ABSTRACT

In this paper, we demonstrated a service robot navigation system based on the Message Queuing Telemetry Transport (MQTT) protocol communication system that updates the real-time robot states for multiple users. The proposed office assistant robot (OABot) consists of a navigable structure, a mobile app, and a central control workstation and these three components intercommunicate via a wireless network. The voice-recognition mobile app is used to interact with users with voice commands; these voice commands are processed inside the workstation and actions are assigned to the moving robot accordingly. The robot can navigate inside the room using real-time maps while localizing itself in the environment. In addition, the robot is equipped with a digital camera to identify people in predefined locations in the room. The WiFi communication system is provided with RESTful and Mosquitto servers for better human-robot communication. Hence, multiple users are notified about the robot status through updates on the real-time states via the MQTT protocol. The developed system successfully navigates to the instructed destinations and identifies the target person with an average accuracy of 96%. Most importantly, in an isolated indoor environment with social distancing restrictions to perform, the proposed system is essentially useful for contactless delivery.

INDEX TERMS

Human-robot interaction; MQTT protocol; RESTful; robot navigation
robots and computer systems for their control, sensory feedback, and information processing. All robots have a particular mechanical construction, a frame, a form, or a shape designed to accomplish a particular task [5]. The mechanical aspect is mostly the creator’s solution to complete the assigned task and deal with the physics of the environment around its form followed by the function [6].

A robotic program helps the robot decide to perform a particular action from a set of possible actions and when to execute it. There are three different types of robotic programs: remote control [7], [8], [9], artificial intelligence [10], [11], and hybrid [12], [13], [14]. A robot with remote control programming has a preexisting set of commands that it will only perform if it receives a signal from a control source, typically a human being with remote control. Robots that use artificial intelligence interact with their environment on their own without a control source and can determine reactions to objects and problems they encounter using their preexisting programming. A hybrid is a form of programming that incorporates both of the above functions [6].

The contributions of this paper are as follows.

- Implementation of a cost-effective mobile navigable robot that can share parcels and documents among users in an isolated indoor environment.
- Synchronization, robot-user communication, and user locations are managed in a single Android application. Users can receive real-time status updates of the robot. With notification system, users can save time without waiting for robot because users are notified after completing robot task through the app. Then user can try to give commands to the robot.
- The robot system manages multiple user requests to the robot using the Message Queuing Telemetry Transport (MQTT) protocol while updating the real-time robot status to every user through the Android mobile application. Human robot communication is the most important part of the service robot system. MQTT protocol handles not only single user robot communication but also multi-user robot communication at the same time, improves the user-friendliness of the human robot communication system.
- The robot can correctly identify users using a face recognition system, and users can give their commands to the robot using voice commands and virtual key pressing.

The rest of the paper is organized as follows. Section II covers a review of recent related research. In Section III, the implementation of robot-human communication using the MQTT and RESTful is presented. Subsequently, Section IV includes the experimentation setup of the human-robot communication system. Then, Section V deals with the results and discussion of the communication and the overall navigation process. Finally, Section VI concludes the paper.

II. LITERATURE REVIEW

In support of the proposed work, we first compared the most recent existing mobile robot systems to our robot in order to validate the choice of the equipment. SLAM algorithm was used for navigation in our robot system. We include a review of SLAM methods that are commonly adopted by many studies. Furthermore, we reviewed the communication technologies related to our work that compares our multi-user connectivity against traditional peer-peer connectivity.

A. COMPARISON OF LAB-SCALE MOBILE ROBOTS

We compared our robot with MobileCharger [15], Autonomous wheelchair [16], OmniNav [17], Zytebot [18] and four other recent lab-scale mobile robots: Semi-autonomous Mobile Robot for Environmental Surfaces Disinfections Against SARS-CoV-2 [19], Autonomous mobile robot for visual inspection of MEP provisions [20], Navigation of Mobile Robot Through Mapping Using Orbeec Astra Camera and the Robotic Operating System (ROS) in an Indoor Environment [21], BIT-NAZA based tracking system [55].

Of all ten robots, eight, including ours, have been built on ROS. Also, five robots used ROS-based platforms as their robotic platform. Three of the five robot systems used the Turtlebot platform. LiDAR and Real sense cameras are popular as vision sensors. However, we used the Xtion Pro camera instead of the common ones because LiDAR is expensive, and the Turtlebot platform is incompatible with Real sense. The robots described in [19] and [55] are provided with trajectory information for navigation in semi-auto mode.

On the contrary, all other systems, including ours used an environment map to navigate.

B. COMPARISON OF HUMAN-ROBOT COMMUNICATION PROTOCOLS

Networked robots maintain machine-to-machine communication using communication protocols like proactive routing or ad hoc routing which incur high computation and memory requirements and high latency. Complex real-life problems that require real-time execution demand sophisticated data analysis and computational capabilities that are challenging for networked robots to handle [22]. Robot Web Tools is an efficient messaging technique for cloud robotics [23]. It uses the rosbridge package in ROS for web based robot-user communication. It enables cloud robot communication through more efficient methods of transporting high-bandwidth topics such as image streams, kinematic transforms, and point clouds.

