Universidade de Aveiro

Real Time Department Occupancy Tracker

João Fernandes (93460), Tiago Pedrosa (93389)

Projeto de Redes e Sistemas Autónomos

Departamento de Eletrónica, Telecomunicações e Informática

June 20, 2023

Contents

List of Figures

Abbreviations

B.A.T.M.A.N. - Better Approach to Mobile Ad-hoc Networking
MQTT - Message Queuing Telemetry Transport
Yolo v8 - You Only Look Once version 8

Chapter 1

Introduction

This report aims to describe the process and the steps taken during the development of the project within the scope of the subject of Networks and Autonomous Systems (Redes e Sistemas Autónomos/RSA).

The code developed fro this project can be found in this github repository: https://github.com/GilFernandes2000/real-time-department-occupation-tracker.git

1.1 Objectives

The main goal of this project is to develop a network that provides real-time information on the occupancy of university departments and buildings accessible to students. By monitoring occupancy, the university can make informed decisions regarding changes, upgrades, or expansions. The project offers several potential expansions and advantages:

- Enhanced Student Experience: With access to real-time occupancy data, students can make informed decisions about which departments or buildings to visit based on current availability. This information can help them plan their schedules, avoid overcrowded areas, and find suitable study spaces or meeting rooms more easily. As a result, students' experience and productivity on campus can be significantly enhanced.
- Safety and Security: Real-time occupancy monitoring can also play a crucial role in ensuring the safety and security of individuals on campus. By tracking occupancy levels, the university can effectively manage emergency situations, such as evacuations or managing crowd flow during events.
- Data-Driven Decision Making: The network can generate valuable data insights that the university can leverage for strategic planning and decision making. By analyzing historical occupancy patterns and trends, the university can gain insights into long-term occupancy fluctuations, peak usage hours, or seasonal variations. These insights can guide decision making related to infrastructure upgrades, facility planning, and policy adjustments.

1.2 Technologies

For the development of the network, an Ad-hoc network with *B.A.T.M.A.N.* protocol was used, integrated with *MQTT*. This allows for an easy way to make small and reliable networks that can be deployed in the various departments.

The Programming language used to integrate the previous protocols was *Python*.

To create a Dashboard for presenting the information on Department occupancy, a web app was developed with interactive map support using *Javascript* and *Leaflet* libraries, with *HTML*, *CSS* and *Flask* as the base.

We employed the *Jetson Nano* as the primary hardware for this task. This powerful yet compact device runs the necessary scripts, creates the network infrastructure, transmits messages, and is responsible for displaying a comprehensive dashboard.

1.2.1 Jetson Nano 2GB

The **Jetson Nano** is a compact yet potent computer designed specifically to propel entry-level edge AI applications and devices. Its performance-optimized capabilities make it ideal for demanding computations, including image processing and machine learning tasks. This is suitable for our project as the **Jetson Nano** will receive images from a camera and perform the image processing algorithms on top of the frames received giving a good and reliable output. Additionally, the **Jetson Nano** offers a high degree of connectivity, making it suitable for creating a network and sending data across that network. Its high processing power and efficient design also make it capable of handling multiple tasks simultaneously, such as running various scripts and displaying a real-time dashboard.

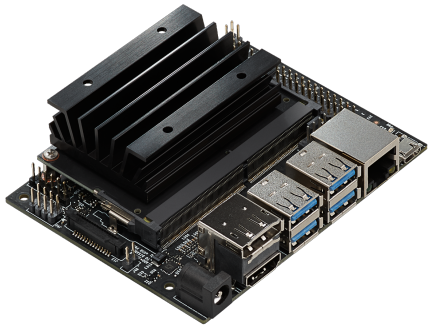

Figure 1.1: *Jetson Nano*

1.2.2 YOLOv8

YOLOv8 (You Only Look Once version 8) is a real-time object detection algorithm developed by Ultralytics. The notable feature of **YOLOv8** useful for the project is its ability to detect objects at multiple scales and aspect ratios, making it suitable for detecting objects of various sizes. It also supports real-time object detection on both CPU and GPU, enabling

efficient deployment on different platforms. It predicts bounding boxes around entities as well at the probability of these entities being of a certain class of image inputs. Classes that **YOLOv8** can identify are People, Vehicles, Animals, some common objects like phones, chairs and laptops, sports related items like basketballs, footballs and food like fruits, vegetables and dishes. For the purpose of the project however, as we are only looking into the number of people that enter and leave buildings, there isn't a real need for other classes, *Person* the only class seemingly relevant.

