D7.5 - INITIAL MONITORING AND COMMAND FRAMEWORK

Deliverable ID	**D7.5**
Deliverable Title	**Initial Monitoring and configuration framework**
Work Package	**WP7 – Deployment Toolchain**
Dissemination Level	**PUBLIC**
Version	**1.2**
Date	**2019-09-30**
Status	**Final**
Lead Editor	**TTTECH, TTA**
Main Contributors	**Artiza Elosegui, Paraskevas Karachatzis, Andreas Eckel (TTTECH, TTA)**

Published by the CPSwarm Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731946.

Document History

Version	Date	Author(s)	Description
0.1	2018-09-14	Artiza Elosegui (TTTECH, TTA)	Table of Contents
0.2	2018-10-08	Artiza Elosegui (TTTECH, TTA)	Inputs from Searchlab. Sections 2, 3, 4, 6 updated
0.3	2019-09-19	Paraskevas Karachatzis & Andreas Eckel (TTTECH, TTA)	Updates according to the latest implementations, Sections 3,4,5,6,7
0.9	2018-10-31	Andreas Eckel (TTTECH, TTA)	First draft ready for internal review
1.1	2019-09-19	Paraskevas Karachatzis & Andreas Eckel (TTTECH, TTA)	Updates according to the latest implementations, Sections 3,4,5,6,7
1.2	2019-09-30	Paraskevas Karachatzis & Andreas Eckel (TTTECH, TTA)	Updates & corrections according to internal review

Internal Review History

Review Date	Reviewer	Summary of Comments
03-11-2018	Gianluca Prato (LINKS)	Approved with minor comments
26-09-2019	Davide Conzon (LINKS)	Approved with minor comments
2019-09-25	Farshid TAVAKOLIZADEH (FRAUNHOFER)	Applied minor modifications. Added comments for further improvements.

1 Executive summary

This deliverable, namely "**D7.5 - Initial Monitoring and command framework**", introduces the development of the so-called "CPSwarm Monitoring Tool". The Monitoring Tool is the result of the "Task 7.3 CPS/SoS Monitoring and Command Tools". The consecutive results until M34 will be reported in "D7.6 – Final Monitoring and command framework".

The document introduces the CPSwarm Monitoring Tool architecture and implementation description as well as screen shots from the implementation. The tool has been developed up to a lab quality prototype level and will be further developed for product beyond the scope of this project.
The Monitoring and Configuration framework is integrated for use in the Search & Rescue and the Automotive Scenarios (Platooning Use Case). It is also designed for use in the Logistics Scenario but not implemented there since the more straight-forward solution for robotics was to use their control / monitoring approach.

Table of Contents

2 Introduction

This "**D7.5 – Initial Monitoring and Command framework**" is a public deliverable focused on the design, development and implementation of the Monitoring Tool. It details the status of the Monitoring Tool component and its implemented features responding to the requirements of the use cases/scenarios.

This deliverable is the result of the "Task 7.3 CPS/SoS Monitoring and Command tools". Another deliverable called "D7.6 Final Monitoring and Command framework" will be released in M34 describing the final version of the tool and the implemented features.

TTTech is the T7.3 leader and responsible for the delivery of D7.5. As such, TTTech leads the development of the Monitoring Tool with the requirements received from the use cases/scenarios. SLAB provided the Communication Library which was crucial for the implementation of the Monitoring Tool.

2.1 Document organization

The document is organized as follows:
Besides the "Executive Summary" and the "Introduction" sections, the document is composed as follows:

- Section 3 provides an overview on the requirements from the three use cases/scenarios for the Monitoring Tool.
- Section 4 provides an overview about the architectural embedding of the Monitoring Tool in the CPSwarm environment
- Section 5 describes how the Monitoring Tool is related to the Communication Library
- Section 6 describes the Monitoring Tool as such including the services and the implementation as seen also from the user interface
- Section 7 is wrapping up and provides a short conclusion
- Section 8 provides the acronyms table

2.2 Related documents

ID	Title	Reference	Version	Date
[RD1]	Final Vision Scenarios and Use Case Definition	D2.2	1.0	M16
[RD 2]	Updated System Architecture Analysis & Design Specification	D3.2	1.0	M18
[RD3]	Final Monitoring and Command framework	D7.6		M34
[RD4]	Initial CPSwarm Abstraction Library	D7.1		M18
[RD5]	Initial Automotive demonstration	D8.5		M24
[RD6]	Final CPSwarm Abstraction Library	D7.2		M32

3 Requirements of the industrial application scenarios

The development of the Monitoring Tool has been scheduled based on the requirements claimed from the partners of the three Industrial Application Scenarios, namely a) the Search & Rescue Scenario, b) the Automotive Scenario and c) the Logistics scenario. The implementation of a first set of requirements is considered completed by now. However, the relevant development activities will continue and focused the additional features until the delivery of the final version of the tool in M34.

Next, the requirements of the three scenarios are listed. This list is the result of the work done in WP2 for the requirements engineering.

3.1 The Automotive Scenario and the Search&Rescue (SAR) scenario requirements

These two Industrial Application Scenarios widely share the same requirements given that the area to be monitored is bigger than the logistics scenario. The Monitoring Tool shall be capable of handling the following services:

1) Visualization of the location of the swarm members (latitude, longitude).
2) A list of all swarm members along with important runtime data (topics), selected by the user.
3) Recording and visualization of all events sent in the swarm.
4) Visualization of the route for each swarm member.
5) Selection of individual swarm members.
6) Ability to manually trigger events in the swarm, either to selected swarm members or the whole swarm (using the discovery of the communication library).
7) Hotkeys for events such as: Start, Stop.
8) Setting global parameters to the swarm: distance_object (distance to keep from front object), max speed (hard limit on the speed), etc.

3.2 Logistics scenario requirements

Visualization of the following:

9) The cart that is assigned to each robot.
10) The path that each robot has to follow.
11) The location which the requested card must be sent to.
12) The number of available robots.
13) The assigned missions to each robot.
14) The list of the missions and their status.

4 The Monitoring Tool in the CPSwarm architecture

Figure 1 shows the final CPSwarm architecture in its functional view, which outlines the relationship between components. This figure refers to this component as the Monitoring & Command Tool to better reflect its functionalities.

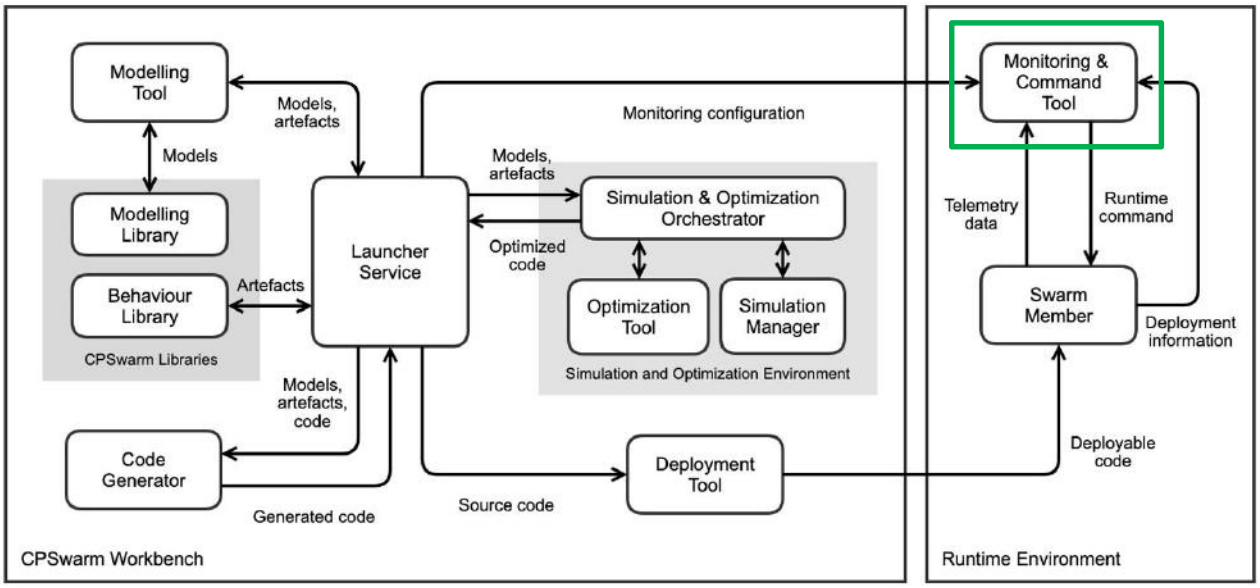


Figure 1: Architecture Design

The Monitoring Tool runs in the Runtime Environment. After the deployment phase, the Monitoring Tool is necessary to monitor the actual status of the swarm, as well as to send configuration commands and, if required reconfiguration commands to modify / update the swarm behaviour, e.g., to abort the mission or to re-purpose part of the swarm members. On one hand, it gathers real-time data from the swarm members and on the other hand, it sends out runtime commands to the individual swarm members. The information gathered will be presented to the user through the Graphical User Interface (GUI) generated in launch time.

Data exchanged between the swarm members and the Monitoring Tool, natively exploits a Publish/Subscribe interaction pattern to account the fact that:

1. Multiple listeners might need to receive telemetry or sensory data, on a dynamic subscription basis. Publish/Subscribe natively support this requirement by decoupling event sources from event consumers.
2. Data may be transferred opportunistically, depending on the actual connectivity and network conditions. This prevents the adoption of any client-server-like interaction paradigm where the Cyber Physical System (CPS) acts as server. Cases in which the CPS plays the client role are possible, however they might not be suited for high-frequency / high-cardinality data streams.

The information flow between the Monitoring and Command Tool and other components within the CPSwarm system are:

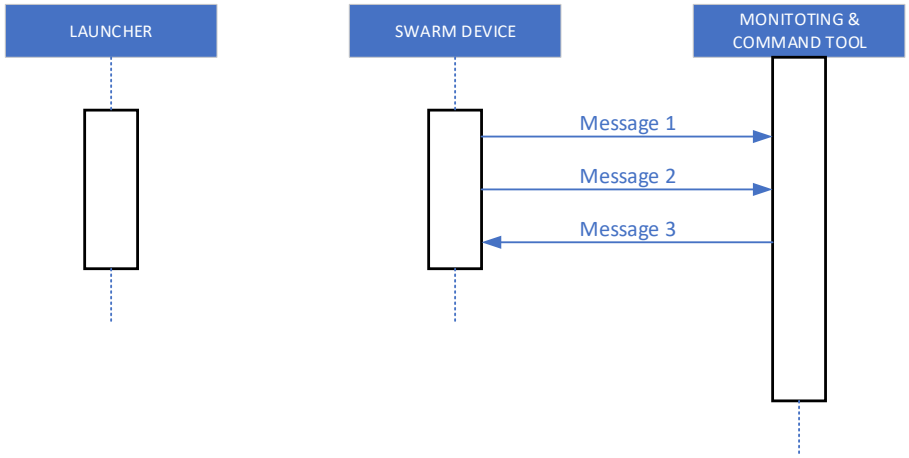


Figure 2: Monitoring Tool phase sequence diagram

The user/operator monitors the real-time status of the swarm as well as changes its behaviour during runtime (Figure 2).

The Web Browser needs to be opened to start the Maintenance and Monitoring Tool. After the Monitoring and Command Tool is launched, a swarm device discovery phase will be carried out, in which the swarm device will send data regarding its properties to the Monitoring and Command Tool (see Figure 2, Message 1). After that, the Monitoring and Command Tool is ready to monitor and command each member in the swarm. Message 2 (see Figure 2) represents the data flowing from the swarm device to the Monitoring and Command Tool. It contains the real-time status of the CPS, such as the current location, current speed, current battery life, etc. Message 3 (see Figure 2) represents the commands sent from the Monitoring Tool to the swarm, such as changing swarm behaviour, shutting down the swarm, etc.

