

智慧停車系統

Smart Parking System

106000266 劉嶧岷、105042004 林立雯

2021/06/08

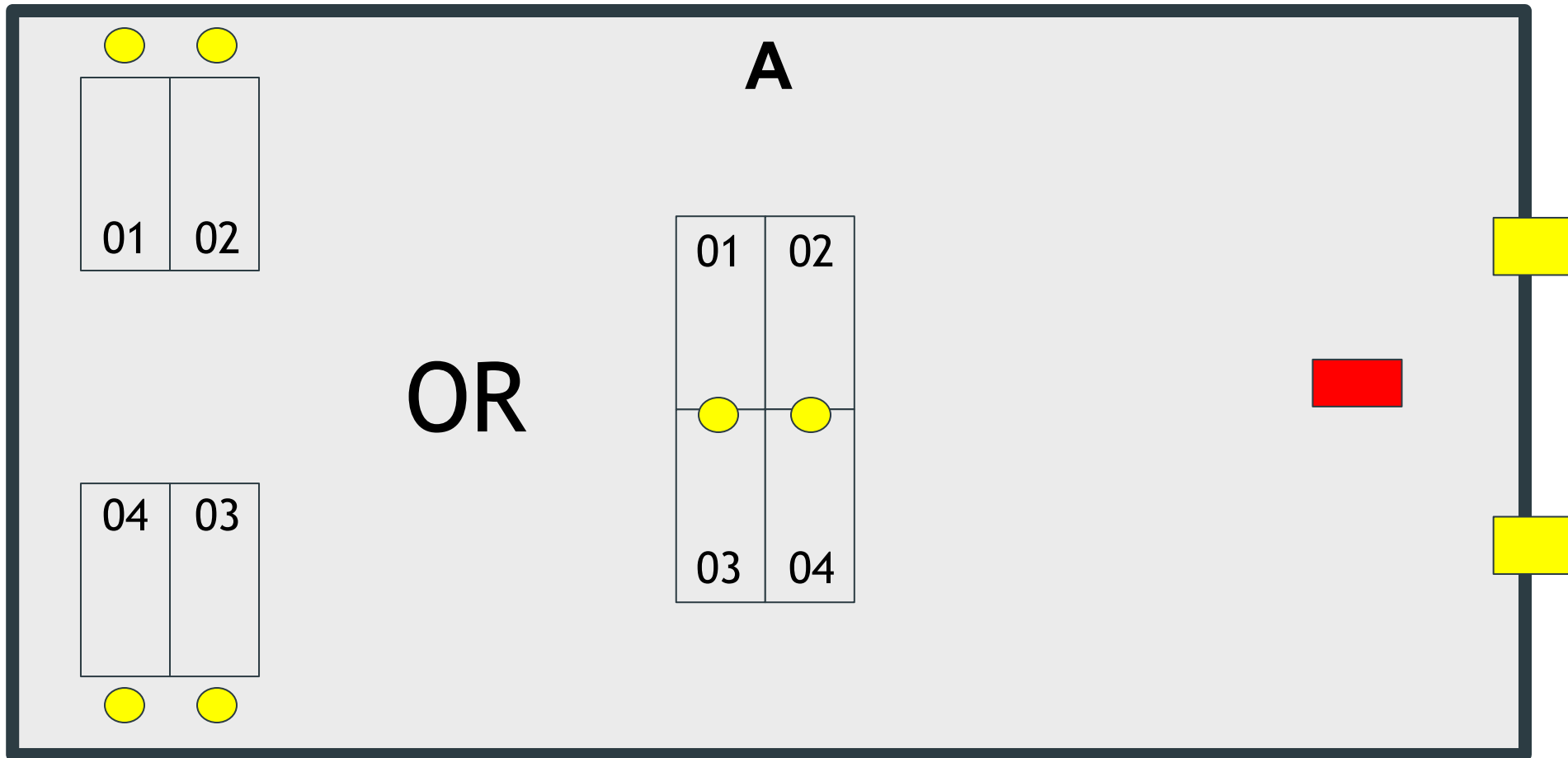
Outline

- 1. Problem Statement and Motivation**
- 2. System Architecture**
- 3. Implementation**
- 4. Results and Evaluation**
- 5. Lesson Learnt**
- 6. Future Work**
- 7. Conclusion**
- 8. References**