MQTT has been used for several robotic applications in the recent literature. Kazala et al. described a research that uses MQTT to exchange messages between a network of multiple robots. They mainly used the MQTT protocol for control purposes and to gather data from robot sensors. The authors stated that this method reduces the energy consumption and computing power of robot nodes [24]. Another study by Atmoko and Yang used the MQTT protocol for online mon-
itoring and control of an industrial arm robot that provides low-latency data transmission [25].

One of the main concerns of wireless data communication is security. Mukhandi et al. conducted a research to achieve this goal in robots using MQTT and ROS. They were able to secure the network communications of robots by providing authentication and data encryption, therefore preventing man-in-the-middle and hijacking attacks. Specifically, remote clients connect to the MQTT server and authenticate using valid digital certificates. After authentication, clients receive authorization from the MQTT server to establish a secure communication channel with robots [26].

As suggested by above recent research, we used MQTT protocol to achieve multi-user connectivity with the robot. The secure message passing is another reason for the selection of the protocol.

C. MQTT PROTOCOL

In our work, the MQTT protocol and RESTful web service were used to communicate inside the OABot system.

According to ISO/IEC 20922, MQTT [27] is the ideal communication and connectivity protocol in Machine-to-Machine (M2M) communications within the Internet of Things (IoT). It works on top of the TCP/IP protocol stack, which was designed as an extremely lightweight broker-based publish/subscribe messaging protocol for small code footprints (e.g., 8-bit, 256KB ram controllers). MQTT is a simple and easy to implement, lightweight, open-source protocol. It has low bandwidth and power, high-cost connections, and high latency. It has a 2 bytes fixed header size, therefore it incurs in lower message overheads compared to other messaging protocols such as CoAP, AMQP and HTTP.

Moreover, MQTT has built-in security features utilizing SSL/TLS that can be implemented to offer standard security and privacy of all data. Protocols such as AMQP and HTTP also offer security with the use of SSL/TLS, IPSec, and SASL. However, studies show that they need higher bandwidth and resource requirements, and are less reliable. Although CoAP requires less bandwidth and resources than MQTT, it does not explicitly provide Quality of Service (QoS) and only supports unreliable UDP communication [28]. There are several widely used open-source software that employ the MQTT protocol, such as Mosquitto [29], Mosca [30], and Paho [31].

In the hub-and-spoke model of Message Oriented Middleware messaging, the server forwards messages from sensor devices to monitor devices. In such architecture, a sensor device whose main task is to continuously produce and send data to the server is defined as the publisher. The central server, an MQTT broker, collects messages from publishers and examines to whom the message needs to be sent. On the other hand, every device that had previously registered its interests with a server will continue to receive messages until the subscription is canceled. Using this architecture, publishers and subscribers do not need to know each other, which is one of the significant advantages of this protocol [32].

D. RESTFUL WEB SERVER

REpresentational State Transfer (REST) defines a set of architectural principles by which web services can be designed that focus on a system’s resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. If measured by the number of web services that use it, REST has emerged in the last few years alone as a predominant web service design model. REST has had such a massive impact on the web that it has mostly displaced SOAP and WSDL-based interface design because it is a considerably simpler style to use. One of the key characteristics of a RESTful web service is the explicit use of HTTP methods in a way that follows the protocol as defined by RFC 2616. HTTP GET, for instance, is defined as a data-producing method that is intended to be used by a client application to retrieve a resource, to fetch data from a web server, or to execute a query with the expectation that the web server will look for and respond with a set of matching resources [33].

REST asks developers to use HTTP methods explicitly and in a way that is consistent with the protocol definition. This basic REST design principle establishes a one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods. According to this mapping [33]:

- To create a resource on the server, use POST
- To retrieve a resource, use GET
- To change the state of a resource or update it, use PUT
- To remove or delete a resource, use DELETE
E. SLAM ALGORITHM

SLAM collect visible data such as images from a camera and/or non-visible data like (SONAR, RADAR and LiDAR) with Inertial Measurement Unit (IMU) is used to collect positional data. Together, these sensors collect data and build a picture, and the SLAM algorithm enables the best estimate of the location/position within the surrounding environment. The algorithm maintains a relative positioning between the device and the surrounding features such as floors, walls, pillars and furniture. Furthermore, SLAM iteratively improves the estimated position with new information, and with a higher iteration process will increase the positional accuracy [51]. The SLAM occupies two tasks: front-end for data collection and back-end for data processing. Visual SLAM and LiDAR SLAM define the front-end data collection of SLAM.

Visual SLAM, also known as vSLAM, employs a camera to collect images from the surrounding. It can use simple cameras, compound eye cameras and RGB-D cameras. Simple cameras include wide-angle, 360 degrees panoramic, and fish-eye cameras, whereas stereo and multi-cameras are used as compound eye cameras.

RGB-D cameras can capture more dimensions using depth cameras and time-of-flight (ToF) cameras. A ToF camera is a range imaging camera system. It employs ToF techniques to resolve the distance between the camera and the subject for each image point by measuring the round trip time of an artificial light signal provided by an LED or a laser.

