IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 22, 2022, accepted June 6, 2022, date of publication June 14, 2022, date of current version June 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3183110

Human–Robot Communication System for an Isolated Environment

ISANKA DIDDENIYA[1], INDIKA WANNIARACHCHI[1], HANSI GUNASINGHE[2], CHINTHAKA PREMACHANDRA[3], (Senior Member, IEEE), AND HIROHARU KAWANAKA[4], (Member, IEEE)

[1]Department of Physics, Faculty of Applied Sciences, University of Sri Jayewardenepura, Nugegoda 10250, Sri Lanka
[2]Department of Computer Science, School of Computing and Mathematical Sciences, University of Waikato, Hamilton 3216, New Zealand
[3]Department of Electronic Engineering, School of Engineering/Graduate School of Engineering and Science, Shibaura Institute of Technology, Tokyo 135-8548, Japan
[4]Graduate School of Engineering, Mie University, Tsu, Mie 514-8507, Japan

Corresponding author: Indika Wanniarachchi (iwanni@sjp.ac.lk)

This work was supported by the University Grant (University of Sri Jayewardenepura) under Grant ASP/01/RE/SCI/2017/13.

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.

I. INTRODUCTION

Social distancing has become an essential part of our everyday lives in the ongoing COVID 19 pandemic. In enclosed air-conditioned indoor environments, such as in an office where people coming from different locations work together; non-adherence to COVID 19 guidelines increases the risk of infection and the spread of the virus [1]. Sharing physical documents among office workers increases the probability of close interaction between people; a service robot can be useful in such situations. An example of a related service robot in pandemic can be found in [2].

A service robot could help humans by performing a number of jobs, such as delivering goods, cleaning floors, and helping with other chores in an office environment. In general, these robots are either autonomous or can be operated using a built-in control system having manual override options. Service robots are categorized according to personal or professional use. There is no strict technical definition for the term ''service robot''. However, it is defined as ''a robot that performs useful tasks for humans or equipment excluding industrial automation applications'', by the International Organization for Standardization (ISO). They have many forms, structures, and application areas [3].i

As stated by ISO 8373, a robot requires ''a degree of autonomy''. It is the ''ability to complete intended tasks without human intervention, but based on current state and sensing.'' Some service robots are partially autonomous with human-robot interaction, while others are fully autonomous without active human-robot intervention. Statistics for service robots in the International Federation of Robotics include robot systems based on some degree of human-robot interaction

The associate editor coordinating the review of this manuscript and approving it for publication was Stefano Scanzio.

or even complete teleportation, as well as fully autonomous systems [3].

Robots have become a vital part of human society as they help people perform their tasks efficiently [4]. The robotics field involves designing, making, operating, and using robots and computer systems for control, sensory feedback, and information processing of robots. All robots have a particular mechanical structure, a form or a shape, or a frame designed to fulfil a specific task [5]. The mechanical facet is primarily the designer's solution to accomplish the given task and work 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. Three different types of robotic programs namely, remote control [7]–[9], artificial intelligence [10], [11], and hybrid [12]–[14] can be found in the field. Remote control enabled robots to have a set of preexisting commands that will only act if it gets a signal from a human being with remote control. A robot that uses artificial intelligence can interact with the environment without a controlling source. It determines responses to objects and encountered problems through preexisting programming. A robot with hybrid programming 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 a 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 is 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 provide their commands to the robot through voice commands and pressing virtual keys.

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], Zytlebot [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 Orbbec Astra Camera and the Robotic Operating System (ROS) in an Indoor Environment [21], BIT-NAZA based tracking system [54].

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 [54] 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

Machine-to-machine communication in networked robots is maintained using ad hoc or proactive routing protocols. They need high memory and computation requirements and high latency. Complex real-life problems that require real-time execution demand computational capabilities and advanced data analysis that the handling is challenging for networked robots [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 to control and gather data from robot sensors. The authors stated that this method reduces the computing power and energy

consumption of robot nodes [24]. Another study by Atmoko and Yang used the MQTT protocol for online monitoring 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 hijacking and man-in-the-middle attacks. Specifically, remote clients form a connection to the MQTT server. Valid digital certificates authenticate that result in clients receiving authorization from the MQTT server to establish a secure communication channel with the robot [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, designed as a highly lightweight broker-based publish/subscribe messaging protocol for small code footprints such as 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 lower message overheads than other messaging protocols such as CoAP, AMQP and HTTP.

Furthermore, the built-in security features of MQTT are capable of utilizing SSL/TLS to implement standard privacy and security for all data. HTTP and AMQP are two protocols that offer data security using SASL, IPSec and SSL/TLS, but they demand higher bandwidth and resource requirements and are less reliable. Requiring less bandwidth and resources than MQTT, the Quality of Service (QoS) of CoAP is not explicit that only supports unreliable UDP communication [26]. MQTT protocol is employed in several widely used open-source software like Mosca [29], Paho [30] and Mosquitto [28].

