Using LabVIEW and ROS for Planning and Coordination of Robot Missions, the Example of ERL Emergency Robots and University Rover Challenge Competitions

Submitted: 18th March 2019; accepted: 16th May 2019

Agnieszka Węgierska, Kacper Andrzejczak, Mateusz Kujawiński, Grzegorz Granosik

DOI: 10.14313/JAMRIS/2-2019/20

Abstract:
The article presents the main functionalities and principles for operating a software for multi-robotic mission coordination developed for competitions ERL Emergency Robots 2017, as well as its adaptation during University Rover Challenge. We have started with an overview of similar software used in commercial applications or developed by other research groups. Then, our solution is thoroughly described, with its user interface made in LabVIEW and the communication layer based on ROS software. Two cases of robotic competitions proved our software to be useful both for planning and for managing the mission. The system supports the operator in teleoperation and during partial autonomy of the robots. It offers reporting on the robots' positions, Points of Interest (POI), tasks status. Reports are generated in KML/KMZ formats, and allow us to replay the mission, and analyze it.

Keywords: *Mission coordination, Planning, Robotic competitions, KML/KMZ, LabVIEW*

1. Introduction

During last few years interest in mobile robots is at a raise. Currently, robots are used not only in inspection, military or human response applications but also are becoming more and more available in industrial, agriculture or service applications. Some of the reasons for this are: the decreasing size and cost of sensors, increasing computing power of microcomputers, development of localization algorithms and growing popularity of neural network algorithms for recognition and reasoning.

An important source of robot development and social awareness about them comes from robotic competitions. The participants customize their solutions for simulated scenarios or application cases, usually taken from the real word. Hence, researchers are able to find various solutions applicable to real life situations more easily. What is more, competitions give a lot of freedom to choose the best solutions and finally compare them in front of other teams. These are then oftentimes taken to the next stage of commercialization.

In this paper we present a situation in which the participation in two international robotic contests required development of a specialized software for the group of our mobile robots. We show research done during the last two years: starting with a short description of the contests' scenarios, analysis of applications available in the market, motivation to develop our own software, detailed description of its structure and functionalities, followed by conclusions and future plans.

One of the most challenging aspects of robotic competitions is mission planning and supporting the operator in action. Proper task strategy and well planned mission time are important success factors. Software for mission management is able to solve both of these problems. Although it is mainly utilized for multi-robotic missions it is also useful in single domain tasks.

An example of such a competition is ERL Emergency Robots, where competing teams came from different higher education institutions, companies or research centers [6]. The main goal for this competition was to draw attention to mission cooperation problems between robots from different domains – aerial, water and ground – in rescue mission. The teams performance was based on the quality of multi-robots cooperation, creation of 2D or 3D terrain maps, localizing missing workers and gas leakages, delivering the first aid kit to the missing worker. The competition required land, water, and aerial robots cooperating with each other. Most of the teams came from different countries and were experts in a single domain. This required coordination not only at robotics level but also human communication level. It was a stimulus to develop software for mission coordination which could reduce communication problems, universalize contact, and increase mission efficiency in situations where several different robots or people needed to cooperate.

What also triggered the research was that each team had to provide logs in KML/KMZ format to show their actions and progress in a clear way.

During the ERL competition, operators of every robot were located in different base stations. They were not allowed to communicate directly to each other. To compensate this, our coordination software provides the mission manager and operators a clear view on the mission status, task execution progress, time remaining, mission problems, manager decisions, and upcoming tasks.

On the other hand, such software can also be used in single domain competitions to increase efficiency and quality of a given task performed by a single ro-

bot. One example is the University Rover Challenge (URC), during which robots simulate a Mars mission. The teams control the robots in tasks such as manipulation, navigation, ground sample evaluation and autonomous traversing. In most cases, missions consist of multiple smaller tasks and the detailed information related to each of the four missions are revealed by the judges only several minutes before the mission begins. Sometimes subtasks have to be made in a specific order or at a determined time. Each subtask could have different scoring. As a result it is extremely hard to memorize and later repeat the whole strategy. So an application for mission planning and coordination could improve the team's performance level. In such instance, the software is limited to planning tasks and their allocated times in proper order. It allows the operator to follow the mission plan without additional work and, in case of problems, to abort tasks with the lowest chances for success.

Finally, lack of similar applications on the market further confirmed the necessity of its development.

2. Overview of Multi-Robots Mission Manager Software

Based on extensive research, this paragraph features an overview of existing mission manager applications. Several ongoing projects show the problem is not new. Five, of the closest to our application are presented below. They became an inspiration for us and helped us determine necessary requirements. The design process was supported by different user interfaces visible on below figures.

2.1. Project Icarus

The application shown in Fig. 1 was developed under ICARUS program [7, 8] designate for use in search and rescue missions. The system consists of two levels of control stations. The higher one is responsible for defining global goals, planning and coordinating the mission progress, while the lower provides more details on how to realize the goals.

Fig. 1. User panel of ICARUS software [8]

The main features of the application are building, monitoring, updating and breaking the mission, adding or relocating resources to other mission sectors, setting inspection or patrolling zones, setting time and preferred robot type to achieve appointed goals, displaying information from robot sensors. ICARUS application used Open Street Map for adding targets, drawing sectors and pointers, adding photos, or planning robot's path (waypoints, starting and end point of the path, properties of each point like velocity or reaching accuracy).

2.2. Mobile Planer – Omron

The requirements of the factory environment are different. The first one is the map which should provide mostly corridors or easy identifiable obstacles, easy to read by the final user. One of the solutions to this problem is provided by MobilePlanner designed by Omron company [11]. The map consists of a previously scanned 2D map with 10 cm segments which inform about traversability. For segments with a low value, the traversability is easy, with high value the overcome is impossible or very difficult. In addition, users can define special zones on the map (e.g., prohibited zones, low-speed zones or one-way zones) and point of interest (e.g., loading or unloading places, battery charging location). The graphical user interface with the map is shown in Fig. 2.

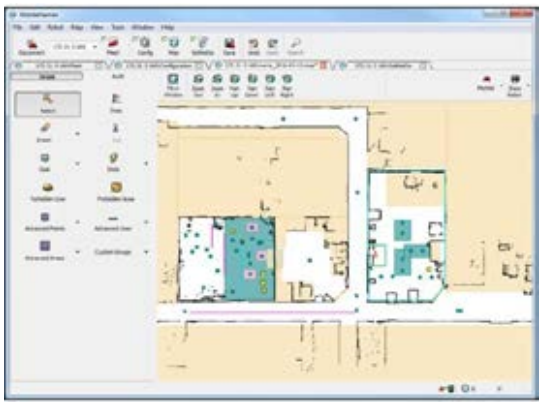

Fig. 2. The user interface of MobilePlanner – Omron [11]

Moreover, an autonomous and intelligent fleet management system is important for factory automation. The presented solution is able to manage up to 100 robots. In MobilePlanner, each target has a defined value of goal position tolerance. Users could link a task with a specific target or create sophisticated tasks from a list of predefined actions. To deal with the problem of reaching one target by several robots at the same time, the software could put the robots into the queue reducing the chances for traffic problems. Tasks can be sequential or parallel.

