PSOT - Public Space Occupancy Tracking

Group A - IoT Project Final Presentation

PSOT

APPLICATION

PSOT is a system that monitors the occupancy of certain public spaces through the collection of unique MAC addresses using Raspberry Pis spread out in those spaces.

The information would then be available through a web dashboard and sent through a message queue. This way people can quickly choose places to study, have lunch, etc.

ABOUT THE PROJECT

Nowadays, it is common for a person to carry a device with connectivity capabilities. These devices usually have associated with them a MAC address that is supposed to be unique.

If we run on the assumption mentioned above, we can measure the amount of people in a space by counting the amount of devices in the vicinity using their MAC addresses.

PLANNED PROOF OF CONCEPT

- Single Raspberry Pi scanning and collecting MAC Addresses around itself, through bluetooth and Wi-Fi
- A database collecting the scan data
- A periodic algorithm that gathers the scan data from the database and calculates the occupancy of each space
- A very simple dashboard that displays the current occupancy of each space, updating it in real time

REALISED PROOF OF CONCEPT

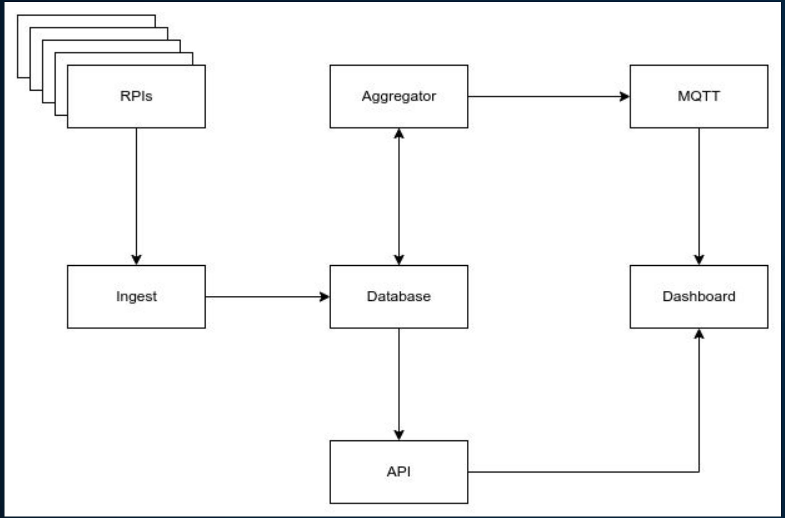
The realised proof of concept contains all elements predicted in the plan, however:

- The Raspberry Pi only collects MAC addresses through Wi-Fi
- As it is more uncommon for devices to communicate through bluetooth making it uncommon to catch their MAC address
- So, for a proof of concept it wasn't worth the effort

ARCHITECTURE

The system architecture is composed of the following components:

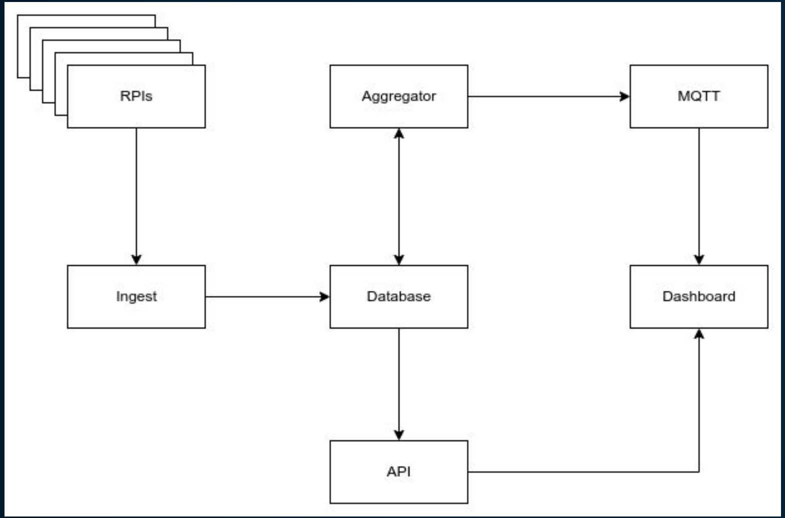
- A Raspberry Pi fleet 01
- An ingest service 02
- A Database 03
- 04 An aggregator service
- 05 An API
- 06 A dashboard
- MQTT 07 An MQTT broker



ARCHITECTURE

The Raspberry Pis/Scanners scan surrounding MAC addresses and send them to the Ingest service through HTTP over IP. The data comes structure in JSON objects.