Figure 1.2: *YOLO v8 Logo*

1.2.3 B.A.T.M.A.N

B.A.T.M.A.N. is a routing protocol for Wireless Ad-hoc Mesh Networks. It is a proactive protocol, meaning it maintains information about the existence of all nodes in the mesh that are accessible via single-hop or multi-hop communication links. This makes it suitable for dynamic and mobile network environments.

B.A.T.M.A.N. determines one single-hop neighbor for each destination in the mesh, which can be utilized as best gateway to communicate with the desired destination node. There is no need to calculate the complete route because every node acts as a router and makes independent decisions on data packet forwarding and learns about the best next-hop for each destination, which makes it a very fast and efficient implementation.

Figure 1.3: *B.A.T.M.A.N Logo*

1.2.4 Mosquitto MQTT

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed for efficient communication between devices in constrained environments, such as low-bandwidth networks or resource-constrained devices. It follows a publish-subscribe model, enabling devices to publish messages and subscribe to topics of interest.

In **MQTT**, there are three key components:

- **Publishers** are the devices that send information to certain specific topics.

- **Subscribers** are devices that receive messages by subscribing to specific topics.
- **Brokers** act as middleware, receiving messages from publishers and forwarding them to subscribers based on their topic subscriptions.

When a device publishes information to a topic, it is sent to the **MQTT** broker that takes a look at every device that subscribes to said topic and forwards the information. **MQTT** offers many possibilities in term of communication patterns fit for many scenarios. For this project the many-to-one, where multiple devices serve as publishers send messages to a single subscriber was decided to be the ideal setup, as this acts more akin to a monitoring setting like surveying a house, where many sensors around a house can send an alarm signal to a single noise-emitting machine in case of entry. In this case, many cameras will send information to the same dashboard.

Figure 1.4: *Modquitto MQTT logo*

1.3 Methodology

First things first, there was a need to study and understand the technologies for the task at hand. Starting with *YOLOv8*, tests were made with computer cameras and by applying the algorithm to videos in order to be conscious of the way it detects people and adapt the algorithm for the desired outcome. A process of understanding when to count that a person has entered or left was developed with the lessening of algorithmic mistakes in mind.

Tests were conducted to evaluate the connectivity between three *Jetsons* using *B.A.T.M.A.N.* and the results were successfully established.

Next, a discussion was initiated regarding the necessary information to be transmitted from one node to another within the network, aiming to ensure accurate and coherent data that can eventually be sent to a dashboard. Different configurations of *MQTT* were considered, and after careful consideration, the Many-to-One communication pattern was selected and implemented. *Mosquitto MQTT* was installed on the *Jetsons* to facilitate the data transfer in the ad-hoc network.

Finally, a dashboard was developed to provide users with access to the information gathered in the previous steps, such as the number of people in a building and device location. This dashboard is hosted on a *Jetson* device.

Chapter 2

System

2.1 System Architecture

2.1.1 Software Architecture

The architecture of the system software is comprised of three distinct modules. The first, known as the **Jetson Worker**, is responsible for identifying individuals as they enter or exit a building. The second component is a **message broker**, which manages the exchange of messages between various devices. Lastly, the **Jetson Master** acts as a receiver for these messages, processes them and displays the results on a dashboard.