2.3. MIRFleet

The MIRFleet [12] is similar to Omron's solution built by MIR company (Mobile Industrial Robots) for centralized control of up to 100 mobile robots. The company's portfolio gives options to choose robots with different payload capacity or hook attachment. The web-based user interface is shown in Fig. 3. It works on any device with a web browser, and in the range of robot's Wi-Fi.

There are two ways to create a floor plan used for robot localization – the robot creates it by itself

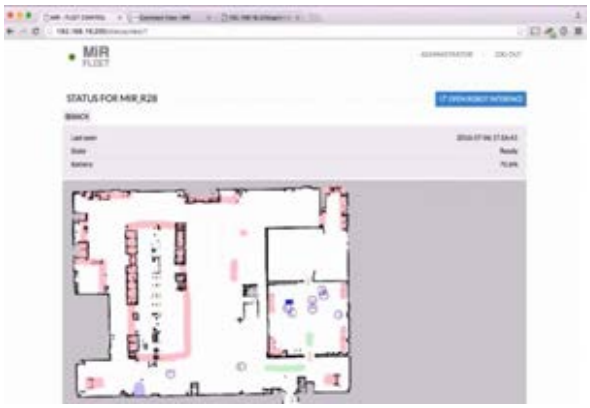

Fig. 3. Tracking single robot in MIRFleet [12]

or a map is uploaded from CAD or PNG file. The map restrictions (walls, prohibited zones, preferred drive zones, blocked zones, speed reducing, a maximum number of robots in a zone) are defined by colors. Therefore, the map may be edited in any graphic program. The map updates made by robots are shared with the whole fleet. Access to the application options is defined by user hierarchy. The additional features are alerts by SMS or e-mail. The tasks are created using a standard form including start and end time, mission name, priority (normal, high), type of the robot and additional task description. The task start time is limited to five days in advance. Tasks are divided into started, pending (robot currently not available or task start in a different time) and finished. This application is only prepared to control the MIR robots and does not give ability to manage robots of any other type.

2.4. ArduPilot – MissionPlaner [5]

Non-commercial users are looking for open source solutions. One of such is ArduPilot built on an open code. It can also be upgraded to support multi-robots and multi-domains capabilities. The user interface shown in Fig. 4 is used for mission planning with a single robot.

Fig. 4. ArduPilot control panel [5]

The software allows to edit the list of target points (add, delete, set tolerance and heading), set flight plan, save or read KML/KMZ files, display sensor data (speed, altitude or heading). Additionally, localization of the robot may be displayed online in Google Maps, Bing or Open Street Map and offline inside the application from flight logs. Additionally, the location of the robot may be displayed online in Google Maps, Bing or Open Street Map and offline inside the application from flight logs.

2.5. NVL Charter

Not strictly related to robotics is the management system used in logistics. NVL Charter (see Fig. 5) is an application for international transport management enabling tracking of specific car courses (place and date of departure, and place and date of arrival), set via points or editing orders. The main goal of this software is to suggest the most optimal route – optimizing trip time and cost.

Fig. 5. NVLCharter application window [4]

In robotics the delivery could be treated as reaching the target point by the robot, while track loading and unloading may be treated as performing the task in a given localization.

2.6. Comparative Analysis

The above overview of recently developed applications shows that most of them are very limited, regarding universality and adaptability to different robot types and environments. It shows difficulties with creating a multipurpose user interface and coordinating multi-robot systems, especially in different environmental domains. Above mentioned applications are strictly dedicated to specific circumstances.

We could divide all presented applications into an indoor and outdoor group. Applications for indoor use are focused on material handling. These applications can be integrated with factory management systems but are very limited to cooperate with different robot types. Different localization systems are used in indoor and outdoor robots. In many factories the workload and complexity of processes require several robots to be used (always with the same kinematic construction). Some proprietary applications (MIR, OMRON) use centralized fleet management system.

On the other side there are applications for outdoor use. These systems are mostly limited to display robots' locations, mission status or working zones. The applications are more often focused on mission coordination than planning repetitive tasks in time. Outdoor environment is much more diverse com-

pared to the industrial one. Therefore, applications are prepared to perform a limited number of scenarios. Two applications support multiple and heterogeneous robots. Outdoor applications seem to be developed more recently hence are less mature than the indoor ones. This lack of previous interest may have been caused by constraints given by the outdoor environment and lower demand coming from the market.

Most of the applications have easy access to all data corresponding to the current task, status and information about possible problems. Most of the systems are designed to work with predefined robots, with no option to expand it. This close structure was one of the main obstacles in adapting available software to our needs. Some of applications have automatic report generation, it may be highly important in rescue missions where time is one of the most important indicators. The closest to our approach and also at the furthest level of development was an ICARUS project.

The purpose of our project is to have an easily customizable application, able to manage different types of robots, missions, and environments (indoor or outdoor).

3. Robot Cooperation Structures

After assigning the task to a specific robot we expect that the task will be finished with a positive result and in defined time, otherwise the system should inform the user about the reason of mission failure. Efficiency may be improved by increasing the number of robots performing a specific mission or a single task, as a result we are dealing with robot cooperation. For simple tasks in most cases we are using single type robots (e.g. area monitoring), however, if the task is more complex, using robots of different types could be much more efficient (e.g. search and delivery during rescue missions – aerial robot searches and ground robots could deliver heavy equipment near to the victim). We can define four basic system types in relation to number of cooperating robots [9]:

- singular – do not belong to MRS (multi-robot systems), robots working by themselves, the task does not require cooperation,
- double – group consists of two robots cooperating with each other, robots are able to perform simple tasks e.g. moving objects,
- multi-unit – consisting of a small group of robots,
- swarms – consisting of a large group of robots.

Alternatively, we could define systems in relation to performance possibilities and morphology [9]:

- identical – the same locomotion of all robots inside the group,
- homogeneous – all robots have similar locomotion but not always the same, there are small differences in functionalities of each robot. Absence of one robot leads to the situation that the task could not be fully finished,
- heterogeneous – different way of locomotion.

Another element defining the way of cooperation is communication between robots [9]:

- all robots communicate with each other – decentralized management,
- communication between robots and control station – centralized management,
- robots communicate with each other and make decision without the base – multimaster.

Multi-robots systems might be divided in terms of ability to cooperate (see Fig. 6), according to [10] four levels could be recognized: cooperation, knowledge, coordination and management.

