University of Stuttgart
Institute of Industrial Automation and
Software Engineering

Dynamic Geofences for Autonomous Transport Robots in Indoor Logistics

Research Project 3265

Submitted at the University of Stuttgart by
Sowjanya Krishna

INFOTECH

Examiner:	Prof. Dr.-Ing. Michael Weyrich
Supervisor:	Hannes Vietz, M.Sc

23.05.2022

Table of Contents

Table of Figures

Table of Tables

Table of Abbreviations

MQTT	**M**essage **Q**ueuing **T**elemetry **T**ransport
AGV	**A**utomated **G**uided **V**ehicle
ARENA	**A**ctive **R**esearch **E**nvironment for the **N**ext generation of **A**utomobiles
IIoT	**I**ndustrial **I**nternet **o**f **T**hings
URL	**U**niform **R**esource **L**ocator
GUI	**G**raphical **U**ser **I**nterface
CoAP	**C**onstrained **A**pplication **P**rotocol
CSS	**C**ascading **S**tyle **S**heets
JSON	**J**ava**S**cript **O**bject **N**otation
SQL	**S**tructured **Q**uery **L**anguage
HTTP	**H**ypertext **T**ransfer **P**rotocol
ORM	**O**bject-**R**elational **M**apping
API	**A**pplication **P**rogramming **I**nterface

Glossary

Geofence	A virtual geographic boundary, defined by GPS or RFID technology, that enables software to trigger a response when a mobile device enters or leave a particular area.
Polygon Offset	Specifies a scale factor that is used to create a variable depth offset for each polygon
Back-end	Refers to parts of a computer application or a program's code that allow it to operate and that cannot be accessed by a user.
Front-end	The frontend of a software program or website is everything with which the user interacts
AGV	Self-guided vehicles are material handling systems or load carriers that travel autonomously throughout a warehouse, distribution center, or manufacturing facility, without an onboard operator or driver
Localization	Deployment of networks of sensors, able to collect and transmit data in order to determine the targets position
NAiSE GmbH	NAiSE enables coordination and traffic control of ALL intralogistics participants (pedestrians, industrial trucks and mobile robots) using sensor infrastructure and intelligent integrated communication.

Abstract

New concepts and technologies, ranging from manufacturing to intralogistics and transportation, are required to kickstart the intelligent factory. Mobile robots in industry, autonomous vehicles in transportation and logistics, IIoT, and augmented reality are already being explored. All of this, however, necessitates that today's networks soon hit their limitations. Industrial 5G's exceptional dependability and reduced latency times, as well as its full IIoT connection, open the way for trend-setting industrial applications.

One of the most important application is identification and eradication of possible dangers, as well as the documenting of evidence for safety cases, are critical to system safety. This is usually done during the design and development phase of the system. During the operating phase of automated systems, however, there is a need to cope with unknowns and uncertainties. This project focuses on virtual borders surrounding geographic zones (i.e., geofences) that may be used as an active countermeasure for dynamic risk management in automated transportation and production environments.

In this paper, a proposal is made for a communication framework capable of connecting to the existing geofencing system to further add dynamic geofence functionalities. To implement this, a novel approach based on computational geometrics is built. The experiments determine how factors such as (Automated Guided Vehicles) AGV speeds, tag transmission, software and AGV capabilities affect the geofence around it. This research describes how this approach could be used to create a safety case for the AGV operation.

Key Words: *Geofence, Automated Guided Vehicles, Risk Management, Dynamic Factors, Communication Framework, Polygon Offset*

1 Introduction

The influence of 5G on the Industrial Internet of Things (IIoT) is a strongly discussed topic. While many consumers are now using low-band 5G as a replacement for LTE, the high-band form of 5G for metropolitan regions and production environments, which is still under development, is the more essential breakthrough from an industrial standpoint. With reduced latency, large data speeds, and high reliability, 5G technology fits the connection needs of digitalized production environments. Wireless communication is critical for the Smart Factory and Industry 4.0 because it offers seamless, ubiquitous, and scalable connectivity between machines, people, and sensors, as well as mobile entities like mobile robots, automated guided vehicles (AGVs), drones, and humans.

Safety-critical indoor localization of mobile robots through 5G is one of the most promising application areas. With the increasing implementation of automated guided vehicles (AGV), the need for reliable control and monitoring of vehicles is growing [1]. AGVs are widely regarded as one of the most important facilitators of shop-floor flexibility in logistics. They have the ability to transport goods and resources without following any pre-determined paths. AGVs with self-guided navigation systems make their way to target workstations using a vast range of commercially available models.

As manufacturing becomes increasingly automated and requires increased flexibility, there is a move to more dynamic factories. This dynamism brings particular challenges when attempting to provide assurance for the safety of operation prior to deployment [2]. Dynamic factories are also inherently interconnected, adding to the complexity of the safety analysis and assurance task. It is crucial that these safety challenges are addressed, particularly for autonomous applications [3]. In a dynamic factory, AGV operation is not limited to defined routes and therefore cannot rely on a fixed infrastructure or environment. In addition, AGVs will frequently be required to work in groups and in collaboration with humans. To ensure factory safety, it must be demonstrated that AGVs can travel safely, avoiding collisions with other objects or persons, and avoiding possibly changing hazardous regions (danger zones). Geofencing is a widely utilized location-based strategy that we investigate for controlling risk in hazardous manufacturing locations.

1.1 Geofence

A geofence is a virtual perimeter for a real-world geographic area. A geo-fence could be dynamically generated or match a predefined set of boundaries. The use of a geofence is called geofencing, and one example of use involves a location-aware device of a location-based service (LBS) user entering or exiting a geo-fence. This activity could trigger an alert to the device's user as well as messaging to the geo-fence operator [4].

The geofences can serve as a countermeasure that could either eliminate the encountered hazards or reduce the risk of a mishap to an acceptable level. Geofencing is a virtual boundary (shape and dimension) defined for each zone (e.g., loading, unloading and transportation zones etc.) [5]. Within the geofencing zone, the mobile robots are constantly monitored. Geometric shapes can be created in site zones in order to establish virtual borders around geographic zones (i.e. geofences).

1.1.1 Types of Geofences

1. Static Geofences- Geofences that are static are fixed or constant. The loading, dumping, and charging regions at the locations may not alter over time. Aside from that, autonomous vehicle movement must be prohibited in a variety of permanent sites, such as hazardous material storage facilities and areas/regions where people labor.
2. Dynamic Geofences- Geofencing that dynamic shifts over time. For example, vehicles that are not equipped with or have malfunctioning obstacle detection equipment, failed hardware, or are transporting dangerous items are hazardous. In the event of bad environmental circumstances, such as a slick surface, automated vehicle movement can be prevented using dynamic geofences.
3. Periodic Geofences- Periodic geofences are active or inactive for a certain amount of time. As a result, they may be used to limit mobility in certain locations for a set length of time.
4. Conditional Geofence- Permissions connected with a geofence are determined by parameters such as the maximum number of cars that can be permitted together. Furthermore, to address issues such as route obstruction, the movement of an autonomous vehicle with a certain human-driven vehicle can be designated as conditional, allowing an alternate travel path to be formed.

1.2 Objective

The goal of this project is to create and construct a system that allows dynamic protection zones to be formed using geofences or spatial points. Static geofences can already be defined using an existing system. The system in this thesis will interface with an existing system using MQTT in order to acquire spatial points that construct static geofences. The updated system will be able to alter these geographical locations, add additional geofences, and enlarge or shrink geofence based on the activity of the AGV. A user interface with appropriate functionality is created for these operations.

2 Literature Review

2.1 AGV Geofencing

Consider a highly automated and flexible factory that includes AGVs that move inventory and equipment around a factory. Due to the dynamic reconfiguration that may occur in the factory, the AGVs cannot simply follow fixed routes around the factory [2]. The AGVs must choose the safest way to complete a task without colliding with industrial items (such as equipment, machinery, personnel, or other AGVs) or entering forbidden areas. There are three techniques mentioned in [2] that combine to achieve this:

- Localization: (as per the survey of techniques in [6] and [7]) This uses the Internet of Things' (IoT) ubiquitous connection to locate and position IoT devices (tags). This is helpful for setting up safe regions with restricted access (geofencing) in dynamic factories, such as places under repair, where people operate, or where the AGV would be dangerous, such as a spillage. It provides a broad perspective of the AGV and its navigation environment, but it is unable to detect barriers or other potential collision spots (local view).
- Local motion planning: Using object detection [8]. This necessitates the use of sensors installed on the AGV, such as IR proximity sensors, LIDAR, or cameras. This navigation is appropriate for recognizing dense items like movable machinery, boxes, other AGVs, and cars. It creates a high-resolution map of the AGV's near surroundings. Because it only gives a fine-grained local picture of the navigation rather than a global overview, the AGV is unable to avoid dangerous zones or places where humans are operating.
- Global path planning: Typically use a map to create a fixed overview and may need prior knowledge of the layout. It's ideal for huge, immovable objects. The map can be either a) pre-defined and fixed, such as a floorplan, or b) built by the AGV utilizing SLAM "to progressively create a map of this environment while concurrently using this map to compute absolute vehicle location" [9]. This is more dynamic, but it is also slower and more computationally costly. We built a mapless navigation system that navigates using just data from on-board sensors (local data) and assessed its safety in [8].

A combined navigation strategy would overcome the shortcomings of each technique and provide a comprehensive local perspective as well as an overall view of the navigation. We focus on assuring the safety of localization utilizing geofencing to generate an overall picture, to give protection for regions of the factory where humans operate, and to prevent AGV expansion into danger zones as a further step toward establishing an integrated solution. Geofenced zones may be added and deleted as needed (for example, during emergency repairs or where people are working) and are more flexible than maps since comparing an AGV's position to a pre-defined map cannot account for dynamic layout changes. The geofencing may be utilized to give the AGV with an emergency stop ability.

2.2 Communication Framework

AGVs in the manufacturing industry must interact with humans and machines in a seamless manner. This emphasizes the significance of communication between AGVs, machines, and operators, allowing AGVs to function with foresight. It is necessary to select an appropriate communication protocol for IoT devices. Scalability, power consumption, and reliability are the most critical issues to consider.

The author in [10] evaluates the MQTT, CoAP, AMQP, and HTTP IoT messaging protocols. CoAP is slightly favoured when it comes to power consumption [10]. MQTT outperforms CoAP in terms of reliability. The MQTT Quality of Service level (QoS) is used to indicate the reliability. It contains three stages: 0, 1, and 2, which correspond to a message sent at most once, at least once, and exactly once. CoAP does not have explicit QoS levels, although it does include dependability protocols that are akin to QoS 0 and 1. Only the QoS level 2 is appropriate for this thesis since each message sent must be delivered. This indicates CoAP isn't a good fit, but MQTT is more suitable.

2.3 Polygon Offsetting

In this review we will consider on how to deploy enlarging and shrinking of geofence during the runtime of the AGV. Offsetting using a set-theoretic technique, which was first used to construct tool paths for contour parallel pocket machining. The central idea of all these techniques is to create the offset curve piecewise and then connect the pieces in order to arrange the final result. Unfortunately, the post-processing of the pieces is so far from trivial that complex heuristics and ad-hoc procedures are needed [11]. Furthermore, counter-examples that break the algorithms exist more or less often in certain real-world data for most of the published techniques, particularly those for polygon offsetting. Therefore, alternative methods have been investigated.

One of this approaches, presented in [11], uses a Voronoi Diagram (VD) to produce rounded or constant-radius offset curves. VD-based offset curves are strictly parallel to the source curve(as seen in Figure 1): each and every point in the offset curve is exactly at the same distance from the source. Thus, a reflex vertex in the source curve generates a circular arc in the offset curve.

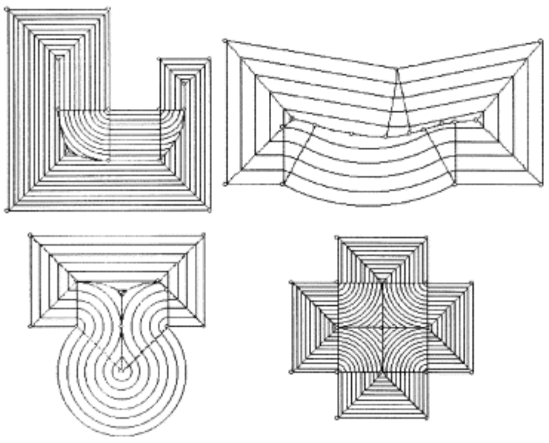


Figure 1: Voronoi Implementation [12]

Another approach has been implicitly described presents a novel geometric structure, called a Straight Skeleton (SK), which is uniquely associated with every simple polygon (possibly multiple connected) and which corresponds to the trace of the vertices of all the offset polygons that can be constructed on the inside. The offset curves traced by a Straight Skeleton are polygons, which means that reflex vertices do not generate circular arcs as in the case of VD-based offsetting. Therefore, such offsetting geometry is usually called mittered, and each offset polygon edge is not at the same distance to the source polygon edges but to the lines supporting such edges [11].