Problem Statement and Motivation

- ▶ 駕駛常常需要花許多時間在賣場、觀光區找尋空停車位, 因此想要建立一個物聯網系統解決此問題, 減少不必要的繞行時間。
- ▶ 建立此系統除了解決**駕駛停車問題**之外, 也提供**停車場管理者**一個便利的渠道, 減少違規停車狀況。
- ▶ 我們希望在兩個層面建立此系統:
 - ▶ 停車現場:**監控並即時上傳車位狀況**至網頁供駕駛人瀏覽、停車排隊系統防止插隊
 - ▶ 後臺:分析停車場流量、平均停車時間

System Architecture



超音波感測器
(HC-SR04)

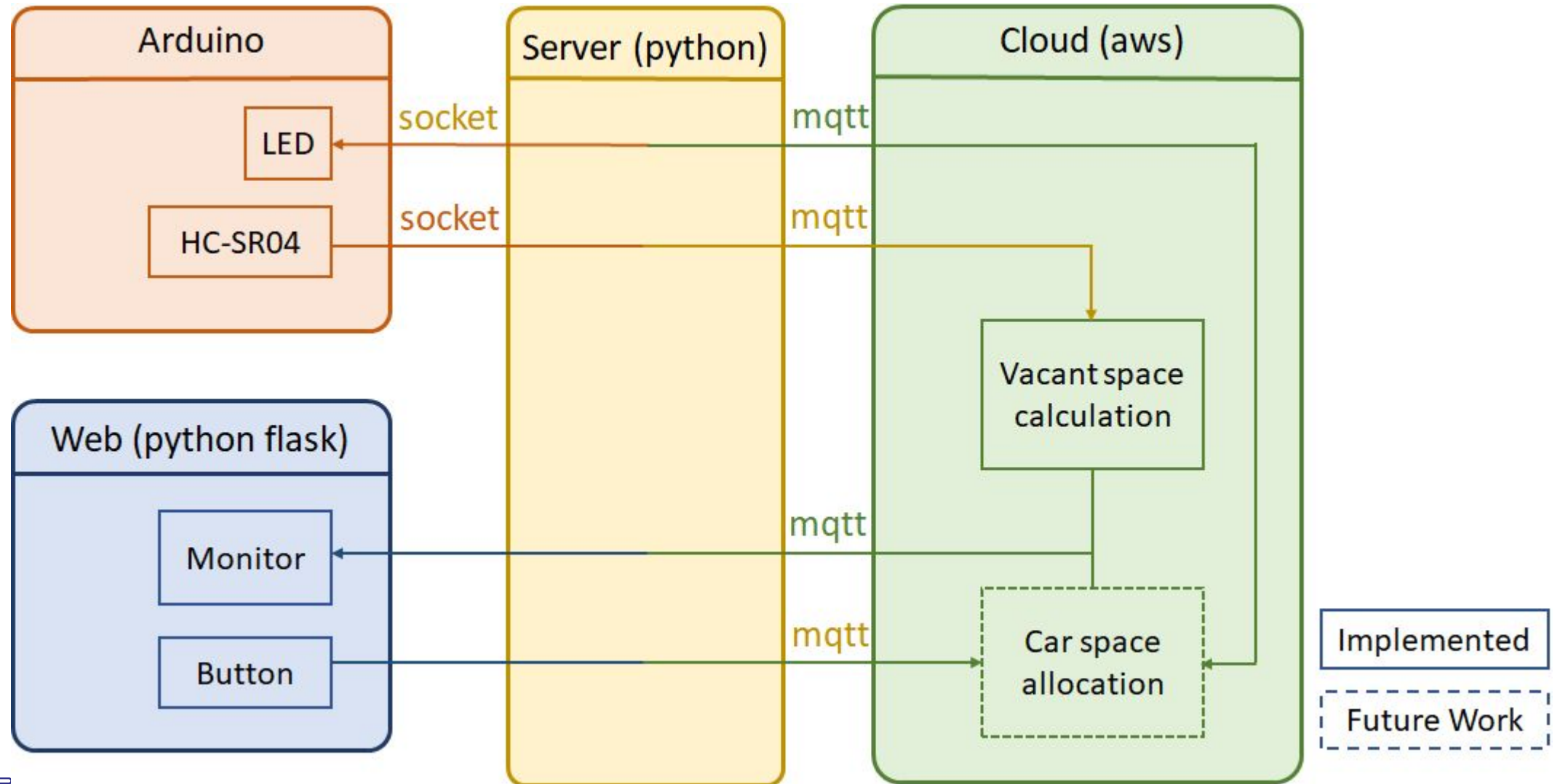
fence 

pillar 

parking lot 

car 

System Architecture



Implementation

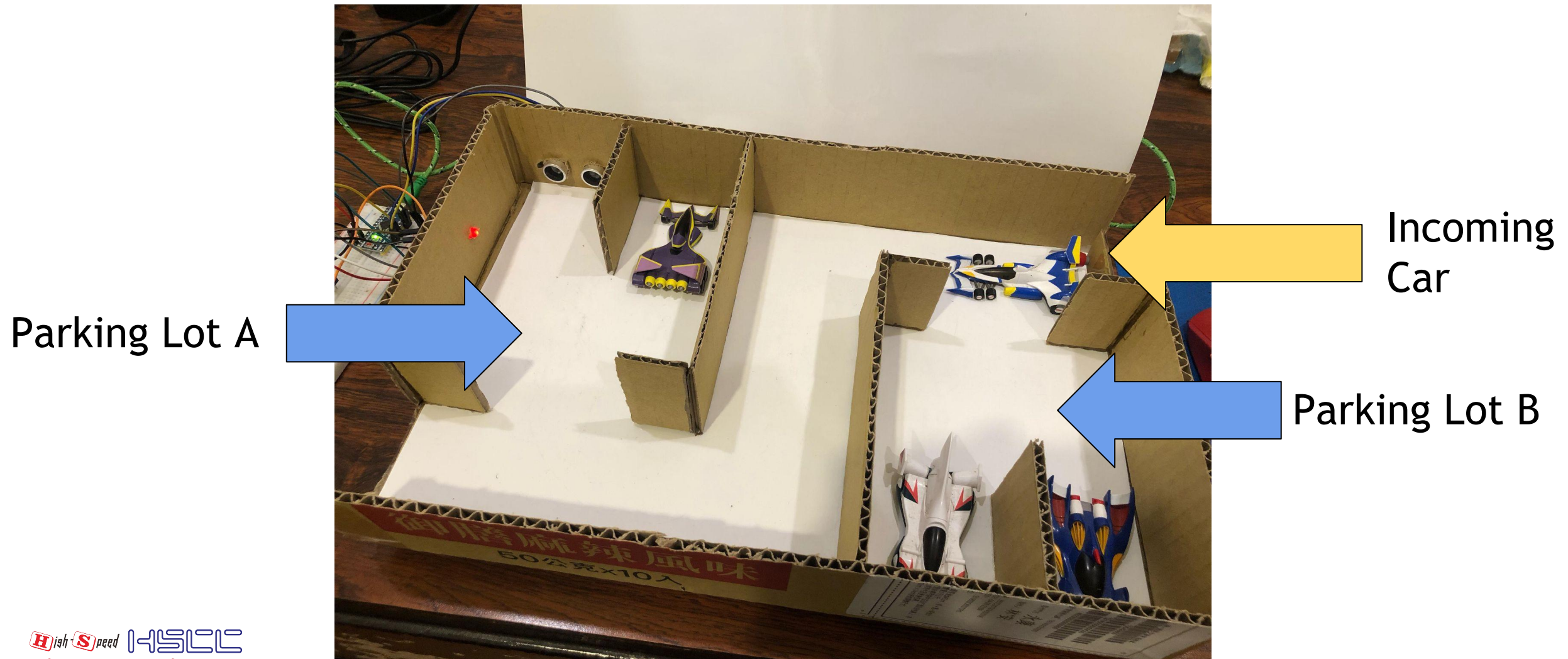
- ▶ Hardware
- ▶ Parking space:
 - a. Super sonic distance sensor for each parking space
 - b. All sensor connected to Arduino
 - c. ESP8266 send available signal to main system
 - d. LED (light = 1, prohibit other drivers to park in; light = 0, the driver has arrived)
- ▶ Parking lot entrance:
 - a. Monitor display map
 - b. Print qr code for driver to unlock reserved parking space

Implementation

- ▶ Software
- ▶ Parking space:
 - a. If distance sensor **detect** distance < 10, update space state on server
- ▶ Parking lot entrance:
 - a. Driver press **button** to reserve space
 - b. Show **parking lot map**
- ▶ Webpage
 - a. Python Flask

