AmbuSync - Ambulance Fleet Management

Miguel Silva
Masters in Artificial Intelligence Engineering (MEIA)
Polytechnic of Porto – School of Engineering (ISEP)
Porto, Portugal
1201045@isep.ipp.pt

Hugo Simão
Masters in Artificial Intelligence Engineering (MEIA)
Polytechnic of Porto – School of Engineering (ISEP)
Porto, Portugal
1222686@isep.ipp.pt

Vitor Silva
Masters in Artificial Intelligence Engineering (MEIA)
Polytechnic of Porto – School of Engineering (ISEP)
Porto, Portugal
1030578@isep.ipp.pt

Mariana Ribeiro
Masters in Artificial Intelligence Engineering (MEIA)
Polytechnic of Porto – School of Engineering (ISEP)
Porto, Portugal
1222746@isep.ipp.pt

Mariana Carneiro
Masters in Artificial Intelligence Engineering (MEIA)
Polytechnic of Porto – School of Engineering (ISEP)
Porto, Portugal
1232152@isep.ipp.pt

***Abstract*—Efficient management of ambulance fleets and hospital resources is crucial to ensure timely emergency responses and enhancing city-wide safety in urban healthcare systems. This project simulates urban scenarios to optimize ambulance response times and operational efficiency. By harnessing advanced technologies such as Edge-AI, intelligent environments have been developed where ambulances, vehicles, hospitals, and traffic control systems interact dynamically.**

Each agent within this simulated environment communicates and collaborates to manage traffic flow and prioritize emergency situations effectively. Ambulances are equipped with sophisticated communication and navigation systems to navigate through complex urban landscapes swiftly and safely. Concurrently, hospitals play a pivotal role in this ecosystem by coordinating resource allocation to handle incoming patients effectively.

The simulation framework integrates SPADE for agent communication and Gazebo for realistic scenario modeling, demonstrating how smart environments and multi-agent systems synergistically enhance urban emergency response capabilities. This approach not only optimizes logistical aspects of ambulance dispatch and patient transport but also ensures efficient allocation of critical resources during crisis situations.

By showcasing the efficacy of these technologies in a simulated urban environment, this project contributes to the advancement of urban healthcare systems. It underscores the transformative potential of intelligent systems in improving emergency response effectiveness and overall urban safety infrastructure. Future research can build upon these findings to further refine and implement these technologies in real-world settings, thereby enhancing healthcare resilience in urban environments.

***Index Terms*—Intelligent environments, multi-agent systems, urban healthcare, emergency response, simulation**

I. Introduction

Efficient management of ambulance fleets and hospital resources stands as a cornerstone of urban healthcare systems, where timely emergency response can significantly influence patient outcomes [1]. Recent technological advancements, particularly in smart environments and multi-agent systems, offer promising solutions to optimize these critical operations. This paper presents a project focused on simulation-based management of ambulance fleets and hospital resources, aiming to explore how technology can enhance efficiency and safety in emergency services within dynamic urban contexts.

The primary objective of this project is to create a detailed representation of complex urban scenarios where ambulances, regular vehicles and the dynamics of traffic rules interact seamlessly. Tackling challenging scenarios such as multi-lane roundabouts and bustling intersections, where effective communication and cooperation among all participants are essential to ensure optimized response times and city-wide safety.

To achieve these goals, four distinct types of agents have been developed: hospitals, ambulances, regular vehicles, and traffic lights. Each plays a critical role in simulated urban ecosystem [2], engaging in dynamic interactions designed to facilitate smooth and coordinated operations. Leveraging advanced technologies the system processes real-time data and makes informed decisions even in unpredictable conditions.

Furthermore, the use of tools like SPADE for agent communication, ROS2 for integrating the robot's operating system along with Gazebo, responsible for real-world simulation, provides a robust and adaptable framework that supports the implementation of adaptive strategies. This project aims not only to optimize ambulance fleet management but also to demonstrate how intelligent systems can significantly improve emergency service efficiency and urban safety

comprehensively. By integrating real-time machine learning algorithms and data analysis, the system is expected to predict future demands and adjust operations accordingly, minimizing response times and enhancing resource allocation.