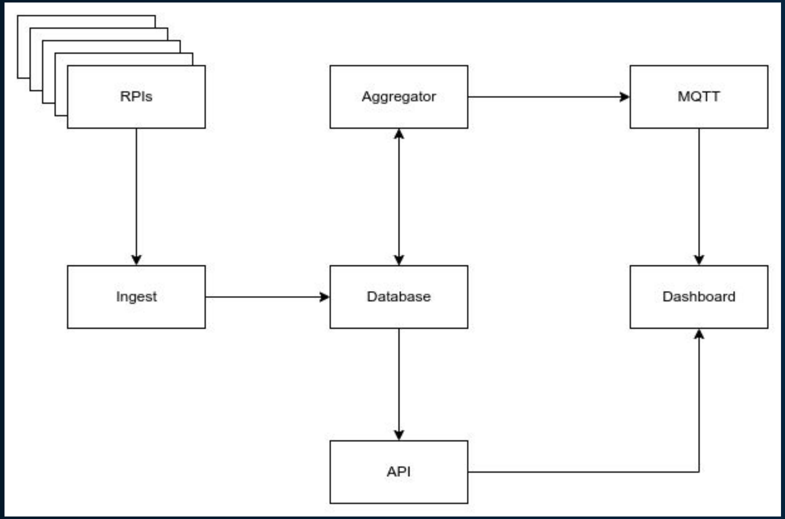
The Ingest service's function is to receive the scan data from the Scanners and insert it into the database. It exists in the middle to ensure good decoupling of the Scanners and database. This way we can swap the database implementation without having to change the software on the scanners as well.



ARCHITECTURE

The Aggregator's function is to periodically collect the most recent scan data and using it to estimate the occupancy level of the spaces.

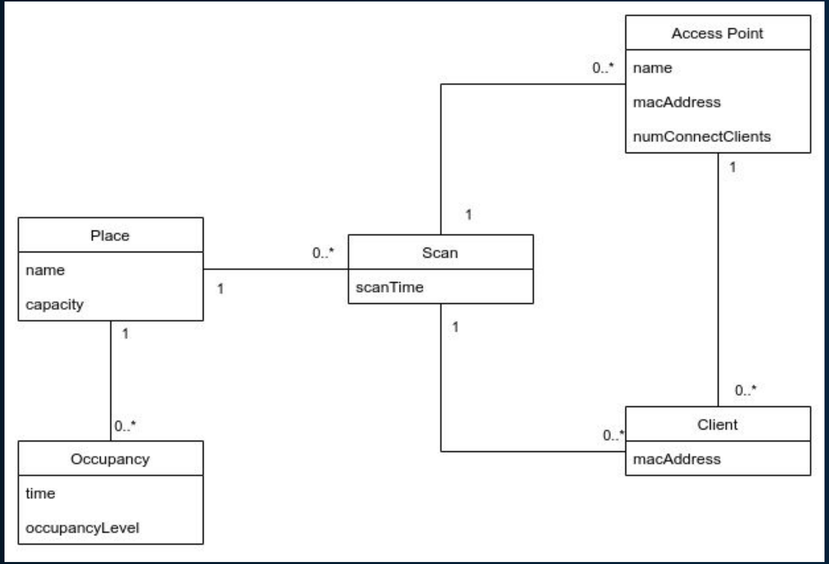
Once those are calculated it publishes the values to the database as well as send them through a message queue to notify any interested parties. It also clears all old scan data for privacy reasons.



ARCHITECTURE

Finally, the Dashboard displays the occupancy of each space through a simple list containing badges informing if the space is free, occupied or full. For each place it also provides a chart with the readings from the last 6 hours.

The data is provided by an API, as well as a message queue to ensure there is no need to reload the page.