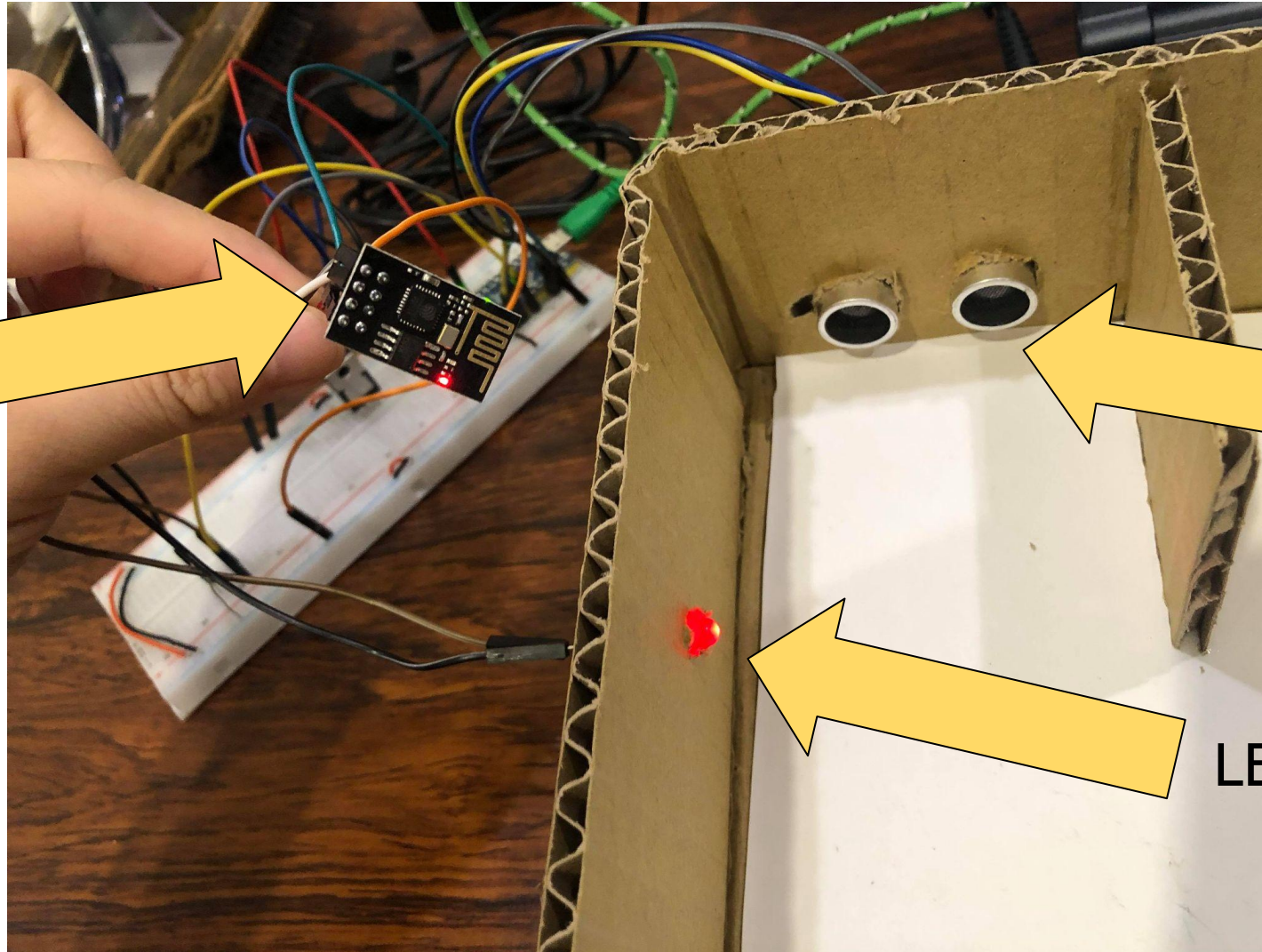
Results and Evaluation

- ▶ Our Smart Parking System prototype



Results and Evaluation

▶ Sensor and modules close-up



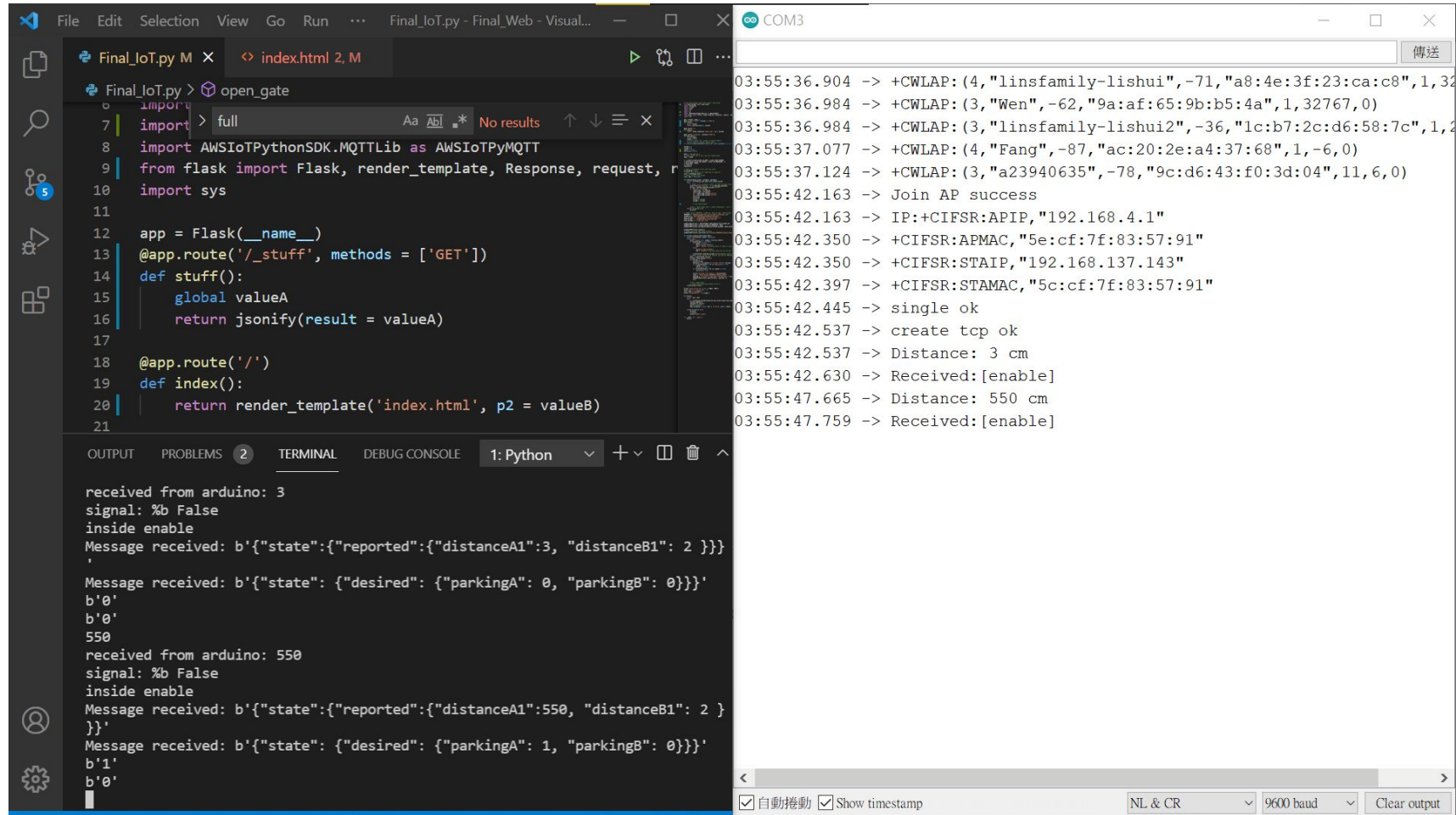
ESP8266:
Communication
between arduino
and server

HC-SR04:
Detect available
space

LED: Parkable signal

Results and Evaluation

► Communication Between Arduino IDE, Python Server and AWS



The screenshot displays a Python IDE with a file named `Final_IoT.py` and a terminal window showing serial communication data. The code in the IDE is as follows:

```
Final_IoT.py > open_gate
7 | import > full
8 | import AWSIoTPythonSDK.MQTTLib as AWSIoTPyMQTT
9 | from flask import Flask, render_template, Response, request, r
10 | import sys
11 |
12 | app = Flask(__name__)
13 | @app.route('/_stuff', methods = ['GET'])
14 | def stuff():
15 |     global valueA
16 |     return jsonify(result = valueA)
17 |
18 | @app.route('/')
19 | def index():
20 |     return render_template('index.html', p2 = valueB)
21 |
```

