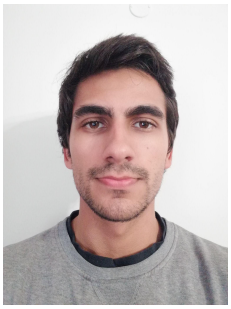
Ambient Intelligence • Alameda • Group 7

Smart Parking System

Rafael Pires 83555 • Gonçalo Guerreiro 95581 • João Fernandes 95605



1- Introduction

On average, 30% of the traffic in a city's busiest streets consists of drivers cruising for parking [3]. The need to drive around and physically look for parking spots is a highly inefficient method that consumes time, energy and causes pollution and traffic congestion, reducing the quality of life in the city.

To address this issue, we propose a Smart Parking System: a solution directed at urban street parking, that allows drivers to remotely find an available parking space and helps them navigate there by showing the quickest route. Using cameras and computer vision, this system assesses the occupancy status of each parking slot and displays the parking availability through a mobile application, where the user can look at each slot status in an interactive map. An available parking slot can be selected and the shortest route to get there is displayed, while it doesn't show that spot as available to the other users anymore. The app can function as a navigation system, as the user can search for any place and it automatically shows the nearest available parking space.

This solution can help the users save time by avoiding physical search and directing them to the desired parking spot, and, as part of a smart city ecosystem, has the potential to reduce traffic and pollution, improving urban mobility and quality of life for the overall population.

2- Literature Review

The main concepts identified in several papers related to our project regard various topics, from smart cities to computer vision. The main keywords we focused on are related to the Internet of Things, such as wireless sensors, smartphone applications, neural networks and machine learning. Others were SPS (smart parking system), traffic congestion and image processing.

The first paper we found relevant to this project (Khalid, Muhammad, et al. "From smart parking towards autonomous valet parking: A survey, challenges and future Works." *Journal of Network and Computer Applications* 175 (2021): 102935) presents a survey that provides different approaches to the design of a smart parking system, including "digitally enhanced parking, smart routing, high density parking and vacant slot detection solutions" [1].

The paper presents many different digitally enhanced parking solutions, but the one that had the most influence on our work was the Intelligent Parking Assistant. Our project's features are mostly similar to this system's, since both of them "collect the information about on-street parking slots" [1], "share the real-time occupancy information about parking slots with the vehicles on road and parking management

system” [1] and “provide a facility to book a parking before arrival” [1]. This system has some more features like the authentication for reserved slots using Radio Frequency IDentification (RFID), ensuring that they can only be accessed by the driver who booked them.

Additionally, the paper also mentions some techniques for smart routing. The one most similar to what we use is the Collaborative Path Finding, which “uses A-star algorithm for path-finding while the city is represented through a combination of square grids, where each square grid shows a part of the city” [1]. In our project we use the Mapbox Directions API, which uses a graph-based variation of the A-star algorithm that also considers obstacles like one-way streets and traffic jams in order to provide accurate and fast routing directions.

Regarding vacant slots detection, our solution is similar to the Unmanned Autonomous Vehicle described in the paper, which “takes images that can be further processed to check: (1) The number of vehicles parked in a car park; (2) Number of the vacant slots of a car park within a certain time duration” [1]. In our project we also take images of the parking slots and then process them using YOLOv3 with OpenCV to identify vehicles and check if the slots are vacant.

The second paper (Agarwal, Y., Ratnani, P., Shah, U. and Jain, P., 2021, May. IoT based smart parking system. In *2021 5th international conference on intelligent computing and control systems (ICICCS)* (pp. 464-470). IEEE.) “discusses problems faced with traditional parking lots” [2] and also “lists the impact and inconvenience caused because of inefficiency in traditional parking spaces” [2]. The authors “designed a Smart Parking System using IOT Technology, which will allow the users to find a vacant parking slot in a given area” [2], which is also the basis for our project, although with some differences such as the Infrared sensors and RFID technology which we do not intend to use. This project also focuses on parking lots instead of street parking.

The goal of Agarwal et al.’s project was “to provide customers with a user-friendly online platform to book the nearest parking spots for their vehicles before arrival” [2]. It also intended to allow automatic payments using the RFID technology and updated information about parking occupation. The paper compares existing solutions with the proposed system and explains its processes through flowcharts. Regarding challenges it states that the “Challenges of implementing smart parking in cities include lack of proper IT infrastructure followed by a learning curve involved to make citizens well versed with the technology, also reliable internet connectivity and proper maintenance” [2]. Good and reliable internet is also a really important part of our AmI project as well, although we believe the learning curve for our system is very small.

