IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 26, 2019, accepted May 1, 2019, date of publication May 7, 2019, date of current version May 17, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2915344*

# Design and Implementation of a Network Robotic Framework Using a Smartphone-Based Platform

**JONGIL LIM[1], GIRMA S. TEWOLDE[2], (Member, IEEE), JAEROCK KWON[2], (Member, IEEE), AND SEYEONG CHOI[3], (Senior Member, IEEE)**

[1]Agency for Defence Development, Daejeon 34185, South Korea
[2]Kettering University, Flint, MI 48504, USA
[3]Wonkwang University, Iksan, Jeonbuk 54538, South Korea

Corresponding author: Seyeong Choi (sychoi@wku.ac.kr)

This work was supported by Wonkwang University in 2018.

**ABSTRACT** With recent advances in robotics along with sensor technologies, robots have richer features and can be used in more diverse applications. However, onboard microcontrollers in robots often take full responsibility for processing data acquired from sensors, controlling actuators, and other tasks. This prohibits us from implementing complex algorithms in microcontrollers. To address this issue, we present a novel design of a network robotic framework using a smartphone-based robotic platform. The design was implemented using a smartphone as a network bridge to distribute computationally burdensome tasks into multiple networked computational resources, as well as, a sensor package that allows the mobile robot to navigate. One of the most practical benefits derived from the outsourced computations is the simplicity in a robot's design. This provides a sturdy robot that consumes less power. With interaction capabilities to networked resources, more diverse robotic applications can be developed. To demonstrate the feasibility of the proposed network robotic framework, several experiments were conducted for indoor localization and navigation using a novel smartphone-based robotic platform.

**INDEX TERMS** Network robot, mobile robot, robotic framework, smartphone-based platform, localization, navigation, Bluetooth.

## I. INTRODUCTION

Recent advances in robotics and sensor technologies have provided robots with richer features and allow them to participate in diverse applications. Onboard microcontrollers in standalone robots, however, take a full responsibility for processing acquired sensor data and controlling actuators, in addition to providing some level of decision making. This prohibits robotics researchers from using complex, enhanced algorithms in small mobile robots in which the microcontrollers cannot computationally afford the algorithms. Using the particle filters algorithm for localization of a mobile robot is an example of this unaffordability [1]. Bayesian recursion equations must be implemented for the particle filters algorithm. The recursive property of this approach is computationally expensive in that all particles must be updated every time new sensor measurements arrive and re-sampling processes are required.

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

Networked robotics has emerged to overcome these restrictions in standalone robots [2]–[10]. A networked robotics system is defined as an integrated single system where the resources at identified network locations can be connected together and used to perform robotic tasks [2]–[7]. Researchers have designed systems with multiple robots that harmoniously work together to accomplish tasks [8]–[10]. Google's object recognition engine has been used in a cloud-based robot to grasp objects [7], and practical implementation research has been conducted by a research group working with Google's Cloud Robotics application program interfaces (APIs) for Java and Android using the Personal Robot 2 (PR2) Willow Garage robot. As a cloud-based robotic platform, RoboEarth has been proposed for sharing knowledge between robots [8], where bold claims have been made such as a World Wide Web for robots. Several architectures for cloud robotics were proposed for both machine-to-machine and machine-to-cloud communications [9].

A smartphone has many tools that aid a mobile robot, such as processors, sensors, cameras, and wireless

communications. Using the high-end microprocessors and a wide variety of accurate, expensive sensors to provide a robot with computational power and sensor package is not novel in mobile robotics [11]–[15]. A smartphone-based Bluetooth wireless robot control system was proposed in [11]. Quick Response (QR)-code based robot navigation using a smartphone was successfully demonstrated in [12]. With the assistance of a QR-code, an inertial measurement unit (IMU)-based robot localization method was introduced in [13]. The quality of communication was analyzed for remote control of a robot using an iPhone in [14]. An Android-based mobile robotic platform was proposed and demonstrated using a real research problem [15]. However, most of the aforementioned approaches simply used a smartphone to add a simple feature, such as a remote controller [11], [14], an additional computing tool [13], [15], and a sensor package [12] to remove the necessity of additional expensive sensors.

Note that none of the aforementioned results is a practical solution for small mobile robots to free up limited resources in onboard microcontrollers in terms of memory, storage, and execution speed. Network robotics often refers to studies regarding either interconnected multiple robots that work together or a single robot that is network-connected to utilize remote resources. Because there are different definitions regarding network robotics, we would like to clarify that our network robotic framework is not intended for multiple robots, but for a small single robot that has a network connection through a smartphone. Connectivity liberates the robot from the limitations of the onboard microcontroller.

