iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** IX **Month of publication:** September 2022

DOI: https://doi.org/10.22214/ijraset.2022.46938

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue IX Sep 2022- Available at www.ijraset.com

Robot Operating Systems (ROS): The Fundamentals of ROS and Its Remarkable Performances in the World of Drones

Koonamneni Janavi[1], Ambarapu Rahitya Teja[2]
[1, 2]*School of Information Technology (IT), Vellore Institute of Technology, Vellore*

Abstract: ***A structured communication mechanism is offered by the open-source operating system ROS. With the integration of its framework and toolkits, and a sophisticated operating system for software development, robot deployment is effective. The development of the unique robotic drone applications and a thorough description of how to program the drone using Ros are covered in this paper's brief discussion of the evolution of the Robot Operating System (Ros). Later in the article, the implementation of using a motion controller to control a drone's motion with basic hand motions is shown. However, there has been a significant increase in the commercial use of drones recently, and many drones have invaded public spaces, potentially invading people's right to privacy. To correct this mistake, we have used the ROS to deploy a drone that does not intrude in forbidden places.***

Keywords: ROS, Unmanned Aerial Vehicles, STAIR, ARM Trust Zone, Parrot AR DRONE 2.0.

I. INTRODUCTION

Prolusion of robot operating system, drones:

A framework for creating robot software is called the Robot Operating System (ROS)

ROS, the "Robot Operating System**",** is a software framework that enables the development of robotic applications – applications that control and interact with robots [6]. In order to promote collaborative robotics software development, ROS was designed from the ground up. ROS is not an operating system but a meta operating system meaning, that it assumes there is an underlying operating system that will assist it in carrying out its tasks [7]. We were unable to categories the Meta Operating system as an operating system, a framework, or a component of a library. But this just partially serves as a framework and offers some of an operating system's features. As a result, This Meta Operating system haven't been classed either of them. For instance, it offers APIs rather than the essential features that an operating system should have. It is a group of software, libraries, and conventions designed to make it easier to programme complicated and reliable robot behavior on a range of robotic systems. Numerous aspects of ROS are intended to make this kind of extensive collaboration easier.

The term **"**drone**"** usually refers to any unpiloted aircraft. Sometimes referred to as "Unmanned Aerial Vehicles" (UAVs) [1] UAVs are a component of an unmanned aircraft system (UAS), which includes adding a ground-based controller and a system of communications with the UAV [2]. In recent years, autonomous drones have begun to transform various application areas as they can fly beyond visual line of sight (BVLOS) [3] while maximizing production, reducing costs and risks, ensuring site safety, security and regulatory compliance [4], and protecting the human workforce in times of a pandemic [5]. They can also be used for consumer-related missions like package delivery, as demonstrated by Amazon Prime Air, and critical deliveries of health supplies.

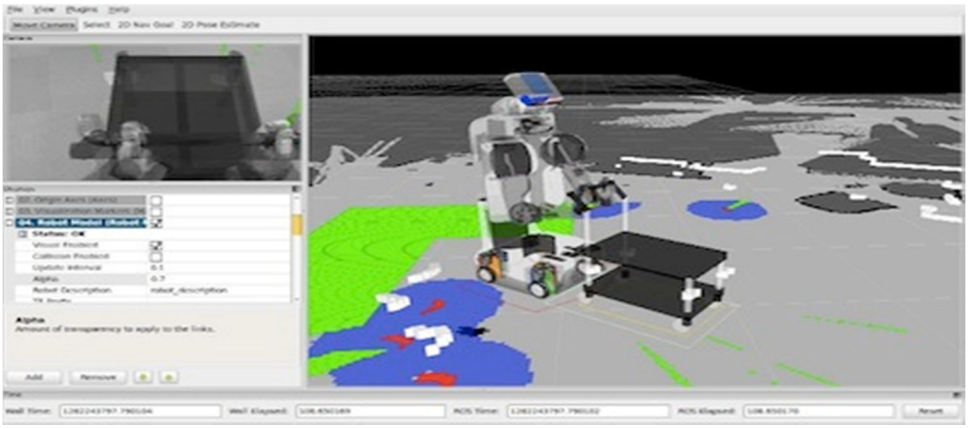
Ros in Drones:

Many groups use the AR. Drone for research purposes, use ROS to control the drone, which is why we are focusing on it here [6].