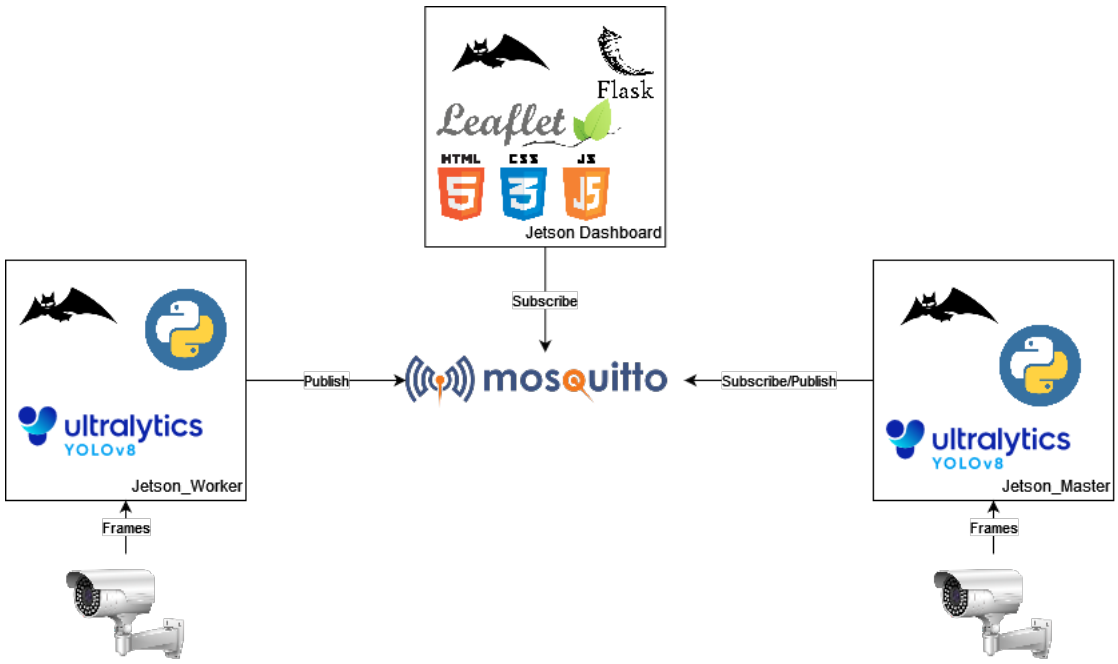

Figure 2.1: *System Software Architecture*

Each Jetson is assigned one of three roles:

- **Jetson Worker** They consistently monitor the flow of individuals entering and exiting, thereby maintaining an accurate record of personnel movements. This information is then periodically transmitted to the Master, ensuring real-time updates and efficient data management.

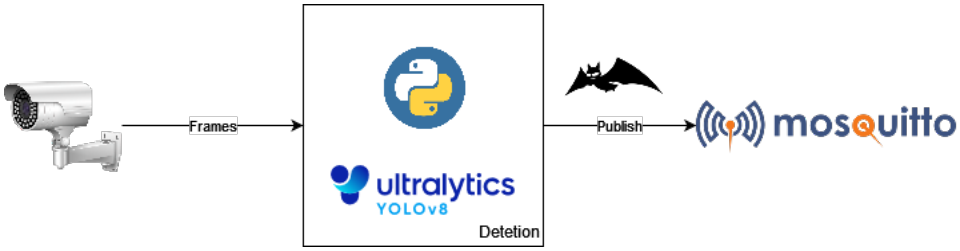

Figure 2.2: *Jetson Worker Architecture*

- **Jetson Master** In addition to performing standard tracking like a regular Jetson, it also gathers information from other departmental Jetsons. This facilitates the creation of a reliable department-wide counter that can be efficiently relayed to the dashboard.

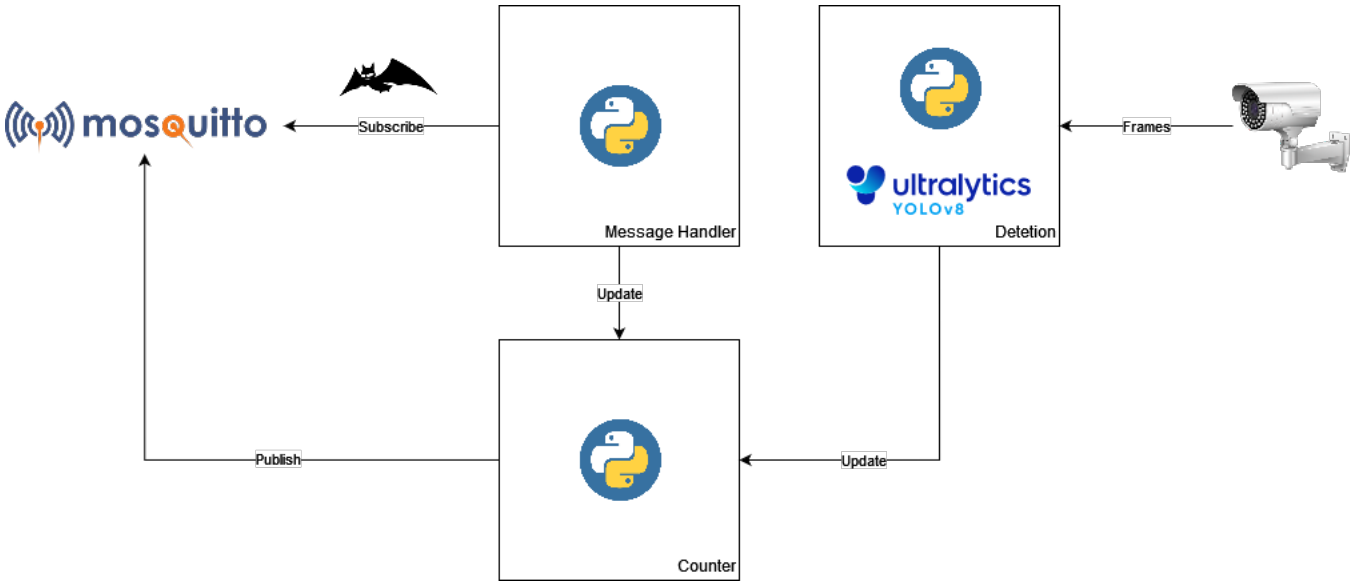

Figure 2.3: *Jetson Master Architecture*

- **Dashboard Jetson** There is only one Dashboard *Jetson*, which serves as the host for the dashboard itself and communicates with all other Master *Jetsons* via Wi-Fi. In future projects, it is recommended to separate it from any specific department and make it an external device.