The design of a novel network framework using a smartphone-based robotic platform is presented in this paper. The design was implemented with a smartphone as a network bridge to distribute computationally burdensome tasks into multiple networked computational resources and as a part of sensors for the mobile robot to navigate. Practical benefits can be derived from outsourced computation. The robot design can be made much simpler, even with wireless capabilities and messaging protocols. This simplicity reduces power consumption and extends battery life, which is often a critical issue for mobile robots.

The proposed robotic framework allows multiple networked computational resources to share the computational burden. These capabilities of interacting with networked resources allow the development of more diverse robotic applications. To show the feasibility of the proposed framework, several experiments for indoor localization and navigation using a novel smartphone-based mobile robotic platform will be described.

## II. PLATFORM STRUCTURE

Fig. 1 shows the overall design of the proposed network robotic framework. A two-layer control structure is designed for navigation. The lower layer is responsible for relatively simple tasks that are mostly conducted in the onboard microcontroller (e.g., Arduino [16], Raspberry Pi [17], or Beagle [18]). Such tasks include processing sensor readings and controlling motors. The higher layer is in charge of processing tasks that require heavy-duty computational power that must be executed in the outside task server.

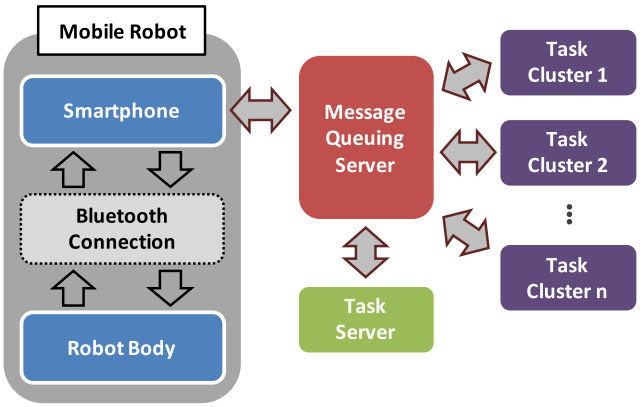


**FIGURE 1.** Schematic diagram of the overall system.

The robot communicates data, such as sensor and movement data, with a smartphone through Bluetooth connection. A smartphone, as a bridge between the mobile robot and a main personal computer (PC), sends sensor readings generated by the robot to the task server via the Message Queuing Server (MQS) so that a main processor in the task server performs complex computations (e.g., the particle filters). If necessary, the task server distributes tasks to task clusters that act as sub-processors. All the computation results from the task clusters are collected by the task server. To determine a proper course of action (e.g., direction of rotation or distance to advance), the final output from the task server is sent to the smartphone, which generates the movement values (e.g., actuator control signals). Finally, these signals are transferred to the robot via the Bluetooth connection.

### A. HARDWARE DESIGN

Hardware components in the mobile robot, including the chassis, a microcontroller, a smartphone, a Bluetooth module, and four ultrasonic sensors with a servo motor, are presented with design details in this section.

A Rover 5 Robot Platform was used for the robot chassis. The Rover 5 is driven by a motor controller board. An Arduino MEGA was used for the microcontroller, which handles the four ultrasonic sensors and servo motor. A smartphone was used to provide communication between the microcontroller and the main PC as a bridge and to act as a sensor that aids navigation. Four ultrasonic sensors were used to measure distances for the particle filters and to detect obstacles for navigation of the mobile platform. Bluetooth was used for serial communication between the Arduino MEGA and the smartphone (see Table 1 for details). Figs. 2 and 3 show the mobile robot design and photographs of the implemented mobile robot, respectively.

Fig. 4 shows the overall schematic diagram of the hardware in the mobile robot. The Rover 5 is a small robotic platform with four independent DC motors and two encoders.