II. HISTORY OF ROS

Consider a basic "displace a cart from one location to another location" assignment, in which a workspace assistance robot is told to move a cart. The robot must first comprehend the request, whether it is communicated vocally or through another modality like a web interface, email, or even SMS. The robot will next need to launch some sort of planner to organize the hunt for the object, which will probably include moving between different rooms in a building, possibly incorporating elevators and doors [8]. When the robot enters a room, it must examine workstations packed with items of a similar size (because all handheld items are roughly the same size) in order to locate the cart. After that, the robot must go back and deliver the cart to the instructed place.

It can be difficult for a person to build an operation from scratch, but the robot operating system has made it simple by managing such massive activities.



Cart pushing simulation [9]

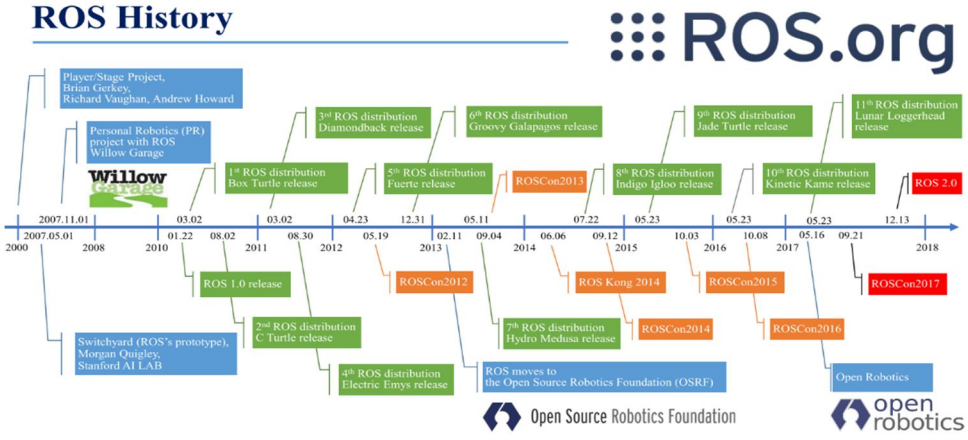
In 2007 Stanford Artificial Intelligence Laboratory developed the Robot Operating System (ROS) in support of the **ST**anford **A**rtificial **I**ntelligence **R**obot STAIR. "ROS is a software framework for robot software development, providing operating system-like functionality on a heterogeneous computer cluster."In 2008 development of ROS continued primarily at Willow Garage, a robotics research institute/incubator in Menlo Park, California, with more than twenty institutions collaborating in a federated development model. Willow Garage shut down in early 2014[10].

THE STANFORD PERIOD

While attending Stanford, Keenan Wyrobek and Eric Berger began working on ROS as a side project to stop robots from constantly having to reinvent the wheel. The two men were concerned about the most prevalent issue in robotics at the time: the length of time required to completely redesign the software infrastructure.Almost no time is spent on clever robots' programmes.

The Stanford Personal Robotics Program was established by Eric and Keenan in 2006 with the goal of developing the framework that would enable processes to communicate with one another as well as some tools to aid in the creation of code on top of that framework. All of that infrastructure was intended to be used to generate code for the Personal Robot, a robot they will also build and utilize as a testbed and an example for others [11]. They would create ten of those robots and provide them to academic institutions so that they could create software using their framework.

At some point in 2008, Scott Hassan, an investor and the creator of Willow Garage, a research facility with an emphasis on robotics technologies, met with Keenan and Eric. Their idea piqued Scott's interest to the point where he chose to fund it and collaborate with them to launch a Personal Robotics Program inside Willow Garage. Willow Garage began developing the PR2 robot as a follow-up to the PR1, and ROS as the software to run it. Groups from more than twenty institutions made contributions to ROS, both the core software and the growing number of packages which worked with ROS to form a greater software ecosystem [12]. The PR2 robot and the Robot Operating System were both born at the same time. In fact, the ROS project grew so crucial that Willow Garage abandoned all its previous initiatives in favor of focusing solely on the growth and adoption of ROS.



2011 was a banner year for ROS with the launch of ROS Answers, a Q/A forum for ROS users, on 15 February;[13]the introduction of the highly successful TurtleBot robot kit on 18 April;[14] and the total number of ROS repositories passing 100 on 5 May.[15] Willow Garage began 2012 by creating the Open Source Robotics Foundation (OSRF) [16] in April.

