**Project Documentation**

1. **Project Structure**

The entire structure of the project consists of 5 main directories:

* **data:** Directory where the data obtained using the ***get\_sensordata*** application is stored.
* **info:** Directory with the following information related to the project:

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

* + - File with project documentation.
    - File with coordinates used at each reference point in training.
    - File with coordinates used at each reference point in testing.
    - Map of the data collection area, indicating the position of each reference point in both training and testing.
* **output:** As the name suggests, this is the output directory:
  + - Images
    - Preprocessed data
    - Tables with results and metrics
    - GIFs
    - …
* **src**: Directory that includes Python scripts with fundamental methods and classes throughout the project development process. It contains functions and classes for preprocessing, visualization tools, data loading, and project constants. It also includes essential scripts to be executed in project processes, which are run from ***obtainDataINITandPositioning.py***.

1. **Requirements**

The project has been developed using the following versions of the libraries:

ipython==8.8.0

keras==2.9.0

matplotlib==3.6.3

numpy==1.23.5

pandas==1.5.3

Pillow==9.4.0

scikit\_image==0.19.3

scikit\_learn==1.2.1

scipy==1.11.4

tensorflow\_gpu==2.9.0

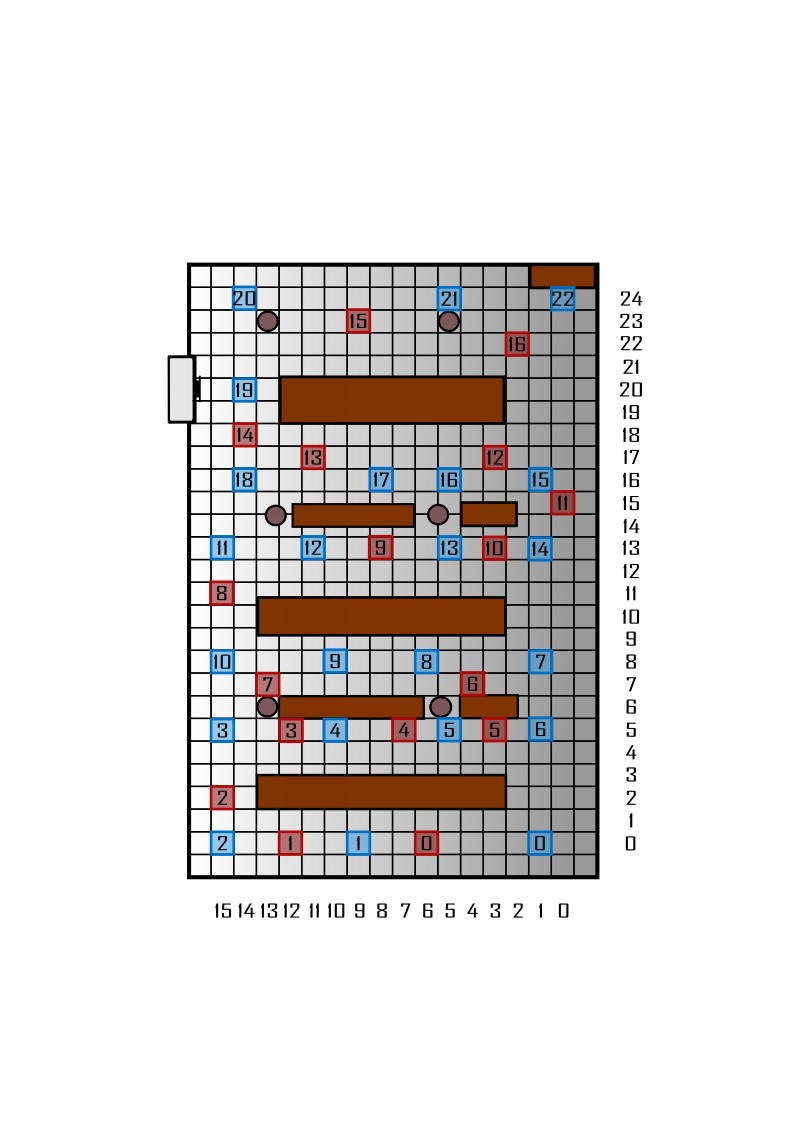
tensorflow\_intel==2.11.0

tqdm==4.64.1

To adjust the versions of the necessary libraries for the project, we have the requirements.txt file, so that to install the dependencies, we can use the following code:

pip install -r requirements.txt

1. **Project data, preprocessing and structuring**
   1. **Datos de Train y Test**



The data has been collected manually using the get\_sensordata application in the INIT building. The image above shows a map of the floor and the locations of various data collection reference points. The blue color represents the Train points, while the red color represents the Test points.