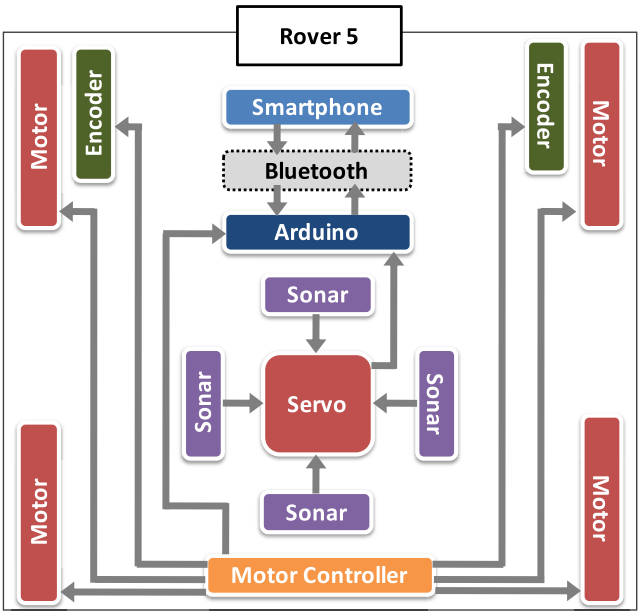


**FIGURE 2.** Design of the mobile robot.

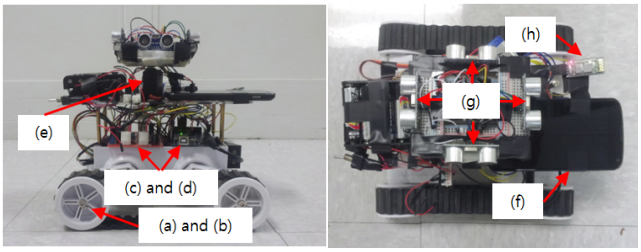


**FIGURE 3.** Lateral view (left) and top view (right) of the mobile robot. (a) Rover 5 with (b) motor encoder, (c) motor controller board (ROB-11593), (d) Arduino MEGA, (e) servo motor, (f) smartphone, (g) ultrasonic sensors, and (h) Bluetooth module.

**TABLE 1.** Hardware specifications.

| Item | Manufacturer | Model |
|---|---|---|
| Robot chassis (Rover 5) | Dagu Electronic | ROB-10336 [19] |
| Microcontroller | Arduino | Arduino MEGA [16] |
| Motor controller | Dagu Electronics | ROB-11593 [20] |
| Motor encoder | Solutions Cubed | 333 CPR [21] |
| Smartphone | LG Electronics | Nexus 4 [22] |
| Ultrasonic sensor | Elecfreaks | HC-SR04 [23] |
| Bluetooth module | Atomic Market | HC-06 [24] |
| Servo motor | Elmwood Electronics | SG-5010 |

Four motors are connected to the output channels of the motor controller, respectively. The direction (DIR) and pulse width modulation (PWM) pins of the motor controller are connected to digital pins on the Arduino MEGA. The direction and the speed of the motors were determined from the input directions with a value ranging from 0 to 1 (i.e., Low and High) and PWM signals ranging from 0 to 255 from the Arduino MEGA. Each encoder was connected to the input channel A and B pins of the motor controller. The Arduino MEGA can obtain the output values from the encoder from the encoder output channel (A and B pins on the motor controller). Both encoders calculate the left and right wheel revolution using an XOR logic operation. Therefore, the mobile robot can be moved as much as required by using the wheel revolution.

An ultrasonic sensor was used to measure distance. The sensor's trigger and echo pins were connected to the digital pins of the Arduino MEGA and the sensor uses the time difference between outgoing and incoming acoustic signals to calculate distance. The measurement angle of one ultrasonic sensor is approximately 30°. To develop a compact mobile robot, instead of using twelve sensors to cover 360°, we used a servo motor with four ultrasonic sensors mounted at 90° intervals. As a result, all directions can be scanned by rotating the servo motor from 0° to 90° with 30° intervals with a much smaller number of sensors.

Nowadays, various high quality, high performance sensors are integrated in most smartphones. The sensors in the Nexus 4 include a proximity sensor, gyroscope, compass, ambient light sensor, accelerometer, temperature sensor, and pressure sensor. In this work, the smartphone mounted on the mobile robot was used as a sensor for navigation, as a processor for complex computations, and as a bridge to communicate data between the Arduino MEGA and the MQS. A Bluetooth module was connected to the Arduino MEGA to communicate serial data between the Arduino MEGA and the smartphone, such as rotation, distance, and motion data.