The statement in August that Willow Garage would be absorbed by its founders, Suitable Technologies, was foreshadowed by OSRF becoming the major software maintainers for ROS in February 2013. Up until ROS Groovy, ROS had launched seven major versions and had a global user base.

Approximately six years of ROS development took place at Willow Garage before Willow closed its doors in 2014. Numerous improvements to the project were made at that time. This promotion throughout the Willow era is what really increased its appeal.

Tens of thousands of users working on everything from small hobby projects to massive industrial automation systems currently make up the ROS ecosystem.

Drones' progression from inclusion to embedded ROS drones:

These crafts can carry out an impressive range of tasks, ranging from military operations to package delivery. Drones can be as large as an aircraft or as small as the palm of your hand.

III. NOMENCLATURE OF ROS

A. Installation

1) Starting with the prerequisites, Ubuntu 12.04 LTS should be operating and configured to use the Ubuntu repositories.

2) Secondly, set up your sources list. In order for the programme from packages.ros.org to run on the computer and then continue setting up your keys so that ROS can begin the software installation.

3) Actual ROS installation. Type the below commands in the terminal window (this may take up to a few minutes)

sudo apt-get update

sudo apt-get install ros-hydro-desktop-full.

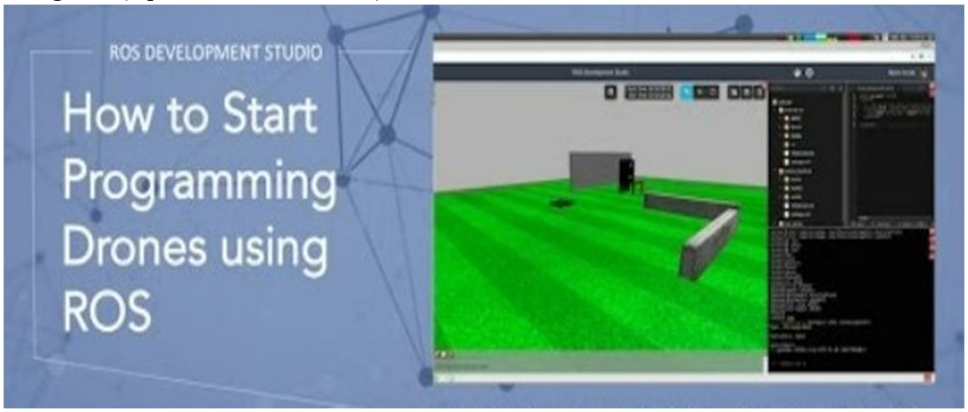
4) Enable the rosdep:

sudo rosdep init

rosdep update

As it is required to run the core components in ROS at last set up the environment. The installation has finished its process.

Programming drones using Ros (a practical introduction):



That the very first step is to start a project in the ROS development studio. To build the project, we use tools like the shell, IDE, Jupyter, etc. Initiate the project's creation, and then run the simulation. So, to launch, navigate to Simulations->Select Launch File->main.launch. Enter the command listed below to see if the topics are being covered by the drone in the shell[19].

$ rostopic list

/* rostopic list returns a list of ROS topics from the ROS master. */[20]

In the default Jupyter notebook. Go to Tools > Jupyter Notebook > Default.ipynb to open it. Using the Python script found in default.ipynb is an additional option.

Use the next command to make the drone take off now.

$ rostopic pub /drone/takeoff std_msgs/Empty "{}"

Use the following command to land the drone if necessary.

$ rostopic pub /drone/land std_msgs/Empty "{}"

B. Program With Drones

For the position control function in the drones use the below command
rostopic info /drone/posctrl
You'll see the output like this.
Type:std_msgs/Bool
Publishers:
*/my_node(http://ip-172-31-35-31:45972/)
Subscribers:
* /gazebo (http://10.8.0.1:44685/)
As the present topic is using the boolean data.In order to get more info about the bool data type the command.
rosmsg show std_msgs/Bool
and got the output
bool data
Let's try to convey a message regarding this bool message. We first put up a monitor before publishing the topic.
rostopic echo /drone/posctrl
Then paste the following code into a Jupyter notebook and run it.

```
from std_msgs.msg import Bool
var_bool = Bool()
pub_posctrl = rospy.Publisher('/drone/posctrl',Bool,queue_size=1)
var_bool.data =True
pub_posctrl.publish(var_bool)
```