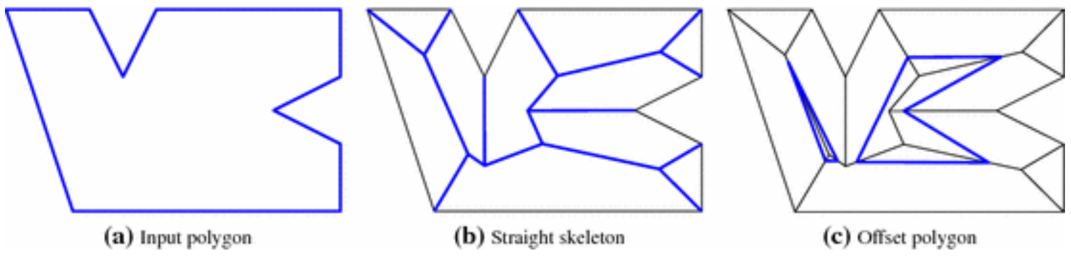


Figure 2: Straight Skeleton of a Polygon [13]

Traditional analytic set-theoretic algorithms are the quickest due to their non-ombinatorial character. The creation of the SK for SK-based offseting takes some time. A actual implementation, however, must be able to deal with inaccurate calculations due to numerical difficulties. One option is to utilize solely quick floating point arithmetic, but to double-check the result and, if necessary, try again after a minor geometric disturbance. As a result, the effective running time for some worst-case data may be large. Hence, we will be implementing a computational geometric algorithm involving distance and bisector calculations.

3 Requirements

3.1 Functional Requirements

Setting up the functional design is the initial stage in designing software. The functional, non-functional, technical, and performance criteria, as well as the look, are described in this section. It serves to define the scope of the project and the features that need be implemented.

- Functional Suitability- Users should be able to submit, edit and delete data.
- Interoperability (Compatibility): the system should be able to connect to different AGV's integrated in the system.
- Message reliability (part of Reliability) - By using an AGV, data is constantly exchanged with the backend. Loss of data could have a big impact on the performance of the system. A suitable communication protocol has to be selected.
- Adaptability (Portability): the measure of how much the software modules can be adapted to other software/hardware and environments.
  The software documentation should be sufficient enough that new users can apply the software to their own system. The interfaces for input and output should be clearly specified. This need is met by a user manual, and an installation guide.
- Scalability - For future purposes the implemented system should be able to massively scale up. Developer may want to integrate new features of the AGV into the geofence system. In order to enforce scalability a lot of parameters could be of influence. This thought has to be kept in mind with all decisions that could affect scalability.

3.2 Hardware Requirements

Hardware required for the thesis are explained in this chapter.

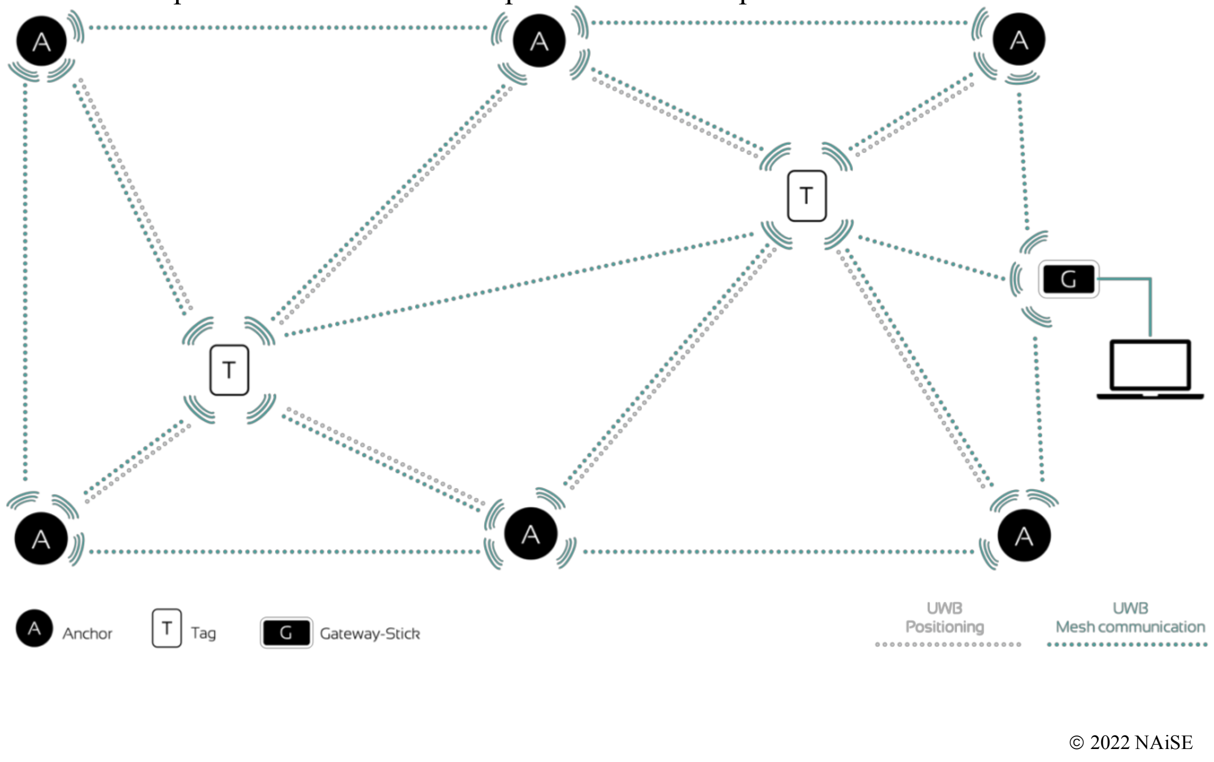
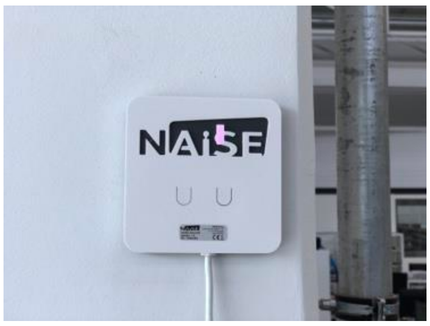


Figure 3: System architecture of the NAiSE Localization Network [14]

The NAiSE Localization Network is shown in Figure 3. The Anchors are the network's infrastructure for localization and communication, and they must be installed in the building. The Anchors allow the Tags to be located and the location data to be passed to the next gateway. The gateway is the interface between the NAiSE localization hardware and the NAiSE software [14].

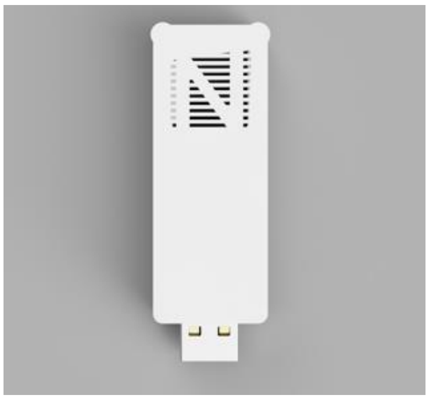
- NAiSE Anchors – These are small sensors that support the NAiSE system's architecture. They allow for the determination of NAiSE Tag locations and a mesh connection between the systems. NAiSE Anchors are easily installed in static situations (walls, ceiling, etc.). They may be readily affixed using industrial adhesive and operated by a Micro-USB interface (5V). The Anchor location should be kept away from metal for optimal effectiveness.



© 2022 NAiSE

Figure 4: NAiSE Anchor [14]

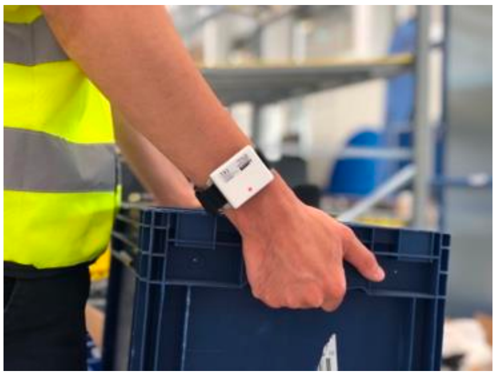
- NAiSE Gateway- It is a versatile and small communication interface. It allows two-way communication between the hardware and the software. It feeds location data from the NAiSE Localization Network to the NAiSE Software and allows the software to communicate with the hardware.



Ⓒ 2022 NAiSE

Figure 5: NAiSE Gateway [14]

- NAiSE Tag- It is a watch-sized distance sensor that monitors the physical distance between individuals (with 10cm precision) and alerts users if they go too near (e.g., 2m). It is self-contained from the anchor infrastructure. The NAiSE Ranger can issue a warning through auditory signal or vibration.



Ⓒ 2022 NAiSE

Figure 6: NAiSE Tag [14]

- Rexroth AGV- Rexroth Active Shuttle is an Automated Guided Vehicle (AGV) which are autonomous robots that move and transport items in production, warehouse, and distribution centres without manual intervention. Sophisticated technology, practical details, perfect interaction of form and function – ActiveShuttle fits seamlessly into existing intralogistics and makes them faster, more efficient and future-proof [15].

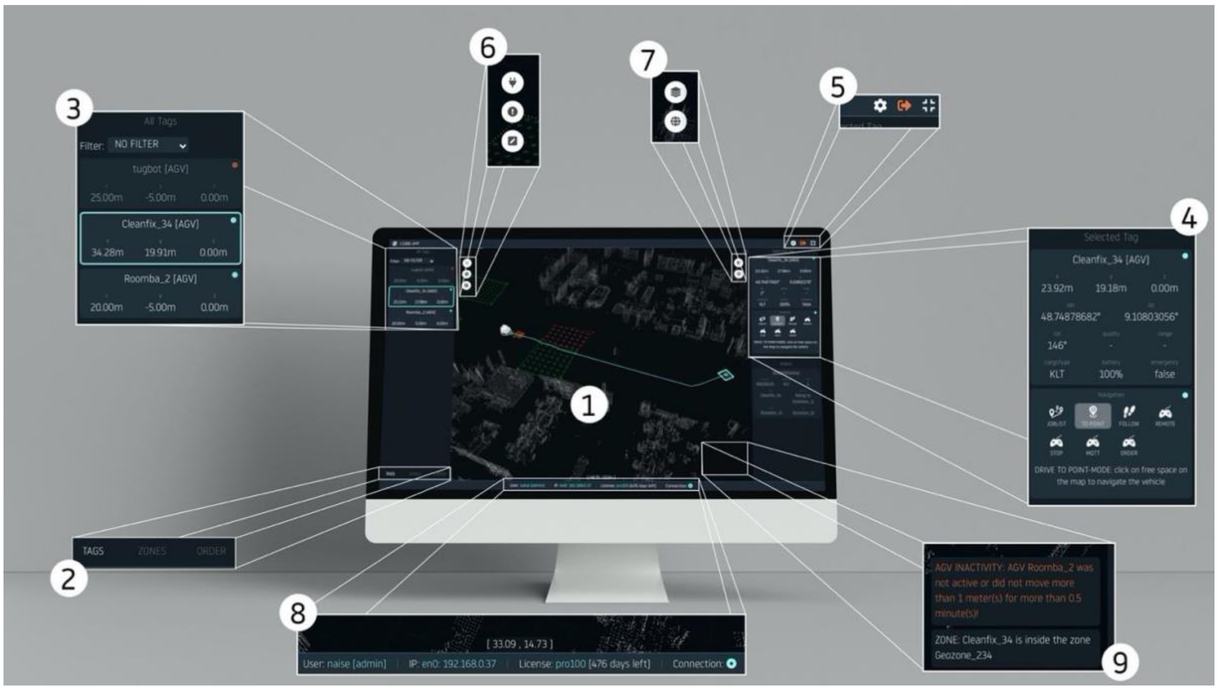
Figure 7: Rexroth AGV [15]

3.3 Software Requirements

Software requirement for this thesis is mentioned in this chapter.

3.3.1 NAiSE Software [14]

To open the GUI with another device (client), you have to type the IP-address of the device, where the server application is running (server). Both devices have to be connected to the same network.



DRIVE TO POINT-MODE: click on free space on the map to navigate the vehicle

[33.09 , 14.73]

User: naise [admin] | IP: en0: 192.168.0.37 | License: pro100 [476 days left] | Connection:

AGV INACTIVITY: AGV Roomba_2 was not active or did not move more than 1 meter(s) for more than 0.5 minute(s)!