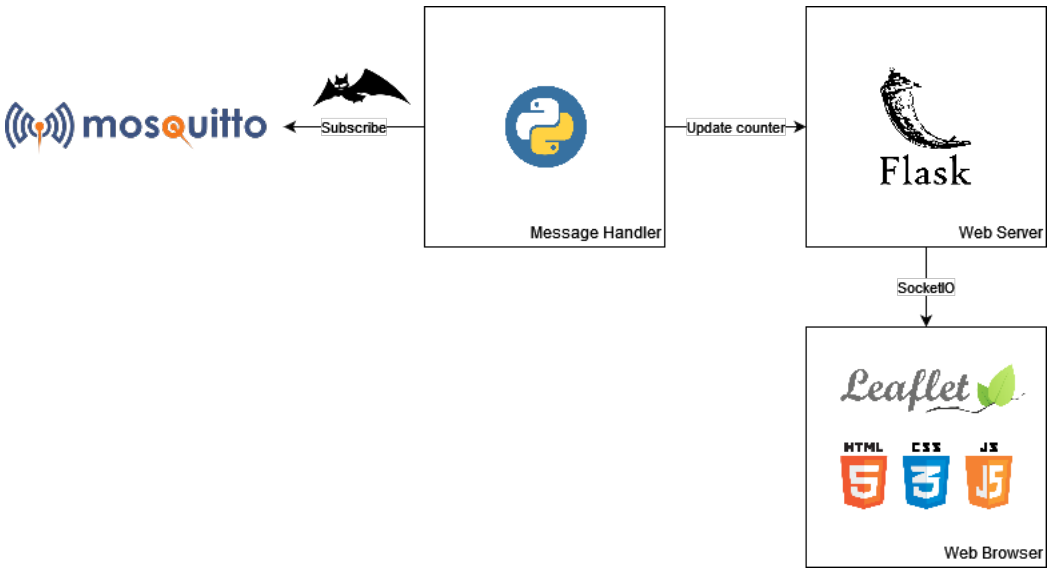

Figure 2.4: *Dashboard Jetson Architecture*

2.1.2 Hardware Architecture

The hardware architecture of the system involves linking Jetsons within a Department via B.A.T.M.A.N, and using WIFI to connect the Jetson that collects the counts from each department's master. The cameras will be physically wired to the Jetson Workers to enable them to receive the frames captured by the Jetson.

Each Department consists of a camera, which is physically connected to a Jetson Nano for every entrance/exit of a building. This is done to ensure that no individual entering or leaving the building is overlooked, thereby maintaining the reliability of the counting system.