5 Communication Library

The Monitoring and Command Tool uses the Communication Library – *libswarmio* – to send and receive events and telemetry, to set and read-back parameters and to discover swarm members on the network.
Alternatively, the *libswarmio* can be used via the Robot Operating System (ROS) bridge allowing the ROS messages to be propagated to other agents or tools (using the *libswarmio*, see Figure 3.

Please, refer to D7.2 "Initial CPSwarm Abstraction Library" for the full description of the Communication Library

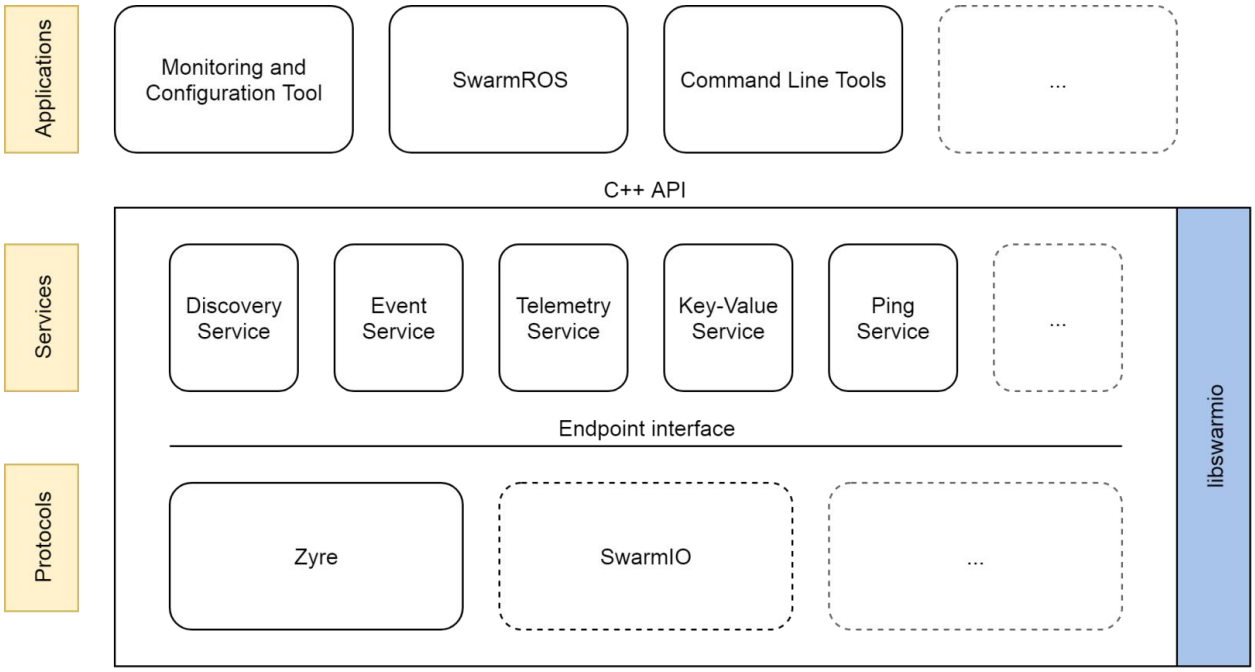


Figure 3: The architecture of the Communication Library

The Communication Library provides a service oriented, cross-platform solution for performing common actions with swarm members. Development of the Monitoring and Command Tool and the Communication Library, and the design of the services provided by the latter and the functionality that is required to be implemented by the former went hand-in-hand with the efforts to efficiently model and describe the networked interaction of swarm members. As such, the following basic services are provided by the Communication Library (pls. note that only some services are highlighted herein, the others are described in detail in D7.2):

a) Event Service
b) Telemetry Service
c) Key Value Service
d) Discovery Service

Event Service

Swarm members send and receive events as their behavior is executing – informing other members of important events and reacting to external and internal stimuli in order to change or modify the current state of execution. An event, on its own, has only a name and a list of parameters – it is only how the behavior reacts that makes the event meaningful. As such, events can represent commands issued by the operator, real events happening on a local or remote node or other simple messages that aid coordination. The

Monitoring and Command Tool can use the Event Service to send arbitrary events to swarm members (in order to issue commands) and can monitor events as they are happening on swarm members.

Telemetry Service

For the operator to receive meaningful information about the state of each swarm member, a continuous stream of information needs to be sent by the swarm members being monitored to the Monitoring and Command Tool, and eventually, to the operator. The Telemetry Service can be used to subscribe to such information on-demand, specifying the required resolution and scope of the information. All data sent back is strongly typed and can have complex schema. Each telemetry value (however complex) is treated as an atomic value relevant to a single time point. The Monitoring and Command Tool uses the Telemetry Service to display and visualize the key elements describing the state of individual swarm members.

Key-Value Service

Parameters such as the operational area or the location of known obstacles are subject to change during deployment, and as such, need a way to be set during the mission. The Key-Value Service provides a way to write (and read) complex named values on swarm members – values the behavior can use to perform calculation and make decisions. The Monitoring and Command Tool uses the Key-Value Service to retrieve and set the parameters that govern swarm member behavior.

Discovery Service

The Discovery Service is responsible for detecting the supported features of participating swarm members. In order to make the Monitoring and Command Tool a universal tool for the management of compatible swarms, regardless of specific behavior or target hardware, the Communication Library provides a way to obtain a description of the events supported by each member and of the different telemetry and parameter values and their underlying data types. The Discovery Service works on two layers: the lower layer, provided by the specific endpoint implementation, is purely responsible for detecting the presence of swarm members and tracking their online-offline states – while the higher layer can request and answer, as well as cache and invalidate information about the supported facilities.

Taking advantage of these services, the Monitoring and Command Tool can present a toolset and user interface that is independent of the communication medium and the concrete implementation of swarm member behavior. Since the Communication Library uses Google Protocol Buffers[1] for serialization, changes in the underlying data schema can be made while retaining backwards compatibility, making it possible to interact with newer (and in some cases, older) versions of the library – while not recommended, this allows for incremental updates and ensures that the serialization format is architecture independent.

The communication between the Monitoring and Command Tool and the Abstraction Library is intermediated by the Communication Library. Exploiting the discovery functionality provided by the Communication Library, the events and topics of the swarm members can be discovered. Using this information, the interface of the Monitoring and Command Tool can be dynamically generated to let the user to send events and to filter the data coming from the swarm member keeping only the one that the user has selected to see.
The list of commands that can be sent to the agent is exposed to the Monitoring and Command Tool through the discovery feature of the Communication Library. Sending commands from the Monitoring and Command Tool is possible to influence the behavior of a specific CPS triggering events that start the execution of a particular sequence of actions.

[1] https://developers.google.com/protocol-buffers

6 The Monitoring Tool and Command Tool

6.1 Overview

The Monitoring and Command Tool addresses the challenges related to the after-deployment phase, i.e., to the swarm device mission execution. Its main objective is to monitor the swarm members' behaviour by constantly supervising the individual swarm members, the swarm behaviour and performance. Rather than applying local control, it offers the means for continuously checking the performance of real swarm with respect to the mission to reach. In addition to monitoring, the Monitoring and Command Tool also tackles (re-)configuration of swarm members' parameters depending on external factors.

Swarm members can receive commands, e.g., to switch between pre-programmed behaviours, and/or configuration parameters through the channel established by the Monitoring and Command Tool, exploiting the telemetry core of the runtime environment.

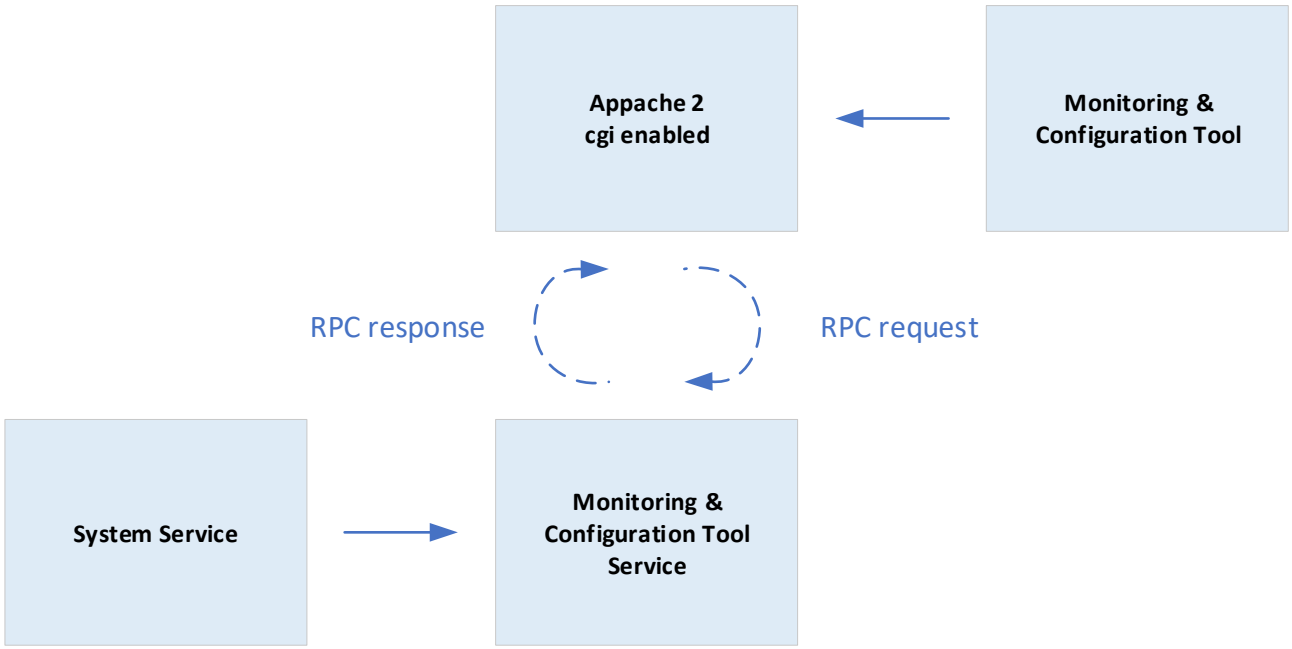


Figure 4: Monitoring & Command Tool structure

The Monitoring and Command Tool operates like a web page which itself consists of several entry points. The System service (like a binary) user space service is using the *libswarmio* library (Communication Library) running continuously on the background polling the status info permanently to have updated values available at any time. The Monitoring and Command Tool sends query and receives response about the information requested (See Figure 4).

Legend:

- Apache 2: open-source HTTP server, pls refer to https://httpd.apache.org/.
- cgi: common gateway interface, pls refer to https://httpd.apache.org/docs/2.4/howto/cgi.html
- RPC: Remote Procedure Call, pls refer to https://de.wikipedia.org/wiki/Remote_Procedure_Call

OpenStreetMaps[2]

The environment visualization is ensured since the Monitoring and Command Tool is created based on the OpenStreetMaps[3] API. OpenStreetMap is a global collaborative (crowd-sourced) dataset and project that aims at creating a free editable map of the world containing information about the environment. It contains data for example about streets, buildings, different services, and land use. OpenStreetMap provides free editable maps of the whole world and is a restriction-free mapping solution that can be used for commercial and non-commercial usage which any limitation.

Although our first attempt was to use the Google Maps API,[4] at the end we concluded that the Open Source solution offered by Open Street Maps was more appropriate since we were looking for more customization and control. Open Street Maps provides ability to manage things according to our requirements.