Because Visual SLAM uses relatively inexpensive cameras, its implementation is generally cost-effective. In addition, cameras can detect landmarks that are previously measured positions as they provide a large volume of information. Combined with graph-based optimization, landmark detection achieve flexibility in SLAM implementation [52].

Since LiDAR SLAM uses a laser sensor, its implementation is more precise and accurate than Visual SLAM. Hence, it can capture data at a high rate with more precision, allowing LiDAR sensors to be used in high-speed applications such as drones and self-driving cars, which are moving vehicles.

LiDAR sensors result in point cloud data, which can be 2D (x, y) or 3D (x, y, z) positional information. This point cloud can measure high-precision distance and effectively works with map building of SLAM. Commonly, sequential movement estimations are calculated by matching the point clouds. The vehicle is localized by using the calculated traveled distance (movement). Normal distributions transform (NDT) and iterative closest point (ICP) algorithms are used for LiDAR point cloud matching. 2D point clouds are represented as a grid, and 3D point clouds are represented as a voxel map [53]. Examples of recent research using LiDAR SLAM can be found in [54], [56].

III. OVERALL ROBOT SYSTEM ARCHITECTURE

The overall robot system architecture is composed of a navigable robot structure, an Android mobile application, and a controller workstation.

First, the movable robotic structure was built using the iRobot Create 2 programmable robot [34] as the base. A rigid structure was set up on top of it to carry documents/parcels, a mini laptop, and a camera. This structure is shown in Figure 1. The robot structure is described in relation to a previous study conducted by the respective authors [35].

Our primary software development environment was Ubuntu 14.04 LTE. We selected the Turtlebot package available in the ROS to customize and train the OABot.

In addition, an Android application was developed for user input, enabling control of the system with any Android device. Every office member can install this Android application and give commands to the robot to deliver their documents. Users can give their commands using voice and virtual key pressing in the Android application.

In addition, the OABot, the central control workstation, and the Android application communicate over a private WiFi network. Finally, a face recognition feature of the OABot was developed. Therefore, the OABot can identify the correct recipient using this face recognition feature, and it can deliver packages safely. The system was previously tested for its behavior under various room illumination conditions and has been shown to operate effectively in dark illumination conditions (15lx –20lx) [36]. Consequently, the general overview of the system is shown in Figure 2.

A. SOFTWARE DEVELOPMENT ENVIRONMENT

ROS is a set of software libraries and tools that help build robot applications. ROS has what is needed for any robotics project, from drivers to state-of-the-art algorithms and powerful developer tools. Furthermore, it is completely open source [37]. ROS is becoming the standard in robotics programming, specially in the service robot sector. Initially, ROS started at universities, but quickly spread to the business world. Large companies as well as start-ups are increasingly basing their businesses on ROS [38].

We have installed ROS Indigo Igloo on top of the OS. Turtlebot, which is an existing ROS package, was selected as the robot kit. Among its numerous functionalities, we used bring-up, gmapping, keyboard teleop, Rviz view, and amcl nodes in our study. As in human navigation, the robot has to be aware of the working environment to avoid collisions and know the available paths in the room. For that purpose, the robot uses a map to navigate autonomously in the environment using a path that was first fed into the robot using a computer program.

1) Turtlebot bring-up

Turtlebot bringup [39] accepts roslaunch scripts to start the Turtlebot base functionality. It works using two launching files, namely minimal launch and appmanager launch. The minimal.launch file is the base launch file for Turtlebot that starts the basic nodes such as kobuki_node or create_node. App_manager.launch on the other hand, starts the Turtlebot app managers and loads the Turtlebot app list.
2) Gmapping

In our work, the Gmapping technique is used to map the environment. Gmapping is a localization technique that runs in an unknown environment to perform simultaneous localization and mapping. It uses the Rao-Blackwellized Particle Filter (RBPF) and receives data from both the sensor and the robot pose to generate a 2D grid map of the environment without IMU information [40]. In Gmapping, the robot constantly updates the pose on each processed particle by estimating odometry. During the execution of mapping, as the first scan is received, it is directly registered on the map. Subsequently, the registration occurs only if the linear distance or angular distance traversed by the robot exceeds specific thresholds. Once the scan match is received, correction of the pose estimation is performed. It is done on the map of each particle for each laser scan. The purpose is to generate a map of the environment [41], [42].
3) Keyboard Teleop

teleop_twist_keyboard is the generic Keyboard Teleop for ROS. It facilitates the smooth control of a robot using the keyboard of a computer. Users can start, stop, or move the robot in the environment using the keyboard. Also, it has the ability to control the linear and angular speeds of the robot while moving [43].

4) Rviz view

The Rviz [44] tool acts as a visualization tool for ROS-based applications. It captures the information from the laser scanners and replays the obtained data in the form of visuals. In the present work, Rviz is used to visualize the generated environment map.

5) amcl

The amcl node creates pose estimates using laser scans, a laser-based map, and transform messages. On startup, the node initializes its particle filter according to the given parameters. The default parameters are set to a moderate-sized particle cloud centered around the origin (0, 0, 0) if no user parameters are given [45].

