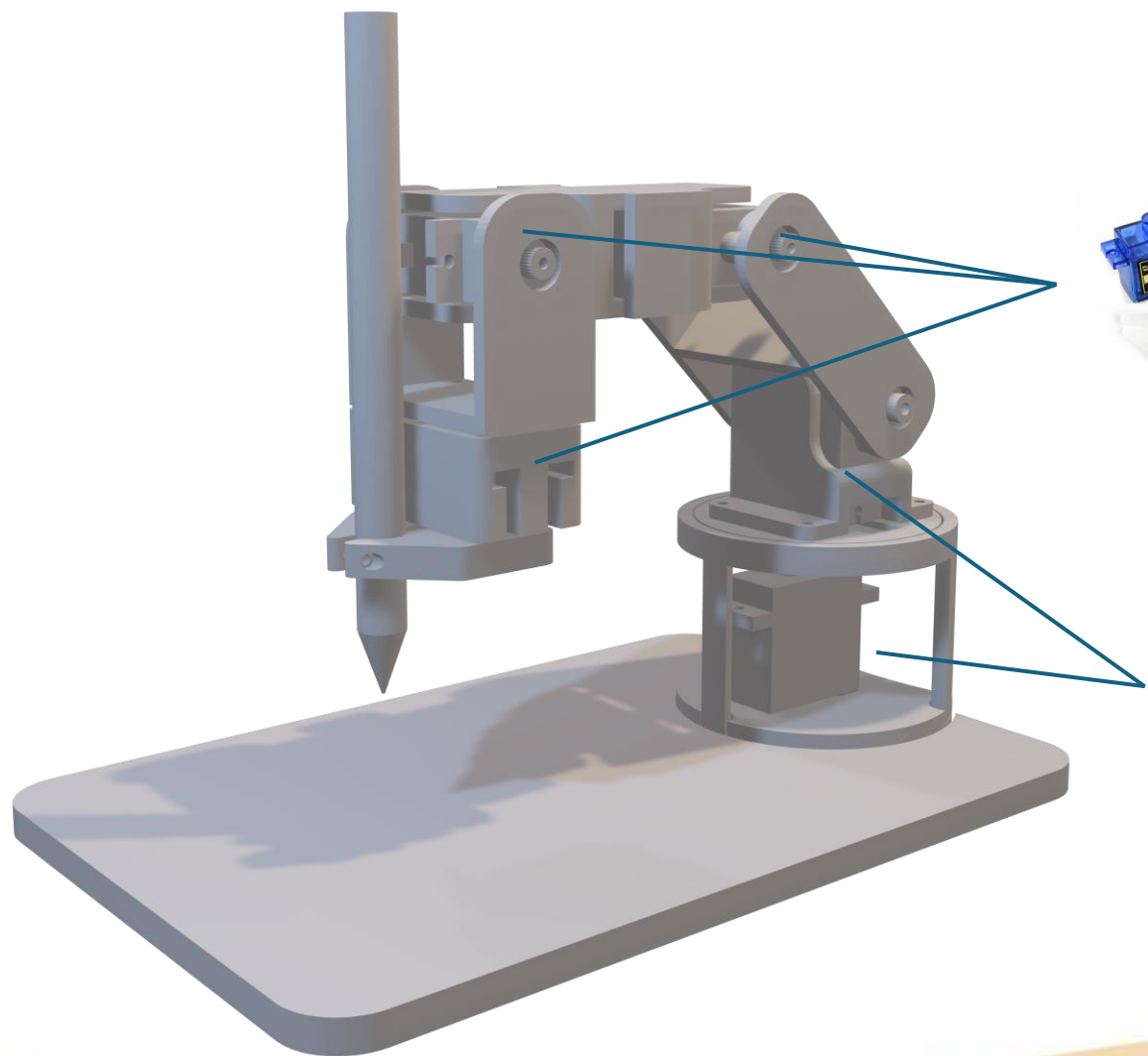
- 研究方法
- 實驗設計以及成果
  - 實驗一：機械臂的基本控制
  - 實驗二：將機械臂用於畫圖
    - 機械結構設計圖
    - 函數設計
    - 下達指令的格式範例
    - 實驗結果
  - 實驗三：機械臂在自動運輸車上的應用



