An autonomous Mobile Robot System based on Serverless Computing and Edge Computing

1st Tri Thong Tran
Dept. of Information Management
National Taiwan University
Taipei, Taiwan(R.O.C)
r08725053@ntu.edu.tw

2nd Yu-Chen Zhang
Dept. of Information Management
Fu Jen Catholic University
New Taipei, Taiwan(R.O.C)
404401258@mail.fju.edu.tw

3rd Wei-Tung Liao
Dept. of Information Management
Fu Jen Catholic University
New Taipei, Taiwan(R.O.C)
404402111@mail.fju.edu.tw

4th Yu-Jen Lin
Dept. of Information Management
Fu Jen Catholic University
New Taipei, Taiwan(R.O.C)
404401284@mail.fju.edu.tw

5th Ming-Chia Li
Dept. of Information Management
Fu Jen Catholic University
New Taipei, Taiwan(R.O.C)
404401612@mail.fju.edu.tw

6th Huai-Sheng Huang
Dept. of Information Management
Fu Jen Catholic University
New Taipei, Taiwan(R.O.C)
hshuang@im.fju.edu.tw

***Abstract*—Strengthen by Artificial Intelligence (AI) and complexly integrated sensors, an autonomous mobile robot (AMR) is extensively applied in coping with various human resources tasks in indoor office environments. However, implementing an AMR system from scratch needs a strong Electric Engineer background due to the complexity of robot-controlling. Besides, Communication between robot-server, sensor, and client-service also increases the difficulty and time-cost in AMR developing. In this paper, the AMR system we proposed aims to be implemented by people without a EE background, and we will achieve this by employing Robot Operating System. Besides, the AMR system should work independently but still capable of responding to human requests. We will demonstrate a serverless could structure that includes the client-side service and integrate the edge-computing, which in charge of the immediacy-demanding job.**

***Index Terms*—autonomous mobile robot, AMR, cloud computing, edge computing, serverless**

I. Introduction

Autonomous mobile robot (AMR) [1] is increasingly popular in industries because of its degree of autonomy. To achieve high-autonomy, AMR needs to combine numerous-integrated sensors, such as lidar and 3D depth cameras. However, this could cause customizing a robot system to suit every sensor as time-consuming and inflexible. Robot Operating System(ROS), a robotic middleware, came to address the problem by modularizing features or sensors, allows developers to implement the opensource module and deal with message passing more easily with its broadcasting design. In the meantime, we want to expand ROS-based robots' modularity and flexibility by integrating ROS robot with cloud service.

The whole robot application would be more easily switch to different services and be more efficient since cloud service can cover computing jobs, reduce the edge device's loading and power consumption [2]. So the robot could focus on the job such as mapping, navigation, and sending the necessary data to the cloud IoT service. Then, the cloud IoT service will trigger other services to process data without running a server standby. By doing so, the whole structure would be more flexible since the developer can decide which services

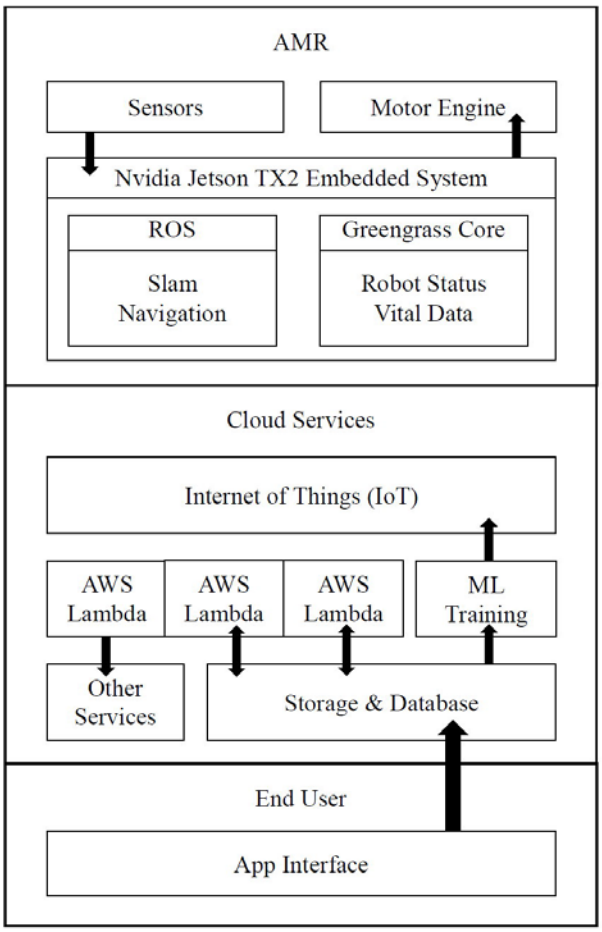

Fig. 1. System Architecture.