B. COMMUNICATION NETWORK

The three main parts of the OABot system (OABot, central control workstation, and the Android device) communicate through a WiFi connection. We used a D-Link DIR-605L powered by Realtek RTL8196E (IEEE 802.11b/g/n standards and the maximum speed is 300 Mbps) router and kept it inside the room. OABot and the central control workstation are communicating through the ROS using their IP addresses, which connect to the Wi-Fi router. Android devices are communicating with both OABot and central control workstation via the RESTful web service and the MQTT protocol. The mobile phones, central control workstation, and the robot used the same connection to maintain the communication between the units.

C. ECLIPSE MOSQUITTO SERVER

The Mosquitto Server by Eclipse [46] was used as the MQTT message broker. It is an open source software. It is lightweight and is suitable for use on a range of devices, from low-power single-board computers to full servers. We used the implementation of the MQTT version v3.1.1 protocol in our study.

D. ANDROID APPLICATION OVERVIEW

The users can provide commands using voice or virtual keys in the Android application [35]. The application was implemented using Android Studio [47]. It provides graphical user interface to get the name of the expected recipient of the document. We have implemented a virtual button to allow the user to speak up and used the Google Voice Input function [48] to receive the voice command. It is recognized using Google Voice Recognition. In Addition we have included name buttons in the application with a previously assigned user name. Users can select the desired recipient from the list of buttons. The string containing the name is sent to the central control workstation. The workstation then searches for the recipient’s coordinates from a predefined list. Those coordinates and the path to the goal obtained through the map are sent to the robot. Finally, the robot moves accordingly.

Furthermore, users can know the statuses of the OABot through the Android application. It is shown as a message on top of the Android application interface, as illustrated in Figure 3. The interface consists of a multi input facility including voice input functionality at the bottom of the screen. In addition, buttons are added to locate the robot and all registered users on the system.

E. COMMUNICATION PROCESS OF THE SYSTEM

ROS requires a bidirectional network between the robot and the central control workstation attached to the private WiFi network. The robot network and the central control workstation can be configured with their IP addresses. Android devices of users are required to connect to the same network to achieve communication. When a command is received from the Android application, it passes the command to the central control workstation, acting as a REST client. The OABot passes robot statuses to the Android application, acting as an MQTT client, and the Android application acts as another MQTT client. There, the MQTT server was hosted on the central control workstation. The ID number of the Android device (the unique number of the device) was used to identify the correct user to send the message.

F. USER IDENTIFICATION

In particular, human face detection and face recognition features were added to the OABot for security purposes. For example, OABot can deliver secret documents by identifying the correct recipient. For this RGB web camera, a Light Emitting Diode (LED) panel was mounted on the top of the OABot. The purpose of LED light is to enable facial recognition and identification in a dark lighting environment. A software was developed using the OpenCV2 computer vision and machine learning software library [49]. createLBPFaceRecognizer() feature was used in particular while the recognizer was trained using 100 images per user. Furthermore, a certain user is identified from real-time images using CascadeClassifier() in OpenCV2.

G. FUNCTIONALITY OF THE OABOT SYSTEM

If a user intends to send a letter or a small parcel to another user in the office environment using OABot, then he/she needs to give the command through the Android application, which is installed on his/her mobile phone. First, the OABot must be present at the sender’s place. For this, the Android application sends the location coordinates of the user to the central control workstation. After receiving this coordinate, the central control workstation passes the coordinate to the OABot as an ROS message. Then, the OABot arrives at the...
user. Next, the OABot waits 2 minutes until he places the letter or parcel on the tray of the robot stack. During the 2 minutes, the sender has to provide the recipient location command; otherwise, the OABot returns to the default OABot location. In this case, the user has to provide the receiver location again using the Android application. Then, the OABot will move to the recipient location. After reaching the target location for the task, a Python script is executed to identify the recipient. If the correct recipient is available on the seat, then the OABot waits 2 minutes until the recipient takes the letter or parcel. Here, we assumed that if the recipient is on the seat, they must take the parcel from OABot. After finishing the task, the OABot moves to its default location (default OABot location) and waits at this place until the next task. If the recipient is not available at their seat, the OABot returns to the sender’s place and waits 2 minutes until the sender retakes the parcel. Here again, we assumed that if the recipient is in the seat, they must take the parcel from OABot. After 2 minutes, the OABot again returns to the default place.

Furthermore, the OABot cannot be used by another user when it is already occupied. The second user can see the notification like “OABot is busy now.” in the Android application on their mobile phone. After finishing the first task by the OABot, the second user can see the notification like “OABot is in the default place.” Then, the second user can use the OABot for their task. Figure 4 shows a flow chart for the functionality of the OABot system.

IV. EXPERIMENTAL PROCEDURES

The OABot system was tested three times during creation. First, it was tested on the laboratory scale two times, and finally, in a physically created office environment. This section includes an experiment to measure the accuracy in achieving the objectives of the OABot overall system, which was conducted in an office environment created by the developers.

The system was tested using four users located to the sides of the created office environment of dimension $10.31m \times 5.74m \times 3.30m$ as illustrated in Figure 5. The room was air-conditioned and remained closed during the experiments. Each user was named using a number between 1 and 5 and assigned coordinates of the user location on the built map of the created office environment. The locations of the users and the default of the OABot in the created office environment are shown in Figure 6. The local map and global map of the created office environment are shown in Figure 7 and the location coordinates of the users are included in Table 1. In addition, we used photographs of office members instead of real users to test our system. Then, these color photographs were hung at users’ locations at a height of 1m.