### B. SOFTWARE DESIGN

Fig. 5 shows the overall design of the software for the mobile robot. In this work, there are two types of the applications in the Android smartphone. One application, called Scripting Layer for Android (SL4A) [25], was used to execute the main code for navigation and localization of the mobile robot on the smartphone. Most microcontrollers in a small mobile robot cannot afford to calculate complex algorithms (e.g., the particle filters [26]) for navigation and localization. One method is to use a smartphone with a better processor than a small microcontroller. In this work, the main code to navigate the mobile robot was written in Python. The SL4A application was used to interpret Python scripts. The smartphone communicates the sensing and motion data with the Arduino MEGA through the Bluetooth module. The Arduino MEGA in the mobile robot provides simple processing (e.g., operating motors and measuring distances). The Arduino is programmed in C.

The other application, called Bridge, was used to communicate data between the mobile robot and the MQS, and to rotate the mobile robot using sensors in the smartphone. In this application, a geomagnetic sensor and an accelerometer in the smartphone were used to rotate the mobile robot [27]. When the smartphone with SL4A was utilized as a main processor, it is sometimes impractical to process much more complex algorithms, although it has a better processor compared to the microcontroller in the robot. Computers with higher performance can be used to solve this problem. A RabbitMQ [28], which is one of the advanced

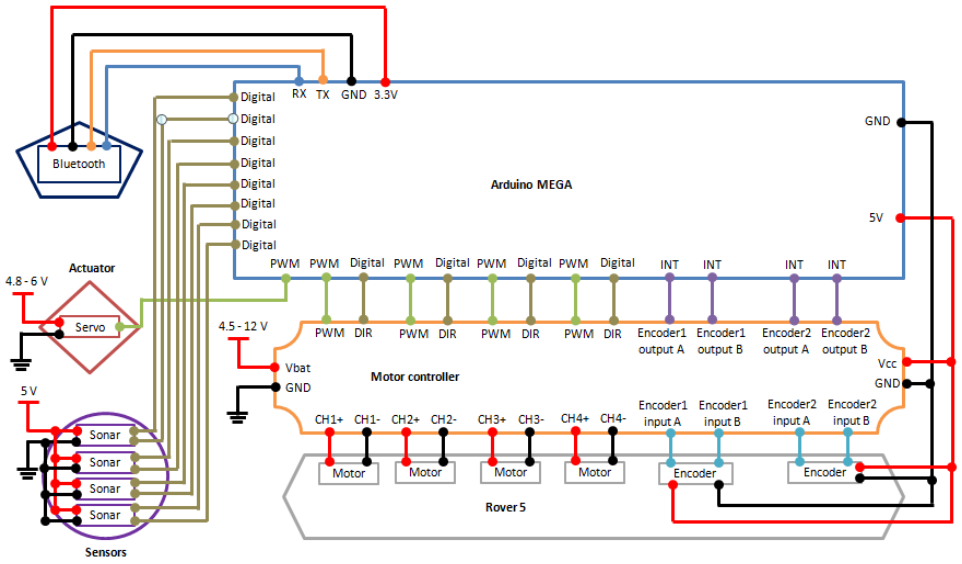


**FIGURE 4.** Hardware schematic diagram of the mobile robot.

message queuing protocols (AMQPs) [29], was used for communication between the computers and the smartphone. The smartphone and the task server send and receive data, such as distance and motion data, through the MQS. Even after distributing the computational burden, task clusters can be added if significant computation time is required. All software, including the Python and Arduino code for the mobile robot, is available at [30].

## III. EXPERIMENTAL RESULTS

### A. MOTION PLANNING

The use of the encoders in the Rover 5 creates inherent problems in the mobile robotic platform in some situations where surface flatness, friction between wheels and ground, and remaining battery power affect the accuracy of the encoders. These factors create uncertainties in the control of the mobile robots. In particular, the uncertainties become problematic when the mobile robot is rotated with low-cost encoders. To solve these innate problems, orientation sensor readings from a smartphone were used to change direction. Consequently, the mobile robot can reduce errors in consecutive movements [27]. The orientation sensor in the smartphone obtains data using a geomagnetic sensor combined with an accelerometer. Using these two sensors, the smartphone generates orientation values in a three-dimensional space. Then, these rotation direction data are sent to the Arduino MEGA through Bluetooth communication. The mobile robot begins to rotate until it receives a stop signal from the smartphone.

#### 1) PATH PLANNING

We used the A* algorithm [31], [32], which calculates the shortest path in a grid-based map, to navigate the mobile robot. The A* algorithm is a combination of Dijkstras algorithm [33], which finds the shortest path, and the greedy best-first-search algorithm [34], which uses heuristics to guide a mobile robot. Taking the advantages of both algorithms, as shown in Fig. 6, the A* algorithm requires less computational time than Dijkstras's algorithm and determines a better path than the greedy best-first-search algorithm.