The server forwards messages from sensor devices to monitor devices in the hub-and-spoke Message Oriented Middle-ware messaging model. In such architecture, a ''sensor device'' whose main task is continuously producing and sending data to the server is defined as the publisher. Messages from publishers are collected and examined by the central server, an MQTT broker, to identify to whom they need to be sent. All the previously registered devices will receive messages continuously until the cancellation of the subscription. In this architecture, subscribers and publishers do not necessarily need to know each other that is one of the significant advantages of this protocol [31].

D. RESTful WEB SERVER

Web services can be designed using architectural principles defined by REpresentational State Transfer (REST). The principles focus on system resources and how resource states are handled and transferred through HTTP by a broad range of clients written in diverse languages. In the last few years, REST has become a predominant web service design model concerning the number of web services that use it. REST replaced WSDL and SOAP-based interface designs. Since it has an easy to use, simpler style, REST has had such a massive impact on the web. RESTful web services explicitly use HTTP methods to adhere to the protocol, as stated in RFC 2616. For example, GET operation in HTTP is a method of producing data. It is used by the client application to get a resource, extract data from a web server, or run a query that will be looked at by the web server and respond with a set of matching resources [32].

In REST, developers are expected to use HTTP methods explicitly and consistently with the protocol definition. A one-to-one mapping between HTTP methods and create, read, update, and delete (CRUD) operations are established by the basic REST design principle. The mapping POST is used to create a resource on the server; GET retrieves a resource; PUT can update or change the state of a resource; DELETE operation removes or delete a resource in this mapping [32].

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 [50]. 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. This range imaging camera system employs ToF methods to fix the distance between the subject and the camera 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 [51].

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 [52]. Examples of recent research using LiDAR SLAM can be found in [53], [55].

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 [33] 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 [34].

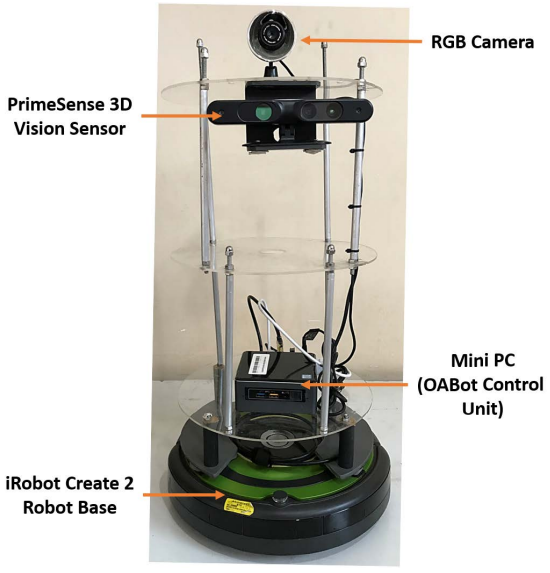

FIGURE 1. The OABot movable robot.

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, the system uses an in-house Android application to control the system by providing user input through any device operated by Android. Installation of the application is available for every office member where they can give commands to the robot to get their documents delivered. Users can give commands using the Android application's voice and virtual key pressing.

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) [35]. 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 [36]. 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 [37].

We installed ROS Indigo Igloo on top of the OS, and an existing ROS package, Turtlebot, was selected as the robot kit. Our study used bring-up, gmapping, keyboard teleop, Rviz view, and amcl nodes, which are built-in to Turtlebot. Like human navigation, the robot has an awareness of the environment it operates to know the available paths in the room and avoid collisions with the surrounding objects. Autonomous navigation is done using a map of the environment fed to the robot before its first use by a computer program.

1) TURTLEBOT BRING-UP

Turtlebot *bringup* [38] 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, environment mapping is achieved by using the *Gmapping* localization technique that can localize and map an unknown environment while it runs. It uses the Rao-Blackwellized Particle Filter (RBPF) for this task. It uses