The Web Interface is implemented by a Javascript application.
In order to handle the maps (recommended: Google Maps, OpenStreetMap, other maps are possible as well) and the layers to visualize on top of them the Leaflet[5] Javascript framework is leveraged. With Leaflet, it is possible to visualize the layers as KML[6] files, which are basically Vector Graphics. They use the leaflet-omnivore functionality provided by Leaflet.
The Web Interface consumes REST Web Services from an HTTP server in order to get static and historical information and it is also connected to an MQTT Broker using an MQTT[7] Javascript client to monitor the agents. In CPSwarm case we would substitute it with request & reply service function as described above.

6.2 Communication Library integration

The first integration of the Communication Library into the Monitoring Tool has been carried out in 8 steps in the first iteration:

- Key Pair Configuration Panel
- Key Pair Monitoring
- Key Pair Value Set
- Event Trigger Generation
- Event Sending
- Telemetry Configuration Panel
- Telemetry Monitoring
- Telemetry Set

The Communication Library does not support arrays for the first iteration. In order to visualize the path of the agent, as required by the Logistics use case, multiple points are needed and now this is not possible to do using the Communication Library.

In the second iteration of the integration of the Communication Library into the Monitoring Tool, the following steps were implemented:

- Communication Library Functions Integration
- Add common Events
- Add user generated Events
- Add object popups
- Agent selection

[2] https://www.openstreetmap.org/
[3] https://www.openstreetmap.org/about
[4] https://developers.google.com/maps/documentation/?hl=de and https://cloud.google.com/maps-platform/
[5] https://leafletjs.com/
[6] https://developers.google.com/kml/documentation/
[7] http://mqtt.org/

In the third iteration of the integration of the Communication Library, two steps were required:

- Area of operation selection
- Send commands to individual targets

The last step is covered in the second version of the tool (already done in the current version):

- Event Monitoring

6.3 Implementation of the State Machine

As kind of a top-level layer we have implemented a State Machine-controlled structure on top of the Monitoring and Command Tool. The Monitoring and Command Tool certainly can also be operated not using the top-level control software as it is currently done in the Search and Rescue Scenario. The state-machine approach is used in the Platooning use case of the automotive scenario.
Initially also meant to be only needed for the platooning use case, we found out that in principle swarm applications will find it useful since they will have similar challenges as also tackled by the State Machine approach. These could be that the drones would fly in formation for a certain time until they reach the area of final operation in order to dissolve and conduct search and rescue functions on an individual basis. The same could arise in the logistics case. Consider that there would be more than one box to transported from position "A" to one and the same final destination. In such case it is quite likely that the carts would form kind of a platoon as well. These are just some ideas and are not generally deployed within the project. Nevertheless, this was the reason to also highlight this in the Monitoring and Command deliverable document since we thought that it is not limited to the automotive scenario and the platooning use case.

A principle design of a State Machine described in D4.6 and contributed by LAKE was the blue-print for the approach implemented in the Monitoring and Command Tool (See Figure 5).
The following "level one" states were defined:

a) Start-up
b) Idle
c) Wait Response
d) Select Ride
e) Goto Meeting Point
f) Joined
g) Follow Lead

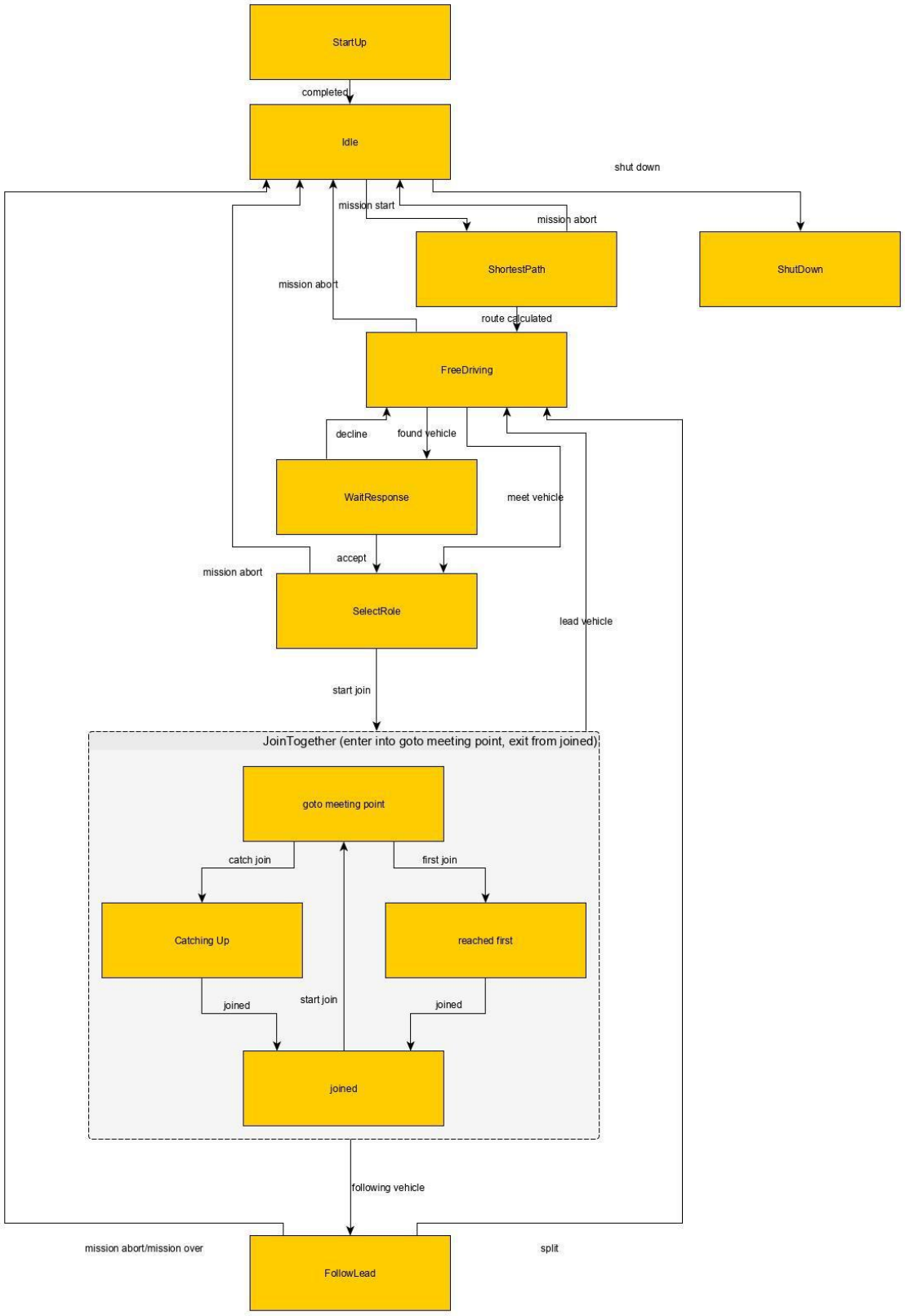


Figure 5: State Machine approach

6.4 Configuration panel

The configuration panel is generated dynamically by the communication library. The user can enable/disable topics.
The following few examples using the Automotive Scenario (Platooning) are documented by screenshots from the Monitoring and Command Tool screen showing a complete platooning mission on the Monitoring and Command Tool screen.

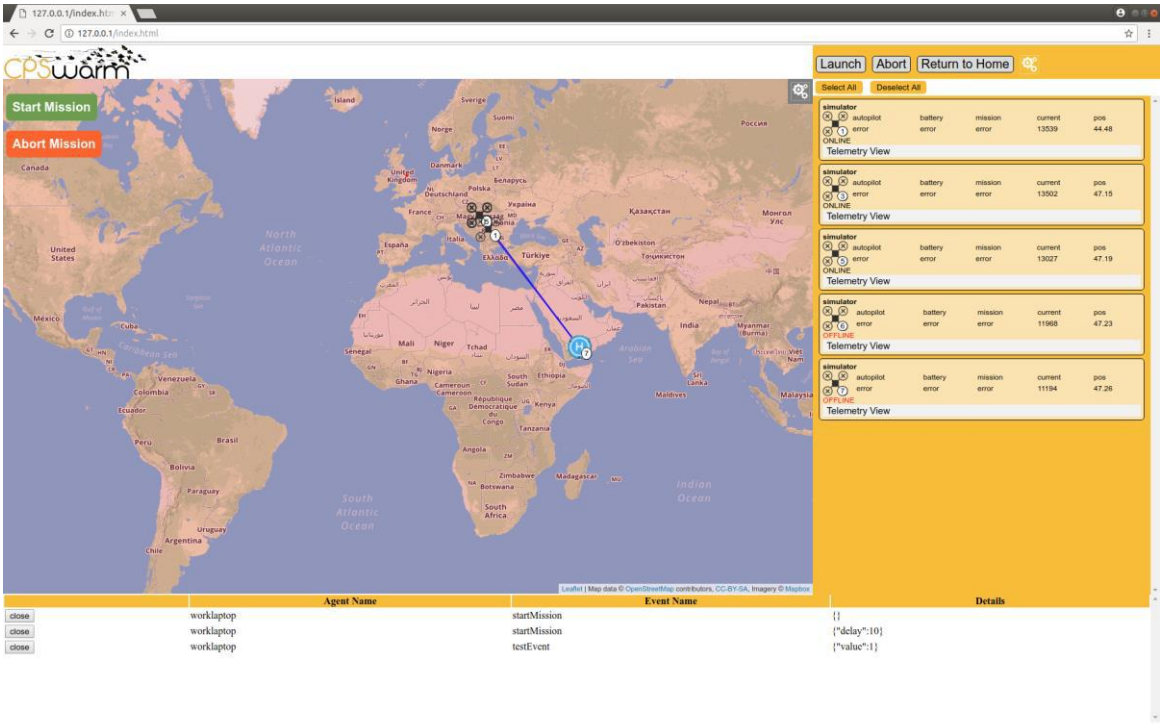


Figure 6: Platooning planning

Figure 6 shows the planning start for the platooning mission on the Monitoring and Command Tool screen.

Figure 7 displays the Configuration Panel of the Monitoring and Command Tool.
In the Configuration Panel the key parameters of the agents/swarm can be set (i.e. in the platooning scenario this would be the cruising speed, accept/deny platoon participation, different kinds of threshold values, etc.). Based on the events advertised by the agents the user can send events and its associated parameters concerning the individual agents or to a group of agents.