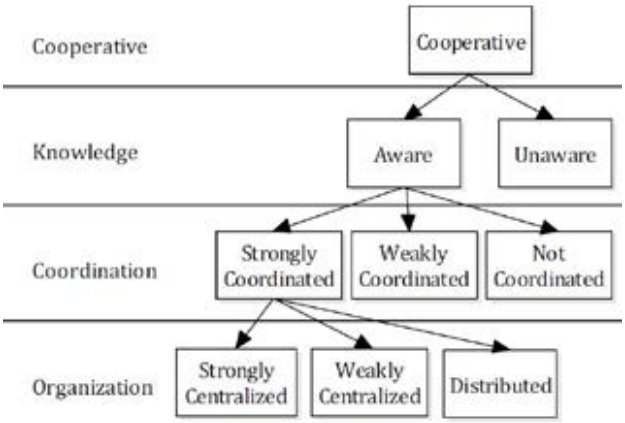

Fig. 6. Taxonomy of Multi-Robot Systems [10]

Cooperation is present only when performing a specific task requires at least two robots. Using more than two, in some cases, could be much more efficient e.g. terrain mapping. The knowledge about other robots working in cooperation could be divided into two groups – aware and unaware. Robot coordination is related to cooperation. All robots take into consideration current states of other ones in a team and make decisions about providing consistent work flow of the task. In systems based on strict coordination we could define the following organization levels:

- strictly centralized – systems have one leader,
- weakly centralized – leader function could be provided by any robot,
- distributed – any robot could make decisions by itself.

4. The Mission Planner App for the ERL Competition

As mentioned, robotic contests serve as motivation to find solutions to real-life problems. The organizers of the ERL Emergency Robots competition were inspired by the Fukushima accident (earthquake and tsunami) and created the following rescue scenario. Due to highly radioactive elements, the rescue team used mobile robots to keep a safe distance. The main problem was the communication between robot operators and robots themselves, because each control station was located in a different place that required coordination. The main goal was to search for missing people in the open space at the seaside, inside the damaged building and underwater, this required robots from different domains, such as: UGV (Unmanned Ground Vehicle), UAV (Unmanned Aerial Vehicle), AUV (Autonomous Underwater Vehicle).

One of the tasks requiring cooperation was the delivery of a first aid kit by an aerial robot to a ground robot. The UGV was supposed to send information to the UAV when the kit was needed, specifying its location. As time was a crucial factor, the task was not as simple as described, especially with the kit due on the ground shortly after the signal. Another task was finding a leaking pipe and closing the correct valve by the UGV and AUV at the same time to avoid radioactive contamination. These types of coordination between robots can be connected to appropriate task delegation, including only reconnaissance. For example, a damaged pipe outdoors can be recognized by an aerial robot or it can provide an approximate GPS position of a missing worker (mannequin-dummy) when it is out of the drone's range. This information can then be used by other robots in their missions. An important requirement was providing a report in KML/KMZ format within one hour after the mission. More details about this format will be provided in chapter 5.3.

For the purpose of the ERL Emergency Robots competition our Mission Planner application was created in LabVIEW software. It acts as an information flow coordinator between robots, so that each member of the group knows the current action status of all the others. It allows cooperation between robots in joint tasks such as examples mentioned before. The coordinator can also send common messages to other robots for work synchronization.

The system architecture is based on a central control system. The Mission Planner is responsible for event logging and is to be located on one of the computers in the base station. The user panel was created in LabVIEW and the exchange of messages is possible through the ROS node. Initially, communication between different masters in the ROS environment was possible thanks to the *multimaster_fkie* package, unfortunately, its operation turned out to be unstable, and we had to create the LabVIEW procedure acting as a multimaster (coordinator). The software consists of the three modules shown in Fig. 7.

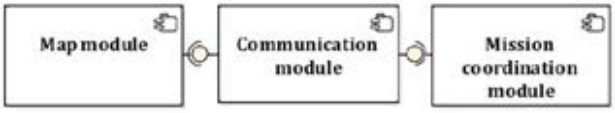

Fig. 7. The Mission Planner app modules

Each module has a separate user interface. The communication module is needed for correct operations. It is mainly responsible for gathering the data from robots and stations, including historical common messages between ROS masters providing data to other Mission Planner modules. The other modules can be started depending on their specific needs. The map module provides visualization of robots' positions on the map, stating the distance between them and GPS coordinates of waypoints. The Mission coordination module displays the mission progress and common messages from others, sends the coordinator's commands, collects navigation and mission data to further convert to KML/KMZ formats.

4.1. Communication Module

The communication module is the most important part of this software because in the past, operators of different robots in our team use separate applications to control them. The UAV and AUV users could communicate and control the robot with ROS, but for the UGV the activities were split between LabVIEW and partially ROS. This mixed system for UGV is connected with the gradual migration of Raptors Rover software from LabVIEW to ROS, currently, the LabVIEW provides a more reliable solution. Regarding our problems with the multi-master configuration and unstable connection between all masters, we decided to create a module in LabVIEW which is responsible for exchanging information between robots and stations. For this reason the communication module should be combined with different environments and meet the following requirements:

– provide the location of the robots (GPS position and orientation),
– provide information about exchanged messages,
–send messages to other robots and stations.

In our solution, communication between the robots and the managing application uses the ROS system. Information exchanged directly between LabVIEW and ROS is obtained through the TCP/IP protocol and the WebSocket technology. A separate master runs on each robot. The diagram in Fig. 8 shows the flow of the received data, which are used in the coordination and location modules.

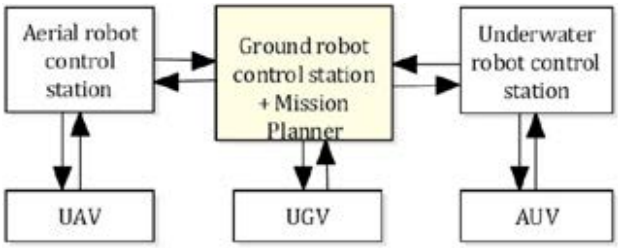

Fig. 8. Information exchange between stations and robots

Robots provide the following information to appropriate stations: GPS position and orientation, stream from the vision system, common messaging when the task was done in autonomous mode (e.g., drone reached waypoint in an autonomous way) etc. Each station is responsible for controlling the robot and communicating with the Mission Planner. They deliver the GPS position in standard *NavSatFix* message from the *sensor_msg* library. The land robot sends the odometry message from the *nav_msgs* library that was used to calculate current orientation, but for the flying robot it was its own message structure that is:

float64 heading

It was not possible to collect the GPS position for an underwater robot during its operation, therefore, it was limited to send the position of the mannequin found underwater after the robot surfaces.

The Mission Planner receives a common message from stations and robots, and sends it to the others.

For the purpose of information exchange, a custom message structure has been created:

string sender_name
string receiver_name
float64 latitude
float64 longitude
int32 valve_number
string status

The message contains information about the sender and recipient of the message, the sender robot's coordinates at the time of sending, the number of damaged valves and a status. The status field was used to send information about the currently performed activities by the robot, e.g. finding a leaking pipe, starting to deliver the first-aid-kit, or waiting for its delivery. Messages can be sent directly from robots to the Mission Planner in an autonomous task execution by an appropriate robot controller. Nevertheless, the robots mainly worked in manual mode, they could send specific messages (e.g., about current activities), which were generated by the robot's controller (as suggestion for the operator) and confirmed by human. The coordinator can sent a common message to the others to provide information and realize cooperation between robots.