By creating a detailed and interactive simulation environment, the main goal of this project is to provide valuable insights into the transformative potential of emerging technologies in crisis management within urban settings. This work not only paves the way for future research and development but also aims to inspire innovations that ensure continuous improvement of public healthcare services in modern cities.

II. State of Art

Efficient management of ambulance fleets in urban settings involves leveraging advanced technologies to enhance emergency response capabilities. Autonomous vehicles equipped with machine learning, computer vision, Edge-AI, and communication systems play a significant role in this evolution. These technologies enable vehicles to detect and respond to emergency situations promptly. For instance, convolutional neural networks process images to identify ambulances, while audio detection systems recognize emergency sirens. Additionally, obstacle avoidance modules using ultrasonic sensors help prevent accidents, ensuring safer interactions with emergency vehicles on the road [2].

Furthermore, integrating these autonomous capabilities into the management of ambulance fleets optimizes route planning and navigation. Intelligent ambulance systems are designed to highlight emergency routes, utilizing technologies like GPS and mobile communication to facilitate efficient transportation of patients to hospitals. This integration not only enhances the operational efficiency of ambulance services but also improves road safety by enabling vehicles to dynamically adjust routes based on real-time traffic conditions and emergency priorities [3].

Thus, the integration of autonomous vehicle technologies into ambulance fleet management represents a transformative approach towards achieving faster, safer, and more efficient emergency response outcomes in urban environments. The implementation of autonomous transportation systems in emergency healthcare services has also been widely discussed. The use of smart environments, equipped with sensors and data processing systems, allows autonomous vehicles to be dispatched more efficiently. Multi-agent systems manage the fleet of vehicles, where each vehicle makes decisions autonomously but cooperates with other vehicles and urban infrastructure to optimize routes and prioritize patients based on the severity of their conditions. This distributed and coordinated approach improves emergency response capability, making the system more flexible and scalable [4].

Edge-AI has also been explored to recommend routes and resources for autonomous ambulances in smart cities. By processing data in real-time close to the source, Edge-AI enables faster and more efficient decision-making. Multi-agent systems are used to coordinate the fleet of ambulances, while intelligent environments within the vehicles monitor patient conditions and adjust the route or internal environment to ensure the best possible care during transport. This combination of technologies allows ambulances to respond more effectively to the dynamic conditions of medical emergencies, improving the quality of healthcare services [5].

Ambulance fleet management is significantly benefiting from advances in machine learning, computer vision, communication systems, and edge artificial intelligence. Recent scientific literature demonstrates that integrating these technologies in urban settings not only optimizes ambulance operations but also enhances response capabilities in critical situations, contributing to the safety and well-being of citizens. Additionally, the exploration of multi-agent systems aims to improve coordination among different elements of the emergency system, such as ambulances, cars, and hospitals, enabling more dynamic and adaptive fleet management to respond quickly and effectively to emerging emergency demands.

This project focuses on simulating intelligent environments and multi-agent systems for ambulance fleet management. Leveraging the SPADE library for agent communication [6] and the Gazebo platform for simulation [7], aiming to develop a robust system where ambulances equipped with sensors collect real-time data and autonomously make decisions based on this information. This interaction model will facilitate the implementation of adaptive decision-making strategies, even under adverse conditions, thereby promoting a faster and more efficient response to medical emergencies. In doing so, the project aim to significantly advance hospital resource management and enhance citizen safety during emergencies.

III. Development

A. Multi-Agent System

The increasing need for optimization and streamlining of emergency medical services has driven research and development of intelligent systems capable of efficiently coordinating the flow of ambulances, vehicles, and pedestrians during critical situations. Within the scope of the Multi-Agent Systems project, an effort was made to create a robust and dynamic solution for this pressing demand.

The developed system comprises agents that play distinct and complementary roles, reflecting the complex dynamics and interdependencies present in the emergency medical management environment. The main entities of the system include ambulances, cars, hospitals, and traffic lights, each performing specific functions and interacting in a coordinated manner to ensure the effectiveness of the service provided.