ZONE: Cleanfix_34 is inside the zone Geozone_234

© 2022 NAiSE

Figure 8: Main Graphical User Interface of NAiSE Application (NAiSE GUI) [14]

1. In the center of the application, you can see a three-dimensional map of the environment. the tracked objects (Tags, AGVs, forklifts etc.) will be visualized in real-time in their corresponding position on the map.
2. The left panel is separated in three sub menus: Tags, Zones and Orders.
3. In the picture above the Tag sub menu is selected, and the tracked objects are listed. The Zones sub menu provides an overview of the created zones.
4. If you select a Tag from the list of the left panel, a more detailed overview will be displayed in the right panel. In case of an AGV, the control of the AGV (send to a point in the map etc.)
5. The user can open the 'settings', 'log-out' or 'open the GUI in Fullscreen-mode'
6. (From top to bottom): 'Connect/Disconnect from external hardware' (for example AGVS), 'activate the emergency mode' (fake fire alarm, in case of a fire alarm AGVs will stop or leave fire zones immediately) and button for opening the 'map-editor'.
7. (From top to bottom) 'enable/disable the map layers' and 'Setup of the camera view' (2D/3D/center the map).
8. In the footer-panel the server related information are displayed.
9. Map-area messages from the application are located inform the user about changes, warnings and errors.

Using the mapping-mode of NAiSE Application the geofence-areas can be easily defined by the user. The assigned geofence names can be used for triggers (drivers), in which functions for leaving and entering specific geofences can be implemented. Any type of geofence-events will be reported over the defined MQTT-interface.

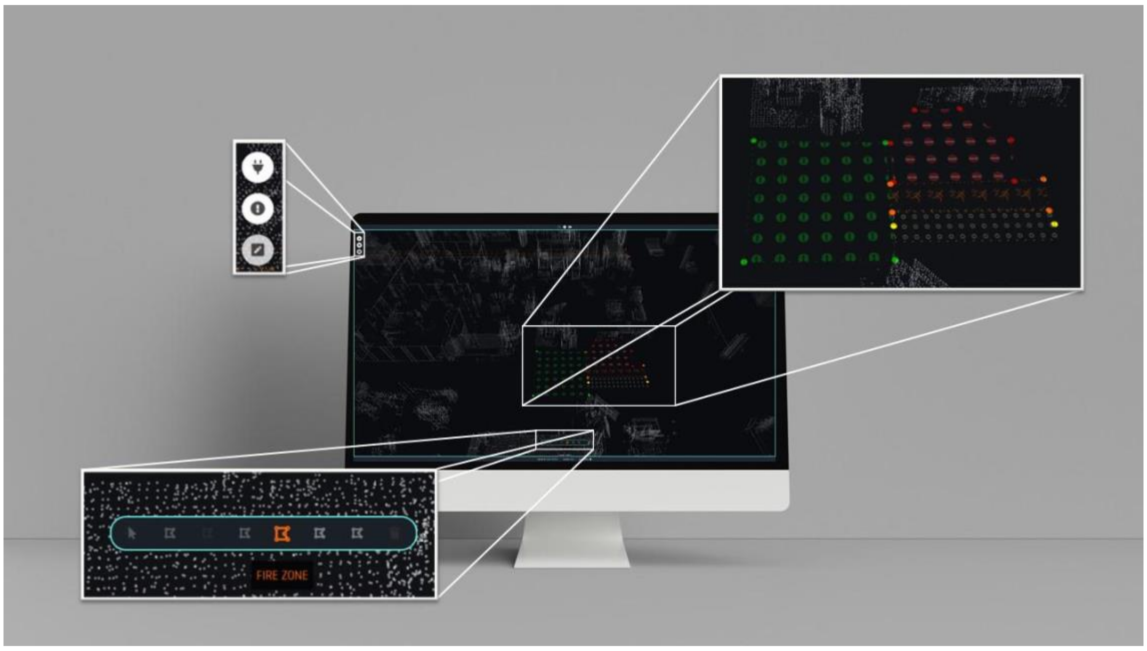


Figure 9: Geofencing interface for creating, editing and deleting geofence-areas [14]

Figure 9 shows the geofence interface in the mapping mode. To access it click on the third button on the top left part of the GUI. The button 2 to 7 on the bottom allows the creation of a different geofencing area. The user can define any number of points for defining a geofence-area. For closing the geofence, you can either click twice in the last point or close the loop by clicking back on the first point. Afterwards a new window will open, in which you may enter the name of the geofence-area. The last button allows the user to delete any geofence-area by clicking on the area of it.

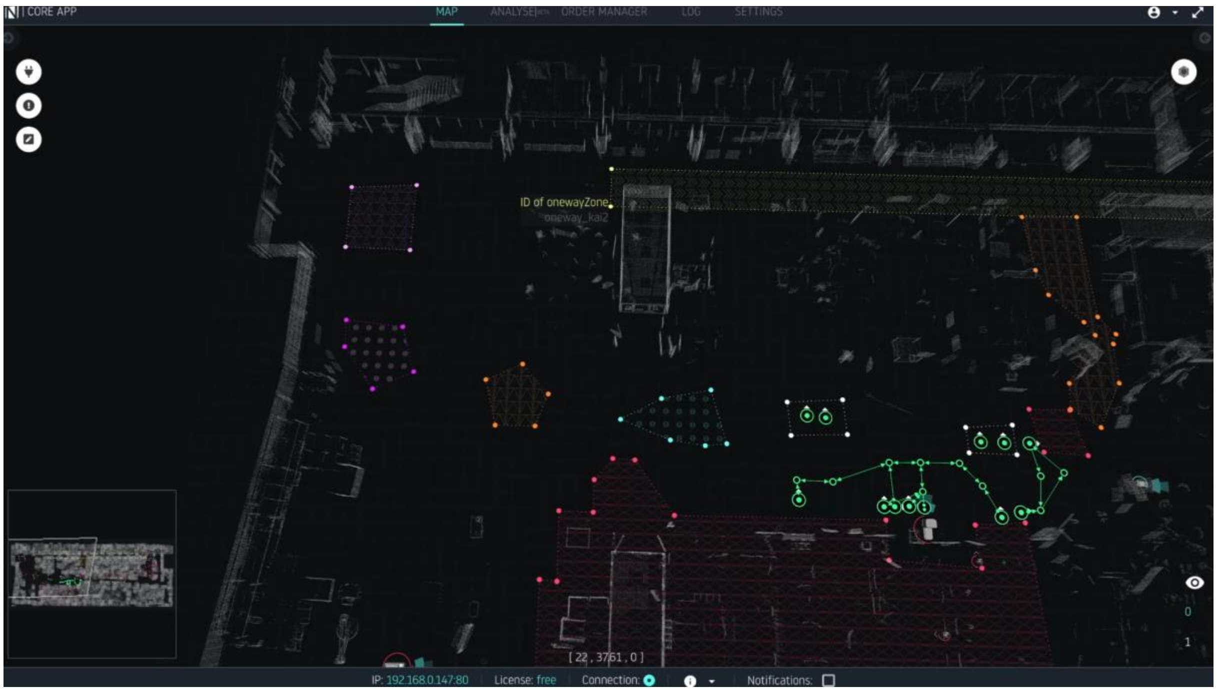


Figure 10: Static Geofences

Figure 10 represents few static geofences that can be drawn on the GUI with different properties as mentioned below:

1. **Geofence Zone**: Sends a message when a Tag enters the zone.
2. **Blocked Zone:** AGVs will not enter this area.
3. **Narrow Zone:** only one Tag or Computed Item can be in this zone. If the AGV's target is in the zone, it will wait at the edge of the zone until the zone is clear.
4. **Fire Zone:** emergency exits and paths must be clear in case of emergency. In case of an alarm, all AGVs leave this zone and stop.
5. **Oneway Zone:** AGVs pass through this zone only in the direction of the arrow
6. **Station**: Can be addressed directly in the order manager.

3.3.2 Mosquitto MQTT

MQTT (also known as Message Queuing Telemetry Transport) is a TCP/IP-based machine-to-machine or “Internet of Things" connection protocol. Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol. Mosquitto is lightweight and is suitable for use on all devices from low power single board computers to full servers. The MQTT protocol provides a lightweight method of carrying out messaging using a publish/subscribe model. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers [16].

All participating modules (MQTT clients) can connect to a broker by utilizing the broker URL and a specified port. MQTT is a publish-and-subscribe messaging protocol that makes use of a message mediator (broker). The customer can subscribe to certain topics after connecting to the broker. If another client publishes (sends a message) on this topic through the broker, it will be received by all subscribing clients. Clients only connect with the broker once. If the client is connected, a keep-alive signal is sent to verify that it is still connected. If not, it is reconnected automatically. This characteristic contributes to smart manufacturing's sustainability.

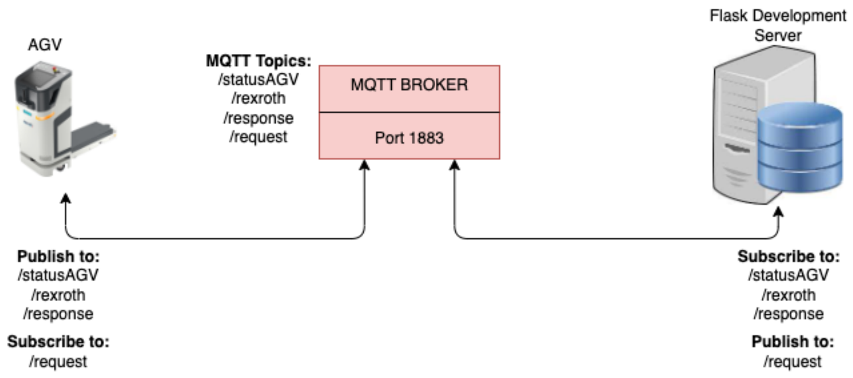

AGV
MQTT Topics: /statusAGV /rexroth /response /request
MQTT BROKER
Port 1883
Flask Development Server
Publish to: /statusAGV /rexroth /response
Subscribe to: /request
Subscribe to: /statusAGV /rexroth /response
Publish to: /request

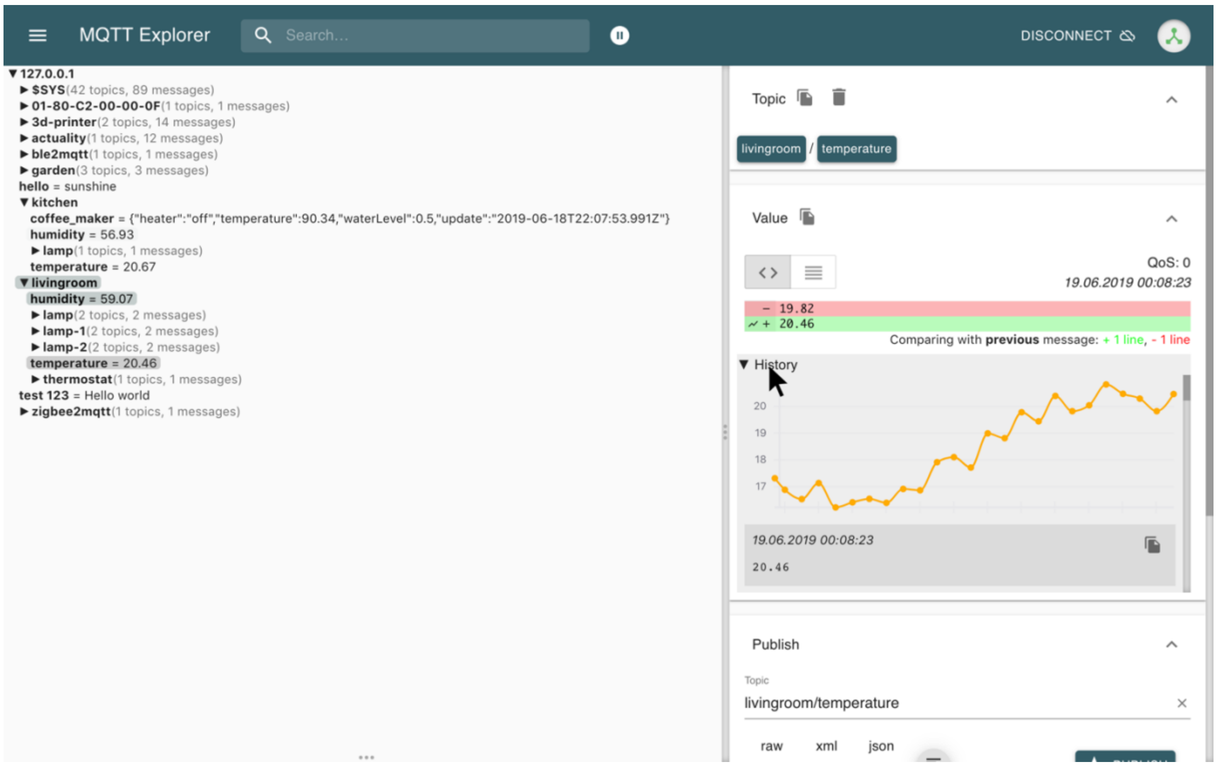
Figure 11: MQTT: a publish/subscribe messaging protocol

3.3.3 MQTT Explorer

MQTT Explorer is a comprehensive MQTT client that provides a structured overview of your MQTT topics and makes working with devices/services on your broker dead-simple [17]. MQTT Explorer is currently a relatively active MQTT client desktop application. Its main features are [18]:

- Basic subscription/push/connection function
- User authentication function
- WebSocket support
- Support diff view and multiple types of Payload
- Basic historical information log
- Support TLS connection
- Support night mode
- In particular, some good features are:
- Automatically subscribe to the topic of $SYS for easy check of broker status information
- Organize the subscription list in a tree structure to facilitate users to view their attribution

- With message visualization function and intuitive and interactive statistical chart design



```
▼127.0.0.1
  ▶$SYS(42 topics, 89 messages)
  ▶01-80-C2-00-00-0F(1 topics, 1 messages)
  ▶3d-printer(2 topics, 14 messages)
  ▶actuality(1 topics, 12 messages)
  ▶ble2mqtt(1 topics, 1 messages)
  ▶garden(3 topics, 3 messages)
  hello = sunshine
  ▼kitchen
    coffee_maker = {"heater":"off","temperature":90.34,"waterLevel":0.5,"update":"2019-06-18T22:07:53.991Z"}
    humidity = 56.93
    ▶lamp(1 topics, 1 messages)
    temperature = 20.67
  ▼livingroom
    humidity = 59.07
    ▶lamp(2 topics, 2 messages)
    ▶lamp-1(2 topics, 2 messages)
    ▶lamp-2(2 topics, 2 messages)
    temperature = 20.46
    ▶thermostat(1 topics, 1 messages)
  test 123 = Hello world
  ▶zigbee2mqtt(1 topics, 1 messages)
```

```
 - 19.82
 + 20.46
Comparing with previous message: + 1 line, - 1 line
```

```
19.06.2019 00:08:23
20.46
```

Topic: livingroom/temperature

Figure 12: MQTT Explorer [17]

The main page is shown in the Figure 12, with the topic search bar and connection configuration at the top. On the lower left of the page, it is the tree structure of the topic, and on the right, it is the Publish column, Subscribe column, Payload column, and History information control column.