First, the OABot is placed at the default place in the created office environment. The user is given a smartphone with

| No. | Users       | Coordinate (X) | Coordinate (Y) |
|-----|-------------|----------------|----------------|
| 1   | User number 1 | -3.750         | -6.370         |
| 2   | User number 2 | 0.663          | -4.730         |
| 3   | User number 3 | -3.340         | -2.280         |
| 4   | User number 4 | -2.020         | 1.190          |
| 5   | User number 5 | 0.237          | -1.4           |
| 6   | Default place of the OABot | 0.0          | 0.0           |

TABLE 1. Predefined coordinates of users’ locations and the coordinate of the OABot’s location.
our Android application installed. Afterwards, the user calls the OABot through the application. Then the location of the OABot moves to the user and identifies the user with the face recognition feature using the user’s face photograph. Secondly, he/she has to provide a recipient through the Android application to deliver the document. Again, the OABot moves to the recipient location and identifies the recipient with the face recognition feature. If the recipient can be identified, then the task of the OABot is over, and it moves to its default place. If the recipient cannot be identified, the OABot returns to the sender’s place and waits 2 minutes at the sender’s location. After that, the OABot again returns to the default place.

In our experiments, a trial is considered successful if the robot completes the task and arrives at a given location in the room. Each trial starts with the robot in its default position, reaches the sender, arrives at the recipient, and returns to the start position. However, the robot does not exactly follow the expected path 100%, and there may be some errors, but we consider it is successful if the robot accomplishes the task between the start and the endpoint. We convey the results based on the counts of such trials. Figure 10 and Figure 11 demonstrate a sample paths for with obstacle and without obstacles scenarios.

We monitored the success rate of completing this process...
for each user by randomly experimenting 200 times per user. This experiment was labeled as experiment No. 1. We then added two foreign obstacles, which were not included in the built map, into the created office environment. The locations of the foreign obstacles are shown in Figure 8. After the OABot has captured the foreign barrier using the 3D vision sensor while navigating the office environment, the OABot system can update the global map for foreign obstacles. The

| User           | No. of successes scenarios per 200 test cases | Success rate (%) |
|----------------|----------------------------------------------|------------------|
| User Number 1  | 194                                          | 97.0             |
| User Number 2  | 196                                          | 98.0             |
| User Number 3  | 192                                          | 96.0             |
| User Number 4  | 194                                          | 97.0             |
| User Number 5  | 196                                          | 98.0             |

TABLE 2. The collected data regarding the experiment No. 1.
updated global map of foreign obstacles is shown in Figure 9. Finally, the experiment was repeated to monitor the success rate of the OABot. It was labeled as experiment No. 2. The experimental setup can be further confirmed through the submitted video.

V. RESULTS AND DISCUSSION

The experiment was conducted to obtain the functional accuracy of the overall OABot system. The accuracy was obtained in a created office environment, both with and without foreign obstacles (experiment No. 1 & experiment No. 2). The experiment data regarding the experiments No. 1 and No. 2 are shown in Table 2 and Table 3, respectively.

The accuracy of the OABot system exceeds 96% on average. In particular, 97.2% accuracy for completing the delivery process was achieved and the OABot failed only 28 times in a total of 1000 test cases during experiment No. 1. Further, the OABot has completed the delivery process with a 96.9% accuracy in 1000 test cases in experiment No. 2. Here, the system failed only 31 times. A line graph with a descriptive result comparison for both experiments is shown in Figure 12.

The failures occurred due to several reasons, such as robot localization, path identification, and user face recognition not functioning correctly. However, the OABot did not fail in overcoming foreign static obstacles. Therefore, The OABot was faced no challenges in autonomously driving by avoiding static foreign obstacles. Overall, the achievement of goal of the OABot system according to the user’s commands can be accomplished not only in the absence of foreign obstacles but also when there are static foreign obstacles in the office environment.

VI. CONCLUSION

Here, the experiment was performed with and without foreign obstacles in the office environment. In these experiments, we have considered how the OABot responds to user commands, how the OABot completes the target location by overcoming obstacles, and the accuracy of face recognition.

| User Number | No. of successes scenarios per 200 test cases | Success rate (%) |
|-------------|---------------------------------------------|------------------|
| User Number 1 | 191                                          | 95.5             |
| User Number 2 | 195                                          | 97.5             |
| User Number 3 | 193                                          | 96.5             |
| User Number 4 | 196                                          | 98.0             |
| User Number 5 | 194                                          | 97.0             |

TABLE 3. The collected data regarding the experiment No. 2.
According to the results of both experiments, the OABot system was shown to have more than 96% accuracy on average to complete tasks for user commands. In particular, 96.9% accuracy was shown to complete tasks in the office environment with obstacles.

Thus, this OABot system is feasible to install in an office environment with a flat surface (because the OABot is a wheeled robot). Finally, we can conclude that the OABot is a low-cost and effective robot system that can help people in an office environment. In addition, it can complete tasks according to user commands with high accuracy.