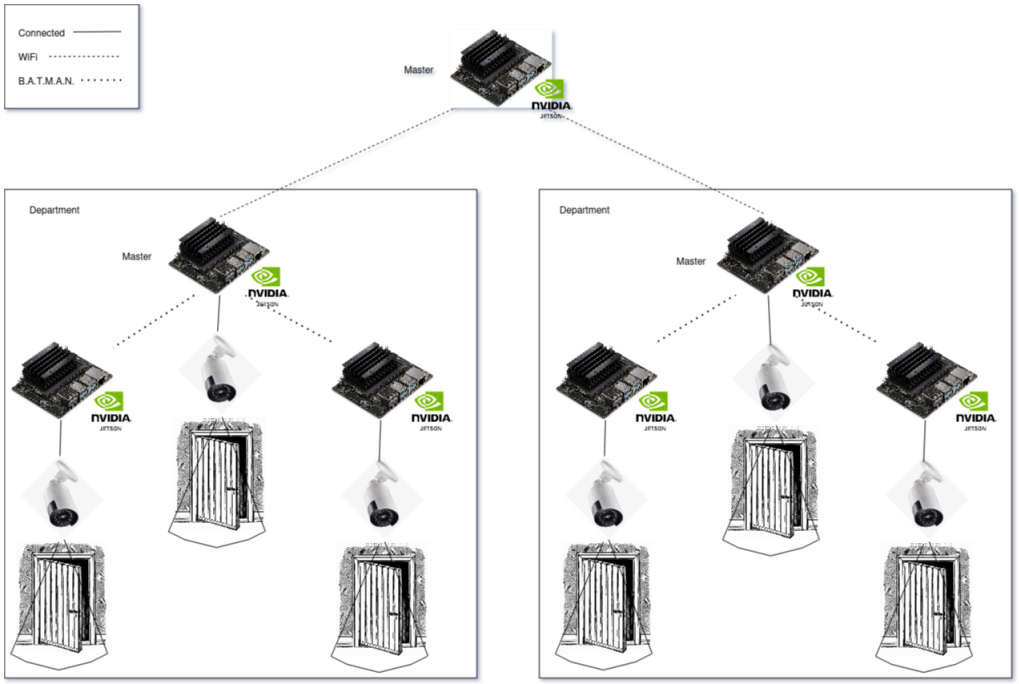

Figure 2.5: *System Hardware Architecture*

2.2 Network generation scripts

For the generation of the *MQTT* over *B.A.T.M.A.N.* networks, a group of *Python* scripts were created and inserted into the *Jetsons*:

- **create_batman_interface.sh** - It is used to create a B.A.T.M.A.N. interface and connect a *Jetson* to the network. This script was adapted from the one of the same name already inside the device so that every node would be in the same channel, same frequency and accessing the network nicknamed "Atlas".

- **Jetson_Worker.py** - This is the implementation of the people counting and tracking system using YOLOv8 object detection algorithm and the *MQTT* protocol. The system utilizes *YOLOv8* to detect and track people in a video stream or camera and establishes an *MQTT* connection with a master server, hosted on the master *Jetson*. Has two versions, the Jetson_Worker_Down_Up.py where the line is placed horizontally and a Jetson_Worker_Left_Right.py

- **Jetson_Master.py** - Also implements the people counting and tracking system using YOLOv8. However when it comes to the MQTT connection, it established connection with on or more worker Jetsons, and serves as a central communication hub for the department's information.

- **Jetson_Dashboard.py** - It collects the counter data from all departments and effectively visualizes them on the map. Each department's counter is displayed in their

corresponding area, providing a comprehensive, geographical view of the entire organization's metrics. In this implementation the Dashboard only has connection to one department.

Chapter 3

Deployment

For the deployment of the project, the proposed approach is to install a combination of *Jetson* devices and cameras at the entryways of each department, as depicted in Figure 3.1(a), taking DETI as an example. In the future, all departments would have their own dedicated *B.A.T.M.A.N.* network and then would be connected through Wifi to a more centralized *Jetson* as shown in Figure 3.2(b).

In the initial implementation, a single *Jetson* device was installed at DETI, as illustrated in Figure 3.2. Due to the unavailability of multiple cameras with sufficiently high quality to ensure reliable usage of the *YOLOv8* model, only one *Jetson* device was utilized to capture entries and exits. These devices would then communicate with a Master *Jetson*, which would consolidate the information and transmit it to the Dashboard.

To summarize, the deployment plan involves placing *Jetson* devices and cameras at department entryways, with one *Jetson* serving as the primary device for capturing data. The collected information is then transmitted to a central Master *Jetson*, which acts as a hub for consolidating the data and transmitting it to the Dashboard for visualization and analysis.

(a) *DETI* **(b)** *University*

Figure 3.1: *Jetson Distribution*

3.1 Dashboard

The Dashboard has a straightforward design and its main purpose is to provide a quick overview of the current occupancy in a building. As individuals are detected entering or leaving the building, the Dashboard updates the displayed information at the top of the screen under the label "Estimated amount of people inside the building." This data is received from the Master *Jetson*, which consolidates the information from the department-specific *Jetson* devices.

In addition to the occupancy count, the Dashboard also includes a map that represents the layout of the university buildings. Each *Jetson* device installed in a department will be marked on the map in the Dashboard. This feature enables easy tracking of the locations where *Jetson* devices are deployed across the university campus.

Furthermore, in addition to the general occupancy count displayed at the top of the Dashboard, there will be a specific label attached to the *Jetson* Master of each department. This label will show the current occupation of that particular department. By offering department-specific occupancy information, users can easely determine the occupancy levels of individual departments without having to navigate through multiple screens or sections of the Dashboard.