A file is generated for each reference point, and once the data is collected using an Android device (such as a mobile phone, tablet, etc.), it is manually transferred to the next directory (in both Train and Test):

Interfaz de usuario gráfica, Texto

Descripción generada automáticamentedata  
|----train  
| |---- initial\_rp\_data (TRAIN DATA)  
|

|----test

| |---- initial\_rp\_data (TEST DATA)

If we consider that each tile, represented by the gray grids in the image, measures 60 centimeters, then we perform a transformation from fictitious coordinates (tiles) to longitude and latitude coordinates (meters). The table can be found on the next page.

**Table of reference points with annotated coords**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Label | Longitude | Latitude |
| TRAIN | 0 | 0.6 | 0 |
| 1 | 5.4 | 0 |
| 2 | 9 | 0 |
| 3 | 9 | 3 |
| 4 | 6 | 3 |
| 5 | 3 | 3 |
| 6 | 0.6 | 3 |
| 7 | 0.6 | 4.8 |
| 8 | 3.6 | 4.8 |
| 9 | 6 | 4.8 |
| 10 | 9 | 4.8 |
| 11 | 9 | 7.8 |
| 12 | 0.6 | 7.8 |
| 13 | 3 | 7.8 |
| 14 | 0.6 | 7.8 |
| 15 | 0.6 | 9.6 |
| 16 | 3 | 9.6 |
| 17 | 4.8 | 9.6 |
| 18 | 8.4 | 9.6 |
| 19 | 8.4 | 12 |
| 20 | 8.4 | 14.4 |
| 21 | 3 | 14.4 |
| 22 | 0 | 14.4 |
| TEST | 0 | 3.6 | 0 |
| 1 | 7.2 | 0 |
| 2 | 9 | 1.2 |
| 3 | 7.2 | 3 |
| 4 | 4.2 | 3 |
| 5 | 1.8 | 3 |
| 6 | 2.4 | 4.2 |
| 7 | 7.8 | 4.2 |
| 8 | 9 | 6.6 |
| 9 | 4.8 | 7.8 |
| 10 | 1.8 | 7.8 |
| 11 | 0 | 9 |
| 12 | 1.8 | 10.2 |
| 13 | 6.6 | 10.2 |
| 14 | 8.4 | 10.8 |
| 15 | 5.4 | 13.8 |
| 16 | 1.2 | 13.2 |

In the main directory, we have two Python scripts: **`process\_train.py`** and ***`process\_test.py`***. The execution of each of these scripts will perform all the necessary preprocessing to obtain the radiomap ready for training the positioning models.

The execution of these scripts will provide us with the following output directory:

(underscored new files and directories)

data  
|----train  
| |---- initial\_rp\_data  
  
|   
|---- test  
| |---- initial\_rp\_data  
|  
|---- outputs \*\*  
| |---- data \*\*  
| | |---- train \*\*  
| | | |---- raw\_radiommap \*\*  
| | | | |---- raw\_radiomap.csv \*\*  
| | | |---- processed\_radiomap \*\*  
| | | | |---- processed\_radiomap.csv \*\*  
| | | |---- checkpoint\_groundtruth \*\*  
| | |  
| | |---- test \*\*  
| | | |---- raw\_radiommap \*\*  
| | | | |---- raw\_radiomap.csv \*\*  
| | | |---- processed\_radiomap \*\*  
| | | | |---- processed\_radiomap.csv \*\*  
| | | |---- checkpoint\_groundtruth \*\*

Where **raw\_radiomap.csv** corresponds to the unscaled radiomap data, so it is presented in units (decibels), while processed\_radiomap.csv presents the RSS levels scaled between 0 and 1.

The preprocessing scripts depend directly on the methods implemented in the ***preprocess.py*** and ***constants.py*** scripts found in the ***src*** directory.

root

|

|----data

|---- src

| |---- utils

| | |----preprocess.py

| | |----constants.py

Within the constants, in the **src/utils/constants.py** script, it is important to define the following correctly and appropriately for the directory system in the project, so that preprocessing is carried out correctly:

**constants.data.train.INITIAL\_DATA:** Path to the initial training data

**constants.data.train.CHECKPOINT\_DATA\_PATH:** Path to the checkpoints for training data

**constants.data.train.RAW\_OUT\_PATH:** Path to the raw training radiomap

**constants.data.train.PROC\_OUT\_PATH:** Path to the scaled training radiomap

**constants.data.test.INITIAL\_DATA:** Path to the initial test data

**constants.data.test.CHECKPOINT\_DATA\_PATH:** Path to the checkpoints for test data

**constants.data.test.RAW\_OUT\_PATH:** Path to the raw test radiomap

**constants.data.test.PROC\_OUT\_PATH:** Path to the scaled test radiomap

**constants.aps:** List of WiFi access points to consider for fingerprinting

**constants.labels\_dictionary\_meters:** Dictionary that transforms labels into coordinates for training

**constants.labels\_dictionary\_meters\_test:** Dictionary that transforms labels into coordinates for testing

**constants.labels\_train:** List of labels (access points) to consider in training

**constants.labels\_test:** List of labels (access points) to consider in testing

**constants.T\_MAX\_SAMPLING:** Maximum sampling time (in seconds) for each training label

**constants.T\_MAX\_SAMPLING\_TEST:** Maximum sampling time (in seconds) for each test label

Finally, using the methods described in ***preprocess.py***, when running the ***process\_train.py*** and ***process\_test.py*** scripts, the execution flow will be as follows:

Diagrama

Descripción generada automáticamenteThe process can be summarized as follows:

The log files from ***get\_sensordata*** are processed to transform them into a format where each row represents the fingerprint of one second, calculated as the mean of all observations in that period of RSS for each access point (AP).

Subsequently, any resulting NAs are replaced by the global mínimum value minus 1. This is done so that, during scaling, 0 represents the absence of RSS.

A moving average is also applied in Windows of 30 seconds with an overlapping of 5 seconds. This way, se smooth out the RSS values obtained

Finally, the process results in obtaining the **raw\_radiomap.csv** file (unscaled) and the **processed\_radiomap.csv** file (scaled)

* 1. **Data partitions on Train**

The project has also considered, as an alternative testing approach, to create multiple partitions for both training and testing using the same data collected for training. This allows studying the effects on data collection for similar time periods. To achieve this, the script ***process\_partitions.py*** has been implemented. The purpose of this script is to obtain the following output directory by applying the same preprocessing steps as in the case of training, but with different reference points.

The script returns the following file directory:

root

|

|----data

| |----train

| |---- test

|---- output \*\*

| |---- data \*\*

| | |---- train

| | |---- test

| | |---- partitions \*\*

| | | |---- partition\_5vs18 \*\*

| | | | |---- train \*\*

| | | | | |---- raw \*\*

| | | | | | |----raw\_radiomap.csv \*\*

| | | | | |----processed \*\*

| | | | | | |----rocessed\_radiomap.csv \*\*

| | | | | |

| | | | |----test \*\*

| | | | | |----raw \*\*

| | | | | | |----raw\_radiomap.csv \*\*

| | | | | |----processed \*\*

| | | | | | |----processed\_radiomap.csv \*\*

| | | |

| | | |---- partition\_10vs13 \*\*

| | | | |---- train \*\*

| | | | | |---- raw \*\*

| | | | | | |---- raw\_radiomap.csv \*\*

| | | | | |---- processed \*\*

| | | | | | |---- processed\_radiomap.csv \*\*

| | | | | |

| | | | |----test \*\*

| | | | | |---- raw \*\*

| | | | | | |---- raw\_radiomap.csv \*\*

| | | | | |---- processed \*\*

| | | | | | |---- processed\_radiomap.csv \*\*

| | | |

| | | |---- partition\_15vs8 \*\*

| | | | |---- train \*\*

| | | | | |---- raw \*\*

| | | | | | |---- raw\_radiomap.csv \*\*

| | | | | |---- processed \*\*

| | | | | | |---- processed\_radiomap.csv \*\*

| | | | | |

| | | | |----test \*\*

| | | | | |---- raw \*\*

| | | | | | |---- raw\_radiomap.csv \*\*

| | | | | |---- processed \*\*

| | | | | | |---- processed\_radiomap.csv \*\*

In this step, three alternatives have been considered with the training data, represented in the following table:

|  |  |  |
| --- | --- | --- |
| Partition | Set | Associated reference points |
| Partición 5 Train over18 Test | Train | [0, 2, 11, 14, 21] |
| Test | [1, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 17, 18, 19, 20, 22] |
| Partition 10 Train over 13 Test | Train | [0, 2, 3, 5, 9, 13, 19, 22] |
| Test | [1, 4, 6, 7, 8, 10, 11, 12, 14, 15, 16, 17, 18, 20, 21] |
| Partition 15 Train over a 8 Test | Train | [0, 2, 4, 5, 7, 9, 10, 11, 14, 15, 17, 18, 20, 21, 22] |
| Test | [1, 3, 6, 8, 12, 13, 16, 19] |

For the proper functioning of the ***process\_partitions.py*** script, it is essential to ensure that the following constants in **src/utils/constants.py** are correctly defined:

**constants.labels\_partition\_5vs18:** List of reference points for training to be used in the 5 vs 18 partition.

**constants.labels\_partition\_10vs13:** List of reference points for training to be used in the 10 vs 13 partition.

**constants.labels\_partition\_15vs8:** List of reference points for training to be used in the 15 vs 8 partition.

**constants.data.partitions.PARTITION\_5VS18:** Output directory for the 5 vs 18 partition.

**constants.data.partitions.PARTITION\_10VS13:** Output directory for the 10 vs 13 partition.

**constants.data.partitions.PARTITION\_15VS8:** Output directory for the 15 vs 8 partition.