# 機械結構設計圖



# 機械結構設計圖



SG90



ESP32 Doit-Devkit



MG90S



# 函數設計

- 簡化了路徑規劃，使大型語言模型在平面座標內進行規劃。
- 目標讓機械臂能夠在紙上或畫布上繪製出預定的圖案。
- 將動作指令簡化為以下函式：
  - `draw(from, to)`: 由變數`from`提供的座標，畫至變數`to`提供的座標。



# 下達指令的格式範例

- 架構使用上述JSON request範本。
- 指令：Draw a star.

```
{  
  "model": "model_type",  
  "prompt": "  
  "max_tokens": 100,  
  "temperature": 0.7,  
  "top_p": 1.0,  
  "n": 1,  
  "stop": null  
}
```

```
{  
  role: "user",  
  content :  
    "The following functions are available:\n  
    draw(from, to): Let the arm draw a straight line from from to to,\n    such as draw([0, 0], [5, 5]).\n    the limit is[0, 5]\n  
    Please help me use the above functions to control the robot arm,\n    and do not output other text other than the above functions.\n    (Use "," to separate each step)"  
},  
{  
  role: "user",  
  content: "Task: Draw a square."  
}
```

函式說明

指令要求





# 下達指令的格式範例

- 架構使用上述JSON request範本。
- 指令：Draw a star.

```
{  
  role="assistant",  
  content="draw([1, 1], [1, 4]),\  
          draw([1, 4], [4, 4]),\  
          draw([4, 4], [4, 1]),\  
          draw([4, 1], [1, 1])"  
}
```

} LLM直接呼叫函式

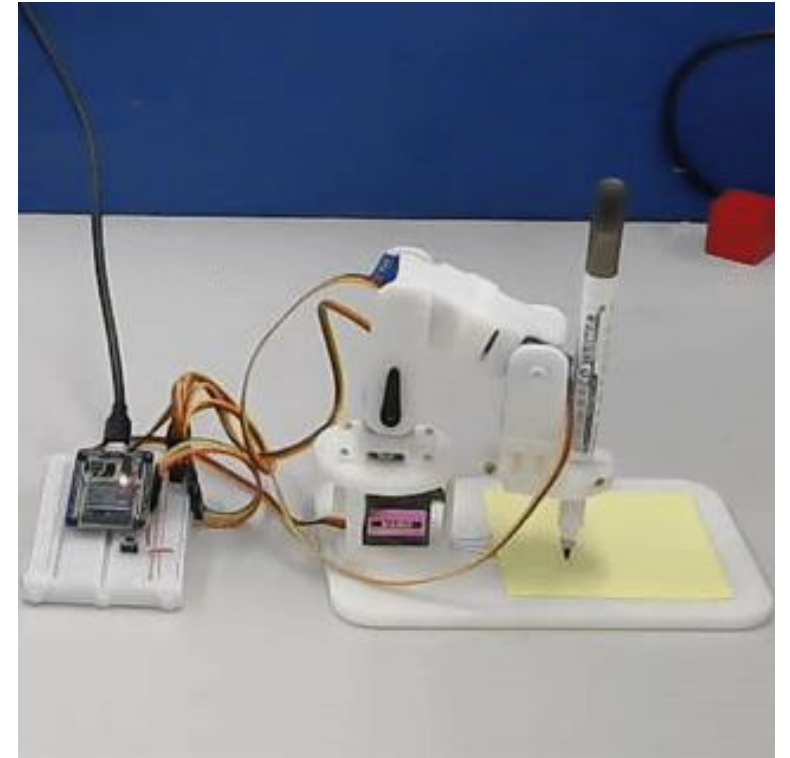


# 實驗結果

- 針對上一個實驗中存在的問題進行了改進，將三維空間路徑規劃簡化為平面路徑規劃。
- 不再需要提供冗長的程式碼與相關條件，只需提供特定的圖形
- 大幅減少了輸入和回傳的資料量，同時也保持了不錯的準確率。

表 4.2: 實驗結果：Draw a square.

