Introduction to IoT Project Final Report (黃則睿，0516230)

Topic: Indoor Location Tracking System for Shopping Customers, Hospital Patients or Preschool Children

1. **Objective:**
   - Implement an indoor location tracking system using wireless signal strength measurement.
   - Tracking targets are required to connect to Raspberry Pi via Wi-Fi Connection. This connection triggers Raspberry Pi to ping those target devices regularly to generate traffics in the air.
   - Several ESP32s sniff packets in the air and measure their RSSI, send RSSI to Raspberry Pi via serial ports.
   - Raspberry Pi receives each packet's RSSI, transform RSSI into distance value, conduct trilateration algorithm to get target location, and upload data to the cloud server.
   - Cloud server receives the data from Raspberry, store/analyze the data, and provide http service in GUI for users to observe target's location.
2. **Circumstances:**
   - In nursery, babysitters can monitor the children's location via Wi-Fi tag with each child, notify the babysitter if one leaves the safe field.
   - For commercial strategists, they can analyze the costumer's behavior in supermarket or coffee shop such as Starbucks, and conduct improvement to maximize the enterprise's benefit.
3. **Spec:**

   3.1. Previous Work (Midterm Demo):
   - Hardware requirement:
     - A Raspberry Pi 4B
     - An ESP32-WROOM-32 development kit
     - A micro USB cable
     - Two RJ-45 cables
     - A Wi-Fi access point with one ethernet port and one WAN port for Internet connection
     - A Laptop (execute programs in Raspberry Pi)
   - Software:
     - C language for firmware in ESP32
     - Python (write analyzing program on Raspberry Pi)
   - External Resources:

- MediaTek Cloud Sandbox (Show the information)

3.2. Current Work (Final Demo):

- Hardware requirement:
  - A Raspberry Pi 4B
  - 4 ESP32-WROOM-32 development kits
  - 4 micro USB cables with an extension cable
  - Two RJ-45 cables
  - A Wi-Fi access point with one ethernet port and one WAN port for Internet connection
  - A laptop for managing the system
  - A cloud server deployed with Internet access
  - Several devices with Wi-Fi connectivity as targets
  - User's devices for visiting web page
- Software:
  - C language for firmware in ESP32
  - Python for analyzing data, RESTful API and web server
  - HTML, JavaScript for web page
- External Resources:
  - AWS EC2 or self-deployed server

4. System Design & Module Implementation

The figure in the next page illustrates the system architecture.