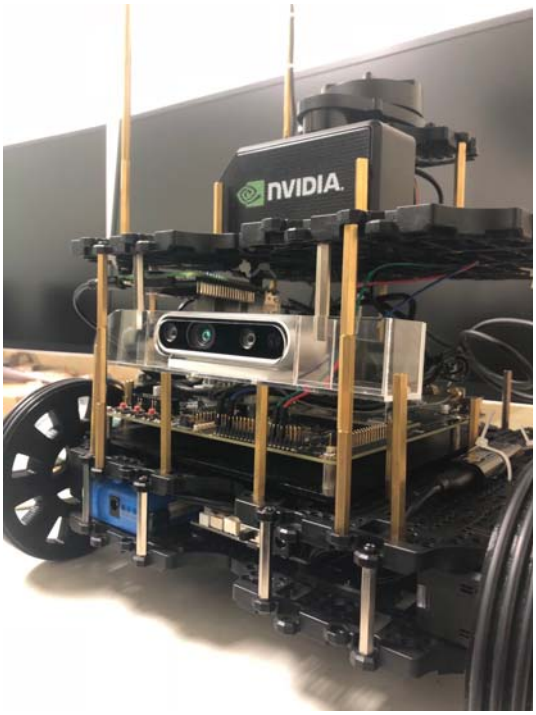

Fig. 2. AMR implementation.

can be triggered. The decoupling of work would make develop features and issue tracking more accessible.

Fig. 1 shows our system architecture. This paper will demonstrate an AMR system that comprises three parts: an AMR with NVIDIA Jetson TX2 module for edge computing, a serverless architecture based on Amazon Web Services (AWS) [3] for cloud computing, and an user interface developed using React Native [17].

As we assume the AMR system is designed for indoor-delivering service, we will take sending documents as our mission. Additionally, the AMR system was equipped with a face-recognizing feature to check the sender and receiver identification. According to our serverless system, it can cache all requests and then trigger a chain of cloud-based micro-services to notify AMRs. In the meantime, a power-efficient module based on the edge computing can handle the AMR working during an unstable or disconnected network connection.

II. System Description

A. Autonomous Mobile Robot Implementation

In Fig. 2, Turtlebot3 [4] is utilized to build fundamental AMR, which provides Open-source Control Module for ROS (OpenCR) [5] and DYNAMIXEL [6], a smart actuator system for robot engine. Considering the need for edge computing, such as obstacle avoidance, NVIDIA Jetson TX2 module [7] is utilized to serve as edge computing device. We add a 360-degree lidar module for surrounding detection to make advances to environment detection. We also use a monocular webcam for human face recognition and a 3D depth camera, Intel RealSense Depth Camera D435 [8], for additional environmental detection. In the aspect of robot software, ROS is used, which provides packages such as Simultaneous Localization And Mapping (SLAM), and navigation. We install the software development kit (SDK) from AWS Greengrass [9], which provides a connection with other AWS IoT services.

B. Serverless Architecture

Fig. 3. shows the serverless architecture based on AWS. The AWS Greengrass is used to manage devices and deploy Lambda function, a collection of code, to the AMR. Also, vital data including AMR status can be sent back to AWS IoT Core [10] through AWS Greengrass. Once the data has been received, AWS IoT can trigger corresponding services by executing a function created on AWS Lambda [11] such as access to Amazon DynamoDB [12], an efficient NoSQL database, to store user's requests and position coordinate.

On the other side, AWS SageMaker [13] is used to train model for face recognition based on the edge computing to identify sender and receiver. Training data such as facial images of users are stored in Amazon S3 [14], which provide reliable object storage. Facial images were regularly retrieved by AWS Lambda, which triggered AWS SageMaker to train facial model recursively. The deployment of the new-trained model on AMR can be done automatically through AWS Greengrass during the wired network is attached (e.g., during AMR is charging its power) or wireless network is reliable.

C. User Interface

Fig. 4. shows the user interface. In response to the user interface such as App, the Amazon Cognito [15] is used to federate with the three-party account such as Google or Facebook. In the aspect of data storage, Amazon DynamoDB is adopted. For reliability, user's requests should be temporarily stored in Amazon SQS [16] first in case of Input /Output Operations Per Second (IOPS) overloading.

Considering to be a cross-platform mobile application, React Native [17] was chosen to develop the user interface. For Interacting with AWS services, such as uploading images to Amazon S3 could be done through AWS SDK.

