BŞEÜ Fen Bilimleri Dergisi
9(1), 514-540, 2022

BSEU Journal of Science
https://doi.org/10.35193/bseufbd.1089501

e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

Araştırma Makalesi - Research Article

Development of Virtual Robotic Laboratory and Materials for Education and Research

Eğitim ve Araştırma Amaçlı Sanal Robotik Laboratuvarı ve Materyallerinin Geliştirilmesi

Alim Kerem Erdoğmuş[1*], Uğur Yayan[2]

Geliş / Received: 18/03/2022 ***Revize / Revised: 11/05/2022*** ***Kabul / Accepted: 11/05/2022***

ABSTRACT

Robotic era is near, has never been before. In this new era, robotic education must have quality and accessibility for everyone. In this study, a disruptive technology for future of distance education based on virtual robotic laboratory (U-Plat) is presented. The created robotics education platform allows users to access a virtual robotics laboratory and training programs using internet interfaces. This environment allows them to control simulated robots, analyze and evaluate the results of experiments performed. U-Plat gives researchers, students and people working in the field of robotics the chance to learn Robot Operating System (ROS) and increase their knowledge with robotics training from beginner to advanced level. Also, U-Plat gives researchers opportunity to try their codes in online platform. Compared to traditional learning methods, U-Plat can be seen as an easier and more effective learning/teaching platform for ROS exercises, as it is an open source system. The user experience has been tried to be provided in all training tools with a very simplified and easily understandable interface. All designed training tools are provided through a web interface supported by advanced cloud systems and Docker container infrastructure. As a result, U-Plat is a novel robotic learning platform that can lead future laboratory research and distance robotic education.

Keywords- Lifelong Learning, Distance Education, Virtual Robotic Laboratory, Robotics, Distributed Learning Environments

ÖZ

Robotik çağı, daha önce hiç olmadığı yakın. Bu yeni çağda robotik eğitim, herkes için kaliteli ve erişilebilir olmalıdır. Bu çalışmada, sanal robotik laboratuvara (U-Plat) dayalı uzaktan eğitimin geleceği için yıkıcı bir teknoloji sunulmuştur. Oluşturulan robotik eğitim platformu, kullanıcıların internet arayüzleri kullanarak sanal bir robotik laboratuvara ve eğitim programlarına erişmelerini sağlar. Bu ortam, simüle edilmiş robotları kontrol etmelerine, gerçekleştirilen deney sonuçlarını analiz etmelerine ve değerlendirmelerine olanak tanır. U-Plat, araştırmacılara, öğrencilere ve robotik alanında çalışmalar yapan kişilere Robot İşletim Sistemini (ROS) öğrenme ve başlangıç seviyesinden ileri seviyeye robotik eğitimi ile bilgilerini artırma şansı verir. Ayrıca U-Plat, araştırmacılara kodlarını çevrimiçi platformda deneme fırsatı verir. U-Plat, geleneksel öğrenme yöntemleriyle kıyaslanacak olursa, açık kaynaklı bir sistem olması sebebiyle, ROS alıştırmalarının daha kolay ve etkili bir şekilde öğrenme/öğretme platformu olarak görülebilir. Eğitim için gerekli tüm araçlar, kullanıcı deneyimleri göz önünde

[1*]Corresponding author contact: kerem.erdogmus@inovasyonmuhendislik.com (https://orcid.org/0000-0001-5111-5965)
Electrical Electronics Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, İnnovation Engineering, ESOGU Teknopark, Eskişehir, Turkey
[2]Contact: ugur.yayan@ogu.edu.tr (https://orcid.org/0000-0003-1394-5209)
Software Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskişehir, Turkey

bulundurularak sadeleştirilmiş bir arayüz ile sunulmaktadır. Bu araçlar, bulut ve Docker konteyner sistemleri kullanılarak, web arayüzü üzerinden kullanıcıya sunulur. Bu sistem, dört farklı türde mobil robotun görev bazlı eğitimleri ile birlikte sunulmaktadır.Sonuç olarak, U-Plat gelecekteki laboratuvar araştırmalarına ve uzaktan robotik eğitime liderlik edebilecek yeni bir robotik öğrenme platformudur.

Anahtar Kelimeler- Hayatboyu Öğrenme, Uzaktan Eğitim, Sanal Robotik Laboratuvar, Robotik, Dağıtımlı Öğrenim Ortamları

I. INTRODUCTION

Nowadays robotic systems are started to use in many areas. Unmanned aerial vehicles, autonomous guided vehicles or combines, smart cars, robots for rehabilitation, reconnaissance, research and education are just a few of the areas of use [1]. On the other hand, we are facing with another problem which is the training of qualified engineers for developing and operating robotic systems. The usage of the real robots also increases the efficiency of the learning. However, the students could not access to the real robots due to Covid-19 restrictions, high number of students taking lectures, the cost of robotic kits or the time constraints of workers [2-5]. In recent years remote and virtual laboratories are used to solve these problems dramatically increased [3,4]. Virtual and Remote laboratories carry the traditional laboratories to the next level and give a great impact on education and research areas for students and researchers. Especially, Virtual laboratories give opportunity for complementary analyses for new teaching methods. Systems created for accessing robotic systems through a digital environment have been created to facilitate the work of students and researchers working in this field. These systems allow researchers and students to evaluate their work and validate it online with a virtual simulation environment very close to real-world applications. Case in point, if researchers want to test planet robot, they require virtual planet environment that simulates space conditions. With enhancing virtual laboratories comfort in developing new technologies and testing new technologies are increasing.

In the last decade, many national and international "Remote/Virtual Laboratory" projects are implemented for many different engineering disciplines to teach the basic concepts. Although its general use, many different terms are used to describe these laboratories and there is no general term for these labs. In this case, difference between virtual and remote laboratories is explained by [6] as: Remote laboratories involve the operation of real/physical equipment but the experimenters are away from physical equipment. Whereas Virtual laboratories simulates laboratory environments and students/researchers can access these experiments via virtual laboratory environment.

It seems that many different universities or organizations are working to take the place of remote / virtual laboratories in the future of education. Studies show us to engineering is the most important discipline for remote / virtual laboratories. In engineering technical area, there are issues that contains several classes inside the concepts. One of these issues is the robotics which is a topic that mostly studied and the one of most popular issue in the area of engineering. In this regard, robotics studies subjected to a standstill because of universities resources' (personnel, laboratory material) insufficiency, excess number of students, cost of robotic training materials. Also, Impact of the pandemic on educational organizations creates a great barrier in physical laboratories [6]. Again, this especially added importance to virtual laboratories and online robotic education have gained momentum to overcome these barriers in education [7].

The proposed virtual robotics laboratory (U-Plat) provides an ideal working environment for students, researchers and lifelong learning users who want to increase their knowledge in robotics and software. This study is designed to be used for research purposes as well as education [8]. The created robotic software learning system allows users to access the robotic laboratory virtually over the web, control it through interfaces and various tools, analyze and evaluate the experimental results. Also, EvaMars robot exploration missions are explained and expected learning outcomes are given. Beyond EvaMars, EvaSec, ATEKS and AGV-OTA, U-Plat has mobile robot experiments like EvaRobot which is educational and research purposed robot, a differential drive Autonomous Guided Vehicle (AGV) and Swedish wheel AGV.

II. RELATED WORK

In the literature, there are different remote and virtual laboratory studies proposed. for each university robotic lab, for each robot, and for each focus of study, these works are remains only local and there is no general standard for these labs. At the same time, these systems are insufficient as an integrated solution for engineering education. When these remote and virtual laboratories are analyzed for robotics education, it is seen that they are separated as mobile robot and robot arm lab studies. At the same time, there are laboratories that provide some basic courses for robot arm or mobile robot experiments. In this case, general view for the studies is given.

Prior project to the remote laboratory studies which could be presented as large-scale is ReLAX; (remote laboratory trial experimentation). Relax is under the framework of the IST program to be supported by the European Commission and is aimed to test the feasibility of working remotely in distance education [5]. In current status, most of the developed remote and virtual laboratory studies focus on mobile robots, have different robots, different experiment mechanisms, different access methods, different technologies and they have been developed in many different ways. MATLAB is being used extensively in academic and industrial studies, but the spread is very difficult because of the software cost. Also, in these studies' primitive robots or LEGO NXT mobile robots that have limited ability and capacity are usually preferred to use in order to reduce the cost of the laboratory [9]. On the other hand, some studies have different robots depending on the laboratory's environments [10]. Studies [11] and [12] propose remote laboratory which are built on MATLAB and LEGO NXT robots are proposed remote access laboratory to perform multi robot experiments by using MATLAB. Proposed laboratory is able to use research and educational purposes and it has predefined mobile robot experiments. Another team [13] offers architecture for implementation of the remote laboratory for the mobile robot (Surveyor SRV-1 mobile robot) applications. Architecture is built on MATLAB and open-source software library (EJS) and the purpose of this remote lab for students to do experiments on mobile robots as a group via the internet. In 2005, Web-based remote laboratory [14] has been developed to teleoperate a mobile robot (WiRobot) that can be controlled over the internet. Server software is developed by using MATLAB with visual feedback, direct control, sensor data monitoring and fuzzy control interfaces. Some remote laboratories for mobile robots have been developed using primitive robots in a small experimental area. For example, in study [15] established a mobile robot remote access laboratory which has specially developed a robot named Robotnačka with Bluetooth communication. These robots can be controlled using a visual programming language LOGO. The laboratory is on a white board covered with various guide signs on ceramic and lighting, 1.6 m above the camera, and there are robot charging stations and is surrounded by plastic walls. Besides the remote labs with mobile robots, there are many remote labs contain robot arm. In another study [16] a web-based interface for remote control of the 3D generated KUKA robot manipulator was proposed the mathematical model of manipulators for transferring 3D medium was carried out using VRML. RACT system [17] which allows only to remote control of the real robot arm (PUMA 560) is developed for educational purposed and some basic experiments like image-based robot control, servo motor control of robot arm. These operations can be conducted by using MATLAB interface and user can watch experiments via the cameras. There are also systems which allows only to control of the real robot arm (PUMA 560) [17], RACT is developed for educational purposed and some basic experiments like image-based robot control, servo motor control of robot arm could be conducted by using MATLAB interface and user can watch experiments via the cameras. Remote and virtual laboratories are usually developed by control engineers or electronic engineers. Therefore, inside of the most of the developed remote robotic laboratories, the experiments of the electronic [18] or control [19], disciplines is located. These studies [20-22] propose a remote laboratory which have been developed to teach the concept of automatic control to engineering students and describes the laboratory structure. In [20] there are two experiment facilities which are mobile robot's formation control system and ballplate system inside the remote laboratory. Also, in [21] visual programming language is designed and develop an interface for robot arm control; students can implement experiments using this language. There are MATLAB and VRML based remote laboratories [22,23] which have inverted pendulum, motor control, robot arm and mobile robot concepts for engineering education. They also provide the advantages of remote laboratories in control engineering graduate and postgraduate studies and training and indicates these systems as a complementary educational tool. In study [24], remote robotic laboratory is proposed. This laboratory is based on Free Architecture for Remote education (FARE). LEGO NXT robot is remotely controlled via created Pyhton IDE.

In another study [25] web-based platform has been developed for use in the engineering training and scientific studies. This platform can be monitored via the web; it may be controlled and re-programmed. MATLAB interface has been customized to make the image processing studies with artificial neural networks but this interface allows set the parameters via a web browser, not allowing any coding. another web-based virtual laboratory application [26] has been developed for robotic activities. The experiments can be conducted with mobile robots; motor control, obstacle avoidance, image processing and orbit control is defined. These labs allow users, microcontroller programming, use of sensors, path planning, trajectory tracking can implement such experiments with real robots and equipment over the Internet. This platform has been developed in JAVA and Visual Basic, and a webcam, 3 IR sensor sonar and encoder which is used on a primitive mobile robot. But there is a limited capacity for this laboratory; only 1 user could connect to the system at one time. Virtual labs with robot arm are more prominent than remote accessible robot arm laboratories. Study [27], proposes a virtual robotics laboratory for robot arm education was developed for the university's mechanical engineering department. Robot control could be done by using MATLAB interface with entering parameters. Robot kinematics and dynamics are taught in training courses. A paper presented by [28] proposes Virtual laboratory for training industrial robots. Given simulation program is created by using Unity 3D game engine and demonstration of methodology is given by virtualization of Delta-type robot. Next study proposed in [29] shows a MATLAB-based virtual robotics laboratory which provides three-dimensional animation for robot arm. All components of this virtual laboratory like GUI, simulations and 3d animations are made by only using MATLAB without any other software. Simscape is employed for modelling robot's physical systems and utilizing linear analysis tool for obtaining linear system model. Aim of this study was to allow students to obtain stable conditions and linear models of robotic system without getting confused its mathematical modelling. [30] Presented virtual laboratory for studies who are making line follower robots. It is aimed to propose unsettled and changeable tracks with virtual laboratory named V-Lab. Study is not fully finished and based on analysis from trails shows that V-Lap is feasible for robot simulations. In next study, virtual laboratory system based on Unity3D engine virtual reality technology. In this virtual laboratory students are able to observe the structure of the robotic arm and its working process during virtual operation [31]. Next mobile virtual loboratory study named m-PaRoLa is proposed on studying some kinematics for five-bar and 3RRR robots. Study aims to improve this lab with adding new robots [32]. However, limitations on mobile systems minimize applicability of this method. In 2017, A virtual laboratory is proposed for processor-in-the-loop concept. In this study, inverted pendulum experiment is given as a case study to verify processor-in-the-loop system [33].

Some laboratories have virtual and remote laboratories [34]. This remote accessible virtual lab has been developed by using EJS technology for robot arm (Scorbot ER-IX) education. The robot control could be done by using certain commands and entering parameters. Next study [35] proposes virtual robot laboratory system with Virtual Reality (VR) application for engineering students. System is still in development and finished application aims to provide not only an immersive experience for students to improve the quality of teaching, but also to enable students to interact with industrial robots in virtual laboratory environment. Given method is planned to be used by University of Alicante robotic engineering. Next study by [36] proposes client/server mode robot remote control system with simulation interface. Case study proposes one 6-DOF robot arm and a 4-DOF SCARA robot. Results show that Robot kinematic simulations are effective and helpful for student education [36].