This paper also included a description of the prototype that was built to validate the proposed solution. We have similar ideas for our prototype, which include building a small model of a street with parking spaces using scaled down cars.

As for the conclusion, the authors state that the RFID was chosen instead of number plate recognition because “this method is less susceptible to error and fault-tolerant” [2] and that the “project's unique features will help people overcome their parking difficulties and manage their time too” [2]. In our solution we do not intend to identify the cars, just recognise their shape, mainly for privacy reasons.

3- Problem

Urban traffic congestion is a major issue in cities worldwide, and one of the primary causes of this problem is the need to search for parking spots. This need results in several negative consequences, including traffic jams, increased pollution, and a waste of time and resources. Moreover, drivers may become frustrated and resort to illegal parking, blocking roads or sidewalks. Our solution can help alleviate this issue. By guiding drivers to the nearest available parking spots, this can increase traffic flow and the efficiency of parking, which can benefit drivers, reduce unnecessary traffic in the city, and prevent abusive parking in public spaces. Ultimately, implementing a parking guidance system can have a positive impact on urban mobility, improving the quality of life for everyone in the city.

3.1- Assumptions

For the solution to operate as intended, we assume that we can place enough cameras to cover all the parking slots with a good view in order to detect if they are vacant or occupied. Also for this reason, the cameras can’t have their view blocked (either by objects or weather conditions) for long periods of time and every parking slot must always be captured by at least one camera. The cameras also need to be stable and should not be moved after being installed to guarantee that the masks are correctly applied to isolate each slot and analyze each one of them separately. Furthermore, the connection between the Raspberry Pi and the server must be stable so the Raspberry Pi can send the images captured by the cameras to be analyzed by the server. Additionally, the image recognition algorithm must be reliable, being able to correctly identify the vehicles and whether they are in a parking slot or not. Finally, the cars should be correctly parked in the center of the slot (inside the lines).