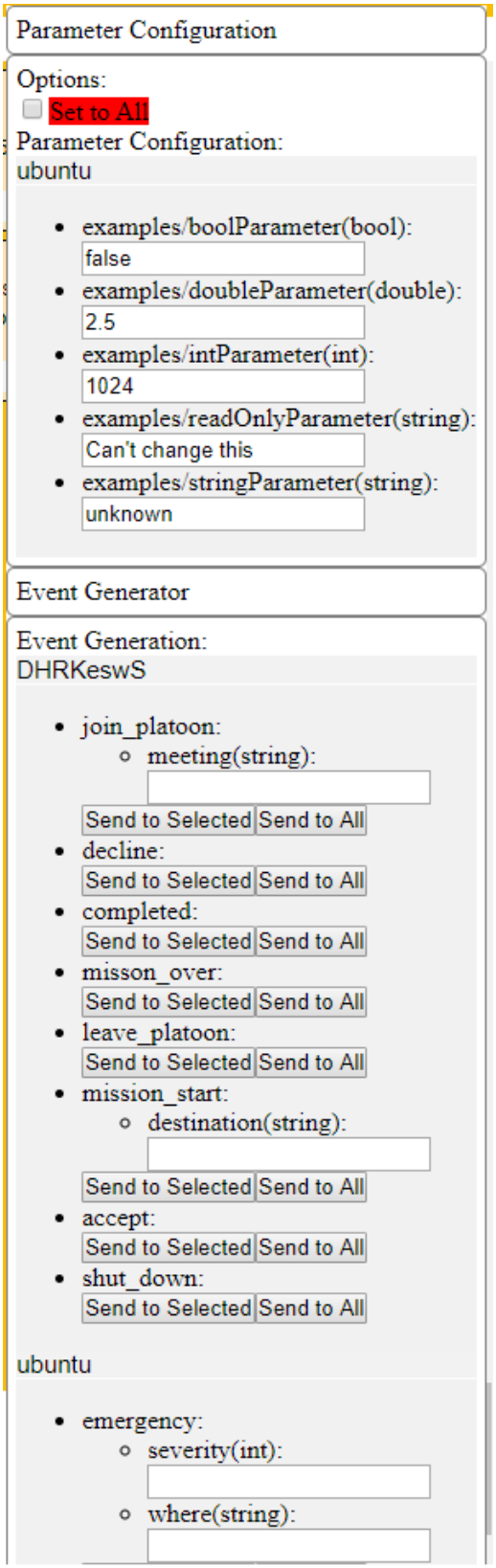


Figure 7: Configuration Panel

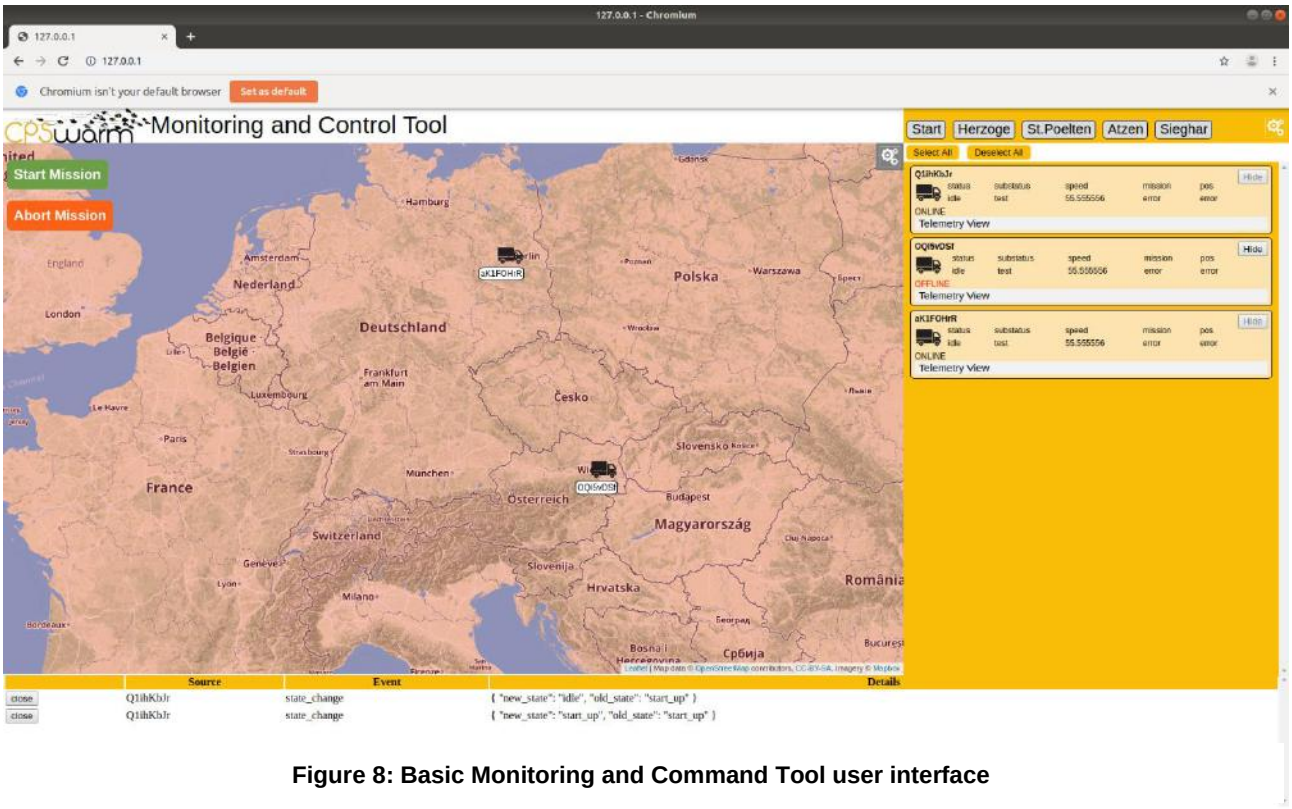
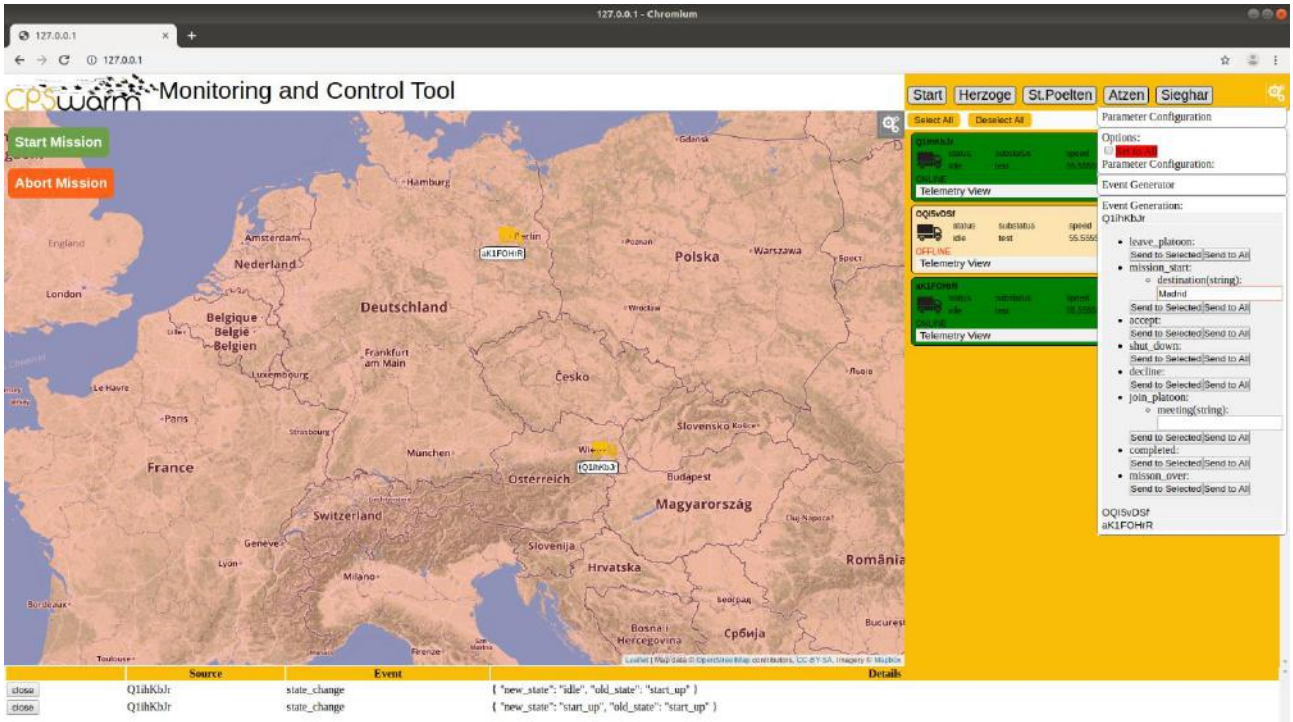


Figure 8: Basic Monitoring and Command Tool user interface

Figure 8 shows that 3 trucks are detected. One of them is OFFLINE due to connection lost.



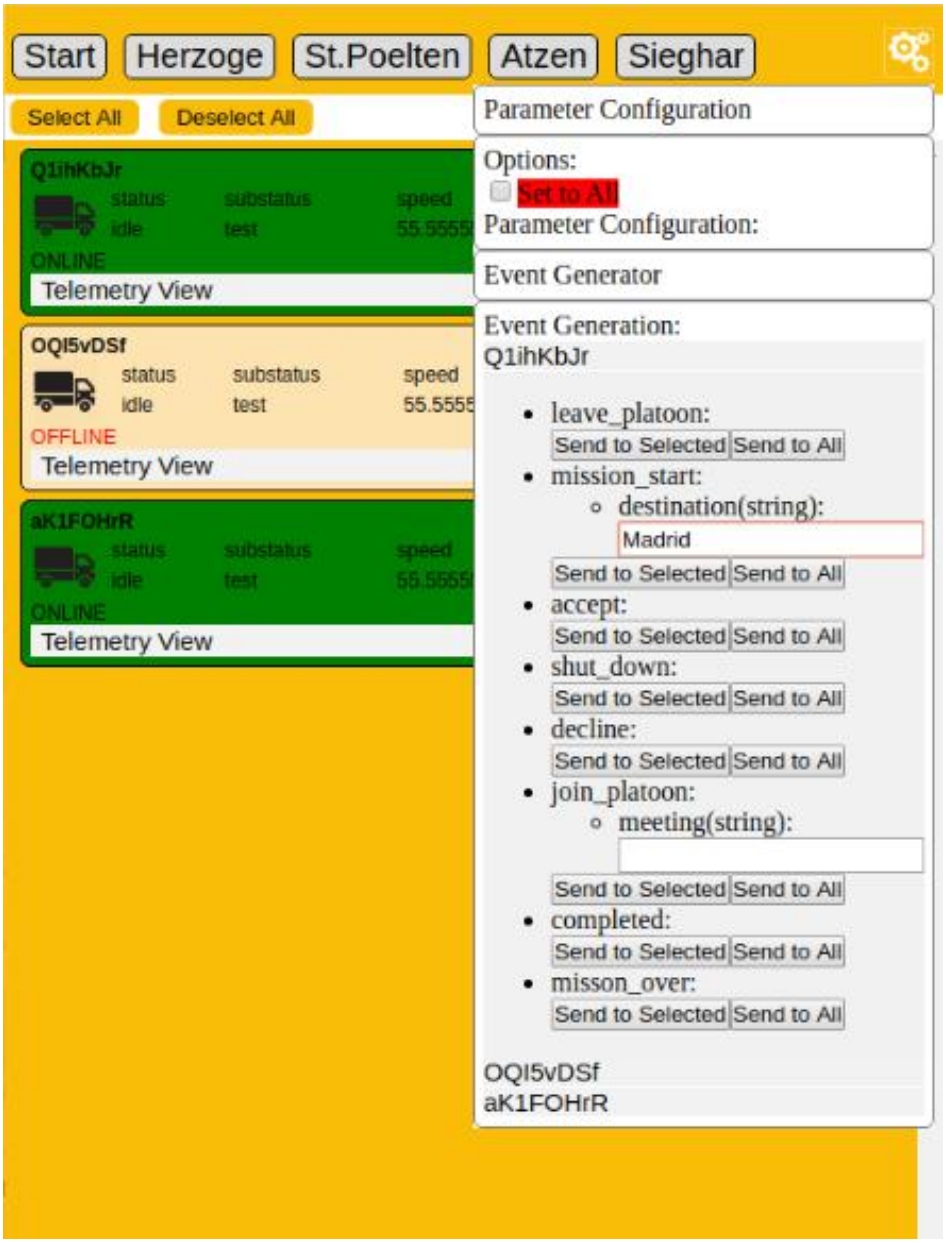


Figure 9: Configuration Panel: in the entire screen view (figure above) and the panel zoomed for better readability

Figure 9 shows the Monitoring and Command Tool with the Configuration Panel opened. The events and parameters are generated from the discovery functionality of the communication library.