III. VIRTUAL ROBOTIC LABORATORY (U-PLAT)

U-Plat is a new multi-layered ROS and robotics training platform that includes various robotics software training. These trainings include exercises adjusted to increase in level from easy to difficult. The greatest convenience of the trainings designed for students is the opportunity to get started without dealing with complex ROS and Linux installations. The user experience has been tried to be provided in all training tools with a very simplified and easily understandable interface. All designed training tools are provided through a web interface supported by advanced cloud systems and Docker container infrastructure. Sample video for U-Plat Virtual Robotic Laboratory is given in the Youtube [37].

When a user starts a tutorial on U-Plat, a user-specific Ubuntu virtual machine is created using Amazon Web Service (AWS). Figure 1 shows the architectural summary of the U-Plat.

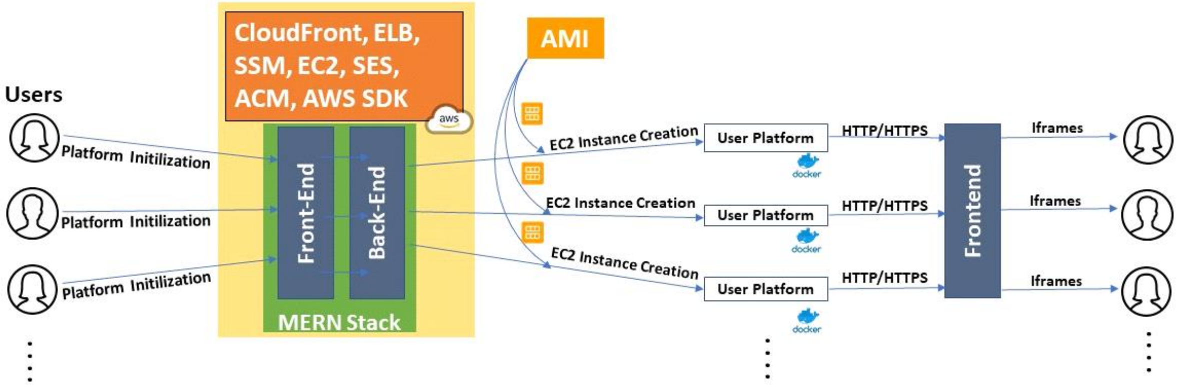


Figure 1. U-Plat architecture

U-Plat uses state-of-the-art systems such as Amazon Route53, Amazon CloudFront, Amazon Certificate Manager (ACM), Simple Storage Service (S3), Amazon Cloud Compute (EC2), Elastic Load Balancer (ELB), Simple. A summary of technologies used such as Email Service (SES), MongoDB, Amazon Machine Image (AMI), Systems Manager (SSM), AWS-SDK can be seen in Figure 2.

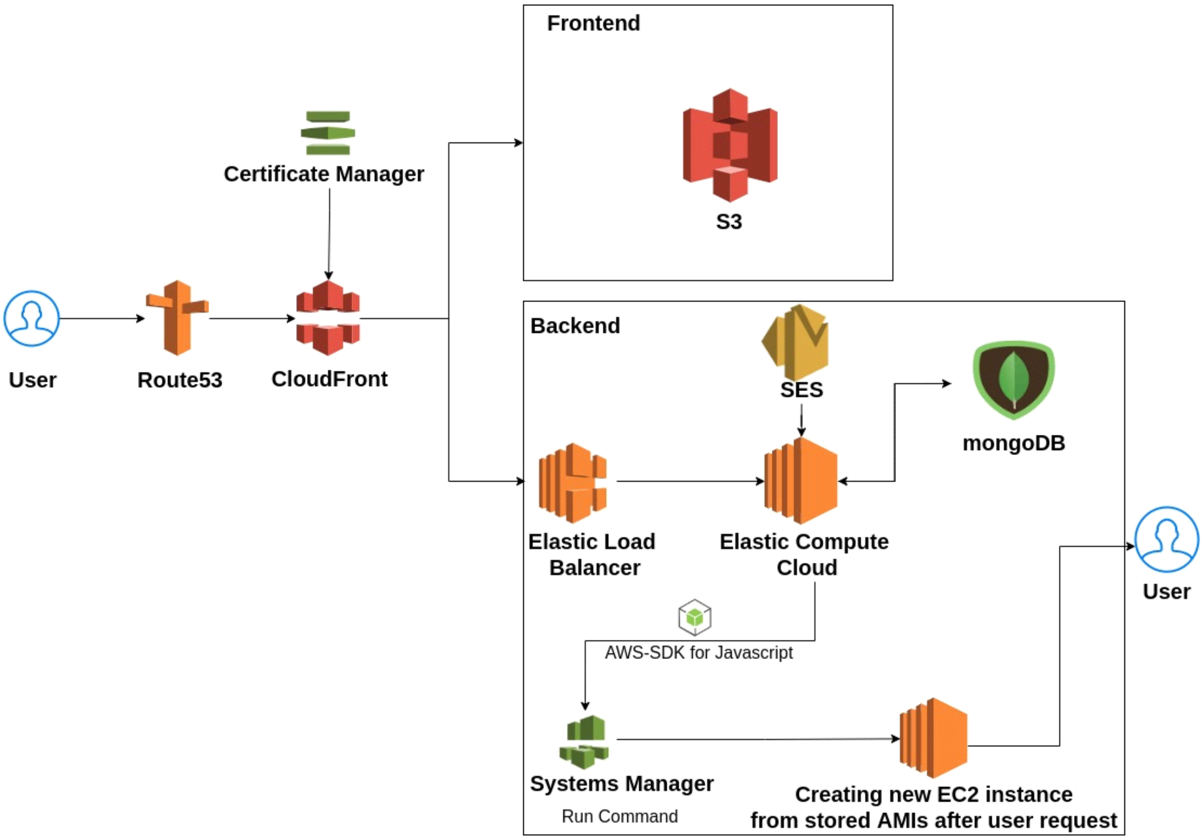
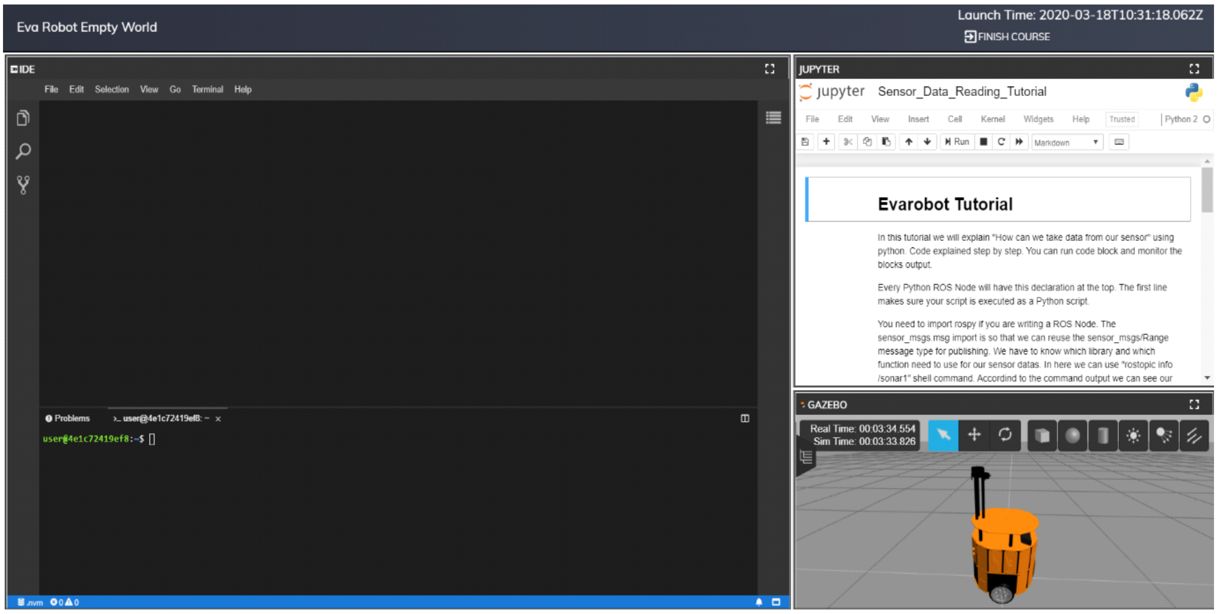


Figure 2. U-Plat technology chain

Gazebo Web (gzweb), Theia [38] and Jupyter Notebook tools are the core components available to the user on the U-Plat platform via Docker containers. By running these containers, the user can easily have them through the web interface without the need to install ROS, Gazebo and the necessary Unix/Linux operating system, which can be very troublesome to install on normal computers. With the existing Gazebo Web client, the robot is simulated based on user-written codes or studies. Compared to using from the local computer, the user can run their work much faster thanks to the client technology provided. Theia, another component used in the system, provides an integrated software development environment. The application makes it possible to write code in many languages. In addition, all code pieces required for the virtual environment used with the terminal provided by Theia can be run. On the other hand, Jupyter Notebook is integrated to provide tutorial document to provide useful information for user to learn and experiment. In this way, the user will be able to test their own codes, perform the tutorials in the document and observe the results. The U-Plat environment created with the mentioned components can be seen in Figure 3.



Eva Robot Empty World

Launch Time: 2020-03-18T10:31:18.062Z

```
user@4e1c72419ef8:~$ 
```

Sensor_Data_Reading_Tutorial

Evarobot Tutorial

In this tutorial we will explain "How can we take data from our sensor" using python. Code explained step by step. You can run code block and monitor the blocks output.

Every Python ROS Node will have this declaration at the top. The first line makes sure your script is executed as a Python script.

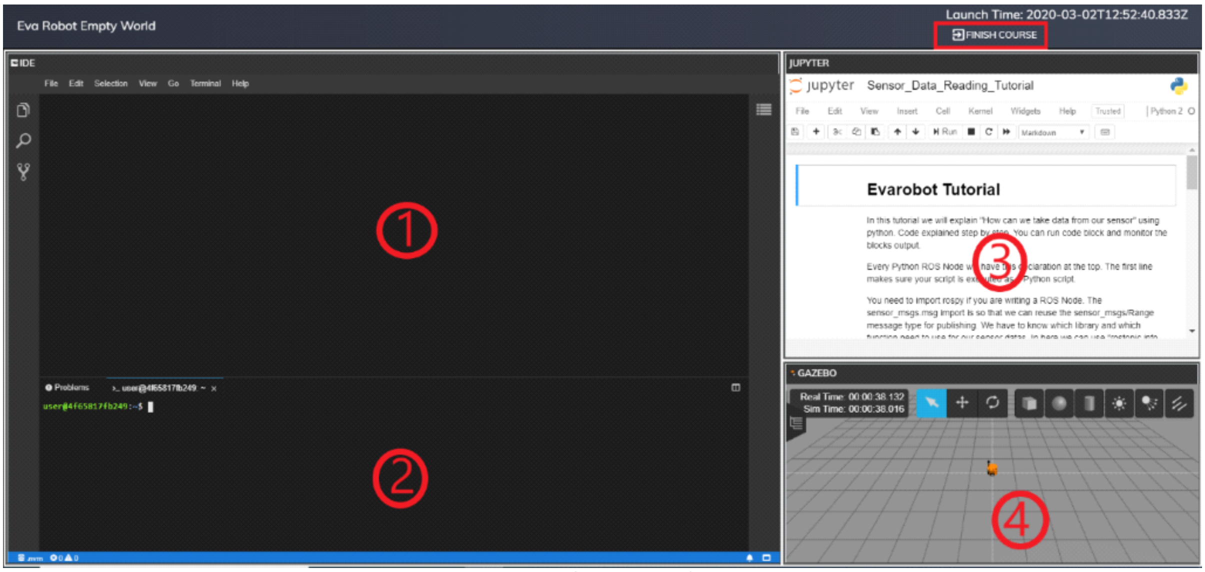
You need to import rospy if you are writing a ROS Node. The sensor_msgs.msg import is so that we can reuse the sensor_msgs/Range message type for publishing. We have to know which library and which function need to use for our sensor datas. In here we can use "rostopic info /sonar1" shell command. Accordind to the command output we can see our

Real Time: 00:03:34.554
Sim Time: 00:03:33.826

Figure 3. U-Plat environment

A.U-Plat Interfaces

While starting a new training, Pressing Start button in academy page creates new instance on AWS through backend software. These processes can take a few minutes. Then AWS EC2 instance is created and this process is depending on user connection speed. After all applications have started, the interface in figure 7 welcomes the user. The areas marked with 1 and 2 in this interface are the parts of Theia IDE. User can use it to write code in the location indicated by 1. There is a terminal in the area indicated by 2. All commands can be given to the instance through this terminal. The area marked by 3 is Jupyter Notebook. User can access the training document with Jupyter Notebook. The area indicated by 4 is the Gazebo Web application. User can view all the simulations through Gazebo Web. In the upper right, it can be viewed when training is started.



Eva Robot Empty World

Launch Time: 2020-03-02T12:52:40.833Z

```
user@4f65817fb249:~$ 
```

Sensor_Data_Reading_Tutorial

Evarobot Tutorial

Real Time: 00:00:38.132
Sim Time: 00:00:38.016

Figure 4. View of the course after running

B.U-Plat Tools

Start a Terminal: User can open many terminals or split them. With a terminal, user can run any command except root commands. U-Plat tutorials are designed so that no root command is required.

File System: The file system appears in the explorer section on the left side of Theia IDE. Many file operations can be performed in the opened section. For example, if filename.py is written on the screen that opens when the New File button is clicked, a python file is created. Theia IDE supports many programming languages. These transactions can be done with CLI commands with the terminal. User can download the selected folder or file with the Download button and with the Upload button; user can upload a folder or file to the selected folder.

Gazebo - Main User Interface: Some parts of gzweb are marked in Figure 11. The button marked 1 opens the Gazebo panel. Real and simulation times are shown in the area marked by 2. There may be a few seconds difference between real time and simulation time at first start. This is due to the difference in the start time between gzserver and gzclient. There are different types of cursors in the field marked 3. User can choose the cursor to move and rotate in this section. There are different objects in the area marked with 4. User can add objects into the Gazebo simulation world.

In Gazebo, user can restart the world, this way simulation models return their original states and simulation times are reset. Also, reset view button defaults the viewing angle.