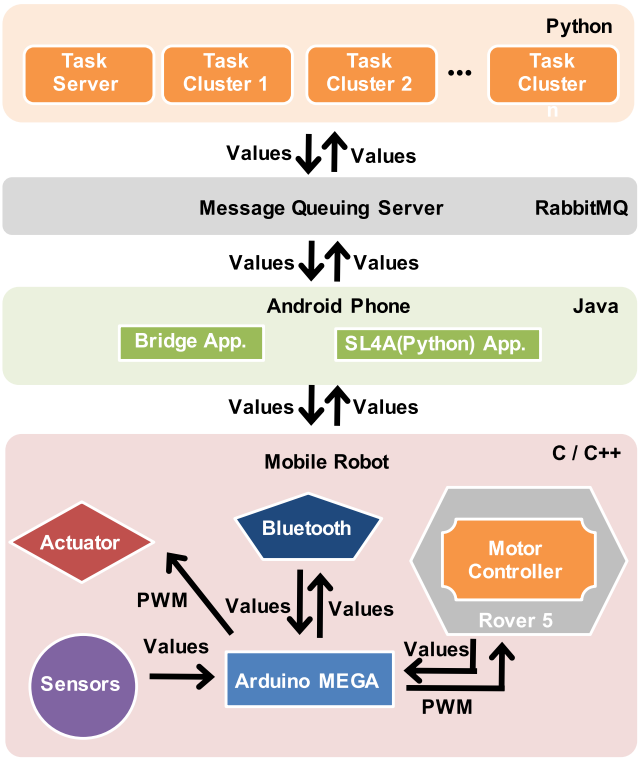


**FIGURE 5.** Software schematic diagram of the mobile robot.

The A* algorithm uses heuristics ($h$) to search for a path considered that leads to a goal point. Using the cost ($g$) from

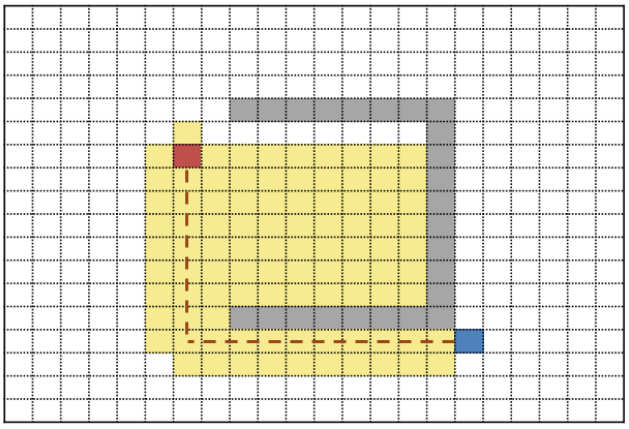

**FIGURE 6.** The A* algorithm with obstacles (gray color), calculated grids (yellow color), and uncalculated grids (white color). The brown dotted line shows the calculated path.

the departure point to the current point, it allows the mobile robot to find the best path without having to search the entire map, and to make the algorithm much faster. Each point will have 3 values $f$, $g$, and $h$ associated with it, as shown in (1).

$$f = g + h. \tag{1}$$

If $f$ of the next position is larger than that of the current position, it is excluded to reduce the calculation time. The algorithm examines the position with the lowest $f$ value during each iteration through the main loop.

### 2) SMOOTH PATH

Once the shortest path is calculated, this path is smoothed with a smoothing algorithm [35]. Let $x_0$, $x_1$, and $x_2$ be three sequential positions. The smoothing algorithm iteratively calculates a new path point $y_1$ until the variation in $y_1$ approaches 0. This process is shown in (2).

$$\begin{aligned} y_1 &= y_1 + \alpha(x_0 + x_2 - 2y_1), \\ y_1 &= y_1 + \beta(x_1 - y_1) \end{aligned} \tag{2}$$

where $\alpha$ and $\beta$ control the level of smoothness. Note that the initial value of $y_1$ is the value of $x_1$ and the smoothing algorithm iteratively applies the updating equations to the waypoint on the mobile robot path. The extent of the smoothed path depends on the movement accuracy of the mobile robot because a mobile robot with the incorrect encoder may encounter and collide something, such as wall or obstacles. From Fig. 7, one can see that the smoothing algorithm can reduce the total travel distance (Original path: 10.05 m, Smooth path: 8.37 m, and Smoother path: 7.59 m).

