Software Engineering
2023; 10(1): 1-5
http://www.sciencepublishinggroup.com/j/se
doi: 10.11648/j.se.20231001.11
ISSN: 2376-8029 (Print); ISSN: 2376-8037 (Online)

Survey Paper on Development of ROS for Fault Detection of Underwater Cables

Vijay Rathod, Haritima Kushwaha, Teheseen Shaikh, Vaishnavi Joshi, Shubham Awantkar

Department of Computer Science and Engineering, G. H Raisoni Institute of Engineering and Technology Pune, Pune, India

Email address:
teheseen26@gmail.com (Teheseen Shaikh)

To cite this article:
Vijay Rathod, Haritima Kushwaha, Teheseen Shaikh, Vaishnavi Joshi, Shubham Awantkar. Survey Paper on Development of ROS for Fault Detection of Underwater Cables. *Software Engineering*. Vol. 10, No. 1, 2023, pp. 1-5. doi: 10.11648/j.se.20231001.11

Received: January 26, 2023; **Accepted**: March 17, 2023; **Published**: May 18, 2023

Abstract: An introduction to ROS, an open source robot operating system, is given in this paper. In terms of process management and scheduling, ROS is not an operating system. This study explains how Robotic Operating System (ROS) can be used to control items (such as vehicles) remotely and cautiously without human intervention at the location. Instead, it gives heterogeneous computing clusters a structured communication layer on top of the host operating system. This document gives a quick explanation of how ROS fits into the current robot software architecture and how we may utilize it for AUVs (Automated under water vehicle). One of the media that connects the entire world to the internet is optical cable, which is typically installed underground or under water. As a result, it is challenging to inspect them thoroughly because it costs more to do so. To address this issue, we are presenting a solution that involves developing a robotic operating system that would assist in checking the underwater/underground cables. To put this into practice, we have been utilizing VMWare Workstation to virtually install Ubuntu OS, where we will be installing ROS packages, with the ROS-Gazebo toolbox serving as one of the primary tools. We are testing the implemented software with the standard inputs. We are using light radiation as the primary factor to assess the condition of the optical cable.

Keywords: Ubuntu, Gazebo, ROS, Simulations

1. Introduction

Writing software for robots is challenging, especially as robotics' scale and scope continue to expand. Because different robot types can have highly varied hardware, code reuse is not simple. Additionally, given that the required code must span a wide range of levels, from driver-level software to perception to abstract thought, etc., its bulk can be overwhelming. Large-scale software integration initiatives must be supported by robotic software architectures as well because the required competence is well beyond the scope of a single researcher. Many robotics researchers, including our own, have developed numerous frameworks to manage complexity and promote quick development of experimental software in the past to solve these issues. Did. Industry and academia [1]. Each of these frameworks was created with a specific objective in mind, possibly to correct perceived flaws in existing frameworks or to concentrate on elements judged most crucial throughout the design process. was planned. ROS operates as a publish subscribe service to distribute data among nodes in a system [2]. The ROS framework, which is detailed in this article, will be used by AUVs to find problems with underwater cables.

2. Underwater Cable and Fault Detection

Early AC power systems' "inseparable binomial" was long power lines and catenary lines [3]. A fault is defined as a total loss of synchronization or failure of the electrical network, but it does not rule out environmental threats like electrocution or potentially disastrous fire hazards. This demonstrates the common notion at the time that overhead cables served as the primary means of signal transmission. For DC underwater link applications, insulated high voltage (HV) and extra high voltage (EHV) cables were employed. Greater environmental consciousness, growing interference with overhead power lines, and increased reliance on