4.1. ESP32

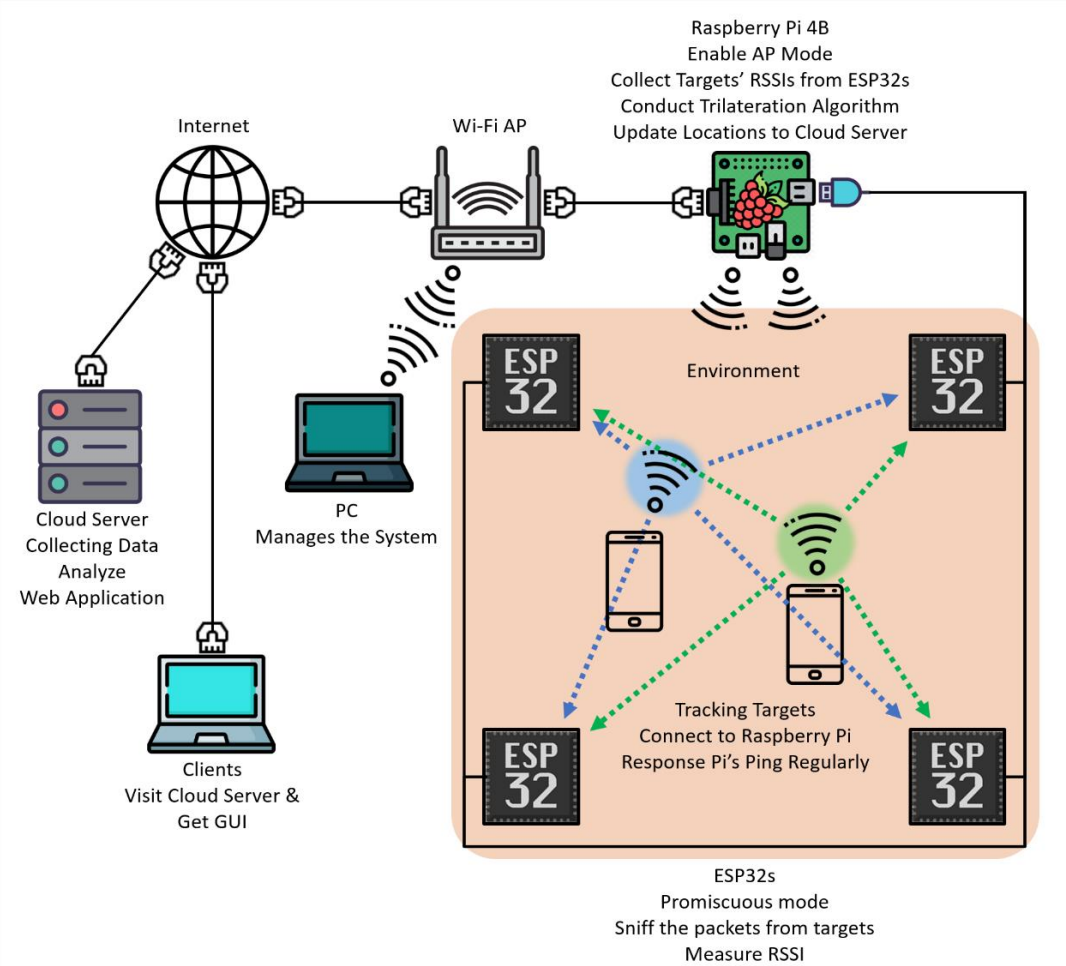
- ESP32 development kits are capable of conducting various wireless functions programmed, compiled, and flash by users. In this project, a C program that enables Wi-Fi promiscuous mode to sniff the packets in the air is written. The flashed ESP32s measure Wi-Fi signal's RSSI sent by target devices and send information to Raspberry Pi via serial with USB connection.

4.2. Tracking Targets

- Tracking Target can be customer's mobiles, Wi-Fi tags or any device with Wi-Fi connectivity. All tracking target are required to connect to Raspberry Pi to generate packets and register to the system.

4.3. Raspberry Pi 4B

- The wireless LAN interface in Raspberry Pi is set to AP mode. A program will be executed in Pi with following behaviors:
  - Detect the target devices that connects to Raspberry Pi for every 2 seconds.
  - Ping every connected device regularly.



- Receive the packet data (source/destination MAC address) and RSSI from ESP32s.
- Filter out the packets information that does not produced by target devices, average the all RSSI values received within 2 seconds and calculate the distance values.
- Calculate each target device's location with trilateration algorithm. In this demo, we use Least Square Algorithm to conduct prove of concept.
- Upload the location data to cloud servers via RESTful API.

4.4. Wi-Fi AP & Laptop

- The AP provides the system to access to the Internet and enables Raspberry Pi to access the Internet to upload location data and provide targets Internet access. The laptop connecting to AP controls (execute programs on) Raspberry Pi via Security Shell.