3.3.4 Flask Framework

Flask is a microframework for web-based applications. It holds the term microframework because of the absence of required tools and libraries in order to run. Although Flask is a microframework, it is provided with a number of extensions in order to vastly extend its functionalities. Flask sets up a server where you can define specific routes (URL scheme) [19]. Its main features are:

- There is a built-in development server and a fast debugger provided.
- Lightweight
- Secure cookies are supported.
- Request dispatching using REST.
- Support for unit testing is built-in.

Virtual environments are separate collections of Python libraries, one for each project. Installed packages for one project do not affect other projects or the operating system's packages. Python has the venv package, which allows you to build virtual environments.

Flask framework is implemented as a backend where all communication towards the modules is initiated or passed through this module. It saves and organizes data, as well as ensuring that the client-side website runs effectively. Almost all intelligence is housed in this component in the form of an algorithm. In order to let all modules communicate with this centralized component, it is set up as a web-based application running on Python.

All web frameworks including flask need to provide several HTTP methods for data communication. We can specify which HTTP method to be used to handle the requests in the route() function of the Flask class. By default, the requests are handled by the GET() method [20]. The methods are given in the following table:

Table 3.1: HTTP methods

Method	**Description**
GET	It is the most common method which can be used to send data in the unencrypted form to the server.
POST	It is used to send the form data to the server. The server does not cache the data transmitted using the post method.
PUT	It is used to replace all the current representation of the target resource with the uploaded content.
DELETE	It is used to delete all the current representation of the target resource specified in the URL.

To add database functionality to a Flask app, we will use SQLAlchemy. SQLAlchemy is a Python SQL toolkit and object relational mapper (ORM) that lets Python interact with any SQL database system, including MySQL, PostgreSQL, SQLite, and others. An ORM transforms data across systems that are incompatible (object structure in Python, table structure in SQL database). SQLAlchemy is essentially a Python-to-SQL database bridge.

3.3.5 VueJS

VueJS is an open source progressive JavaScript framework used to develop interactive web interfaces. It is one of the most famous frameworks for web development. It can be seamlessly incorporated into large projects for front-end development. It builds on top of standard HTML, CSS and JavaScript, and provides a declarative and component-based programming model that helps you efficiently develop user interfaces, be it simple or complex.

The name of the framework – Vue – is the same phonetically in English as view, and it corresponds to the traditional Model-View-Controller (MVC) architecture. Simply put, view is a UI of an application/website, and the core library of Vue.js focuses the view layer by default [21].

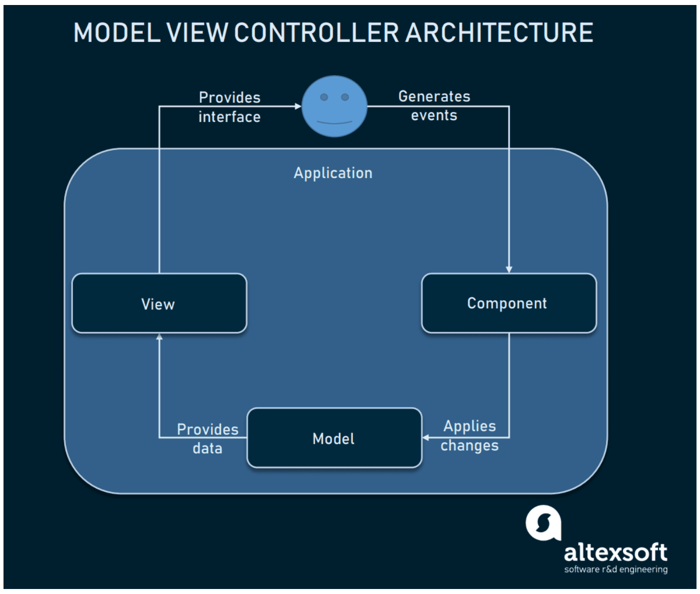


Figure 13: Model View Controller Architecture [21]

4 Concept and Design

This chapter incorporates concepts implemented in the project is covered in this chapter. This chapter explains the use cases that have been developed, how they have been analyzed, and how they have been shown on an extended web interface.

4.1 UML Use case

Use case diagrams are used to define the functioning of a system in terms of its end users. The entire functionalities are depicted in these diagrams. "The use case diagrams provide interactions between roles known as actors and system to achieve a certain goal" [22]. A system boundary is defined for each functionality to indicate where it occurs inside the system. Include and extend relationships are the two types of relationships employed. Include denotes a function that is always called after the origin function. Extend denotes a function that might be called but is not mandatory.

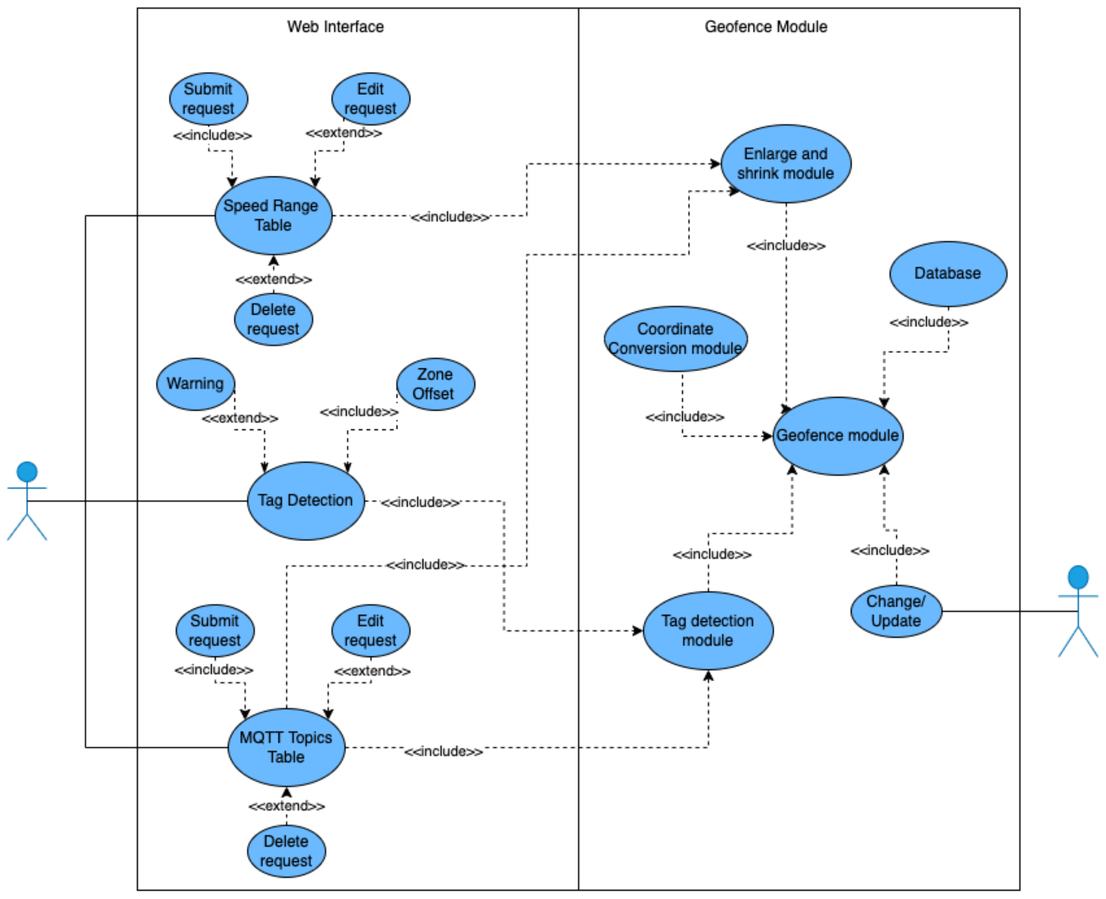


Figure 14:Use case diagram of geofence system

This project extends to build a few features on previously developed geofencing system. The following three use cases have been implemented: Enlarge and shrink of geofence, tag detection and coordinate conversion. Through an expanded web interface, the user can add, edit, and delete i) speed range values with a corresponding offset value, ii) MQTT topics that need to be subscribed iii) zone offset value. The user may acquire the warnings based on the position of the tags in the geofence zones. The developer has direct access to the geofence module and can modify or update it as needed to accommodate new features.

4.2 Coordinate conversion

Importing all the objects into the application (3D map) directly, they would be nested within each other at (0,0,0) at the global origin. We define a position for each object to position them inside a larger world. In global space, the coordinates of all the vertices relative to a global are precisely what they sound like. These coordinates are relative to some global origin of the map, together with many other objects also placed relative to this global originThis is the coordinate space where the objects are transformed so that they are scattered in the global coordinate system. The model matrix is used to translate the object's coordinates from local to global space.

The Figure 15 shows a global coordinate system (X, Y, Z) at the point Q with its corresponding local coordinate system (x*, y*, z*), and the point P and its local coordinate system (x, y, z).

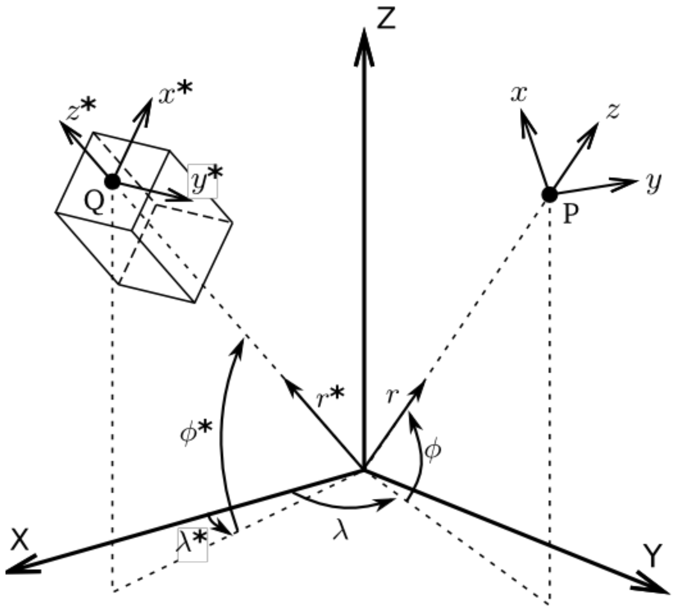


Figure 15: Global and local coordinate system

According to [23] the formula requires that position of P in the global coordinate system (X1, Y1,θ) here θ is the orientation relative to x axis. Using the matrices to convert the position of Q from its global coordinates (X, Y) to a local coordinate (x*, y*).

$$R=\begin{bmatrix}\cos\theta & -sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}$$

R is a counterclockwise rotation matrix

$$\vec{P}= (X1, Y1)$$

$$\vec{Q}_{\text{global}} = (X, Y)$$

$$\vec{Q}_{\text{local}} = (x^*, y^*)$$

To convert from global coordinates to local of point Q in relative to the global point P.

$$\vec{Q}_{\text{local}} = R^{-1} \times (\vec{Q}_{\text{global}} - \vec{P})$$

(where R^{-1} is the inverse of the matrix R)

To convert from local coordinates to global of point P in relative to the global point of P

$$\vec{Q}_{\text{global}} = \mathrm{R} \times (\vec{Q}_{\text{local}} + \vec{P})$$

4.3 Polygon Offsetting Algorithm

Offsetting is a fundamental geometric issue that is used in a variety of applications. Even though it is conceptually simple to establish an offset geometry of a geometric structure in any dimension, computing a precise offset curve efficiently, even in a plane, is far from simple [12].The application for offsetting is to enlarge and shrink the geofence which is a polygon of n number of points.