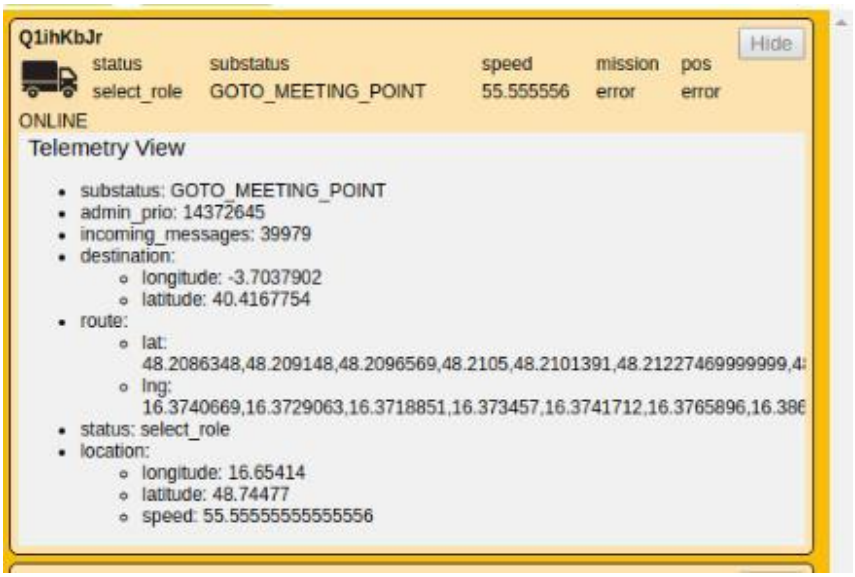


Figure 10: Telemetry data displayed

Figure 10 shows the telemetry data panel opened for the first truck (agent).

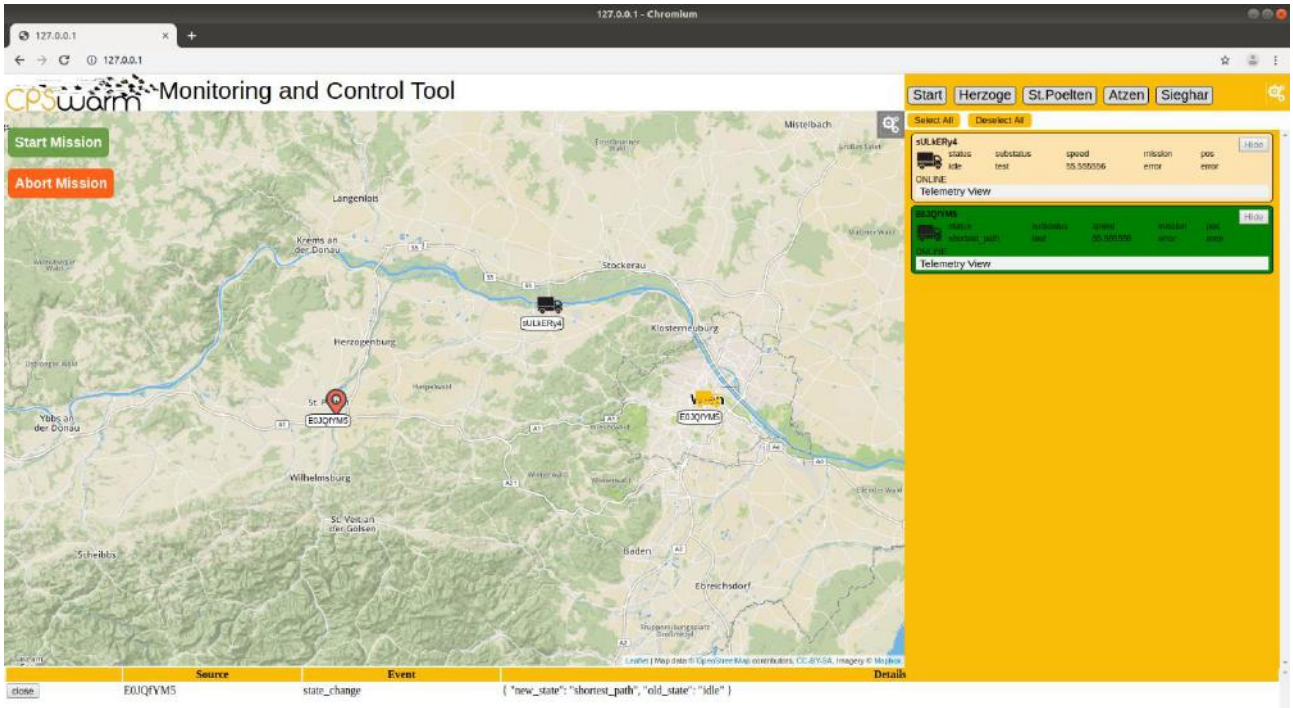


Figure 11: Platooning preparation phase (second truck seeking platooning)

Figure 11 shows the situation at the panel where the platooning is prepared. The request for platooning is sent by the first Truck to the second Truck ("first truck" & "second truck": these names must be defined in the communication library) and is awaiting confirm to agree in driving in platooning configuration.

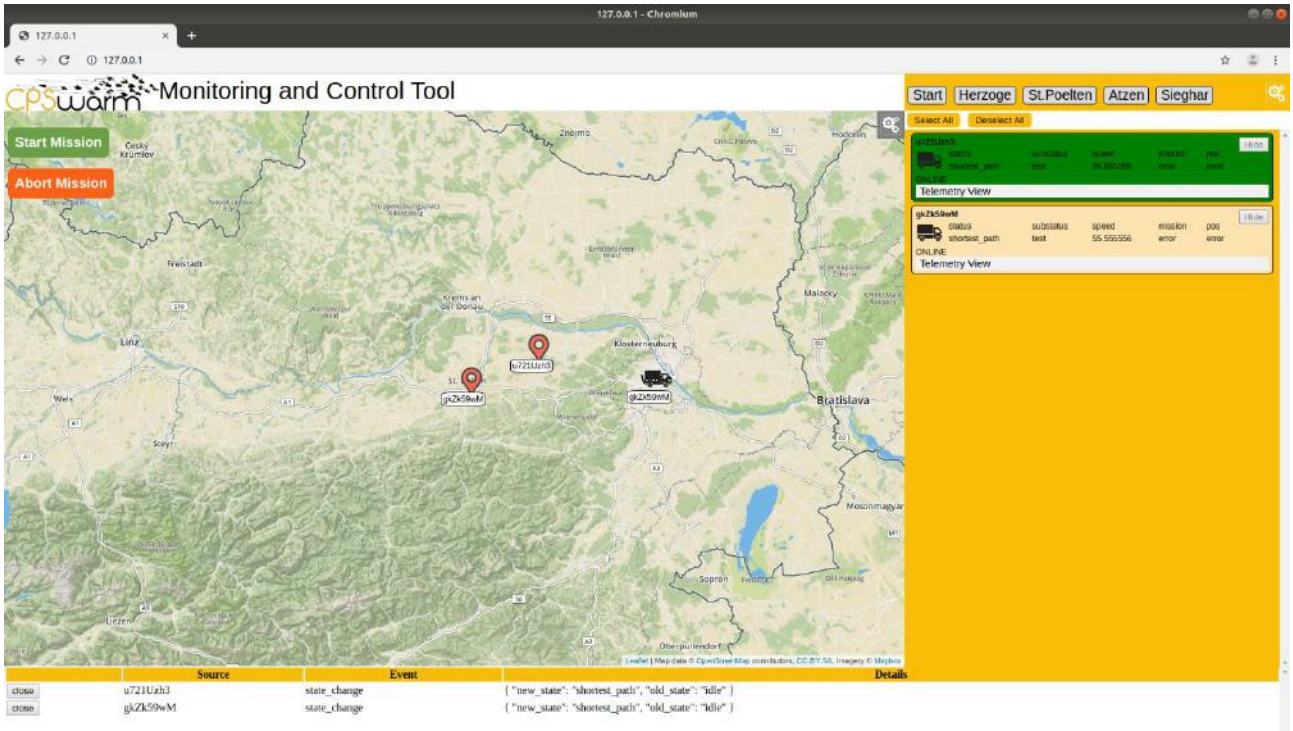


Figure 12: Platooning preparation phase (first truck ready to agree in platooning lead)

Figure 12 shows the status of the first truck (platooning lead).

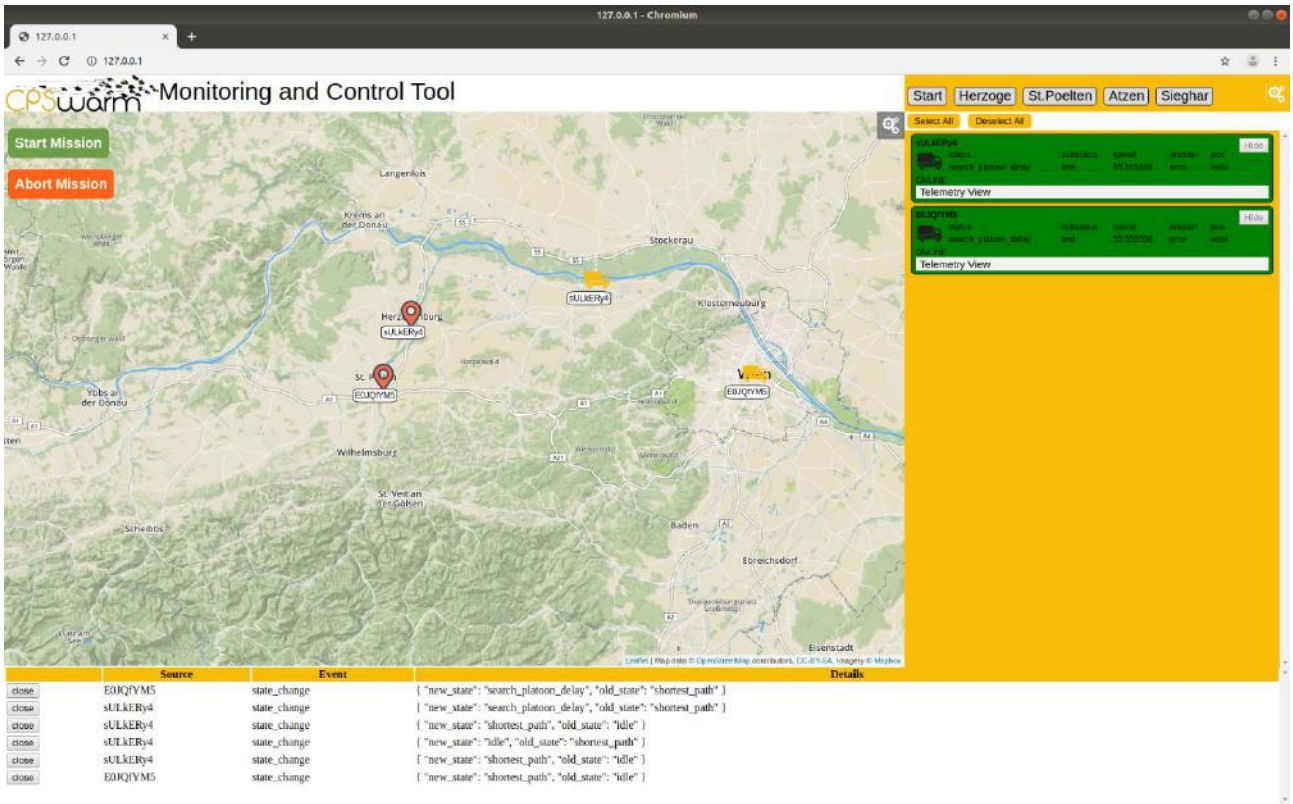


Figure 13: Start the "platooning mission"

Figure 13 shows the status when both trucks agree to go in platooning configuration and start the "mission".