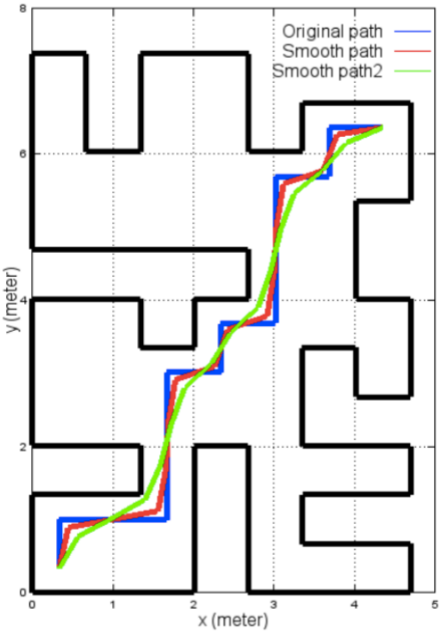


**FIGURE 7.** Smooth path planning with the original path (blue line), smooth path (red line, $\alpha = 0.4$, $\beta = 0.1$), and smoother path (green line, $\alpha = 0.1$, $\beta = 0.1$).

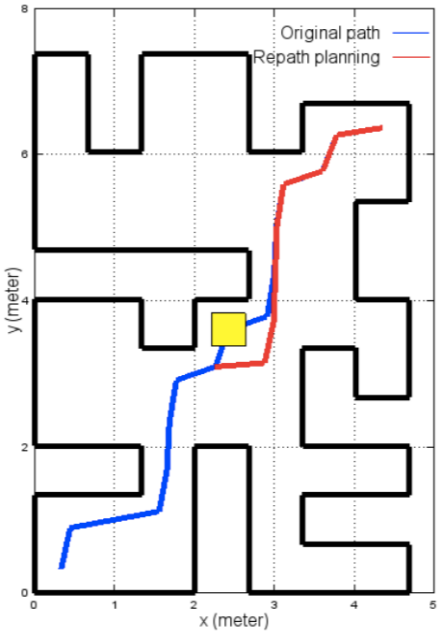


**FIGURE 8.** Obstacle avoidance and path re-planning in the presence of an obstacle (yellow square) with the original path (blue line) and new path (red line).

### 3) PATH RE-PLANNING

An autonomous mobile robot should cope with unexpected situations because the real world is not a static environment. The mobile robot in this work navigates dynamic environments where people or unexpected objects are present. The ultrasonic sensors measure the distance from the current position to the next position before the mobile robot moves to next position. If the distance measured by the ultrasonic sensors is shorter than distance from the current position to the next position, the mobile robot is aware that there are some objects. As a result, the path of our mobile robot is re-searched to avoid obstacles, as shown in Fig. 8.

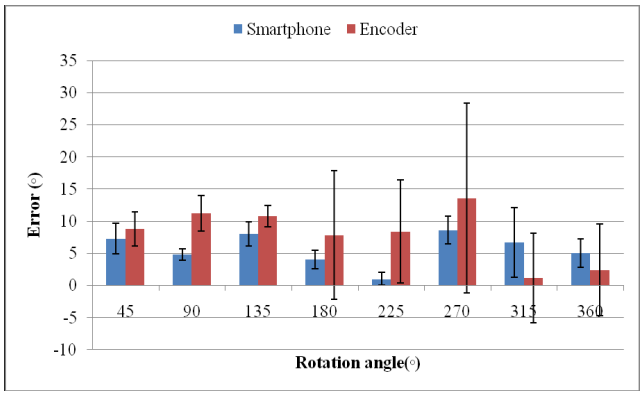


**FIGURE 9.** Comparison of the rotation performance between orientation sensors in a smartphone and an encoder.

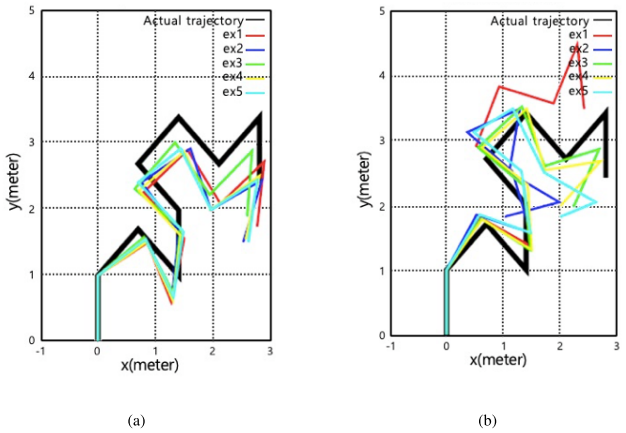


**FIGURE 10.** Comparison of the path trajectories between orientation sensors in a smartphone and an encoder. (a) Smartphone. (b) Encoder.