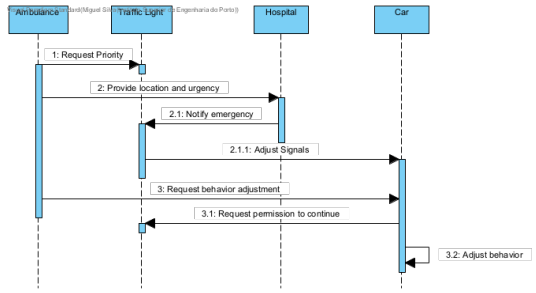

Fig. 1. Activity Diagram - Agents Interactions

One of the approaches adopted was the use of the SPADE library in Python, which provided the necessary resources for implementing the agents and facilitating communication between them. Through carefully delineated logic, agents were programmed to interact according to the following guidelines:

- **Ambulance:** Responsible for transporting patients to hospitals, the ambulance communicates its location and emergency status, interacting with other agents to ensure a safe and rapid passage. It cooperates with traffic lights to obtain priority, communicates with cars to divert or stop, and coordinates with hospitals to ensure the arrival of patients.
- **Hospital:** Receives patients transported by ambulances, communicating with them to provide their location and urgency of service. It notifies ambulances to create emergencies and coordinates with them to prioritize emergency cases.
- **Traffic Light:** Regulates traffic and communicates with ambulances to change the signal. It adjusts dynamically based on events and traffic flow needs.
- **Car:** Travels a random path, interacting with traffic lights to determine whether it can continue or not. It must stop and/or divert based on the presence of other entities. It communicates with ambulances to adjust its behavior and allow quick passage during emergencies.

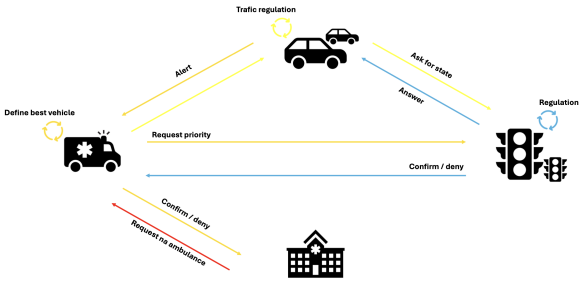

Fig. 2. Agents Interactions

In this system, agents are classified into reactive, and hybrid based on their behavior and interaction patterns. The traffic light and car agents are reactive agents as they respond dynamically to environmental changes based on basic and predefined rules. The ambulance is a hybrid agent combining reactive and deliberative mechanisms, it not only responds to real-time traffic conditions but also engages in decision-making processes such as negotiating the best ambulance to perform an emergency task. The hospital is also hybrid, incorporating deliberative capabilities to assess emergency situations and request ambulance services when needed.

TABLE I
INTERACTIONS BETWEEN ENTITIES IN THE TRAFFIC SYSTEM

Entity	Functions
Ambulance	• Travels a safe path • Communicates with traffic lights asking for priority • Communicates with car asking to move aside or stop • Communicates with the hospital receiving emergency requests
Hospital	• Communicates with ambulances to provide location starting an emergency
Traffic Light	• Regulates traffic • Communicates with ambulance to change traffic flow • Adjusts signal based on events
Car	• Travels a "random" path • Communicates with traffic lights for permission to proceed • Stops/moves aside based on distance to other cars or the presence of an ambulance

The development of the system also involved creating diagrams using ACL messages (figure 3), providing a detailed view of the interactions between agents and aiding in the precise definition of their behaviors. Additionally, the study of the MQTT protocol was initiated to ensure effective communication between agents and unify the work of different curriculum units.

The Multi-Agent Systems project represents a significant step towards improving emergency medical services, demonstrating how the application of artificial intelligence techniques and distributed systems can contribute to a more efficient and coordinated response in critical situations.