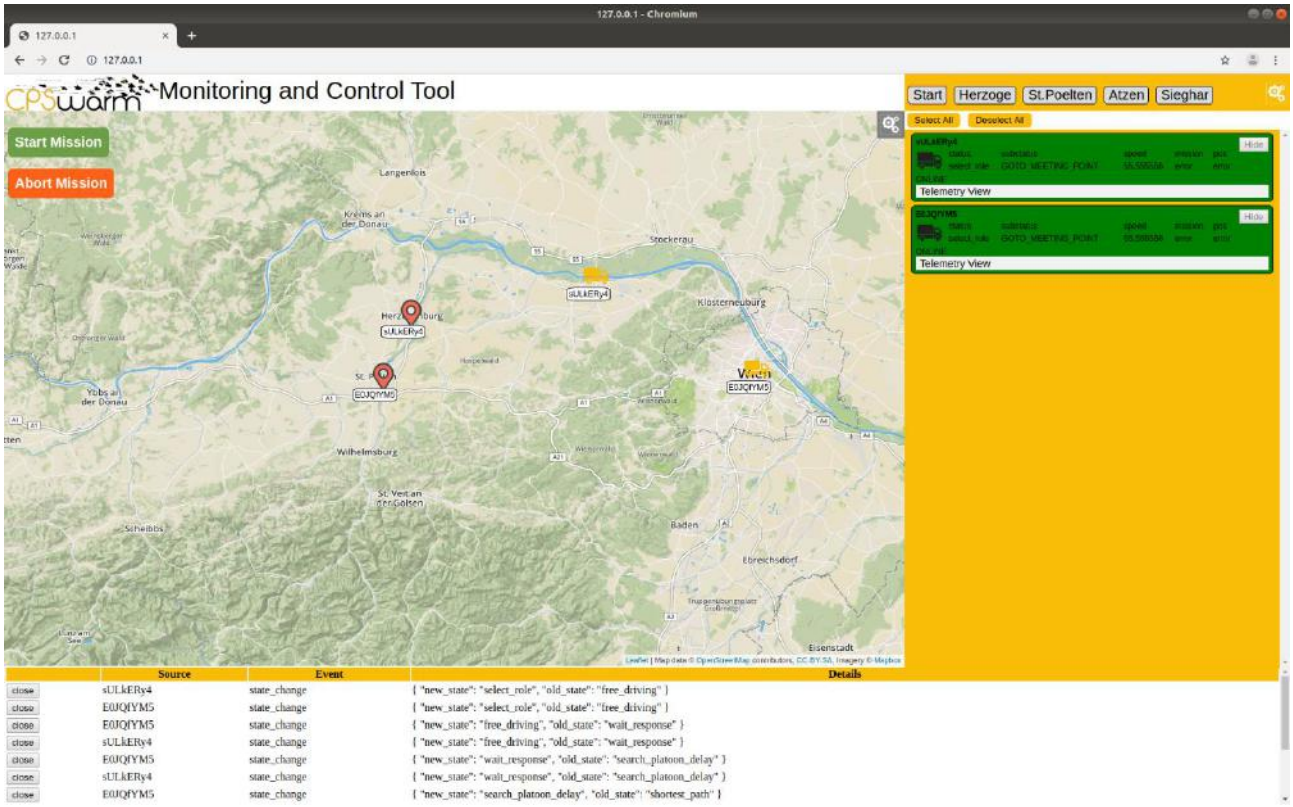


Figure 14: preparing for the "rendezvous"

Figure 14 shows the situation where both trucks are preparing for the rendezvous (the procedure of the two vehicles meeting at agreed position).

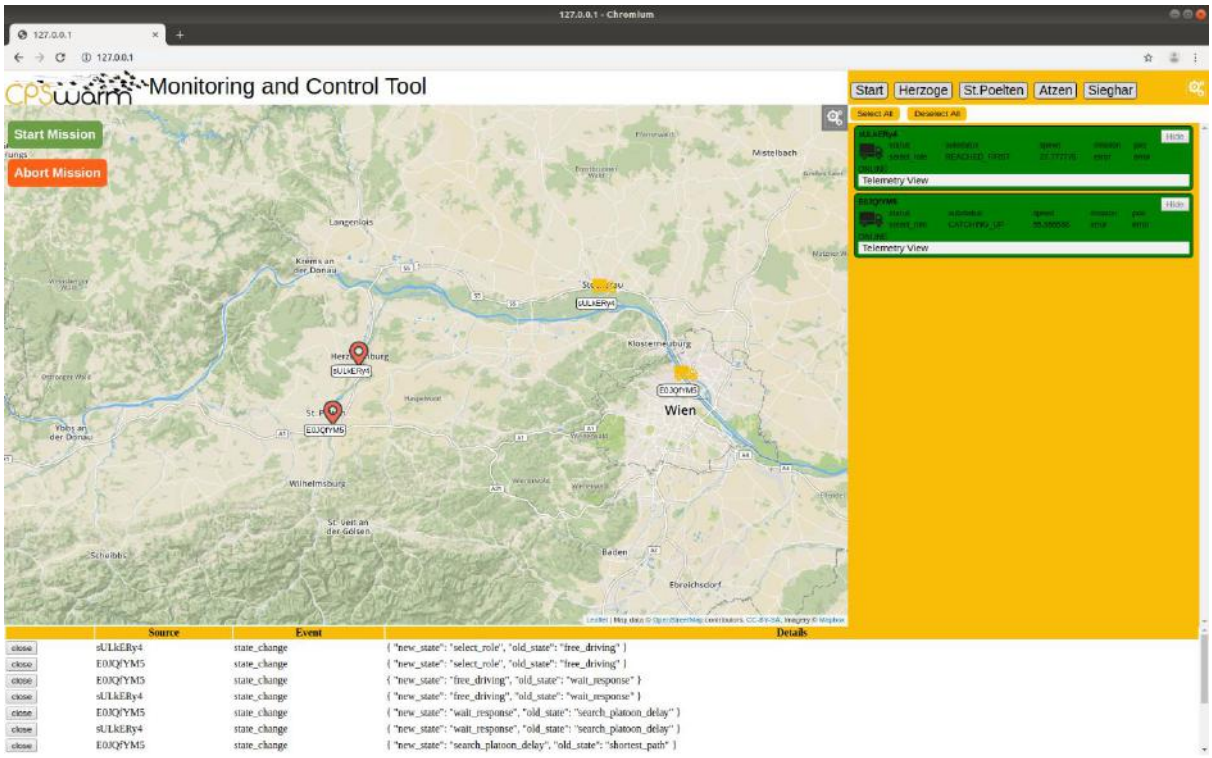


Figure 15: Rendezvous

Figure 15 shows that the first of the two trucks is arriving at rendezvous point and slows down to meet the other one for starting to drive in platooning configuration

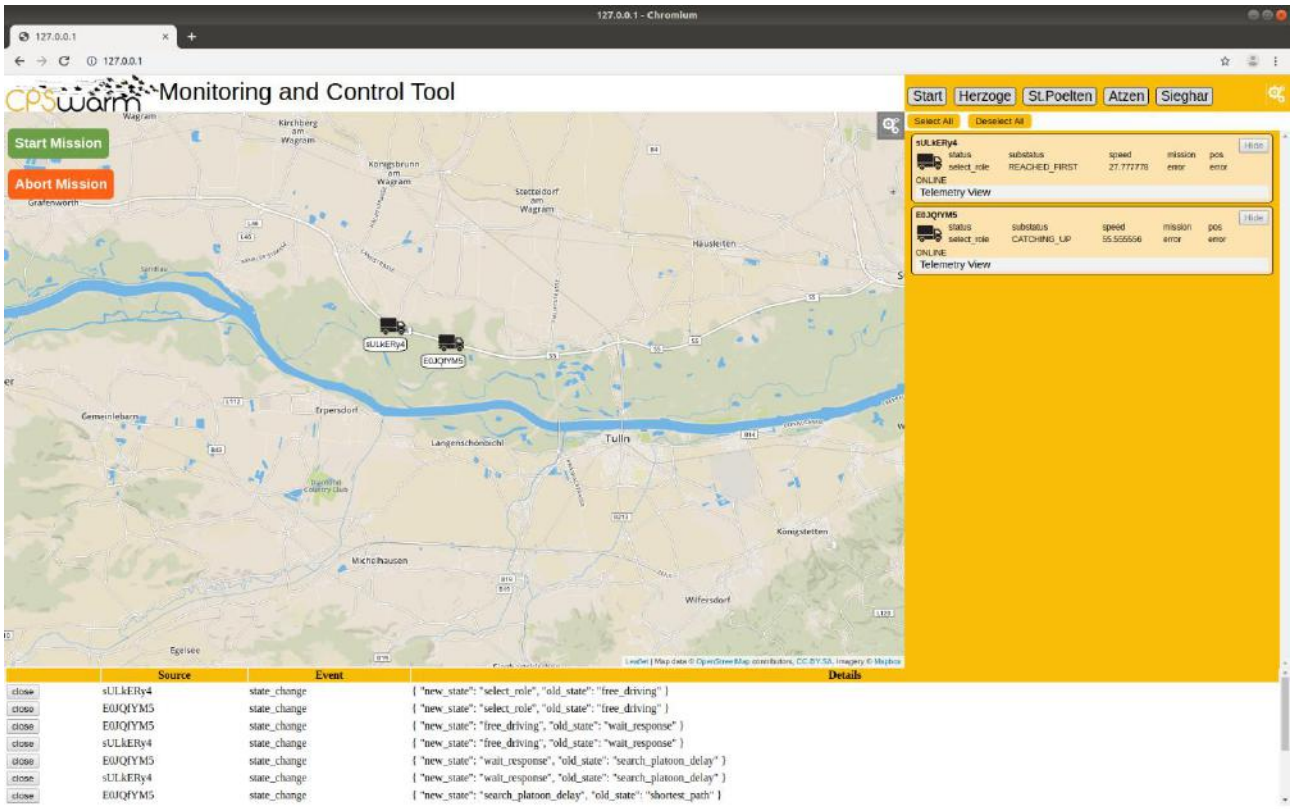


Figure 16: 2nd truck arrives for rendezvous

Figure 16 shows the second truck catching up for rendezvous.