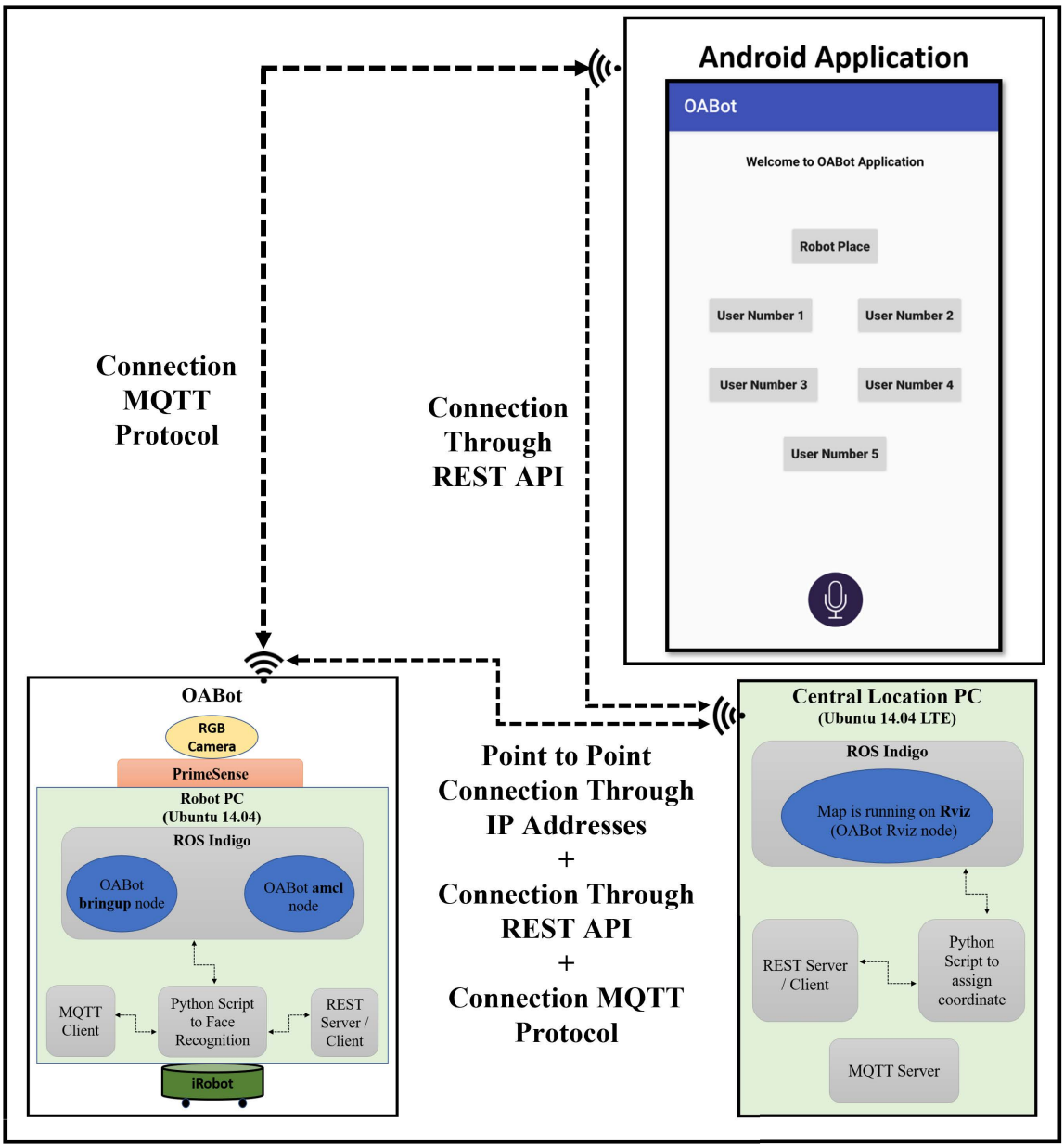

FIGURE 2. The overall system overview.

received data from both the robot pose and the sensor to generate a 2D grid map of the environment without IMU information [39]. In Gmapping, odometry estimation is performed to constantly update the pose of each processed particle by the robot. As the first scan is received during the mapping process, it is directly registered on the map. Afterwards, the registration only occurs if the linear or angular distance traversed by the robot exceeds a specific threshold. Correction of the pose estimation is performed once the scan match is received. It is done on the map of each particle per laser scan to generate a map of the environment [40], [41].

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 [42].

4) RVIZ VIEW

The *RViz* [43] 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

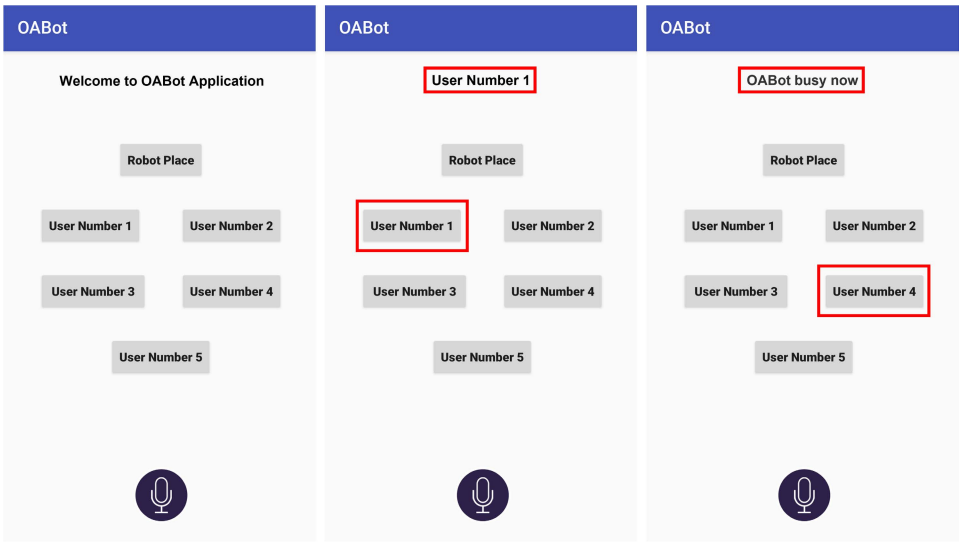

FIGURE 3. The android application interface.