大型語言模型	總次數	成功次數	失敗次數	成功率
GPT-4o	10	10	0	1
GPT-3.5 Turbo	10	10	0	1



成功的影片



# 實驗結果

- 針對上一個實驗中存在的問題進行了改進，將三維空間路徑規劃簡化為平面路徑規劃。
- 不再需要提供冗長的程式碼與相關條件，只需提供特定的圖形
- 大幅減少了輸入和回傳的資料量，同時也保持了不錯的準確率。

表 4.3: 實驗結果：Draw a triangle.

大型語言模型	總次數	成功次數	失敗次數	成功率
GPT-4o	10	10	0	1
GPT-3.5 Turbo	10	10	0	1



成功的影片

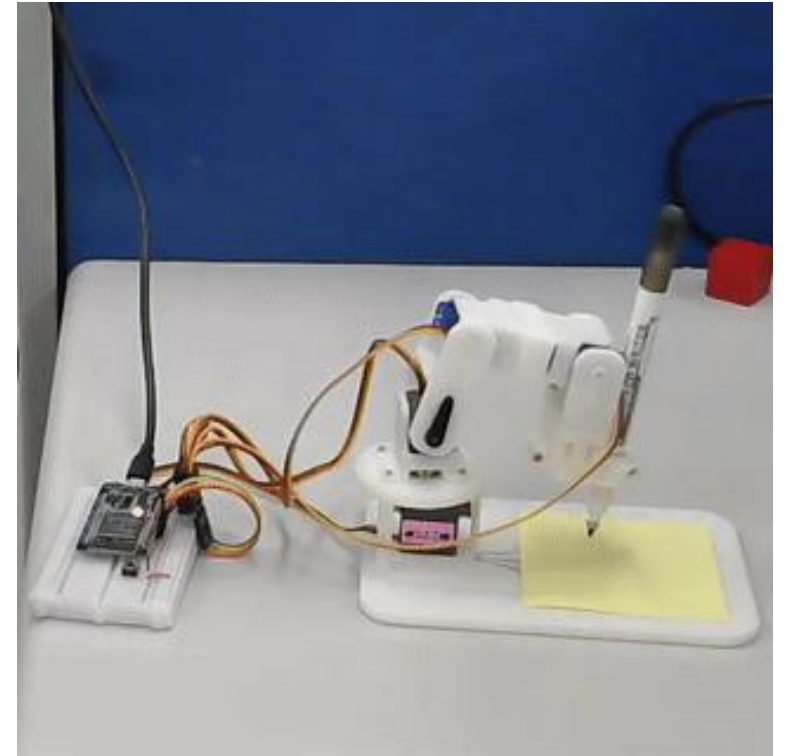


# 實驗結果

- 針對上一個實驗中存在的問題進行了改進，將三維空間路徑規劃簡化為平面路徑規劃。
- 不再需要提供冗長的程式碼與相關條件，只需提供特定的圖形
- 大幅減少了輸入和回傳的資料量，同時也保持了不錯的準確率。

表 4.4: 實驗結果：Draw a star.

大型語言模型	總次數	成功次數	失敗次數	成功率
GPT-4o	10	9	1	0.9
GPT-3.5 Turbo	10	5	5	0.5



成功的影片





# 大綱

- 研究方法
- 實驗設計以及成果
  - 實驗一：機械臂的基本控制
  - 實驗二：將機械臂用於畫圖
  - 實驗三：機械臂在自動運輸車上的應用
    - 機械結構設計圖
    - 函數設計
    - 下達指令的格式範例
    - 實驗結果