D. Demonstration

Fig. 5. shows the demo demonstration. Users should upload profile information such as facial image on register page at the first time. Afterwards, once a delivery request has been made, the AMR would start its mission, sending the package to receiver's position and then identify receiver's face. Once the identification had been confirmed, the mission was completed.

III. Conclusion

To sum up, it is essential to take advantage of edge computing and cloud services to design a well-developed AMR system. Cloud computing can reduce the power consumption of AMR and be more easily integrated AMR with other IoT devices. Besides, AMR should make use of the increasingly powerful edge computing device, which can help AMR

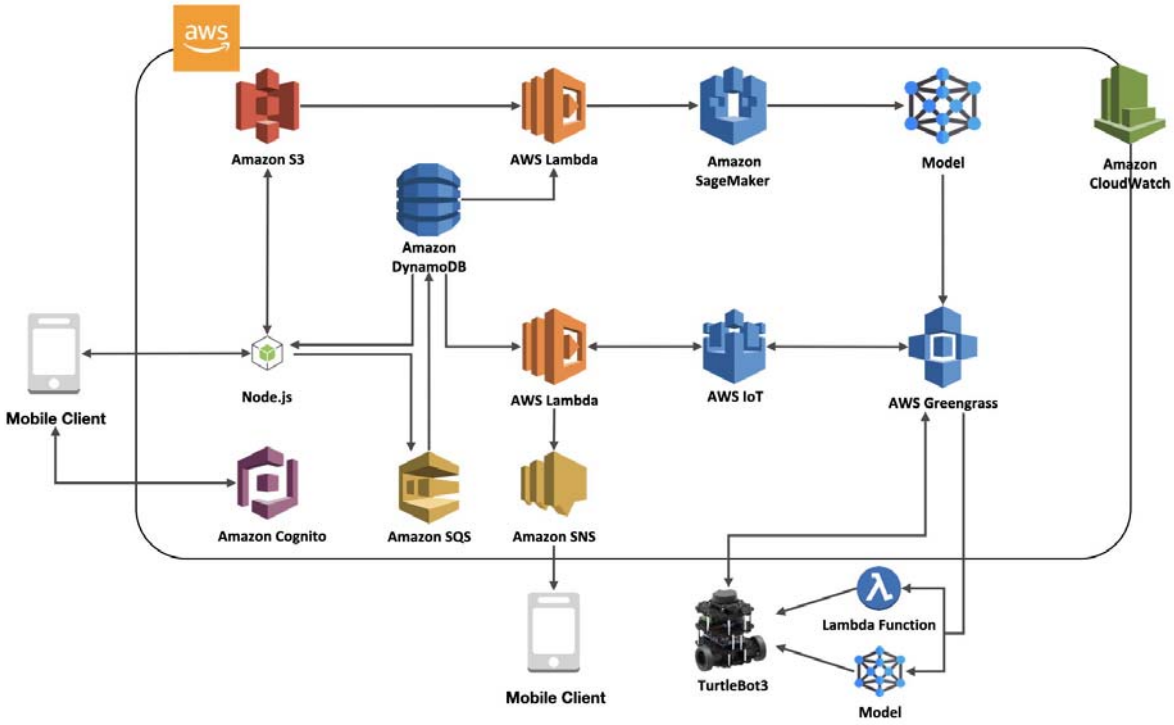

Fig. 3. Serverless Architecture.

quickly respond to any undesirable situation. With the help of cloud computing and edge computing, we can develop a powerful AMR elegantly and swiftly.

As 5G technologies bolster the IoT industry, people are seeking what the best approach to managing and developing IoT devices is? Here, our implementation serves as a feasible solution we tried. We hope it can offer an insightful view about developing AMR solutions.

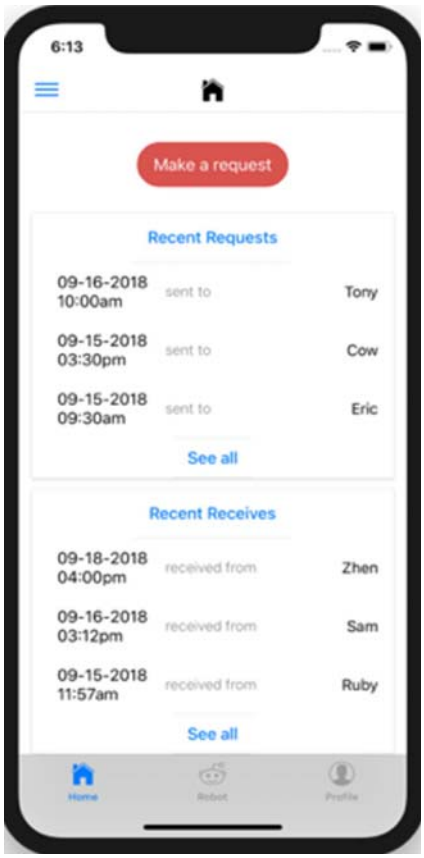

Fig. 4. App Interface Screenshot.