We need an algorithm that does the following operations in order to offset a polygon with known vertices. The first step is to replicate each vertex. Then, in counterclockwise sequence, we connect each new vertex to the previous vertex. Presented in this sub-chapter is a simple but efficient algorithm to generate an offset of a polygon.

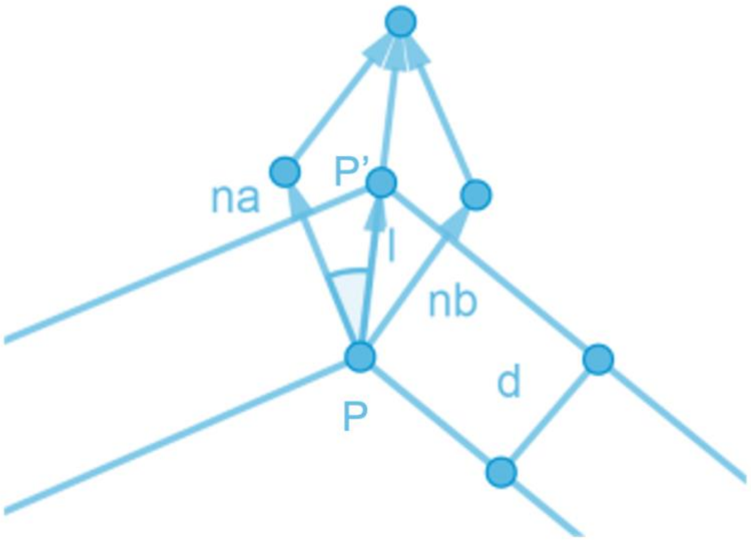


Figure 16: Polygon offsetting

- For every polygon, first count the number of points.
- Calculate the outer bisector vector for each polygon vertex as the sum of the normalized normals a and b of two adjacent edges, then normalize it.

$$a = \mathrm{normalized(vnX)} \times \mathrm{normalized(vpX)}$$

$$b = \mathrm{normalized(vnY)} \times \mathrm{normalized(vpY)}$$

$$bis = a + b$$

- Then find needed length of bisector to provide offset distance as (d=offset value)

$$L = \frac{d}{sqrt(1 + dot\ product(a, b))}$$

- Get offset polygon vertex (use minus for shrinking polygon)

$$\mathrm{P' = P + L \times bis}$$

5 Implementation

The concepts that have been implemented are thoroughly explored in a technical manner. First, the software architecture is described, along with existing localization system. Second, the frontend and backend implementation and sequence flow is explained.

5.1 System Architecture

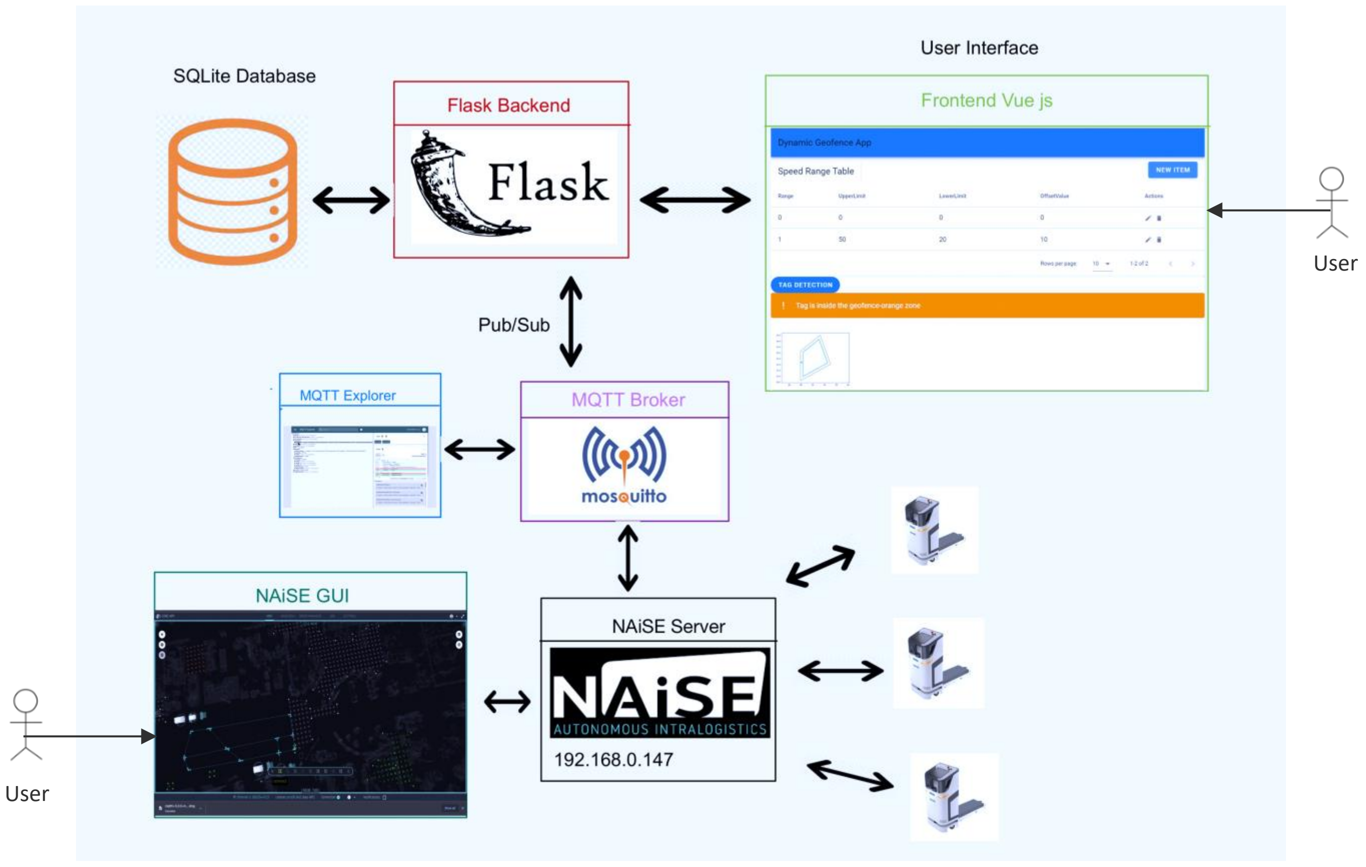


Figure 17:System Architecture

In Figure 17, the implemented architecture can be seen with all modules defined. The backend module is implemented in Flask python framework. All the communication towards the remaining modules is initiated through this module. The database used is SQLite as a backend database. The client side (User interface) is implemented on Vuejs. The user input from the frontend is communicated to the SQLite database using the library SQLAchemy which enables Object-Relational Mapping (ORM). ORM is used for data conversion between two incompatible applications. This practically means that the operations done on SQLite data base can be initiated from flask using Python queries. This speeds up the process of converting SQL queries into python. NAiSE GUI rendered from NAiSE server which provides as 3D map of the ARENA with real-time movements of the AGV’s, Tags and with a feature to draw polygon shaped geofences. The AGV’s (Rexroth active shuttle) communicate to the NAiSE server, an MQTT broker is implemented as communication protocol to enable bidirectional communication.

5.2 Frontend

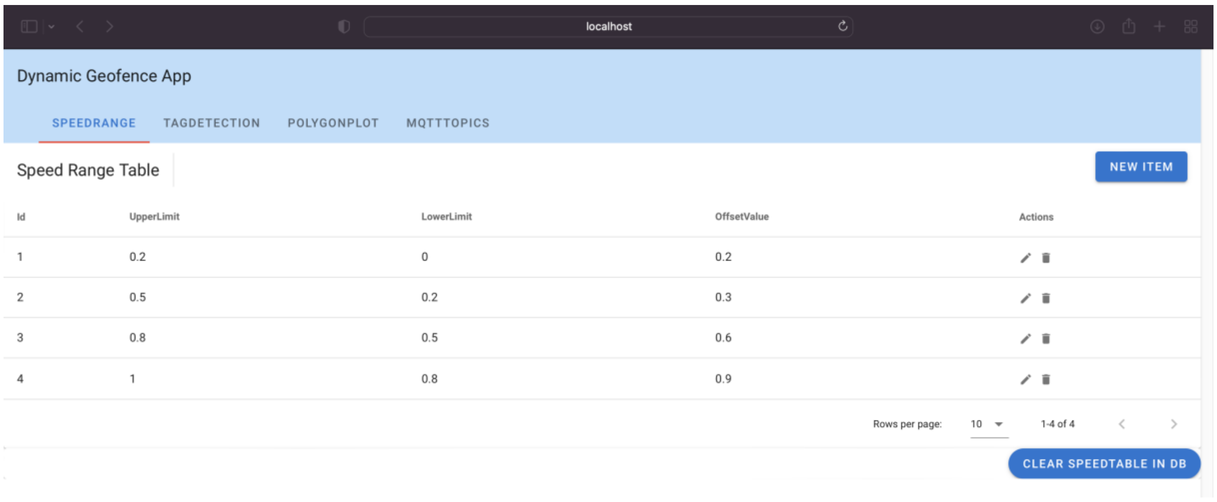
Vue.js framework is utilized to implement the frontend. HTML, CSS, and JavaScript are the three most common front-end technologies. Its structured standard language is HTML. Its standard language is CSS. The standard behavior language is JavaScript. The primary concept is to separate the structure, design, and behavior of web pages in order to match the Web standard, hence there are three basic languages.

Vue.js makes use of an HTML-based template syntax to link the displayed DOM (Document Object Model) to the data of the specified Vue instance. The <script> part handles all back-end tasks such as API calls, event handling, database connectivity, server-side methods, and so on. The styling of the component is created with CSS or any preprocessor in the <style> section.

5.2.1 User interface Design

The user interface is divided in three components:

- Speed Range Table – Figure 18 shows the speed range table. User has the flexibility to add in the speed values ranging from 0 to 1. This speed value corresponds to the speed of the AGV.



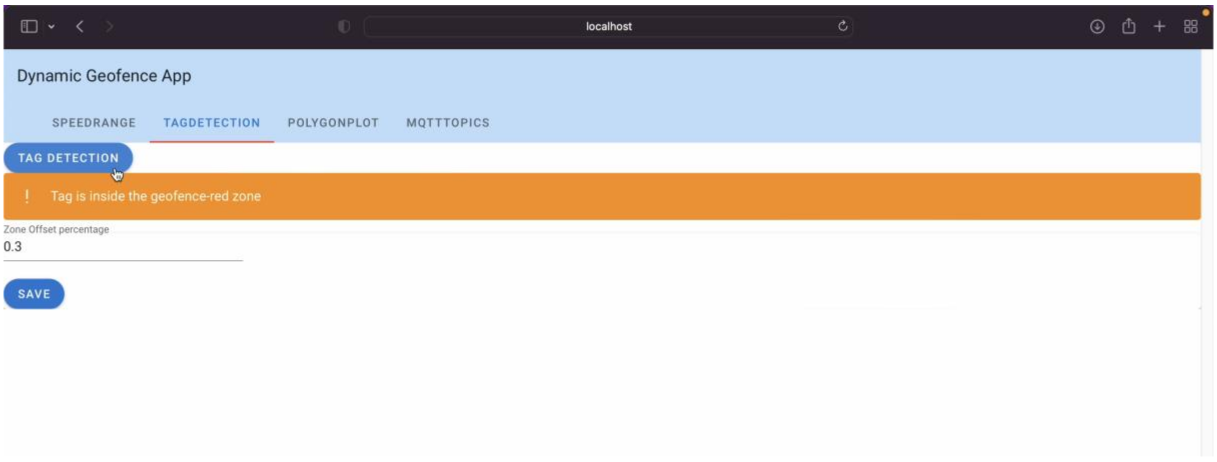
Id	UpperLimit	LowerLimit	OffsetValue	Actions
1	0.2	0	0.2	
2	0.5	0.2	0.3	
3	0.8	0.5	0.6	
4	1	0.8	0.9	

Figure 18: Speed Range Table user interface

Table 5.1: Speed range table description

Column	Description
Id	Unique identifier for each row. There could be at the most 10 rows.
Upper Limit	Upper range of the AGV's speed
Lower Limit	Lower range of the AGV's speed
Offset Value	The geofence offset. If the offset value is positive then the geofence is enlarged, If the value is negative then the geofence is shrunk.
Actions	User can either edit or delete the respective row.

- Tag Detection – Figure 19 shows the tag detection UI. User has the ability to add zone offset value ranging from 0 to 1 and a warning bar to notify which zone the tag is located.