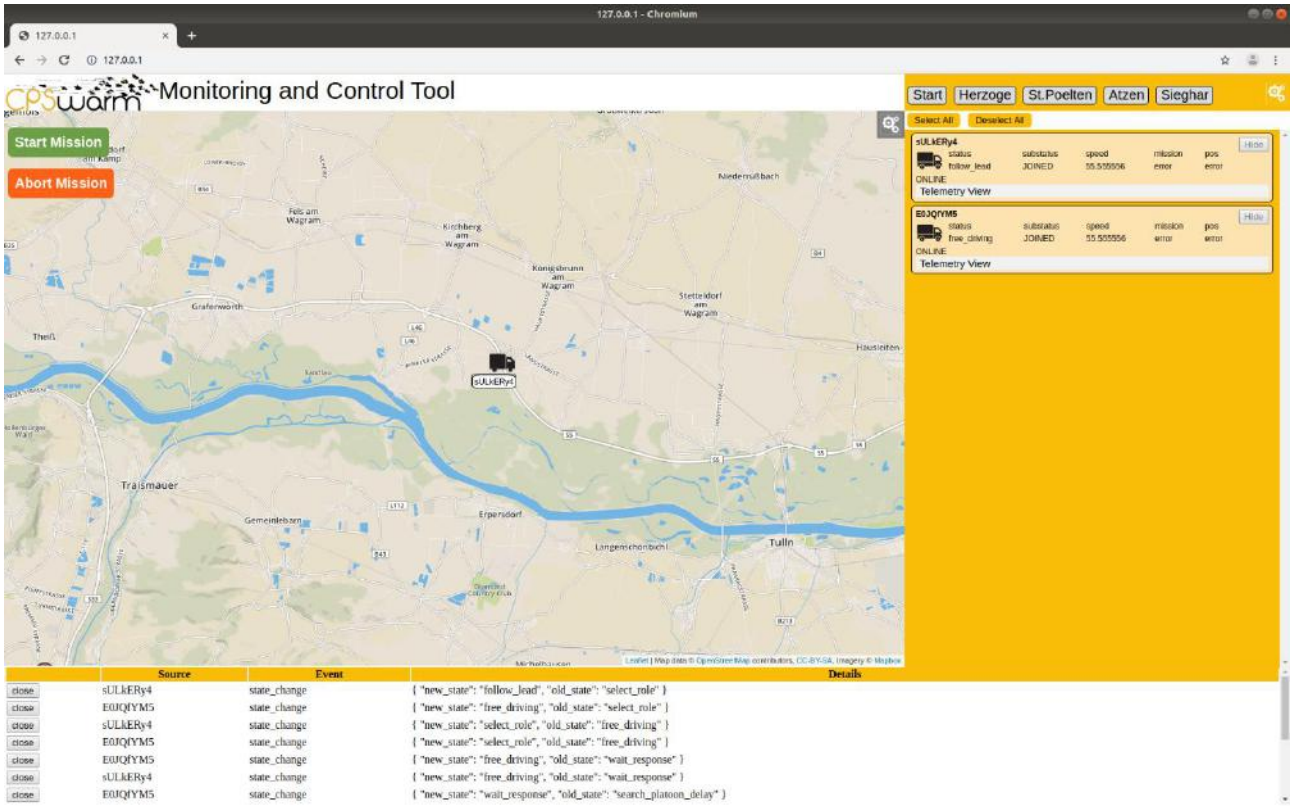


Figure 17: Both trucks meet at rendezvous point

Figure 17 shows that both trucks have arrived at rendezvous point and start driving in platoon configuration.

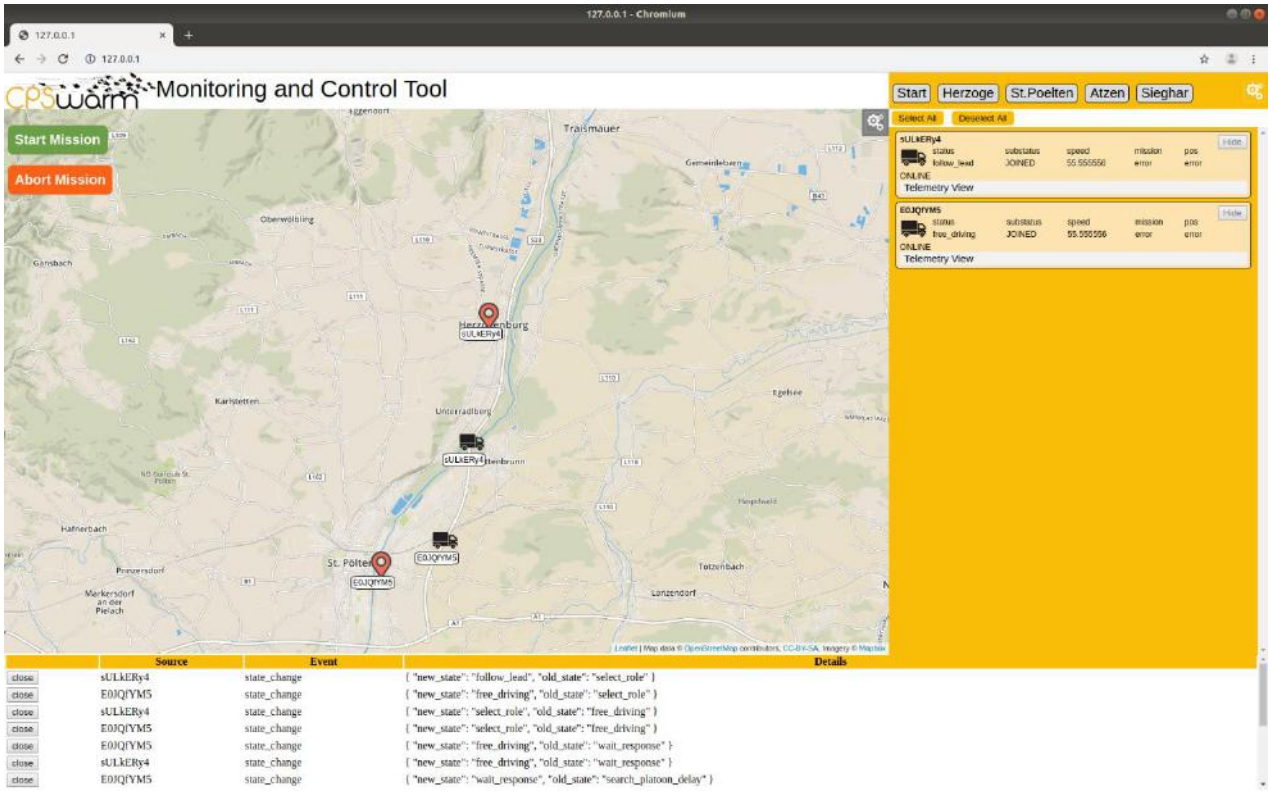


Figure 18: Trucks arrive at "split point"

Figure 18 shows that the two trucks have reached the point where they need to depart and stop platooning configuration.

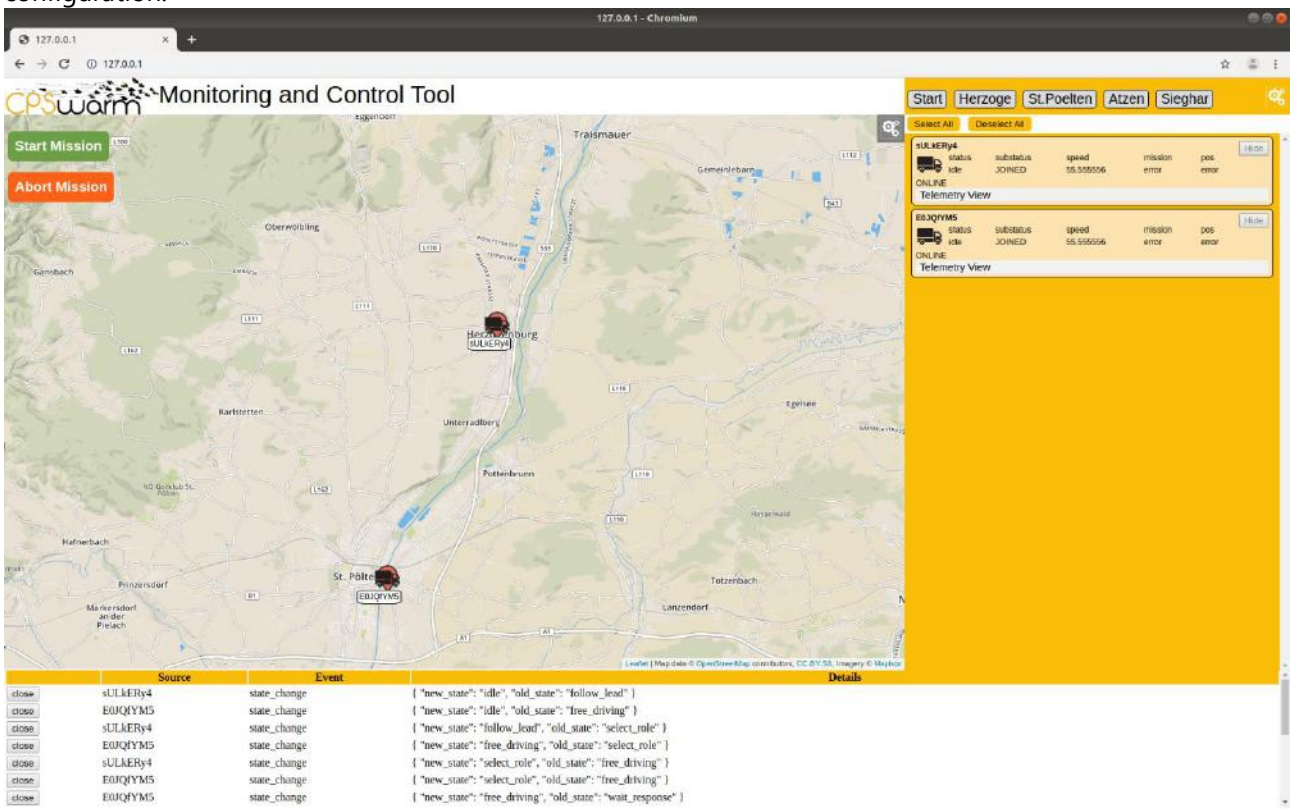


Figure 19: trucks reach their destination

Figure 19 shows that both trucks have reached their final destination points.

The Monitoring and Command Tool has a configuration file (See Figure 20). This is the only file the Monitoring and Command Tool needs. The file contains configuration for agent icons (default and selected, which telemetry to show, home icon). The configuration file also includes the buttons (both global events and custom events) to be generated (as well as their parameters.

configuration.js

```
guiEventButtons = {
  customEvents : [
    {
      buttonText: "Launch",
      eventName: "launch",
      parameters: {},
      styleClass: "event_button"
    },
    {
      buttonText: "Abort",
      eventName: "abort",
      parameters: {},
      styleClass: "event_button"
    },
    {
      buttonText: "Return to Home",
      eventName: "rth",
      parameters: {},
      styleClass: "event_button"
    }
  ],
  globalEvents : [
    {
      buttonText: "Start Mission",
      eventName: "startMission",
      parameters: {},
      styleClass: "start_button"
    },
    {
      buttonText: "Abort Mission",
      eventName: "abortMission",
      parameters: {},
      styleClass: "abort_button"
    }
  ]
}

agentTypes = {
  default:{
    quickdata: {},
    defaultIcon : "drone.png",
    selectedIcon : "drone_selected.png"
  },
  drone: {
    quickdata: {
      autopilot : "autopilotState/mode",
      battery: "batteryState/percentage",
      mission: "",
      current: "incoming_messages",
      pos: "position/latitude"
    },
    defaultIcon : "drone.png",
    selectedIcon : "drone_selected.png",
```

Figure 20: Sample from the Monitoring and Command Tool's Configuration File

6.5 Implementation

The following table shows the status of the implementation of the requirements listed earlier.

Requirement	Use case	Status	Comments
Visualize the location of the agents (lat,lng)	Automotive Search&Rescue	Implemented	
A list of all agents along with important runtime data (topics), selected by the user	Automotive Search&Rescue	Implemented	
Record and visualize all events sent in the swarm	Automotive Search&Rescue	Implemented	
Visualization of the route for each swarm member possible (not part of the current implementation, but could be added easily)	Automotive Search&Rescue	Implemented	
Selection of individual agents	Automotive Search&Rescue Logistics	Implemented	
Ability to manually trigger events in the swarm, either to selected agents or the whole swarm	Automotive Search&Rescue	Implemented	
Hotkeys for events such as: Start, Stop	Automotive Search&Rescue Logistics	Implemented	
Setting global parameters to the swarm	Automotive Search&Rescue	Implemented	
The cart that is assigned to each robot	(not used by Logistics use case)	ready for Use by the application engineer	
The path that each robot has to follow		ready for Use by the application engineer	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731946.

The location which the requested card must be sent to	(not used by Logistics use case)	ready for Use by the application engineer	
The number of available robots	(not used by Logistics use case)	ready for Use by the application engineer	
The assigned missions to each robot	(not used by Logistics use case)	ready for Use by the application engineer	
The list of the missions and their status.	(not used by Logistics use case)	ready for Use by the application engineer	

7 Use case specific configuration of the Monitoring and Command Tool

The following passage provides user information about how to configure and use the Monitoring and Command Tool.