Furthermore, the system was tested in lab scale prototype environment, but there is an extending possibility up to 200m distance and increase the number of system users with the current hardware such as MQTT protocol and others. Furthermore, the ROS based movable robot base can move around a larger environment. Hence, in future, we hope to test the system using high performance wireless network in a real-time large-scale environment for multiple users.

Improving the human-robot interaction and communication to be more user friendly is a one possible future direction. We believe that introducing simple commands, designing the Android application interface, managing more users at a time will enhance the user friendliness.

REFERENCES

[1] Hills S., Eraso Y., "Factors associated with non-adherence to social distancing rules during the COVID-19 pandemic: a logistic regression analysis", BMC Public Health. 2021 Feb 13, 21(1), 352, doi: 10.1186/s12889-021-10379-7. PMID: 33581734, PMCID: PMC7881344.

[2] Yang H., Balakunatla M. V., Quiñones J. J., Kaur U., Moser A. E., Doosttalab A., Esquivel-Puentes A., Purwar T., Castillo L., Ma X., Zhang L. T., Voyles R. V., "Occupant-centric robotic air filtration and planning for classrooms for Safer school reopening amid respiratory pandemics", Robotics and Autonomous Systems, 2021, 147, pp. 103919, issn: 0921-8890, doi: 10.1016/j.robot.2021.103919.

[3] Zielinska, "History of service robots", in Robotics: Concepts, Methodologies, Tools, and Applications, IGI Global, 2014, pp. 1–14.

[4] Built In, "What is Robotics? What are Robots? Types & Uses of Robots", https://builtintech.com/robotics, Accessed on: 07, 28, 2021, [Online].

[5] Fifete, "Helicopter Dynamics. Incorporated. "Robotics", https://builtintech.com/robotics, Accessed on: 07, 28, 2021, [Online].

[6] S. S. Khan and A. S. Khan, "A Brief Survey on Robotics", International Journal of Computer Science and Mobile Computing, vol. 6, no. 9, pp. 38–45, 2017. Accessed on: 12, 26, 2020, [Online].

[7] Tsuoda M., Premachandra C., "Remote Control of a Wheeled Robot by Visible Light for Support in Infectious Disease Hospitals", IEEE Access, 2021, vol. 9, pp. 124165-124175, publisher: IEEE.

[8] Tsuoda M., Premachandra C., and Sarathchandra H. A., Harindu Y., Perera K. L. A. N., Lakmal I. T., Premachandra H. W. H., "Visible light communication by using LED array for automatic wheelchair control in hospitals", 2019 IEEE 23rd International Symposium on Consumer Technologies (ISCT), 2019, pp. 210-215, organization: IEEE.

[9] Das A., Styilinger M., Harris D. M., Zenit R., "Force and torque-free haptic teleoperation for robotic surgery", 2019 IEEE International Conference on Robotics and Biomimetics, and Intelligent Computational Systems (Robionetics), 2019, pp. 608-613, publisher: IEEE.

[10] Kazala R., Taneva A., Petrov M., Penkov S., "Wireless network for mobile robot applications", IFAC-PapersOnLine, 2015, vol. 48, no. 24, pp. 231-236. Publisher: Elsevier.

[11] Atmoko R. A., Yang D., "Online monitoring & controlling industrial arm robot using MQTT protocol", IEEE International Conference on Robotics, Biomimetics, and Intelligent Computational Systems (Robionetics), 2018, pp. 12-16, organization: IEEE.

[12] Mukhandi M., Portugal D., Pereira S., Couceiro M. S., "A novel solution for securing robot communications based on the MQTT protocol and ROS", IEEESICE International Symposium on System Integration (SII), 2019, pp. 608-613, organization: IEEE.

[13] MQTT.org, "MQTT: The Standard for IoT Messaging", https://mqtt.org/, Accessed on: 2021 Nov, [Online].

[14] Mukhandi M., Portugal D., Pereira S., Couceiro M. S., "A novel solution for securing robot communications based on the MQTT protocol and ROS", 2019 IEEE/SICE International Symposium on System Integration (SII), Paris, France, 2019, pp. 608-613, doi: 10.1109/SII.2019.8703590.

[15] Eclipse Foundation, "Eclipse Mosquito: An open source MQTT broker", http://www.mosquitto.org, Accessed on: 2021 Nov, [Online].

[16] mcollina, "Mosca: MQTT broker as a module", http://www.mosquitto.org, Accessed on: 2021 Nov, [Online].

[17] Eclipse Foundation, "Eclipse Mosquito: An open source MQTT broker", http://www.mosquitto.org, Accessed on: 2021 Nov, [Online].

[18] Eclipse Foundation, "Paho is an IoT project", https://www.eclipse.org/paho/, Accessed on: 2021 Nov, [Online].

[19] G. Krésmál, Š. Ivan, and H. Ivan, "A web-based IoT solution for monitoring data using MQTT protocol", in International Conference on Smart Systems and Technologies (SST), 2016, pp. 249–253.

[20] A. Rodriguez, "RESTful web services: The basics", IBM developerWorks, vol. 33, pp. 18, 2008. Accessed on: 2021 Apr 26, [Online].