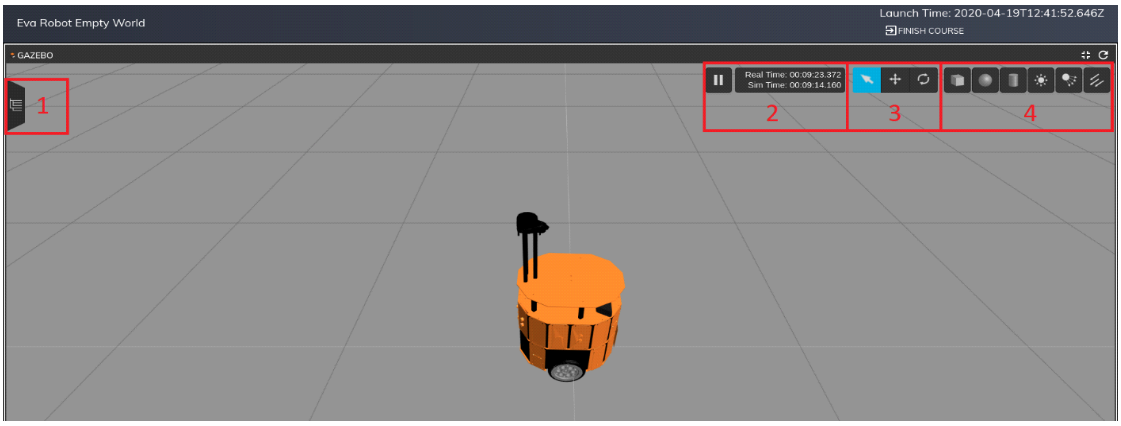
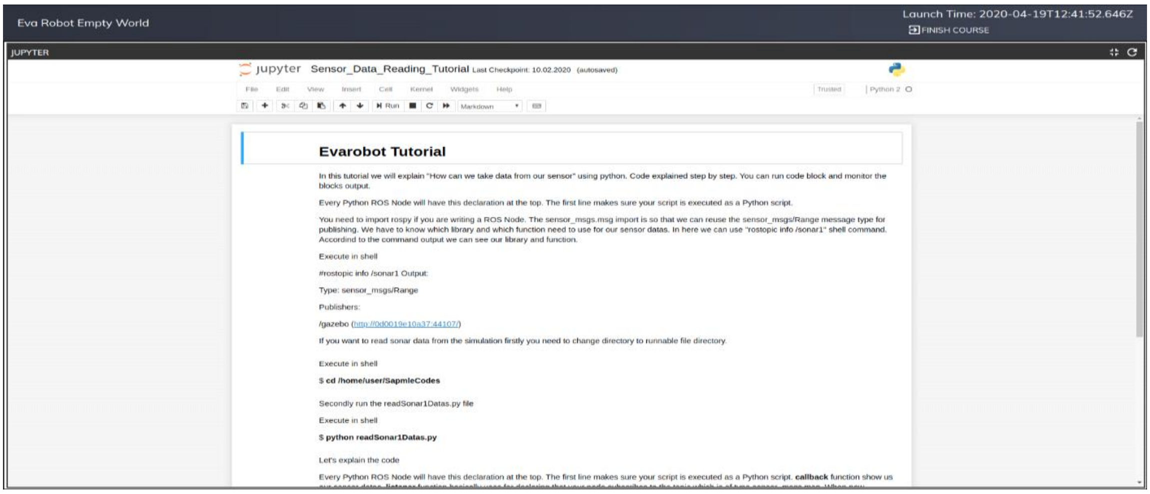


Figure 5. Gazebo main user interface

Jupyter: The Jupyter shown in Figure 6 is used to show training documents. The user will learn through the educational content available here and will test himself with questions. In this way, the user will both learn and reinforce what they have learned.



Evarobot Tutorial

In this tutorial we will explain "How can we take data from our sensor" using python. Code explained step by step. You can run code block and monitor the blocks output.

Every Python ROS Node will have this declaration at the top. The first line makes sure your script is executed as a Python script.

You need to import rospy if you are writing a ROS Node. The sensor_msgs.msg import is so that we can reuse the sensor_msgs/Range message type for publishing. We have to know which library and which function need to use for our sensor datas. In here we can use "rostopic info /sonar1" shell command. Accordind to the command output we can see our library and function.

Execute in shell

#rostopic info /sonar1 Output:

Type: sensor_msgs/Range

Publishers:

/gazebo (http://0d0019e10a37:44107/)

If you want to read sonar data from the simulation firstly you need to change directory to runnable file directory.

Execute in shell

$ cd /home/user/SapmleCodes

Secondly run the readSonar1Datas.py file

Execute in shell

$ python readSonar1Datas.py

Let's explain the code

Every Python ROS Node will have this declaration at the top. The first line makes sure your script is executed as a Python script. **callback** function show us

Figure 6. Jupyter interface view

IV. EVAMARS MISSON TUTORIAL

These tutorials are an educational guide designed for the "EvaMars Mission Robot" in which the necessary studies for a possible Mars mission training are set up. A robotic simulation has been created with a

Gazebo environment constructed from a real Mars map. With the training to be carried out in this simulation environment, users are expected to complete this training by performing Mars exploration missions through various ways of participation. A training plan consisting of seven missions designed for seven Mars days has been made. In line with this plan, a robotics training has emerged that can be completed by users with intermediate ROS knowledge and Python knowledge. Sample demonstration video on EvaMars missions is given in the Youtube [37].

A. EvaMars Robot

The EvaMars robot is a Mars reconnaissance robot model developed for the Uplat environment. It is modeled to be able to perform exploration missions in the Martian environment, conduct drilling works, and move in the harsh terrain of the environment. A possible Mars exploration scenario was created by means of Eva-Mars and robotics training packages were revealed with this scenario.

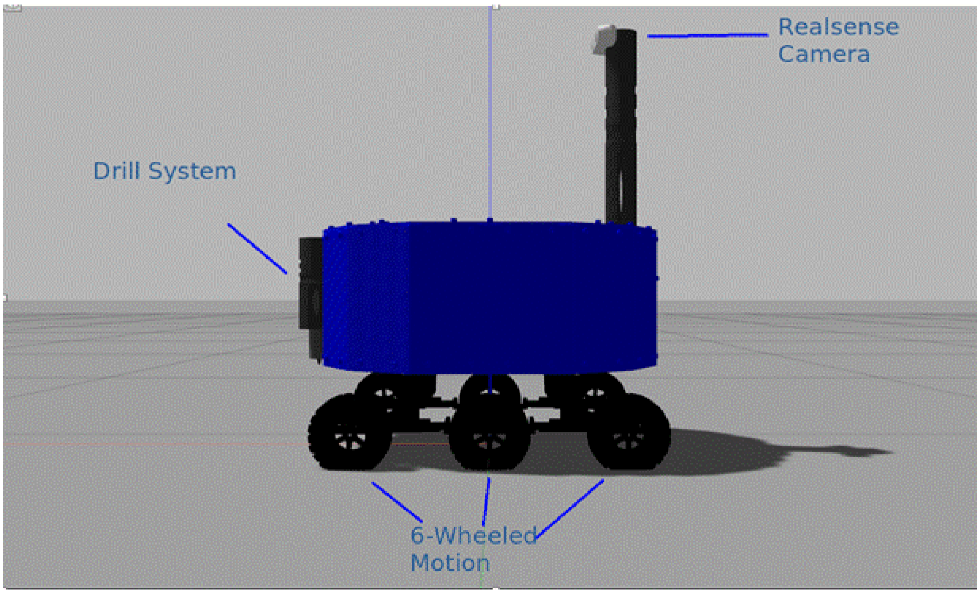


Figure 7. EvaMars robot’s components

Drilling System: Eva-Mars Robot has a drilling system with 3 nested cylinders. The system, which has a piercing tip that can go down to a depth of 15 cm from the surface, is designed to simulate drilling activity scenarios on the Martian surface.

6-Wheel Movement: Eva-Mars Robot has 6-wheel drive wheels that can move easily on rough terrain on Mars surface.

Realsense Camera: The Intel Realsense D435 Camera, the eye of the Eva-Mars Robot on Mars, has been integrated for simulations of Mars exploration missions thanks to both RGB image and depth perception. The robot gains a 360-degree perspective, thanks to the rotating port on which the camera is attached.

B. Mars Missions

The Mars missions defined for the EvaMars robot developed for the missions begins with EvaMars' first descent to the Martian surface and its first environmental survey mission. Then, it ends with the completion of the robot's drilling work and final research tasks in the predetermined areas. This process is defined by seven different tasks. The tasks given to the users are that the robot makes various contributions to complete these tasks and enables the robot to complete these tasks. Contributions to be made are editing the Python codes to be applied to the robot in some robotic tasks, designing the initialization files for the robot and making various observations by providing manual control of the robot. The details of these tasks are as shown in the diagram below.

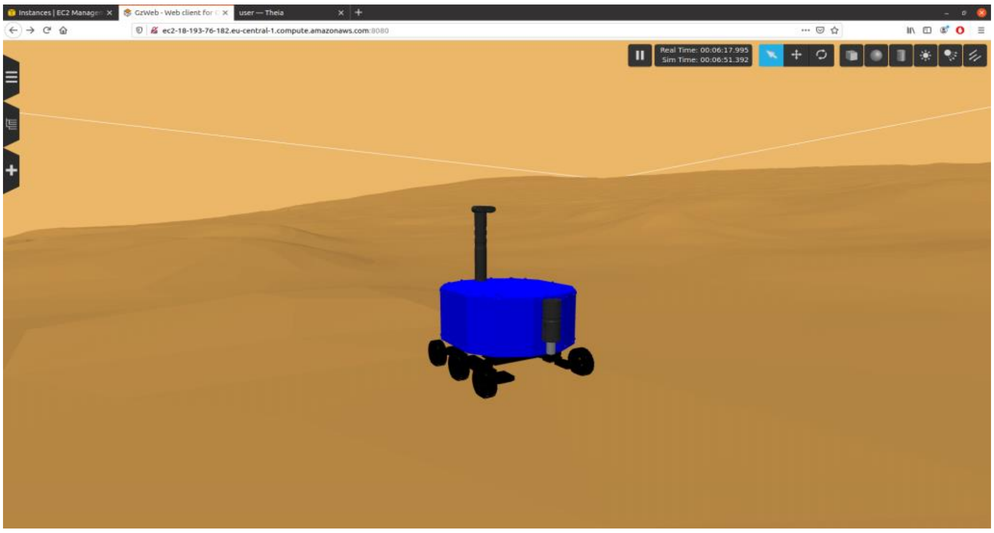
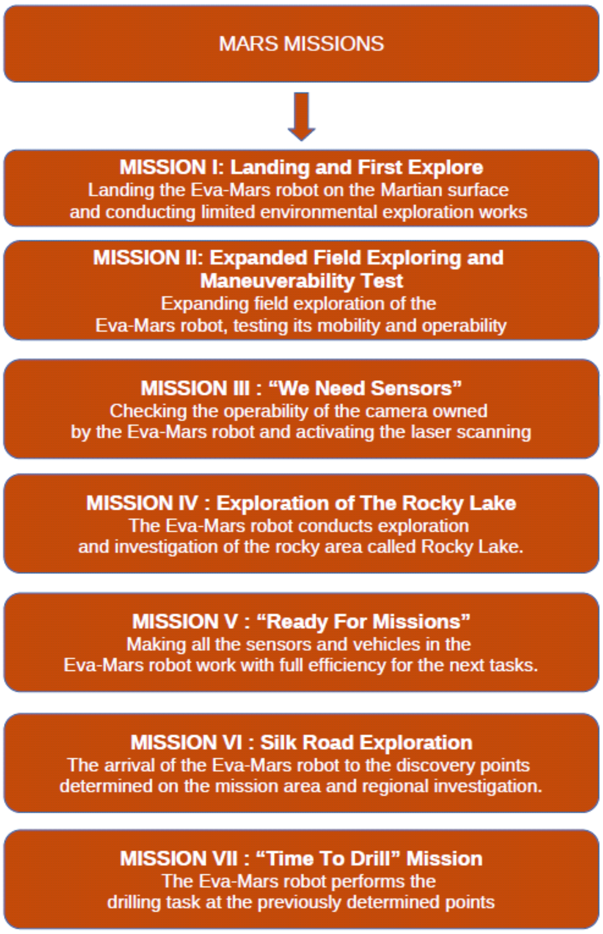


Figure 8. EvaMars robot



MARS MISSIONS

MISSION I: Landing and First Explore
Landing the Eva-Mars robot on the Martian surface and conducting limited environmental exploration works

MISSION II: Expanded Field Exploring and Maneuverability Test
Expanding field exploration of the Eva-Mars robot, testing its mobility and operability

MISSION III : "We Need Sensors"
Checking the operability of the camera owned by the Eva-Mars robot and activating the laser scanning

MISSION IV : Exploration of The Rocky Lake
The Eva-Mars robot conducts exploration and investigation of the rocky area called Rocky Lake.

MISSION V : "Ready For Missions"
Making all the sensors and vehicles in the Eva-Mars robot work with full efficiency for the next tasks.

MISSION VI : Silk Road Exploration
The arrival of the Eva-Mars robot to the discovery points determined on the mission area and regional investigation.

MISSION VII : "Time To Drill" Mission
The Eva-Mars robot performs the drilling task at the previously determined points

Figure 9. EvaMars missions

1)Mission I: Landing and First Explore Mission:

Mission Purposes: EvaMars Mission Robot will be landed in a crater area that is in the most ideal position for the missions it will perform on the Mars floating. This landing must take place smoothly and the robot should discover its limited environment and inform the center that it has completed this discovery. With this small mission, it is aimed to test that the landing of the robot is smooth and the basic motion functions of the robot are working.

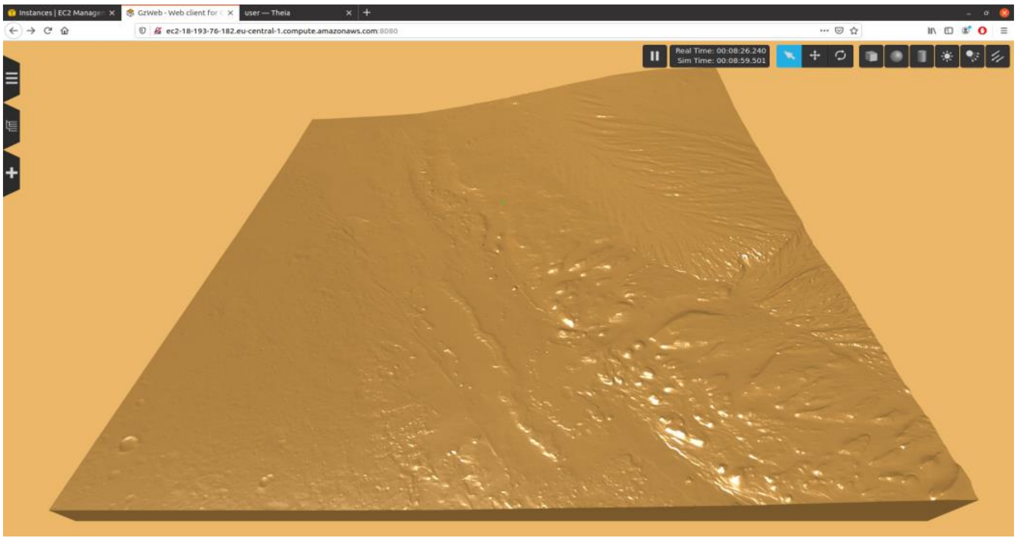


Figure 10. Mars Gazebo environment

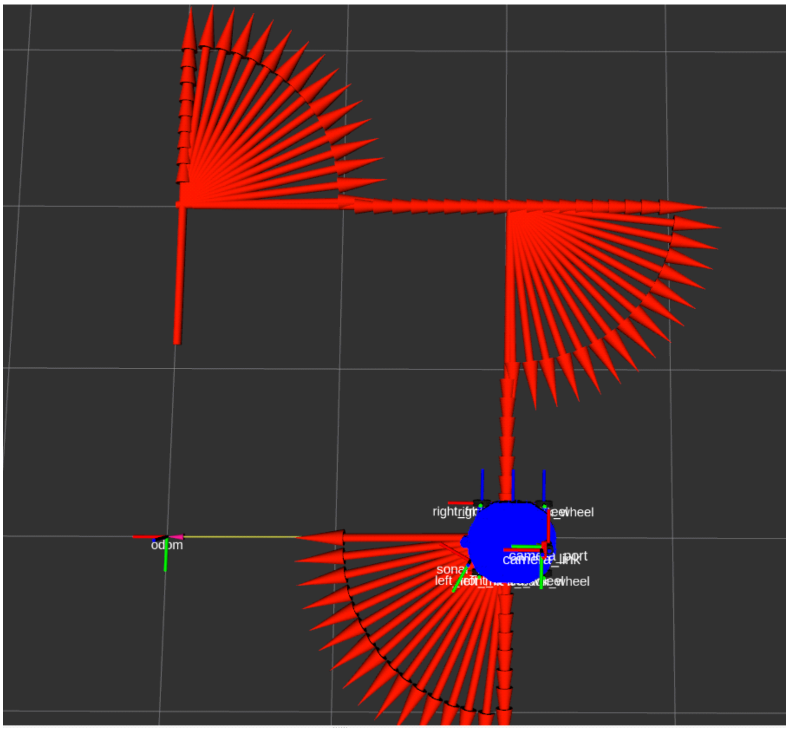
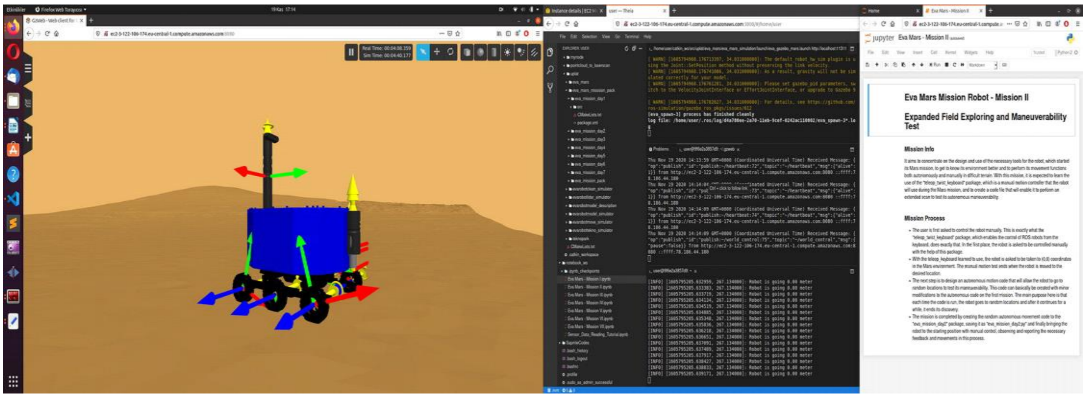


Figure 11. Graphical output of firsr mission

2)Mission II: Expanded Field Exploring and Maneuverability Test:

Mission Purposes: It aims to concentrate on the design and use of the necessary tools for the robot, which started its Mars mission, to get to know its environment better and to perform its movement functions both autonomously and manually in difficult terrain. With this mission, it is expected to learn the use of the "teleop_twist_keyboard" package, which is a manual motion controller that the robot will use during the Mars mission, and to create a code file that will enable it to perform an extended scan to test its autonomous maneuverability.



```
[ WARN] [1605794988.176713397, 34.031000000]: The default_robot_hw_sim plugin is using the Joint::SetPosition method without preserving the link velocity.
[ WARN] [1605794988.176741086, 34.031000000]: As a result, gravity will not be simulated correctly for your model.
[ WARN] [1605794988.176761281, 34.031000000]: Please set gazebo_pid parameters, switch to the VelocityJointInterface or EffortJointInterface, or upgrade to Gazebo 9
[ WARN] [1605794988.176782027, 34.031000000]: For details, see https://github.com/ros-simulation/gazebo_ros_pkgs/issues/612
[eva_spawn-3] process has finished cleanly
log file: /home/user/.ros/log/d4a708ee-2a70-11eb-9cef-0242ac110002/eva_spawn-3*.log
$
```

```
Thu Nov 19 2020 14:13:59 GMT+0000 (Coordinated Universal Time) Received Message: {"op":"publish","id":"publish:~/heartbeat:72","topic":"~/heartbeat","msg":{"alive":1}} from http://ec2-3-122-106-174.eu-central-1.compute.amazonaws.com:8080 ::ffff:78.186.44.180
Thu Nov 19 2020 14:14:04 GMT+0000 (Coordinated Universal Time) Received Message: {"op":"publish","id":"publish:~/heartbeat:73","topic":"~/heartbeat","msg":{"alive":1}} from http://ec2-3-122-106-174.eu-central-1.compute.amazonaws.com:8080 ::ffff:78.186.44.180
Thu Nov 19 2020 14:14:09 GMT+0000 (Coordinated Universal Time) Received Message: {"op":"publish","id":"publish:~/heartbeat:74","topic":"~/heartbeat","msg":{"alive":1}} from http://ec2-3-122-106-174.eu-central-1.compute.amazonaws.com:8080 ::ffff:78.186.44.180
Thu Nov 19 2020 14:14:09 GMT+0000 (Coordinated Universal Time) Received Message: {"op":"publish","id":"publish:~/world_control:75","topic":"~/world_control","msg":{"pause":false}} from http://ec2-3-122-106-174.eu-central-1.compute.amazonaws.com:8080 ::ffff:78.186.44.180
```

```
[INFO] [1605795205.632955, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.633383, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.633719, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.634124, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.634519, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.634885, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.635348, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.635836, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.636218, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.636651, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.637091, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.637409, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.637917, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.638427, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.638833, 267.134000]: Robot is going 0.00 meter
[INFO] [1605795205.639171, 267.134000]: Robot is going 0.00 meter
```

Eva Mars Mission Robot - Mission II

Expanded Field Exploring and Maneuverability Test

Mission Info

It aims to concentrate on the design and use of the necessary tools for the robot, which started its Mars mission, to get to know its environment better and to perform its movement functions both autonomously and manually in difficult terrain. With this mission, it is expected to learn the use of the "teleop_twist_keyboard" package, which is a manual motion controller that the robot will use during the Mars mission, and to create a code file that will enable it to perform an extended scan to test its autonomous maneuverability.

Mission Process

- The user is first asked to control the robot manually. This is exactly what the "teleop_twist_keyboard" package, which enables the control of ROS robots from the keyboard, does exactly that. In the first place, the robot is asked to be controlled manually with the help of this package.
- With the teleop_keyboard learned to use, the robot is asked to be taken to (0,0) coordinates in the Mars environment. The manual motion test ends when the robot is moved to the desired location.
- The next step is to design an autonomous motion code that will allow the robot to go to random locations to test its maneuverability. This code can basically be created with minor modifications to the autonomous code on the first mission. The main purpose here is that each time the code is run, the robot goes to random locations and after it continues for a while, it ends its discovery.
- The mission is completed by creating the random autonomous movement code to the "eva_mission_day2" package, saving it as "eva_mission_day2.py" and finally bringing the robot to the starting position with manual control, observing and reporting the necessary feedback and movements in this process.

Figure 12. EvaMars U-Plat screen

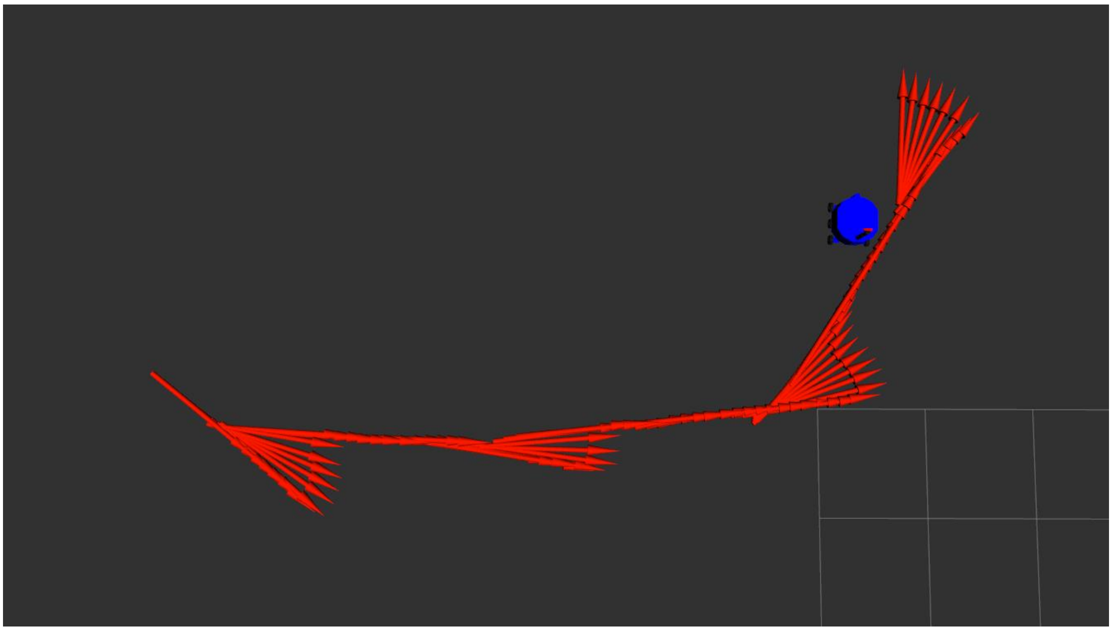


Figure 13. A graph showing the random paths the robot makes

3)Mission III: "We Need Sensors":

Mission Purposes: In the third of Mars missions, it is now expected that the discoveries made with the robot will be deepened and the user will benefit from the sensors on the robot. The robot's mobility tests are now complete and the robot's environment must be well known by the user and continue the mission. The Eva Mars robot is equipped with an Intel Realsense D435 camera. In this mission, the operability of the camera on the robot is expected to be tested.

4)Mission IV: Exploration to the Rocky Lake

Mission Purposes: The fourth of the Mars missions focused on the exploration of the rocky region called Rocky Lake, located near the robot's landing point (Figure 14). The robot is asked to pass through that area autonomously. During this transition, five images are requested to be recorded from the camera. The user is expected to observe that this image can be easily taken thanks to the autonomous progress of the robot. The mission is completed at the end of the autonomous movement starting from a point in the rocky area, recording the desired images and reporting the general mission.

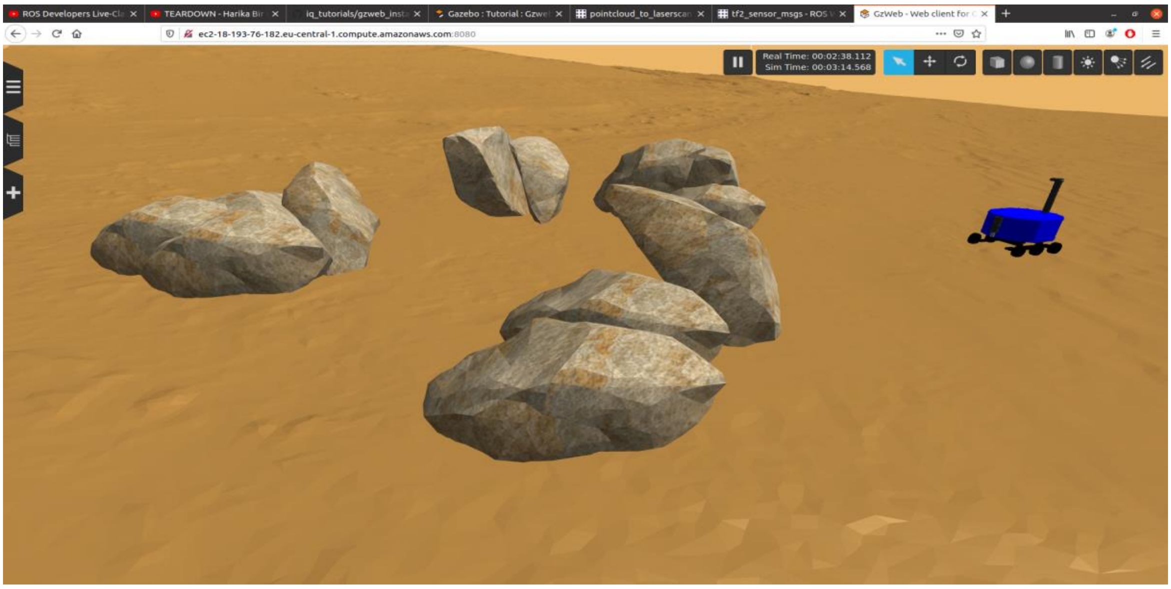


Figure 14. The Rocky Lake scenario environment

5)Mission V: "Ready for Missions":

Mission Purposes: In the fifth of Mars missions, it is aimed to activate the drilling system, which is the last tool that is not used on the robot anymore and to develop the exploration capability by the user by bringing the existing camera system to a rotary system. The user will now be able to benefit from all functions of the robot with high efficiency and will be ready to go on drilling-exploration missions with the robot. For this, it is required to create a Python code file that will make the drilling system usable and provide the ability to turn right/left to the port where the camera is attached.

6)Mission VI: Silk Road Exploration:

Mission Purposes: In this mission, which is the sixth of the Mars missions, it is about drawing a long travel route and the follow-up of this route in such a way that the robot has been prepared with all its mechanisms. The robot is expected to write a code that will enable it to be transferred to 5 different coordinates that will be received from the user (Figure 15). One image should be recorded from each point that is transferred, from the moments when the camera looks to the right, left and opposite. The mission ends with the collection of the records made by the robot at each point in the required package and the successful completion of the route.

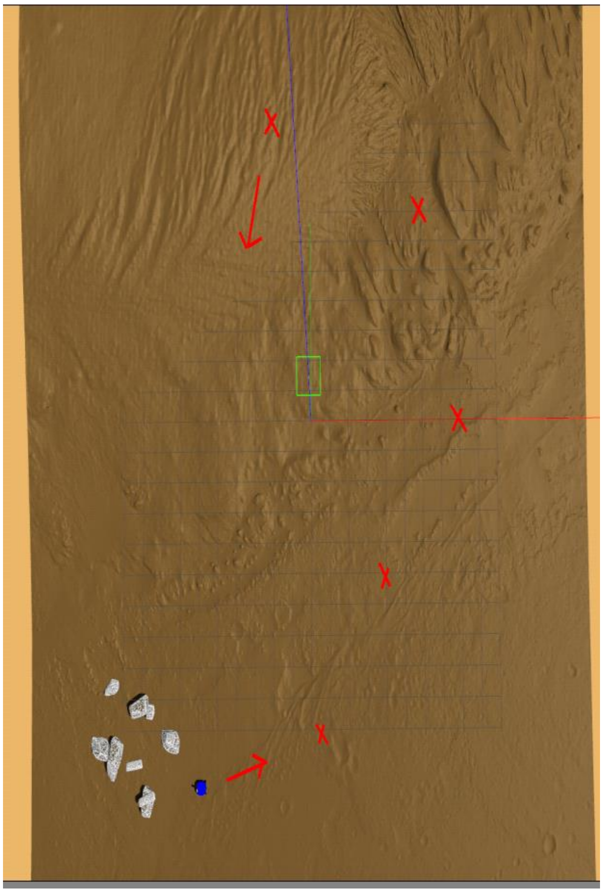


Figure 15. Travel points and route

7)Mission VII: Time to Drilling

Mission Purposes: In this mission, which is the last of the Mars missions, it is aimed to use the drilling system activated in Mission 5. With the long exploration route and investigations made in mission 6, a scenario in which certain drilling points have been discovered has been designed. The robot needs to go to two of the previously discovered spots, drill and return to the point where it was landed on Mars. The user is expected to create a code file to complete this mission.

V. EVASEC MISSION TUTORIAL

This tutorial is an educational guide in which the necessary studies for a possible facility safety inspection training designed for the "EvaSec Safety Robot" are set up and explained (Figure 16). With this simulation training set up in a sample facility map in the Gazebo environment, users are expected to complete this training by performing basic ROS functions, message and service package creation, robot status checks and robotic coding missions in various ways of participation. With this study, a robotic education that can be completed by users with basic Python and ROS knowledge has been revealed.

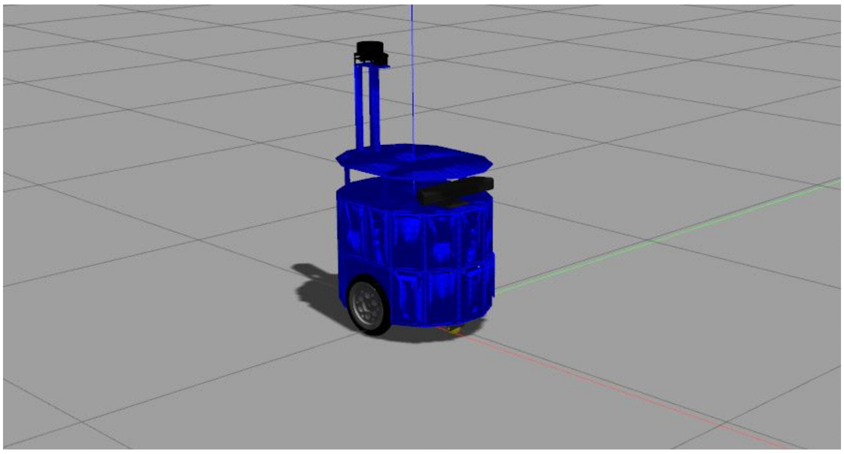
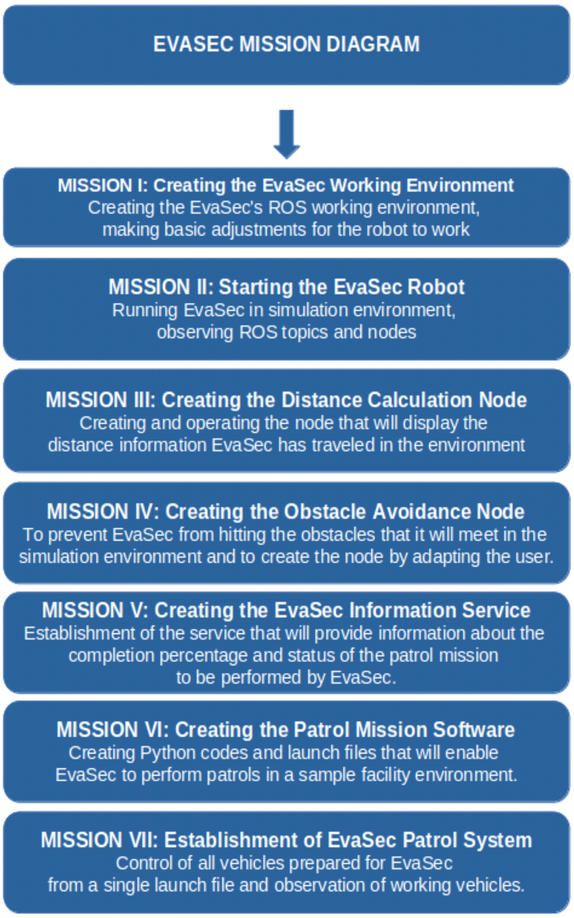


Figure 16. EvaSec security robot

A Patrol Missions

Exercises given to users include tasks that require the robot to make adjustments to enable it to complete robotics tasks that require basic knowledge of ROS. The basis of these tasks is focused on writing the necessary Python codes for the robot to complete the task, creating the launch files or creating the necessary srv/msg files. The details of these tasks are as shown in Figure 17.



EVASEC MISSION DIAGRAM

MISSION I: Creating the EvaSec Working Environment
Creating the EvaSec's ROS working environment,
making basic adjustments for the robot to work

MISSION II: Starting the EvaSec Robot
Running EvaSec in simulation environment,
observing ROS topics and nodes

MISSION III: Creating the Distance Calculation Node
Creating and operating the node that will display the
distance information EvaSec has traveled in the environment

MISSION IV: Creating the Obstacle Avoidance Node
To prevent EvaSec from hitting the obstacles that it will meet in the
simulation environment and to create the node by adapting the user.

MISSION V: Creating the EvaSec Information Service
Establishment of the service that will provide information about the
completion percentage and status of the patrol mission
to be performed by EvaSec.

MISSION VI: Creating the Patrol Mission Software
Creating Python codes and launch files that will enable
EvaSec to perform patrols in a sample facility environment.

MISSION VII: Establishment of EvaSec Patrol System
Control of all vehicles prepared for EvaSec
from a single launch file and observation of working vehicles.

Figure 17. EvaSec missions

Mission I: Creating the EvaSec Working Environment:

Mission Purposes: Before the use of the EvaSec robot, the ROS working environment should be prepared, if there is no working environment, a new environment should be created.

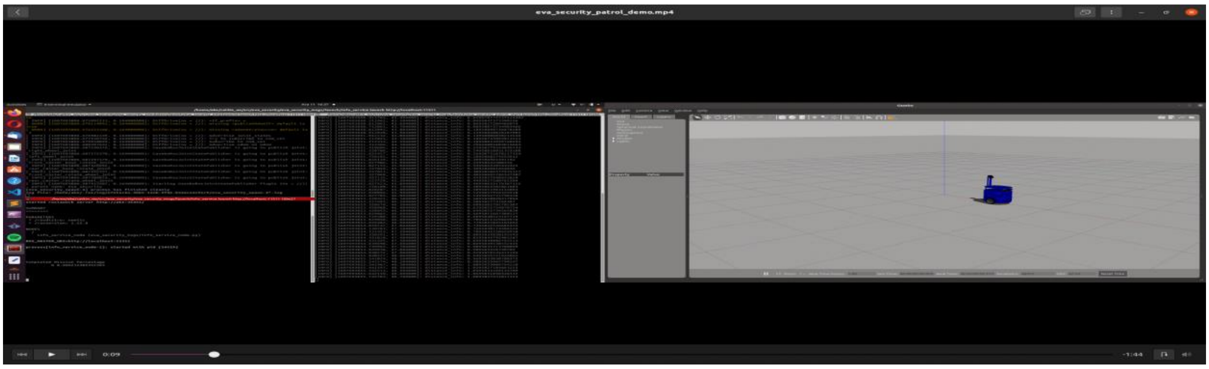


Figure 18. EvaSec patrol mission

Mission II: Starting to EvaSec Security Robot

Mission Purposes: In the ROS environment prepared for the EvaSec robot, it is necessary to download and run the robot's packages and to examine the activity status of the robot's topics and sensors.

Mission III: Creating the Distance Calculation Node

Mission Purposes: The node that will enable the EvaSec robot to broadcast the expression in meters of the distance traveled during the patrol mission from the user terminal should be designed (Figure 19).

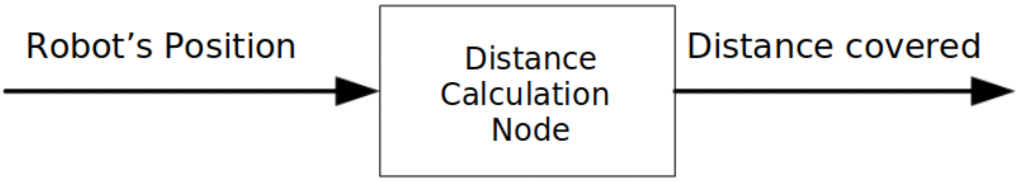


Figure 19. EvaSec distance calculation node diagram

Mission IV: Creating the Obstacle Avoidance Node:

Mission Purposes: By subscribing to the sonar sensor value of EvaSec, it is necessary to create a node that reads the distance value of the sensor to an obstacle around and makes a request to move or stop the node that performs the movement according to the specified limit value (Figure 20).

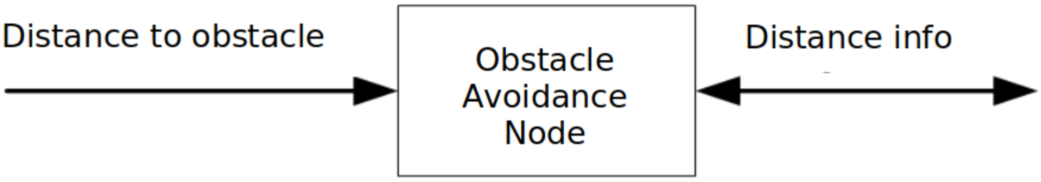


Figure 20. EvaSec obstacle avoidance node diagram

Mission V: Creating the EvaSec Information Service

Mission Purposes: A node that makes a request to the service will be created to get information about how much of the mission EvaSec has completed. The information service node can be seen in the Figure 21.

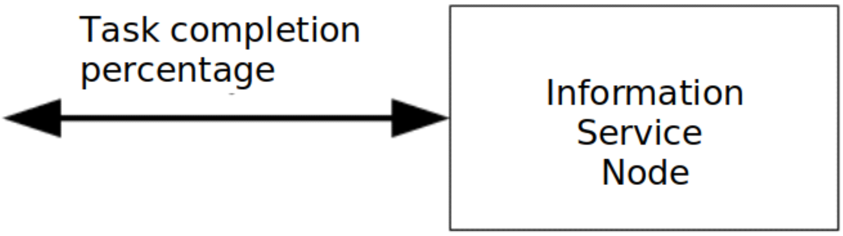


Figure 21. EvaSec information service node diagram

Mission VI: Creating the EvaSec Information Service:

Mission Purposes: The node that performs the patrol function of EvaSec should be created and run. Figure 22 shows the patrol node block diagram of the robot.

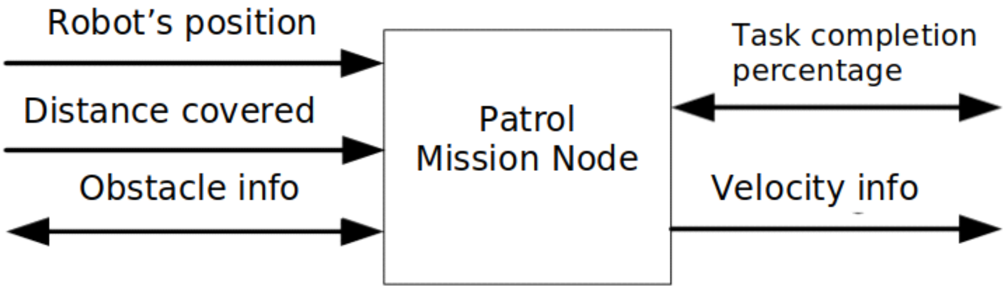


Figure 22. EvaSec patrol mission node diagram

Mission VII: Establishing the EvaSec Patrol System

Mission Purposes: It is necessary to run all of the arrangements such as file, package, and service/message attachments made for EvaSec by combining them and to create the necessary launch file to complete a simple patrol mission designed for EvaSec.

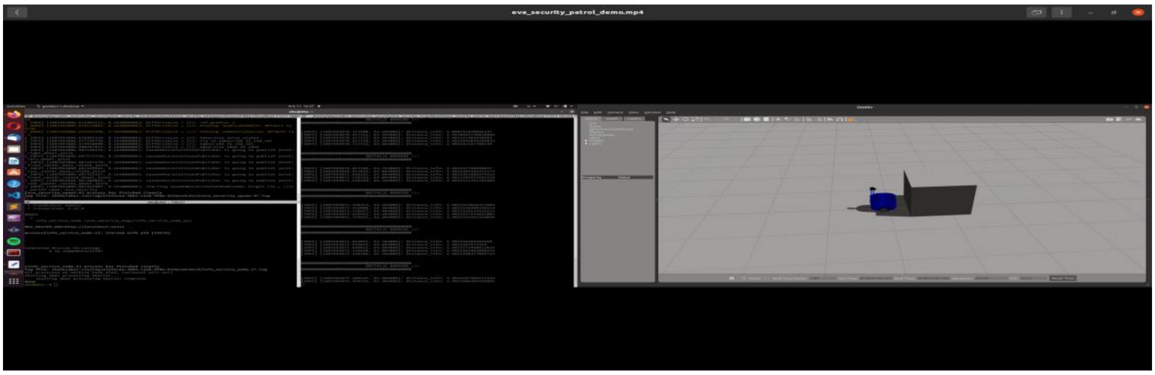


Figure 23. EvaSec object detection

VI. ATEKS MISSION TUTORIAL

This tutorial is an educational guide in which the necessary studies for a possible hospital duty training scenario designed for "ATEKS Smart Wheelchair" are designed and explained. With this simulation training designed to work in a hospital model in a Gazebo environment, users are expected to complete this training by performing the ROS and Python-based missions requested from them in various ways of participation. With this study, it was designed as a robotic training that can be completed by users with pre-intermediate ROS knowledge and Python knowledge. Training topics include various stages from reading robot sensor data to using mixed motion algorithms.

A. Hospital Missions

In-hospital service tasks designed for the ATEKS robot begin with the reading of the sensor data of ATEKS and the examinations to be made in the simulation environment with the manual use of ATEKS. The exercises end with the completion of tasks that require the robot to use various motion algorithms effectively. The basis of these tasks is focused on writing the necessary Python codes for the robot to complete the task, creating the launch files or creating the necessary srv/msg files. These missions and their explanations are as shown in the mission chart below.

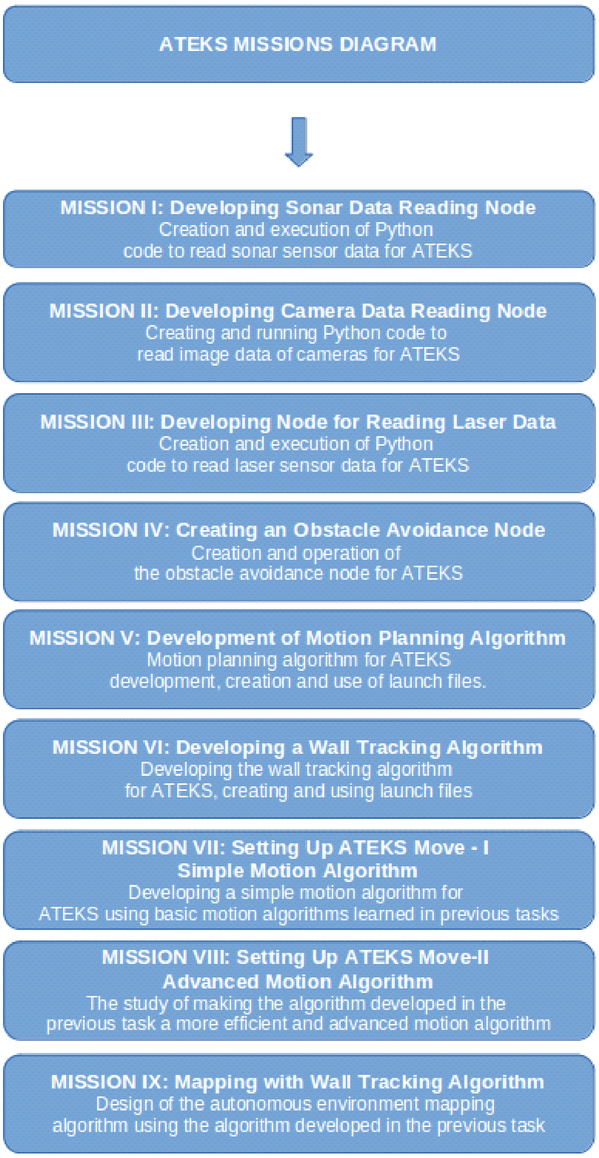
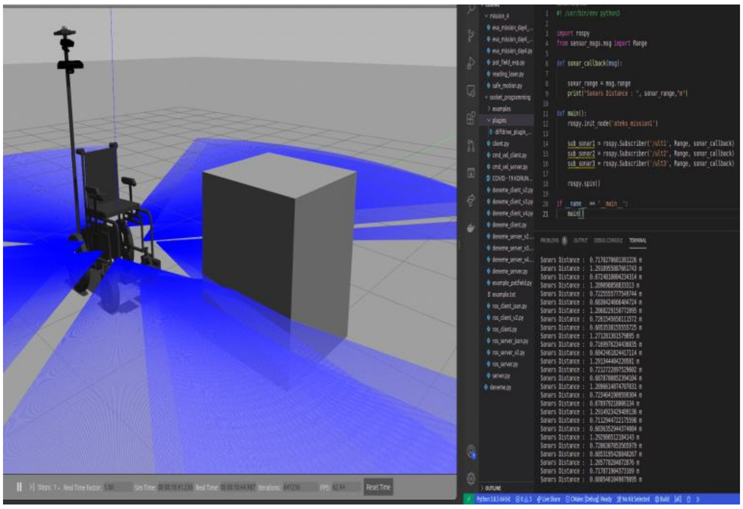

ATEKS MISSIONS DIAGRAM
MISSION I: Developing Sonar Data Reading Node
Creation and execution of Python code to read sonar sensor data for ATEKS
MISSION II: Developing Camera Data Reading Node
Creating and running Python code to read image data of cameras for ATEKS
MISSION III: Developing Node for Reading Laser Data
Creation and execution of Python code to read laser sensor data for ATEKS
MISSION IV: Creating an Obstacle Avoidance Node
Creation and operation of the obstacle avoidance node for ATEKS
MISSION V: Development of Motion Planning Algorithm
Motion planning algorithm for ATEKS development, creation and use of launch files.
MISSION VI: Developing a Wall Tracking Algorithm
Developing the wall tracking algorithm for ATEKS, creating and using launch files
MISSION VII: Setting Up ATEKS Move - I Simple Motion Algorithm
Developing a simple motion algorithm for ATEKS using basic motion algorithms learned in previous tasks
MISSION VIII: Setting Up ATEKS Move-II Advanced Motion Algorithm
The study of making the algorithm developed in the previous task a more efficient and advanced motion algorithm
MISSION IX: Mapping with Wall Tracking Algorithm
Design of the autonomous environment mapping algorithm using the algorithm developed in the previous task

Figure 24. ATEKS missions

Mission I: Developing Sonar Data Reading Node

Mission Purposes: It is aimed to design and read the data of the sonar sensors owned by ATEKS first from the terminal and then a ROS node that will do this reading. The main element of the mission is to follow ATEKS's sonar sensors over the Python code and use this data in the future.



```
#! /usr/bin/env python3

import rospy
from sensor_msgs.msg import Range

def sonar_callback(msg):

    sonar_range = msg.range
    print("Sonars Distance : ", sonar_range,"m")

def main():
    rospy.init_node('ateks_mission1')

    sub_sonar1 = rospy.Subscriber('/ult1', Range, sonar_callback)
    sub_sonar2 = rospy.Subscriber('/ult2', Range, sonar_callback)
    sub_sonar3 = rospy.Subscriber('/ult3', Range, sonar_callback)

    rospy.spin()

if __name__ == '__main__':
    main()
```

```
Sonars Distance :  0.7178270681381226 m
Sonars Distance :  1.2918955087661743 m
Sonars Distance :  0.6724810004234314 m
Sonars Distance :  1.2800968568333013 m
Sonars Distance :  0.7225555777549744 m
Sonars Distance :  0.6839424666404724 m
Sonars Distance :  1.2868229158772095 m
Sonars Distance :  0.7261545658111572 m
Sonars Distance :  0.6853538155555725 m
Sonars Distance :  1.2712813615179895 m
Sonars Distance :  0.7169976234436035 m
Sonars Distance :  0.6842461824417114 m
Sonars Distance :  1.291344404220581 m
Sonars Distance :  0.7212723897529602 m
Sonars Distance :  0.6878708852394104 m
Sonars Distance :  1.2896614074707031 m
Sonars Distance :  0.7234642909598304 m
Sonars Distance :  0.6789792180096134 m
Sonars Distance :  1.2914923429489136 m
Sonars Distance :  0.7112944722175598 m
Sonars Distance :  0.6836352944374084 m
Sonars Distance :  1.2929865122184143 m
Sonars Distance :  0.7206367053565979 m
Sonars Distance :  0.6853195428848267 m
Sonars Distance :  1.2857782044072876 m
Sonars Distance :  0.717871994373169 m
Sonars Distance :  0.6885463049879895 m
```

Figure 25. Sonar data when an obstacle is placed in front of ATEKS

Mission II: Developing Camera Data Reading Node

Mission Purposes: It is aimed to design and read the data of the sonar sensors owned by ATEKS first from the terminal and then a ROS node that will do this reading. The main element of the mission is to monitor ATEKS's sonar sensors over the Python code and use this data in the future.

Mission III: Developing Laser Data Reading Node

Mission Purpose: It is aimed to design and read the data of the laser sensor owned by ATEKS first from the terminal and then a ROS node that will do this reading. Tracking the laser sensor of ATEKS through Python code and enabling the use of this data in the future forms the main element of the mission.

Mission IV: Creating an Obstacle Avoidance Node

Mission Purpose: In the fourth of ATEKS hospital missions, it is aimed to create an algorithm that will prevent the robot from hitting the obstacles by using the data of the sensor tools used in the previous sections and to prepare the ROS node on Python.

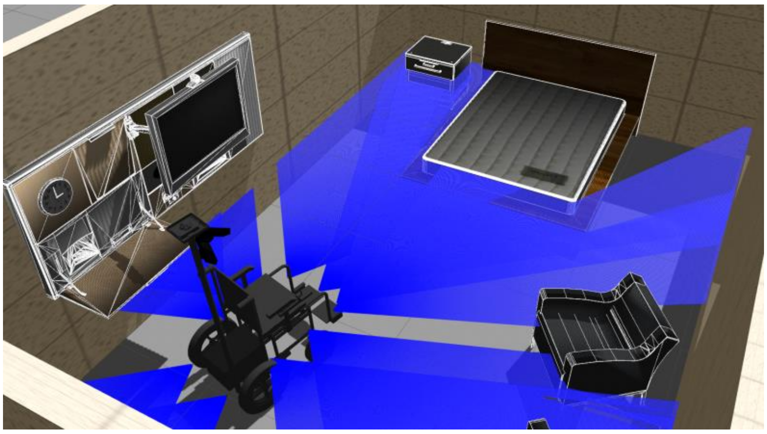


Figure 26. ATEKS obstacle avoidance mission in hospital environment

Mission V: Development of Motion Planning Algorithm

Mission Purpose: In the fifth of ATEKS hospital duties, it is aimed to create an algorithm that will enable the robot to go to any desired location by using the data of the sensor tools (laser sensor) used in the previous sections and to prepare the ROS node via Python.

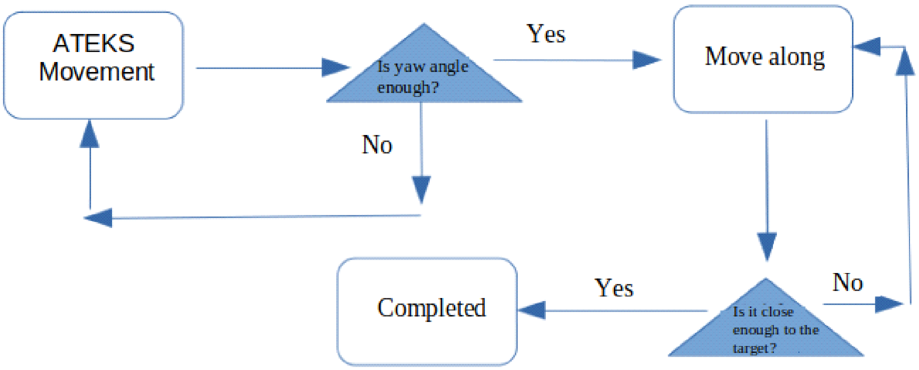


Figure 27. ATEKS motion planning algorithm diagram

Mission VI: Developing a Wall Tracking Algorithm

Mission Purpose: In the sixth of ATEKS hospital missions, it is aimed to create an algorithm that will allow the robot to follow the wall by keeping a certain distance from the wall by using the data of the sensor tools used in the previous sections and to prepare the ROS node on Python.

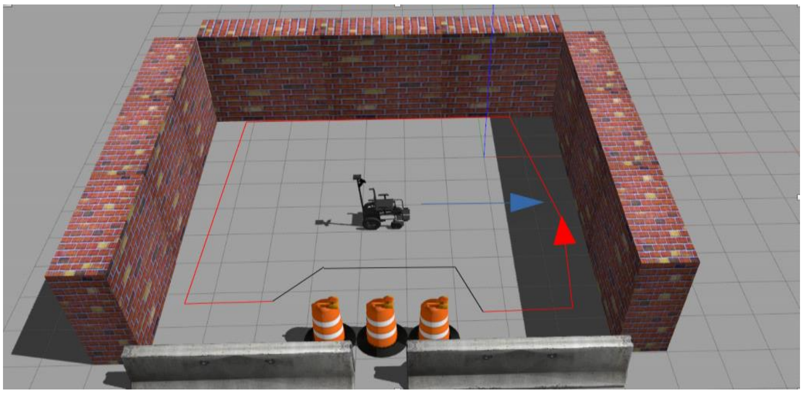


Figure 28. ATEKS wall tracking algorithm

The wall tracking algorithm in Figure 29 consists of three main movements:

- Finding the Target: The first action of ATEKS takes place by making a progress movement until the obstacle is within its range.

- Obstacle Tracking: ATEKS puts a certain distance between the obstacle and proceeds by maintaining this distance.

- Obstacle Relief: ATEKS continues its movement by reacting to the momentary changes in the distance between the obstacle by getting rid of the obstacle.

Mission VII: Setting Up ATEKS Move-I Simple Motion Algorithm

Mission Purpose: In the seventh of ATEKS hospital tasks, it is aimed to create a simple motion algorithm called ATEKS Move-I by using the motion algorithms designed in the previous sections and to prepare the ROS node on Python. With this movement node, ATEKS's wall tracking, obstacle avoidance and going to a certain point algorithm are combined and a basic target transfer node is formed.