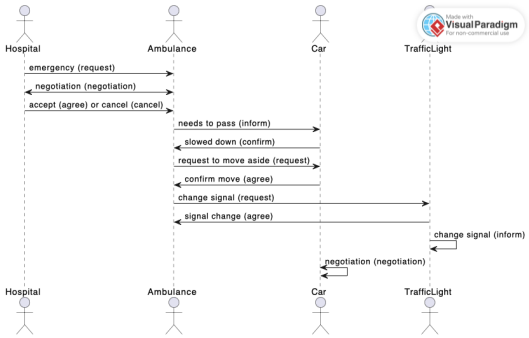

Fig. 3. Sequence Diagram - ACL Messages

B. Intelligent Environment

The progress achieved in the smart environments project reflects a collaborative and comprehensive approach, where various actors play complementary roles to ensure the effectiveness and safety of the system. Initially, it was planned to create a Gazebo simulated environment with ambulances, conventional vehicles, pedestrians, roundabouts, and intersections. However, due to time constraints and technical challenges, the current simulated environment includes only a basic scenario: two ambulances, two cars and two traffic lights. Although this scenario is simpler than originally envisioned, it still offers significant practical utility.

*1) **Vehicles**:* At the core of this system are vehicles, the ambulances which are responsible for transporting patients to hospitals and the normal vehicles with the objective to simulate real life traffic situation. Equipped with advanced communication technology, such as location and alert systems, ambulances dynamically interact with other agents, ensuring a safe and swift passage. Conventional vehicles represent the usual traffic, but their collaboration during emergencies is vital. By cooperating with ambulances, these actors facilitate the response, allowing ambulances to pass and follow their guidance to minimize congestion. Initially, it was aimed to use customized cars with meshes to make the simulation as close to the real world as possible, figure 4.

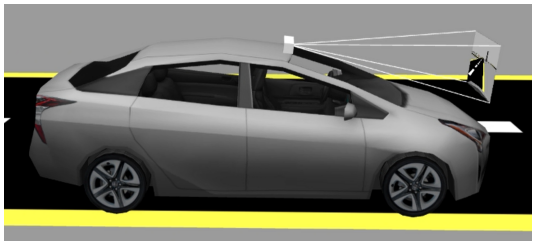

Fig. 4. Vehicle Prius in Gazebo world

However, due to computational constraints, that possibility was discarded and simplified. Considering this, the Gazebo models built for the vehicles consist of a box as the chassis and three spheres as the wheels. The cars are also equipped with two sensors: a camera for detecting traffic lights and road markings, and a laser for obstacle detection. The final result may be found in figure 5.

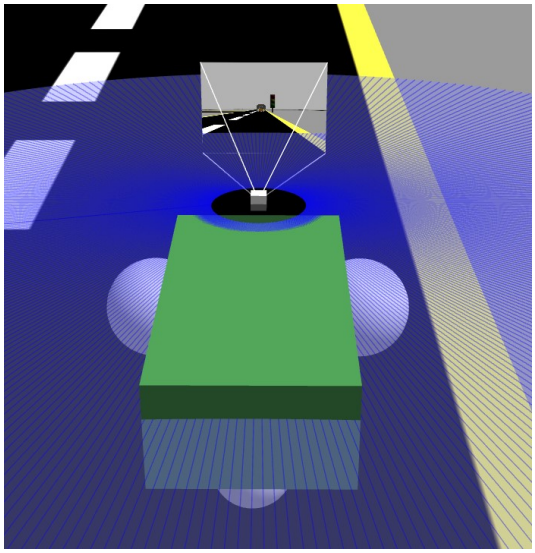

Fig. 5. Vehicle in Gazebo world

The *rqt_graph* [8] provides a comprehensive overview of the nodes and their communication through topics. Given the extensive and complex nature of this graph, a concise summary of each node's purpose and role within the system will be provided. This approach will help to clarify the interactions and responsibilities of the individual nodes without overwhelming detail.

ambulance_signal_publisher node is exclusive to normal vehicles and is designed to continuously publish a Boolean signal indicating the presence of an ambulance. This node creates and publishes a message with the data set to True on the **/ambulance_signal** topic, signaling that an ambulance is detected. The purpose of this node is to simulate or provide an ongoing signal that can be used by other nodes to react appropriately to the presence of an ambulance.