[21] iRobot Corporation, "iRobot Create® Open Interface (OI) Specification based on the iRobot Roomba 600", iRobot Corporation, 2015.

[22] S. I. A. P. Diddeniya, A. M. S. B. Adikari, H. N. Gunasinghe, P. R. S. De Silva, N. C. Ganegoda and W. K. I. L. Wanniarachchi, "Vision Based Office Assistant Robot System for Indoor Office Environment", 2018 3rd
International Conference on Information Technology Research (ICITR), 2018, pp. 1-6, doi: 10.1109/ICITR.2018.8736141.

[36] S. I. A. P. Diddeniya, W. K. I. L. Wanniarachchi, P. R. S. De Silva and N. C. Gamage, "Efficient office assistant robot system: autonomous navigation and controlling based on ROS", International Journal of Multidisciplinary Studies (IJOSS), 2019, vol. 6, no. 1, pp. 64-71.

[37] M. Quigley, B. Gerkey, K. Conley, J. Faust, J. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng., "ROS: an open-source Robot Operating System", ICRA workshop on open source software, Kobe, Japan, 2009, pp. 3.

[38] R. Tellez, "What is ROS?", https://www.thecoductorism.com/what-is-ros/, Accessed on: 2021 Apr 26, [Online].

[39] Willow Garage, "turtlebot", https://github.com/turtlebot/turtlebot, Accessed on: 2021 Apr 26, [Online].

[40] Giorgio G., Cyrill S., Wolfram B., "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters", IEEE Transactions on Robotics, 2007, vol. 23, no. 1, pp. 34-46, doi: 10.1109/TRO.2006.889486.

[41] Norzam W.A.S. and Hawari H.F., Kamarulzaman K., "Analysis of Mobile Robot Indoor Mapping using GMapping Based SLAM with Different Parameter", IOP Conference Series: Materials Science and Engineering, 2019 Dec, vol. 705, pp. 012037, doi: 10.1088/1757-899X/705/1/012037.

[42] Santos J. M., Portugal D., Rocha R. P., "An evaluation of 2D SLAM techniques available in Robot Operating System", 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2013, pp. 1-6.

[43] Open Robotics, "teleop_twist_keyboard", https://github.com/ros-teleop/teleop_twist_keyboard, Accessed on: 2021 Apr 26, [Online].

[44] Kam H. R., Lee S., Park T., Kim C., "RViz: a toolkit for real domain data visualization", Telecommunication Systems, 2015, vol. 60, pp. 337-345.

[45] Open Robotics "amcl", http://wiki.ros.org/amcl, Accessed on: 2021 Apr 26, [Online].

[46] R. A. Light, "Mosquito: server and client implementation of the MQTT protocol", The Journal of Open Source Software, 2017 May, vol. 2, no. 13, DOI: 10.21105/joss.00265.

[47] Google. (n.d.), Retrieved May 22, 2020 from the Android Developer Website: http://developer.android.com/guide/topics/resources/drawable-resource.html.

[48] M. Schuster, "Speech recognition for mobile devices at Google", in Pacific Rim International Conference on Artificial Intelligence, Springer, 2010, pp.8-10.

[49] Bradski, G., "The OpenCV Library", Dr. Dobb's Journal of Software Tools, 2000.

[50] S. I. A. P. Diddeniya, "Office Assistant Robot (OABot)", https://youtube.be/BLaP_Ko1_M, Published on: 2021 Apr 29, [Online].

[51] GIS Resources, "What is SLAM Algorithm and Why SLAM Matters?", https://gisresources.com/what-is-slam-algorithm-and-why-slam-matters/, Accessed on: 2022 Apr 04, [Online].

[52] Taketomi T, Uchiyama H, Taketomi K., "Visual SLAM algorithms: a survey from 2010 to 2016", IPSJ T Comput. Vis. Appl. 9, 16 (2017).

[53] Hess W., Kohler D., Rapp H., Andor D., "Real-time loop closure in 2D LiDAR SLAM", 2016 IEEE international conference on robotics and automation (ICRA),pp. 1271–1278, 2016, IEEE.

[54] Hening S., Ippolito C. A., Krishnakumar K. S., Stepanyan V., Teodorescu M., "3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments", AIAA Information Systems-AIAA Infotech@ Aerospace, pp. 0448, 2017.

[55] Li J., Wang J., Peng H., Hu Y. and Su H., "Fuzzy-torque approximation-enhanced sliding mode control for lateral stability of mobile robot", IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021, IEEE.

[56] Li J., Qin H., Wang J. and Li J., "OpenStreetMap-based autonomous navigation for the four-wheel-legged robot via 3D-Lidar and CCD camera", IEEE Transactions on Industrial Electronics, pp. 2708 - 2717, 2021, vol. 69, no. 3, IEEE.