You should see
data: True
This indicates that the message was published properly. The drone's position control feature was successfully enabled. The drone can also be moved by posting a Twist message to the /cmd vel topic. This script serves as an example.

```
from geometry_msgs.msg import Twist
var_twist = Twist()
pub_position = rospy.Publisher('/cmd_vel',Twist, queue_size=1)
var_twist.linear.x = 1
var_twist.linear.y = 1
var_twist.linear.z = 2
pub_position.publish(var_twist)
```

IV. REAL-TIME DRONE OPERATIONS BASED ON ROS

A. Controlling Drones in Confined Areas

Most of the time, drones used for business and for work are equipped with sensors. hence limiting the use of commercial drones in public areas. We argue for the use of drones in restricted places, or zones where a host can establish its privacy standards. Then, we argue that the idea of a framework for enforcing drone policy based on information flow control and construct a prototype framework built on top of the Robot Operating System (ROS).

REGULATING DRONE BEHAVIOUR:

The vision discussed above can be realized on drones equipped with the ARM Trust Zone. The methods used by the host to determine whether a drone is in policy compliance also depend on the software stack running on the drone. We describe a concrete implementation for drones running the Robot Operating System (ROS) [17]. We make the case for dynamic information-flow control (IFC) as the core policy enforcement mechanism within the drone's software stack and describe how ROS lends itself well to IFC.

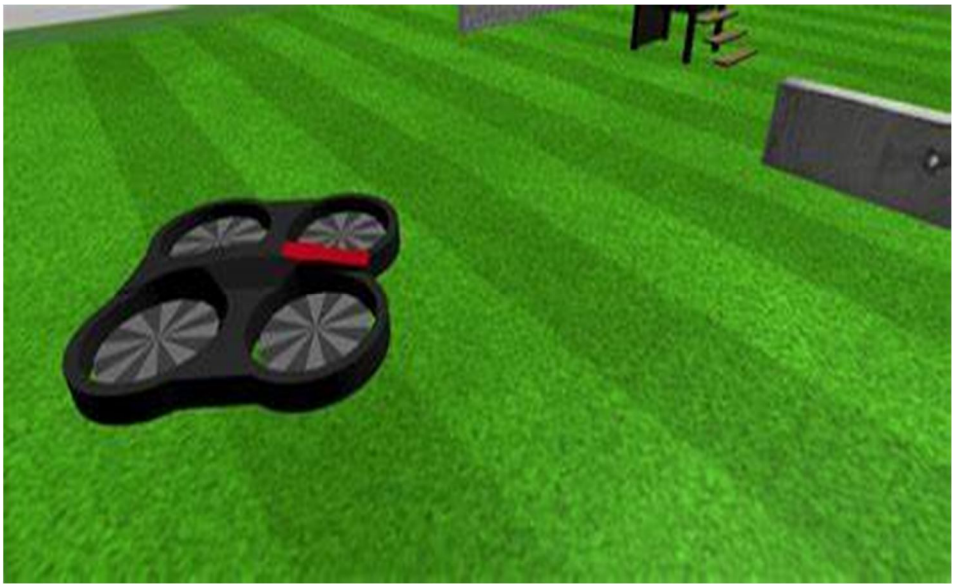
We built our IFC prototype on ROS. ROS is a set of middleware libraries that run atop Linux and provides a convenient platform to author robotics applications. ROS is used by several drone manufacturers, several popular software packages for drones are also built atop ROS. For example, MAVROS [2], which is a communication driver for autopilots such as PX4 and ArduPilot that use the MAVLink communication protocol (a widely used protocol for communicating with small drones), are built atop ROS.

By matching publishers and subscribers depending on the topics they initially promoted, ROS libraries jumpstart communication. By establishing an IPC channel between the relevant publishers and subscribers, it does this.

The Linux processes that implement the publishers and subscribers then interact directly with one another. The ROS architecture is ideally suited for IFC. We can directly use the topics that have already been assigned to messages as IFC labels. The themes of the data items that subscribers can consume are simply determined by the IFC policy. By registering the application as a listener for the topic of the input data object and a publisher for the output data object, trusted apps can conduct label transitions on data objects. The trusted blur-filter application, for instance, can sign up as a subscriber to the camera topic and publish objects with the topic blurred-img. This application is used to enforce the Blur-Exported-Images specification. By adding just under 200 lines of Python code to the ROS source, we were able to construct a basic IFC mechanism on top of ROS and utilize it to enforce the sample regulations covered above. This code is mostly concerned with analyzing the host's policy and limiting the topics that applications can publish and subscribe to. In it publish-subscribe architecture, ROS' topic-matching algorithms then automatically handle enforcement.