ambulance_evader node also exclusive to normal vehicles, subscribes to the ambulance signals published by the previous node. Its role is to decide if the vehicle should pause to let the ambulance pass, publishing a pause signal on the **/pause_signal** topic when necessary.

laser_controller node controls the LIDAR from each vehicle, and publishes proximity scan data on the **/proximity/scan** topic. This node is crucial for detecting obstacles and other objects around the vehicle.

road_detector node plays a crucial role for all vehicles in the system, responsible for detecting road characteristics using the vehicle's camera data. It processes the image received via the **/camera/image_raw** topic and utilizes vital camera information available in **/camera/camera_info**. Detection of lane markings is performed using the Hough transform after specific

preprocessing steps: for parallel lines on the road, a region of interest is created, followed by grayscale conversion, GaussianBlur, and edge detection using Canny. For perpendicular lines, a yellow mask is used instead of the region of interest, followed by the same detection process. The results for parallel lines are then published on the **/lines_detected** topics to indicate lane detection and **/angle_car** to provide the angle required to keep the vehicle between these lines.Additionally, results for perpendicular lines are published on the **/intersection_detected** and **/intersection_distance** topics. The distance to intersections is accurately calculated using camera dimensions and horizontal field of view, ensuring precise vehicle navigation in the simulated environment. All parameters used in image processing and distance calculation have been meticulously adjusted to optimize accuracy and efficiency, ensuring correct and safe vehicle positioning during simulation

car_controller node is a central node that subscribes to various sensor data and control signals. It is responsible for managing multiple cars by processing information from different topics. The node subscribes topics related to road lines **/lines_detected**, **/angle**, perpendicular lines **/intersection_distance**, **/intersection_position**, LIDAR sensor **/proximity/scan**, **/pause_signal**, topics that facilitate car movement and speed **/start_stop_signal**,**/slow_down_signal**, finally, it subscribes to the **/pause_signal** which is responsible for controlling all connections and stopping them if necessary. The node publishes to the**/set_velocity** topic, which is responsible for controlling the car associated with the namespace. Based on all this data, it makes safe driving decisions.

diff_drive node receives the velocity commands from the car controller and converts them into direct drive instructions for the vehicle. It also publishes odometry data on the **/get_odometry topic**, providing feedback on the vehicle's movement.

semaphore_detector_node detects semaphores, using image data and odometry. It subscribes to the **/camera/image_raw** and **/get_odometry** topic to analyze camera frames and track the vehicle's position, respectively. This node identifies semaphore colors, and trough both the vehicle's and semaphore's odometry it determines which semaphore is being detected. It publishes the vehicle's ID on the semaphore **/semaphore_detected** topic providing crucial data for vehicle navigation and interaction with traffic signals.

*2) **Semaphores**:* The semaphores are also crucial agents since they will manage the traffic to avoid overflow. The gazebo model was designed to resemble the traffic light consisting on a cylinder to serve as post, a box to be the base and six spheres to serve as the lights: three of them resemble the lights turned off and the other on.

Since there is no plugin that would be compatible with this model, in order to make the semaphore share and switch its current color, the team decided to develop their own plugin, not only for the sake of functionality but also for the sake of learning. The **model plugin** [9] developed does the following: it subscribes a topic with the name of its ID and switches the color accordingly to the number that was published in that topic: 0 for red, 1 for yellow, 2 for green and 3 turned off. The changing of the color, works by hiding the spheres by setting their visibility off, this plugins also publishes the position of the traffic light for interaction purposes. The final result may be found in figure 6.

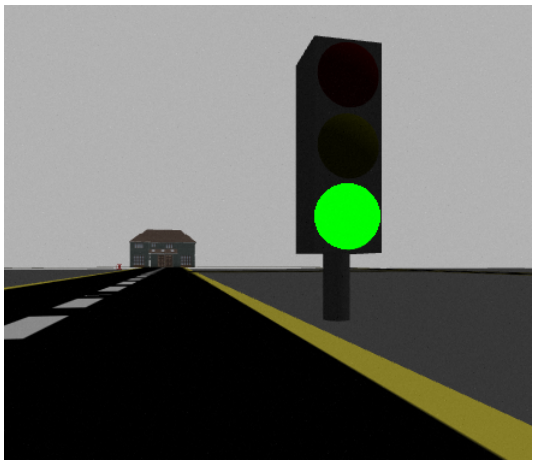

Fig. 6. Semaphore in Gazebo world

Furthermore, it is noteworthy to highlight the implementation of an IoT traffic light using the ESP32. The hardware not only facilitated efficient integration between the simulated environment and physical hardware but also enriched the development experience by offering a more realistic visual understanding of the project.