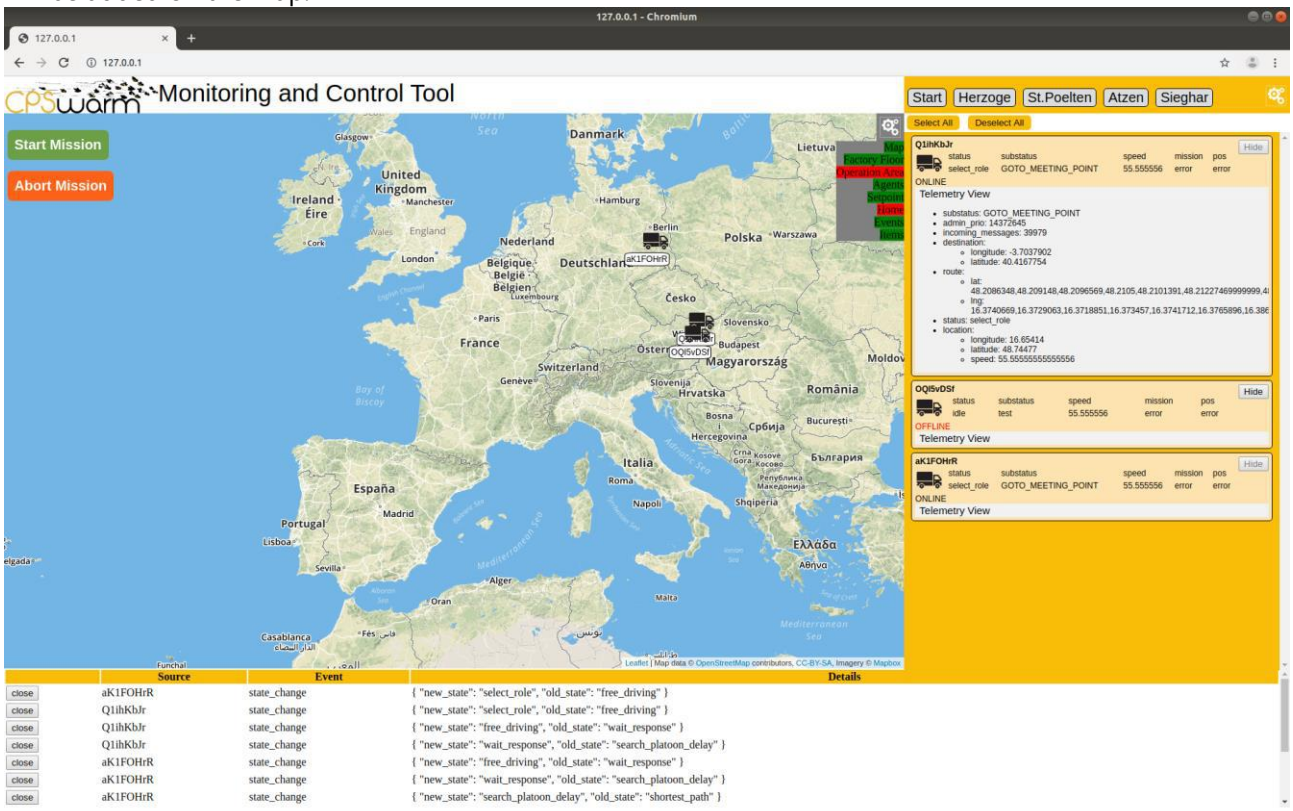
7.1 The Monitoring and Command Tool Installation

The monitoring tool consists of two main parts. The font-end GUI which is a dynamically generated html page, and the bridge to the swarmio library. The bridge (monitoring-tool-service) is responsible for getting the information from the swarmio library to the front end.
The source code for these two parts can be found in GitLab under the following links (recommended to install the monitoring tool and then the bridge), the repos contain a markdown file with details on installation:
https://git.repository-pert.polito.it/pkarachatzis/monitoring-tool

https://git.repository-pert.polito.it/pkarachatzis/monitoring-tool-service

7.2 The Monitoring and Command Tool Configuration

Any agents discovered by the communication library will be appended to the list, and their telemetry data will be displayed, additionally if the agents publish location data (location/longitude, location/latitude) a marker will be added on the map.




```
agentTypes = {

  truck: {   <== Display Type will be selected based on the type advertized by the swarmio lib

   quickdata: { <== Attributes added here will be displayed as seen in the image
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731946.

```
above

      status : "status", <== The format is <displayedName> :
"libswarmioTelemetryName"

      substatus : "substatus", <== The format is <displayedName> :
"libswarmioTelemetryName"

      speed : "location/speed", <== The format is <displayedName> :
"libswarmioTelemetryName"

      mission: "", <== The format is <displayedName> : "libswarmioTelemetryName"

      pos: "position/latitude" <== The format is <displayedName> :
"libswarmioTelemetryName"

    },

    defaultIcon : "truck.png", <== non seleted icon displayed (required swarmio
topics: position/longtitude, position/latitude)

    selectedIcon : "truck_selected.png", <== selected icon displayed (required
swarmio topics: position/longtitude, position/latitude)

    homeIcon: "location_pin.png" <== icon to display for docking station (required
swamio topics: destination/longtitude, destination/latitude)

  }

}
```

In the image above we can also see two groups of buttons, the button to the left are global event button that will be sent to the entire swarm, white buttons to the right will be sent only the selected agents. These buttons can be customized in the following sections in the configuration file globalEvents and customEvents the definition format is similar for both of them

```
{

  buttonText: "Start", <== Text to be displayed

  eventName: "completed", <== Event name to send

  parameters: {}, <== parameters of the event

  styleClass: "event_button" <== adds the following class to the button (the class
be defined using css to customize the button)

}
```

The parameters are specified in the following way: parameters: { "<parameterName1>" : ["<parameterValue1>", <parameterType1>], "<parameterName2>" : ["<parameterValue2>", <parameterType2>] }. Parameter type is an integer showing the type of the parameter. The id for each type can be found at the beginning of the configuration file.
Ex. parameters: {"destination" : ["Vienna", 5]},
At the bottom is the GUI we can see global event sent via the library. Events here can be filtered by adding the name of the event in the configuration file in the ignoredEvent list.
Ex ignoredEvents = ["position"];
Also events can be drawn on the map if they are specified in the drawableEvent section, using the following format:

```
target_found : {

  duration: 6000, <== duration before the event is removed

  lat: pose/latitude, <==  latitude to draw the event at

  lng: pose/longtitude", <== longtitude to draw the event at

  icon: "test.png", <== icon to use for the event

  icon_size: [40, 40], <== size of the icon

  focusEvent: true <== adjust the viewport to view the event

}
```

7.3 The Monitoring and Command Tool Running

The Graphical User Interface (GUI) can be accessed via a browser at the following address (127.0.0.1). The monitoring tool also comes with a configuration file (called 'configuration.js' under the html pages) which can be adjusted to the needs (we will go through it step by step). You will be greeted with the following screen (see Figure 21):

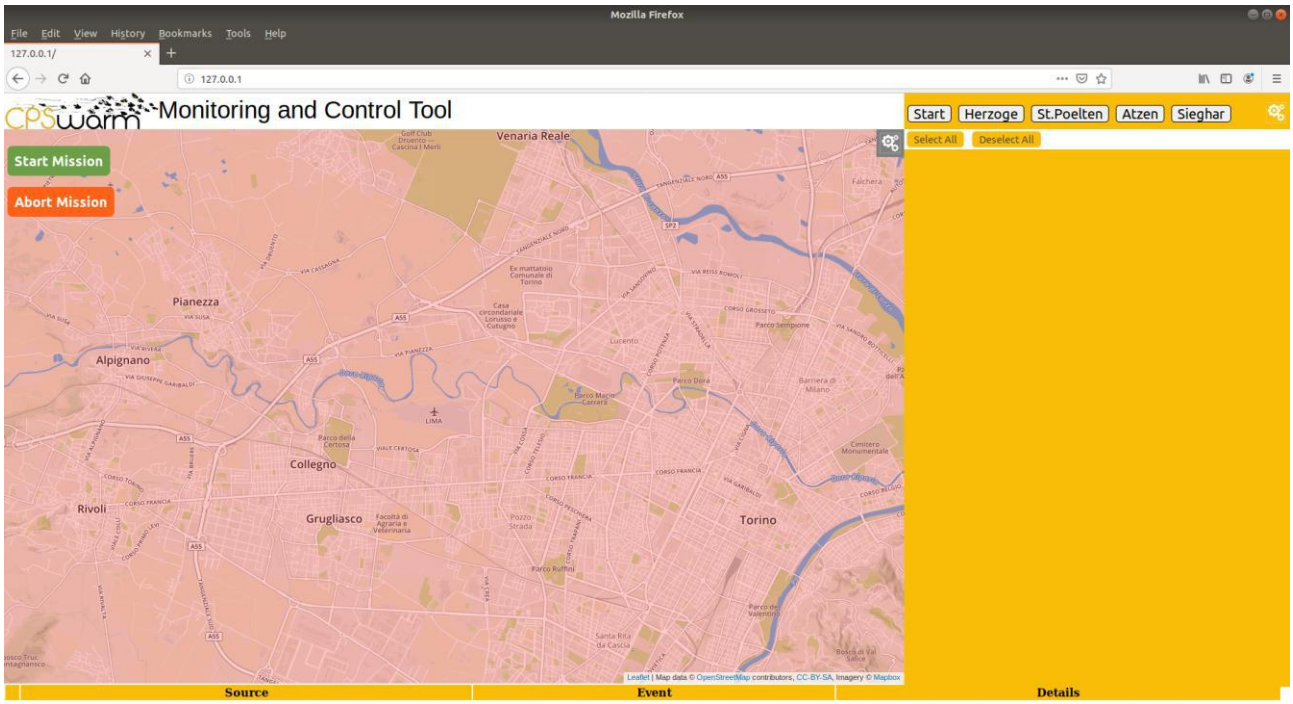


Figure 21: The Monitoring and Command Tool Screen when starting to run the tool

8 Next steps

The next steps planned are the integration of the Monitoring & Command Tool into the Logistics Scenario upon decision of Robotnik to make use of it.

9 Conclusion

This deliverable has presented the work done in Task 7.3 to design, develop and implement the first version of the CPSwarm Monitoring and Command Tool. As this task will continue till M34, the tool will be further enriched by implanting the requirements of the three scenarios. The final status and implementation will be documented in later deliverable D7.6, *Final Monitoring and Command framework*.

The Monitoring and Command Tool developed within the frame of the CPSwarm project is a first start into an area of applications that will need the following implementations / Technologies available:

1) Vehicles that are capable of autonomous driving
2) A wireless data communication link that can route safety critical control data satisfying safety requirements for control data (deterministic wireless link using WLAN as an example for TTEthernet developed within this project). However, the technology developed for wireless communication can now be used on different technologies such as e.g. V2X as well.
3) A reliable tool approach to configure, agree, start, conduct and stop the mission of platooning (the Monitoring and Command Tool).

10 Acronyms

The related reference documents to D7.5 are summarized in §2.2

Acronyms

Acronym	Explanation
API	Application Program Interface
CPS	Cyber Physical System
HTTP	Hypertext Transfer Protocol
MQTT	Message Queuing Telemetry Transport
REST	Representational state transfer
ROS	Robot Operating System
SoS	System of Systems
TTEthernet	Time-Triggered Ethernet (SAE AS6802 Standard)
WLAN	Wireless Local Area Network

11 List of figures