Estimated number of people inside the building: 1

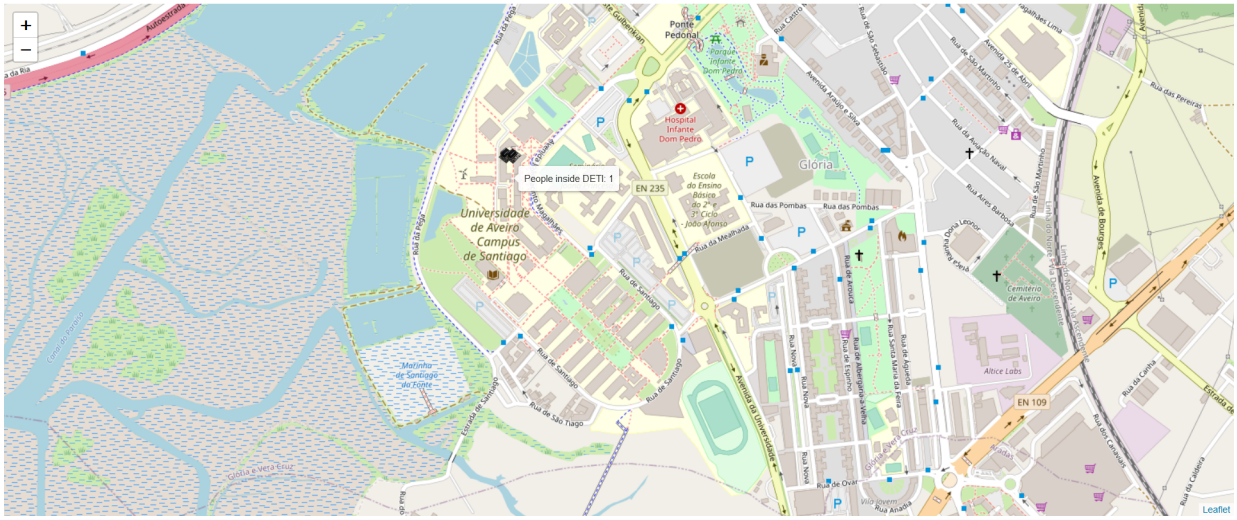

Figure 3.2: *Dashboard and terminal*

3.2 Tracking Process

The process of tracking using the camera and *YOLOv8* involves a virtual line and centroids, which are points calculated to be in the middle of the detected person. When a person is detected, a bounding box is created around them, and they are assigned an ID. Additionally, a centroid is generated at the center of the box. This centroid is used to determine if the person crosses the virtual line. The virtual line, visible in the camera's feed, serves as the detection point for entering or leaving. When a person's centroid passes through the line, it is counted as either entering or leaving, based on the direction they were walking. This direction information is easily accessible through the ID. The person's direction is calculated each time they are detected by *YOLOv8*.To balance precision and efficiency, the number of frames used to determine a person's direction can be adjusted. Using more frames can provide higher tracking precision, but it may require additional processing power and time. Adjusting this parameter can be useful, especially when working with units that have limited processing capabilities.

For versatility there are two ways of monitoring the entrances:

- **Parallel Configuration** (Figure a): The camera is positioned parallel to the entrance, capturing the movement of individuals along the entrance path. A virtual line is placed perpendicular to the camera's field of view, facilitating the detection of individuals crossing the line when entering or exiting.
- **Perpendicular Configuration** (Figure b): The camera is positioned perpendicular to the entrance, capturing the movement of individuals as they approach or walk away from the entrance. A virtual line is positioned parallel to the camera's field of view, allowing precise detection of individuals who cross the line while passing through the entrance.

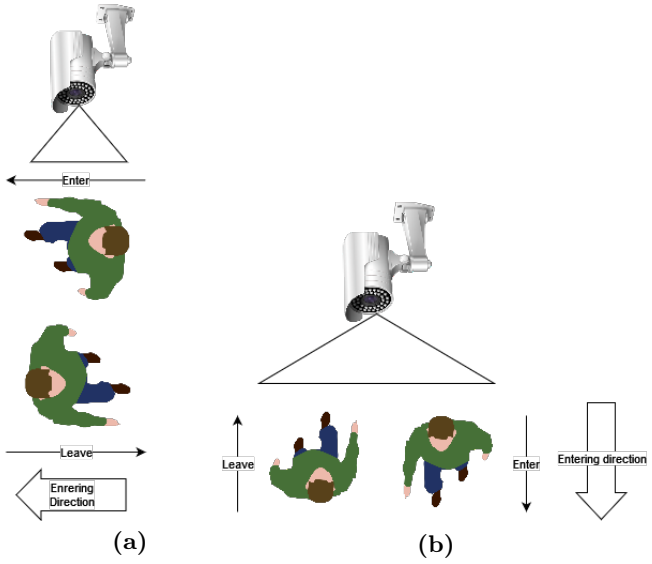

Figure 3.3: *Types of detection*

3.3 Pseudocode

Considering a horizontal line as the entrance point of the room, we define a person moving down in the video and crossing the line as entering the room. Conversely, a person moving up and crossing the line is considered to be leaving the room.