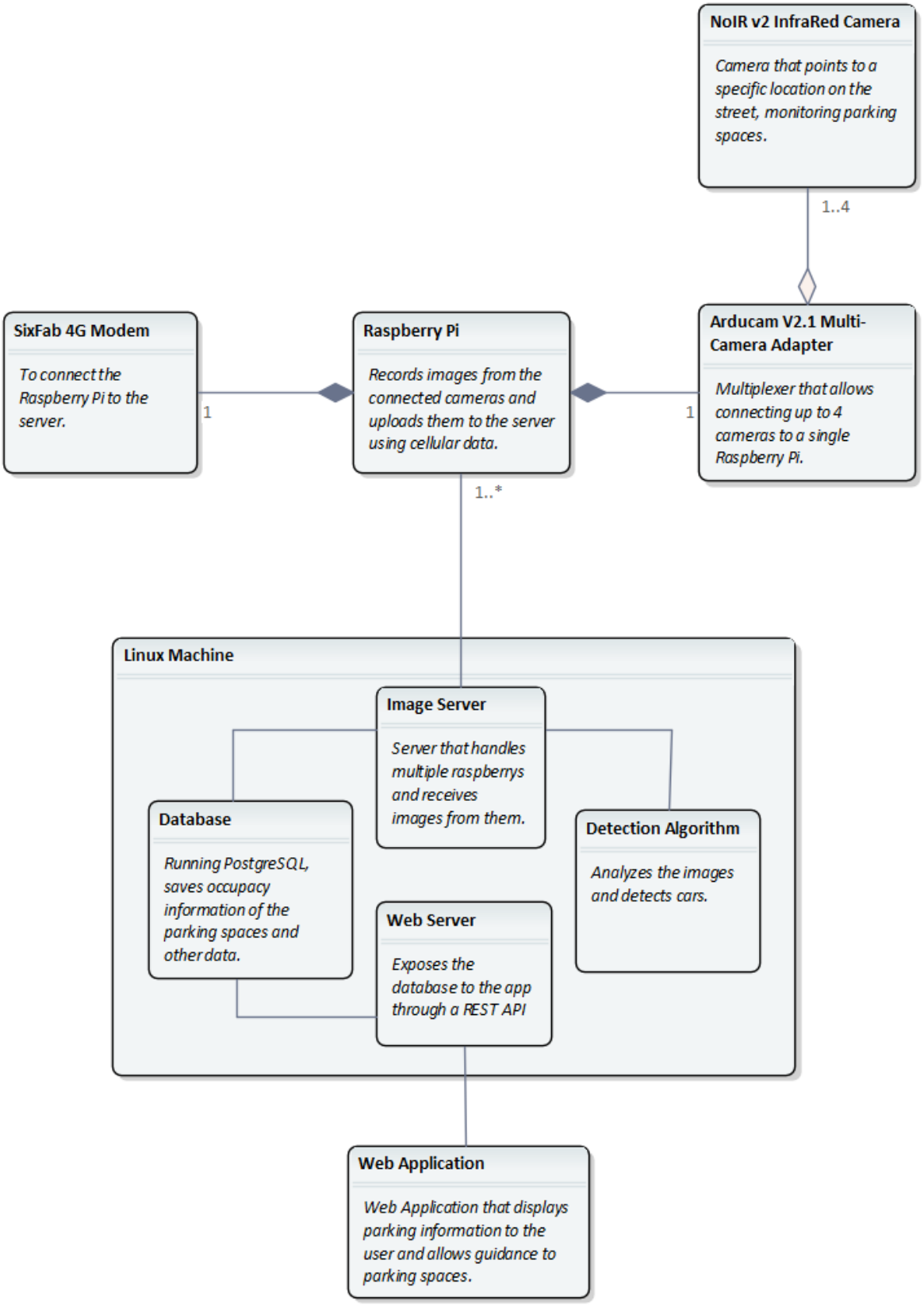
3.2- Solution Requirements

The identified requirements consist of:

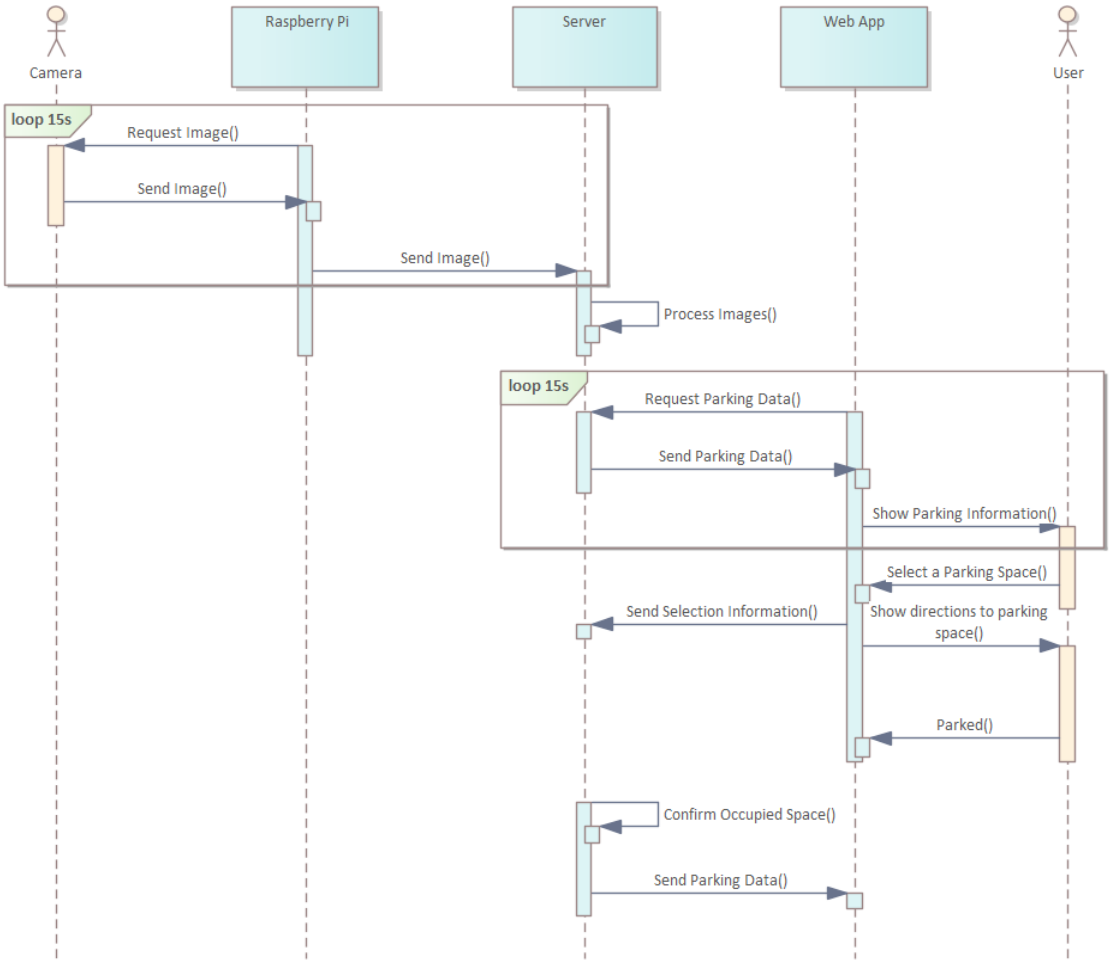
- Ability to recognise cars parked in a street;
- Ability to recognize empty parking spaces;
- Server to collect and analyze information from cameras and connect to an app;
- Display parking availability through a mobile app;
- Allow the user to search for parking spaces on a map;
- Check a specific parking space status (occupied or available);
- Guide the user on the app to a parking space;
- Warn users being guided to an available parking spot if the spot was occupied.