#### 4) RESULTS

Figs. 9 and 10 show the rotation and navigation results for a mobile robot through the use of an orientation sensor in a smartphone and an encoder, respectively. Comparisons of experimental results for the robot's rotation using two methods are presented in Fig. 9. The bars in the figure show the average errors from five trials. This figure shows that errors from the smartphone sensor are smaller than errors from the encoder. In addition, the fact that the smaller standard deviation provided by the smartphone sensor compared to the encoder shows that the smartphone sensor provides consistent outputs.

Fig. 10 shows path trajectories as the mobile robot navigates using two methods. The size of the area is 3 m × 5 m. The mobile robot reaches its destination through eight rotations from the starting point. The black line shows an actual reference trajectory. The experimental results show that the sensor in a smartphone produces trajectories that are closer to the reference trajectory than those obtained with the encoder. Furthermore, the experimental results in Fig. 10 (a) show that five destination points are closely distributed, whereas they are widely distributed in Fig. 10 (b). Therefore, using the smartphone sensor can be considered as being more accurate than using the encoder to navigate the robot.

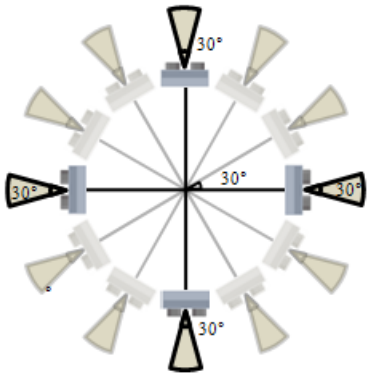


**FIGURE 11.** Four ultrasonic sensors with one servo motor for particle filters. Each sensor covers from 0° to 90° in 30° intervals.

### B. LOCALIZATION

Localization is one important problem related to autonomous mobile robots. In particular, in the proliferation of navigation technologies, the location information and the pose information of a mobile robot in a given environment are indispensable. Researchers have been solving localization problems of autonomous mobile robots introducing many methods, such as using a landmark and external sensors [36]–[39]. In particular, owing to the lack of a global positioning system (GPS) or other reliable technologies, indoor robot localization is considered to be a challenging problem in mobile robotics. To solve these problems, we investigate a probabilistic approach, known as the particle filters [1], with four ultrasonic sensors and one servo motor.

#### 1) PARTICLE FILTERS

In the past decades, the particle filters or sequential Monte Carlo (SMC) have been used for localization of a mobile robot [26]. The particle filters estimate the posterior probabilities of the unobservable state variables from sensor measurements. This consists of three phases; movement update, measurement update, and resampling. These three methods are repeated every time the mobile robot moves from the current position to the next position. Each particle is assigned weights, which are the likelihood of the statistics approximated by the particle filter. After motion and measurement are updated, the particles are resampled by its own weight value. Repeated resampling causes all particles to converge into one cluster. As a result, the mobile robot can predict its own position based on the locations of the particles. All complex processes, such as calculating the number of particles, are handled by the smartphone in the mobile robot. Then, the smartphone sends commands such as the measuring distance and motion of the robot to the Arduino MEGA through the Bluetooth connection.

#### 2) SENSOR DEPLOYMENT FOR MEASUREMENT UPDATE OF PARTICLE FILTERS

Four ultrasonic sensors are attached to a servo motor for measurement update with the particle filters. The servo motor

rotates from 0 to 90° in 30° intervals, as shown in Fig. 11. Thereby, it is possible to cover 360°. This reduces the required scanning time as well as the number of required sensors. Note that the scanning time will increase if the number of sensors is decreased, and vice versa. In this work, 30° was chosen to reduce interference from neighboring ultrasonic sensors. Because the servo motor rotates in 30° intervals, twelve distance values from the four ultrasonic sensors were obtained, where each sensor reads three times per scanning session.