B. Drone Gesture Control with a Motion Controller

Drones have been useful in various fields, but their use with ROS gives them unique functionalities that make them operate effectively. In this example,we manage the motion of the drone with the use of human gestures. For this implementation, we used the Parrot AR DRONE 2.0 and the Leap as the motion controller.



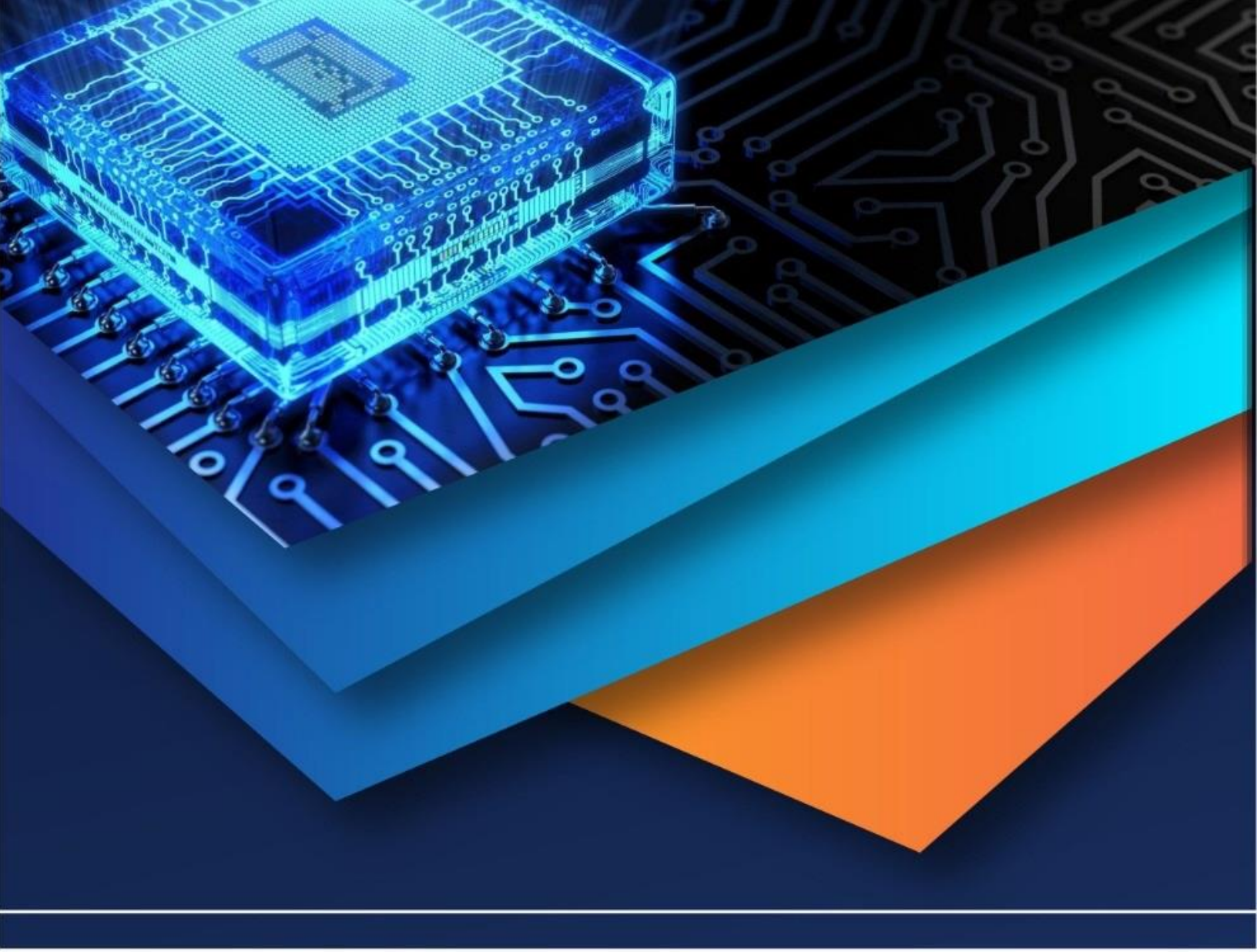
A quadrotor with an onboard Wi-Fi system is available off the shelf and is called the Parrot AR Drone. The AR drone is connected to the Wi-fi, and the jump is connected to the ground station through a USB connection [21].The ground station receives the hand movements from the LEAP Motion Controller after it detects them.Additionally, the ROS (Robot Operating System) is run on Linux by the ground station for the implementation. And Python is used to analyze the hand movements collected by the LEAP and transmit them to control the motion of the AD Drone. It also transmits hand gestures to the AR Drone.