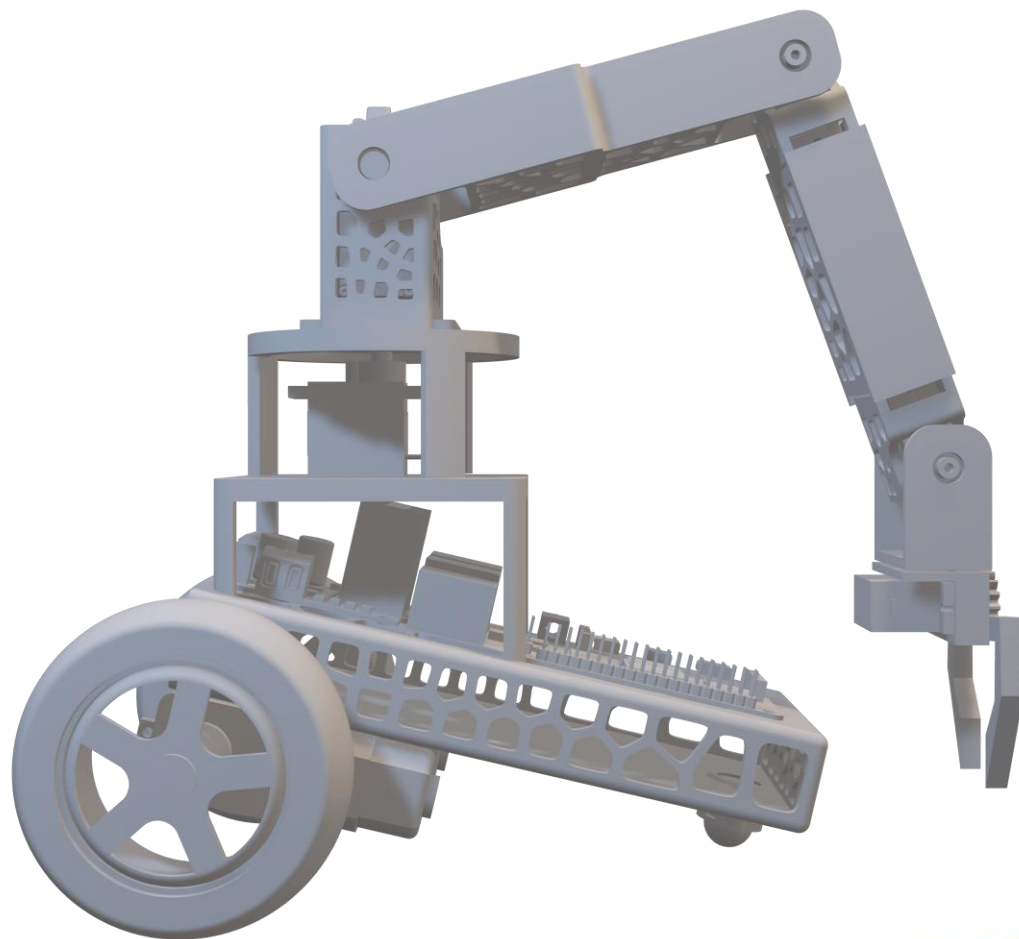
CHILab

結論與未來展望

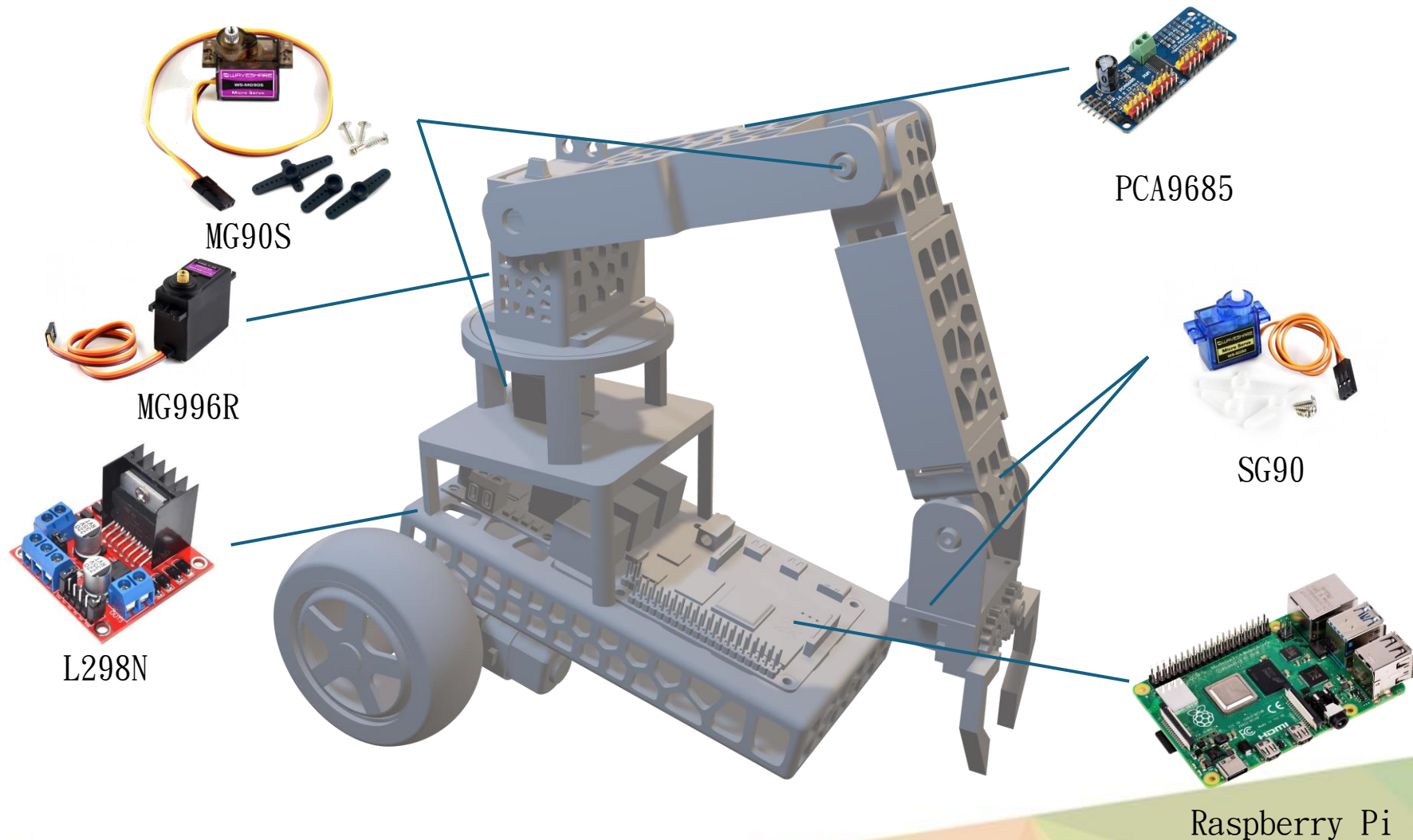
Computational Intelligence and Human-Computer Interaction Lab.



# 機械結構設計圖



# 機械結構設計圖



# 函數設計

- 將控制函式進行細分，並模組化。
- 大型語言模型能夠通過簡單的指令來控制機械臂和車輛的行動。
- 通過簡單的模組化指令讓機械臂和車輛能夠協同工作，完成更複雜的任務。
- 將動作指令簡化為以下函式：
  - `find(color)`: 配合相機定位，驅動輪子尋找動應顏色的方塊。
  - `aim(color)`: 配合相機定位，驅動機械臂瞄準對應顏色的方塊。
  - `grab()`: 驅動機械臂，抓取正下方的物品。
  - `reset()`: 驅動機械臂，回到初始位置



# 下達指令的格式範例

- 架構使用上述JSON request範本。
- 指令：Please grab the red block, and then grab the blue block.

```
{  
  "model": "model_type",  
  "prompt": "  
  "max_tokens": 100,  
  "temperature": 0.7,  
  "top_p": 1.0,  
  "n": 1,  
  "stop": null  
}
```



```
{  
  role: "user",  
  content :  
    "The following functions are available:\n\n    find(color): Let the robot look for a block of a specific color.\n    color option: red, blue.\n    aim(color): Let the robot aim at a block of a specific color.\n    color option: red, blue\n    grab: Make the robot arm grab the block and put it down.\n    reset: Return the robotic arm to its initial position\n    (needs to be executed before each aiming).\n\n    Please help me use the above functions to control the robot arm,\n    and do not output other text other than the above functions.\n    (Use ',' to separate each step)"  
},  
{  
  role: "user",  
  content: "Task: Please grab the red block, and then grab the blue block."  
}
```