4- Proposed Solution

4.1- Overview



4.2- Logical Design



4.3 - Technology Selection

Main development platform: Visual Studio Code

Languages for implementation:

- Raspberry Pi: Python, Bash;
- Server: Python;
- Server database: SQL;
- Web app: HTML, JavaScript, CSS.

For our implementation, to keep costs down we decided to simplify some aspects of the project:

- Instead of multiple NoIR Cameras and respective adapter, as shown in the diagram in section 4.1, we use a single PS3 Eye USB Camera;
- Use of a single Raspberry Pi Zero 2W;
- Ideally, the server would be hosted in a cloud service (e.g. AWS, Azure), but we are using a laptop with a tunnel to expose the server to the web and the app is hosted on IST Sigma.
- Instead of using the 4G modem represented in diagram 4.1, we connect the Raspberry Pi directly to Wi-Fi in order to communicate with the server.
- Instead of a mobile app, we decided to develop a simple web app, mostly due to time constraints.
- The communication between the Raspberry Pi and the server is done through a TCP Socket, and the communication between the server and the web app is done through the Flask REST API.
- The communication between the server and the SQL database is done using the Python psycopg2 module.

The capture of the pictures on the Raspberry is done through Motion, a highly configurable program for Linux that monitors video signals from many types of cameras. In our project, it was configured to take pictures every 15 seconds. These pictures were saved in a directory, which is used when sending the images to the server also every 15 seconds. The program that uploads the pictures sends all that are present in the directory to the server, erasing them afterwards.

The server then takes the received images and analyzes them. This is done through previously setup masks that divide the pictures into parking spaces, therefore the detection algorithm only searches for cars inside those parking spaces. This algorithm uses neural networks and a pre-trained model to perform the search. Then, the result is sent to the database and becomes available to the app through the Flask REST API.

The database contains a table of parking slots, which, for each slot, saves its id (the primary key), the id of the camera that monitors it, its longitude and latitude, and its state. The state of the slot is an enum data type with the values “free”, “occupied”, “reserved” and “unknown”. It also saves the time of the last update to the slot and the time until which the slot will remain reserved (if it is in “reserved” state). Every 15 seconds the server checks, for each slot, if the last update received was more than 5 minutes ago and, if so, changes its state to “unknown”. After that, the server also

checks, for each reserved slot, if the reservation time has been exceeded and, if so, it changes its state to “free”.

The API between the server and the app has only one endpoint (/slots) which can receive GET and POST requests. The GET requests are used to send the current state of all the parking slots. The POST requests are used to make and cancel reservations of slots and are sent with the following parameters: id of the slot, its new state (“reserved” if it is making a reservation, or “free” if it is canceling) and the number of minutes for which the slot should be reserved (only sent if the request corresponds to a reservation).