The terminal window shows the following serial communication data:

```
03:55:36.904 -> +CWLAP: (4, "linsfamily-lishui", -71, "a8:4e:3f:23:ca:c8", 1, 32
03:55:36.984 -> +CWLAP: (3, "Wen", -62, "9a:af:65:9b:b5:4a", 1, 32767, 0)
03:55:36.984 -> +CWLAP: (3, "linsfamily-lishui2", -36, "1c:b7:2c:d6:58:7c", 1, 2
03:55:37.077 -> +CWLAP: (4, "Fang", -87, "ac:20:2e:a4:37:68", 1, -6, 0)
03:55:37.124 -> +CWLAP: (3, "a23940635", -78, "9c:d6:43:f0:3d:04", 11, 6, 0)
03:55:42.163 -> Join AP success
03:55:42.163 -> IP:+CIFSR:APIP,"192.168.4.1"
03:55:42.350 -> +CIFSR:APMAC,"5e:cf:7f:83:57:91"
03:55:42.350 -> +CIFSR:STAIP,"192.168.137.143"
03:55:42.397 -> +CIFSR:STAMAC,"5c:cf:7f:83:57:91"
03:55:42.445 -> single ok
03:55:42.537 -> create tcp ok
03:55:42.537 -> Distance: 3 cm
03:55:42.630 -> Received:[enable]
03:55:47.665 -> Distance: 550 cm
03:55:47.759 -> Received:[enable]
```

The terminal window also shows the following output:

```
received from arduino: 3
signal: %b False
inside enable
Message received: b'{"state":{"reported":{"distanceA1":3, "distanceB1": 2 }}}'
Message received: b'{"state": {"desired": {"parkingA": 0, "parkingB": 0}}}'
b'0'
b'0'
550
received from arduino: 550
signal: %b False
inside enable
Message received: b'{"state":{"reported":{"distanceA1":550, "distanceB1": 2 }}}'
Message received: b'{"state": {"desired": {"parkingA": 1, "parkingB": 0}}}'
b'1'
b'0'
```

Demo:

<https://www.youtube.com/watch?v=zEWIbAGMJQk>

Results and Evaluation

- ▶ We have built a **sandbox** to simulate the parking lot.
- ▶ Although there is only one ultrasonic sensor, if the companies like our idea, they can **use multiple ultrasonic sensor** and copy the action of that ultrasonic sensor to expand their scenario.
- ▶ LED is one of the tools to inform the people not to park the area without authorization. Companies can use other mechanism like auto-gate, spike, scalable-pillar and so on to take over LED.

Lesson Learnt

- ▶ Multithread and Multiprocessing, learning from implement multiple process with sharing same variables
- ▶ New Programming Language(Python) to write a webpage (besides C# and Javascript)
- ▶ Combine Python Flask and TCP Connection
- ▶ IoT system design to help users resolve problems

Future Work

- ▶ Analysis:
 - a. Analyze car flow data, which parking lot is mostly used
 - b. Average parking time
 - c. Is parking space sufficient?

- ▶ Hardware Equipment
 - a. Merge with Parking Fee System
 - b. QR code scanning or better idea to reserve parking space

- ▶ Implement Database System

Conclusion

- ▶ In this final project, we have faced multiple difficulties which we didn't face before, like modules failure, solving multi processes sharing same resources, web page communicate with arduino IDE and so on, especially worse pandemic than last year. Despite that, we tried to solve them in our own way.
- ▶ Thanks to this final project, besides the knowledge of IoT, we need to revise what we have learnt before (Thread) and enhanced it in our memory.
- ▶ Finally, Thank You, Professor Sheu and lecturers.

Reference

- ▶ Python Multiprocessing
- ▶ <https://docs.python.org/3/library/multiprocessing.html#sharing-state-between-processes>
- ▶ Dynamic Update Variable Using Flask, by Pratik Jain
- ▶ <https://www.youtube.com/watch?v=vKhVvbhNOig>
- ▶ Car Parking System
- ▶ <https://create.arduino.cc/projecthub/embeddedlab786/car-parking-system-ef9fc1>