4.2. Mission Coordination Module

The module is responsible for managing the course of the mission. It provides information about the mission in the form of a static list of tasks to do. This plan is prepared in advance. It means that an operator can easily follow the mission's orders and recorded data can be related to time, but cannot alter the task during the mission. Next, in a separate application, the gathered data are converted to KML/KMZ files, because logs in this format are required. The architecture of the module is shown in Fig. 9. At the beginning, the module imports and stores a list of all the tasks and OPIs (Object Point of Interest) to be found. Data about unfinished tasks and last occurred reports are kept separately when

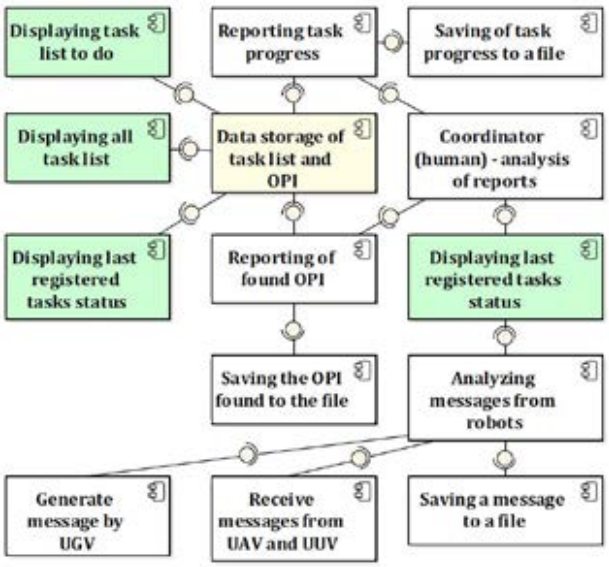

Fig. 9. Mission coordination module architecture

the status is changed. The robot can be in one of the following states for each task: setup, ready, start, done, canceled. The first three statuses require explanation. Setup means that the robot is preparing to start a task, for instance when its configuration is changing. Ready means that the robot is prepared to start its mission. In this state it is usually for a short time, however, it can take much more time, e.g. when the UGV needs the first-aid-kit and it is waiting for its delivery by the UAV. It was one of the tasks that required coordination of cooperation.

We assumed that when the UGV finished all outdoors tasks, it sends a common message to the UAV with information about its GPS position and demands for the first-aid-kit to be delivered. For this moment, the status of UGV is setup, because the next task cannot be started without the kit. The UAV receives a message, finishes the current task and comes back to the station for the kit. For now, the status is setup, because the drone is preparing for the delivery task. When the kit is mounted the status is ready to begin. Next, the status (start) occurs when the drone begins to fly to deliver the kit to the UGV. When it finished its task, the UGV takes the first-aid-kit and changes its status from ready to start. The search for the victim begins. The task is done when the kit is delivered to the victim.

In general, the Mission Planner can be located anywhere, assuming ethernet (cable or WiFi) connection with robot controllers. In the presented situation it was used in the UGV control station and t was directly connected with its controller. Both are using LabVIEW as software environment and the position of the UGV is privileged (the UGV can send common message to the UAV and AUV, while the communication module provides messages from other robots). Nevertheless, the universal structure allows system extension for other robot controllers. The Mission coordination module is responsible for collecting data from each common message, task status changes, as well as each robot's GPS position and found OPI. The human coordinator analyzes the common messages from the robots and manually reports the progress of the mission. When the task required searching damages or victims the coordinator reports twice. One time to update mission progress and the other to confirm the found OPI.

User interface used by coordinator is shown in Fig. 10.

- 'Subtask to do' (a) – displays a list of all tasks to be performed along with their current status,
- 'All subtask Status' (b) – displays the status of all tasks,
- 'Subtask registered event' (c) – displays information about registered events: UTC time, operating mode, robot name and coordinates, content, status and additional information about the task,
- 'OPIs found' (d) – a table with a list of found items, contains information: time of finding the item, name of the robot, necessary information about the item and the name of the file with its picture,

Fig. 10. Mission coordination User Panel

- 'Task Reporting' (e) – a form for reporting completed tasks with a selection list containing the names of unfinished tasks,
- 'Sending and displaying a common message' (f) – received common messages sent by robots are displayed in the table. The operator analyzes them and reports (manually) in (e). The form allows you to select the sender and recipient of the message. This is necessary because it is not known in which base station the management software is located, but default is UGV.

4.3. Map Module

The idea of creating an offline map module for the robot's location is related to another competition, the University Rover Challenge, during which there is no access to the Internet. For the purposes of the ERL competition, this application has been expanded to include the display of the positions of several robots and the distance between the two selected machines. The architecture of the Map module to present the robot's location is shown in Fig. 11.

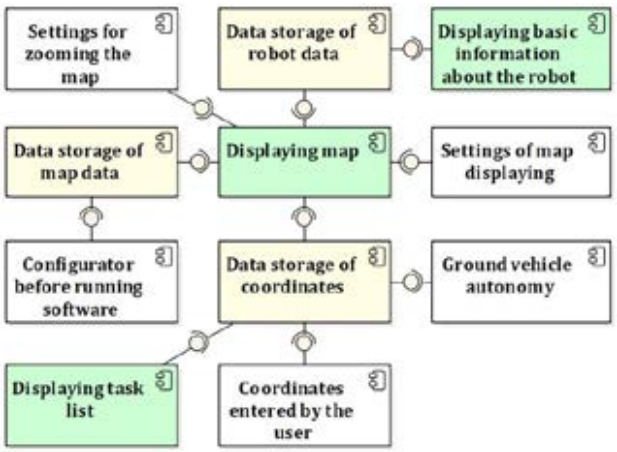

Fig. 11. Map module architecture

Data storage of robot data contains information about the current GPS position and the orientation of each robot which are required to display the appropriate position on a map. The map is loaded into the memory as a .jpg file with a scale that allows the user to set two markers and enter the distance between them. For correct functioning of the application, a map created in the UTM (Universal Transverse Mercator) should be uploaded to one zone. Furthermore, before running the software, the user is responsible for its appropriate configuration. That is an indication path to its map and logs, set source of data only to UGV (e.g. odometry and orientation or GPS), reference GPS coordination point and Earth projection parameters if different than default setting which is WGS84.

Subsequently, the first action taken after starting the application should be scaling the map. For a non-scale map, the operator indicates any two points on the map and enters their GPS coordinates. Based on the distance determined, the length of one pixel is determined. User must enter coordinates and indicate the location of the reference point. The location of the robot on the map is determined by the distance in meters between the reference point and the robot's position, which is then scaled and calculated in pixels. The accuracy of displaying items on the map depends primarily on the accuracy of the map's ratio and the correct reference point display.