! Tag is inside the geofence-red zone

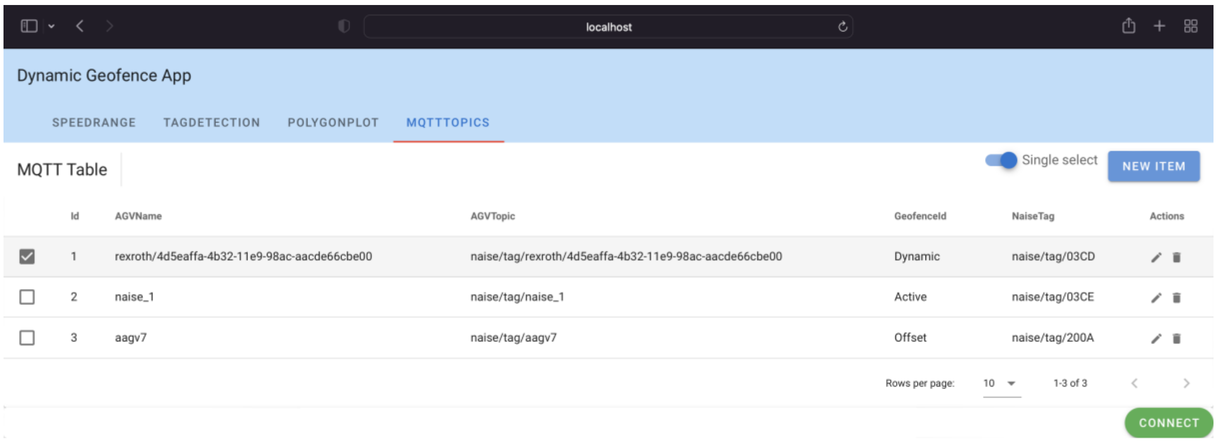
Zone Offset percentage
0.3

Figure 19:Tag Detection user interface

Table 5.2: Tag detection UI description

UI components	Description
Tag Detection	It is an "on click" button to notify the user regarding the tag position
Warning Bar	This bar gives the warning notification on click of the tag detection button. There area three warning: 1. Tag is inside the geofence-red zone 2. Tag is inside the geofence-blue zone 3. Tag is outside the geofence
Zone offset percentage	Percentage value of the geofence divded into zones

- MQTT Topics – Figure 20 shows the MQTT topics table. The user can add mqtt topics that needs to be subscribed at the backend. This gives the user flexibility to connected to any AGV and tag in the system. This data is also stored in the database, so the user can click on the select box to select the respective row.



	Id	AGVName	AGVTopic	GeofenceId	NaiseTag	Actions
✓	1	rexroth/4d5eaffa-4b32-11e9-98ac-aacde66cbe00	naise/tag/rexroth/4d5eaffa-4b32-11e9-98ac-aacde66cbe00	Dynamic	naise/tag/03CD	
☐	2	naise_1	naise/tag/naise_1	Active	naise/tag/03CE	
☐	3	aagv7	naise/tag/aagv7	Offset	naise/tag/200A	

Rows per page: 10 1-3 of 3

Figure 20: Mqtt topics table

Table 5.3: MQTT Topics UI description

UI components	**Description**
Select box	User can select the row based on the topics that needs to be subscribed during run time
AGV Name	Name of the AGV as in NAiSE GUI
AGV Topic	MQTT topic to connected to the AGV and receive payload
Geofence Id	Name of the ID entered by the user in NAiSE GUI
Naise Tag	MQTT topic to connect to the NAiSE tag and receive real-time location
Actions	User can either edit or delete the respective row.
Connect	To connect to the MQTT broker

5.3 Backend

Flask framework is utilized to implement the logic of the application. It contains all the intelligence in the form of algorithms. This centralized component communicates with all the modules. Flask runs the application locally at the address http://127.0.0.1:5000/. In Flask, routes are the primary interface for the application or the end user, they are frequently built using the URL pattern. The route may be built directly in the decorator by using the normal URL pattern for the name. In a Flask application, adding a route is simple.

Table 5.4: Flask Routing

URL routes	**Methods**	**Description**
"/addspeeddata"	POST	Takes the data from the Speed Range table in frontend and adds it to the database.
"/editspeed"	PUT	Takes an edited speedrange as json object from the front end and updates the database with the new values.
"/deletespeed"	DELETE	It deletes a row from the database based on the id passed in the request.
"/getSpeedTabledata"	GET	It queries the database for all the data in the SpeedRange table
"/Clearspeedrangetable"	DELETE	It deletes all the rows in the SpeedRange table.
"/tablemqtt"	POST	It takes the MQTT topics as JSON object from a POST request triggered by frontend, and then adds it to a database table
"/editmqtt"	PUT	It takes a edited row from MQTT table as json object from the front end, and updates the database with the new values.
'/mqttselectedrow'	POST	ID selected in the user interface is passed in the request and query functions for the MQTT topics are called here.
"/deletemqtt"	DELETE	It deletes a row from the database based on the id passed in the request
"/getMqttTabledata"	GET	It queries the MqttTopics table in the database.
"/getZoneOffset"	POST	It takes a value from a javascript function and stores it in a global variable.
"/Connect"	POST	It connects to the MQTT broker and starts the application.
'/TagDetection'	GET	It sends the warning to the user interface.

5.3.1 Backend-MQTT communication

The communication sequences are the key emphasis when it comes to system validation. The communication validation is very crucial between the AGV and the Backend. The MQTT protocol is used in this connection. MQTT messages play an important role in inter-modular communication, as shown in the UML Sequence diagrams in Figure 21.

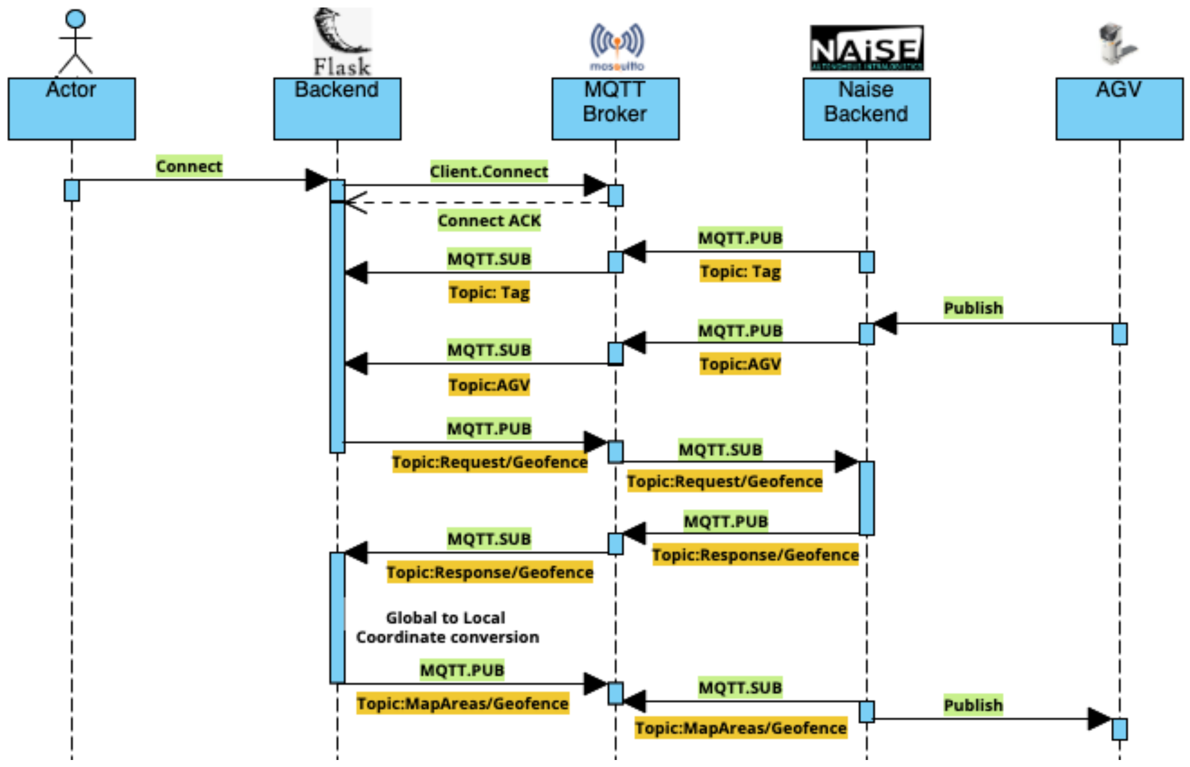
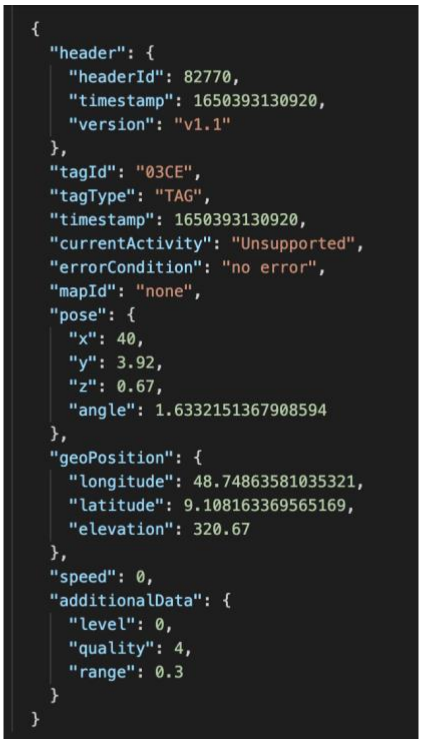


Figure 21: Backend-MQTT sequence diagram

- Upon receiving a connect trigger from the user interface to the backend. The MQTT client connection is initiated. A unique client ID is assigned, and a connection request is sent to the MQTT broker. An acknowledgment is received stating that the client is connected to the broker.
- The tag position and AGV position, speed is published to the respective topics from the NAiSE server. At the flask backend, these MQTT topics are subscribe and the messages are decoded.
- The geofence drawn by the user on NAiSE GUI is forwarded to the NAiSE backend. At the flask backend a topic is published requesting for the coordinates of the geofence on the topic “Request/geofence”. On the topic “Response/Geofence” the coordinates of the geofence are published by the NAiSE backend.
- The geofence coordinates are converted from global to local and is made relative to the AGV and published in the topic “MapAreas/Geofence”.

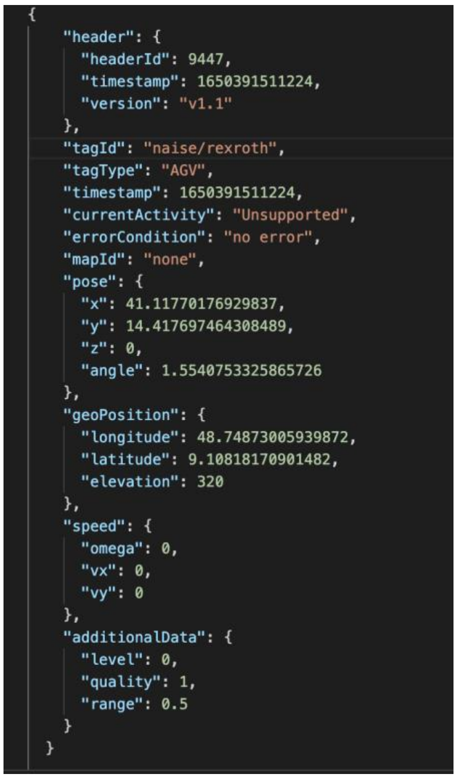
- Subscribing to Naise Tag to get the global position of the person wearing the tag.
  Topic: naise/tag/03CE



```
{
  "header": {
    "headerId": 82770,
    "timestamp": 1650393130920,
    "version": "v1.1"
  },
  "tagId": "03CE",
  "tagType": "TAG",
  "timestamp": 1650393130920,
  "currentActivity": "Unsupported",
  "errorCondition": "no error",
  "mapId": "none",
  "pose": {
    "x": 40,
    "y": 3.92,
    "z": 0.67,
    "angle": 1.6332151367908594
  },
  "geoPosition": {
    "longitude": 48.74863581035321,
    "latitude": 9.108163369565169,
    "elevation": 320.67
  },
  "speed": 0,
  "additionalData": {
    "level": 0,
    "quality": 4,
    "range": 0.3
  }
}
```

Figure 22: Tag message payload

- Subscribing to naise Rexroth AGV to get the global position of the AGV.
  Topic: naise/AGV/Rexroth