ISANKA DIDDENIYA received the B.Sc. degree from the University of Sri Jayewardenepura, Nugegoda, Sri Lanka, in 2014 and currently studying for the M.Phil. degree at the same university. From 2015 to 2016, he was a Temporary Demonstrator at the Department of Physics, Faculty of Applied Sciences, University of Sri Jayewardenepura, Nugegoda, Sri Lanka. From 2016 to 2019, he was a Research Assistant with the same department. From 2019 to 2020, he was a Temporary Demonstrator at the Department of Engineering Technology, Faculty of Technology, Sabaragamuwa University of Sri Lanka. He is currently a Research Officer in eVision Microsystems (Pty) Ltd., Panadura, Sri Lanka since 2020. His research interests include robot navigation, robot control and mobile robotics, deep learning and IOT.

INDIKA WANNIARACHCHI is a Senior Lecturer in the Department of Physics at the University of Sri Jayewardenepura. He received his B.Sc. degree with a first-class honor from the same department in 2005. He joined to Department of Science and Technology, Uva Wellassa University of Sri Lanka as a probationary Lecturer at its inception in 2006 and contributed to development of the university in many ways. He completed his PhD thesis in the field of computational physics titled “Low Energy Positron Interactions with Biological Molecules” under the supervision of Prof. Croline Morgan and graduated from the Wayne State University, Michigan, USA (2008-2013). He resumed his duties at Uva Wellassa University in 2013 and extended his contributions to develop the university further in teaching, research, and social responsibilities. In 2015 he joined to the current working place, the Department of Physics, University of Sri Jayewardenepura. After completing his doctorate, he decided to expand his research interest more on applied physics areas: Computer Vision and Image Processing, Computational Physics, Electronics and Embedded Systems and Electronics Structure.

HANSI GUNASINGHE was born in Sri Lanka. She received the B.Sc. degree from the University of Sri Jayewardenepura, Nugegoda, Sri Lanka, in 2014, and currently studying for the Ph.D. degree in the University of Waikato, Hamilton, New Zealand. From 2015 to 2016, she was a Temporary Instructor at the Department of Computer Science, Faculty of Applied Sciences, University of Sri Jayewardenepura, Nugegoda, Sri Lanka. From 2016 to 2017, she was a Temporary Lecturer at the Department of Computer Science, Faculty of Natural Sciences, The Open University of Sri Lanka. Since 2017, she is a Probationary Lecturer at the Department of Computing and Information Systems, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka. Her research interests include computer vision, pattern recognition, robotics, natural language processing and applied machine learning. She has published in several conferences and served as a reviewer for conferences.
CHINTHAKA PREMACHANDRA (Senior Member, IEEE) was born in Sri Lanka. He received the B.Sc. and M.Sc. degrees from Mie University, Tsu, Japan, in 2006 and 2008, respectively, and the Ph.D. degree from Nagoya University, Nagoya, Japan, in 2011. From 2012 to 2015, Dr. Chinthaka Premachandra was an assistant professor in the Department of Electrical Engineering, Faculty of Engineering, Tokyo University of Science, Tokyo, Japan. From 2016 to 2017, he was an assistant professor, from 2018-2022 he was an associate professor in the Department of Electronic Engineering, School of Engineering, Shibaura Institute of Technology, Tokyo, Japan. In 2022, he was promoted to Professor in the Department of Electronic Engineering, Graduate School of Engineering, Shibaura Institute of Technology, where he is currently the Manager of the Image Processing and Robotic Laboratory. His research interests include AI, UAV, Image processing, Audio processing, Intelligent Transport Systems (ITS), Mobile Robotics. Dr. Premachandra is a member of IEICE, Japan; SICE, Japan; and SOFT, Japan. He received the FIT Best Paper Award and the FIT Young Researchers Award from IEICE and IPSJ, Japan, in 2009 and 2010, respectively. He is a recipient of the IEEE Japan medal in 2022. He has served many international conferences and journals as a Steering Committee Member and an Editor, respectively. He is the Founding Chair of the International Conference on Image Processing and Robotics (ICIPRoB) which is technically co-sponsored by the IEEE.

HIROHARU KAWANAKA (Member, IEEE) was graduated from Faculty of Engineering at Mie University in 1999 and received Dr. of Eng. from Mie University Graduate School of Engineering in 2004. After the graduation, he again entered Mie University Graduate School of Medicine to study Medical Informatics and Medical Information Systems. In 2004, he established a company for medical information systems "Medical Engineering Institute, Inc." The company develops "CLISTA!", which is a Data Warehouse for clinical use. Currently, more than 100 large-scale hospitals use our system to analyze clinical data archived in the EHR for clinical studies or business analyses. He is one of the founders of this company and also a director of the company from 2004. From 2006, he joined in Mie University Graduate School of Engineering as an assistant professor from 2007. He received Ph. D. in Medical Science (D.M.Sc) from Mie University Graduate School of Medicine in 2009. From 2017, he is an associate professor at Mie University Graduate School of Engineering. Also, he as invited as a visiting associate professor at Suzuka University of Medical Science, Japan. In 2005, he received a grant "NEDO*1 Industry Fellowship Program" and worked for Mie Technical License Organization (Mie TLO) to support research collaborations, venture companies. His current research topics are Medical Informatics, Medical Document Image Analysis, Graphics Recognition, Welfare Information Systems, Ergonomics, Evolutionary Computations, and its application. He is a member of IEEE (and SMC Society), HIMSS and some Japanese academic societies.