Outline:

- **Data Collection**
  1. COAP
  2. MQTT

- **Data Processing**
  1. Estimating missing RSSI values
  2. Creating the final Fingerprint Dataset

- **Positioning**
  1. Cosine similarity
  2. Result

- **Delivered Files**

Data Collection:

In this part, we collected data based on two approaches. The first set of them were obtained from the COAP server and the second one from the MQTT server.

1. COAP:

For collecting the data by COAP we used different commands, such as observe, put, delete and post. The following table contains the hidden data that could not be extracted regularly from the COAP server.

It should be noted that we got other data in a regular way (based on put and delete too). For instance, we got (2,4) based on the put command or we got (4,8) based on the delete command.

approach	command	Position
coap://131.175.120.117/root/BarrierReef/Apps?fingerprint=True&gps=False	get	(8,2)
coap://131.175.120.117/root/BarrierReef/Dory?answer=yes	**get**[1]	**Dorry's RSSI**
coap://131.175.120.117/root/PostMe6?search=entry	post	(2,2)
coap://131.175.120.117/root/BarrierReef/FishLocator?user=Dory	get	(10,10)
coap://131.175.120.117/root/BarrierReef/Anemone?owner=Marlin	post	(6,10)
coap://131.175.120.117/root/questions?answer=yes	get	(10,0)
coap://131.175.120.117/root/BarrierReef/HiddenTreasure	get	(8,10)
coap://131.175.120.117/root/BarrierReef/Doctor?problem=memory	Post	(0,2)

Table.1. Main variables

[1] By this command we could get the Dorry RSSI but we should try several times since it has written that Dorry doesn't remember her RSSI vector.

2. MQTT:

The second approach was gathering data from the MQTT server. We used the following subscription command:

mosquitto_sub -v -h 131.175.120.117 -p 1883 -t '#'

By this command, other RSSI vectors corresponded with the positions are revealed (such as (0,10), (8,4)). After collecting all the accessible data we could proceed with constructing the dataset.

Note: In the procedure of collecting data we got some useless points. such as (8,14) , (6.6,6.6), (0,1), and (3,x.y).

At this point, we have all RSSI vectors of the green dots showed in the following figure. We have insert them into a CSV file, to use it for estimating the RSSI vectors of the orange positions in the next part.

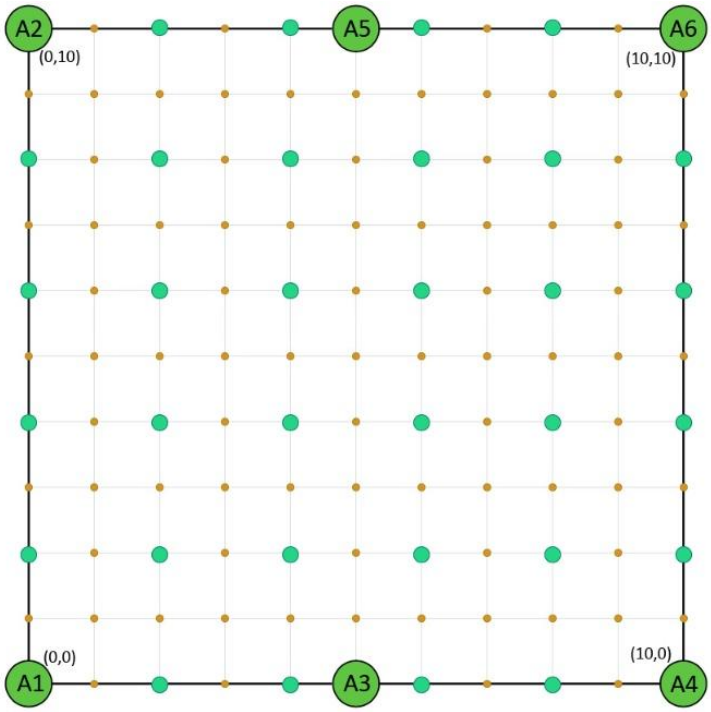

Figure.1. Feasible region represents the dataset

Data Processing:

In this part, the following steps have been taken:

1- reading the created dataset from a CSV file
2- estimating the missing values (RSSI position vectors)
3- Creating the final Fingerprint dataset for localizing Dory.

RSSI vectors of points belonging to odd rows and columns are missed. Generally, We have used the mean value of the previous and next columns (and rows) as an estimation of the missed columns(and rows). The following table is the specifications of the main variables used in the code.

Name	**Type**	**Dimension**	**Description**
fp_ds	Pandas DataFrame	(121,32)	Fingerprint dataset obtained from previous part
final_estimated_fp_ds	Pandas DataFrame	(121,33)	Fingerprint dataset with estimated missing values
final_estimated_fp_ds_new	Pandas DataFrame	(121,9)	The final Fingerprint dataset obtained by substituting mean(max/min) RSSI value of each position with its 6 element RSSI vector
X	Numpy Array	(121,6)	Converted Numpy type of final_estimated_fp_ds_new variable
dory_rssi	List	(1,6)	-
similarity	List	(121,1)	Similarity matrix between dory's RSSI vector and each row of the Fingerprint dataset
pos_x	Int	-	Predicted position of dory along with the x-axis
pos_y	Int	-	Predicted position of dory along with the y-axis
result	Tupple	-	Dory's predicted position

Table.2. Main variables

1. Estimating missing RSSI values:

Definition: Super vector is a row (or column) vector that contains the RSSI vectors of each position that is placed in that row (or column).

For estimating missing values in odd rows, we have performed an average based on the above and below super vectors. Afterward, we extract the column super vectors and have done exactly the same approach by computing the average based on the left and right sides of the odd columns.

Then by concatenating all super column vectors, we create the complete fingerprint dataset. At this step, we converted our dataset to a DataFrame. After converting, 3 following columns have been joined to the dataFrame: X-axis, Y-axis and, Index (which maps 2d points to a single index).

2. Creating the final Fingerprint Dataset:

At this point, we reduced the dimension of the Fingerprint dataset by choosing one value among each RSSI vector of each position with respect to the anchors; based on 3 different criteria (min or max, or mean). In fact, size of vectors reduced from 30 to 6. To do so, we defined 3 flags in order for the code to be run in 3 different statements, upon turning the flag to true. However,the final result was not affected so we wrote the report based on using mean approach.

Positioning:

Finally, we can predict Dory's position. We converted the Fingerprint dataset to Numpy and also dropped the "x", "y", and "index" columns. Now we have a dataset of RSSI vectors (corresponding to each position) and Dory's RSSI vector.
We have computed the similarity of Dory's RSSI vector with each row of the dataset, using cosine similarity. We stored all the similarities in the similarity matrix and by performing argMax

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

$$\|\vec{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2}$$

$$\|\vec{b}\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots + b_n^2}$$

across values, we have found the index mapped to (x,y) positions. At last, Dory's position was revealed as **(2,3)**.

```
[403]  1 #Computing cosine similarity between Dory's RSSI vector and the fingerprint dataset
       2 from sklearn.metrics.pairwise import cosine_similarity
       3 dory_rssi = [-57,-63,-58,-64,-63,-66]
       4 similarity = []
       5 for i in range(0,len(X)-1):
       6   similarity.append(cosine_similarity([X[i]], [dory_rssi]))
       7 similarity
       8
       9 max_index = np.argmax(similarity)
      10 pos_x = final_estimated_fp_ds_new.loc[final_estimated_fp_ds_new['index'] == max_index ,'x'].values[0]
      11 pos_y = final_estimated_fp_ds_new.loc[final_estimated_fp_ds_new['index'] == max_index ,'y'].values[0]
      12 result =(pos_x , pos_y)
      13 print("Dory's position: ",result)

     Dory's position:  (2, 3)

[411]  1 np.max(similarity)

     0.9999919807876007

       1 np.argmax(similarity)

     35
```

Figure.2. The last part of the code showing the result.

Delivered Files:

A zip file comprises of:

1. **fp_ds:** A CSV file contains all data obtained from COAP and MQTT servers
2. **final_estimated_fp_ds:** A CSV file contains both estimated and obtained RSSI values
3. **final_estimated_fp_ds_new:** A CSV file contains both estimated and obtained RSSI values with "point RSSI" for each position that has been calculated by averaging over each RSSI vector
4. **WI_IoT_ProjectV4:** Source code of the project in Python (in both .ipynb and .py format)