Universidade de Aveiro

DETI

Projeto em Engenharia Informática

# Mobility platform in Aveiro Tech City Living Lab Infrastructure

Technical Report

Grupo 3

Diogo Mendes, 88801

Hugo Leal, 93059

Luísa Amaral, 93001

Maria Strecht, 93089

Pedro Loureiro, 92953

6 de julho de 2021

Universidade de Aveiro

DETI

Projeto em Engenharia Informática

# Mobility platform in Aveiro Tech City Living Lab Infrastructure

Relatório de Projeto em Informática do Mestrado Integrado em Engenharia de Computadores e Telemática da Universidade de Aveiro, realizado por Diogo Mendes, Hugo Leal, Luísa Amaral, Maria Strecht e Pedro Loureiro sob a orientação da Dr.ª Susana Sargento, Professora Catedrática do Departamento de Engenharia, Eletrónica e Telecomunicações da Universidade de Aveiro e do Dr. Pedro Rito, Investigador do Instituto de Telecomunicações.

6 de julho de 2021

## 1 Abstract

Following the increasing growth of devices connected to Wi-Fi and with the evolution of IoT, the concept of a modern inter-connected city is gaining popularity, therefore the Aveiro City Hall proposed the University of Aveiro the task to monitor certain areas of the city, in order to extract conclusions upon the patterns of peoples flows in those areas. In this report, two solutions (sniffing and video detection) to this problem are presented, as well as results and conclusions from several weeks of testing. The detection module is active in two smart lamp posts in both Cais da Fonte Nova and near the Institute of Telecommunications of Aveiro 2 department and is set to detect people, vehicles, two-wheelers and *moliceiros*. The sniffing module uses an Accelerated Processing Unit's wireless network adapter, which is inside each post, set in monitor mode so that it can capture probe requests emitted by the devices that pass near the posts. This module is also running inside two *moliceiros* but instead of using APU's, we use On Board Units that connect to the nearby post to send data throughout the trips. The results are shown in our web application which displays the real-time detection of objects and sniffing as well as past values in charts. These charts allow us to understand which are the hours when more people are passing by monitored areas. Moreover, we are able to track the average number of people inside each *moliceiro* with a OBU, as well as the number of trips.

## 2 Acknowledgements

We would like to acknowledge and to thank all the Aveiro Tech City Living Lab team for their help and patience and for teaching us the basics related to the project. And, lastly, we would like to thank the supervisors Drª. Susana Sargento and Dr. Pedro Rito for not only helping us and guiding us throughout the project but as well for giving us this opportunity.

# Contents

# List of Figures

# List of Tables

# Listings

# 3 Introduction

One of the characteristics of the city of Aveiro is how it is gradually growing to be a smart city where several devices, vehicles and stations are connected between each other and trade information in order to share information with the public and provide a better quality of government service and citizen welfare. The Aveiro Tech City Living Lab (ATCLL) is an innovative platform that provides an open and large-scale technological laboratory in the city and is supported by state-of-the-art fiber technology, reconfigurable radio units and 5G network services. It is a service developed jointly by the Municipality of Aveiro, the Institute of Telecommunications, Altice Labs and the University of Aveiro.

The ATCLL infrastructure is a part of the Institute of Telecommunications of Aveiro 2 (IT2) and it's composed by 44 stations installed throughout the city and interconnected with fiber between them and to the institute's data center. Each station contains different types of sensors such as environmental sensors, radars, LIDARs, video cameras and computer edge units.

This infrastructure allows the development of an authentic tech city living lab for the experimentation of future services and applications in a real environment.

Up until the start of this project, the ATCLL group was already detecting vehicles through RADARs and people through the analysis of video streams. These projects were fundamental to better understand how to send data in real-time from the city's stations to the appropriate gateways for receiving and processing the information, persisting it and analyzing the obtained data. The group had also developed a solution to estimate the number of people inside a bus using the capture of Wi-Fi packets that served as a point of launch for this project.

The next step was to enrich the ATCLL infrastructure in order to continue improving the lifestyle of citizens and expanding research projects. To do this, the Aveiro's Town Hall proposed the monitoring of people in public spaces and the quantity, frequency and passengers of *moliceiros* in the *Ria of Aveiro*.

This project aims to use the Aveiro Tech City Living Lab (ATCLL) to complete that request. The main goal is to estimate the number of people, vehicles and *moliceiros* in certain areas of *Aveiro* city, and use this information to gather statistics and identify patterns of movements in certain time periods.

Two areas were chosen to develop this project. The first one (P1) was in the University of Aveiro campus, near the IT2 building (Figure 1) and the second (P22) was in the Cais de Fonte Nova (Figure 2), where all the *moliceiros* pass during their journeys.

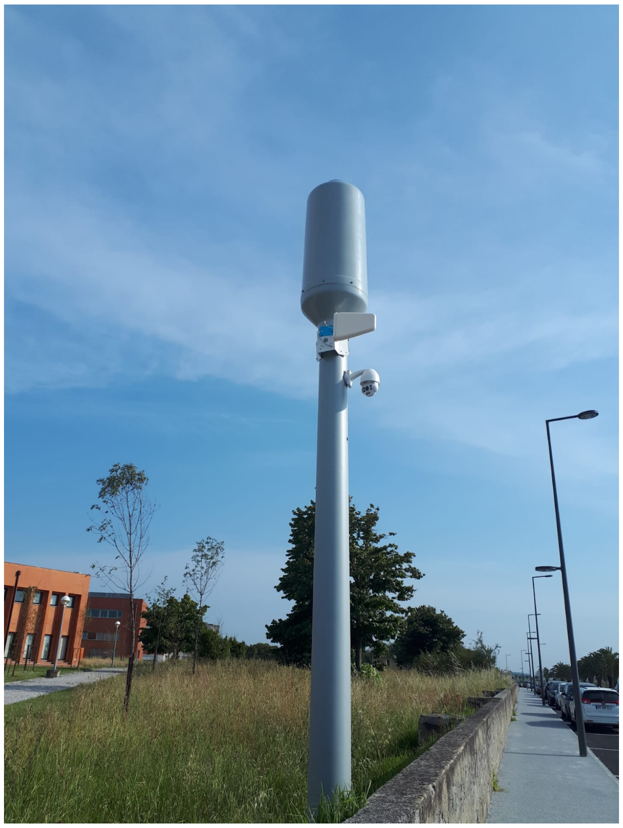


Figure 1: Smart Lamp Post in Instituto de Telecomunicações

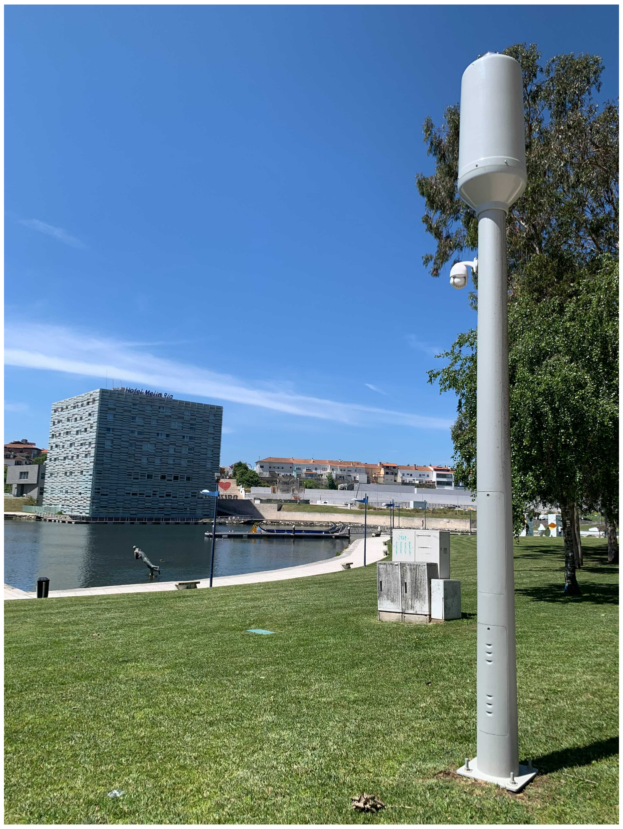


Figure 2: Smart Lamp Post in Cais da Fonte Nova

There is a smart lamp post connected to the ATCLL infrastructure installed in both of these areas and each post contains a video camera that rotates 360º degrees, an Accelerated Processing Unit (APU) and a Jetson Nano, which is a small, powerful computer that allows the execution of multiple neural networks in parallel for applications like image classification and object detection.

It was necessary to understand how to use these different devices to perform two different types of detection. To detect people, vehicles, two-wheeler vehicles and *moliceiros* the method chosen was the analysis of video streams using object detection algorithms running on the Jetson Nano of each lamp post (Figure 3).

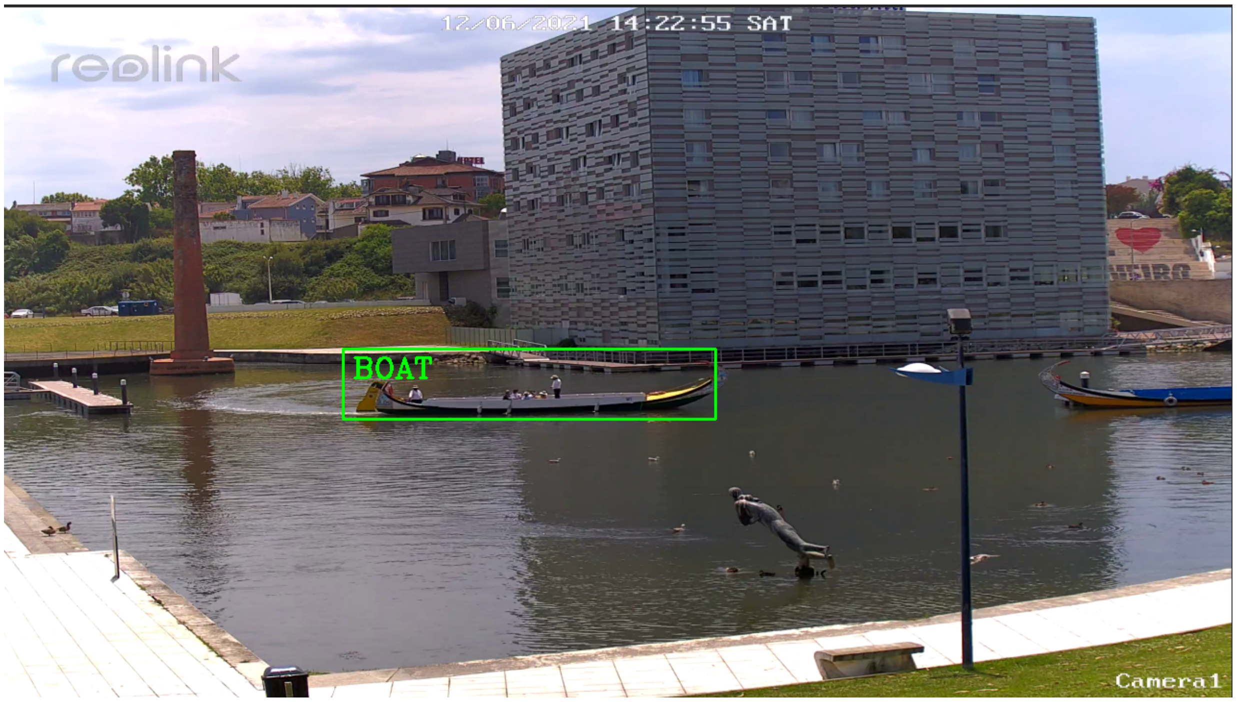


Figure 3: Moliceiro being detected

A second type of detection was also adopted so that the values obtained from the previous method could be compared and verified with other sources of data. For this detection alternative, the method chosen was the capture of Wi-Fi packets sent from smartphones to estimate the number of devices in an area. This capture was performed using the APU inside the smart lamp post, and the estimation values obtained were then compared to the number of people obtained by the first method, since this module was calibrated to detect the number of devices closest to the number of people, as will later be discussed.

To estimate the number of people travelling inside *moliceiros*, On-Board Units (OBUs) were placed inside them running the devices' detection module and sending the processed data over WAVE to the several stations that came in contact with it during the *moliceiro's* journey (Figure 4). These OBUs were composed of two batteries, an APU with two antennas and a GPS module to obtain the *moliceiros* coordinates.

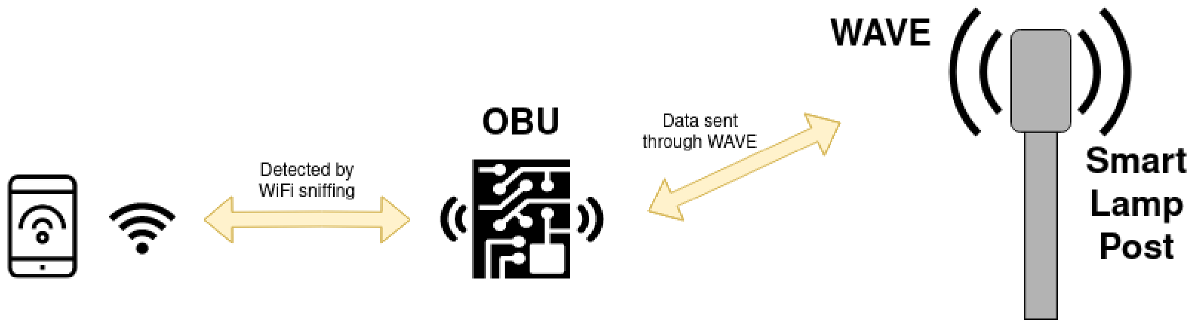


Figure 4: OBU communication diagram

All the collected information is then sent in real-time to local brokers, which in turn send it to a central subscription broker. These values are also persisted in the IT2 central database in order to accumulate all the past-time detected data for analysis.

By the end of this project, all of our goals were achieved. The data is acquired with success, being sent through the brokers and persisted in the central database. All the data, real-time and historical, is shown in the Web Application that was developed to display these values to the public.

In this report there is a brief introduction to the state of the art where it is described previous work regrading our problem. Following that section, a description of the process of requirements gathering, the actors and use cases of the system and a description of the requirements to solve the given problem. Afterwards, the domain model, the architecture diagram and the deployment model are shown, followed up by the implementation of all the modules, the results and the conclusion of the project.

# 4 State of the art

## 4.1 WiFi Sniffing

Mobile devices regularly broadcast Wi-Fi probe requests in order to discover available nearby Wi-Fi access points for connection. A probe request, sent automatically in the active scanning mode, consists of the MAC address of the device, and expresses an advertisement of its presence. A real-time wireless sniffing system is able to sense Wi-Fi packets and analyze wireless traffic, which provides an opportunity to obtain estimations of the number of humans carrying the mobile devices.

In Ribeiro et al. [2018] [1] a system and an algorithm capable of collecting information regarding Wi-Fi capable devices inside a bus was developed. The goal was to gather data regarding the use of transportation to provide accurate OD matrices and to improve public transport efficiency. An implementation of this system was deployed in a public bus to collect data for several weeks. This resulted on over 15000 traveler routes collected in 9 different days.

A case study of Wi-Fi Sniffing performance was approached in Li, Yan, et al. [2020] [2]. The goal was to identify the possible factors including channel settings and access point configurations that affect Wi-Fi sniffing behaviors and performances. The authors came to the conclusion that four main factors affect the sniffing performance, those being the number of access points and their corresponding operating channels, the signal strength of the access point and the number of devices in the vicinity. It is suggested to assign one sniffer as close as possible to the AP in each sub-area, and fix the monitor channel to be the one that the local strongest AP operates on.

Another work that proved interesting was presented at Basalamah [2016] [3]. The author developed a system for crowd behavior analysis using a solar-powered, beagle-bone based standalone system that continuously sniffs for probes and extracts their MAC addresses. The system was deployed in the world's largest gathering (The Hajj) and the objective was to build an infrastructure for non-invasive mass crowd analysis. The deployment had a total of 8 sniffers covering a population of 185,000 people, and 37.5% of this population was detected. This allowed the authors to analyze their arrival and departure behaviors, identify their smartphone manufacturers, and extract their transition patterns from one sub-location to another, validating the potential of Wi-Fi sniffing for crowd mobility analysis.

Finally, in similarity to Ribeiro et al. [2018] [1], another study (Tu, Lai, et al. [2021])[4] observes human mobility via Vehicular Internet Service and tries to understand human mobility using the passenger's Wi-Fi mobile devices.

## 4.2 Image Analysis

There are several studies and research papers that cover the subject of detection of objects using live stream video data. In order to better understand what already had been developed in the area, a few practical applications were analyzed.

The first one is *Heptasense*, a platform that provides intelligence about the behavior of people and vehicles to improve operations. This platform helps prevent incidents and assures safety as well as giving a chance to see and analyze the patterns of pedestrian habits.

Another real life application is *Parquery* and it uses AI technology to detect vehicles and any kind of object in images from any camera so that it can produce a smart parking solution by efficiently and effectively managing parking spaces.

A detection module involving objects classification was already being developed in Institute of Telecommunications of Aveiro 2 (IT2). However, the performance of video frame's processing could be improved in order to increase the processing rate. This could be achieved with the use of new tools.