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 [44].

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 [45] 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 [34]. The application was implemented using Android Studio [46]. It provides a graphical user interface to get the document's expected recipient's name. We incorporated a virtual button to allow the user to speak up. The application receives the voice command using the Google Voice Input function [47], which is then recognized using Google Voice Recognition. In Addition, the application is provided with name buttons containing previously assigned usernames. It enables the users to select the desired recipient from the list of buttons. Afterwards, a name string is sent to the central control workstation. The workstation then searches for the recipient's coordinates from a predefined list. Path to the goal obtained through the map and coordinates are sent to the robot to move 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

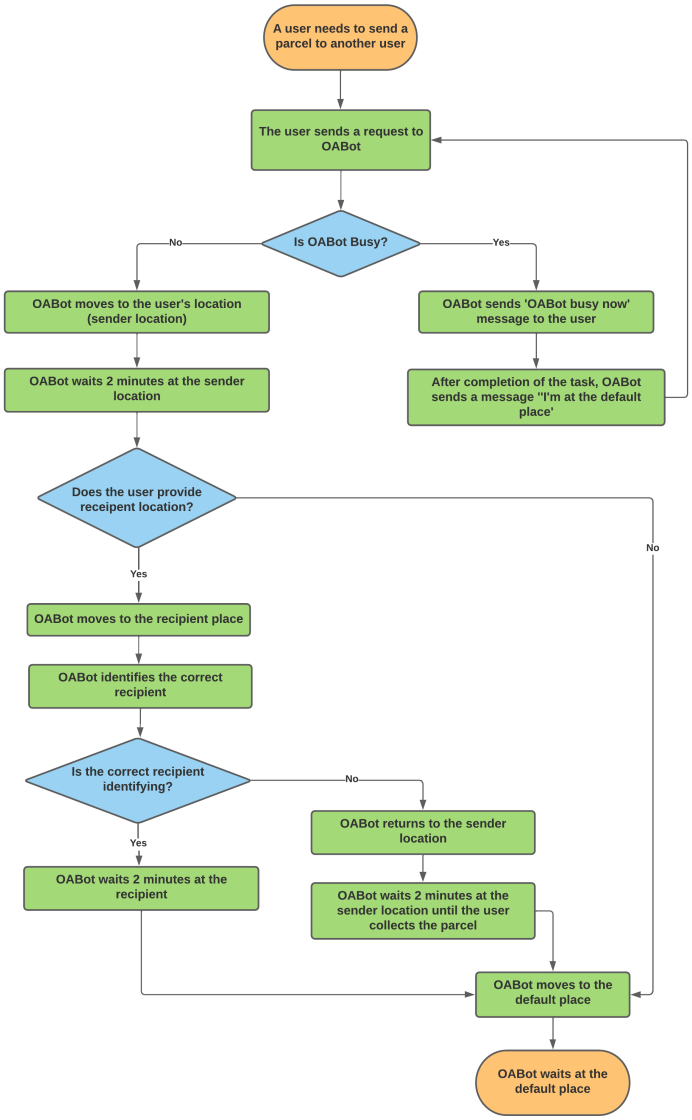

FIGURE 4. The overall process of OABot.

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 [48]. createLBPHFaceRecognizer() 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

FIGURE 5. The created office environment.

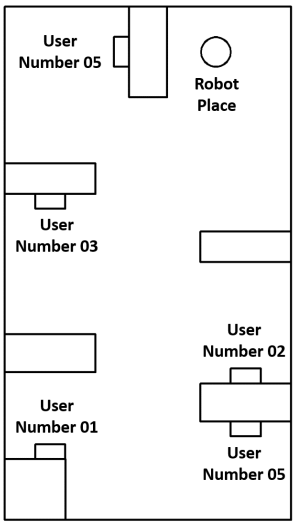

FIGURE 6. The top view design of the created office environment.

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 × 5.74m × 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.

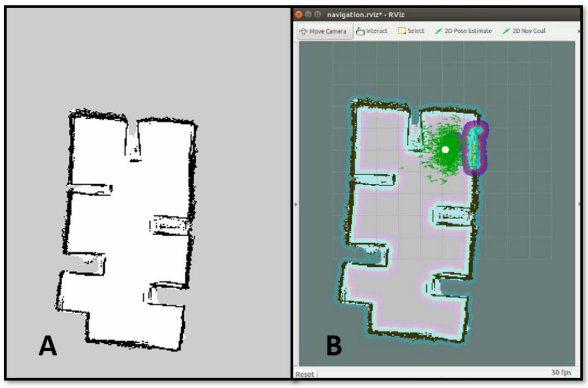

FIGURE 7. The local map and global map of the created office environment.