V. CONCLUSIONS

We don't assert that the ROS framework is the best for any robotics software. As a matter of fact, we don't think such a framework exists because robotics is too vast to have just one solution. In order to address a particular set of issues that arise when creating large-scale robots, ROS was created. We believe that because of its open-ended architecture, others will be able to expand upon it and create robot software systems that can be applied to a range of hardware platforms and runtime needs. Many organizations that employ the AR drone for scientific purposes often use ROS to operate the drone. In the study, we described how to programme the ros for drones and described drones which use Robot Operating System. I.e., restricting drone access to public spaces and directing drone motion with hand gestures. To increase the viability, drones were implemented using ROS. Because of the framework, developing robotic applications was simple.

REFERENCES

[1] https://builtin.com/drones
[2] Hu, J.; Niu, H.; Carrasco, J.; Lennox, B.; Arvin, F., "Fault-tolerant cooperative navigation of networked UAV swarms for forest fire monitoring" Aerospace Science and Technology, 2022.
[3] "How Autonomous Drone Flights Will Go Beyond Line of Sight". Nanalyze. 31 December 2019.
[4] McNabb, Miriam (28 February 2020). "Drones Get the Lights Back on Faster for Florida Communities". DRONELIFE.
[5] Peck, Abe (19 March 2020). "Coronavirus Spurs Percepto's Drone-in-a-Box Surveillance Solution". Inside Unmanned Systems.
[6] https://robohub.org/up-and-flying-with-the-ar-drone-and-ros-getting-started/
[7] https://www.geeksforgeeks.org/introduction-to-ros-robot-operating-system/https://ai.stanford.edu/~mquigley/papers/aaai2007_robotics_work shop_stair.pdf
[8] https://books.google.co.in/books?hl=en&lr=&id=Hnz5CgAAQBAJ&oi=fnd&pg=PR2&dq=history+of+robot+operating+system+&ots=-6ujKXZAO2&sig=qDTvWfl8pEr7VwF0x--BlGGvdO8&redir_esc=y#v=onepage&q=history%20of%20robot%20operating%20system&f=false
[9] By Open Source Robotics Foundation - http://www.ros.org/wp-content/uploads/2013/12/cart_pushing_rviz_holonomic.jpg, CC BY 3.0, https:/ /commons .wikimedia.org/w/index.php?curid=39662126
[10] https://historyofinformation.com/detail.php?id=3661
[11] https://svrobo.org/a-history-of-ros-robot-operating-system/
[12] https://en.wikipedia.org/wiki/Robot_Operating_System
[13] *"Announcing ROS Answers – ROS robotics news". ROS.org. Open Robotics. Retrieved 12 December 2017.*
[14] ^ *"ROS on the Move: TurtleBots available for preorder". Willow Garage. Retrieved 12 December 2017.*
[15] ^ *"100 Repositories – ROS robotics news". ROS.org. Open Robotics. Retrieved 12 December 2017.*
[16] ^ *"Willow Garage Spins Out OSRF".*
[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng. ROS: An open-source Robot Operating System. In ICRA workshop on open-source software, 2009
[18] MAVROS – MAVLink extendable communication node for ROS with proxy for ground control station. http://wiki.ros.org/mavros.
[19] https://www.theconstructsim.com/start-programming-drones-using-ros-video-answer/
[20] https://www.mathworks.com/help/ros/ref/rostopic.html
[21] https://www.researchgate.net/profile/AyanavaSarkar/publicatio n/301800528_Gesture_control_of_drone_using_a _motion_controller /links/59e3 10480f7 e9 b97fbeab1d4/Gesture-control-of-drone-using-a-motion-controller.pdf



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY