Implementation of PLC controller connected Gazebo-ROS to support IEC 61131-3

Yungjun Kim
Korea Electronics Technology Institute
Seongnam, Republic of Korea
hkrboy@keti.re.kr

Seung-yong Lee
Korea Electronics Technology Institute
Seongnam, Republic of Korea
michalen2006@keti.re.kr

Sun Lim
Korea Electronics Technology Institute
Seongnam, Republic of Korea
sunishot@keti.re.kr

***Abstract*—IEC61131-3 is an international standard related to programming languages used to develop PLC control systems. This paper introduces a system that can verify and validate a process automation program developed in compliance with IEC61131-3 using a 3D simulator. Robot driven commands and codes related to object manipulation communicate with Gazebo-ROS system to verify the automation program. To this end, the PLC controller and Gazebo-ROS exchange data using TCP/IP communication protocol and ROS nodes and Linux threads are developed to translate ROS-type messages into PLC controller messages. Gazebo-ROS integrated software architecture is developed after designing the task allocation algorithms using the IEC61131-3 language to verify the developed PLC program. In particular, the task allocation algorithms consist of three static algorithms, each of which is used for a specific scenario, and one dynamic algorithm that interchanges between the three static algorithms given some variation in the task scenarios.**

***Index Terms*—PLC, IEC 61131-3, Gazebo-ROS, simulation, industrial automation, task allocation**

I. Introduction

Standardized programming environment has become an essential tool in the recent years, as the demand for industrial process automation is rapidly increasing [1], [2]. In general, the standardized tool allows for program reusability as well as its development and integration. The need for such standardization has been further emphasized as more large-scale distributed systems are being developed and interlocked with real-time industrial applications, such as OPC-Unified Architecture (OPC-UA) and Industrial Internet of Things (IIOT). In line with this trend, PLCopen Technical Committees are actively carrying out various standardization designation activities, such as PLC Programming IEC 61131-3 [3], PLC motion function block, and PLCopen OPC-UA information model.

3D simulation software is another essential tool used in the development of process automations; simulation programs provide a virtual environment in which automation programs can be tested. This process of testing the developed program in process automation reduces the time and costs of physical testing and allows developers to analyze their program's effect on the whole system, consequently simplifying the process of software verification and validation (V&V). Given the economic benefits, the industrial 3D simulation solutions are being actively developed to provide highly accurate physics engine that is indistinguishable from the actual industrial settings. In addition to the growth of the industrial simulator market, today's researches in integrating PLC software with ROS [4], [5] brings about novel technologies related to industrial simulations.

However, contemporary software V&V solutions often lack the ability to validate programs by interlocking a PLC controller and a 3D simulation software. Prior to writing the programs, simulation programs that support Offline Programming Language (OLP) develop and validate the system using the script language provided by the robot controller. Such method can provide V&V for a single robot application, but there are limits to what it can do for a large-scale process level. The issue of scalability can be resolved by supporting a separate PLC language to provide the simulator functions, but it becomes more complex to reuse a program when the PLC controller is modified or replaced on-site, as the PLC languages are not standardized. Moreover, if a developer who is familiar with the PLC Integrated Development Environments (IDEs) is developing a simulation program, the burden for simulator development may increase because the developer is required to know the simulator tools.

In order to overcome the problem of software V&V using 3D simulations, this paper proposes a process development verification system using the IEC 61131-3 IDE and the open-source, ROS integrated Gazebo simulator. In particular, the task allocation algorithms that are frequently used in today's industrial processes are developed in the IEC 61131-3 programming language, and the environment that can be verified on Gazebo simulation is presented: the task allocation algorithms are developed using IEC 61131-3 to control the operation of two robots, and the Gazebo-ROS interface is provides a virtual V&V solution for the developed algorithms. This paper also presents the experimental results from different scenarios in which the PLC controller determines which robot to process a simple pick-and-place task after placing arbitrary objects in the simulation environment. The results confirmed that the algorithm in the PLC controller works quite well in all scenarios, and that the task allocation algorithm that had been developed on PLC editors can be validated in Gazebo-ROS environment.

This paper is categorized as the following: Chapter 2 presents the detailed design for the robot task allocation algorithm used in industrial process automation developed in

compliance with the standard IEC 61131-3 as well as the implementation of Gazebo-ROS system to the PLC controller, Chapter 3 shows the actual experimental results, and Chapter 4 concludes this paper.

II. PLC Controller connected Gazebo-ROS Design

A. System Architecture

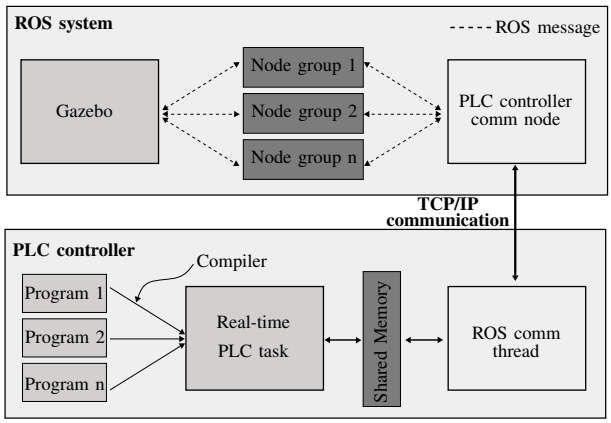

Fig. 1. System architecture for PLC connected Gazebo-ROS

This paper proposes a system that can validate a process automation application programs written in accordance with IEC 61131-3 using Gazebo-ROS interface. Most of today's process automation programs are developed with regards to the international standard for their reusability and generality. Simulating the developed program prior to applying it to an actual process automation can greatly reduce the development/integration time as well as the cost. This speaks to the recent emphasis of OLP and the trend of how different manufacturing tools are starting to support simulator functions.

To develop a process automation program that abides by IEC 61131-3 and to validate the framework through ROS and Gazebo, a system architecture is designed as in Fig. 1. For the PLC IDEs, process automation programs like PLC and manufacturing programs are developed. For ROS and Gazebo, a typical manufacturing environment, manufacturing robots, and target objects are modeled in an XML data file and simulated using Gazebo. Lastly, the Gazebo-ROS system and PLC controller communicate with each other to validate the developed process automation programs.

The PLC controller performs I/O control, instructs robots with tasks, and provides integrated control for process peripherals. However, robot kinematics and motion planning algorithms are generally different for different robot manufacturers, which calls for robot's dependency on its controller. Therefore, the ROS system in Fig. 1 utilizes an open-source robot motion planning framework, namely MoveIT. The PLC controller commands the robot to follow the trajectory defined by MoveIT library based on robot geometry and target position. In addition, the rest of the equipment is developed in the form of a standard PLC program and also controlled by the PLC controller.

B. IEC 61131-3 Standard

IEC 61131-3 defines the standard from a PLC developement perspective to develop process automation applications. It includes two text languages and three graphic language definitions, program organization units (POUs), and data types. The standard was written to develop a system using standardized process equipment that can easily reuse and quickly develop the PLC programs used in process automation systems. Following the increase in demand for smart factories, many PLC controller manufacturers have started to adopt the above standard by supporting automatic conversion or expansion of their own PLC language to the IEC 61131-3 language.

The programs in Fig. 1 is developed using the five programming languages defined in IEC 61131-3. Among these languages, FBD language or SFC language in the form of a function block is often used to control large-scale processes, and ST language is generally used for developers who are familiar with programming languages such as C or C++. Other examples include assembly language IL and ladder type graphic programming language LD. The PLC IDE defines the compiler internally so that the PLC programming can be operated on the PLC controller.

C. Task Allocation Algorithms for Multi-Robot Automation

Developing process automation that uses multiple robots requires each robot's motion to be defined based on specific task conditions, such as target object's position, object's instantiation time, and the robot's task space. To realize this, the task allocation algorithm serves the purpose of distributing the tasks among multiple robots. A static task allocation algorithm uses the first algorithm set on the robots throughout the process, whereas a dynamic algorithm can change in between different algorithms according to the changes in the process in real time.

Table I presents a pseudo-code representation of the three developed static task allocation algorithms. To effectively communicate with and allocate required tasks to the robots, the algorithms take the vector of positions of instantiated target objects *obj_position* in the Gazebo environment as their input. Upon receiving the input, the task allocation algorithms are executed only if the input vector is not empty or a new object has been added to the vector. Given the target object positions, the algorithms output two goal position vectors, *robot1_goal* and *robot2_goal*, based on the conditions given by each algorithm, and sends the data back to ROS for Gazebo simulation.

Table I (a) is an algorithm based on object positions in which their x position determines which robot to use for the pick-and-place task. For example, if an object is spawned at some x position that is greater than the specified x value *designated_xposition*, then the object position is added to robot1_goal vector so that robot 1 can manipulate the object in Gazebo (line 6-9). Table I (b) is based on the availability of robot 1 in which only robot 1 performs the pick-and-place task only if it is not currently in motion. To realize this, an additional input is required for algorithm (b), namely a bool

type data which returns true if robot 1 is currently not in motion and is available to use (line 7-10). Table I (c) is based on equally distributing the pick-and-place task between the two robots, in which each robot is alternatively used when a new object is added to the Gazebo environment.

Line 1 to 5 for all three algorithms are the same; the IF statement in line 1 makes sure that when there is no object instantiated in Gazebo the PLC program returns no value. The clear() function (line 4-5) makes sure that only the new objects' positions are added to the output vectors. When an object or multiple objects are added, the FOR statement (line 6) iterates through every index of the input vector *obj_position* and decides which robot should carry out the pick and place task based on conditions like object position and robot availability (line 7-9). The decision made from the specific algorithm in use then constructs a vector of target positions, *robot1_goal* and *robot2_goal*, by calling the pushback function and sends the message back to ROS for simulation

D. ROS Interface

ROS is an open-source, distributed framework of processes called ROS nodes, which communicate with other nodes to execute functions like hardware control, motion planning, and sensor integration. Gazebo is an open source. stand-alone, 3D dynamic simulator that offers physics simulation capable of various functions like testing robotics algorithms. Integration of ROS with Gazebo allows simulation of robots using ROS messages, services, and dynamic reconfigure. The ROS node group in Figure 1 represents ROS-Gazebo interface in which each group refers to a group of ROS nodes required for simulating a specific robot. For the experiment, each group was designed to include ROS nodes capable of robot control and motion planning to execute simple pick-and-place task using open-source MoveIT library. In addition to the ROS-Gazebo interface, PLC controller communication node was developed to send target object position vector *obj_position* to and receive goal position vectors *robot1_goal* and *robot2_goal* from the PLC controller using TCP/IP protocol.

III. EVALUATION

In order to validate the system proposed in this paper, the following is implemeneted. Beremiz IDE [6] solution, an open source IEC 61131-3 IDE project was used as a PLC editor; the Beremiz solution supports 5 PLC programming languages that is based on the standard. First, the internal compiler converts the written PLC program into an ANSI C program so that the PLC controller is operable. Second, the PLC controller is developed using Xenomai open source project [7], a Linux based real-time operating system. In doing so, a PLC task was designed to be able to control in real time using Xenomai API, and ROS communication thread was developed in the form of Linux posix thread. Lastly, the task allocation algorithm was designed as in Fig. 2 to simulate the algorithms through ROS-Gazebo system.

TABLE I
TASK ALLOCATION ALGORITHM FOR MULTI-ROBOT PROCESS

```
input: vector of Cartesian target positions as obj_position
input: availability of robot1 as robot1_state
output: vector of target positions for Robot 1 as robot1_goal
output: vector of target positions for Robot 2 as robot2_goal
begin procedure /* (a)Object Position Based() */
1.    IF obj_position.size() == 0 THEN
2.       return 0;
3.    ELSIF obj_position.size() >0 THEN
4.       robot1_goal.clear();
5.       robot2_goal.clear();
6.       FOR i=0 to obj_position.size()-1 BY 1 DO;
7.          IF obj_position[i].x >= designated_xposition THEN
8.             robot1_goal.pushback(obj_position[i]);
9.          ELSIF obj_position[i].x <designated_xposition THEN
10.            robot2_goal.pushback(obj_position[i]);
11.         END_IF
12.      END_FOR
13.   END_IF
end procedure
begin procedure /* (b)Robot 1 Availability Based */
1.    IF obj_position.size() == 0 THEN
2.       return 0;
3.    ELSIF obj_position.size() >0 THEN
4.       robot1_goal.clear();
5.       robot2_goal.clear();
6.       FOR i=0 to obj_position.size()-1 BY 1 DO;
7.          IF robot1_state == true THEN
8.             robot1_goal.pushback(obj_position[i]);
9.          ELSE THEN
10.            robot2_goal.pushback(obj_position[i]);
11.         END_IF
12.      END_FOR
13.   END_IF
end procedure
begin procedure /* (c)Alternation Based */
1.    IF obj_position.size() == 0 THEN
2.       return 0;
3.    ELSIF obj_position.size()>0 THEN
4.       robot1_goal.clear();
5.       robot2_goal.clear();
6.       FOR i=0 to obj_position.size()-1 BY 1 DO;
7.          IF (i+1)%2 != 0 THEN
8.             robot1_goal.pushback(obj_position[i]);
9.          ELSIF (i+1)%2 == 0 THEN
10.            robot2_goal.pushback(obj_position[i]);
11.         END_IF
12.      END_FOR
13.   END_IF
end procedure
```

TABLE II
DETAILS OF PLC IDE CONNECTED GAZEBO-ROS

Item	Description
PLC Controller	
CPU	Intel Core i5 6500 operating at 3.2 GHz
Memory	4GB DDR4 DRAM
Operating system	Linux Ubuntu kernel 3.4.13 with Xenomai 2.6.3 RTOS
Network	Intel EXPI9301CTBLK 1GB/s Ethernet adapter
Gazebo-ROS system	
CPU	Intel Core i7-7500U operating at 2.7 GHz
Memory	16GB DDR4 DRAM
Operating system	Linux Ubuntu 14.04 LTS
Software	ROS Indigo, Gazebo 2.0, MoveIT

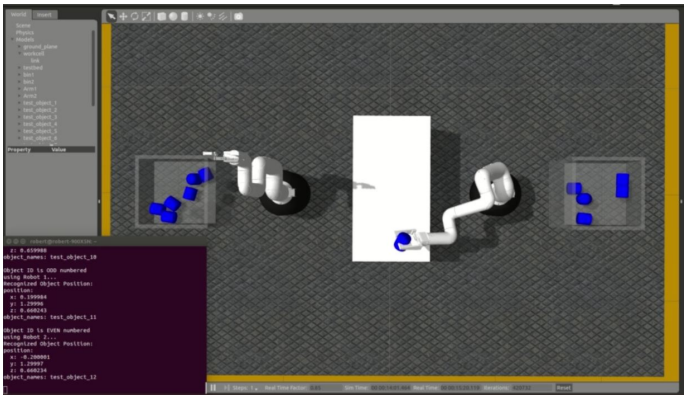

Fig. 2. Gazebo-ROS experimnet environment

The experimental method is as follows. On the PLC editor, the algorithm proposed in Table I is developed using ST language, a language based on IEC 61131-3 standard. Then, the algorithms are executed on the PLC controller. A target object is spawned randomly in the Gazebo simulation environment, and the position of the spawned objects is transmitted to the PLC controller. The PLC controller uses the algorithms to determine which robot to process the pick-and-place task and transmits the command to each robot in the Gazebo-ROS system. After executing the task in simulation, the Gazebo-ROS system waits for the spawning of a new target object to repeat the process. Figure 2 presents a GUI of the Gazebo program simulating two robots using one of the algorithms and a Linux terminal that prints the outputs from the PLC controller.

Prior to system implementation and testing, a dynamic task allocation algorithm is also developed for the experimental scenario. The dynamic task allocation algorithm is one that can change in between algorithms if certain conditions are met. For example, it would be much more efficient to use algorithm (b) if target objects are consecutively spawned at a timeframe in which robot 1 is always available; alternatively, algorithm (a) should be used if the objects are consecutively spawned at some position. For this experiment, the dynamic task allocation algorithm is designed to use algorithm Table I (c) as the default. When three objects are consecutively spawned whenever robot 1 is available, the dynamic algorithm switches to algorithm Table I (b), and when three objects are consecutively spawned at some specific position, position based algorithm Table I (b) is used. On the other hand, if the succeeding objects are spawned without satisfying the above conditions, it switches back to using algorithm Table I (c). Thus, the dynamic task allocation algorithm can transition between the three static algorithms to optimize the usage of multiple robots and efffectively react to the changes in the automation process.

The experimental conditions are stated on Table II. As a result of spawning 20 objects at a random position and instantiation time in the Gazebo environment, the four task allocation algorithms showed an accuracy of 98.71% in which the two robots were able to successfully execute the pick-and-place task. This amounts to the ability of Gazebo-ROS system to verify and validate the program developed in PLC IDE. In addition, the Gazebo-ROS system is able to provide V&V for a scenario as complex as the one used in dynamic task allocation.

IV. Conclusion and Future Work

This paper presents an integrated system that can verify and validate process automation programs developed in compliance with IEC 61131-3 standards through 3D simulations. The PLC controller communicates with the Gazebo-ROS simulator in real time to exchange various messages to verify the automation program; the programs developed using the international standard operates on the PLC controller, whereas multi-robot process is designed using Gazebo-ROS interface. In oder to test the 3D simulator's V&V capability, three types of industrial robotic algorithms that are widely used in process automation are developed. In addition, a dynamic task allocation algorithm that can change algorithms according to changing process conditions to react to more complicated automation scenarios in real time is also developed and tested.

The results above confirm how Gazebo-ROS system can provide verification and validation of the PLC program developed in the IEC 61131-3 IDE. For future research, a PLC scenario program that integrates peripheral I/O devices with robots in Gazebo-ROS system will be developed. Using the PLC program, the reliability of the developed programs will be tested by designing a simulation program that is similar to the actual process environment.

V. ACKNOWLEDGMENT

This work was supported by the Technology Innovation Program (20005055, Development of Integrated Control Technology supporting both control of 4 robots and PLCopen standard-based PLC language(5 types) for SME's smart factory) funded By the Ministry of Trade, Industry & Energy(MOTIE, Korea).

References

[1] Tiegelkamp, Michael, and K. H John. IEC 61131-3: Programming industrial automation systems. New York, Springer-Verlage BH, 1995.
[2] M. De Sousa, "Proposed corrections to the IEC 61131-3 standard," Computer Standards & Interfaces, vol. 32, no. 5-6, pp. 312-320, 2010.
[3] International Electrotechnical Commission," International Standard IEC 61131-3, Programmable Logic Controllers Part 3 – Programming Languages, edition 2, " Geneva, 2003.
[4] M. De Sousa and H. Sobreira, "On adding IEC 61131-3 support to ros based robots," IEEE Emerging Technologies & Factory Automation (ETFA), pp. 1-4, 2013 [2013 IEEE 18th conference].
[5] T., Arrais, R., and Veiga, "Bridging Automation and Robotics: an Interprocess Communication between IEC 61131–3 and ROS, " IEEE 16th International Conference on Industrial Informatics, pp. 1085-1091, 2018.
[6] "An Open Source IEC 61131-3 Integrated Development Environment," IEEE International Conference on Industrial Informatics, pp. 24-26, 2007.
[7] Xenomai - An open source real-time OS based Linux, [online] Available:https://gitlab.denx.de/Xenomai/xenomai/-/wikis/home.