The next important point is the ability to enter GPS coordinates in the following formats: DMS (degrees, minutes, seconds), DM (degrees, decimal minutes), D (decimal degrees). It is very useful because during the ERL competition, the GPS was given in DMS format but during URC it was DM. Additionally, points can be added by pointing on the map, loading the file with coordinates or giving the distance of the point from the robot in meters in the east and north direction. Points can be deleted from the list. The list of coordinates is responsible for displaying the robot's distance from the entered points.

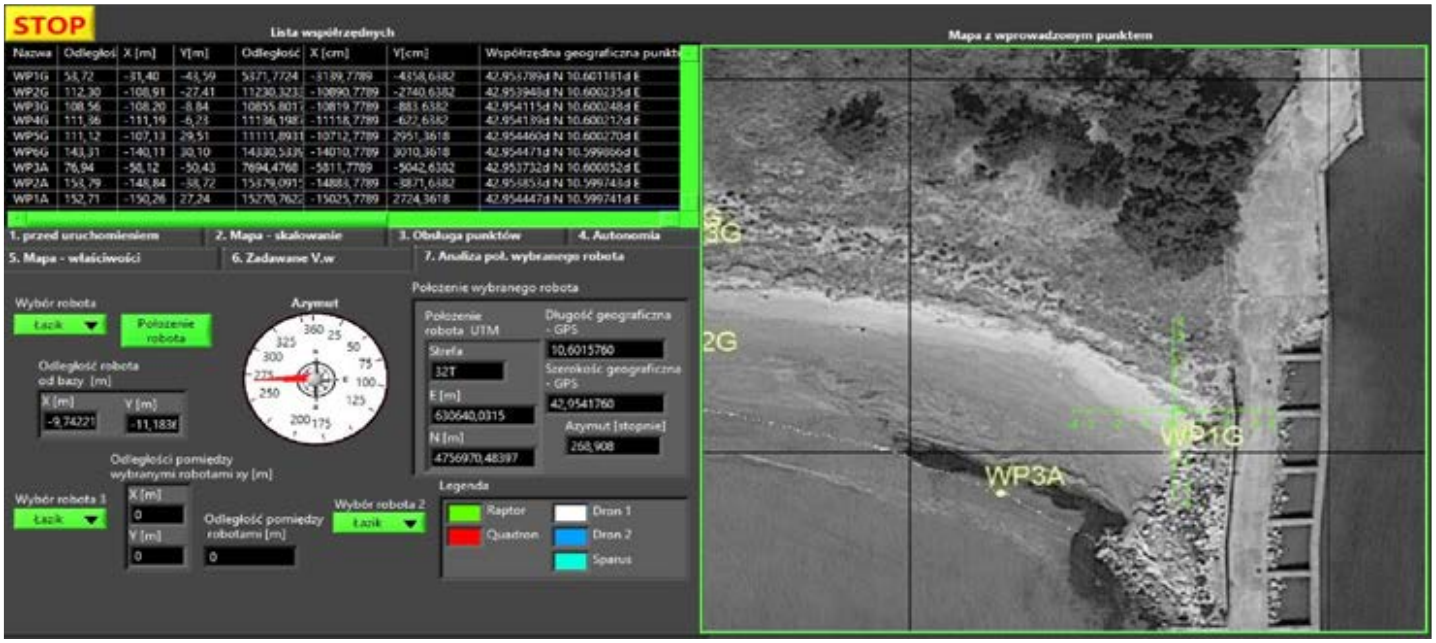

Fig. 12. User Panel of the Map module

The appearance of the user interface is presented in Fig. 12. On the left side of the panel there are configuration options, the upper table provides data with the distance between the current tracking robot and each GPS point. The presented panel displays basic information about the position and orientation of tracking the robot and its distance to another chosen one. This panel is used most frequently during the mission. On the map the robots are all visible but only the one tracked is highlighted .

5. Application for Coordinating Single Robot at URC Competition

The first edition of the URC competition was launched in 2006, for teams of international students. Each team designed and built their own version of the Mars rover which competed in 4 missions. In three of them, the operator was responsible for memorizing a lot of tasks during the mission. One of them was Extreme Retrieval and Delivery during which the robot had to deliver items between locations indicated by GPS coordinates. Scoring points for a task could be achieved separately by picking up or dropping the item, successful delivery, and finding indicated elements. The tasks may seem very simple for humans but for robots they were quite difficult (assuming teleoperation and delays). For example, picking up an item may take a few minutes. The other mission was called Science and was divided into two parts. In the first one the robot was exploring the terrain, gathering and testing samples, and looking for rocks indicated by the judges.

The application for the URC competition is an adaptation of the same software, although it is limited to mission planning, management and a single robot tracking. The application could be used for any robot sending GPS coordinates and heading direction information via ROS. The application section responsible for the mission's plan could work offline without physical connection to the robot. In this version the user interface was simplified and some new functionalities required at URC were added:

- displaying the information about delays in the mission plan
- the remaining mission time
- possibility to change tasks order during the mission
- ability to read and save the mission plan form to a file
- ability to edit plan during the mission
- KML/KMZ report generation based on recorded data

The mission report can be presented based on data from autonomy tests and Science Task.

5.1. Software Architecture

The application is still based on LabVIEW environment connected with ROS. Communication takes place in the same way as at the ERL competition. User interface and architecture was simplified and some external modules integrated. In this version we can distinguish four main parts: communication, map, mission management and report generation module. The software architecture is based on events and is shown in Fig. 13.

Firstly, communication with the robot is required to provide all needed data to proper action of software. GPS coordinates and robot orientation are received by standard ROS nodes (NavSatFix from library sensor_msg and Odometry from *nav_msgs*). The module map has the same functions as the module described in chapter 4.3. It was extended only by the possibility to import the map configuration from a file. The mis-

sion management was changed significantly and limited to one robot, and the report module is completely new. More information about the last two modules will be provided in the next chapters.

Due to the simplified startup configuration: the scale and position on map, OPI or mission configuration are imported from a file. It is important wherever the robot operator has very limited time for system setups, such a situation appears for example during competitions when the operator has just several minutes for system setup. The possibility of importing a text file with a predefined mission plan reduces the number of mistakes.

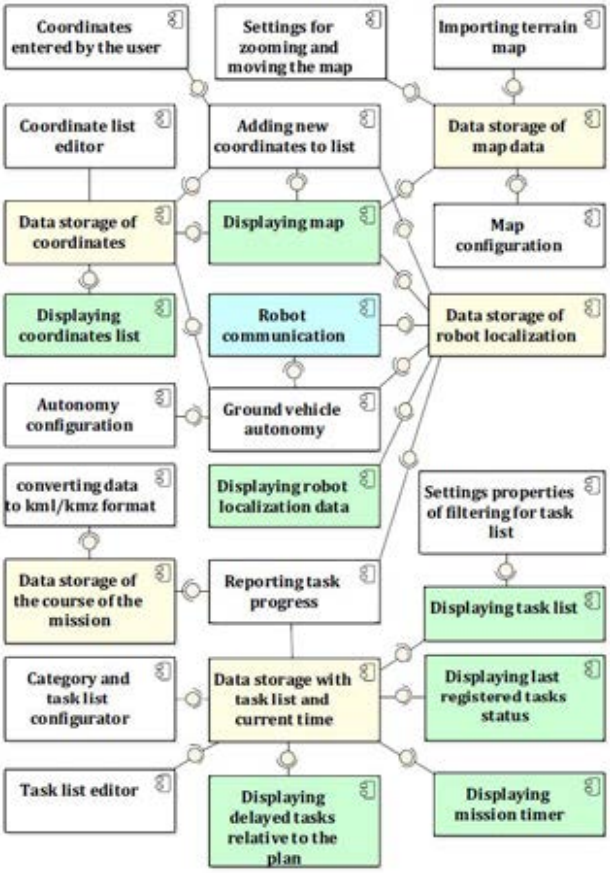

Fig. 13. Software architecture

In the user interface some significant changes have been made compared to the ERL software, all configuration windows were merged together. The map was placed on the left side of the window, a clock counting remaining mission time with a mission delays indicator was added above the map, as presented in Fig. 14.

Fig. 14. Application front-end

5.2. Mission Management Module

The main requirement for this module is to help the operator to follow the mission plan. It provides information about the mission progress and occurring delays. Each task can be edited or shifted during the mission depending on the current situation or possibility. All actions taken are registered in order to prepare a report.

Collecting data is crucial for further analysis and creating a final report. Each log contains UTC (Universal Time Coordinated) time, GPS position and heading of the robot during action. Moreover, the logs connected with taken actions provide information about the task: description, status, type of robot work, user comments, priority, and photo title when required. Without the software, users had to save each photo separately. The stored data provides information about the mission's start time, the configuration of each task and the status of the robot.

Firstly, the configuration of the mission is required. The mission configuration window is shown in Fig. 15. This configuration panel allows the user to import or export mission settings. It is very useful, because it allows planning to be started much earlier. The plan is limited to the sequence of specific tasks, required deadlines and the global task that should be done during the whole mission (e.g. searching for some objects). In terms of creating new tasks, the operator has to fill out a form consisting of: the task description and duration, type of job (autonomous, semi-autonomous, manual), priority (7 levels), attribution to one of the categories created by users and information whether taking a photo is required.

Fig. 15. Configuration list

The mission configuration is displayed above the form in a table which contains all collected information about created tasks and the last reported status. The next benefit is the possibility to change the order of tasks or delete them. Changing the configuration of each task is possible through the same form through which have added them. The option of filtering tasks in a table is available by selecting: task status, type of job, priority, required OPI, or user category.

Moreover, reporting the completion of a task should be as simple as possible, because the operator has many things to control. Our software limited this

operation to selecting the tasks from a list and entering additional information only if it's required. The reporting form is shown in Fig. 16.

Fig. 16. Form for reporting the tasks completion

Choosing a specific task is followed by filling out a form with the type of job and the next predicted status based on current information. In the application we distinguish the following states: ready to start, start, in progress, done and canceled. The user is able to change the robot's mode or status if it is required. For tasks requiring taking a photo, an additional textbox appears asking to fill out its title or generate a default name. Mostly, the operator selects only the task and confirms it. In addition task lists are connected through filters. That means that only tasks visible in a table can be chosen. Last 50 reports of tasks are visible in one of the page tab in a table with the same headers as in the configuration table. Reports are sorted from the most recent to the oldest.

Finally, the module takes into consideration the current time and time when the last report of each task occurred in order to evaluate if the plan is realized. The application highlights the delays corresponding to task start time and end time. This feature is shown in Fig. 17.

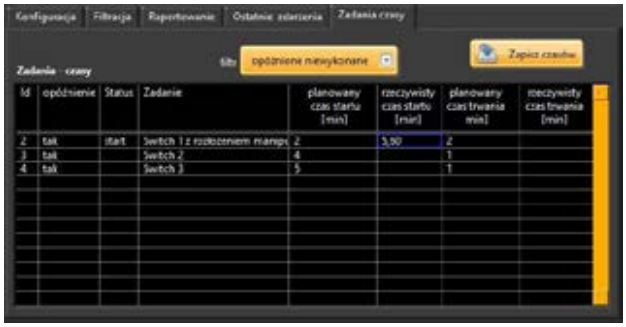

Fig. 17. Table with planned and real start times

Delay calculations are skipped for the global task which can be completed at any moment of the mission. For other tasks their deadlines are the basis for this calculation, the expected start time of a specific task is the sum of deadlines of all of the preceding. During the mission, the order of the task could be changed, in such a situation software automatically recalculates mission time. The delays corresponding to start time in the plan are counted when a specific task has not been switched to "in progress" or "done" status before the assumed time. The duration of the task is counted from "in progress" status to "done" status.

5.3. Report Generating Module

The report generating module is responsible for preparing a report of the mission course which should be readable for the end user. It involves a few information to display: the GPS locations of POI and the robot, photos of interesting items in chosen place, actions taken during the mission at a specific moment, and their status. In this chapter we will show the most important feature of this module which is the possibility to display the whole mission as an animation which provides clear information about actions taken in time.

In order to provide a suitable report, the module converts logged mission data to KML (Keyhole Markup Language) or KMZ (KML file after compression to ZIP file) format. Logged mission data is saved as a .txt file which contains information about UTC time of the events, GPS position and orientation of robot, description, status, type of job and additional information of each task, file name of photos in POI or searched items. The KML format allows to display detailed geographical data in an international standard Open Geospatial Consortium (OGC). The details for creating the report could be found in [18]. The KML format could be used with Google Earth and World Wind [19].

The generated report consists of one KMZ file and three KML files. KMZ file is required because it contains necessary photos and KML file provides GPS position and time when the photo was taken. The KML files are responsible for displaying: only waypoints, robot GPS navigation data with time and changing the status of task on time during mission.

In the beginning, we will describe two KML files (waypoint, navigation data) which provide the result presented in the Fig. 18.