TABLE 1. Predefined coordinates of users' locations and the coordinate of the OABot's location.

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	-2.620	1.190
5	User number 5	0.237	-7.4
6	Default place of the OABot	0.0	0.0

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

TABLE 2. The collected data regarding the experiment No. 1.

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

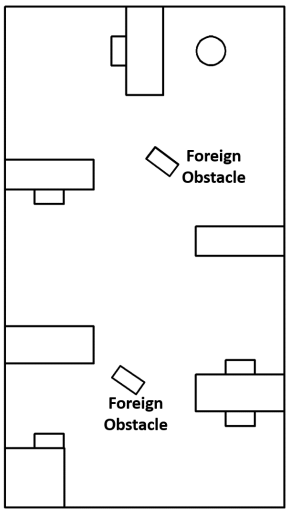

FIGURE 8. The top view design of the created office environment with foreign obstacles.

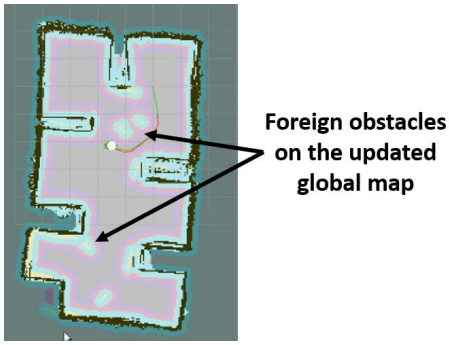

FIGURE 9. The top view design of the created office environment with foreign obstacles.

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

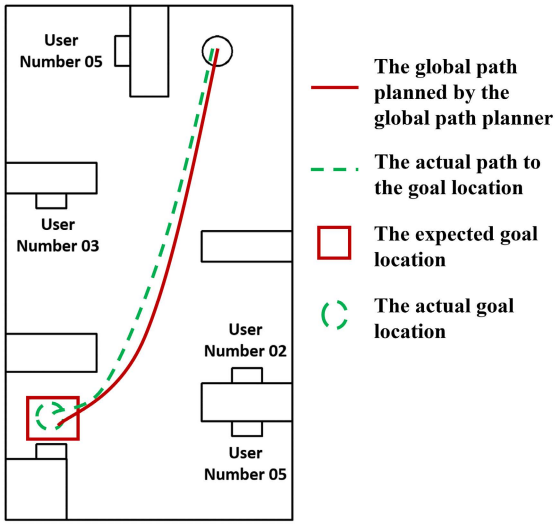

FIGURE 10. A successful trial in the created office environment without foreign obstacles.

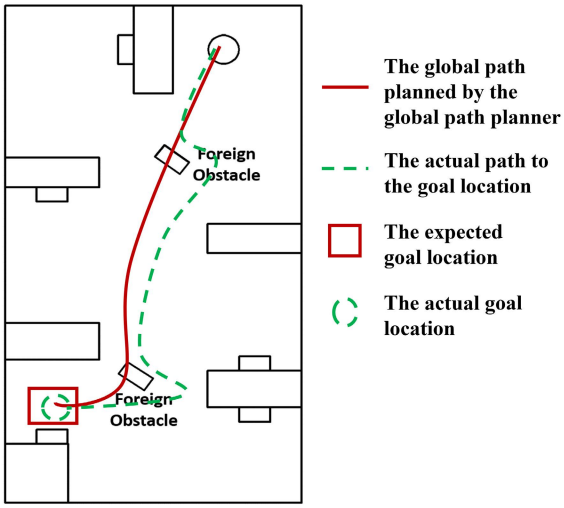

FIGURE 11. A successful trial in the created office environment with foreign obstacles.

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 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

TABLE 3. The collected data regarding the experiment No. 2.

User	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

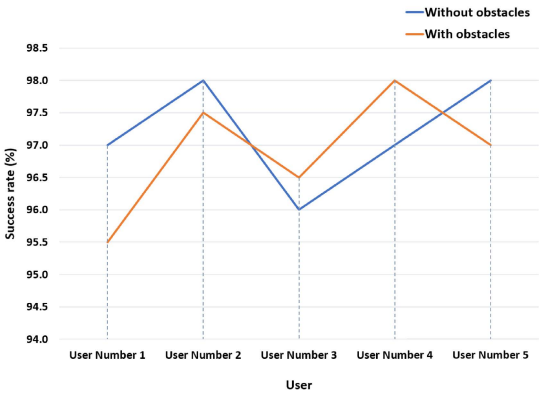

FIGURE 12. The OABot system performance with and without foreign obstacles in the created office environment.

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. 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] S. Hills and Y. Eraso, "Factors associated with non-adherence to social distancing rules during the COVID-19 pandemic: A logistic regression analysis," *BMC Public Health*, vol. 21, no. 1, pp. 1–25, Dec. 2021, doi: 10.1186/s12889-021-10379-7.

[2] H. Yang, M. V. Balakuntala, J. J. Quiñones, U. Kaur, A. E. Moser, A. Doosttalab, A. Esquivel-Puentes, T. Purwar, L. Castillo, X. Ma, L. T. Zhang, and R. M. Voyles, "Occupant-centric robotic air filtration and planning for classrooms for safer school reopening amid respiratory pandemics," *Robot. Auto. Syst.*, vol. 147, Jan. 2022, Art. no. 103919, doi: 10.1016/j.robot.2021.103919.

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

[4] Built In. *What is Robotics? What are Robots? Types & Uses of Robots*. Accessed: Jul. 28, 2021. [Online]. Available: https://builtin.com/robotics

[5] Finite Helical Dynamics Incorporated. *Robotics*. Accessed: Jul. 28, 2021. [Online]. Available: https://builtin.com/robotics

[6] S. S. Khan and A. S. Khan, "A brief survey on robotics," *Int. J. Comput. Sci. Mobile Comput.*, vol. 6, no. 9, pp. 38–45, 2017. Accessed: Dec. 26, 2020.

[7] M. Tsunoda and C. Premachandra, "Remote control of a wheeled robot by visible light for support in infectious disease hospitals," *IEEE Access*, vol. 9, pp. 124165–124175, 2021.

[8] M. Tsunoda, C. Premachandra, H. A. H. Y. Sarathchandra, K. L. A. N. Perera, I. T. Lakmal, and H. W. H. Premachandra, "Visible light communication by using LED array for automatic wheelchair control in hospitals," in *Proc. IEEE 23rd Int. Symp. Consum. Technol. (ISCT)*, Jun. 2019, pp. 210–215.

[9] A. Das, M. Styslinger, D. M. Harris, and R. Zenit, "Force and torque-free helical tail robot to study low Reynolds number microorganism swimming," 2021, *arXiv:2112.00850*.

[10] S. Panesar, Y. Cagle, D. Chander, J. Morey, J. Fernandez-Miranda, and M. Kliot, "Artificial intelligence and the future of surgical robotics," *Ann. Surg.*, vol. 270, no. 2, pp. 223–226, 2019.

[11] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, "Artificial intelligence for long-term robot autonomy: A survey," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4023–4030, Oct. 2018.
[12] M. Demirhan and C. Premachandra, "Development of an automated camera-based drone landing system," *IEEE Access*, vol. 8, pp. 202111–202121, 2020.
[13] C. Premachandra, R. Gohara, T. Ninomiya, and K. Kato, "Smooth automatic stopping for ultra-compact vehicles," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 4, pp. 561–568, Dec. 2019.
[14] N. Ando and R. Kanzaki, "Insect-machine hybrid robot," *Current Opinion Insect Sci.*, vol. 42, pp. 61–69, Dec. 2020, doi: 10.1016/j.cois.2020.09.006.
[15] I. Okunevich, D. Trinitatova, P. Kopanev, and D. Tsetserukou, "MobileCharger: An autonomous mobile robot with inverted delta actuator for robust and safe robot charging," in *Proc. 26th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2021, pp. 1–8.
[16] A. M. Aljafar Muhammad Afan, R. Angga, and P. Irwan, "Sistem navigasi kursi roda otonom dengan sensor laser," *eProc. Eng.*, vol. 8, no. 5, p. 4554, 2021.
[17] J. M. Faria and A. H. J. Moreira, "Implementation of an autonomous ROS-based mobile robot with AI depth estimation," in *Proc. Annu. Conf. IEEE Ind. Electron. Soc.*, vol. 47, 2021, pp. 1–6.
[18] M. Ryota, T. Naofumi, K. Sho, O. Masashi, and T. Hideki, "ZytleBot: FPGA integrated ROS-based autonomous mobile robot," in *Proc. 2021 Int. Conf. Field-Program. Technol. (ICFPT)*, 2021, pp. 1–4.
[19] H. M. Montes, R. Humberto, E. Octavio, and P. Víctor, "Semi-autonomous mobile robot for environmental surfaces disinfections against SARS-CoV-2," in *Proc. Climbing Walking Robots Conf.* Cham, Switzerland: Springer, 2021, pp. 317–328.
[20] R. Jawad, R. Anikesh, and G. Gayathri, "Autonomous mobile robot for visual inspection of MEP provisions," *J. Phys., Conf. Ser.*, vol. 2070, no. 1, Nov. 2021, Art. no. 012199.
[21] M. Basavanna, M. Shivakumar, and K. R. Prakash, "Navigation of mobile robot through mapping using Orbbec Astra camera and ROS in an indoor environment," in *Recent Advances in Manufacturing, Automation, Design and Energy Technologies*. Singapore: Springer, 2022, pp. 465–474.
[22] O. Saha and P. Dasgupta, "A comprehensive survey of recent trends in cloud robotics architectures and applications," *Robotics*, vol. 7, no. 3, p. 47, 2018.
[23] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills, and S. Chernova, "Robot web tools: Efficient messaging for cloud robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 4530–4537.
[24] R. Kazala, A. Taneva, M. Petrov, and S. Penkov, "Wireless network for mobile robot applications," *IFAC-PapersOnLine*, vol. 48, no. 24, pp. 231–236, 2015.
[25] R. A. Atmoko and D. Yang, "Online monitoring & controlling industrial arm robot using MQTT protocol," in *Proc. IEEE Int. Conf. Robot., Biomimetics, Intell. Comput. Syst. (Robionetics)*, Aug. 2018, pp. 12–16.
[26] M. Mukhandi, D. Portugal, S. Pereira, and M. S. Couceiro, "A novel solution for securing robot communications based on the MQTT protocol and ROS," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Paris, France, Jan. 2019, pp. 608–613, doi: 10.1109/SII.2019.8700390.
[27] *MQTT: The Standard for IoT Messaging*. Accessed: Nov. 20, 2021. [Online]. Available: https://mqtt.org/
[28] Eclipse Foundation. *Eclipse Mosquitto: An Open Source MQTT Broker*. Accessed: Nov. 20, 2021. [Online]. Available: http://www.mosquitto.org
[29] mcollina. *mosca: MQTT Broker as a Module*. Accessed: Nov. 20, 2021. [Online]. Available: http://www.mosca.io/
[30] Eclipse Foundation. *Paho is an IoT Project*. Accessed: Nov. 20, 2021. [Online]. Available: https://www.eclipse.org/paho/
[31] K. Grgić, I. Špeh, and I. Hedi, "A web-based IoT solution for monitoring data using MQTT protocol," in *Proc. Int. Conf. Smart Syst. Technol. (SST)*, Oct. 2016, pp. 249–253.
[32] A. Rodriguez, "RESTful web services: The basics," *IBM Developer Works*, vol. 33, p. 18, Nov. 2008. Accessed: Apr. 26, 2021.
[33] *iRobot Create 2 Open Interface (OI): Specification Based on the iRobot Roomba 600*, iRobot Corp., Bedford, MA, USA, 2015.
[34] 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," in *Proc. 3rd Int. Conf. Inf. Technol. Res. (ICITR)*, Dec. 2018, pp. 1–6, doi: 10.1109/ICITR.2018.8736141.
[35] S. I. A. P. Diddeniya, W. K. I. L. Wanniarachchi, P. R. S. De Silva, and N. C. Ganegoda, "Efficient office assistant robot system: Autonomous navigation and controlling based on ROS," *Int. J. Multidisciplinary Stud.*, vol. 6, no. 1, pp. 64–71, 2019.
[36] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, 2009, p. 5.
[37] R. Tellez. *What is ROS?* Accessed: Apr. 26, 2021. [Online]. Available: https://www.theconstructsim.com/what-is-ros/
[38] W. Garage. *TurtleBot*. Accessed: Apr. 26, 2021. [Online]. Available: https://github.com/turtlebot/turtlebot
[39] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao–Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007, doi: 10.1109/TRO.2006.889486.
[40] W. A. S. Norzam, H. F. Hawari, and K. Kamarudin, "Analysis of mobile robot indoor mapping using GMapping based SLAM with different parameter," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 705, no. 1, Nov. 2019, Art. no. 012037, doi: 10.1088/1757-899X/705/1/012037.
[41] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in robot operating system," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Oct. 2013, pp. 1–6.
[42] *Open Robotics*. Accessed: Apr. 26, 2021. [Online]. Available: https://github.com/ros-teleop/teleop_twist_keyboard
[43] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "RViz: A toolkit for real domain data visualization," *Telecommun. Syst.*, vol. 60, no. 2, pp. 337–345, Oct. 2015.
[44] Open Robotics. *AMCL*. Accessed: Apr. 26, 2021. [Online]. Available: http://wiki.ros.org/amcl
[45] R. A. Light, "Mosquitto: Server and client implementation of the MQTT protocol," *J. Open Source Softw.*, vol. 2, no. 13, p. 265, May 2017, doi: 10.21105/joss.00265.
[46] (May 22, 2020). *Google*. [Online]. Available: http://developer.Android.com/guide/topics/resources/drawable-resource.html
[47] M. Schuster, "Speech recognition for mobile devices at Google," in *Proc. Pacific Rim Int. Conf. Artif. Intell.* Daegu, South Korea: Springer, 2010, pp. 8–10.
[48] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, vol. 25, no. 11, pp. 120–123, 2000.
[49] S. I. A. P. Diddeniya. *Office Assistant Robot (OABot)*. Accessed: Apr. 29, 2021. [Online]. Available: https://youtu.be/RIaP_Ko1_M
[50] GIS Resources. *What is SLAM Algorithm and Why SLAM Matters?* Accessed: Apr. 4, 2022. [Online]. Available: https://gisresources.com/what-is-slam-algorithm-and-why-slam-matters/
[51] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: A survey from 2010 to 2016," *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, no. 1, p. 16, Dec. 2017.
[52] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LiDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1271–1278.
[53] S. Hening, C. A. Ippolito, K. S. Krishnakumar, V. Stepanyan, and M. Teodorescu, "3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments," in *Proc. AIAA Inf. Syst.-AIAA Infotech @ Aerosp.*, Jan. 2017, p. 448.
[54] J. Li, J. Wang, H. Peng, Y. Hu, and H. Su, "Fuzzy-torque approximation-enhanced sliding mode control for lateral stability of mobile robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 4, pp. 2491–2500, Apr. 2022.
[55] J. Li, H. Qin, J. Wang, and J. Li, "OpenStreetMap-based autonomous navigation for the four wheel-legged robot via 3D-LiDAR and CCD camera," *IEEE Trans. Ind. Electron.*, vol. 69, no. 3, pp. 2708–2717, Mar. 2022.

ISANKA DIDDENIYA received the B.Sc. degree from the University of Sri Jayewardenepura, Nugegoda, Sri Lanka, in 2014, where he is currently pursuing the M.Phil. degree.

From 2015 to 2016, he was a Temporary Demonstrator with the Department of Physics, Faculty of Applied Sciences, University of Sri Jayewardenepura, where he was a Research Assistant, from 2016 to 2019. From 2019 to 2020, he was a Temporary Demonstrator with the Department of Engineering Technology, Faculty of Technology, Sabaragamuwa University of Sri Lanka. He has been a Research Officer at eVision Microsystems (Pvt) Ltd., Panadura, Sri Lanka, since 2020. His research interests include robot navigation, robot control and mobile robotics, deep learning, and the IoT.

INDIKA WANNIARACHCHI received the B.Sc. degree (Hons.) from the University of Sri Jayewardenepura, in 2005, and the Ph.D. degree from Wayne State University, Michigan, USA, in 2013.

He joined the 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 Ph.D. thesis in the field of computational physics titled ''Low Energy Positron Interactions with Biological Molecules'' under the supervision of Prof. Croline Morgan. 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, such as computer vision and image processing, computational physics, electronics and embedded systems, and electronics structure. He is currently a Senior Lecturer with the Department of Physics, University of Sri Jayewardenepura.

HANSI GUNASINGHE was born in Sri Lanka. She received the B.Sc. degree from the University of Sri Jayewardenepura, Nugegoda, Sri Lanka, in 2014. She is currently pursuing the Ph.D. degree with the University of Waikato, Hamilton, New Zealand. From 2015 to 2016, she was a Temporary Instructor with the Department of Computer Science, Faculty of Applied Sciences, University of Sri Jayewardenepura. From 2016 to 2017, she was a Temporary Lecturer with the Department of Computer Science, Faculty of Natural Sciences, The Open University of Sri Lanka. Since 2017, she has been a Probationary Lecturer with the Department of Computing and Information Systems, Faculty of Applied Sciences, Sabaragamuwa University of Sri Lanka. She has published in several conferences and served as a reviewer for conferences. Her research interests include computer vision, pattern recognition, robotics, natural language processing, and applied machine learning.

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, he was an Assistant Professor with 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 to 2022, he was an Associate Professor with the Department of Electronic Engineering, School of Engineering, Shibaura Institute of Technology, Tokyo. In 2022, he was promoted to a Professor with 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), and mobile robotics. He 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 was 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) received the degree from the Faculty of Engineering, Mie University, in 1999, and the Dr.Eng. degree and the Ph.D. degree (D.M.Sc.) in medical science from the Graduate School of Engineering, Mie University, in 2004 and 2009, respectively. After the graduation, he again entered the Graduate School of Medicine, Mie University, to study medical informatics and medical information systems. In 2004, he established Medical Engineering Institute, Inc., company for medical information systems. 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 has been the Director of the company, since 2004. In 2006, he joined the Graduate School of Engineering, Mie University, as an Assistant Professor, where he became an Associate Professor, in 2007. Since 2017, he has been an Associate Professor wit the Graduate School of Engineering, Mie University. He was invited as a Visiting Associate Professor with the 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 interests include medical informatics, medical document image analysis, graphics recognition, welfare information systems, ergonomics, evolutionary computations, and its application. He is a member of SMC Society, HIMSS, and some Japanese academic societies.