### 4.3 Next steps

The analysis of these studies and platforms drew conclusions of concepts that were important to consider throughout this project. Capturing probe requests frames can be achieved with any IEEE 802.11 compliant wireless adapter set in monitor mode while listening on specific Wi-Fi channels. Each received probe request allows the access to typical information from the client device as, for example, the source device MAC address. A person with more than one Wi-Fi enabled device is detected as more than one person, and people that don't carry a Wi-Fi enabled device are not detected.

Relating to the object detection module, it was decided to use a different tool from what already was being used in IT2, so that the processing rate of the video stream could be improved. A Software Development Kit from *Nvidia* was chosen to develop this module.

Lastly, it is also important to calibrate both of the detection modules and analyze the areas where the smart lamp posts are installed to check for possible false positives cases.

# 5 Requirements Gathering

Data needs to be gathered in two different locations, *Cais da Fonte Nova* and IT2. On both locations it is necessary to capture the number of people on the sidewalks and the number of vehicles passing by. In *Cais da Fonte Nova* specific case it is also a goal to capture the number of people inside the *moliceiros* as well as to count the number of *moliceiros*.

To obtain data and in order to increase its accuracy both wifi sniffing and object detection from the feedback of the camera will be used. Comparing the data acquired from both methods, the variations of the flow of people and moliceiros can be calculated.

All the data acquired and treated must be shown on a dashboard. This dashboard needs to be intuitive and simple. Also, charts to show the data in a more visually appealing way will be used. Furthermore, the shown data can be filtered to allow the visualization of the variation of traffic in certain days and/or hours and other relevant information. Finally, live data will be available.

The target users are all the economic businesses in the area, that may use the data to improve their advertisement efficacy or for any other purpose, as well as the tourism and cultural branch that may find the information of the peak hours and flow of people useful. Finally, the main target is the Aveiro's Town Hall, that specially requested this information to use, for example, on the planning of certain activities.

# 6 Actors and Use Cases

After analysing what are the requirements of our project, we stipulated that our system has the following actors:

| **Actor** | **Role** |
|---|---|
| Guest | Can access the web application and obtain data regarding the detection of people, vehicles, two-wheeler vehicles in the IT2 and also *moliceiros* in the Cais the fonte nova. For example, event organizers, Aveiro's city hall, emergency services, nearby stores, tourism companies, etc. |
| User | Has access to data like the Guest and can access information regarding the position and number of people travelling in *moliceiros*. The User also receives information regarding sensor failures. |
| Administrator | Has access to all data, validates and modifies accounts user types and receives information regarding sensor failures |
| Sensor | Capture and send all the data to the broker in real-time. This englobes the monitoring Wi-Fi interface, cameras and APU/Jetson Nano |
| Broker | Stores all the data collected by the sensors and provides it to the subscribed clients in real time |

Table 1: Actors and its roles

The following subsection presents our project's Use Cases, which are divided into two different diagrams: one relative to the API and one relative to the Web App.

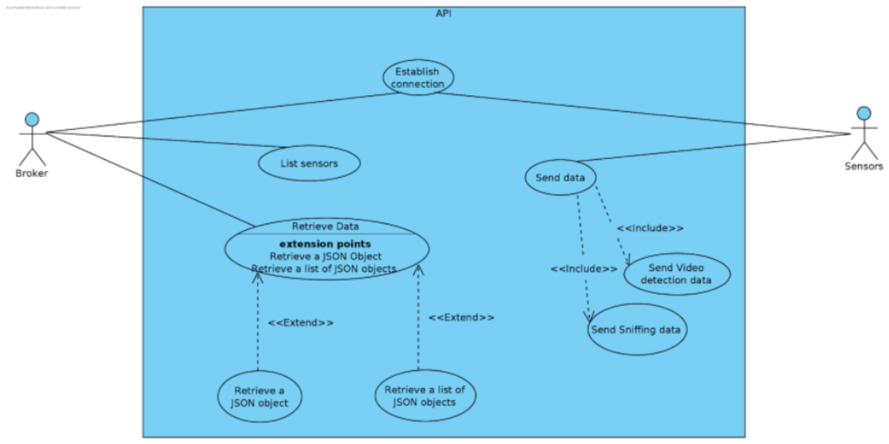


Figure 5: API - Use Case Diagram

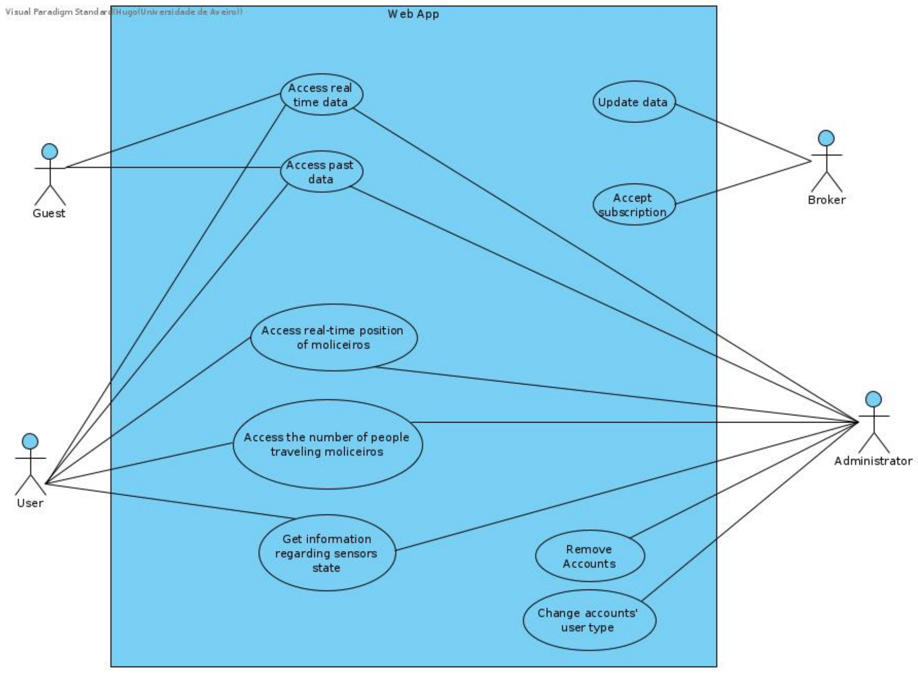

Visual Paradigm Standard(Hugo(Universidade de Aveiro))
Web App
Access real time data
Update data
Access past data
Accept subscription
Guest
Broker
Access real-time position of moliceiros
Access the number of people traveling moliceiros
Administrator
User
Get information regarding sensors state
Remove Accounts
Change accounts' user type

Figure 6: Web App - Use Case Diagram

# 7 Functional Requirements

In this section there are presented some functional requirements that our system must perform, in order to fulfil the web application's potential.

With these functional requirements the application can then satisfy the non functional requirements, which are going to be presented next (Section 8);

## 7.1 Sensor modules

| **Reference** | **Functional Requirements** |
|---|---|
| RFS-1 | It should be possible to count the number of devices emitting Wi-Fi requests in the surroundings |
| RFS-2 | It should be possible to estimate how many people are in the surroundings |
| RFS-3 | It should be possible to send the sensorial data to the broker |
| RFS-4 | It must be possible to estimate how many moliceiros pass in that area |

Table 2: Functional Requirements - Sensor modules

## 7.2 API

| Reference | Functional Requirements |
|---|---|
| RFA-1 | It should be possible to obtain information when there are changes in the data (atual value, average of values, list of values) |
| RFA-2 | The sensors of Wi-fi sniffing should provide the data in real-time |
| RFA-3 | The cameras should provide new data in real-time |
| RFA-4 | It should be possible to list and visualize the types of sensors used and their location |

Table 3: Functional Requirements - API

## 7.3 Front-End

| Reference | Functional Requirements |
|---|---|
| RFF-1 | It must be possible to visualize the geographical places where the data is being collected |
| RFF-2 | It must be possible to check the occupation in terms of people and and the traffic of moliceiros in the area |
| RFF-3 | Statistical analysis of the data from the sensors must be presented |
| RFF-4 | Historical data must be presented |
| RFF-5 | Must be possible to manage and to see details of the data collected by the sensors |
| RFF-6 | It must be possible to perform the user authentication in the platform |
| RFF-7 | It must be offered the possibility of sending notifications to the users when there are changes in the data |

Table 4: Functional Requirements - Front-End

# 8 Non Functional Requirements

In this section there are introduced the non functional requirements that must be fulfilled by the system.

## 8.1 Performance

| **Reference** | **Non Functional Requirements** |
|---|---|
| RD-1 | The application must present the data in real-time with the least amount of delay possible |
| RD-2 | The web application must be responsive and able to adapt to any device where it is being accessed |
| RD-3 | The web application load time should not be more than one second for users |

Table 5: Non Functional Requirements - Performance

## 8.2 Usability

| Reference | Non Functional Requirements |
|---|---|
| RU-1 | The website's interface has to be user optimized and easy to interact with |
| RU-2 | The solution must be versatile so that future sensors can be added or future functionalities to the API |
| RU-3 | The Web Application must allow to view information in Portuguese as well as in English |

Table 6: Non Functional Requirements - Usability

## 8.3 Security

| Reference | Non Functional Requirements |
|---|---|
| RS-1 | The data received must not have private information of the devices and respective users before being sent to the broker |
| RS-2 | Only administrators can manage the sensors |
| RS-3 | The access to the sensor data must be managed by the system |
| RS-4 | Only the admin can access and change the sensors information |

Table 7: Non Functional Requirements - Security

## 9 Domain Model

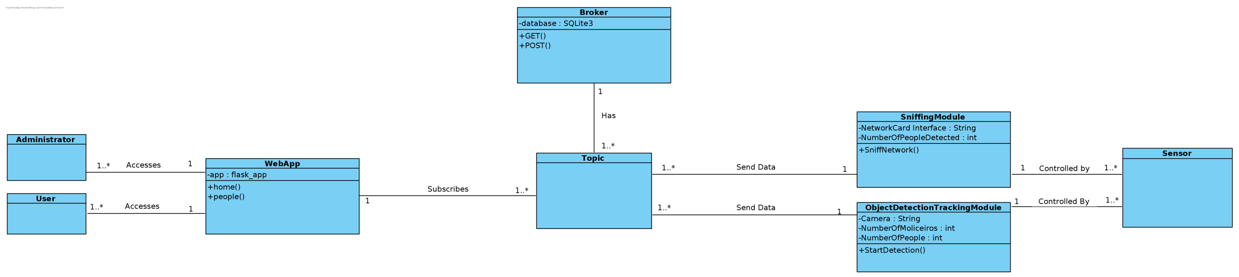


Figure 7: Domain diagram

The domain model of the project (Figure 7) represents the real-world conceptual classes of the system, their attributes and how they are associated. In the right side of the diagram, it is shown that the system has 2 modules: Sniffing and ObjectionDetectionTracking which both use one or more sensors to work. The SniffingModule has only one task "SniffNetwork" which is responsible to constantly looking for new devices and capture its MACs addresses whereas the ObjectionDetectionTrackingModule has the task "StartDetection" which will enable it to detect objects in real-time. All these detect values are stored within each module and posteriorly are sent to the respective topic of the Broker. The broker also allows its topics to be subscribed in order to receive a message with the data sent to the topics. An example of subscription to these topics is the WebApp which through this method is able to display all the values obtained by sensors and sent to the broker, in real time. The WebApp has two different roles with different permissions, that being the Administrator and the User. The latter can see in real time all the object detection values whereas the Administrator manages the WebApp in case of an unexpected error and has access to private information.

## 10 Architecture Diagram

In this section the architecture of the project is presented (Figure 8). There are three different processing tools used in this architecture which are Jetsons Nano, APU2e4 and OBU. Each JetsonNano is responsible for processing data relative to each post where a camera is placed and the streams of video captured are analyzed frame by frame in order to detect two wheelers, cars and people. The APU is entitled to capture devices through Wi-Fi sniffing that pass nearby each post where the APUs are placed. Finally, the OBUs are similar to the APUs but instead of being located inside a post, the OBUs are placed inside *moliceiros*, however the purpose is very similar, count devices but inside a *moliceiro*.

All this data captured and processed by each sensor and processing tool is sent to a broker through the respective topic through a MQTT connection where the data is persisted in a SQLite Database. This data is available to the WebApp (Flask) through an API which allows the WebApp to show real-time data of number of devices detected as well as other objects detected through video analyzis. There is also a MQTT connection between the WebApp and the MQTT Broker as the WebApp can subscribe the topics directly.

The web server (Gunicorn) interacts with the Web service and communicates with the Reverse Proxy Server (Nginx) through a Unix socket. The Reverse Proxy Server manages the load balance of requests in order to maintain the system free of shutdowns to a huge amount of requests. The user visualizes the web application in his device which can be in a desktop or mobile browser as the WebApp supports both in responsively, sending and receiving HTTPS requests and responses.

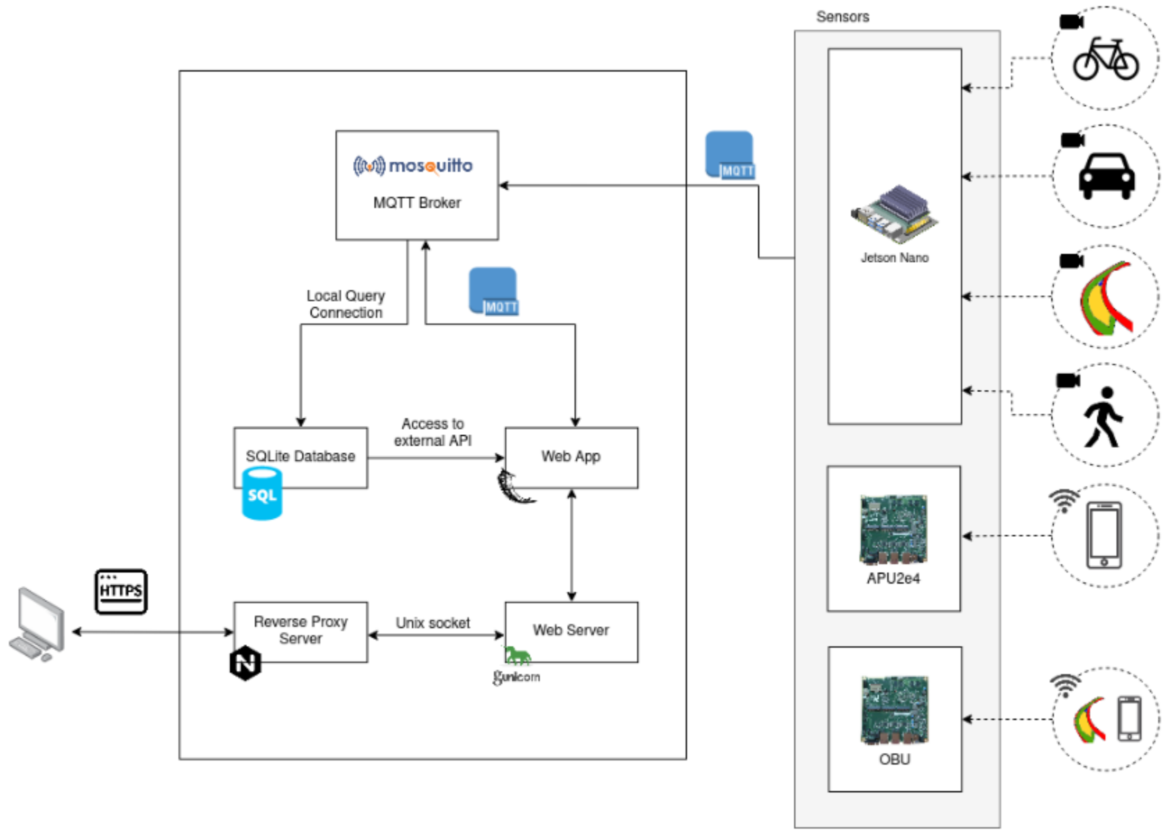


Figure 8: Architecture diagram

## 11 Deployment Diagram

This section shows the deployment diagram of the project (Figure 9). In the diagram it is possible to view the several components of the system and how they interact between them.

In the bottom, is possible to verify that the sensors are running in a Jetson Nano and in APUs in order to obtain real-time data. This data is sent to the Broker through a MQTT connection which posteriorly is sent to the Database Server and to the WebApp. The WebApp was developed using Flask and uses Gunicorn as Web Server. The data, after being persisted in the SQLite Server, is also sent to the WebApp. In order to connect to the WebApp to see the values displayed, the user device has to establish a HTTPS connection with the Nginx, which is the ReverseProxy, so that this application can manage the requests made to the WebApp and not overload it.