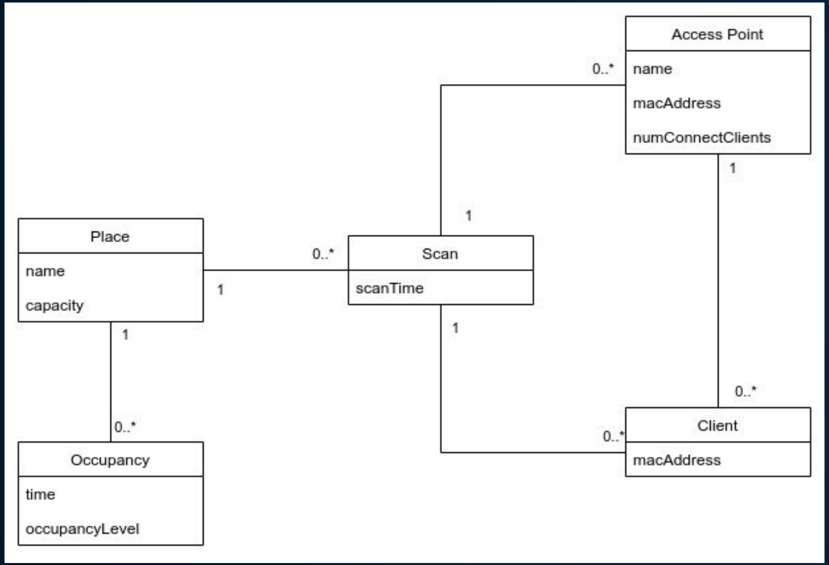


DATA MODEL

The system's basic data model is as shown in the picture. Each place has a name and capacity.

Scans are associated with a single place, and have a timestamp in them. Each scan detects Access Points and respective connected Clients. Clients are associated with scans for easier queries.

Both Clients and Access Points have their own MAC addresses.



DATA MODEL

Places are also associated with Occupancy records, which are calculated periodically, and stored with timestamps. Occupancy levels are in percentages.

AGGREGATOR ALGORITHM

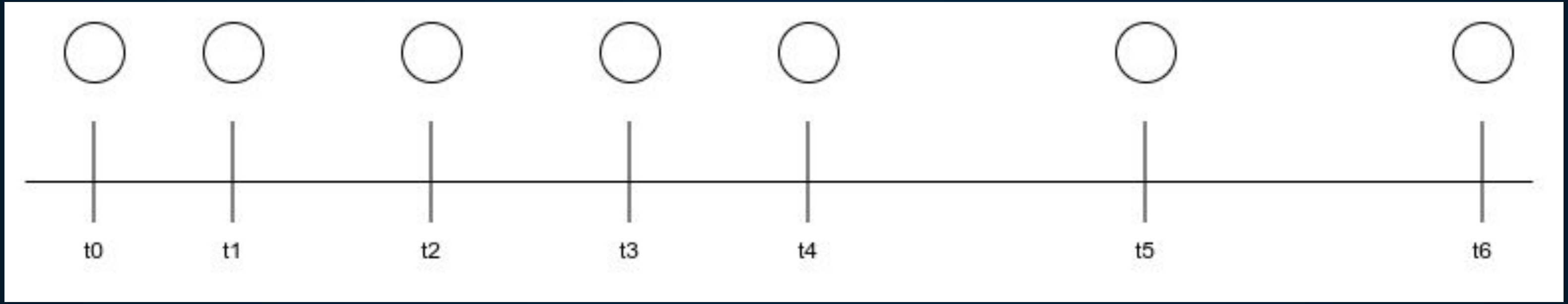
The aggregator algorithm has the responsibility of estimating the number of people in a space as correctly as possible. We assumed a person is in a space if they stay in it for at least a specific amount of time, so the algorithm has to know how to detect when the scanners detect a person that was only passing by the space (like a person walking on a corridor adjacent to a classroom). We also assume that there can be more than one scanner collecting MAC addresses in each space.

With these assumptions the algorithm runs as follows for each space:

AGGREGATOR ALGORITHM

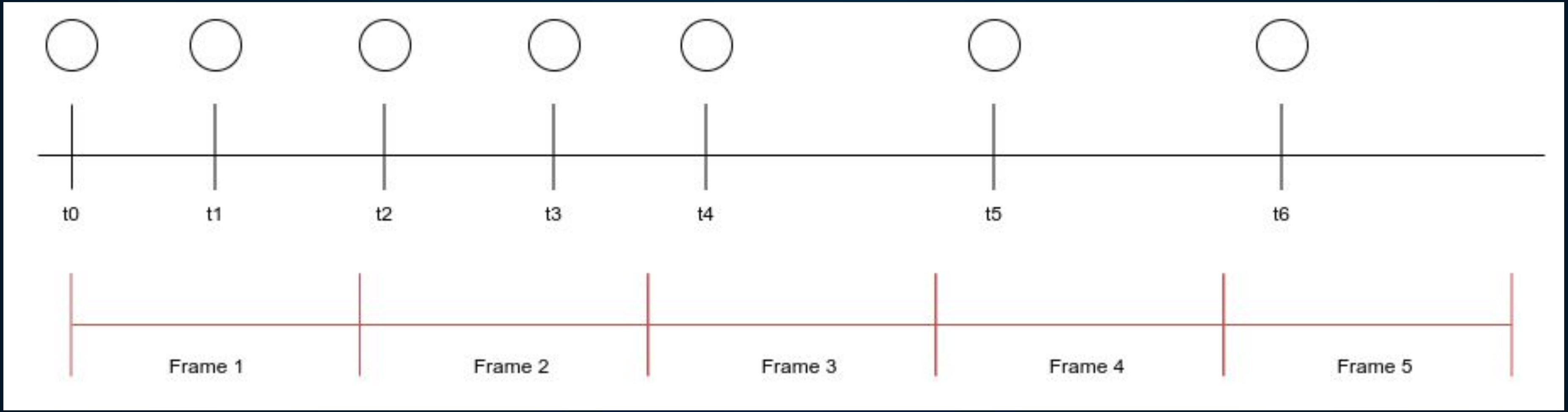
1 - Collect the most recent scans.