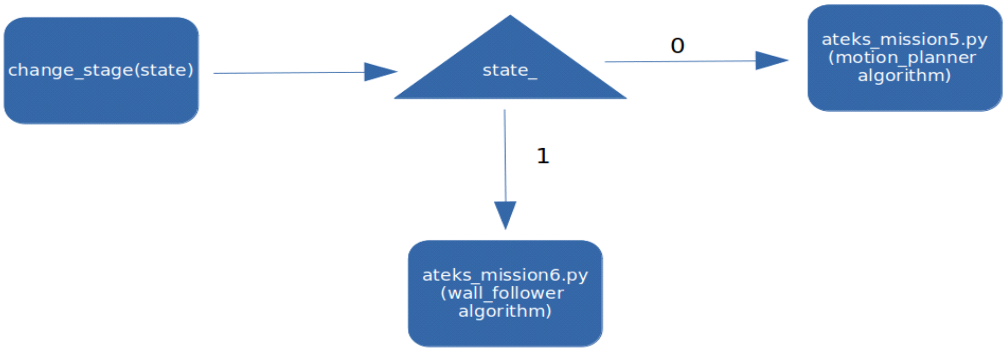

change_stage(state)
state_
0
ateks_mission5.py (motion_planner algorithm)
1
ateks_mission6.py (wall_follower algorithm)

Figure 29. Change state motion diagram

Mission VIII: Setting Up ATEKS Move-II Advance Motion Algorithm

Mission Purpose: In the eighth of ATEKS hospital duties, it is aimed to create a more advanced motion algorithm called ATEKS Move-II and to prepare the ROS node on Python by using ATEKS Move-I, the motion algorithm designed in the previous section. With this movement node, ATEKS's wall tracking, obstacle avoidance and going to a certain point algorithm are combined and a basic target node is more optimized.

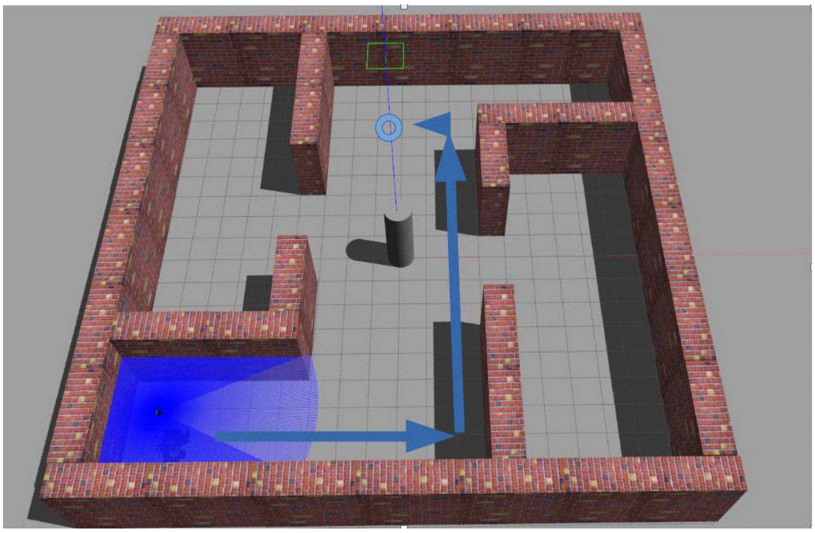


Figure 30. ATEKS Move-II movement route

Mission IX: Mapping with Wall Tracking Algorithm

Mission Purpose: In the last of ATEKS hospital missions, it is aimed to perform autonomous environment mapping with motion algorithms developed during the task package.

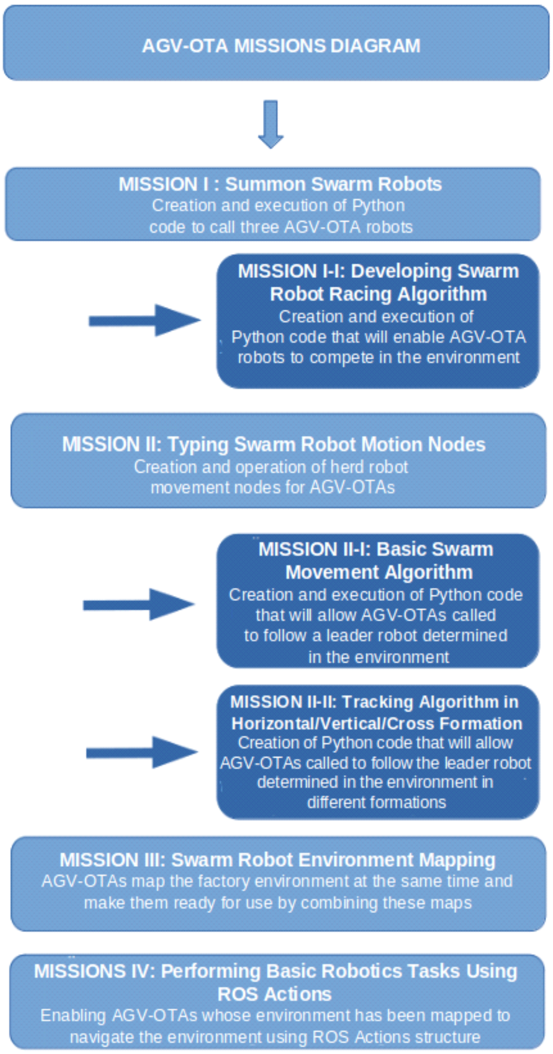
VII. AGV-OTA MISSION TUTORIAL

This tutorial is an educational guide in which the necessary studies for a possible factory mission training scenario designed for "AGV-OTA Robot" are set up and explained. With this simulation training designed to work in a factory model in a Gazebo environment, users are expected to complete this training by performing the ROS and Python-based missions requested from them in various ways of participation. Training topics mainly include robotic motion and herd robot motion algorithms, focusing on the basic coordinated motion algorithms required for a real closed factory, especially on the use of this robot in multiple states.

Figure 31. AGV-OTA

A. Factory Missions

AGV-OTA in-plant service tasks begin with the creation of the spawn files of three AGV-OTA robots and are completed with environmental mapping work using swarm robots. The basis of these tasks is focused on writing the necessary Python codes for the robot to complete the task, creating the launch files or creating the necessary srv/msg files. The details of these tasks are as shown in the diagram below.



AGV-OTA MISSIONS DIAGRAM

MISSION I : Summon Swarm Robots
Creation and execution of Python code to call three AGV-OTA robots

MISSION I-I: Developing Swarm Robot Racing Algorithm
Creation and execution of Python code that will enable AGV-OTA robots to compete in the environment

MISSION II: Typing Swarm Robot Motion Nodes
Creation and operation of herd robot movement nodes for AGV-OTAs

MISSION II-I: Basic Swarm Movement Algorithm
Creation and execution of Python code that will allow AGV-OTAs called to follow a leader robot determined in the environment

MISSION II-II: Tracking Algorithm in Horizontal/Vertical/Cross Formation
Creation of Python code that will allow AGV-OTAs called to follow the leader robot determined in the environment in different formations

MISSION III: Swarm Robot Environment Mapping
AGV-OTAs map the factory environment at the same time and make them ready for use by combining these maps

MISSIONS IV: Performing Basic Robotics Tasks Using ROS Actions
Enabling AGV-OTAs whose environment has been mapped to navigate the environment using ROS Actions structure

Figure 32. AGV-OTA missions

Mission I: Summon Swarm Robots

Mission Purpose: The first task of this training, which is based on the scenario of using the AGV-OTA robot in a closed factory in multiple ways, is a simple Python that will enable to write the launch file that will allow three calls from AGV-OTA to the prepared closed factory environment and to test that all sensors of the called robots are working aims to write the code.

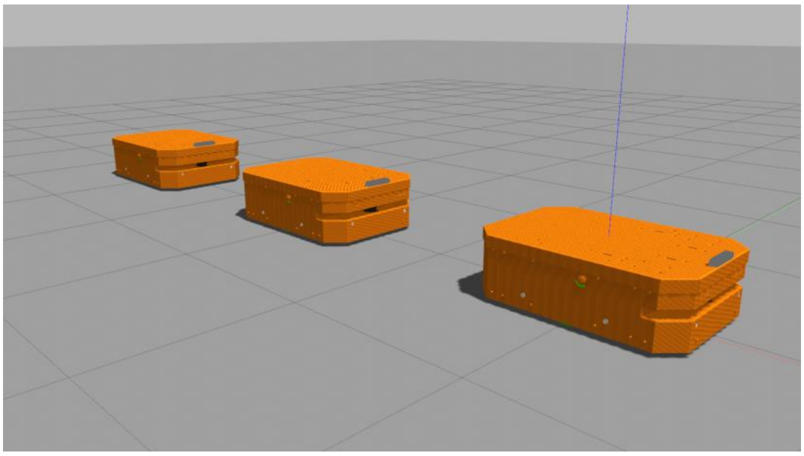


Figure 33. Summoned three AGV-OTA robots

Mission I-I: Developing Swarm Robot Racing Algorithm

Mission Purpose: This first side task of the AGV-OTA factory training missions is one of the sub-tasks created on the basis of the first task, which does not affect the main task series very much. The purpose of these side missions is to enable the user to use the knowledge acquired in the main tasks in a more diverse way. In this first side task, it is aimed for the user to create a lot of robot racing algorithms and apply it on AGV-OTAs. The user, who will ensure that random speeds are given every three seconds for the three robots called side by side, will set a designated zone as the end zone of the race, and the robots that reach there will feedback their degrees to the user.

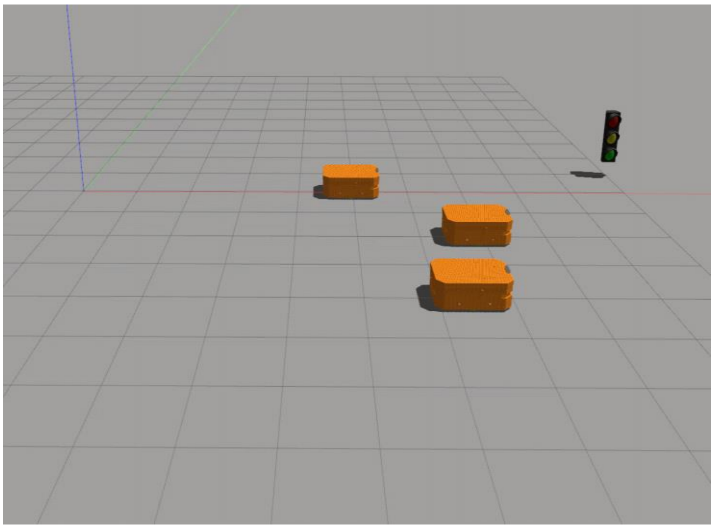


Figure 34. Racing of AGV-OTA's

Mission II: Typing Swarm Robot Motion Nodes

Mission Purpose: In this second task of AGV-OTA factory training tasks, the user is expected to designate one of the AGV-OTAs as the leader robot and perform various movement and formation commands that will enable this leader robot to imitate the movements of the other two robots. These movement and formation types are also divided into two sub-tasks.

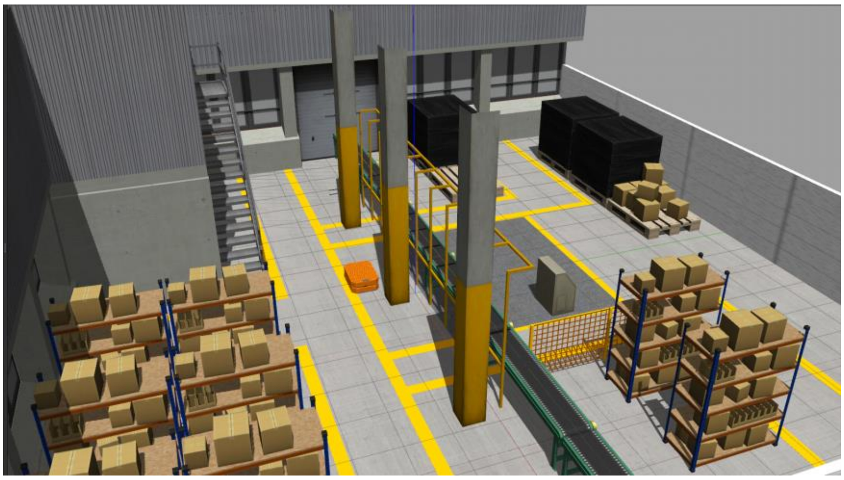


Figure 35. Factory environment of AGV-OTA's

Mission II-I: Basic Swarm Movement Algorithm

Mission Purpose: It is aimed to ensure that AGV-OTAs accompany the movements of the leader robot. The user should observe that he can control all robots via a single robot in order to perform various handling tasks effectively within the factory.

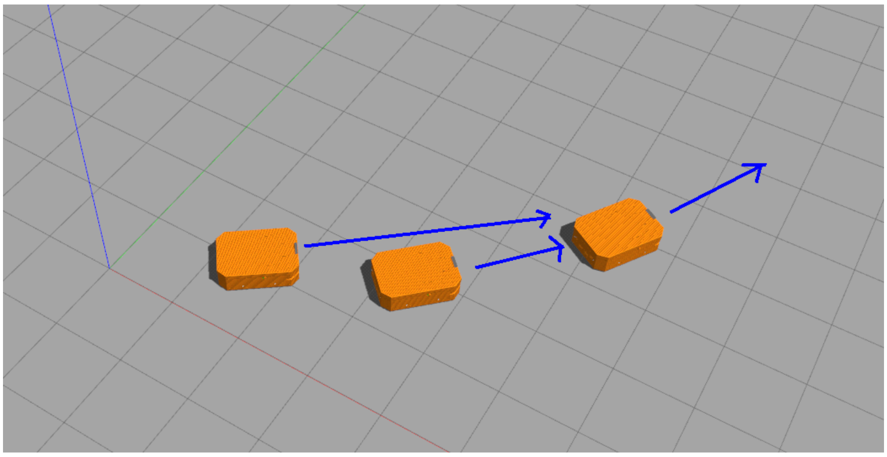


Figure 36. Leading robot (AGV-OTA1) and its followers progress direction

Mission II-II: Tracking Algorithm in Horizontal/Line/Cross Formations

Mission Purpose: In this sub-task, it is aimed to ensure that robots position themselves in different formations compared to the leader robot. These formations should be developed to increase the mobility of robots within the factory.