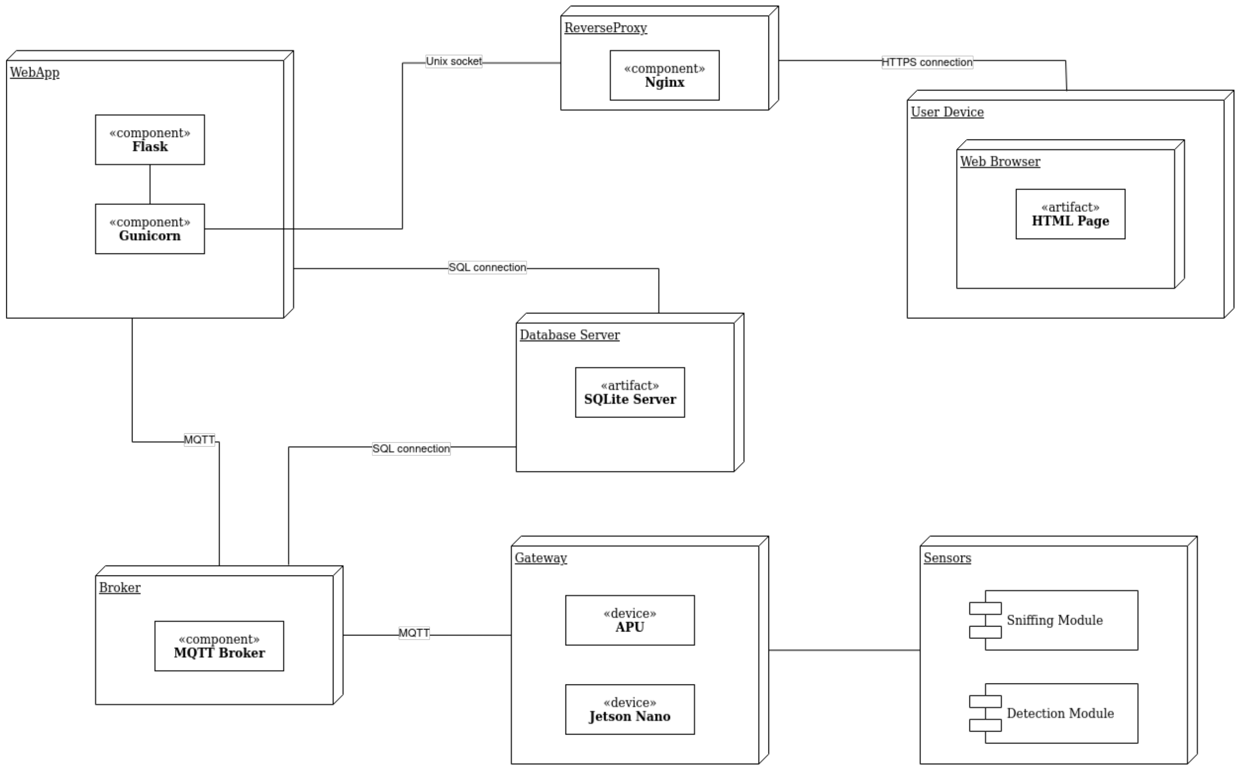


Figure 9: Deployment diagram

# 12 Implementation

## 12.1 Sniffing Module

### 12.1.1 Introduction

The sniffing module is written in Python3 and runs as a service on the Smart Lamp Post's APU. It is responsible for capturing probe requests packets and send to a broker values about current devices as well as unique devices that have passed by the post, in real-time. Moreover, it is running on 2 OBUs that are placed inside 2 *moliceiros* in order to estimate the number of people inside each *moliceiro*.

### 12.1.2 Configuration

In this section, it is shown how to configure the Smart Lamp Post's APU in order to enable monitor mode and run the sniffing module on it. Firstly, the package **wireless-tools** has to be installed: **sudo apt install wireless-tools**. In order to set a interface of the network adapter in monitor mode there the following steps have to be made:

Listing 1: Setting up a network interface in monitor mode

```
ifconfig wlan1 down
iwconfig wlan1 mode monitor
ifconfig wlan1 up
```

After this initial setup, the requirements of the sniffing module have to be installed, firstly the **tshark** has to be installed by doing the following command in a terminal: **sudo apt install tshark**. Then, a python venv has to be created and the requirements have to be installed.

Listing 2: Requirement installation

```
python3 -m venv venv
source venv\bin\activate
pip install -r requirements.txt
```

For testing purposes, the module can be run doing the following command on the terminal:

Listing 3: Test Sniffing Module

```
python3 pyshark_ps.py wlan1
```

In order to start capturing packets at all time, the module in the APU has to be running inside a service. Firstly, a systemd service has to be created inside /etc/systemd/system and the it has to be created a file named **wifisniffing.service**.

Listing 4: Create a systemd service

```
sudo su
nano /etc/systemd/system/wifisniffing.service
```

Then, the following lines have to be added inside the file which was created to configure the service:

Listing 5: Configuring the systemd service

```
[Unit]
Description=Wifi Sniffing Service
After=multi-user.target
```

```
[Service]
WorkingDirectory=/root/WiFi_Sniffing
User=root
Type=idle
ExecStart=/root/WiFi_Sniffing/venv/bin/python3 /root/WiFi_Sniffing/pyshark_ps.p$
Restart=always

[Install]
WantedBy=multi-user.target
```

Lastly, to run the sniffing module in the APU, the service has to be started by doing the following commands:

Listing 6: Starting the service

```
sudo su
systemctl daemon-reload
systemctl start wifisniffing
```

### 12.1.3 Method used

The module uses PyShark which is a wrapper for TShark (TShark is a terminal based version of Wireshark). In order to capture WiFi packets, it was turned on monitor mode on wlan1 interface of the APU's wireless network adapter. The PyShark's capture filter is set to probe requests, which are packets that are sent by a device scanning for access points in the area and are sent periodically. From these packets it is possible extract each device's mac address.

As probe request packets are being captured by PyShark, they are processed by a function, which filters packets that only are sent by client devices searching for access points in broadcast. Therefore, all AP's sending beacon frames advertising their SSID as well as devices already connected to a certain SSID/access point are all dropped. After this process, the function retrieves the mac address from the packet and marks the current time. This pair of values are saved this in a dictionary and represent the current list of in range devices. Afterward, the same mac address is introduced into the unique devices set, in order to prevent repetitions of the same mac address. In the end of every iteration of this function another function is called, which goes through the dictionary of the current devices and checks the difference between the current time and the time when each mac address was captured and if the difference is greater than 5 seconds that pair of value stored in the dictionary is removed.

### 12.1.4 Broker and datasets

This data is sent to the IT's central datacenter with a connection to the IP address and a port of the broker installed on the Jetson's post and publishes messages on 2 different topics: "sniffing/current" and "sniffing/unique". The first topic is where the current number of devices is published every 5 seconds while the second topic contains the number of unique devices, sent every 10 minutes.

The current devices being detected through Wifi Sniffing are stored until they don't send probe requests for at least 5 seconds. After these 5 seconds they are removed from the list that stores them. These current values are sent to the central broker every 5 seconds, in order to be used by the Web App, showing the number of devices captured in real time. The unique values are stored before being sent to the central broker, every 10 minutes. This prevents an accumulation of values and of misleading data as one person can pass near the Smart Lamp Posts during the morning and during the afternoon only being counted once. This implementation also allows the calculation of the average unique devices that were detected. In order to achieve this, the code calculates the average of all the captured devices in intervals of 1 minute. After 10 minutes, all values that were stored are excluded from the set. In order to get the detected values in intervals of 1 hour, the Web App gets the average of all the 10 minutes interval values in that specific hour from the API.

### 12.1.5 Improvements

Currently the sniffing module is working with good accuracy and power efficient, however throughout the development of this script there was some setbacks that eventually were resolved.

During the first tests, although being successfully capturing probe requests and eliminating them after the threshold time, the script was using 100% of the APU's processor power which eventually led to its shutdown. In order to fix this issue, the code was rewritten with a new program to capture packets: PyShark. These changes resulted in a significant decrease in the usage of the CPU which is fixed around 35%.

After more tests and packet analysis, it was discovered that the script was storing MAC addresses that belonged to Access Points as the SSID of the packet had a value. The first step was to find out more about SSIDs, Wildcard SSID and after the research the function handling the packets captured was changed so that it filtered out all the packets with SSID name as well as not broadcasted packets. The latter were also a problem because some packets were sent by devices already connected to a network and could led to misleading values.

The last problem that had a major impact on this module was finding a balanced time for how long a MAC address should be stored since its last probe request captured by the sniffing module. When adjusting this parameter, high time difference between last MAC's probe request and the current time led to over-accumulation whereas a small time-to-live of MAC address would result in more accurate results.

## 12.2 Estimation of people inside *moliceiros*

### 12.2.1 Introduction

In order to estimate how many people are inside *moliceiros* the creation of OBUs was the chosen option. Each *moliceiro* carries one OBU running the sniffing module presented before. Each OBU is composed by a water and heat resistant box, one APU, two batteries and one GPS module, as it is possible to see on Fig. 10.

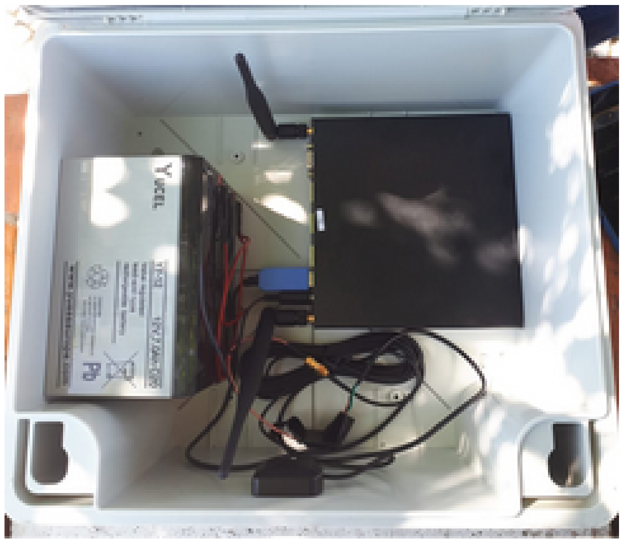


Figure 10: Inside of an OBU

### 12.2.2 Method used

The APU is running the Sniffing module and is also responsible for sending the information from it as well as the GPS coordinates to the stations around the *moliceiro*. From these stations the data is then forwarded to the central database at IT2. The reason for this indirect way of sending information is because the OBUs do not have direct connection to the internet/ATCLL's network. So, using the stations near the *ria de Aveiro* the OBUs send data through WAVE to these stations, whenever a *moliceiro* is in range of one and from the stations the data is then sent to the central server.

### 12.2.3 Improvements

During the first field tests some problems were detected and later solved.

Firstly, the signal strength was very low and unstable. Later it was found out that it was due to the position of the OBU beneath a wooden plank that was interfering with the signal. A repositioning of the OBU to the bow of the *moliceiro* as show in Fig. 11 improved considerably the signal strength and stability.

Secondly, a higher than expected battery consumption due to the gps module addition (which was not considered when calculating the battery necessary) ended up with the APU turning off before the end of the work hours. The solution for this was the addition of another battery, connected in parallel with the original one, therefore doubling the working time of the APU and allowing it to be more than enough for the working hours of the *moliceiros*.

Finally, since the OBUs are running the Sniffing module, the same improvements that happened to this module were also applied here. In particular, a very high number of devices was being detected and so a balanced time for how long a MAC address should be stored since its last probe request captured by the sniffing module had to be found.

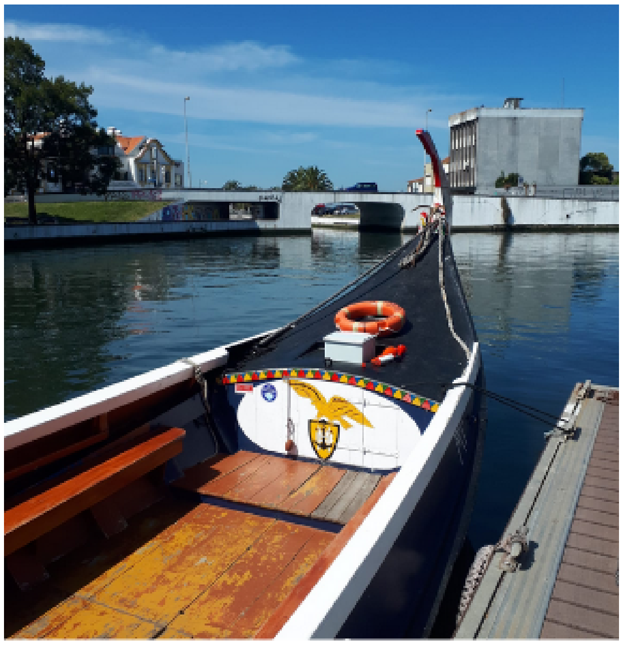


Figure 11: OBU on a *moliceiro*

## 12.3 Detection Module

### 12.3.1 Introduction

Object Detection through video frames analysis was the second method used to detect people, vehicles and *moliceiros*. Each of the two smart lamp posts used for this project contained a video camera that can be rotated along 360º degrees. The process of the frames obtained from the live streams of these cameras allowed the analysis of the movement of people, vehicles and *moliceiros* in the areas surrounding the posts. The detection module runs on a Jetson Nano (Figure 12), which is a small and powerful computer that allows the execution of multiple neural networks in parallel dedicated to image classification, object detection, segmentation, and speech processing.

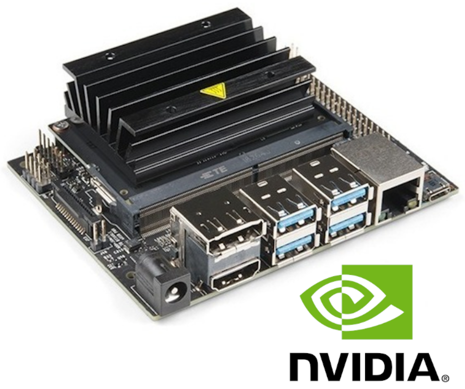


Figure 12: Jetson Nano

### 12.3.2 Method used

In order to analyze the video frames with a significant processing speed, a Software Development Kit developed by *Nvidia* was chosen as the main tool to use in this module. This kit is called *DeepStream* and it allows obtaining complete streaming analytics for AI-based multi-sensor processing, video, audio and image understanding.

One of the python bindings that the development kit offered was particularly useful to obtain information regarding the number of objects crossing the camera's point of view in particular frames. This script was adapted in order to obtain the number of people, vehicles and two-wheeler vehicles that appear in each frame. The script decodes data from a URI into raw media to accept any type of input such as an RTSP URL or a mp4 file, or any GStreamer supported container format. A batch of frames is generated and infered for better resource utilization. The stream metadata is extracted from the inference, and it contains useful information about the frames in the batched buffer. The achieved process rate of the received frames was 30 frames per second, which is a very good performance.

In Figure 13 a person is being detected and that information is being displayed in the dashboard.

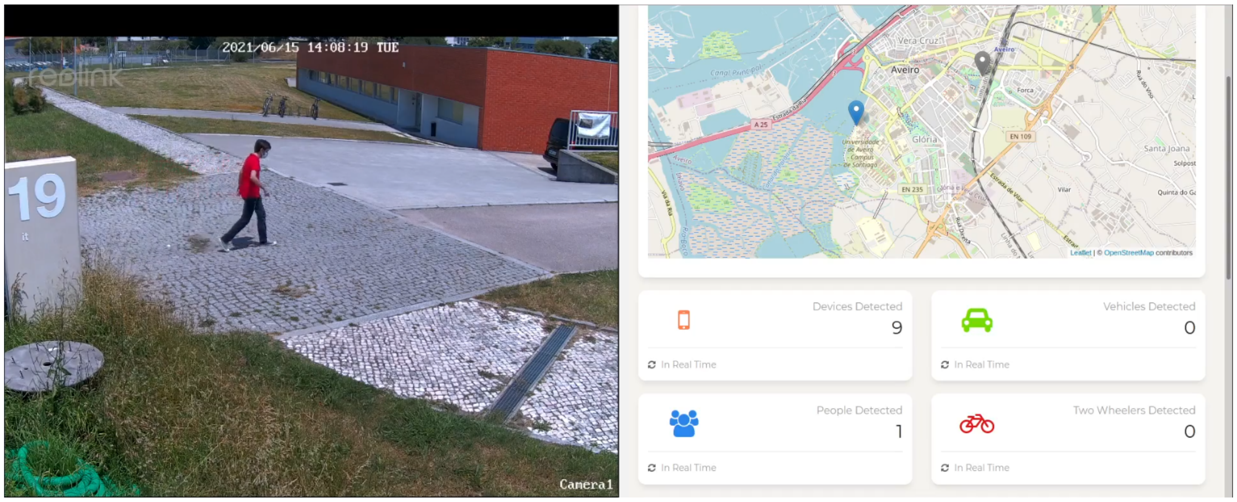


Figure 13: Person being detected

The data related to the detection of people and vehicles is obtained through the use of an object detection model named *ResNet10* running on a Jetson Nano.

This model was trained to detect vehicles, people, two-wheeler vehicles and road signs. Since the last type of object was not relevant for this project, its detection was excluded from the script.

To obtain the data of the numbers of *moliceiros* that appeared in the camera's view, a different model had to be used. The solution adopted was to use a python library named *OpenCV* that ran the SSD Mobilenet v3 model, which has over 90 labels. However, only the label "boat" was necessary to achieve our goal. A new script was developed to perform this detection and deployed on a Jetson Nano.

### 12.3.3 Increasing the accuracy

Throughout the capture of the data in this module, a few problems were detected. These problems were leading to the gathering of data with a low level of accuracy and thus, needed solving.

The first one was related to the speed with which the frames were processed. The video stream sent 25 frames per second, but initially the processing rate was about 12.4 frames per second. This lead to a delay when processing all the information that arrived from the camera stream. Naturally, this would have big consequences when the script would run for many hours straight, because the delay in processing would only increase, leading to the collection of wrongs values, since after a person passed the camera that information would only arrive a couple of seconds later, when the person had already left the camera's viewing point.