The C++ code, developed in the Arduino IDE for ESP32, integrates IoT capabilities by configuring WiFi connectivity and MQTT communication. It establishes a stable WiFi connection, initializes an MQTT client to connect to a specified server, and monitors the "fromROS/semaphore" MQTT topic for incoming messages. Upon receiving a message, the callback function processes it, controlling LEDs connected to ESP32 pins to simulate traffic light states (red, yellow, green, or off). The reconnect function ensures automatic reconnection to the MQTT server if the connection is lost, while setup() initializes pin modes and serial communication, and loop() continuously runs MQTT client operations to handle incoming messages. On the ROS side, a node subscribes to the semaphore topic and publishes to the "fromROS/semaphore" topic whenever the traffic light's color changes.

The circuit designed for traffic light control was initially sketched in Tinkercard, where, due to platform limitations, it was not possible to use an ESP32 or an actual traffic light. To work around this restriction,

three LEDs and an Arduino were used to illustrate the traffic light's functionality. This sketch served as a visual and conceptual foundation for the subsequent development in physical hardware and it may be seen in figure 7.

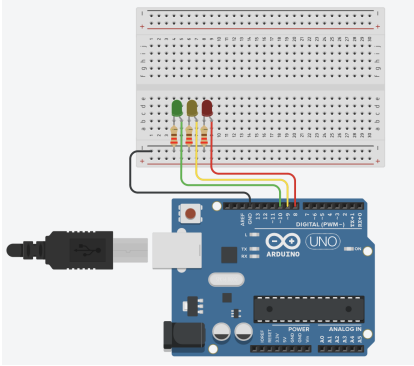

Fig. 7. Tinkercard Structure

Regarding the circuit itself, it was composed by an ESP32, a traffic light module, four jumpers and a 300 Ω resistor.

The traffic light module, depicted in figure 8, features four pins: one ground pin linked to the ESP32 GND pin, and R, Y, and G pins connected to digital output pins. These pins activate the red, yellow, and green lights, respectively, when triggered.

Fig. 8. Semaphore Module [10]

The inclusion of a 300 Ω resistor in the circuit serves a crucial role in current regulation. It is specifically chosen to limit the current flowing through the traffic light LEDs to a safe level, thereby preventing potential damage from excessive current. The resistance value of 300 Ω is selected based on the operating voltage of the ESP32 and the forward voltage of the LEDs in the traffic light module. By calculating the appropriate resistor value, the current is ensured to remain within the safe operating range for both the ESP32 and the LEDs, enhancing the longevity and reliability of the components in the circuit

In conclusion, the circuit effectively integrates an ESP32 with a traffic light module, ensuring proper functionality and safety through the use of a 300 Ω resistor. This design highlights the importance of current regulation in protecting electronic components. The complete circuit layout can be visualized in figure 9.