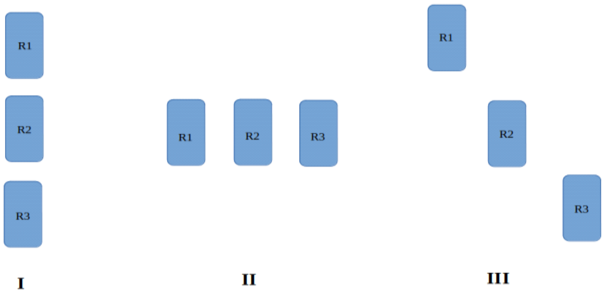


Figure 37. I-Line, II-Horizontal, III-Cross Formations

Mission III: Swarm Robot Environment Mapping

Mission Purpose: In the third of the AGV-OTA factory training tasks, it is aimed for AGV-OTAs to map the factory environment in which they are found. However, since each robot will create its own map separately, these maps must be combined and saved as a single map at the end of the mission. The user is expected to write the code to provide this.

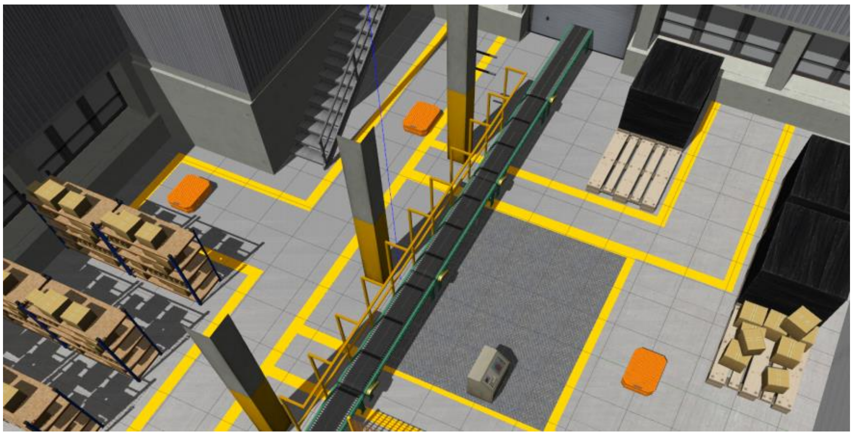


Figure 38. AGV-OTA’s in the factory for mapping

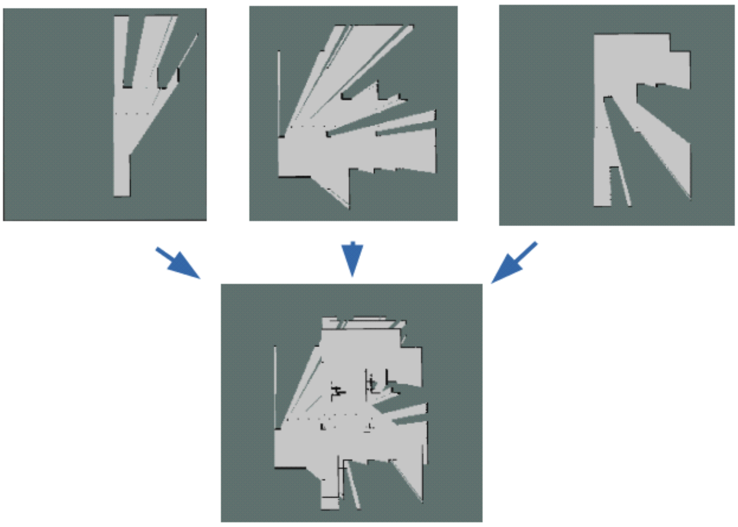


Figure 39. Combining the maps of three robots that they received at first

Mission IV: Performing Basic Robotics Tasks Using ROS Actions

Mission Purpose: In the last of the AGV-OTA factory training tasks, it is aimed to carry out a sample study from the applications of ROS Actions structure to swarm robots. With this work, the user should understand the ROS Actions structure, see the difference with the ROS Service structure and apply it.

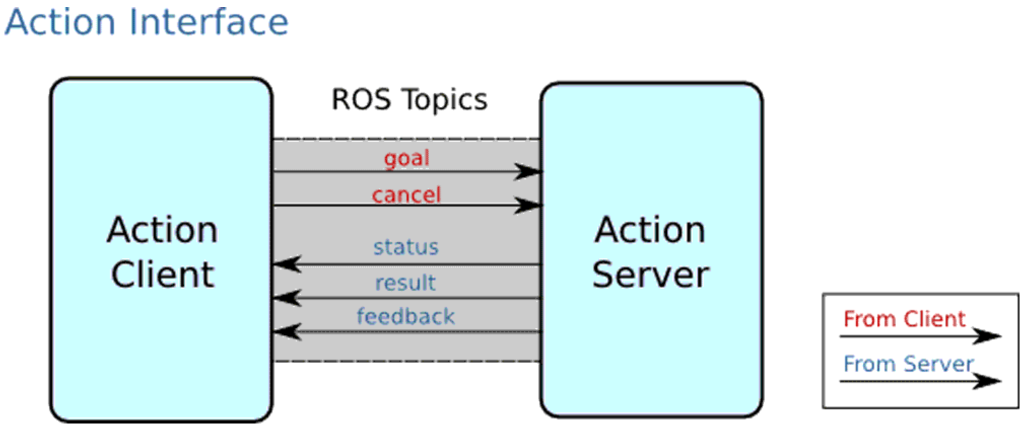


Figure 40. ROS Action interface

VIII. DISCUSSIONS AND CONCLUSION

In this study, virtual robotic laboratory (U-Plat) which is accessible through link https://www.theuplat.com has been proposed. U-Plat is a virtual laboratory platform for learning ROS. U-plat provides tools for more effective educational and research activities compared to conventional approaches to teaching and researches in Robotics. Uplat tutorials can be applied via any internet browser without having to deal with complex ROS and Ubuntu installations on the local computer. All software tools designed and used for robotic education studies are presented with a simplified interface, considering the user experience as comfortable and easy as possible. All designed training tools are provided through a web interface supported by advanced cloud systems and Docker container infrastructure.

EvaMars robot exploration missions and others are presented with tutorials. Completion of all tutorials provides many learning outcomes in ROS and Gazebo.

In the future work, integration API will be developed for Massive Open Online Course (MooC) websites like edx, coursera, udemy, khanacademy etc. Furthermore, U-Plat system will be integrated with the real-world robots and become a remote & virtual robotic laboratory. Besides Mars Robot and Mars missions, U-Plat supports all ROS compatible robots and more ROS compatible robots will be added to U-Plat in near future.

ACKNOWLEDGMENT

This study is supported by İnovasyon Mühendislik Ltd. Şti.

REFERENCES

[1] Yayan, U., Akar, B., Inan, F., & Yazici, A. (2014). Development of indoor navigation software for intelligent wheelchair. In *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*, 325-329.

[2] Edrees, M. E. (2013). eLearning 2.0: Learning management systems readiness. In *2013 Fourth International Conference on e-Learning" Best Practices in Management, Design and Development of e-Courses: Standards of Excellence and Creativity"*, 90-96.

[3] Gomes, L., & Seta, B. (2009). Current trends in remote laboratories. IEEE Transactions on industrial electronics, 56(12), 4744-4756.

[4] Lin, Y., Wang, S., Wu, Q., & Chen, L. (2019). Key technologies and solutions of remote distributed virtual laboratory for E-learning and E-education. *Mobile Networks and Applications*, *24*(1), 18-24.

[5] Schmid, C., Eikaas, T. I., Foss, B., & Gillet, D. (2001). A remote laboratory experimentation network. *IFAC Proceedings Volumes*, *34*(9), 515-520.

[6] Alkhaldi, T., Pranata, I., & Athauda, R. I. (2016). A review of contemporary virtual and remote laboratory implementations: observations and findings. *Journal of Computers in Education*, *3*(3), 329-351.

[7] Tzafestas, S. G. (2009). Web-based control and robotics education.

[8] Santana, I., Ferre, M., Izaguirre, E., Aracil, R., & Hernandez, L. (2012). Remote laboratories for education and research purposes in automatic control systems. *IEEE transactions on industrial informatics*, *9*(1), 547-556.

[9] Minamide, A., Yamada, H., Takemata, K., & Hastie, M. (2012). Redesigned Long-Distance-Controlled Robot System for distance education. In *2012 7th International Conference on Computing and Convergence Technology (ICCCT)*, 472-475.

[10] McKee, G., & Barson, R. (1995). NETROLAB: a networked laboratory for robotics education, 8, 8.

[11] Casini, M., Garulli, A., Giannitrapani, A., & Vicino, A. (2010). A matlab-based remote lab for multi-robot experiments. *IFAC Proceedings Volumes*, *42*(24), 162-167.

[12] Casini, M., Garulli, A., Giannitrapani, A., & Vicino, A. (2014). A remote lab for experiments with a team of mobile robots. *Sensors*, *14*(9), 16486-16507.

[13] Neamtu, D. V., Fabregas, E., Wyns, B., De Keyser, R., Dormido, S., & Ionescu, C. M. (2011). A remote laboratory for mobile robot applications. *IFAC Proceedings Volumes*, *44*(1), 7280-7285.

[14] Xue, X., Yang, S. X., & Meng, M. H. (2005). Remote sensing and teleoperation of a mobile robot via the internet. In *2005 IEEE International Conference on Information Acquisition*, 6.

[15] Petrovič, P., & Balogh, R. (2012). Deployment of Remotely-Accessible Robotics Laboratory. *International Journal of Online Engineering*, *8*.

[16] Yang, X., Petriu, D. C., Whalen, T. E., & Petriu, E. F. M. (2004). A web-based 3d virtual robot remote control system. In *Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No. 04CH37513)*, 2, 955-958.

[17] Casini, M., Chinello, F., Prattichizzo, D., & Vicino, A. (2008). RACT: A remote lab for robotics experiments. *IFAC Proceedings Volumes*, *41*(2), 8153-8158.

[18] Marin, R., León, G., Wirz, R., Sales, J., Claver, J. M., Sanz, P. J., & Fernández, J. (2009). Remote programming of network robots within the UJI industrial robotics telelaboratory: FPGA vision and SNRP network protocol. *IEEE Transactions on Industrial Electronics*, *56*(12), 4806-4816.

[19] Kuzucuoglu, A. E., & Erdemir, G. (2011). Development of a web-based control and robotic applications laboratory for control engineering education. *Information technology and control*, *40*(4), 352-358.

[20] Ionescu, C. M., Fabregas, E., Cristescu, S. M., Dormido, S., & De Keyser, R. (2013). A remote laboratory as an innovative educational tool for practicing control engineering concepts. *IEEE Transactions on Education*, *56*(4), 436-442.

[21] Balestrino, A., Caiti, A., & Crisostomi, E. (2008). From remote experiments to web-based learning objects: An advanced telelaboratory for robotics and control systems. *IEEE Transactions on Industrial Electronics*, *56*(12), 4817-4825.

[22] Masár, I., Bischoff, A., & Gerke, M. (2004). Remote experimentation in distance education for control engineers. *Proceedings of Virtual University*, 16-17.

[23] Šafarič, R., Truntič, M., Hercog, D., & Pačnik, G. (2005). Control and robotics remote laboratory for engineering education. *International Journal of Online Engineering*, *1*(1), 1-8.

[24] Dwivedi, Y. K., Hughes, D. L., Coombs, C., Constantiou, I., Duan, Y., Edwards, J. S., ... & Upadhyay, N. (2020). Impact of COVID-19 pandemic on information management research and practice: Transforming education, work and life. *International journal of information management*, *55*, 102211.

[25] Sagiroglu, S., Yilmaz, N., & Wani, M. A. (2006). Web robot learning powered by bluetooth communication system. In *2006 5th International Conference on Machine Learning and Applications (ICMLA'06)*, 149-156.

[26] Popescu, D., Selisteanu, D., Dinulescu, I., & Popescu, L. C. (2008, July). Web based telematics application for robotics. In *2008 The Third International Multi-Conference on Computing in the Global Information Technology (iccgi 2008)*, 19-24.

[27] Potkonjak, V., Vukobratović, M., Jovanović, K., & Medenica, M. (2010). Virtual Mechatronic/Robotic laboratory–A step further in distance learning. *Computers & Education*, *55*(2), 465-475.

[28] Moody, J. A. O., Alonso, R. E. S., Barbosa, J. J. G., & Morales, G. R. (2016). Virtual laboratories for training in industrial robotics. *IEEE Latin America Transactions*, *14*(2), 665-672.

[29] Ghayoor, F. (2020). A MATLAB-based virtual robotics laboratory: Demonstrated by a two-wheeled inverted pendulum. *The International Journal of Electrical Engineering & Education*, *57*(4), 301-320.

[30] Suwasono, S., Prihanto, D., Wahyono, I. D., & Nafalski, A. (2017). Virtual Laboratory for Line Follower Robot Competition. *International Journal of Electrical and Computer Engineering*, *7*(4), 2253.

[31] Lin, M., San, L., & Ding, Y. (2020, March). Construction of robotic virtual laboratory system based on Unity3D. In *IOP Conference Series: Materials Science and Engineering*, 768(7), 072084.

[32] Peidró, A., Tendero, C., Marín, J. M., Gil, A., Payá, L., & Reinoso, O. (2018). m-PaRoLa: a Mobile Virtual Laboratory for Studying the Kinematics of Five-bar and 3RRR Planar Parallel Robots. *IFAC-Papers On Line*, *51*(4), 178-183.

[33] Favario, L. (2018, December). Remote Programming Environments: the Robotic Laboratory Case. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 820-824.

[34] Jara, C. A., Candelas, F. A., & Torres, F. (2008). Virtual and remote laboratory for robotics e-learning. In *Computer Aided Chemical Engineering*, 25, 1193-1198.

[35] Román-Ibáñez, V., Pujol-López, F. A., Mora-Mora, H., Pertegal-Felices, M. L., & Jimeno-Morenilla, A. (2018). A low-cost immersive virtual reality system for teaching robotic manipulators programming. *Sustainability*, *10*(4), 1102.

[36] Xu, X., Guo, P., Zhai, J., & Zeng, X. (2020). Robotic kinematics teaching system with virtual reality, remote control and an on–site laboratory. *International Journal of Mechanical Engineering Education*, *48*(3), 197-220.

[37] Yayan, U. (2020). *Virtual Robotic Laboratory (Uplat) Cloud Challenge* [Video]. Youtube. https://youtu.be/rggLZCkaN6k

[38] Helming, Jonas; Koegel, Maximilian (2018-06-20). *"What is Eclipse Theia?"*. EclipseSource. Retrieved 2022-03-12.