2 - Arrange the scans along a timeline.



AGGREGATOR ALGORITHM

3 - Split the timeline into equally sized time frames.



AGGREGATOR ALGORITHM

4 - Enumerate the MAC addresses that are detected in each frame.

- Frame 1: mac1, mac3, mac6, mac2…
- Frame 2: mac2, mac6, mac8, mac9…
- …

5 - Count the number of time frames in which each MAC address was detected:

- mac1: 4 frames
- mac2: 2 frames
- mac3: 1 frame
- …

AGGREGATOR ALGORITHM

6 - Now, choose all the MAC addresses whose count has gone above a certain threshold.

With this we now have a list of MAC addresses we are sure have stayed in a space for a certain amount of time.

We now need to deduce the amount of people that are in that space considering the number of detected MAC addresses. For this, we multiply the size of the list to a constant which is a real number between 0 and 1. This is because sometimes people have more then one connected device (like a phone and laptop, or people with two phones). This constant can be deduced from historical data.

An optimization we have done to the algorithm is a whitelist of devices connected to certain networks, that is because, sometimes we want to monitor spaces that polluted with neighbouring networks and we know that the people inside that space will only be connected to a specific set of network (like a house, scanners might pick up network from the neighbours).

AGGREGATOR ALGORITHM CALIBRATION

The aggregator algorithm can be calibrated, we can change how recent we want the scans to be, the size of the time frames or the count threshold for the MAC addresses.

This allows the algorithm to be calibrated to certain spaces as different spaces are not used the same way, for example:

- A coffee shop might just have people stay in it for a couple of minutes, so we might want to relax the algorithms to accept devices who’ve been detected for a few seconds.
- While in libraries, people will generally stay for longer, so we might only want to count a device if it has stayed in it for more than a few minutes.

DECISIONS

Raspberry PI:

- The RPI was chosen as it was relatively simple to work with.
- Since it is essentially a small computer, it is easy to write software for it.
- Also, since it can run docker, the setup needed to get the scanner code running was minimal.

DECISIONS

HTTP over IP and JSON between RPIs and Ingest:

- Ingest is a web server running on the cloud, so HTTP made a lot of sense from the beginning.
- Since the Raspberry Pi is basically a small computer, it also had all the capabilities needed to access the Internet, making this choice the simpler course of action.
- JSON is also a simple format that is used for communication over the web, and many languages already implement a parser and encoder for it.

DECISIONS

Central processing of the scan data:

- The data is processed centrally by a single service.
- This is done because each space can have more than one Raspberry PI scanning for MAC addresses, which would greatly increase the complexity of the data processing as we would need to have the raspberry pis sharing information with each other.
- It also make changes to the estimation algorithm easier since changing software running on the cloud is simpler than changing software running on machines distributed geographically.

DECISIONS

Use of MQTT for occupancy notifications:

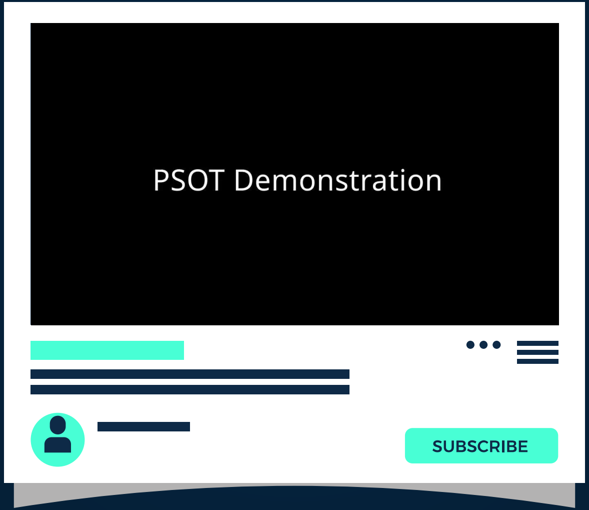
- In a more complete and production ready version of the system, there could devices physically deployed around spaces that display the occupancy level of the spaces, as well as users who choose to be notified when a certain space becomes available.