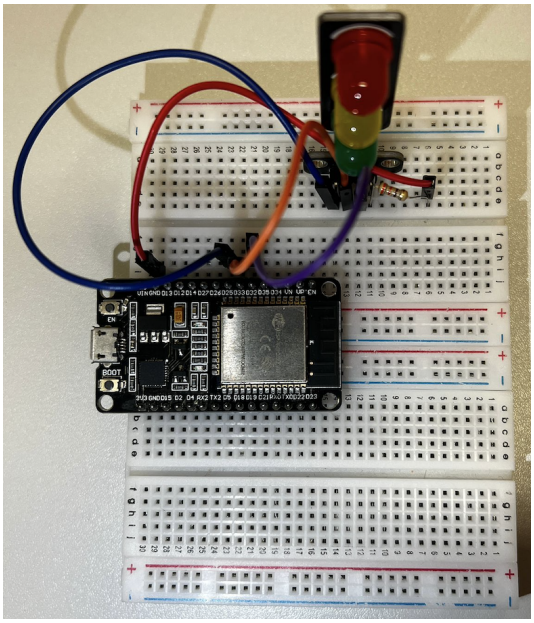

Fig. 9. Semaphore Circuit

*3) **Hospitals**:* Hospitals complete the cycle by receiving patients brought by ambulances playing a crucial role in managing demand and prioritizing emergency cases. The model built for the hospital consists on a model of a house downloaded from the internet and the team added a sign with the letter H to symbolize the building. The final result may be found in figure 10.

Fig. 10. Hospital in Gazebo world

IV. Tests and Results

The progress achieved in the project underscores a collaborative and multifaceted approach, ensuring the system's effectiveness and safety through the complementary roles of various actors. While the initial goal was to create a comprehensive Gazebo simulated environment with ambulances, conventional vehicles, pedestrians, roundabouts, and intersections, practical constraints led to a more simplified yet functional scenario that still provides significant practical utility and serves as a solid foundation for further developments, in figure 11 its possible to observe a physical overview of the final solution.

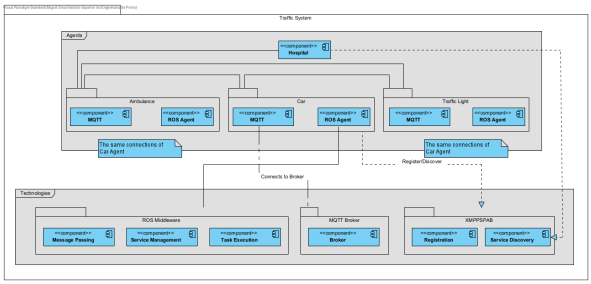

Fig. 11. Physical View

Throughout the development process, numerous difficulties were encountered, particularly with Gazebo and ROS2. The lack of comprehensive documentation, coupled with different versions having varying and sometimes conflicting documentation, posed significant challenges. Additionally, the Gazebo community's relatively low activity level meant that finding solutions and support was often difficult. These factors contributed to a steep learning curve and delayed progress.

Another major challenge was integrating the smart environment with multi-agent systems due to the hardware limitations of the team members. The individual hardware constraints meant that, while the test functionalities in isolation could be tested with positive results, integrating and testing the entire system proved infeasible. Consequently, this solution serves more as a proof of concept rather than a fully integrated solution.

Furthermore, the available time for this project was quite limited. Adapting to a new platform and dealing with associated challenges became the priority, hindering the exploration of new learnings as we aimed to develop a robust and complex project. With more time, it would have been possible to develop a more comprehensive and integrated solution, overcoming the technical challenges and hardware limitations faced by the team.

Despite the numerous challenges faced, several functionalities were tested successfully in isolation, demonstrating significant advancements. These include:

1) **MQTT Communication**: The communication between the two main components of the project via MQTT was successfully implemented. This allowed for effective data exchange and coordination between different parts of the system.
2) **Traffic Light Recognition**: The system was able to correctly recognize traffic lights and respond appropriately. The hardware implementation of the traffic light could accurately replicate a specific traffic light from the simulated environment, showcasing the integration of IoT components.
3) **Obstacle Detection**: The vehicles were equipped with sensors that successfully detected obstacles in their path. This functionality is crucial for ensuring safe navigation and was tested with positive results.
4) **Line Detection-Based Driving**: The vehicles could detect road lines and use this information for navigation. This feature ensured that the vehicles stayed within their lanes and identifying and acting correctly at intersections..

Despite the scaled-down simulation environment and the numerous challenges faced, the project demonstrates significant advancements in smart environments. The development process highlighted the importance of collaboration, innovative problem-solving, and practical implementation. The various models and nodes provide a robust framework for future enhancements, paving the way for more complex and realistic simulations. This foundational work sets the stage for ongoing improvements and broader applications in real-world scenarios.