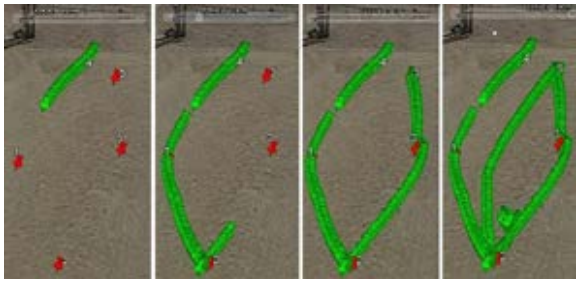

Fig. 18. Mission report with target points and robot path

For displaying the red marker on maps in waypoint KML file we have used the following template script (target points):

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
<Folder>
 <name>Waypoints</name>
 <Style id="WP">
  <IconStyle><color>ff0000ff</color></IconStyle>
 </Style>
 <Folder>
 <name>UAV-1</name>
 <Placemark>
  <name>1</name>
```

```
  <styleUrl>#WP</styleUrl>
  <Point>
   <coordinates>-
118.175659,33.923909,0</coordinates>
  </Point>
 </Placemark>
 </Folder>
</Folder>
</Document>
</kml>
```

In order to display more waypoints the script should be extend only by:

```
 <Placemark>
  <name>1</name>
  <styleUrl>#WP</styleUrl>
  <Point>
   <coordinates>-
118.175659,33.923909,0</coordinates>
  </Point>
 </Placemark>
```

The script for displaying a robot's path was almost the same. One significant difference is in <Placemark> which doesn't have a name but contains timestamp and heading. The <Placemark> structure is as follows:

```
 <Placemark>
  <description>Heading: 0.00</description>
  <styleUrl>#V1</styleUrl>
  <TimeStamp>
    <when>2018-05-31T17:45:02Z</when>
  </TimeStamp>
  <heading>0.00</heading>
  <Point>
   <coordinates>-
118.175659,33.923909,0</coordinates>
  </Point>
 </Placemark>
```

Displaying a green marker is realized by define style id in <Folder>:

```
 <Style id="V1">
  <IconStyle><color>9900FF04</color></IconStyle>
 <LabelStyle><color>FF00FF04</color></LabelStyle>
 </Style>
```

The next KML file is responsible for displaying mission progress and the final status. The file contains, for each reported change of task's status, the proper <Placemark> which includes information about its name, task description, timestamp and position when report occurred. These points are visible in the Fig. 19 e.g. the name of point "4s" means that task no. 4 has started in that place. <Placemark>s are grouped by task id in separate folders and their structure is as follows:

```
 <Placemark>
  <name>1d</name>
  <description>
    Mode:manual
    Task: gathering sample
  </description>
  <TimeStamp>
    <when> 2018-05-31T17:50:49Z</when>
  </TimeStamp>
  <Point>
  <coordinates>-
110.79298,38.3983,0</coordinates>
  </Point>
 </Placemark>
```

One advantage of this file is providing a table (Fig. 19) which sums up all the tasks and displays the last status of each task after the mission is located in the description field in <Placemark> without timestamp, which is located in the start position of robot. The description provides the legend of meaning for each letter used to describe a point name.

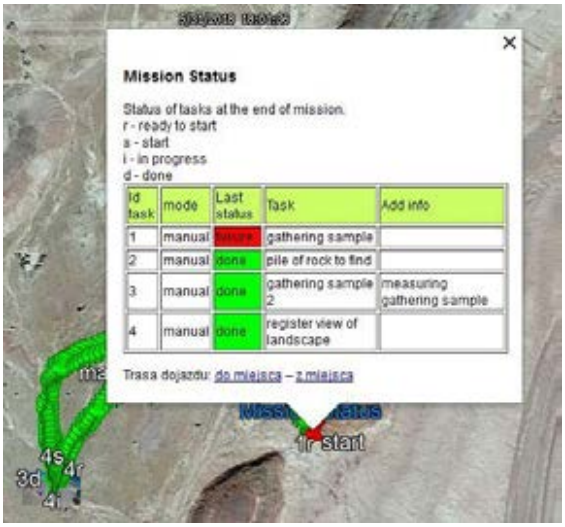

Fig. 19. Task status in mission "Science Task"

For this feature we have to add the following script:

```
<table border="1">
 <tr style="background-color: rgb(204, 255, 102);">
  <td>Id task</td>
  <td>mode</td>
  <td>Last status</td>
  <td>Task</td>
  <td>Add info</td>
 </tr>
 <tr>
  <td>1</td>
  <td>manual</td>
  <td style=»background-color: red;»>failure</td>
  <td>gathering sample</td>
  <td></td>
 </tr>
 <tr>
  <td>2</td>
  <td>manual</td>
  <td style=»background-color: lime;»>done</td>
  <td> pile of rock to find</td>
      <td></td>
    </tr>
    //and the following rows
   </table>
```

The last report file is a KMZ file and its aim is to create proper <Placemark> with photos. On the map the small pictures are visible but after opening them

the user can see a bigger photo with description. Final result are shown in Fig. 20.

Fig. 20. Photo visualization in Google Earth

To generate a KMZ file with photos, they have to be placed in "zd" directory in the same directory tree as KML file. The structure of the file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
<Folder>
 <name>Object recognition Information</name>
 <Folder>
  <name>UGV-1</name>
  <Placemark>
   <description>
    <![CDATA[<img style="max-width:500px;" src="zd/
pile.jpg">]]><br>pile of
     rock</br>
   </description>
   <TimeStamp>
    <when>2018-05-31T17:47:59Z</when>
   </TimeStamp>
   <Point>
    <coordinates>-
110.792930,38.398331,0</coordinates>
   </Point>
   <Style>
    <Icon><href>zd/pile.jpg</href></Icon>
   </Style>
  </Placemark>
</Folder>
</Folder>
</Document>
</kml>
```

6. Conclusions

This paper presents the Mission Planner – our application to support a mission coordinator, a person preparing the structure of the robotic mission, keeping an eye on mission progress, time and objectives, and finally preparing report. Our approach is based on LabVIEW technology, connected via ROS-bridge with ROS system. This choice was strongly based on expertise we had in LabVIEW used to control our first mobile robot – Raptors Rover. Then, this graphical software is very helpful to simultaneously build a core application and user interface, which was very important to quickly prepare a useful and supportive environment. Mission Planner is composed of several modules and was used for planning and management tasks for multi domain robotic mission. The application has graphical user interfaces to display the mission's progress and broadcast its status to all robots and control stations connected. Commands are sent via text messages, therefore, robots in a team do not need to use LabVIEW.

The application was tested on two robotic competitions ERL Emergency and URC. Huge differences between competitions, forced us to customize the software but with LabVIEW again it appeared to be quite fast. Good documentation and a lot of toolboxes was beneficial.

The application created on ERL allowed to coordinate three different robots, working in different environments which were controlled by various software. In that time we managed to quickly create a system architecture template and carry out successful tests. We have used a simple solution of sending predefined text messages to inform other operators about the mission's progress or problems. Using a list of such messages further supported the mission controller in reporting the status of the running task. Therefore, in stressful situations the operator is able to report faster on a task and the risk of mistakes is decreased. Tasks are usually very complex (most teams are not able to finish all of them), which means that reporting should be simplified as much as possible, and autonomous tasks should be recorded automatically by the robot. The latter functionality is very limited in our system and it is still the operator's responsibility to confirm the robot's and task's status. Also the introduction of new robots is too impractical now, because it requires changes in program code and adding new lines of communication to other ROS Masters. While for several robots this is not a problem, for a larger number of them this procedure will be boring and too long.