This problem was solved by changing the video sink used to process the frames. The video sink is the destination for the media streams, and it was configured to drop frames that arrived later than the default threshold for observing out-of-sync frames.

The second problem observed was that there were frequent false detections of objects that were not in the image observed in the video stream. These false detections would appear due to the presence of ”noise” in the images, that was not ignored by the model and instead was assumed to be a vehicle or a person.

To correct this problem, the false detection patterns were captured in recording and all the coordinates of the detected objects were saved. This allowed the identification of the zones in the frame that originated the most ”noise”, and it was possible to filter out these zones due to the fact that they were mostly in areas where it was highly improbable to detect one of the chosen objects, like for example the sky or corners cluttered with objects that don't allow the passage of people. By filtering this ”noise”, false detections became rare and the accuracy of the captured data increased.

Lastly, another main problem that was confronted was related to the detection of *moliceiros*. In the beginning phase of the detection, this type of object was never detected. The restriction of the camera zoom provided a difficulty towards the detection of the boats that crossed the camera's view. For this problem, the solution found was to adjust the resolution of the frames that would be processed. It was verified that when the frames would be resized to a smaller resolution, the model would easily detect the *moliceiros*, even from far away. Applying this knowledge, this solution was implemented, and it even allowed a lower processing CPU usage, due to the fact that processing a smaller resolution requires less effort.

## 12.4 Web App

### 12.4.1 Introduction

When developing the Web Application, the main goal was to provide a platform in which the user could have access to information about the fluctuation of people, vehicles and *moliceiros* in some areas in *Aveiro* city. To do that, the data was first collected and then displayed in real-time as well as historic values.

### 12.4.2 BackEnd

For the BackEnd service, it was used Flask, which is a web framework written in Python, running in a Gunicorn server and using NGINX as a proxy server. To obtain all the data from the sensors of both modules, Detection Module and Sniffing Module, a Rest API was developed.

The Rest API possesses a set of endpoints developed with the help of the Flask framework, making it easier for the developers and users to have access to the data and even interact with it to have access to past values. To specify the endpoints, a Rest API design was constructed so that the API resources and their functionalities would be mapped in different endpoints.

| **Action** | **Method** | **Description** |
|---|---|---|
| Create | POST | Create a new unique object |
| Read | GET | Obtain information about a object or a collection of objects |

Table 8: Resume of API's Actions

**Detection and Sniffing's Data**

For the Detection and Sniffing data, similar structures were used, by having an endpoint that returned all the data collected and another with only the unique values. These unique values represented the number of unique people, for example, in an area during a certain time period.

| **Endpoint Base Path** | **Resource** |
|---|---|
| /api/sniffing | Information related to the sniffing module |
| /api/sniffingunique | Information related to the sniffing module |
| /api/people | Information related to the detection of people |
| /api/peopleunique | Information related to the detection of people |
| /api/vehicle | Information related to the detection of vehicles |
| /api/vehicleunique | Information related to the detection of vehicles |
| /api/twowheeler | Information related to the detection of bicycles |
| /api/twowheelerunique | Information related to the detection of bicycles |

Table 9: Resume of the Detection and Sniffing resources

For accessing the historical data, the API had to access an external API since the data is being persisted in an IT2's database. Therefore, to obtain the necessary data successfully, the API had to request a daily token to the NAP's group API. This followed the same method used to obtain the camera's token. For instance below (Listing 7) there's the URL parameters used to obtain the persisted data for a specific time interval.

Listing 7: URL parameters for accessing persisted data

```
start = [integer]
end = [integer]
bucket = day|hour|minute
group = count|avg|sum|min|max

Example:
   https://dev.aveiro-open-lab.pt/api/vehicle?
start=1621300462000&end=1621372462000&bucket=hour&group=count
```

In the example displayed before (Listing 7), there's the URL for accessing the data related to the vehicles detected in the Post 1, IT2, on the 18th of May between 1:00 and 21:00 hours (1621300462000 - 1621372462000), and it was chosen to display the number of vehicles detected for each hour in the time interval. In the snippet below (Listing 8), there's a part of the data returned by the API.

Listing 8: API's Response to the Request shown in the example on Listing 7

```
{"urn:ngsi-ld:Vehicle:aveiro_detection:p1":{
    "bucket":[
        "2021-05-18T01:00:00.000Z",
        "2021-05-18T02:00:00.000Z",
        ...
        "2021-05-18T21:00:00.000Z"],
    "count_value":[
        10,
        14,
        ...
        2]
}}
```

**Services' Status**

To make the web application aware when each service, in different detection modules, is currently running, or not running, a few endpoints were created.

| **Endpoint Base Path** | **Resource** |
|---|---|
| /api/sensors/p22_apu | Status of the sniffing module in Post 22 |
| /api/sensors/p1_apu | Status of the sniffing module in Post 1 |
| /api/sensors/pei_jetson | Status of the detection module in Post 1 |

Table 10: Resume of the services status

**Get Data from Broker**

Firstly to obtain the data from the broker the application's MQTT client had to be configured (Listing 9), this was made in the file that initializes the Flask application, '__init__.py'.

Listing 9: Example of a MQTT's client configuration

```
#Flask app initialization
def create_app():
    # ...
    #Borker Configuration
    app.config['MQTT_BROKER_URL'] = 'atcll.exemplo.pt'
    app.config['MQTT_BROKER_PORT'] = 1999
    app.config['MQTT_REFRESH_TIME'] = 1.0
    # ...
```

After, the MQTT client subscribed to the topics with the information to be displayed on the web application and then the MQTT messages were handled by storing the information acquired in global variables. For instance in the code snippet below (Listing 10), there is the configuration of the MQTT subscriptions and the function that stores all the data received through the MQTT messages received.

Listing 10: MQTT client's connection and message reception handlers (main.py)

```
#mqtt connection handler
@mqtt.on_connect()
def handle_connect(client, userdata, flags, rc):
    # MQTT client subscribes to the topics
    mqtt.subscribe('p22/sniffing/current')
    mqtt.subscribe('p1/sniffing/current')
    #...
    mqtt.subscribe('p22/sniffingmoliceiro')

#mqtt message reception handler
@mqtt.on_message()
```

```
def handle_mqtt_message(client, userdata, message):
    global p1_sniffing_data
    #...
    #the message is loaded to a Json
    topic=message.topic
    payload=message.payload.decode()
    json_rec = json.loads(payload)

    if topic=='p1/sniffing/current':
        # if the topic of the message received is related to the number of
           devices detected by sniffing on the post 1, IT2, stores the value in
           the correspondent global variable
        p1_sniffing_data = json_rec["value"]
    #...
    elif topic=='p1/detection/twowheelers/current':
        p1_detection_twowheelers = json_rec["value"]
```

**Display Data on Web Application**

To display the data obtained from the Broker, some sockets were opened using Server Sent Events (SSE). With SSE the application can exchange the data from the Flask application, more precisely the python file, to the HTML file, through a ”text/event-stream” type Response.

For instance in the code snippet below (Listing 11), there is the function that sends the Response to the HTML file with the information related to the real time values, which is accessed through the path ’/realtimedata’.

Listing 11: Response from the main.py to the HTML file (main.py)

```
#p1 sniffing real time data
@main.route("/realtimedata")
def p1_peopledetection_chart_data():
    return Response(generate_all_realtime_data(), mimetype="text/event-stream")
```

Through the Response it is sent the data obtained from the broker, this data is formatted to a Json in the function 'generate_all_realtime_data()'. Like is observed below in the code snippet (Listing 12), the data stored on the global variables is all gathered in a Json file to then be sent to the HTML through the Response explained previously.

Listing 12: Json sent to the HTM (main.py)

```
#current data loaded to a Json
def generate_all_realtime_data():
    while True:
        # ...
        json_data = json.dumps(
            {
                "time": datetime.now().strftime("%H:%M:%S"),
                "sniffing_p1": p1_sniffing_data,
                "sniffing_p22": p22_sniffing_data,
                # ...
                "sniffing_mol1": p_sniffingmoliceiro1,
                "sniffing_mol2": p_sniffingmoliceiro2,
            }
        )
        yield f"data:{json_data}\n\n"
        time.sleep(1)
```

Then in the HTML file, on the places were the data is going to be displayed, in this case cards, because we are talking about real time values, the information is updated with the data received from the respective SSE (Listing 13). That way the information got from the broker is shown in the web application in real time.

Listing 13: Part of the message handler related to the real time data (template.html)

```
//handles the Response received from the Flask app
const source = new EventSource("/realtimedata");
source.onmessage = function (event) {
    var data = JSON.parse(event.data);
    if (poste == '?location=22') {
        //if the page is currently showing the information related to the post
            22, Cais da Fonte Nova, displays the data related to that post on
            the cards
        $('#sniffing_div').html(data.sniffing_p22);
        $('#peopledetection_div').html(data.detection_p22);
        //...
    } else {
        //otherwise loads the data related to the other post in the cards
        $('#sniffing_div').html(data.sniffing_p1);
        //...
    }
    //...
}
```

In order to display the past values in the graphs, similar SSEs were implemented, one for each post. For showing the data in the graphs a similar Response reception handler was also used, apart from the fact that for past values the data is displayed in the graphs, instead of the cards.

### Translation of Web Application's UI

The Web Application was initially developed for Portuguese speakers, but in order to be accessible to everyone, a language translation system was implemented. For the Language translation it was Babel, which has a Flask extension, Flask-Babel, that makes the integration with Jinja, Flask's template engine, easier.

Firstly, in the HTML files, the texts to be translated are marked with a Babel's specific format and then the Babel's script, pybabel, was used to extract the expressions to translate into a file, 'messages.po'. Afterwards the translation had to be done manually by completing the file with each expression's translation (Listing 14).

Listing 14: Example of a translation (messages.po)

```
#: templates/template.html:316 location of the expression
msgid "Notifications" # expression generated through the script
msgstr " Notificaes "   #  translation added manually
# ...
```

After all of the expressions' translations had been added to the file the pybabel is run again to generate the translation catalog from the 'messages.pot' file. Finally, the script is run one last time to compile the translation catalog to 'messages.mo', which is the file used by the application.

In order to switch languages through the web application a Babel configuration was needed. In the code snippet below (Listing 15), there's the configuration of the Babel for the UI's translation. And as can be seen in the 'get_locale()' function, which is called every time the language is switched or the page is loaded, the application loads the file accordingly to the language selected by the user.

The Web Application's expressions are changed, based on the file loaded, automatically by the extension.

Listing 15: Babel configuration (__init__.py)

```
#Flask app initialization
def create_app():
    # ...
    #Babel Configuration
    app.config['BABEL_DEFAULT_LOCALE'] = 'en'
    #creation of babel object for the app
    babel = Babel(app)

    #set up babel
    @babel.localeselector
    def get_locale():
        #if a language is selected the file with the selected language
           translations is loaded
        if request.args.get('lang'):
            session['lang'] = request.args.get('lang')
        #otherwise the file with the english translations is loaded
        return session.get('lang', 'en')
    # ...
```

### Camera's API

For controlling the cameras, some endpoints were also created. To have a successful connection with each camera, a new token had to be requested every time the status of the request to the Cameras' API was not successful.

| **Endpoint Base Path** | **Camera Movement** |
|---|---|
| /api/cam/up | Move camera up |
| /api/cam/down | Move camera down |
| /api/cam/left | Move camera left |
| /api/cam/right | Move camera right |
| /api/cam/upleft | Move camera up in the left direction |
| /api/cam/upright | Move camera up in the right direction |
| /api/cam/downright | Move camera down in the right direction |
| /api/cam/downleft | Move camera down in the left direction |
| /api/cam/zoomin | Zoom in the camera |
| /api/cam/zoomout | Zoom out the camera |
| /api/cam/focusin | Focus in the camera |
| /api/cam/focusout | Focus out the camera |
| /api/cam/irisin | Iris in the camera |
| /api/cam/irisout | Iris out the camera |
| /api/cam/auto | Start the camera's auto feature |
| /api/cam/autostop | Stop the camera's auto feature |

Table 11: Resume of the Cameras resources

**Users authentication**

Some endpoints were also created to manage the access to certain pages and to manage the authentication of users. To store the account's information, an SQLite database was created.

| Endpoint Base Path | Action performed |
|---|---|
| /login | Authenticates the user's account and logins if it is successful |
| /signup | Creates a new account |
| /logout | Logs out the current account logged in |

Table 12: Resume of the account management resource

#### Notification System

A notification system was developed for account users be aware when some services are down. To know when a service is down the paths related to the service's status, shown in Table 10, are accessed. Like the detection and sniffing information the services' status is also sent through a SSE, which a part of the configuration is shown in Listings 11 and 12. Then in the HTML file the notifications are shown based on the status, like in the code snippet shown below (Listing 16).

Listing 16: Part of the message handler related to the notifications (template.html)

```
//handles the Response received from the Flask app
const source = new EventSource("/realtimedata");
source.onmessage = function (event) {
    var data = JSON.parse(event.data);
    //...
    {% if current_user.is_authenticated %}
        //in case the user is using an account, a notification is shown if a
            service is down
        if (data.apu22_status != '0' && apu22_is_down != 1) {
            // if the APU22 is down then a notification with the message below
                is shown
            AddNotifications('Post 22', 'Error: Sniffing module is down',
               data.time);
            apu22_is_down = 1;
        }
        //...
    {% endif %}
}
```

### 12.4.3 FrontEnd

For the FrontEnd of the web application a dashboard template was used and modified to its needs.

**Home Page**

On the main page, *Moliceiros* and People Detection, it's displayed a map, where there are two markers, both of them representing the two locations where the two modules of detection are run. These two locations are IT2 and *Cais da Fonte Nova*. The map was implemented with the intent of making it more intuitive for the user to know from what location the values displayed are from and to navigate easier between them. It was implemented with the help of Leaflet, which is a JavaScript library.

The cards on the bottom of the map are displaying the values obtained in real time from the currently selected location. On the IT2 is shown the number of people, vehicles, bicycles detected by the camera and the number of devices detected on the area with the WiFi Sniffing. As for the *Cais da Fonte Nova* location it is displayed the number of people, of vehicles and of *Moliceiros* detected with the camera, and also the number of devices detected. For instance, on the image below (Figure 14), the values on the cards are from the IT2 location.

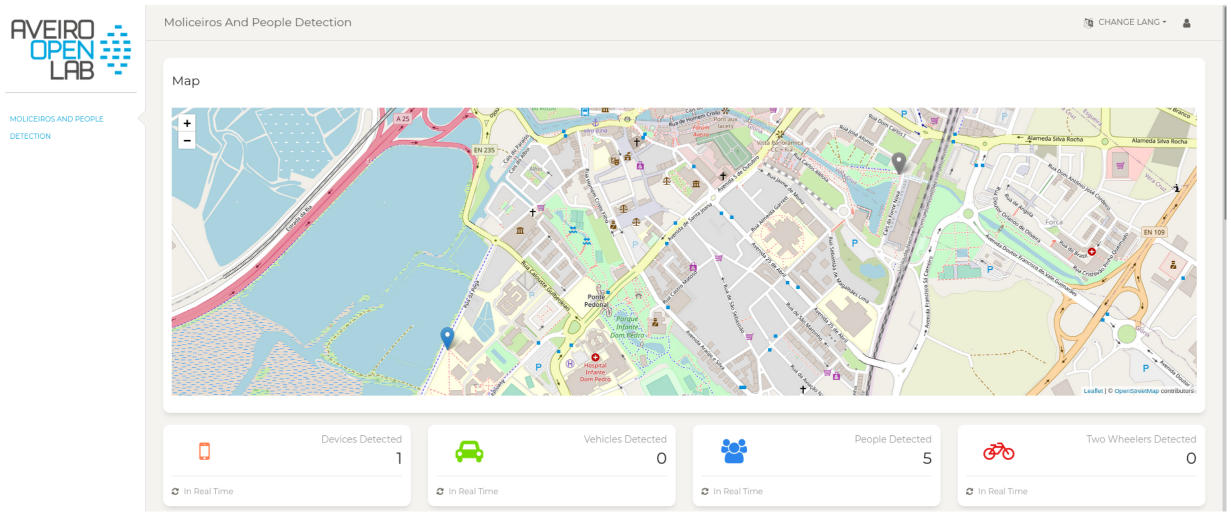


Figure 14: Home Page, showing the real-time values detected for the IT2 location

In order to display historic values some graphs were implemented for each location. One of the graphs shows the last ten values obtained, from the people detection and the device detection, and this one is updated in real time.