In conclusion, the project not only underscores the potential of smart environments but also provides valuable insights and lessons for future research. The isolated functionalities tested successfully offer a glimpse into the system's capabilities, and with additional time and resources, a fully integrated solution is within reach.

V. Discussion of Results and Conclusions

This project has demonstrated the significant potential of intelligent environments and multi-agent systems in optimizing the management of ambulance fleets and hospital resources in urban settings. The integration of SPADE for agent communication and Gazebo for simulation facilitated a robust and adaptable framework. Despite the efforts to develop the most comprehensive simulation environment possible, challenges arose in implementing all planned scenarios. Time constraints and technical challenges within the Gazebo and ROS2 frameworks prevented the integration of pedestrians, crosswalks, roundabouts, and intersections in simulations.

Nevertheless, the project remained committed to developing a foundational scenario that retains practical utility. The simulation features an intersection with two traffic lights, two cars, and an ambulance on an emergency response mission, striving to reach its destination swiftly and safely. Additionally, during project development, the simulation was augmented with a hardware traffic light to enhance visual clarity and understanding.

While the project scope was constrained, the results clearly illustrate how intelligent technology integration can enhance the efficiency and safety of urban emergency responses. Looking ahead, there are several avenues for future work to expand and refine this approach. One crucial area is to elevate the simulation complexity by incorporating additional scenarios inspired by real urban environments.

Furthermore, plans include implementing advanced scheduling algorithms for ambulance deployment to

optimize resource allocation. Techniques such as TOPSIS or genetic algorithms offer potential for developing scheduling strategies that are both logical and maximize system efficiency. These enhancements aim to further improve emergency response effectiveness in dynamic urban settings.

In conclusion, this work serves as a foundational step towards future research and development efforts. By addressing these challenges and pursuing these opportunities for enhancement, the goal is to foster the creation of more advanced and comprehensive systems for managing urban emergencies.

REFERENCES

[1] Lucas Binhardi Branisso. Sistema multiagente para controle de veículos autônomos. Dissertação de mestrado, Universidade Federal de São Carlos, São Carlos, SP, June 2014. Orientador: Prof. Dr. Edilson Reis Rodrigues Kato.

[2] Aparajit Garg, Anchal Kumar Gupta, Divyansh Shrivastava, Yash Didwania, and Prayash Jyoti Bora. Emergency vehicle detection by autonomous vehicle. *International Journal of Engineering Research & Technology (IJERT)*, 08(05):104–108, May 2019.

[3] AbdelGhani Karkar. Smart ambulance system for highlighting emergency-routes. In *2019 Third World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4)*, pages 255–259, 2019.

[4] Muhammad Khalid, Muhammad Awais, Nishant Singh, Suleman Khan, Mohsin Raza, Qasim Badar Malik, and Muhammad Imran. Autonomous transportation in emergency healthcare services: Framework, challenges, and future work. *IEEE Internet of Things Magazine*, 4(1):28–33, 2021.

[5] Syed Thouheed Ahmed, Syed Muzamil Basha, Manikandan Ramachandran, Mahmoud Daneshmand, and Amir H. Gandomi. An edge-ai-enabled autonomous connected ambulance-route resource recommendation protocol (aca-r3) for ehealth in smart cities. *IEEE Internet of Things Journal*, 10(13):11497–11506, 2023.

[6] Javi Palanca. *SPADE Documentation*, 3.3.0 edition, September 2023. Release 3.3.0.

[7] Mirella Santos Pessoa de Melo, José Gomes da Silva Neto, Pedro Jorge Lima da Silva, João Marcelo Xavier Natario Teixeira, and Veronica Teichrieb. Analysis and comparison of robotics 3d simulators. In *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, pages 242–251, 2019.

[8] The Robotics Back-End. rqt_graph – visualize and debug your ros graph, 2024. Accessed: 2024-06-22.

[9] Open Source Robotics Foundation. Model plugins. https://classic.gazebosim.org/tutorials?tut=plugins_model, 2014. Accessed: 2024-06-20.

[10] ESP32IO. Esp32 traffic light. https://esp32io.com/tutorials/esp32-traffic-light#content_function_references. Accessed: 2024-06-22.