premium extruded insulation all contributed to this. In the electrical sector of several nations, grid operators are thinking of switching out these overhead wires and cables with underground cables or introducing hybrid systems (i.e., replacing overhead and underground cables) [4]. There are numerous factors, including price. Some of the advantages of Montage are listed below. 1. Considerably lower possibility of being damaged by environmental risks like lightning, wind, and ice, to mention a few. Second, underground cable networks narrow the spectrum of electromagnetic fields that are released (EMF). Underground cables are installed, although there are fewer components. When additional components are positioned adjacent to overhead lines for protection, maintenance, or repair, this is reversed. 4. Underground cable systems remove the possibility of injuring wildlife and in-flight aircraft. 5. Unauthorized contact, sabotage, and conductor theft less than 6. Large trees can be planted and grown in green countries using the Underground Cable System. In some circumstances, the advantages of phasing in an underground system outweigh the disadvantages. The difficulty of locating a problem in the event of one would be one of the most glaring and practical disadvantages of the subsurface system concept. Many low and medium voltage distribution lines throughout the world have been equipped with underground cables for many years. Because they can survive weather, torrential rain, storms, snow, and pollutants, high voltage underground cables are being utilized more and more. Despite advancements in cable manufacturing techniques, cables can still fail during testing and operation for a variety of reasons. If properly placed and maintained, a good cable can survive for roughly 30 years. However, incorrect installation or malfunction can swiftly ruin cables, and Excavation and embankment work during construction can harm later-arriving third parties.

3. Nomenclature

Nodes, messages, subjects, and services are the fundamental principles for implementing ROS, where nodes are processes that carry out calculations [5]. ROS is intended to be modular in minuscule steps. Typically, a system has a large number of nodes. The terms "node" and "software module" are synonymous in this context. The term "node" is used in relation to the runtime visualization of ROS-based systems. When numerous nodes are active, it is important to graph peer- to-peer relationships. Both peer nodes and graph nodes are processes. using a call-based peer-to-peer connection. Nodes exchange messages with one another to communicate. A strongly typed data structure is a message. There is support for arrays of primitive types and constants, as well as the common primitive types (integers, floating-point numbers, Booleans, etc.). Messages can be composed of arrays of other messages and other messages themselves, nested to arbitrary depth. A node posts a message to a certain subject, which is a short string like "odometry" or "map," in order to disseminate a message. Nodes subscribe to a subject when they are interested in a specific form of data. A single node may publish to and/or subscribe to multiple topics concurrently, and there may be multiple concurrent publishers and subscribers to a single topic. Most of the time, publishers and subscribers are completely unaware of one another.

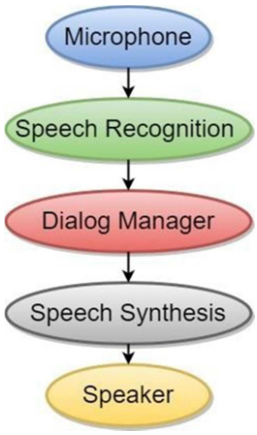

Figure 1. *Structure of voice recognition interface and dialogue system.*

Through pipelines, communication is at its most basic.

Graphs, on the other hand, are typically significantly more intricate, frequently containing cycles and one-to-many or many-to-many links [11]. Although the topic-based publish-subscribe model provides a flexible paradigm for communication, its "broadcast" routing technique does not work well for synchronous transactions even though it may make some nodes' designs simpler. This is referred to as a service in ROS. A pair of highly typed messages, one for requests and one for responses, along with a string name are used to identify a service. This is comparable to a web service with well-defined types of request and response documents that is defined by a URI. Notably, only one node can promote a service with a given name, unlike topics.

4. ROS Framework Design

The ROS Robot Operating System is the primary software framework used in this robot. It is a framework composed of several programmers and libraries that are integrated with one another, greatly reducing the complexity of programming. The ROS is utilized in this AUV to facilitate communication between all of the applications, from the detection software to the PPM output programmer produced by the Arduino [12]. There are 4 nodes and 4 subjects with various functions in this ROS system.

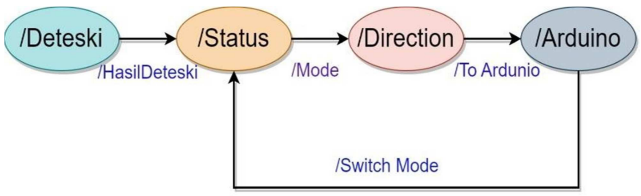

Figure 2. *Basic structure of ROS System.*

The Deteksi-node is the first node. A publisher for object

detection is present in this node, and it publishes information to the Status node via the Deteksi outcome topic. The Status node comes next. The following nodes' publish and subscribe programmes are located on this node and are used to read, combine, and publish data for those nodes: This node subscribes to data from the SwitchMode subject from the Arduino node and the ResultDeteksi topic from the Deteksi node. This node's programming algorithm is: The node emits data in the form of integers retrieved from the Detect node if the switch points to 1. The node will output data in integer format if the switch value is 0. In this instance, the author set the node to emit 1000. so that the following node may compare and analyses the data with ease later. The Status node's programming algorithm is shown in the flowchart.

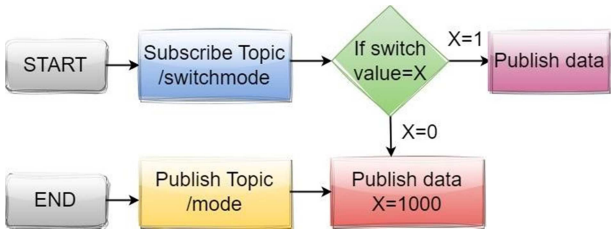

Figure 3. *Status Node's programming algorithm.*

The /Direction node is the following node. A programmer for simplifying and determining the robot's direction of motion is contained in this node. The /Mode topic from the /Status node before this one is subscribed to by this node. The programmer algorithm in the /Direction node is shown below.

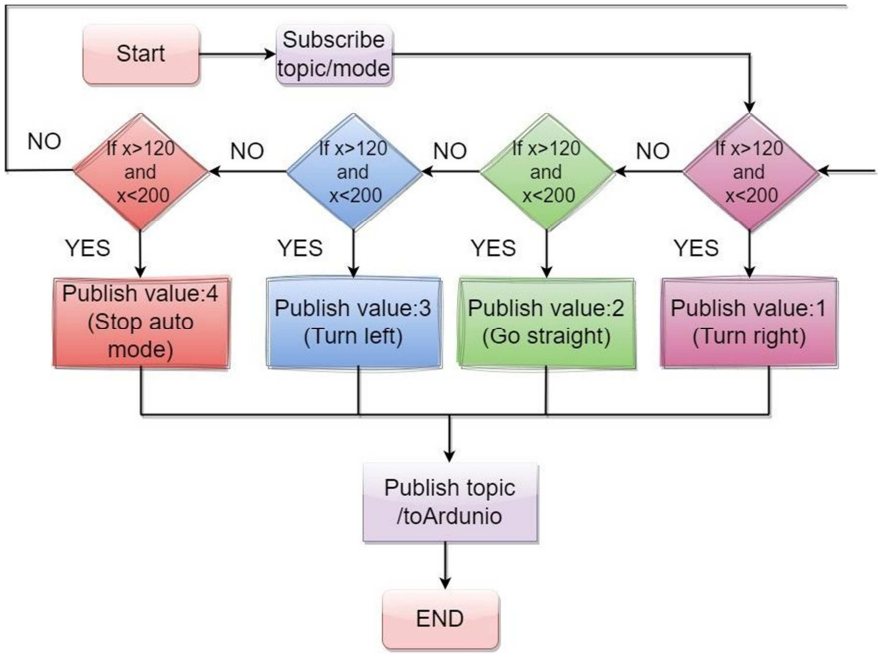

Figure 4. *Direction Node's programming algorithm.*

5. Proposed Methodology

Table 1. *Pseudo Code.*

Task	Pseudo code

Initializing ROS Node	int main (int argc, char **argv)
	{
	ros::init(argc, argv, "name_of_node")
	.....
	}

Printing Messages in a ROS Node	ROS_INFO (string_msg,args): Logging the information of node ROS_WARN (string_msg,args): Logging warning of the node ROS_DEBUG (string_msg,args): Logging debug messages ROS_ERROR (string_msg,args): Logging error messages ROS_FATAL (string_msg,args): Logging Fatal messages
Creating a Node Handle	ros::NodeHandle. ros::NodeHandle nh;
Publishing a Topic in ROS Node	ros::Publisher publisher_object = node_handle.advertise<ROS message type >("topic_name",1000) publisher_object.publish (message)

Callback Function in ROS Node	Void callback_name (const ros_message_const_pointer &pointer)
	{
	// Access data pointer->data
	}

The AUV used in the earlier research is built to be able to dive dynamically while autonomously locating and following underwater cables [10]. While doing this study, a virtual computer running Ubuntu was using a robotic operating system. The orders that cause the stimulated robot (Turtle bot) to move on the ROS-Gazebo toolbox will be inputs, and the CMD (Command line) Interface will be shown as an output. Our presumptive vehicle, which is the AUV, will be the stimulated robot (Turtle bot) in the Gazebo toolkit. It will be able to command line control that stimulated robot (Turtle bot), and as a result, the output/movements of that robot will be visible (assumed AUV). We may view the presumed AUV's readings and look for cable issues using the command line. Additionally, the robot utilised in this study is distinct from the robots used in earlier investigations, which use AUVs (Turtle bot). To offer input to ROS, the Gazebo simulator simulates robot hardware in software [7]. A low-cost, open-source robot kit called TurtleBot is available. The same duties can be carried out by the TurtleBot robot simulation without endangering the actual robot. Learning ROS and testing robot algorithms are done using the TurtleBot robot simulation.

6. Literature Survey

Middleware rt Robotics Technology Middleware (RT-middleware) is a common platform standard for robotics that is based on distributed object technology [6]. RTmiddleware assists in the creation of various networked robotic systems by the incorporation of various network-enabled robotic parts known as RT-Components [8]. The RT-component specification standard is discussed and decided by the Object Management Group. 2. AUV The performance of the object detection system utilising the OpenCV library is good under ideal conditions, with a light intensity of greater than 25 lux. The AUV cannot track in a straight line of 2 metres in less than 25 lux of light. The detection mechanism operates most effectively at speeds of 0.27 to 0.42 m/s. The vertical motion control system and speed control are less stable as a result of the absence of PID control. [9] The AUV can track on a straight and wavy course of 2 metres in bright light at 493 lux, low light at 107 lux, and dark light at 25 lux thanks to an LED beam with a light intensity of 773 lux. The scoping experiment's three straight and curved track trials had a success rate of 75%.

7. Software Requirements

UBUNTU: Ubuntu is a Linux distribution based on Debian that uses mostly free and open-source software. The three official editions of Ubuntu are Core for robotics and Internet of Things devices, Server, and Desktop.

TOOLBOX FOR ROS-GAZEBO: The total toolbox usage is organised and streamlined using a Robotic Operation System (ROS) manager node. A plugin replicates the internal motor dynamics and implements the specific actuator's particular compliance properties. The toolkit can be used to design different compliant joint structures to undertake precise and accurate simulations of ASRs, including those in which they interact with the environment.

RVIZ: With the help of the three-dimensional visualizer rviz, robots, the settings in which they operate, and sensor data can all be observed. It has several different visualisations and plugins, making it a very flexible tool.

ROSBAG: To record and playback ROS message data, rosbag is a command-line tool. In order to capture and log ROS communications, rosbag uses a file format called bags to listen to topics and record messages as they come in [14]. Since playing messages back from a bag is essentially equal to having the original nodes that produced the data in the ROS computation graph, bags are a useful tool for storing data that will be utilised in future development. While rqt bag includes a graphical user interface, rosbag is a command-line-only programme.

CATKIN: Rosbuild has been replaced with catkin as the ROS build mechanism as of ROS Groovy. Catkin is cross-platform, open-source, and language-neutral, much like CMake. ROSBASH: The rosbash package contains a number of tools that enhance the functionality of the bash shell. These tools, which comprise rosls, roscd, and roscp, mimic the actions of ls, cd, and cp, respectively. When utilising the ROS versions of these utilities, the file path where the package is located can be changed to ros package names. Additionally, the package contains rosrun, which runs executables in ROS packages, and rosed, which edits a given file using the selected default text editor. Most ROS tools now also feature tab completion. The same functionalities are supported by rosbash to a lesser extent for zsh and tcsh.

ROSLAUNCH: Several ROS nodes can be started both locally and remotely, and the ROS parameter server's parameters can be changed using a tool called roslaunch. Using roslaunch configuration files, which are produced using XML, it is simple to automate a complicated startup and configuration operation into a single command. Roslaunch scripts can launch nodes on specific machines, incorporate other roslaunch scripts, and even restart failed operations [13].

8. Conclusion

We have demonstrated the use of a robotic operating system to control objects carefully and remotely without requiring human assistance. The ROS can be used once more. Communication is straightforward because ROS can understand any language, including C++, Python, and many others. Ros enables developers to turn on their robot remote controls. This system will aid in underwater communication and be helpful in locating underwater cable faults.

Acknowledgements

We are quite appreciative to our professors for allowing us

to work on this subject as well. Our team would like to extend a special thank you to Prof. Vijay Rathod, who has been assisting us with this project since the very beginning. We also want to express our gratitude to our college for giving us the top resources we needed for this project. All in all, we would want to express our gratitude to everyone who contributed to this initiative.

References

[1] Vol. 11, No. 3, Juli 2022 ISSN 0216 – 0544 e-ISSN 2301–6914 119 Development Of Autonomous Underwater Vehicle (Auv) Based On Robotic Operating System For Following Underwater Cable.

[2] N. DeMarinis, S. Tellex, V. P. Kemerlis, G. Konidaris and R. Fonseca, "Scanning the Internet for ROS: A View of Security in Robotics Research," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 8514-8521, doi: 10.1109/ICRA.2019.8794451.

[3] V. Dupourque, "A robot operating system," Proceedings. 1984 IEEE International Conference on Robotics and Automation, 1984, pp. 342-348, doi: 10.1109/ROBOT.1984.1087185.

[4] M. Hellmund, S. Wirges, Ö. Ş. Taş, C. Bandera and N. O. Salscheider, "Robot operating system: A modular software framework for automated driving," 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), 2016, pp. 1564-1570, doi: 10.1109/ITSC.2016.7795766.

[5] Breiling, B. Dieber and P. Schartner, "Secure communication for the robot operating system," 2017 Annual IEEE International Systems Conference (SysCon), 2017, pp. 1-6, doi: 10.1109/SYSCON.2017.7934755.

[6] Kehoe, S. Patil, P. Abbeel and K. Goldberg, "A Survey of Research on Cloud Robotics and Automation," in IEEE Transactions on Automation Science and Engineering, vol. 12, no. 2, pp. 398-409, April 2015, doi: 10.1109/TASE.2014.2376492.

[7] Ramos, Fernando, and Enrique Espinosa. "A self-learning environment based on the PBL approach: An application to the learning process in the field of robotics and manufacturing systems." International Journal of Engineering Education, 19.5, pp. 754-758, 2003.

[8] J. C. Kinsey, D. R. Yoerger, M. V. Jakuba, R. Camilli, C. R. Fisher and C. R. German, "Assessing the Deepwater Horizon oil spill with the sentry autonomous underwater vehicle," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 261-267, doi: 10.1109/IROS.2011.6095008.

[9] P. J. B. Sánchez, M. Papaelias and F. P. G. Márquez, "Autonomous underwater vehicles: Instrumentation and measurements," in IEEE Instrumentation & Measurement Magazine, vol. 23, no. 2, pp. 105-114, April 2020, doi: 10.1109/MIM.2020.9062680.

[10] M. Bradley, M. D. Feezor, H. Singh and F. Yates Sorrell, "Power systems for autonomous underwater vehicles," in IEEE Journal of Oceanic Engineering, vol. 26, no. 4, pp. 526-538, Oct. 2001, doi: 10.1109/48.972089.

[11] Yousuf, A., & Lehman, W., & Mustafa, M. A., & Hayder, M. M. (2015, June), Introducing Kinematics with Robot Operating System (ROS) Paper presented at 2015 ASEE Annual Conference & Exposition, Seattle, Washington.

[12] Lessard, R. A. (1999, June), Embedded Systems Course Focuses on Autonomous Robot Applications Paper presented at 1999 Annual Conference, Charlotte, North Carolina.

[13] Michalson, W., & Looft, F. (2010, June), Designing Robotic Systems: Preparation for An Interdisciplinary Capstone Experience Paper presented at 2010 Annual Conference & Exposition, Louisville, Kentucky.

[14] Barut, S., M. Boneberger, P. Mohammadi and J. J. Steil. Benchmarking Real-Time Capabilities of ROS 2 and OROCOS for Robotics Applications. in 2021 IEEE International Conference on Robotics and Automation (ICRA). 2021. IEEE.