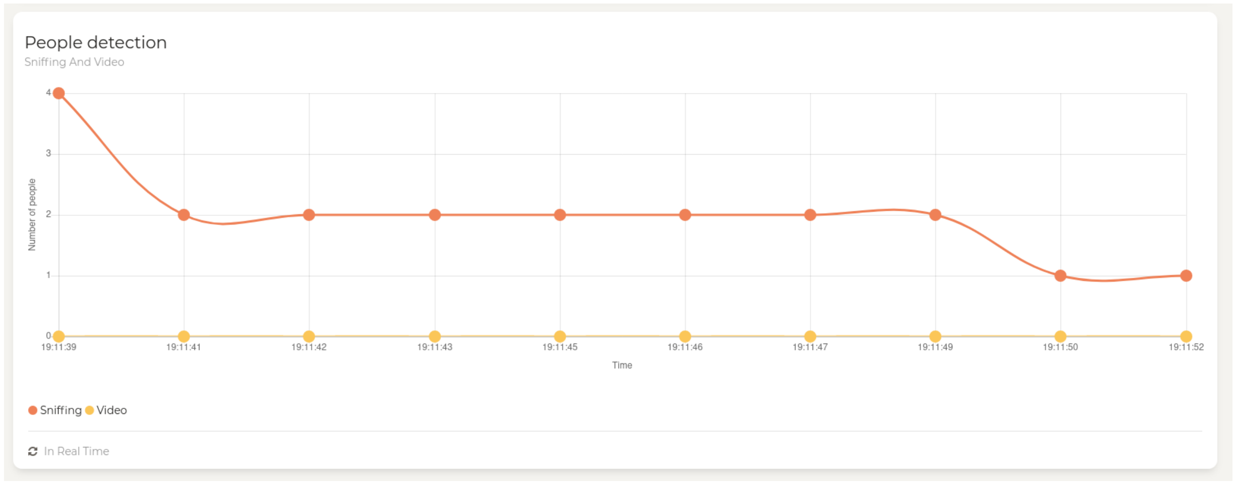


Figure 15: Graph showing with last 10 values

To provide a way in which the user could have a more in-depth perspective of the fluctuation of people, vehicles and devices it was implemented an graph were the values obtained on the last 24 hours are displayed, and this one is updated every 3 minutes.

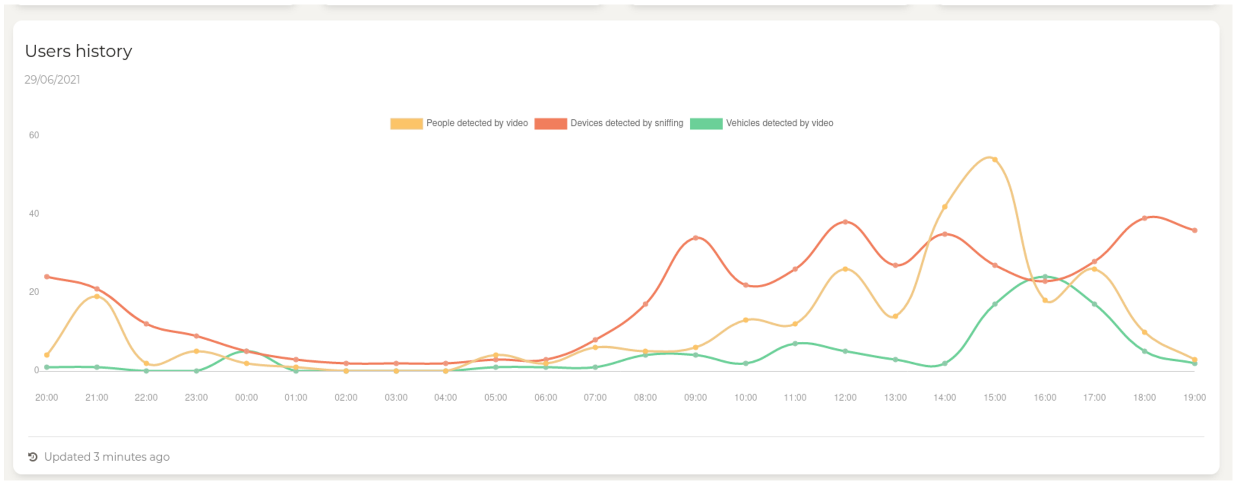
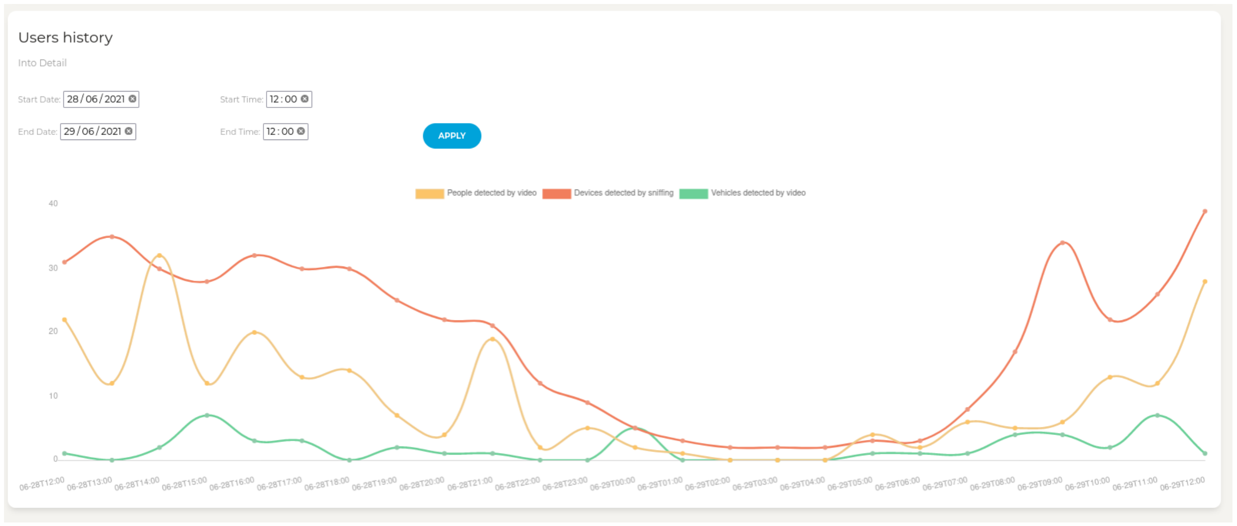


Figure 16: Graph with the values obtained on the last 24 hours

Furthermore, it was also implemented a way in which the user could give an interval, day and hour of the beginning and day and hour of the end, for which a graph with the values obtained in that time window was generated and shown to the user. As you can see in the image below (Figure 17), where the user wanted to get the values of obtained from the 20th to the 26th of June.


Users history

Into Detail

Start Date: 28/06/2021 Start Time: 12:00

End Date: 29/06/2021 End Time: 12:00

Figure 17: Graph with the values obtained on the time interval defined by the user

### ***Moliceiros* information page**

For displaying the information related to the movement of the *Moliceiros* and the people inside them, it was implemented another page. In order to access this page the user has to have an account.

This page is in part similar to the home page, as you can see below (Figure 18). On the top is shown an map in which the user can see the *moliceiros* move throught the *ria* of *Aveiro*, if they are in a journey. The information related to the number of devices detected in that instant in each *moliceiro* is show in the cards on the bottom of the map.

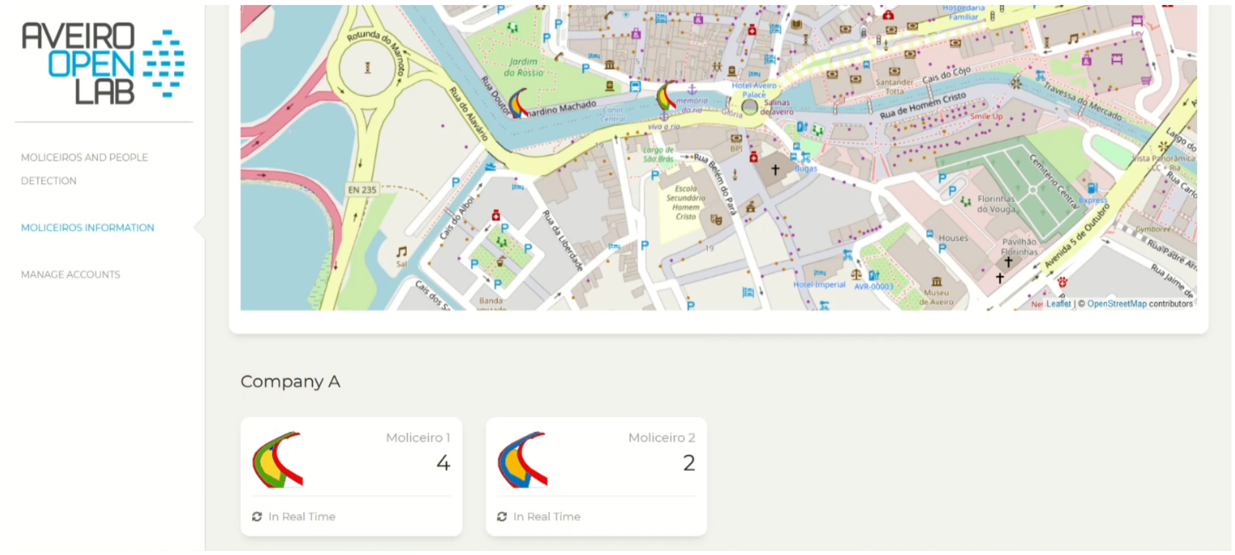
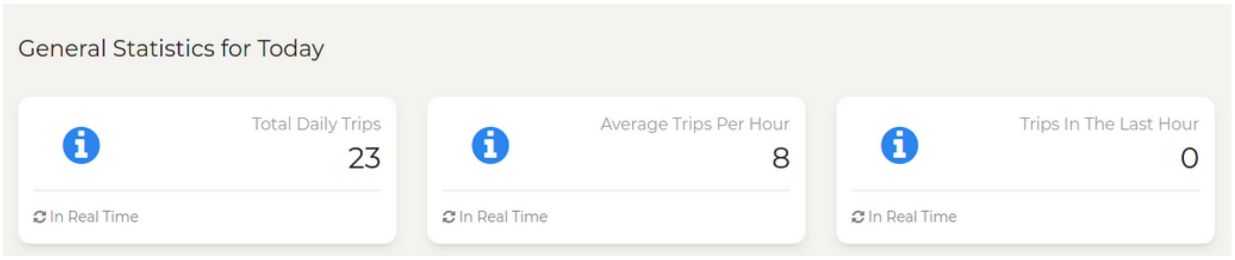


Figure 18: *Moliceiros* information page, where both *moliceiros* are in journey

Besides the real time values there are also displayed some daily statistical values, related to the total number of trips made in that day, the number of trips made in that hour and the number of trips made per hour.


General Statistics for Today

Total Daily Trips
23
In Real Time

Average Trips Per Hour
8
In Real Time

Trips In The Last Hour
0
In Real Time

Figure 19: *Moliceiros* statistical vinformation

Like on the home page, to show past values related to the devices detected on both *moliceiros* it is shown a graph with all of the values detected in the last 24 hours, along with it there is also an graph for which the interval it is defined by the user.

### Accounts Management

In order to make interdict for users to access some pages and resources an account management system was created. The accounts are separated in two levels of access, being the superior one the admin role and the other the standard one. The admin accounts are the ones in which the user has full access, one of resources being the management of the accounts.

Therefore an user that wants to create an account goes to the Sign up page where the fields have to be filled along with a security code. Apart from the security code, in order to make the account valid, an user with an admin role has to validate the account, that way the application has two security methods.

On the image below (Figure 20), there is the accounts management page. In this page the admins can view and manage the accounts, change permissions or delete them. On the top there are the the admin accounts and after the standard accounts. On the bottom there are the accounts that are not yet validated, waiting for the validation of an admin.

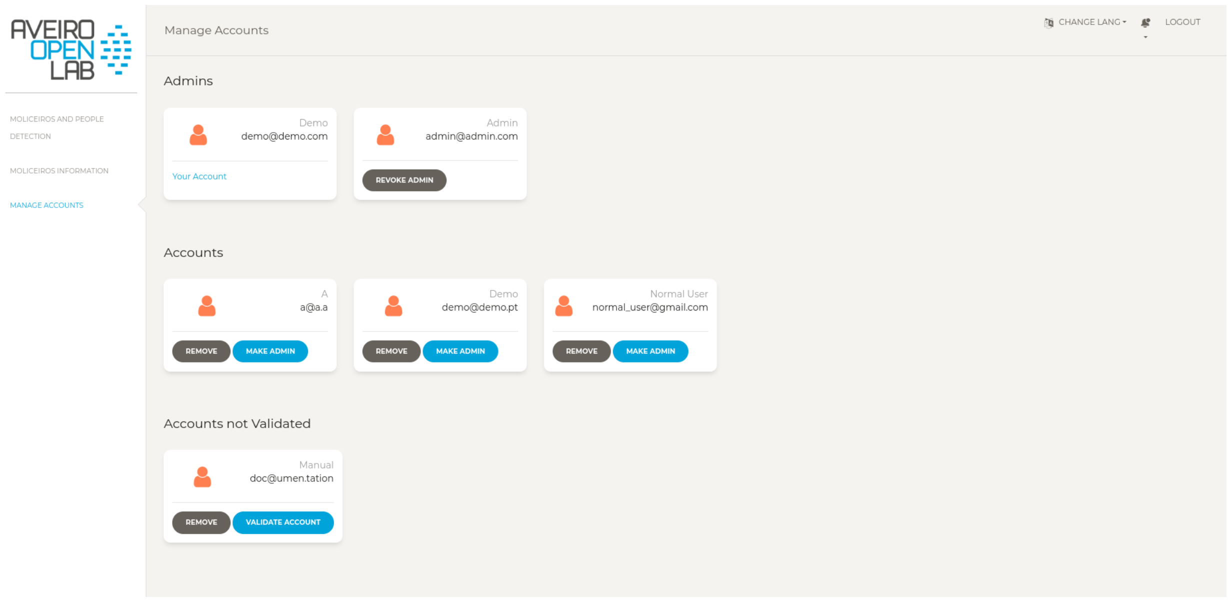
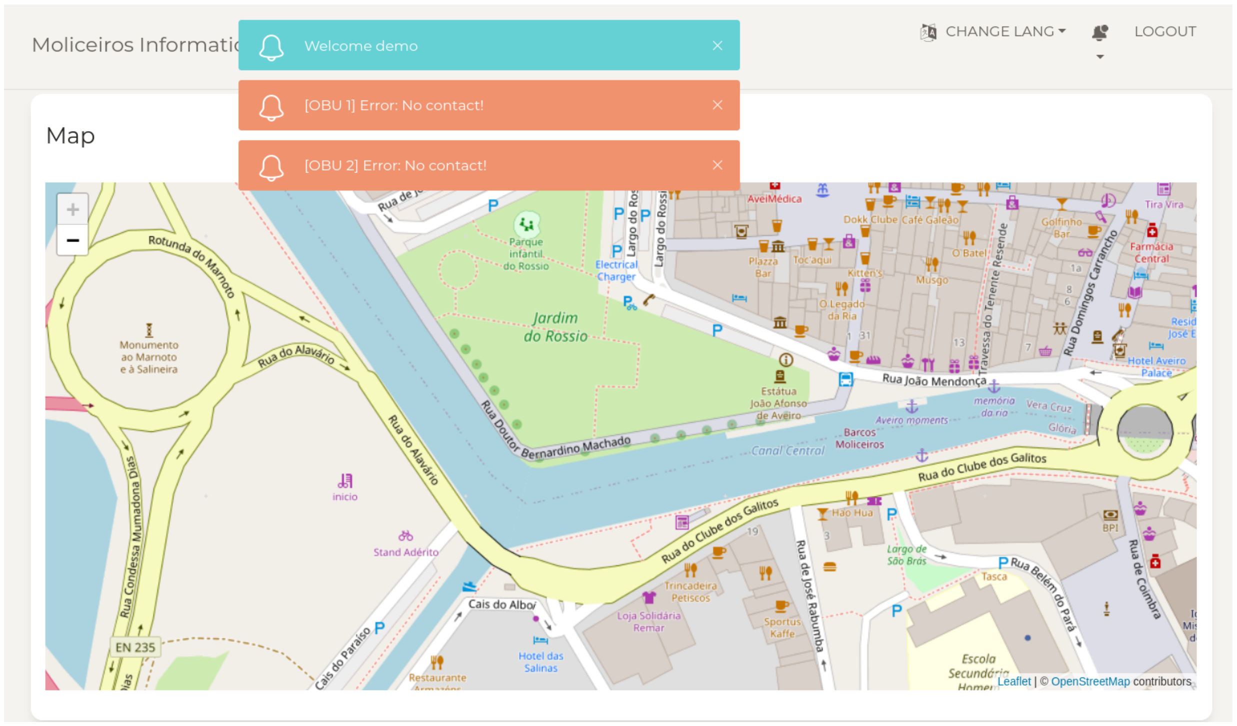


Figure 20: Accounts management page

### Notifications

One of the features that account users have access to is the notification system. With this system users can be notified when a service is down. For instance, looking at the image below (Figure 21) both of OBUs are down, meaning that it is not being collected any data related to the devices detected on the *moliceiros*.

Therefore with this feature an user can be notified when an service is down unexpectedly.


Moliceiros Informati

Welcome demo

[OBU 1] Error: No contact!

[OBU 2] Error: No contact!

Figure 21: *Moliceiros* information page, when the OBUs are not being detected

**UI Translations**

Another feature on the web application is the translation of the dashboard. This feature makes the web application more inclusive to everyone. As can be seen in the pictures by selecting a language through the drop down the names are changed accordingly.

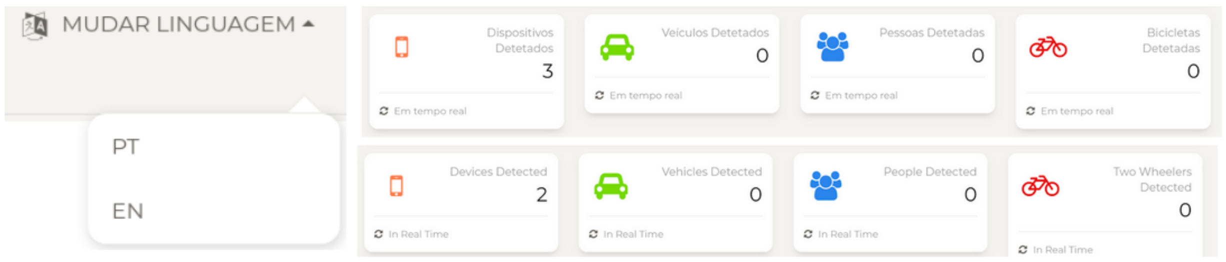


Figure 22: Web application's UI translation

# 13 Results

## 13.1 Wi-Fi Detection Module

The Wi-Fi Sniffing module was deployed in two different locations, IT2 and *Cais da Fonte Nova*. Part of the captured data can be seen in charts that present values captured during 24 hours in the same day for each location (Figures 23 and 24).

Using this module, it is possible to obtain the number of devices in the area. It is important to consider that one person may not be carrying a Wi-Fi capable device or may be carrying several, or may just have the Wi-Fi turned off and thus will not be detected. These factors influence the accuracy of the information captured by the Wi-Fi sniffing module. For this reason, comparing them with the numbers obtained from the image detection module is essential to verify the accuracy of the data. It was possible to conclude that despite not always counting every person, this module was effective and obtained good values. The reason found for this better accuracy is that the Wi-Fi Sniffing works in a 360 degrees angle, meaning that there are no dead angles where we are not detecting devices, while in comparison the Object Detection only gathers information from the angle that the camera is able to capture.

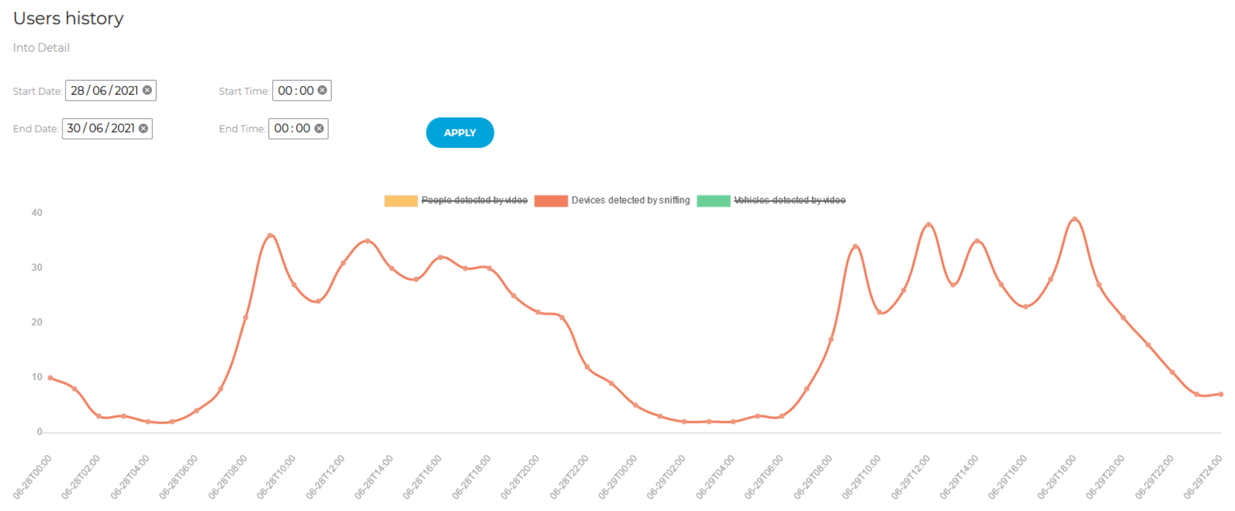


Figure 23: Devices detected by sniffing on the Smart Lamp Post in IT2 between 28/06/2021 (00:00H) - 30/06/2021 (00:00H)

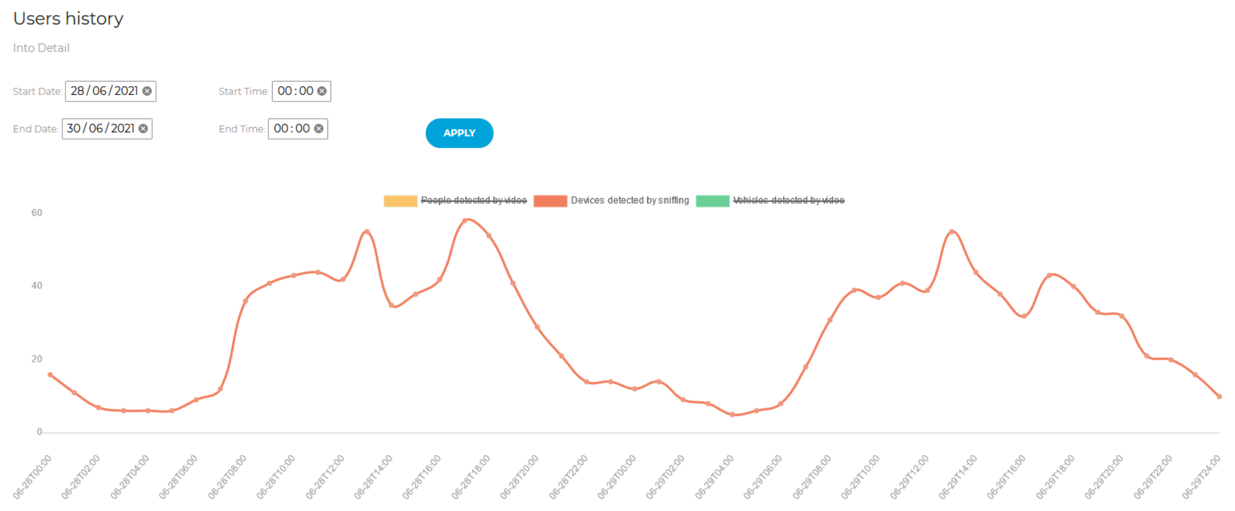
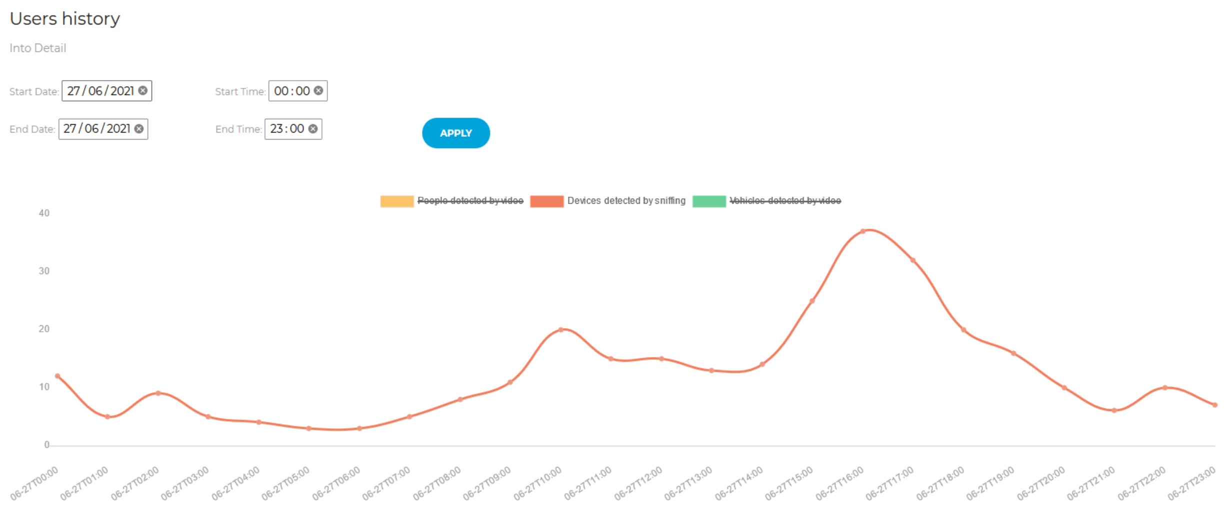


Figure 24: Devices detected by sniffing on the Smart Lamp Post in *Cais da Fonte Nova* between 28/06/2021 (00:00H) - 30/06/2021 (00:00H)

As it is possible to see in these figures above, there are certain patters that can be extrapolated, as the affluence of people near or passing by the post is relatively repetitive. At a first glance there are two major dips in the graph's values displayed which correspond to the night hours in both places, which further corroborates the good accuracy of the sniffing module.

Starting by analyzing the first figure (Figure 23), which corresponds to the values captured in the IT2 Smart Lamp Post, it is possible to see that around the 9:00 hours (in both day 28 and 29) a major flux of people is captured which corresponds to the hour that the workers of the IT2 start work. Afterwards the values go up and down in time as a parking lot is near and some students of the University pass through this Post and are then detected. Before and after around lunchtime, the count goes up, down and up which illustrates very clear when workers leave and go back to the IT2. In the afternoon, the values are more and less stabilized and then at around 18:00 there are some fluxes of people again as they leave the University building. After this hour, the values captured correspond to sports enthusiasts that pass in the nearby sidewalk jogging or bicycling.

Regarding the second graph (Figure 24), in *Cais da Fonte Nova*, there are some differences comparing with the first figure displayed. The first peak isn't at 9:00 as it is in the IT2's Smart Lamp Post because it isn't an area where workers pass regularly. Moreover, the peak is achieved during the afternoon as there are plenty of sports related fields near the Smart Lamp Post and cafes. During the hours after dinner (around 20:00), the values don't decrease as fast as in the other location because it is a place that many people like to walk during these night hours. It is expected that during the summer the values will increase dramatically both during the day and during the night, especially as it becomes more appealing to walk in the *Cais da Fonte Nova*.


Users history

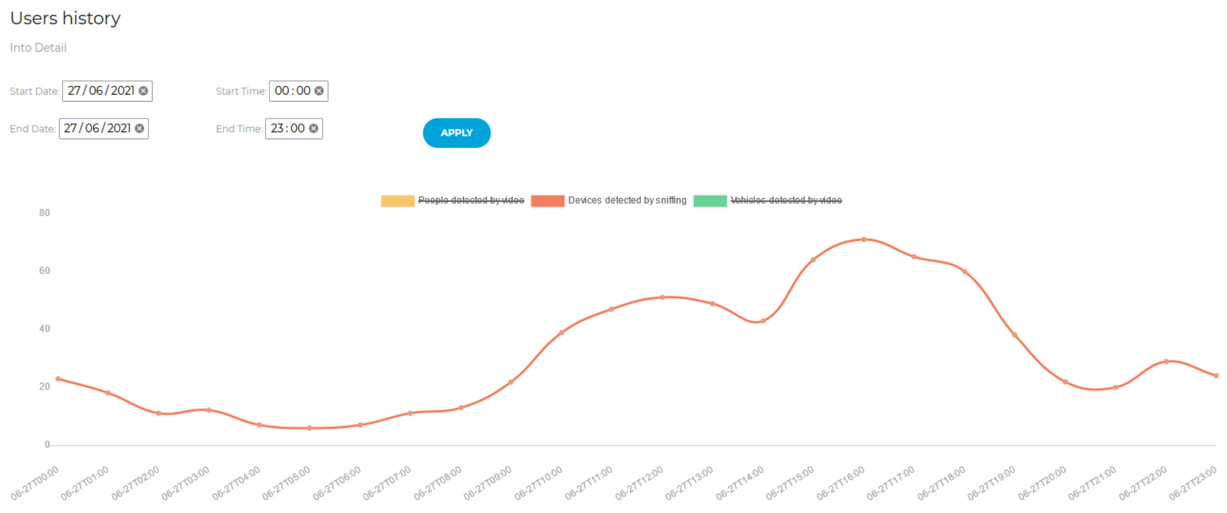
Into Detail

Start Date: 27/06/2021 Start Time: 00:00

End Date: 27/06/2021 End Time: 23:00

APPLY

Figure 25: Devices detected by sniffing on the Smart Lamp Post in IT2 during a Sunday, day 27


Users history

Into Detail

Start Date: 27/06/2021 Start Time: 00:00

End Date: 27/06/2021 End Time: 23:00

APPLY

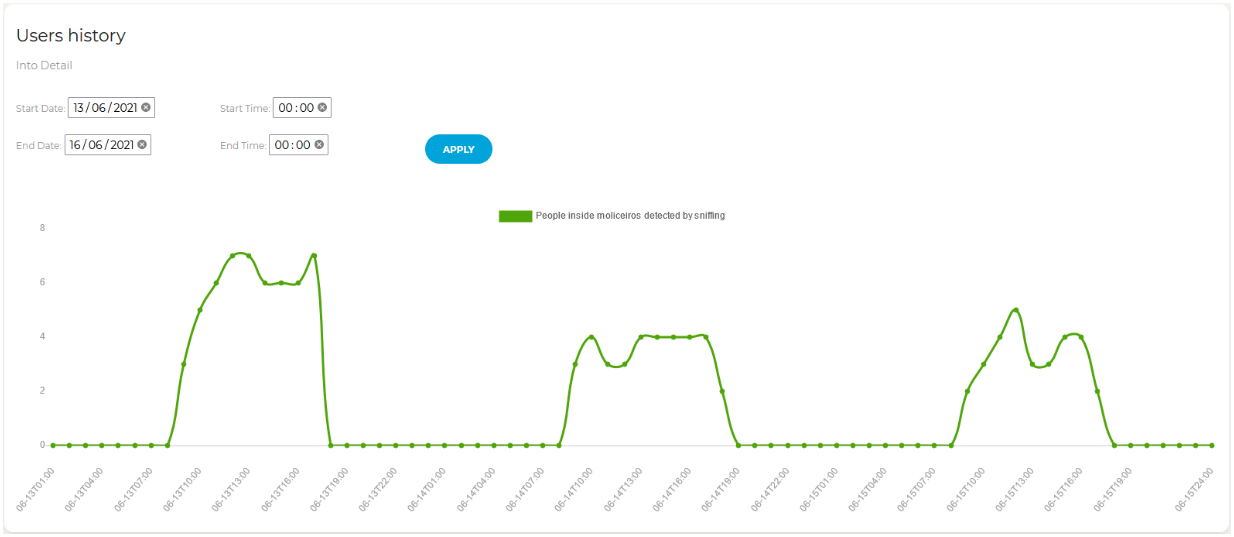
Figure 26: Devices detected by sniffing on the Smart Lamp Post in *Cais da Fonte Nova* during a Sunday, day 27

As a final remark, the figures 25 and 26 show the values captured during day 27 of June which was a Sunday. During this day, it is clear that there are some differences relative to the previous figures captured during weekdays. Firstly, there are less devices captured during the morning near the IT2 Smart Lamp Post due to the fact that the IT2 is closed during Sundays. The peak of devices is achieved during the afternoon at around 17:00 which can be explained that many people pass by the Smart Lamp Post practicing sports regularly during weekends. On the *Cais da Fonte Nova*, the values are consistently higher than the captured during weekdays. Moreover, it is achieved a higher peak during the morning but especially during the afternoon, due to the fact that it is usually an off day for workers and students in general and correlate to people spending their afternoon or morning in the park.

## 13.2 OBUs - On Board Units

An example of the results obtained from the Wi-Fi Sniffing inside *moliceiros*, using the OBUs, is shown in Figure 27. The graphic shows the average per hour of devices detected inside each *moliceiro*. It is possible to notice clearly the working hours of the *moliceiro* company, since there is a slight decrease in passengers during lunchtime, because the members of the crew go lunch alternately, meaning that only one *moliceiro* is working during more or less one to two hours, as well as a gradual reduction in passengers as the day comes to an end. On the same chart it is also visible that on the first day shown (which is a Sunday) the average number of people is clearly higher than the next two days (which are weekdays). By going in trips inside the *moliceiros*, the group was able to register that the number of devices detected were at most 1 or 2 off compared to the number of people actually inside the *moliceiros* during weekdays.

Another conclusion, that was possible to be inferred from the data acquired and from visual observation of the passengers, is that during weekends the number of devices detected (Figure 28) raises only by a small margin compared to weekdays. So, the difference on the weekends between real people inside the *moliceiros* and devices detected raises from one to two people to around six to seven people. The reason found, to justify this difference, is that during weekends a bigger percentage of older people and small children travel in *moliceiros* with their families. Older people have a bigger tendency to not have very advanced technological devices (without Wi-Fi or with it turned off), while small children usually do not own any devices. Which, therefore, justifies the difference compared to the weekdays when mostly middle-aged couples or small groups of people tend to be the main passengers.