# 下達指令的格式範例

- 架構使用上述JSON request範本。
- 指令：Please grab the red block, and then grab the blue block.

```
{  
  role="assistant",  
  content="find(red), aim(red), grab(), reset(), find(blue), aim(blue), grab()"  
}
```



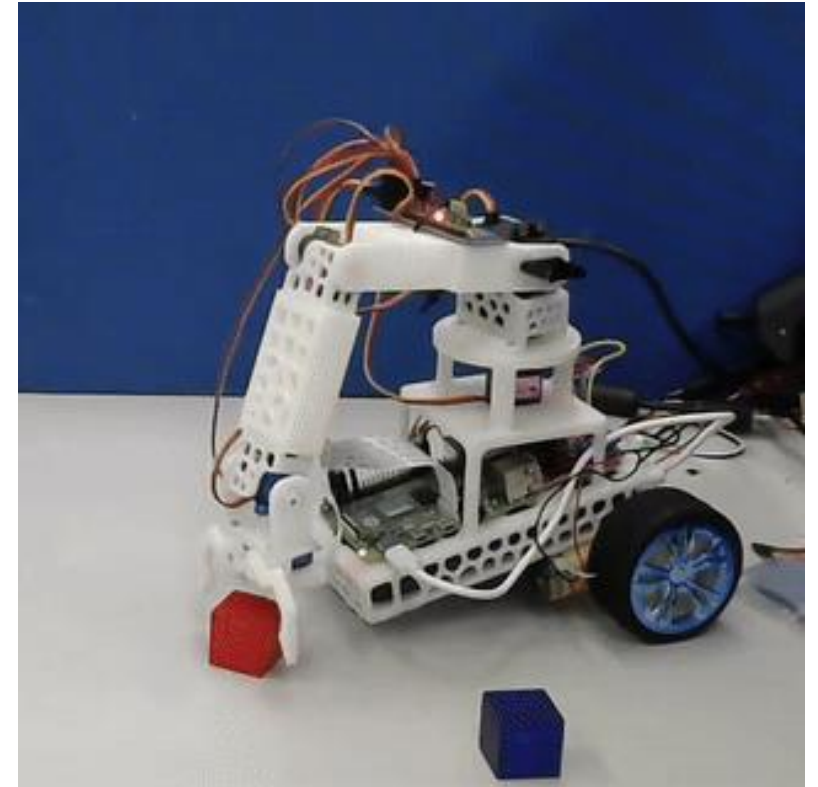


# 實驗結果

- 需要協調各個硬體才能順利完成，故將各個重要指令整理成更加簡化且直觀的獨立函式。
- 這種方法取得了非常好的成效，顯著提高了控制指令的生成效率和準確性。
- 由於連動的硬體較多，在呼叫控制函式時難免會發生一些機械性的失誤。

表 4.5: 實驗結果：Please grab the red block, and then grab the blue block.