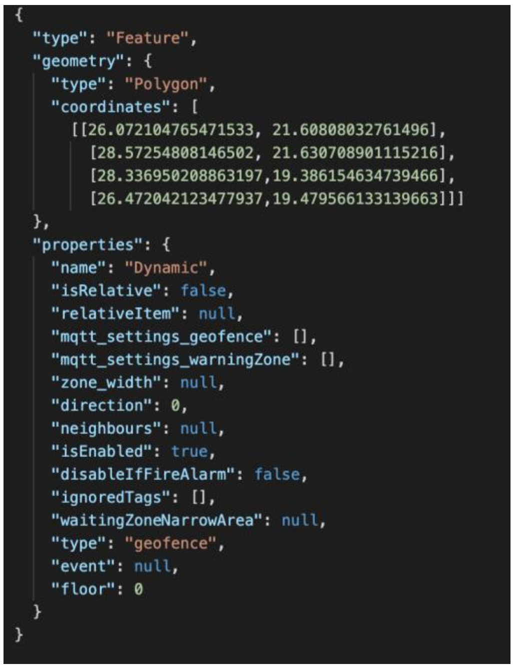


```
{
    "header": {
      "headerId": 9447,
      "timestamp": 1650391511224,
      "version": "v1.1"
    },
    "tagId": "naise/rexroth",
    "tagType": "AGV",
    "timestamp": 1650391511224,
    "currentActivity": "Unsupported",
    "errorCondition": "no error",
    "mapId": "none",
    "pose": {
      "x": 41.11770176929837,
      "y": 14.417697464308489,
      "z": 0,
      "angle": 1.5540753325865726
    },
    "geoPosition": {
      "longitude": 48.74873005939872,
      "latitude": 9.10818170901482,
      "elevation": 320
    },
    "speed": {
      "omega": 0,
      "vx": 0,
      "vy": 0
    },
    "additionalData": {
      "level": 0,
      "quality": 1,
      "range": 0.5
    }
  }
```

Figure 23: AGV message payload

- Sending request to Naise server to get geofence global coordinates.
  Topic: naise/request/geofence
  Payload: {"id": "Dynamic"}

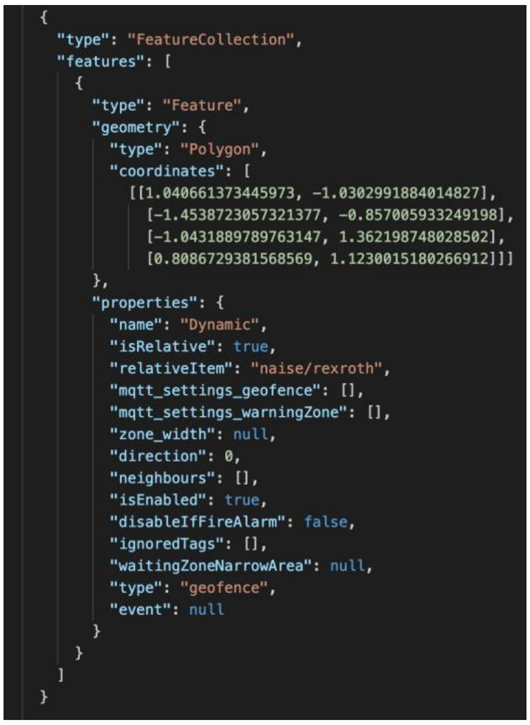
- Recieving global coordinates of the request id of geofence from Naise serve
  Topic: naise/response/geofence



```json
{
  "type": "Feature",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [[26.072104765471533, 21.60808032761496],
        [28.57254808146502, 21.630708901115216],
        [28.336950208863197,19.386154634739466],
        [26.472042123477937,19.479566133139663]]]
  },
  "properties": {
    "name": "Dynamic",
    "isRelative": false,
    "relativeItem": null,
    "mqtt_settings_geofence": [],
    "mqtt_settings_warningZone": [],
    "zone_width": null,
    "direction": 0,
    "neighbours": null,
    "isEnabled": true,
    "disableIfFireAlarm": false,
    "ignoredTags": [],
    "waitingZoneNarrowArea": null,
    "type": "geofence",
    "event": null,
    "floor": 0
  }
}
```

Figure 24:Geofence Message payload

- To make the geofence relative to an AGV by publishing local coordinates
  Topic: MapAreas/geofence



```json
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [[1.040661373445973, -1.0302991884014827],
            [-1.4538723057321377, -0.857005933249198],
            [-1.0431889789763147, 1.362198748028502],
            [0.8086729381568569, 1.1230015180266912]]]
      },
      "properties": {
        "name": "Dynamic",
        "isRelative": true,
        "relativeItem": "naise/rexroth",
        "mqtt_settings_geofence": [],
        "mqtt_settings_warningZone": [],
        "zone_width": null,
        "direction": 0,
        "neighbours": [],
        "isEnabled": true,
        "disableIfFireAlarm": false,
        "ignoredTags": [],
        "waitingZoneNarrowArea": null,
        "type": "geofence",
        "event": null
      }
    }
  ]
}
```

Figure 25: Local coordinates message payload

5.3.2 Enlarge Shrink Sequence

The UML diagram with a complete sequence flow of enlarging or shrinking a geofence during runtime can seen in the Figure 26.

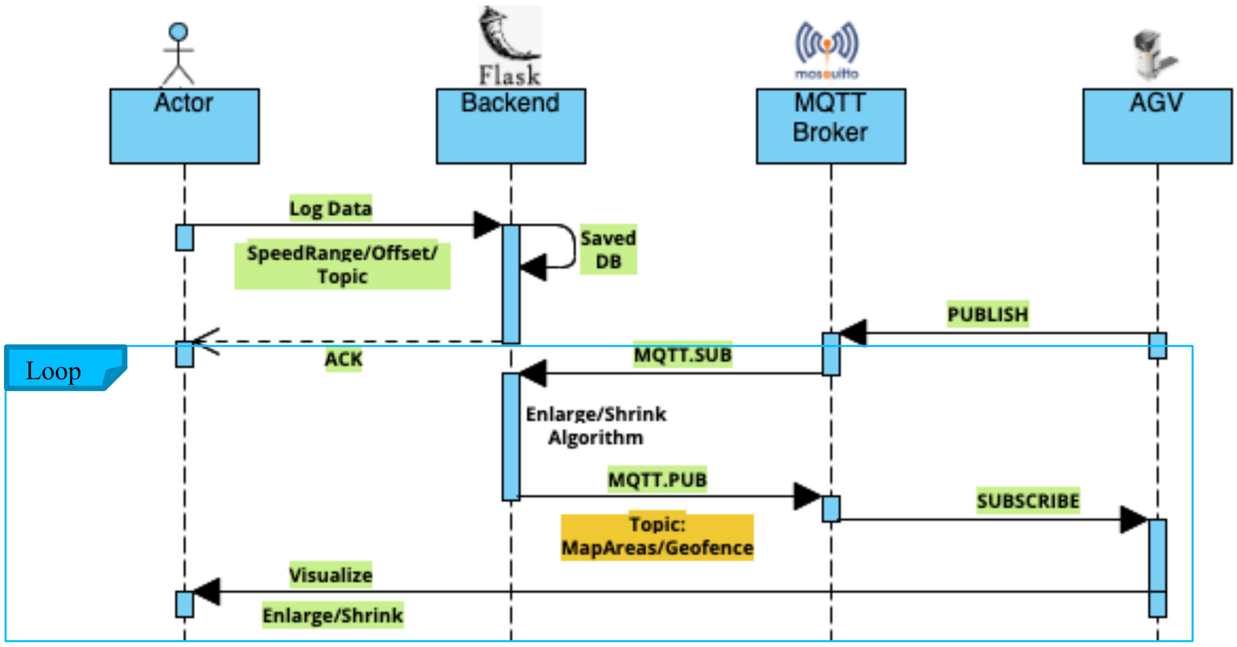


Figure 26:Enlarge and shrink sequence diagram

- The user logs the data in the speed range table, and the MQTT topics that he wishes to subscribe. All this data is saved in the SQLite database.
- At the flask backend, real-time location and speed of the AGV is decoded from the MQTT message payload. Based on the users speed range and respective offset value, the geofence around the AGV is either enlarged or shrunk by running through the Enlarge/Shrink algorithm and new set of coordinates are generated.
- As seen in the figure above, Enlarge/Shrink Algorithm is in an endless loop that repeatedly gets the real-time speed of the AGV until the program is interrupted.
- To the topic MapAreas/Geofence, new set of coordinates are published every 2secs that was generated from the algorithm.
- The user can visualize the offsetting of the geofence in the NAiSE GUI.

5.3.3 Tag Detection Sequence

The UML diagram in Figure 27 shows a complete sequence of Tag detection algorithm during the runtime.

- The user logs Zone offset value in the frontend.
- At the flask backend, realtime location of the TAG and the AGV is decoded from the MQTT message payload. The values are then utilized in the tag detection algorthim which is running in an endless loop until the program is interrupted.
- On click of the button “TAG DETECTION” in the frontend, a warning is displayed to the user which shows in which zone the tag is located in the geofence.

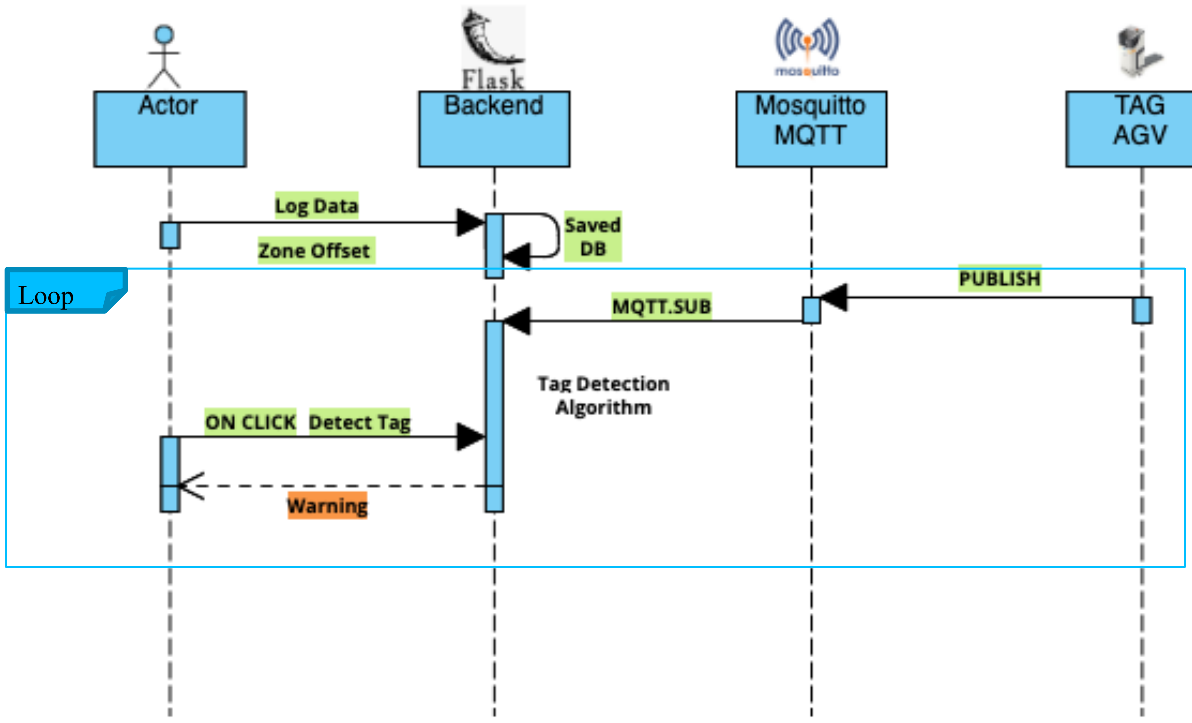


Figure 27: Tag Detection sequence diagram

5.3.4 Flow of Execution

In this section, the flow of the program execution and the method implemented to run parallelly the two new dynamic geofence features (enlarge/shrink, tag detection) will be briefly explained.

The two algorithms as mention in the section 5.3.2 and 5.3.3 have different dynamicity factors affecting the geofence. Enlarge and shrink algorithm, varies the geofence coordinates by redrawing the original shape of the geofence. In Tag detection algorithm, the geofence is divided into multiple zone areas maintaining the geofence's geometry. As seen in the sequence diagrams in Figure 26 and Figure 27, both the algorithms are running in endless loops, independent of each other. Since the flask framework is written in python and one of python's main weakness is its inability to have true parallelization due to global interpreter lock. Some functions, however, can still virtually run in parallel [24]. Multithreading process is used to deal with this problem.

A thread is a set of instructions that are executed sequentially within a process. Multiple threads can be created by a single process, but they will all share the same memory. Running many threads is equivalent to running multiple applications at the same time, but with the following advantages:

- Multiple threads inside a process share the same data space as the main thread, making it easier for them to share information and communicate than if they were independent processes.
- Threads are sometimes known as light-weight processes since they consume less memory and are less expensive than processes.