Users history

Into Detail

Start Date: 13/06/2021 Start Time: 00:00

End Date: 16/06/2021 End Time: 00:00

Figure 27: Devices detected by sniffing on the OBUs aboard *moliceiros* in between 13/06/2021 (00:00H) - 16/06/2021 (00:00H)

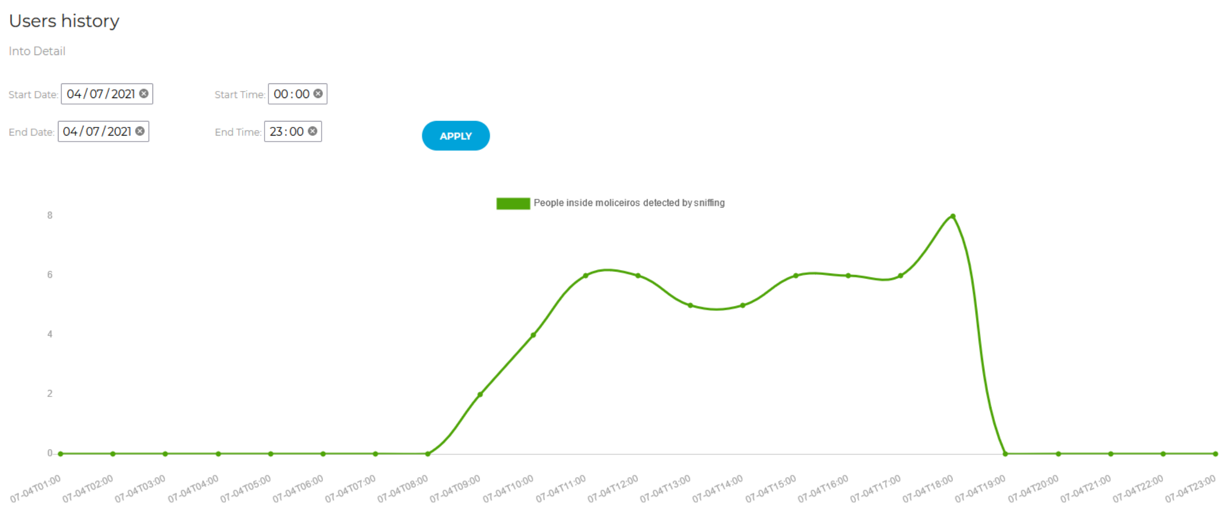


Figure 28: Devices detected by sniffing on the OBUs aboard *moliceiros* in between 04/07/2021 (00:00H) - 04/07/2021 (23:00H)

## 13.3 Image Analysis and Object Detection

This module captured data in both of the areas previously referred. The module ran continuously during approximately two months, and by analyzing the captured information, several movement patterns could be recognized.

In the IT2 location (Figure 29), it is noticeable that the amount of people detected starts rising at 8:00, which is when the work day and university classes normally start. The hours with the most traffic of people are between 12:00 and 14:00 and between 15:00 and 17:00. The first time interval coincides with what usually is the lunch period, and the second time interval could be explained by people leaving classes. Lastly, after 18:00 the traffic of people reduces by the hour until it reaches very small values. This coincides with the end of classes and of the work day.

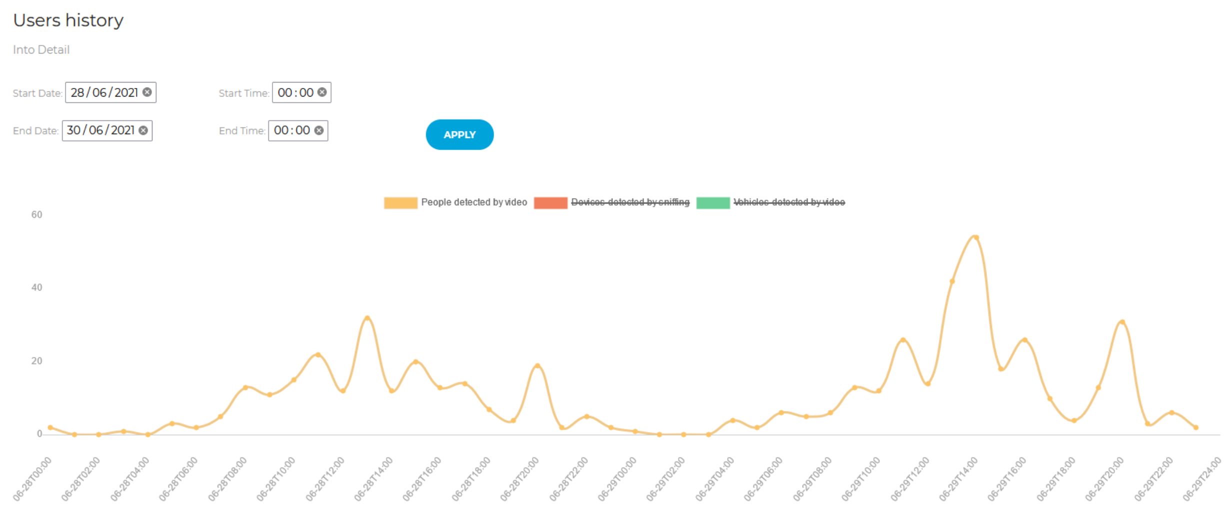


Figure 29: People detected by image analysis in IT2 between 28/06/2021 (00:00H) - 30/06/2021 (00:00H)

In the location of *Cais da Fonte Nova*, the peaks of movement would usually occur in the middle of the afternoon, which makes sense due to the fact that this is an area usually frequented for leisure activities.

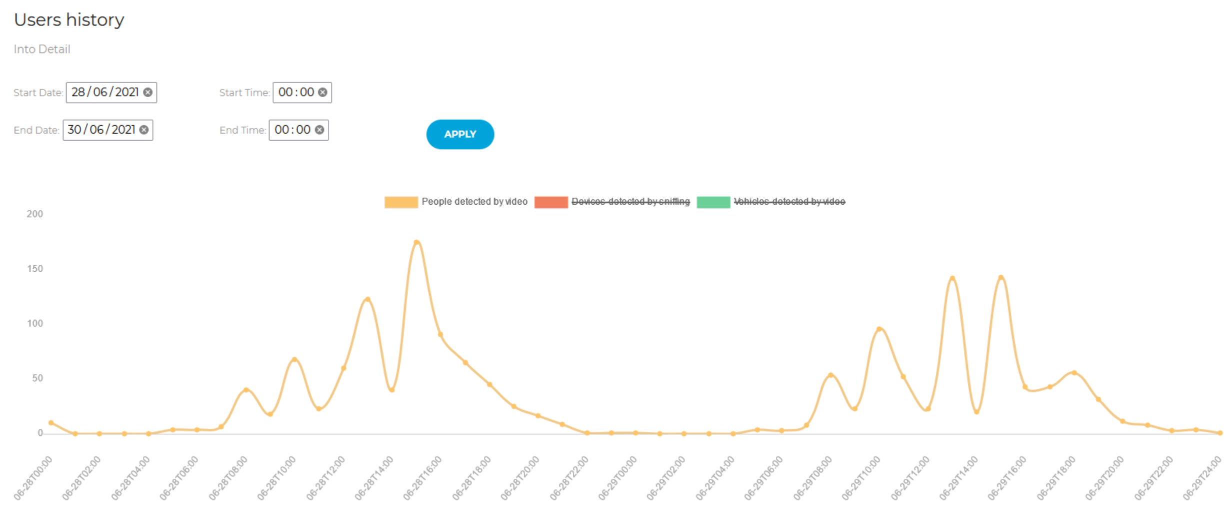
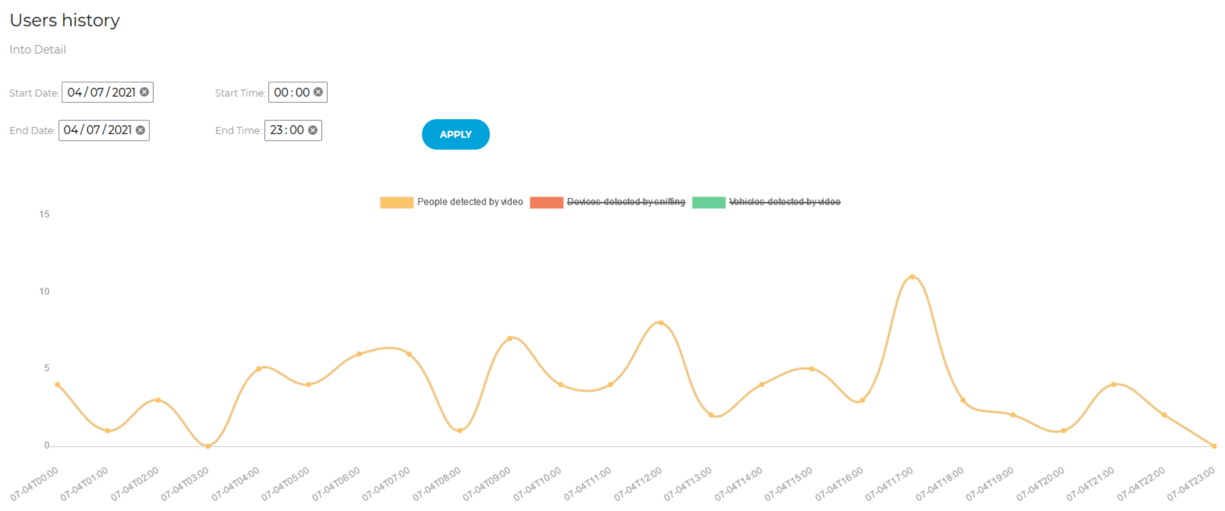


Figure 30: People detected by image analysis in *Cais da Fonte Nova* between 28/06/2021 (00:00H) - 30/06/2021 (00:00H)

All of these results were captured during weekdays, but it was necessary to evaluate the results captured in weekends as well.

In IT2, the traffic of people during weekends was small, due to the fact that being an area of a university campus, there is not a high movement of people. In Figure 31 the data from a Sunday is shown, allowing to verify that the peak of detected people corresponds to approximately 10 people from 17:00 to 18:00. This value can be compared with the peak of nearly 60 people detected in Figure 29. This shows that between weekdays and weekends, this area is more frequented in weekdays, mainly due to university activities.


Users history

Into Detail

Start Date: 04/07/2021

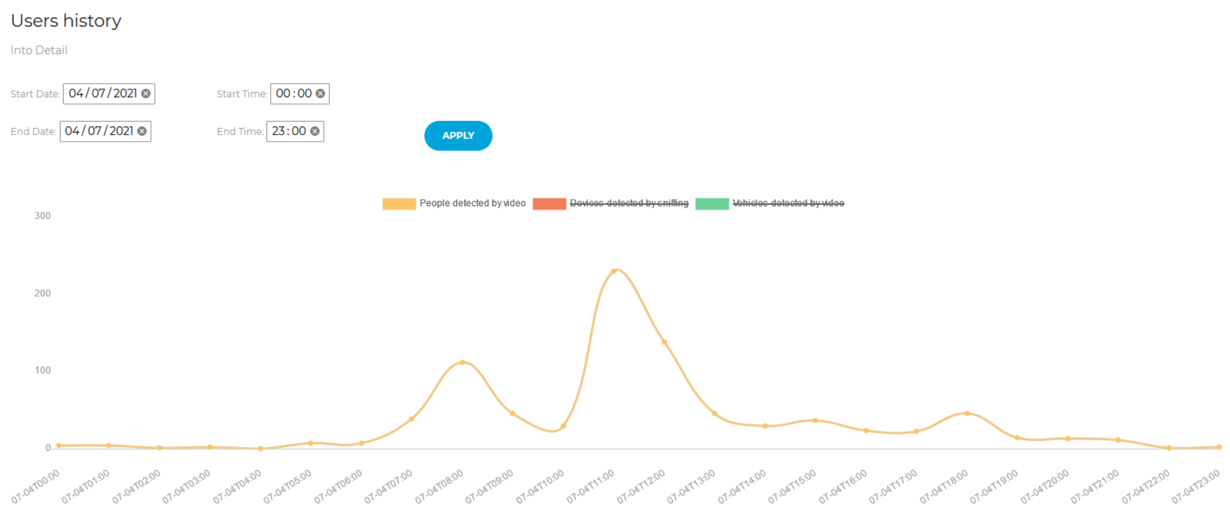
Start Time: 00:00

End Date: 04/07/2021

End Time: 23:00

Figure 31: People detected by image analysis in IT2 between 04/07/2021 (00:00H) - 05/07/2021 (00:00H)

Lastly, in *Cais da Fonte Nova*, the amount of people that are detected on the weekends pass the number of people detected during weekdays. Again, this is explained with the fact that this location is highly frequented for leisure activities. This is evident in Figure 32, where more than 200 people are detected during the peak hour, whereas in Figure 30, which is in a weekday, the peak corresponded to a value between 150 and 170 people.


Users history

Into Detail

Start Date: 04/07/2021

Start Time: 00:00

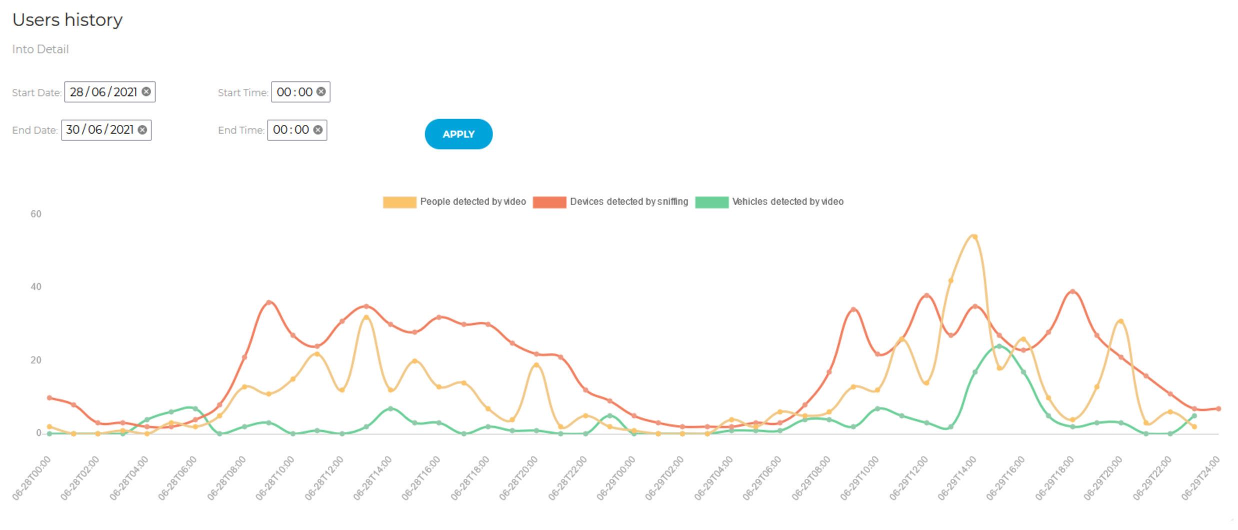
End Date: 04/07/2021

End Time: 23:00

Figure 32: People detected by image analysis in *Cais da Fonte Nova* between 04/07/2021 (00:00H) - 05/07/2021 (00:00H)

## 13.4 Comparison between modules

Comparing the results obtained through detection and sniffing, it is noticeable that both are usually a bit different, because both modules have different advantages and disadvantages. While the Wi-Fi Sniffing has a 360 degrees capture range, not everyone has Wi-Fi turned on or is carrying a device with them, or may be carrying multiple, thus making the numbers obtained not real. On the other hand, Object Detection has a very high accuracy, there is no way for a person to turn off the detection as they may do to the Wi-Fi, however the camera angle is limited and cannot catch all 360 degrees making it so that there are always dead angles where people, vehicles and other objects may be passing by and are not being counted. However, it is still important to compare results obtained from both methods and reach conclusions about patterns and comparisons in between them. Below, two charts of weekdays (Figures 33 and 34) are shown, one for each location where we captured data.

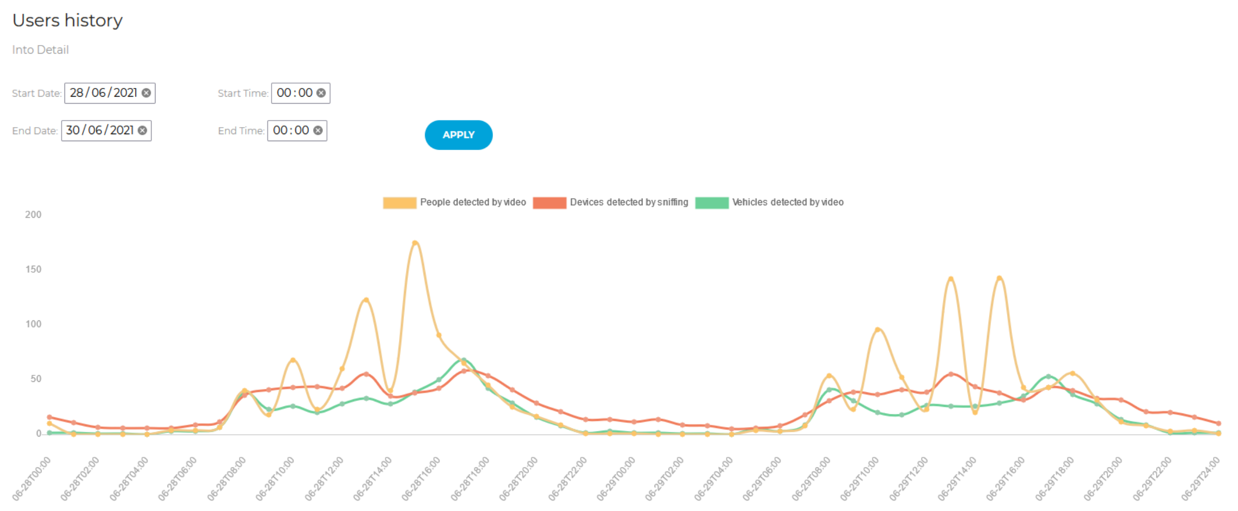

Users history

Into Detail

Start Date: 28/06/2021 Start Time: 00:00

End Date: 30/06/2021 End Time: 00:00

Figure 33: Data acquired on the smart lamp post in IT2 between 28/06/2021 (00:00H) - 30/06/2021 (00:00H)


Users history

Into Detail

Start Date: 28/06/2021 Start Time: 00:00

End Date: 30/06/2021 End Time: 00:00

Figure 34: Data acquired on the smart lamp post in *Cais da Fonte Nova* between 04/07/2021 (00:00H) - 30/06/2021 (00:00H)

Each chart refers to a 48-hour period on the same days. In both cases, there is a noticeable difference between day hours and night hours as well as clear peaks at around 9:00, 13:00 and 17:00 corresponding respectively to the usual work starting hour, lunch hour and work exiting hour.