In the second version (for URC) we have successfully implemented some improvements, which eliminated or reduced disadvantages mentioned before. Software was limited to the management and control of the mission with one robot, according to contest rules. Simplifying the user interface reduces the time required to complete a single report to only a few seconds. A report of the whole mission without images can be generated within a few minutes after its completion. A report with photos is generated longer, because filenames are verified with names in reports. This problem can be resolved by adding a camera view to the software and when a task report is created, the selected camera view is automatically saved as an image. The timer and the delay indicator are important elements showing the remaining time to complete the mission and failure to comply with previously agreed plan. This supported the operator in controlling the plan and re-organizing tasks order. Additionally, having the list of tasks to do, during the complicated mission, the operator will not forget about important tasks.

7. Future work

The experience gained during the competitions, and verification of our application, led us to further development of a distributed mission planning system. Here are the most important requirements we have identified and will use in the next version of the application.

The current approach forces the operator to have an additional computer (Win OS) for our application, it increases the cost and amount of appliances operated by the user. This is an important disadvantage especially when robots have to be controlled by only one person. Therefore, the next app will be based on web technologies and ROS. Running a local server will allow us to use the application from any device equipped with a browser, without the need to install additional software on the client's side. ROS would be responsible for receiving and delivering the necessary data to robots. For further research and software development, we will continue with the simulation platforms Gazebo or V-rep [4] and turtlebot robots.

The application should work stably after losing connection with any device (station or robot). Time synchronization will be introduced, which will allow to update the mission plan for all robots and stations without any time delays or shifts. To exchange information between devices located close to each other, it is worth to consider alternative ways of communication, which were described in [14].

We consider the solutions with a separate planning layer, a higher level will be responsible for the implementation of the assumed goals, while a separate lower level for planning the trajectory of robot motion [2,3].

Moreover, robots connected to the system would provide information about the equipment and modes of implementation of particular tasks. The system configuration should be minimized, therefore, based on robot equipment and the configuration of the mission, the system could assign robots automatically to individual tasks so that the plan can be optimally implemented. Each task should have a specific cost of execution depending on the selected robot. Due to the dynamic changes in cost, it may be a good idea to use the LRT Switchback algorithm [2]. Reporting a malfunction or loss of communication should result in a dynamic reconfiguration of the plan. In some situations, it turns out that binary logic is not enough and it is necessary to specify additional states such as contradiction or unknown. To this end, it is worth using 4QL [1].

Another important possibility is displaying the robot's location on the terrain map in real time together with Lidar (or other sensors) readings – such information can be helpful when entering or exiting building or traveling in narrow spaces.

As important as planning and coordination is reporting about mission results – especially in search and rescue or inspection mission. Therefore we want to provide the ability to generate a report not only in KML/KMZ (requiring the GPS position), but in some others user friendly formats especially for indoor missions. The above mentioned ideas of using simulators and sensor readings (in order to locate on the indoor map) will be of crucial importance.

Acknowledgements

This work was partially supported by the Ministry of Science and Higher Education under grant No. MNiSW/2017/78/DIR/NN2.

AUTHORS

Grzegorz Granosik* – Lodz University of Technology, Institute of Automatic Control, Stefanowskiego 18/22, Lodz, 90-924, e-mail: granosik@p.lodz.pl, www: www.robotyka.p.lodz.pl.

Agnieszka Węgierska – Lodz University of Technology, Institute of Automatic Control, Stefanowskiego 18/22, Lodz, 90-924, e-mail: agnieszka.wegierska@p.lodz.pl, www: www.robotyka.p.lodz.pl.

Kacper Andrzejczak – Lodz University of Technology, Institute of Automatic Control, Stefanowskiego 18/22, Lodz, 90-924, e-mail: kacper.andrzejczak@p.lodz.pl, www: www.robotyka.p.lodz.pl.

Mateusz Kujawiński – Lodz University of Technology, Institute of Automatic Control, Stefanowskiego 18/22, Lodz, 90-924, e-mail: mateusz.kujawinski@p.lodz.pl, www: www.robotyka.p.lodz.pl.

*Corresponding author

REFERENCES

[1] Ł. Białek, A. Szałas, A. Borkowski, M. Gnatowski, M. M. Borkowska, B. Dunin-Kęplicz, and J. Szklarski, "Coordinating multiple rescue robots", *Prace Naukowe Politechniki Warszawskiej. Elektronika*, vol. 194, 2014, 185–194.

[2] M. Przybylski, "Hierarchiczne planowanie akcji robota usługowego w środowisku dynamicznym", *Prace Naukowe Politechniki Warszawskiej. Elektronika*, vol. 194, 2014, 471–480.

[3] K. M. Mówiński and E. Roszkowska, "Sterowanie hybrydowe ruchem robotów mobilnych w systemach wielorobotycznych", *Prace Naukowe Politechniki Warszawskiej. Elektronika*, vol. 195, 2016, 255–264.

[4] "K. Dorer, Applications of multi-agent systems in logistics", gki.informatik.unifreiburg.de/teaching/ws0809 /map/mas_lect10_dorer.pdf. Accessed on: 12.08.2019.

[5] "Ardupilot, Mission Planner Home", ardupilot.org/ planner/index.html. Accessed on: 12.08.2019.

[6] "European robotics league", eu-robotics.net/ robotics_league/erlemergency/about/index.html. Accessed on: 12.08.2019.

[7] "ICARUS research project", www.fp7-icarus.eu/. Accessed on: 12.08.2019.

[8] S. Govindaraj, P. Letier, K. Chintamani, J. Gancet, M. N. Jimenez, M. Á. Esbrí, P. Musialik, J. Bedkowski, I. Badiola, R. Gonçalves, A. Coelho, D. Serrano, M. Tosa, T. Pfister, and J. M. Sanchez, "Command and Control Systems for Search and Rescue Robots", *Search and Rescue Robotics – From Theory to Practice*, 2017, 10.5772/intechopen.69495.

[9] M. Garzón, J. Valente, J. J. Roldán, D. Garzón-Ramos, J. de León, A. Barrientos, and J. del Cerro, "Using ROS in Multi-robot Systems: Experiences and Lessons Learned from Real-World Field Tests". In: A. Koubaa, ed., *Robot Operating System (ROS). Studies in Computational Intelligence,* Springer, Cham, 2017, 10.1007/978-3-319-54927-9_14.

[10] L. Iocchi, D. Nardi, and M. Salerno, "Reactivity and Deliberation: A Survey on Multi-Robot Systems". In: M. Hannebauer, J. Wendler, and E. Pagello, eds., *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, BRSDMAS 2000. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2001, 9–32, 10.1007/3-540-44568-4_2.

[11] "OMRON, LD Series – Self navigating Autonomous Intelligent Vehicle", automation.omron.com/en/us/ products/family/LD. Accessed on: 12.08.2019.

[12] "MiR, Mobile Industrial Robots", www.mobile-industrial-robots.com/en. Accessed on: 12.08.2019.

[13] "Google Developers, Keyhole Markup Language", developers.google.com/kml. Accessed on: 12.08.2019.

[14] "NASA, WorldWind", worldwind.arc.nasa.gov. Accessed on: 12.08.2019.