3.3.1 Algorithm for the Detection

```
define MQTT_Path with the B.A.T.M.A.N. IP of the Jetson Master
connect to the mqtt service

define the number of frames to determine the direction of the person

Start a thread to get the last frame of the stream
WHILE True:
    get the last frame from the thread
    process the frame using YOLOv8n model
    track the person and based on the number of frames defined above determine the direction o
    calculate the centroid(center of the person)
    IF a person is moving down and the centroid connects with the line THEN
        publish to the mqtt broker that a person entered
    ELIF a person is moving up and the centroid connects with the line THEN
        publish to the mqtt broker that a person left
    END IF
END WHILE
```

3.3.2 Algorithm for Data gathering and Dashboard

```
define the MQTT_PATH with the defined B.A.T.M.A.N. IP of this Jetson
connect to the mqtt service
subscribe to the MQTT_PATH

declare the people_inside counter
Initialize the flask app

WHEN it catches a message from a Jetson Worker
    IF message == 'Enter' THEN
        people_inside += 1
        sends the updated counter through socketIO to the flask app
    ELIF message == 'Leave' THEN
        people_inside -=1
        sends the updated counter people_inside through socketIO to the flask app
    END IF
END WHEN
```

Chapter 4

Simulation

4.1 Material used

- 2 *Jetson* Nano 2GB
- 2 antenas
- 1 switch
- 1 camera

4.2 Setup

For this simulation we used a shorter version of our main project description.

We will monitor the occupancy of a Lab in IT2. In this case we utilized a single *Jetson* Worker and a J*Jetson* Master integrated with the dashboard. Both devices communicate with each other using the B.A.T.M.A.N. network. The *Jetson* Worker acts as the sender, while the *Jetson* Master functions as the receiver for the messages, which are transmitted via MQTT. To establish connectivity between the computer, *Jetsons*, and cameras, we employed a switch.

4.2.1 Steps to start the simulation

- Mount the antenas in the *Jetsons*
- Connect the *Jetsons*, cameras and camera
- Connect the computer to the switch to have access to the *Jetsons* and cameras
- Access the *Jetson* assigned as Master and run the command ./setup.sh wlan0 10.1.1.20. This script creates the B.A.T.M.A.N. interface and runs the *Jetson_ Master.py* script
- Wait for the *Jetson* Master to be up. Try to access the dashboard at 192.168.1.20:5000 to see if the setup is complete.
- Access the *Jetson* assigned as Worker and run the command ./setup.sh wlan0 10.1.1.13. This will setup the *Jetson* Worker.

- the *Jetson* Worker should now be monitoring and sending messages to the *Jetson* Master.

The hardware setup should look like this:

Figure 4.1: *Setup*

4.3 Simulation Results

In this simulation, we observe the process of tracking individuals entering or leaving a building. The simulation takes place within a lab located in IT2, where the door is monitored to count the number of people currently inside the building. This setup allows for a demonstration of the occupancy tracking system in a specific department, rather than using the example of DETI as previously mentioned.

4.3.1 Person Entering

When a person enters, the *Jetson* Worker monitoring the entrance detects the person and reports it to the *Jetson* Master.

What the camera captures:

In the image below we can observe a person who is about to enter the Lab. The counter increases when the person moves forward and the centroid(represented by a red dot) crosses the line. At this point the person is considered to be inside.

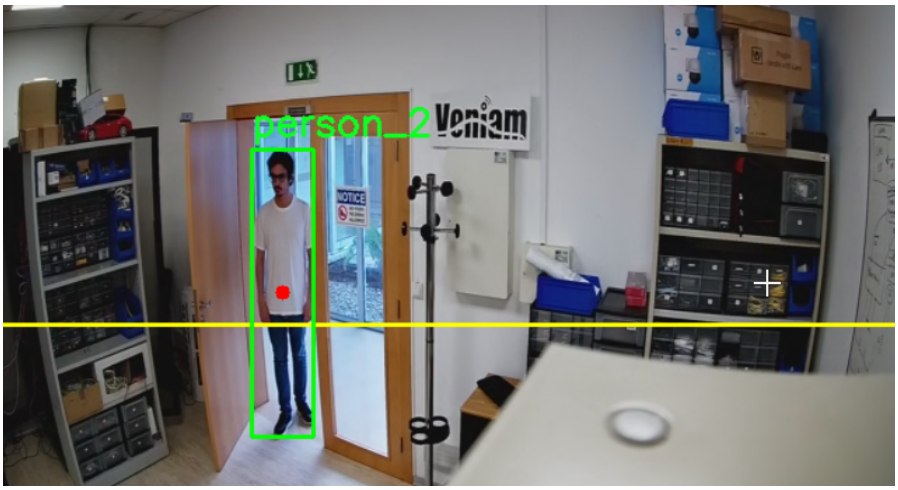

Figure 4.2: *Jetson view of a person entering*

Event of a person entering in *Jetson* Worker:

When the person enters, the *Jetson* Worker that's watching the entrances, logs:

```
Person 2 entered the building
```

Figure 4.3: *Jetson Worker Log*

Event of a person entering in *Jetson* Master:

When the *Jetson* Worker detects a person entering, it sends a report to the *Jetson* Master. The *Jetson* Master logs the event once it receives the report, confirming the reception of the

update from the *Jetson* Worker. Additionally, it displays the current number of people inside the Lab and prints the following message: "Update from *Jetson* Worker: Person entered the building. People inside: X."

```
Update from Jetson Worker: Person entered the building. People inside: 1
```

Figure 4.4: *Jetson Master Log*

4.3.2 Person Leaving

When a person leaves, the *Jetson* Worker watching the entrance detects the person leaving and reports it to the *Jetson* Master.

What the camera captures:

In the image below, we can observe a person who is on the verge of leaving the Lab. The counter increments when the person moves forward and the centroid (represented by a red dot) crosses the line. At that point, the person is considered to be outside.

Figure 4.5: *Jetson view of a person leaving*

Event of a person entering in *Jetson* Worker:

When a person enters the building, the *Jetson* Worker responsible for monitoring the entrances logs the following message to confirm the person's exit: "Person X exited the building.

```
Person 2 exited the building
```

Figure 4.6: *Jetson Worker Log*

Event of a person entering in Jetson Master:

When the *Jetson* Worker detects a person leaving, it sends a report to the *Jetson* Master. The *Jetson* Master logs the event once it receives the report, confirming the reception of the update from the *Jetson* Worker. Additionally, it displays the current number of people inside the Lab and prints the following message: "Update from *Jetson* Worker: Person left the building. People inside: X."

```
Update from Jetson Worker: Person left the building. People inside: 0
```

Figure 4.7: *Jetson Master Log*

Chapter 5

Conclusion

For this project we worked and became familiar with Ad-hoc networks and communications within one, more specifically using *B.A.T.M.A.N.* and *MQTT*. These networks are important for establishing dynamic communications without the need for traditional infrastructure, especially when it is either nonexistent or unreliable.

We also delved into object detection algorithms, which proved to be beneficial in simplifying the task of identifying individuals within an environment. These algorithms automate the process of detecting and locating people in the monitored area, thus facilitating accurate and efficient occupancy tracking.

However, we encountered our fair share of challenges. *B.A.T.M.A.N.* can be inconsistent with the devices relevant to the project. We experienced some trouble with the 2GB *Jetson's* processing capabilities. From a visual standpoint, when a *Jetson* attempts to transmit video while simultaneously performing the aforementioned tracking process, it causes a slowdown in the video feed and leads to overheating within a few seconds. The biggest issue, though, is the limited number of frames it can process from the camera feed. It can only handle about 3 or 4 frames per second. This poses a potential risk of failing to detect people, particularly if they are walking too fast or even running.

Chapter 6

Future Work

In our future endeavors, we aim to embark on the exciting challenge of deploying *B.A.T.M.A.N.* networks across all university departments and relevant buildings, while also considering potential upgrades to the *Jetson* devices. This ambitious undertaking will result in an interconnected network that provides real-time occupancy information, thereby revolutionizing campus management and greatly improving the student experience. The availability of accurate occupancy data will greatly assist in resource allocation, space utilization, and facility planning. Students will be empowered with up-to-date information on department and building occupancies, enabling them to make well-informed decisions about study spaces and facility usage. With an expanded network and upgraded devices, the system's capacity, reliability, and accuracy in tracking occupancy levels will be significantly enhanced. Overall, the deployment of *B.A.T.M.A.N* networks and *Jetson* upgrades holds tremendous potential in creating a comprehensive and interconnected system, propelling efficient campus management and elevating the overall university environment to new heights.

Chapter 7

Demo

The project's demo video showcases the monitoring of the entrance of a lab in IT2 to illustrate how the system functions. It is important to note that there might be a mismatch between the person ID displayed in the video footage and the ID shown in the *Jetson* Worker terminal. This discrepancy occurs because the displayed image is from the same script running on a personal computer, allowing for visualization of the detection process using the *Jetson* device. Although the person ID may not align perfectly between the video and the *Jetson* Worker terminal, the overall purpose of the demo remains to demonstrate the detection and tracking capabilities of the system.

Chapter 8

Acknowledgements

We would like to extend our special thanks to Professors Pedro Rito and Susana Sargento for their invaluable contribution of materials, assistance, and expertise, without which this project would not have been possible. Additionally, we express our sincere appreciation to Gonçalo Perna for his valuable support in the camera integration, as well as other aspects of the project.