Meanwhile, Figures 35 and 36 are two charts of data acquired during a Sunday, one for each location. As we have seen before, the traffic of people on weekends is lower in IT2 and higher in *Cais da Fonte Nova* comparatively to the weekdays.

The results show that on the weekends the amount of vehicles that are detected in *Cais da Fonte Nova* is lower, which can be explained by the fact that in this area people might prefer to walk than to commute on a vehicle, which is more frequent in work days.

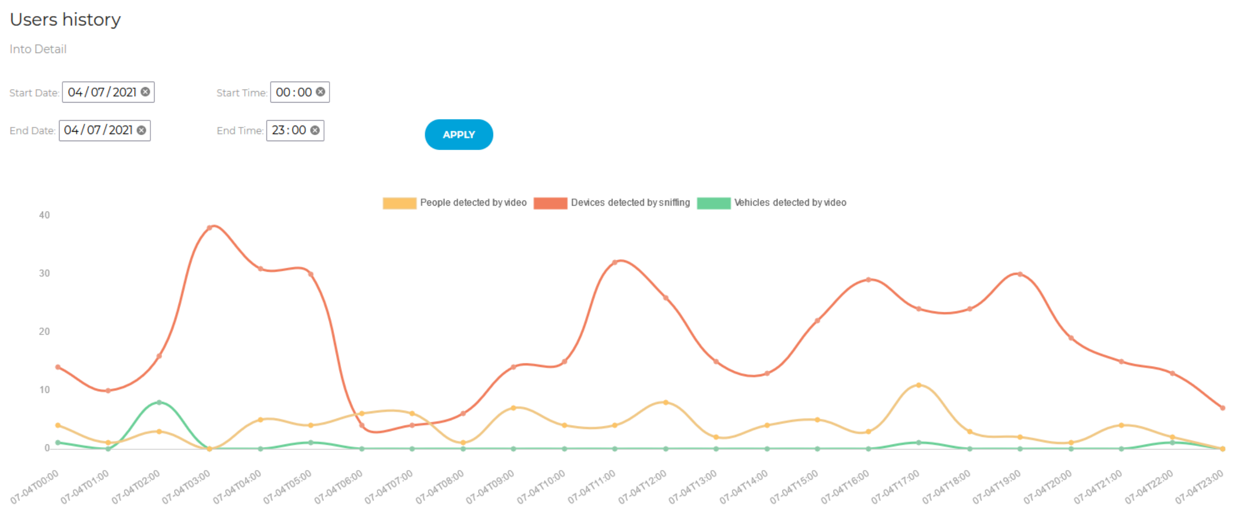


Figure 35: Data acquired on the smart lamp post in IT2 between 04/07/2021 (00:00H) - 04/07/2021 (23:00H)

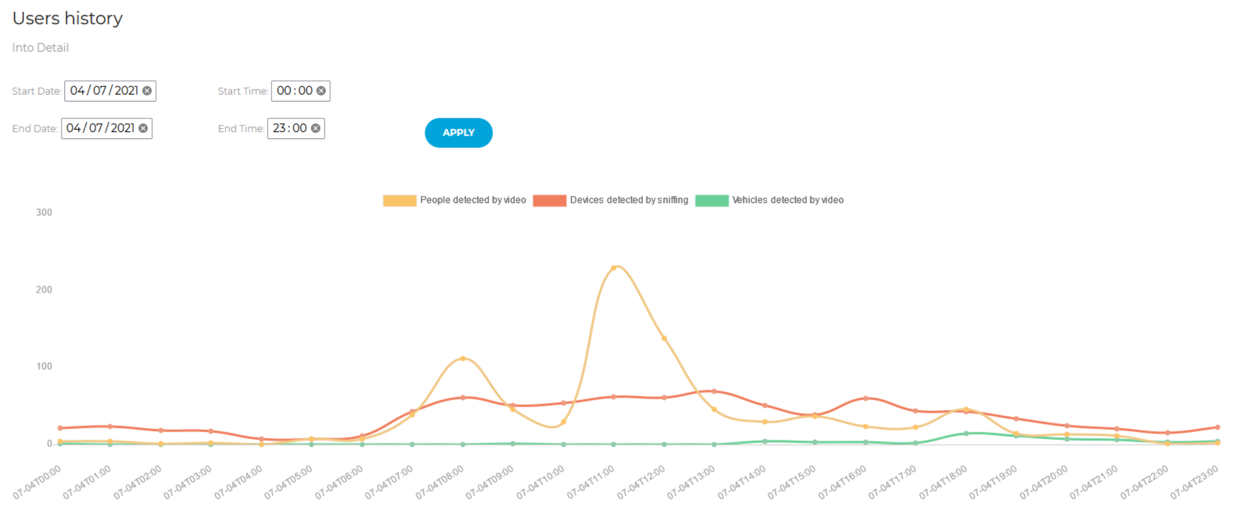


Figure 36: Data acquired on the smart lamp post in Cais da Fonte Nova between 04/07/2021 (00:00H) - 04/07/2021 (23:00H)

It is import to refer that calibration methods were applied to all the data acquired, and this is visible in Figure 33. Usually the values from sniffing and image detection are very similar with the exception of some peaks that may happen in only one of the modules in accordance to the reasons presented before.

## 13.5 Web App

Regarding the Web Application, as said before, our main goal was to give the user a platform in which they could have access to information about the fluctuation of people and *moliceiros* of some places in *Aveiro* city. Therefore, as it is possible to verify in the images (Figure 37) below a simple, intuitive and user-friendly interface was created. This Web Application not only displays live data, but also historical values.

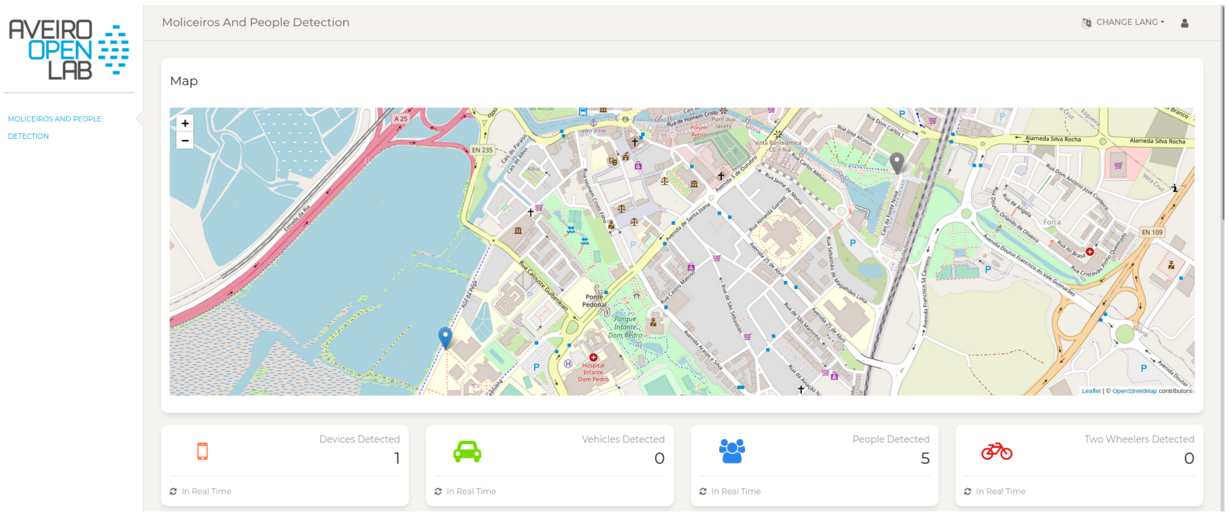
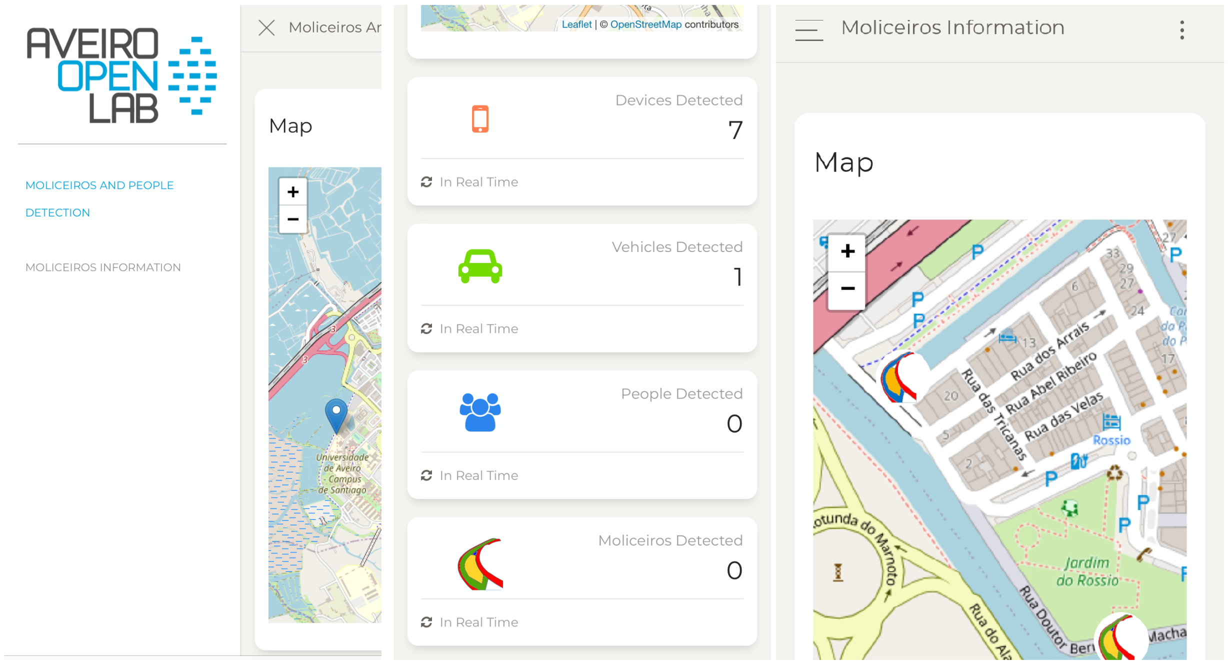


Figure 37: First screen when opening the web application, showing the smart lamp post selection map and the real-time detection cards for the IT2 chosen location

One of the goals established, was that the web application had to be mobile responsive, which can be seen in the figure down below (Figure 38). In the middle picture, in Figure 38, is being displayed the main page, "*Moliceiros* and People Detection", where the data related to the Post 22, located on *Cais da Fonte Nova*, is shown. In the final picture there's the page related to the *moliceiros*, "*Moliceiros* Information", where the data about the number of devices detected for each *moliceiro* is displayed along its location, which is shown in the map.


MOLICEIROS AND PEOPLE DETECTION
MOLICEIROS INFORMATION
Moliceiros Ar
Map
Leaflet | © OpenStreetMap contributors
Moliceiros Information
Map

Devices Detected 7
In Real Time

Vehicles Detected 1
In Real Time

People Detected 0
In Real Time

Moliceiros Detected 0
In Real Time

Figure 38: Web Application seen through a mobile device

Considering all the goals established in the beginning of the project, it can be concluded that all of them were achieved successfully.

# 14 Conclusion

This project is organized in three main modules. The first allows the detection of people, vehicles and *moliceiros*, through the analysis of video streams and the use of object detections modules. The second module captures packets sent from devices involved in Wi-Fi communications to estimate the number of people in a certain area. The data obtained from these two modules is compared to establish patterns of movement of people and vehicles near the Smart Lamp Posts spread throughout the city. The third module is the web application that displays these statistics and the data collected in real-time so that it can be observed by the public.

With the development of this project, it was possible to detect certain patterns of movement. During the weekdays, in the university campus, the peaks of people fluxes occur at between 08:00 and 09:00 and between 12:00 and 14:00, respectively coinciding with the start of classes and lunch period. At 17:00 it is noticeable another peak as it coincides with the end of classes and of the work day. After 18:00, the amount of people detected gradually reduces as the day comes to an end. In *Cais da Fonte Nova*, the peak is usually registered in the middle of the afternoon, which can be explained by the fact that this area is more frequented for leisure activities.

On the weekends, the amount of people detected in IT2 decreases and in *Cais da Fonte Nova* it increases. These observations can be explained by the fact that university activities stop on the weekends and areas more dedicated to leisure acitivites are more frequented by the public during weekends.

A script to detect the *moliceiros* that cross the Ria of Aveiro was also developed, making this an important contribution to the study and analysis of the movements and frequency of these boats. With the capture of Wi-Fi packets emitted by Wi-Fi capable devices, like smartphones, it was possible to estimate how many people were aboard each *moliceiro*. This information proves to be very useful to Aveiro's Town Hall, since it allows them to draw conclusions related to the traffic of tourists in the city and their adherence to the *moliceiros* travel throughout different days and seasons of the year. One of the main conclusions brought by this module was that the *molceiros* were more active during the weekends, transporting more tourists.

In order to achieve all modules' goals, some challenges had to be overcome. The biggest challenges can be resumed to three main problems.

It was essential to find a detection mechanism that was adequate, which means that, it had to run on the hardware, achieve a high threshold of certainty of the object detected and have a high processing number of frames per second. One of the main problems that were solved in this module was the loss of video frames during the processing phase. This was solved by verifying clock discrepancies in the hardware in order to lose as minimal amount of frames as possible every second and therefore valuable data. Another problem that had to be solved was the fact that in analyzing the video stream in real time, several objects that were not there were being detected. This had to be fixed by filtering noise in the images and removing unwanted areas that generated false positives.

The calibration of both modules was necessary to obtain more accurate results. Regarding the detection module, an object (person, vehicle, etc.) was being detected and counted more than once when moving. This problem led to the capture of inaccurate values, that were much higher than what would be expected. To fix this issue, the module was calibrated by counting the number of objects per video frames. With this solution, the analysis of each frame would involve counting how many objects were detected in it, and then after analyzing a batch of frames an average would be calculated. After this calibration, the captured data mas more accurate and made more sense regarding what would be observed in real life. The sniffing module was also calibrated. A very superior number of devices was being detected compared to the number of people detected by video, and this implied that the values were not being captured accurately. To solve this problem, the time to live of the captured MAC address had to be reduced, and a valid time interval was found based on trial and error and on observation of the video stream of the corresponding area. By adjusting the amount of time that the MAC addresses was saved, these were discarded more often and the values adjusted to the real life situation, providing more accurate results.

Finally, the OBUs calibration and problem-solving was also one of the main challenges. Any problem with the OBUs was usually specially hard to solve as it would be noticed only when it was working and therefore the OBUs would be inside the *moliceiros* moving around, with no connection to the ATCLL network. So, in order to remotely access it, it was necessary to time the approach of the *moliceiros* with the next station they would pass by and access the OBUs through a direct connection with that station and then solve any issues before they moved out of the range of the station. While it was possible to solve these issues by bringing the OBUs back to the IT2 it was not practical as the OBUs had to be in *moliceiros* to test if the problem was solved.

For future work, it would be interesting to allow the execution of the *moliceiros* detection script and the people and vehicle script at the same time, to be able to obtain different types of data in the same time periods. It would be also important to improve the Wi-Fi Sniffing module, in order to obtain even more accurate values through the filter of unwanted MAC addresses and the optimization of the capture of probe requests.

# References

[1] João Ribeiro, André Zúquete, and Susana Sargento. Survey of Public Transport Routes using Wi-Fi:. In *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems*, pages 168–177, Funchal, Madeira, Portugal, 2018. SCITEPRESS - Science and Technology Publications.

[2] Yan Li, Johan Barthelemy, Shuai Sun, Pascal Perez, and Bill Moran. A Case Study of WiFi Sniffing Performance Evaluation. *IEEE Access*, 8:129224–129235, 2020.

[3] Anas Basalamah. Crowd Mobility Analysis using WiFi Sniffers. *International Journal of Advanced Computer Science and Applications*, 7(12), 2016.

[4] Lai Tu, Shuai Wang, Desheng Zhang, Fan Zhang, and Tian He. ViFi-MobiScanner: Observe Human Mobility via Vehicular Internet Service. *IEEE Transactions on Intelligent Transportation Systems*, 22(1):280–292, January 2021.