1. 附上本次實驗 Q1 4 個 terminals 的結果圖，Q2 溫溼度接收的結果圖

Q1

```
pi@raspberrypi:~ $ sudo python Lab2_1_server.py
Worker 1121 is running ...
Compute 82 + 81 and send response
Compute 69 + 86 and send response
Compute 40 + 4 and send response
```

```
pi@raspberrypi:~ $ sudo python Lab2_1_server.py
Worker 1152 is running ...
Compute 47 + 80 and send response
Compute 82 + 78 and send response
Compute 81 + 37 and send response
```

```
pi@raspberrypi:~ $ sudo python Lab2_1_server.py
Worker 1182 is running ...
Compute 11 + 64 and send response
Compute 95 + 86 and send response
Compute 57 + 41 and send response
```

```
pi@raspberrypi:~ $ sudo python Lab2_1_client.py
Compute 11 + 64
 = 75 ( from worker 1182)
Compute 47 + 80
 = 127 ( from worker 1152)
Compute 82 + 81
 = 163 ( from worker 1121)
Compute 95 + 86
 = 181 ( from worker 1182)
Compute 82 + 78
 = 160 ( from worker 1152)
Compute 69 + 86
 = 155 ( from worker 1121)
Compute 57 + 41
 = 98 ( from worker 1182)
Compute 81 + 37
 = 118 ( from worker 1152)
Compute 40 + 4
 = 44 ( from worker 1121)
```

Q2

```
pi@raspberrypi:~ $ sudo python Lab2_2_client.py
Start!!

temp = 27
humidity = 61
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 62
temp = 27
humidity = 61
avg temperature: 27.0
avg humidity: 61.8
```

2. 考慮本次 Q1 和 Q2 的實驗流程下來，請問 ZMQ 的 client 與 server 間的 connect、bind 順序不同的話會對實驗內容有影響嗎? 請說明原因。

兩種方法都可以。ZMQ 有別於傳統使用的 Connection-oriented socket，它不須考慮連線順序。不管是先 connect 還是先 bind，只要不能建立連線，ZMQ 就會自動重試，直到成功為止。這樣就只要考慮怎麼連接還有對方的位址是什麼就好。

而在一般情況下，相對穩定和位址較清楚的節點會用 bind，較為動態易變的節點則會用 connect。因此 server 端較常使用 bind，而 client 端較常使用 connect。

3. 列舉 ZMQ 的三種常見模式的優缺點。

Request-Reply 模式

優點:

- 每個發送的訊息都會被驗證(透過接收 Reply)。
- 是一種簡單的通信模式，易於理解和使用，適合許多應用場景。
- 易擴展，不需太多的程式碼更改。

缺點:

- 每個 client 在得到 Reply 之前無法發送下一個 Request。
- 可能在有大量請求時導致性能瓶頸。

Publish-Subscribe 模式

優點:

- Subscriber 可以接收自己感興趣的訊息就好。
- 一個 Subsrciber 連接多個 Publisher 時，會均衡的從每個 Publisher 收訊息，不會出現一個 Publisher 淹沒其他 Publisher 的情況。

缺點:

- 中途離開或比較晚加入的 Subsrciber 會丟失一部份訊息。
- 可靠性較低，訊息丟失較難避免。

Parallel-Pipline 模式

優點:

- 因為是平行處理，適合用在高 throughput 和低延遲需求的應用程序。
- 除非節點意外段開連接，不然不會丟棄訊息，是最可靠的模式。
- 可伸縮，節點可以在任意時候連接。
- 將任務公平地分配給多個 workers。

缺點:

- 不適用於所有應用程序，對於簡單、線性的任務，會有更簡單適合的模式可以選擇。

4. 試想 ZMQ 的三種常見模式 Request-Reply、Publish-Subscrible、Parallel-Pipeline 分別有哪些應用? (愈詳細且創新分數越高)

Request-Reply 模式

智能家居系統可以使用 Request-Reply 模式，設備可以向中央控制系統發出請求以執行不同任務。像是燈光控制、冷氣溫度控制等等。

Publish-Subscribe 模式

即時的股市數據更新可以使用 Publish-Subscribe 模式，Publish 端發布市場數據給投資者(Subscribe 端)，而 Subscribe 端訂閱想看的股票或指數的數據更新，以利他們更新其投資策略。

Parallel-Pipline 模式

在機器學習模型中，可以構建一個 pipeline，每個 stage 專門處理不同的任務，以加速模型訓練和計算速度。

5. 本次實驗心得，你學到了什麼東西?

這次實驗學到了 ZMQ 這個可伸縮的分散式或並行應用程式設計的高效能非同步訊息庫。它有連線先後順序無關、自動重連的機制。而從 Q1 可發現，request 是依序分配給三個 client。REQ 端會將訊息用 Round-robin 的方式分給所有連線。不論有多少連線，都會照這個規則分配，此即自動負載平衡。而為了避免在尚未完成所有連線前就已經將 request 大量分配給少數完成的連線的情況發生，通常會加上 time.sleep()來確保所有 worker 都連接上。另外 ZMQ 也提供在一筆訊息中傳送多段資料的功能，send_multipart()和 recv_multipart()作用即為如此。這麼一來可以不用擔心通訊協定，只需專心處理訊息即可。

而第二個實驗中我一開始忘記幫 Subscribe 訂閱，所以完全接收不到任何消息，而這也讓我從實例中學到如果沒有 Subsrcibe 端的話，發送的消息會被直接丟棄。

我自己覺得這次的實驗沒有很難，只需要寫點程式碼就好，也能很快就能理解兩種模式的應用。

6. Reference

https://zh.wikipedia.org/zh-tw/%C3%98MQ
https://juejin.cn/post/6844903463671824391
https://juejin.cn/post/6844904061712465928
https://juejin.cn/post/6844904100849516551
https://blog.ez2learn.com/2011/12/31/transport-lib-of-new-era-zeromq/
https://www.syscom.com.tw/ePaper_New_Content.aspx?id=404&EPID=201&TableName=sgEPArticle