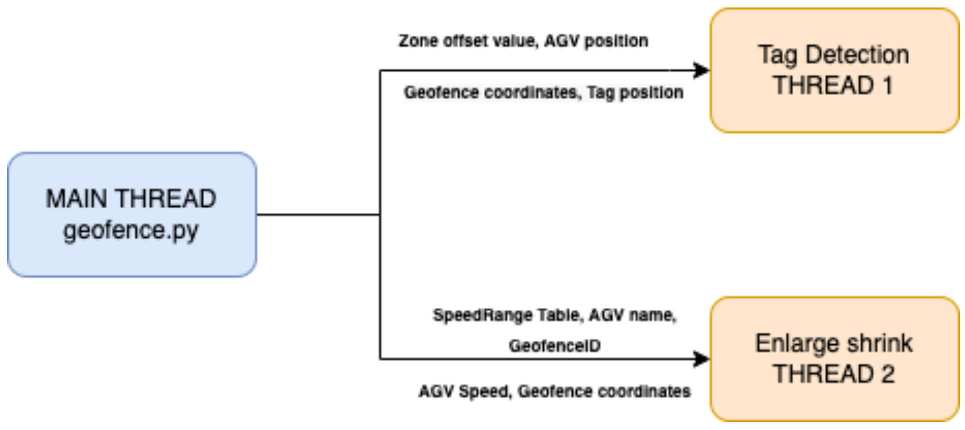

Zone offset value, AGV position
Geofence coordinates, Tag position
Tag Detection THREAD 1
MAIN THREAD geofence.py
SpeedRange Table, AGV name, GeofenceID
AGV Speed, Geofence coordinates
Enlarge shrink THREAD 2

Figure 28: Multithreaded Program

Figure 28 represents a graphical overview of the multithread program. In Python, threads can be implemented with the use of threading module in the main program (main thread). The program has two threads, each of which work on real-time data and user defined data from the database. In THREAD1, tag detection function is passed as target. A list containing the arguments to the function are Zoneoffset value from the frontend, current position of AGV and Tag as well as geofence coordinates from MQTT are passed in this thread. In THREAD2, enlarge shrink function is passed as target. A list containing the arguments to the function are SpeedRange table, AGV name, geofenceID from the frontend, current speed of AGV and geofence coordinates from MQTT are passed in this thread. Multithreading is an excellent alternative for running both of these threads at the same time and having both functionalities during the application's run time.

6 Results

The Dynamic geofence app is created to introduce dynamic functionality to the already existing geofence system built by NAiSE. The application works in perfect synergy with the NAiSE system. When the user executes the backend and frontend, it generates a web page for the user to add relevant data and to monitor the tag detection warnings as describe in section 5. The backend is running on http://127.0.0.1:5000 , frontend on http://localhost:8080 and NAiSE GUI on http://192.168.0.147/gui/. Once the application is successfully running, both the features i.e. Tag Detection and Enlarge/Shrink of geofence can be visualized in the NAiSE GUI.

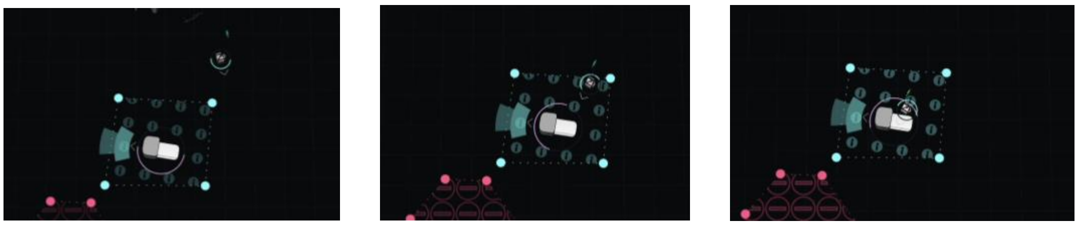
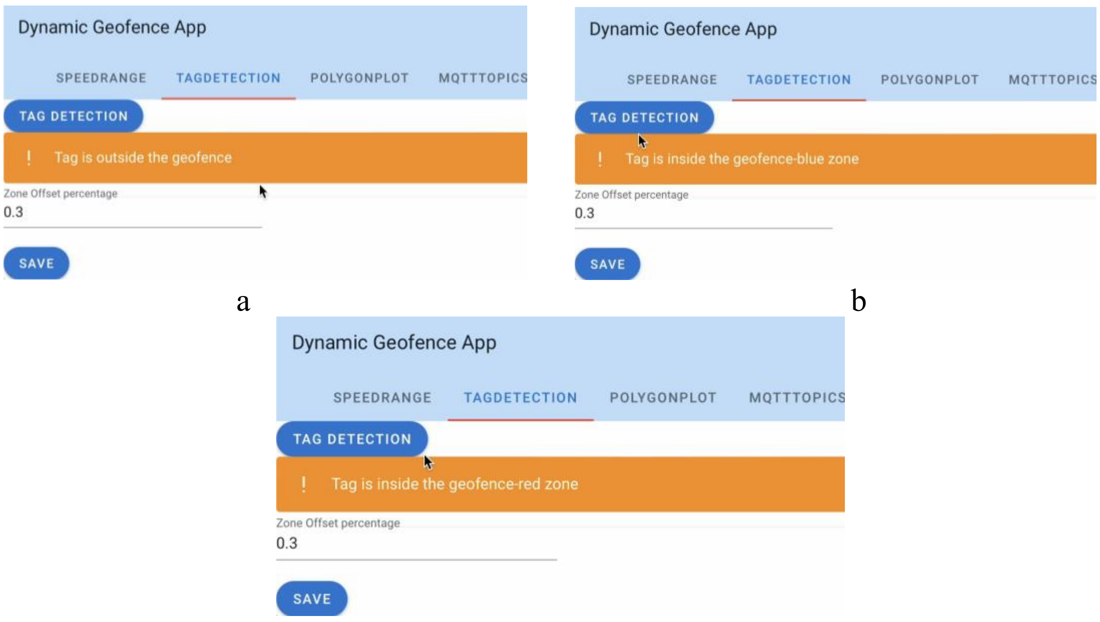


Figure 29: Tag detection a) Tag is outside the geofence b) Tag is in the blue zone of the geofence c) Tag is in the red zone of the geofence

Figure 29 depicts the Tag detection function, which divides the geofence surrounding the AGV into two zones: blue and red based on user defined zoneoffset percentage. Depending on the tag's real-time position, it determines which zone it is in and delivers a warning to the frontend as seen in Figure 30.



! Tag is outside the geofence

Zone Offset percentage
0.3

! Tag is inside the geofence-blue zone

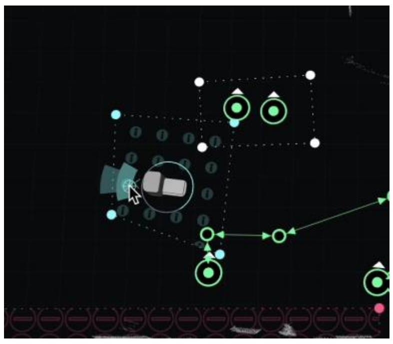
Zone Offset percentage
0.3

! Tag is inside the geofence-red zone

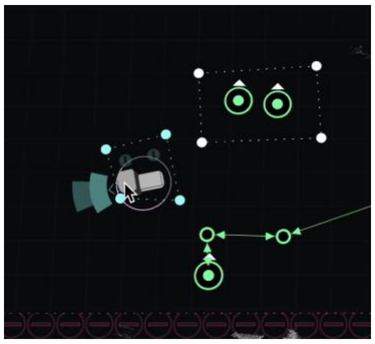
Zone Offset percentage
0.3

Figure 30: Warning a) Tag is outside the geofence b) Tag is in the blue zone of the geofence c) Tag is in the red zone of the geofence

Enlarging and shrinking of geofence takes place as the speed of the AGV is varied as seen in Figure 31. The data from the speedrange table is retrieved from the database at the backend, and the appropriate offset value is applied based on the AGV speed.



a



b

Figure 31: Enlarge/Shrink a) Geofence is enlarged as AGV speed is high b) Geofence shrunk as AGV speed is less

7 Conclusion & Future Work

The goal of this research project was to analyze and implement dynamic geofences in industries. Geofencing technology may be applied to both static and dynamic mapped objects. Because the geometry of the geofence is not limited, any closed polygon with any number of points may be drawn. On an existing localization system, an application is designed and constructed to establish dynamic protection zones utilizing geofence.

This research proposes a comprehensive communication architecture in which algorithms can be evaluated in an AGV capable of bidirectional communication. The objective and scope of the project are identified first, followed by research into communication frameworks. The expertise gathered was utilized and translated to match the project's multi-tenant nature. The resultant framework is created by combining different modules (User Interface, Database, Geofence Module, AGV, Tag, and the MQTT communication protocol) and allowing them to communicate with one another via communication interfaces. The dynamic geofence application consists of two modules: Enlarge/Shrink and Tag Detection were implemented to divide the geofence into zones and to analyze the accuracy of tag detection inside a zone. We also examined the speed factor of the AGV to perform dynamic geofencing. We can also draw the conclusion that the previously constructed localization system may be enhanced by adding new dynamic features utilized for a more enhanced automation of indoor logistics.

With regards to this project, further development can be carried out as follows:

- Future work will include investigating other factors that impact AGV performance and safety, as well as adding more dynamicity to the geofence.
- To implement an emergency stop on the AGV when a tag is detected in the red zone and reduce the AGV's speed when a tag is detected in the blue zone.
- To develop object detection for the AGV so that it can recognize persons and industrial signs, as well as build a geofence around the discovered item and transmit it to other AGVs in the system.
- Other alternatives must be researched and implemented to eliminate any sort of delay caused by multithreading limitations.

Bibliography

[1] Fraunhofer Institute for Integrated Circuits IIS, "5G Positioning in Industry and Logistics 4.0," [Online]. Available: https://www.iis.fraunhofer.de/en/ff/lv/lok/5g/industrie_logistik/ueberwachung.html. [Accessed 25 April 2022].

[2] R. J. I. Victoria.hodge, "Analysing Ultra-Wide Band Positioning for Geofencing in a Safety Assurance Context," ResearchGate.Net, 2022.

[3] I. H. T. L. J. M. P. M. a. Z. P. S. Burton, "Mind the gaps: Assuring the safety of autonomous systems from an engineering, ethical, and legal perspective," Artificial Intelligence, vol. 279, 2020.

[4] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Geo-fence. [Accessed April 2022].

[5] M. A. Javed, F. U. Muram, A. Fattouh and a. S. Punnekkat, "Enforcing Geofences for Managing Automated Transportation Risks in Production Sites," Researchgate.net, 2020.

[6] V. J. Hodge, "Sensors and data in mobile robotics for localisation," Encyclopedia of Data Science and Machine Learning, IGI Global, 2022.

[7] A. G. a. K. K. L. F. Zafari, "A survey of indoor localization systems and technologies,," IEEE Communications Surveys & Tutorials, 2019.

[8] R. H. a. R. A. V. J. Hodge, "Deep reinforcement learning for drone navigation using sensor data," Neural Computing and Applications, vol. 33, 2021.

[9] P. N. S. C. H. F. D.-W. a. M. C. M. G. Dissanayake, "A solution to the simultaneous localization and map building (slam) problem," IEEE Transactions on robotics and automation, vol. 17, 2001.

[10] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP,," IEEE International Symposium on Systems Engineering, 2017.

[11] F. Cacciola, "A Survey of Polygon Offseting Strategies," May 2003. [Online]. Available: http://fcacciola.50webs.com/Offseting%20Methods.htm. [Accessed April 2022].

[12] D.-S. Kim, "Polygon offsetting using a Voronoi diagram and two stacks," ScienceDirect-Computer Aided Design, vol. 30, no. 14, pp. 1069-1076, 1998.

[13] L. Yan and A. Vigneron, "A Faster Algorithm for Computing Motorcycle Graphs," Springer Link, 2014.

[14] NAiSE, "NAiSE Autonomous Intralogistics- Manual," 2021.

[15] Bosch Rexroth, "boschrexroth.com," Bosch GmbH, [Online]. Available: https://www.boschrexroth.com/de/de/produkte/produktgruppen/montagetechnik/themen/intralogistik-loesungen-activeshuttle/. [Accessed April 2022].

[16] "Mosquitto," [Online]. Available: https://mosquitto.org/. [Accessed April 2022].

[17] "MQTT Explorer," [Online]. Available: http://mqtt-explorer.com/. [Accessed April 2022].

[18] "emqx," [Online]. Available: https://www.emqx.com/. [Accessed April 2022].

[19] "Analyticsvidhya," [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/easy-introduction-to-flask-framework-for-beginners/. [Accessed April 2022].

[20] "Javapoint," [Online]. Available: https://www.javatpoint.com/flask-http-methods. [Accessed April 2022].

[21] "Altexsoft," [Online]. Available: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/. [Accessed April 2022].

[22] F. Alhumaidan, "A Critical Analysis and Treatment of Important UML Diagrams Enhancing Modeling Power," Intelligent Information Management, vol. 04, 2012.

[23] D. Nagy, G. Papp and a. J. Benedek, "The gravitational potential and its derivatives for the prism," Journal of Geodesy, 2000.

[24] Y. Tang, "Run Multiple Functions in Parallel in Python3," PythonAlgos.

Declaration of Compliance

I hereby declare to have written this work independently and to have respected in its preparation the relevant provisions, in particular those corresponding to the copyright protection of external materials. Whenever external materials (such as images, drawings, text passages) are used in this work, I declare that these materials are referenced accordingly (e.g. quote, source) and, whenever necessary, consent from the author to use such materials in my work has been obtained.

Signature: [illegible]

Stuttgart, on the <23.05.2022>