The web application consists of a map provided by the Mapbox GL JS library, that shows the user's location obtained from the device’s GPS receiver and every parking space in the system with its current status, which is updated with an API request every 15 seconds. When a parking slot is selected, its details are displayed along with a “Go” button, if it is free, or a “Find nearest free slot” button, if it is occupied/unknown. In the first option, when the button is clicked, the map shows the route to get there, obtained from the Mapbox directions API, and the selection is sent to the server. The slot is then marked on the map as selected and the “Go” button is replaced by a “Cancel” button that is used to cancel the selection and restore the slot status as “free”. When a “Find nearest free slot” button is clicked, the app automatically shows the closest available parking space and the option to select it. The user can always use the search bar, provided by the Mapbox Geocoder plugin, to find a destination, and in this case, the app also directs them to the nearest available parking space.

5- Bill-of-materials

5.1- Hardware

The hardware components used are:

- Raspberry Pi running a Python program to send the captured images to the server;
- PlayStation3 Eye USB Camera to capture the images of the parking slots;
- IST Sigma to host the app;
- Laptop HP 15-da1032np as the server host.

5.2- Software

The software tools and libraries used are:

- OpenCV library and the YOLOv3 algorithm to identify the vehicles in the captured images and check if they are in parking slots;
- Motion to take pictures with the USB Camera;
- GIMP 2.10.12 to build the masks;
- Flask REST API for the communications between the server and the app;
- PostgreSQL for the database;
- Mapbox GL JS library for the map, with the Geocoder plugin for the search bar;
- Mapbox Directions API for routing on the app;
- LocalTunnel to expose the server.

6- Conclusion

With the project implementation finished, we were able to satisfy almost all of the requirements identified in section 3.2. There are two issues that prevent us from respecting the entirety of the requirements. Firstly, having a web app and not a mobile app; and secondly, if another driver parks in a spot which someone is being guided to, we do not warn the person driving to the spot that it was taken. Aside from this, we consider our solution to be an effective way to monitor parking spaces.

The detection algorithm is mostly able to reliably identify vehicles parked in spots, satisfying the first two requirements. Furthermore, the server is able to correctly communicate with the Raspberry Pi to receive the captured images and analyze them, and is also able to satisfy the app's requests. The app allows the user to search for empty parking spaces and is able to calculate the nearest available parking spot given a location. The app shows clearly the parking spots that are available and occupied and guides the user to a selected parking space. This solution, when compared to other types of sensors (e.g. ultrasound), has a greater ease of installation and efficiency since one camera monitors several parking spaces.

As for weaknesses, we consider that a fast and reliable internet connection is key for our solution to work properly and, since we did not focus on security, the solution is vulnerable to possible attacks. The detection could also be improved if we trained

the detection algorithm specifically for this task instead of using a pre-trained model. We could also develop a way to build the masks automatically using artificial intelligence, or simply replace the masks with automatic parking space recognition, for example using the markings on the road. In addition, we cannot guarantee that drivers always park correctly inside the lines, which can lead to false positives or false negatives. There is also the risk of the camera having its view obstructed or the lens becoming dirty, although the latter could be mitigated by an automatic cleaning system.

For future work, our focus would be fixing the weaknesses, especially security, and scalability. With a team of developers and more resources, we would be able to have a cloud server to handle connections from multiple Raspberries, and with enough processing power to run detections quickly, host the database and host the app. We would devote greater effort to ensure security and privacy and guarantee compliance with GDPR, while also getting the necessary certifications, and buy batches of raspberries and cameras to employ them at various locations. We would develop iOS and Android mobile applications to replace our web app or build a plug-in integration for Google Maps or Waze. Talking with municipalities or parking enforcement authorities to collaborate on the implementation of the project would also be important, to have official support and explore other ways of using this technology, such as illegal parking monitoring or automatic payments to replace parking meters. This could be done with license plate recognition.

To conclude, we enjoyed developing this solution and our team effort led to exciting results that, considering the few weeks we had, exceeded our expectations.

7- Bibliography

[1] Khalid, M., Wang, K., Aslam, N., Cao, Y., Ahmad, N. and Khan, M.K., 2021. From smart parking towards autonomous valet parking: A survey, challenges and future Works. *Journal of Network and Computer Applications*, *175*, p.102935.
[2] Agarwal, Y., Ratnani, P., Shah, U. and Jain, P., 2021, May. IoT based smart parking system. In *2021 5th international conference on intelligent computing and control systems (ICICCS)* (pp. 464-470). IEEE.
[3] Hampshire, R.C. and Shoup, D., 2018. What share of traffic is cruising for parking?. *Journal of Transport Economics and Policy (JTEP)*, *52*(3), pp.184-201.