ACKNOWLEDGMENT

The research is co-sponsored by Ministry of Science and Technology (MOST), grants 109-2221-E-030-014-MY3.

REFERENCES

[1] Robotics Online Marketing Team, "The Latest Technological Innovations in Autonomous Mobile Robots", Robotics Online Blog, May 2018.
[2] Ken Goldberg, "Cloud Robotics and Automation", Fall 2018.
[3] Amazon Web Services, "https://aws.amazon.com".
[4] Turtlebot3, "http://www.robotis.us/turtlebot-3".
[5] OpenCR: Open Source Control Module for ROS, "https://robots.ros.org/opencr/".
[6] DYNAMIXEL, "http://www.robotis.us/dynamixel/".
[7] NVIDIA, "Jetson TX2 Module", https://developer.nvidia.com/embedded/jetson-tx2.
[8] Intel, "RealSens Depth Camera D435", https://www.intelrealsense.com/depth-camera-d435.
[9] AWS, "Greengrass", https://aws.amazon.com/greengrass.
[10] AWS, "IoT Core", https://aws.amazon.com/iot-core.
[11] AWS, "Lambda", https://aws.amazon.com/lambda.
[12] AWS, "Amazon DynamoDB", https://aws.amazon.com/dynamodb.
[13] AWS, "SageMaker", https://aws.amazon.com/sagemaker.
[14] AWS, "Amazon S3", https://aws.amazon.com/s3.
[15] AWS, "Amazon Cognito", https://aws.amazon.com/cognito.
[16] AWS, "Amazon SQS", https://aws.amazon.com/sqs.
[17] React Native, "https://facebook.github.io/react-native".

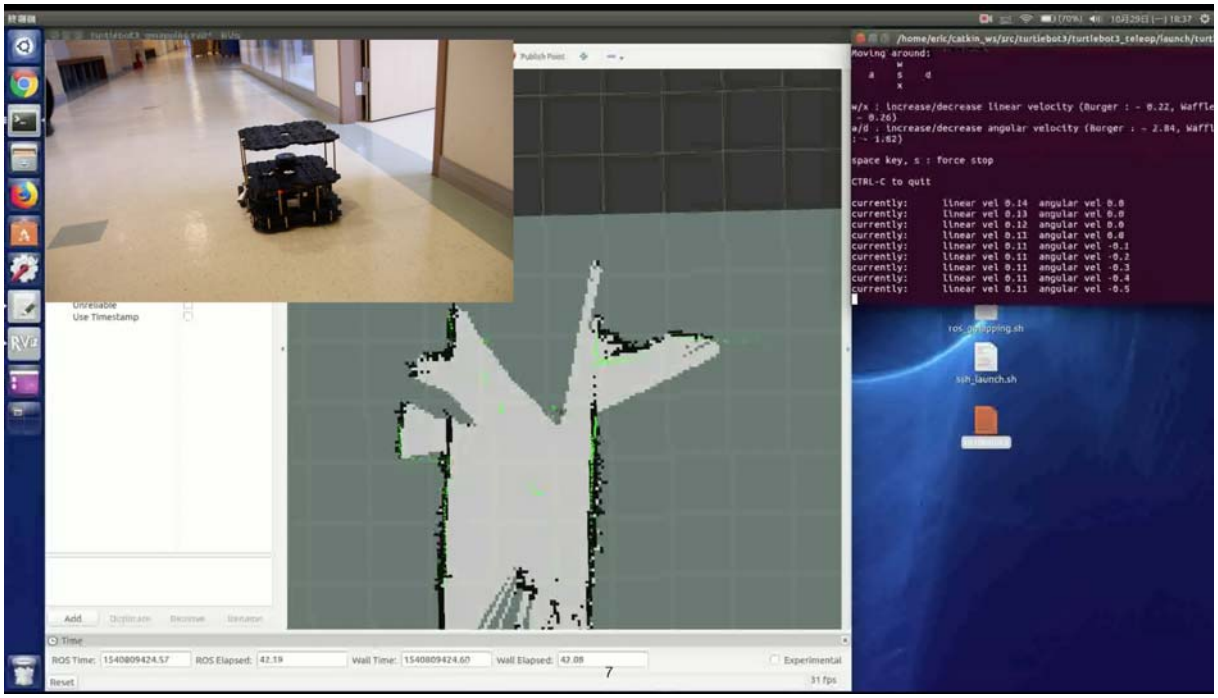

Fig. 5. Demonstration.