大型語言模型	總次數	成功次數	失敗次數	成功率
GPT-4o	10	9	1	0.9
GPT-3.5 Turbo	10	7	3	0.7



成功的影片

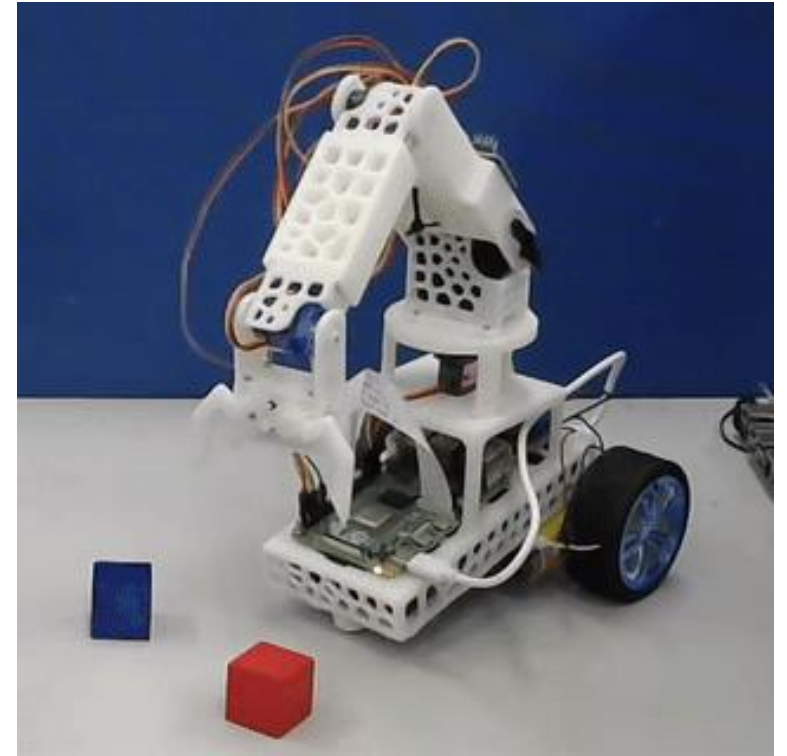


# 實驗結果

- 需要協調各個硬體才能順利完成，故將各個重要指令整理成更加簡化且直觀的獨立函式。
- 這種方法取得了非常好的成效，顯著提高了控制指令的生成效率和準確性。
- 由於連動的硬體較多，在呼叫控制函式時難免會發生一些機械性的失誤。

表 4.5: 實驗結果：Please grab the red block, and then grab the blue block.

大型語言模型	總次數	成功次數	失敗次數	成功率
GPT-4o	10	9	1	0.9
GPT-3.5 Turbo	10	7	3	0.7



失敗的影片



# 大綱

- 研究動機與目的
- 背景知識與相關研究
- 研究方法
- 實驗設計以及成果
- 結論與未來展望
  - 結論
  - 未來展望



# 結論

- 嘗試了機械臂在不同場景中的應用，並分析了其中的控制方法和結果。
- 透過設計機械結構、撰寫控制程式和協調大型語言模型等方式，成功的展示出了機械臂在基本控制、畫圖和自動運輸車上的應用。
- 使用大型語言模型進行程式碼生成在控制機器上有很大的潛力，大型語言模型不僅能夠自動生成可執行的控制程式碼，在擁有良好控制函式的前提下，即使連動較多硬體時也表現出色。





# 結論

在使用大型語言模型進行機器控制指令的自動化生成為前提下：

- 實驗一：實現了機械臂基本控制，通過給予大型語言模型範例程式，讓其補全其中缺失的路徑規劃部分，使機械臂能夠精確的完成指定任務。
- 實驗二：實現了簡單圖形的繪製，通過簡化指令格式和優化控制函數，展示了此方法在精細操作方面的潛力。
- 實驗三：實現了機械臂在自動運輸車上的應用，儘管存在一定的機械性失誤，但通過進一步優化硬體間的協調機制，可以顯著提升系統的穩定性和精確度。





# 未來展望

- 希望進一步優化機械臂的控制系統，尤其是在硬體協調方面。開發更便利的互動方式，例如通過語音指令直接控制機械臂，使操作更加直觀和方便。
- 利用 3D 列印技術來製作更多客製化的機械臂及其零部件，並協調整合更多的硬體設備。快速設計和生產符合特定應用需求的機械臂，從而滿足不同行業和應用場景的需求。
- 開發更多模組化的機械臂控制函式庫，這些模組化函式庫將包含多種預先定義的操作和功能，使得開發者可以輕鬆地構建和定制機械臂的操作流程，進一步提升開發效率和彈性。



感謝聆聽



Computational Intelligence and Human-Computer Interaction Lab.