4.5. Cloud Server

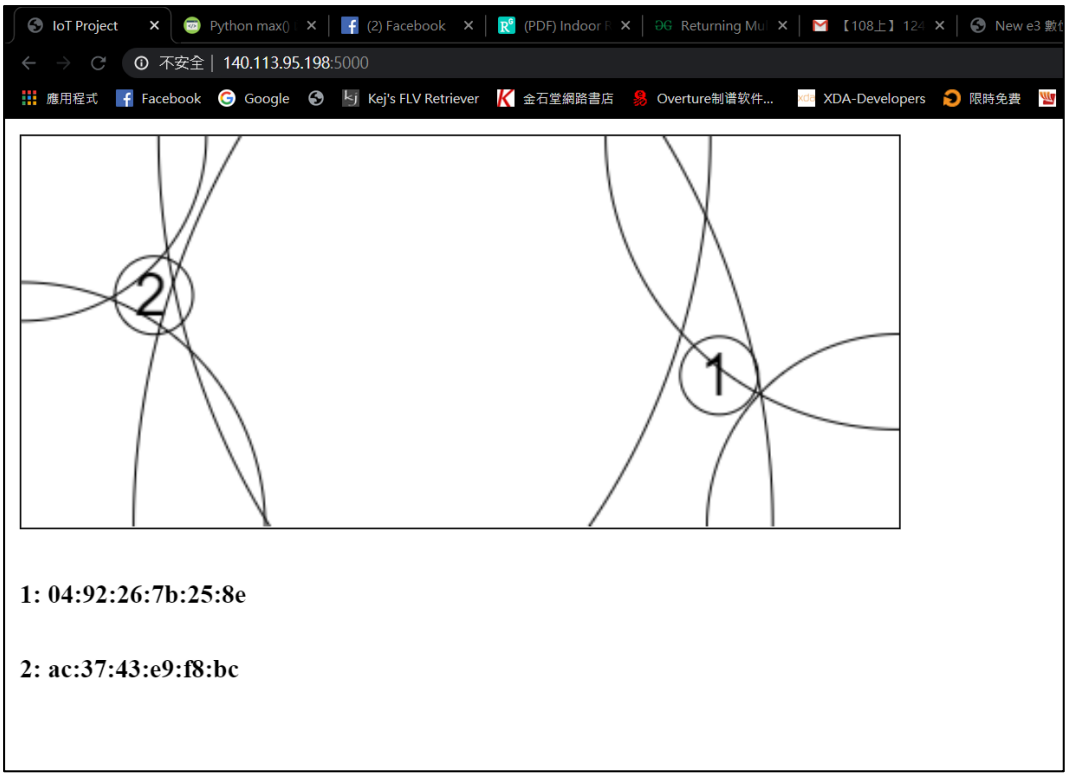
- The cloud server can be deployed everywhere in the Internet. The server provides RESTful API to enable the Raspberry Pi (or more than one) to update target device's location. Also, cloud server can be featured in storing data,

conduct statistics, and provide user function. For users to observe the target's location, the cloud server provides websites and show each target's location in GUI.

5. Demonstration

Following picture illustrates the actual deployment of system in a room of dorm 13. 4 ESP32s are set in each corner of the room.

Next, two roommate's cellphones are connected to the Raspberry Pi AP. Once the Raspberry Pi detects these two devices, each cellphone's location in the room will be frequently uploaded to the cloud server. Observers can visit the website provided by cloud server, and targets' location will show on the website in real-time.



6. Faced Challenges and Future works:

- The precision of the location tracking is not high enough, due to following reasons:
  - The antennas of target devices are not omnidirectional, and people holding the devices can undermine the signal.
  - The ESP32s' built-in antenna is such simple that different angles between the devices and ESP32s can significantly diverse the signal strength evaluation.
  - Different target devices have their signal strength (Measured Power), a formula matches a specific device may not work properly on another device.

  To enhance the precision for all devices, the system brings the different Measured Powers as parameter into the distance formula and find the optimal result which can make the circumferences of predicted ranges of two diagonal ESP32s overlap reasonably. Other methods for improving precision can be done in the future.
- More components and functions can be applied in the system, such as notifying target device's user some message, send alert to observers if a device is at dangerous location, etc.

7. Reference

- MCS tutorial: https://oranwind.org/-raspberry-pi-tou-guo-python-chuan-song-wen-shi-du-zi-xun-dao-mediatek-cloud-sandbox-mcs-api/
- RSSI-Distance Algorithm: https://iotandelectronics.wordpress.com/2016/10/07/how-to-calculate-distance-from-the-rssi-value-of-the-ble-beacon/
- Main Program: https://www.quora.com/How-can-we-know-how-many-users-are-connected-to-our-WiFi-router
  https://my.oschina.net/jamesflyit/blog/614820
- ESP32 Program reference: https://github.com/ESP-EOS/ESP32-WiFi-Sniffer/blob/master/WIFI_SNIFFER_ESP32.ino
- RESTful with flask: https://blog.taiker.space/python-shi-yong-python-he-flask-she-ji-restful-api/