- Using an MQTT broker to distribute these notifications makes it super simple for future consumers to plug into the data.

CHALLENGES

- We can’t force devices to give us their MAC address, this means we have to scan for long periods of time. This won’t guarantee that the scanners find every MAC address, however, so the aggregator has to consider multiple scans over a period of time.
- Another point is since the scanners are only scanning for Wi-Fi packets, we can only detect devices if they are connected the a Wi-Fi access point, making this system only suitable for spaces with Wi-Fi networks that many devices are connected to.
- Detecting only MAC addresses isn’t enough, since one person can have more than one device, like a laptop and phone, and there are devices that don’t belong to anyone, like printers.
- To counteract this, there must be many calibration parameters, like scaling constants to adjust the number of detected mac addresses or MAC address blacklists to filter out devices we know are not associated with people.

VIDEO DEMO



Just in case:
https://drive.google.com/file/d/1SfzJ2hdXoxZzysMr1iBNw1JLm-vEdrLt/view?usp=sharing

WHAT WE LEARNED

- Working with the Raspberry PI, mostly on setting up it’s configurations so it needs the least possible setting up to get the scanner running. We learned about editing the network files on the OS and enable startup services like ssh to make it easier to connect to the device to work on it.

- We also learned about creating a data processing system with many services that collect data and apply algorithms over it.

- Finally, we learned of the challenges that come with this subject of measuring the occupancy of spaces through the scanning and detection of devices.

SELF-EVALUATION

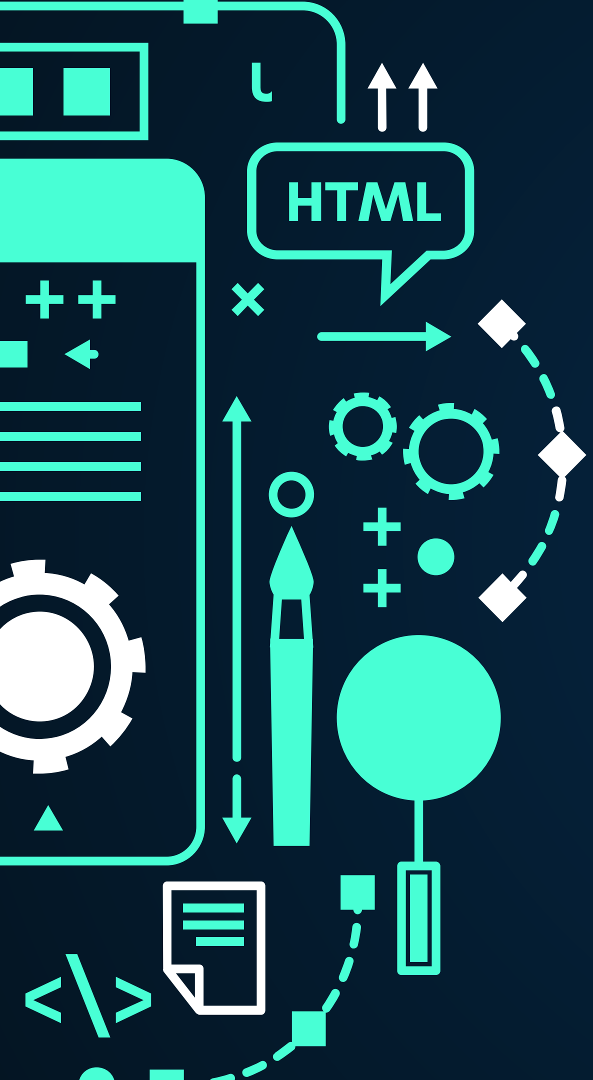
GRADE PROPOSAL: 17

DIOGO PEREIRA
20%

JOÃO PINTO
20%

MIGUEL PIRES
20%

MOISÉS ROCHA
40%



THANK YOU!

Does anyone have any question?

Group A:
Diogo Pereira - 201505318
João Pinto - 201705547
Miguel Pires - 201406989
Moisés Rocha - 201707329