When the ultrasonic sensor does not receive its own reflected sound signal after emitting the sonar beam, the sensor reading is absolutely not reliable. In this case, the wrong data were excluded from the measurement data. The reason is that these incorrect readings will disturb the particle resampling process because the weight of a particle is calculated based on the difference between the actual sensor readings and the map data.

### 3) FEASIBILITY OF ULTRASONIC RANGE SENSORS

The sensor was tested in various environmental situations to verify whether the sensor reading is reliable enough to be used in measurements for the localization algorithm. An ultrasonic sensor, called HC-SR04 [40], was chosen because the measurement accuracy fits the requirements in this work. The sensor distances ranged from 2 to 400 cm. The sonar beam angle of the ultrasonic sensor was approximately 30°.

Ultrasonic range sensors use the time difference between an incident sonar beam and the reflected beam to calculate distance. If the angle of reflection with respect to the normal line from the reflector is too large, then the sound wave receiver in the sensor fails to receive the reflected sound wave. In other cases, reflected sonar beams can arrive at the sensor after multiple reflections. These cause incorrect distance readings.

Based on this concern, the feasibility of using ultrasonic sensors in localization with the particle filters was tested. Walls with five different shapes were designed for the test, as shown in Fig. 12. The distances were measured from 0 to 180° in 2° intervals. The measurements will be compared to reference values acquired with a parallax laser range finder [41]. The experiments for distance measurements were conducted to validate the number of ultrasonic sensors required to read the correct values in 5 different environments, as shown in Fig. 12. The results shown in Fig. 13 reveal that the ultrasonic sensors occasionally read incorrect values owing to reflection of outgoing sound signals. However, this problem does not affect localization of the mobile robot. Hence, one can conclude that the sensor readings with the ultrasonic sensor have sufficient accuracy to localize the robot using the particle filters so that the mobile robot can be successfully maneuvered.

### 4) LOCALIZATION OF THE MOBILE ROBOT

The mobile robot cannot identify its own location with high certainty during the early steps because the probability of determining the location is evenly spread throughout the state-space. The variances of particles with position $(x, y)$ and heading $(\theta)$ are compared with a certain threshold to localize the mobile robot. When the variance of particles is smaller than the threshold, the mobile robot adjusts its position to the

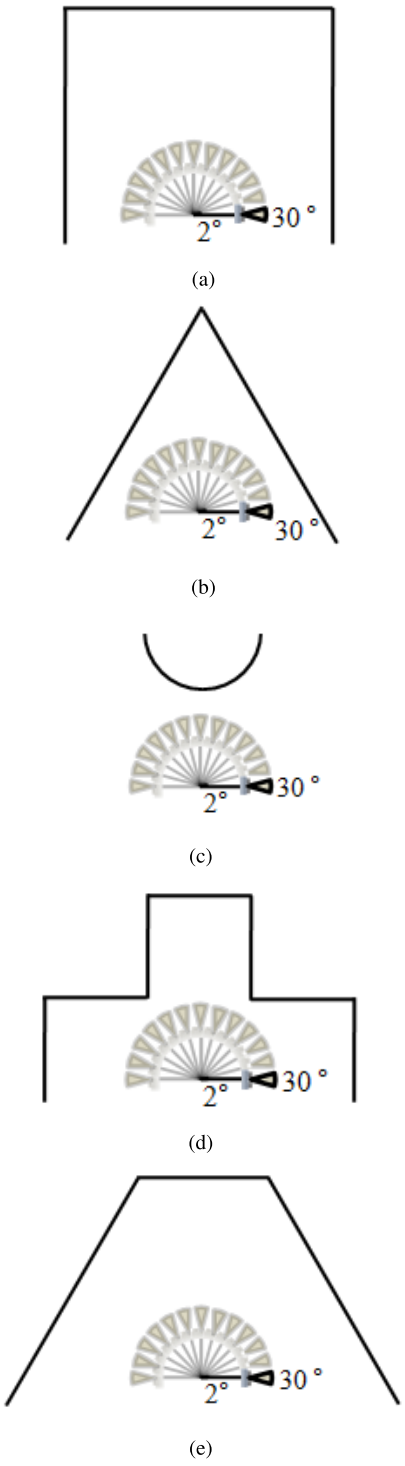


**FIGURE 12.** Five different experimental environments to test the feasibility of the ultrasonic sensor. (a) Environment 1. (b) Environment 2. (c) Environment 3. (d) Environment 4. (e) Environment 5.

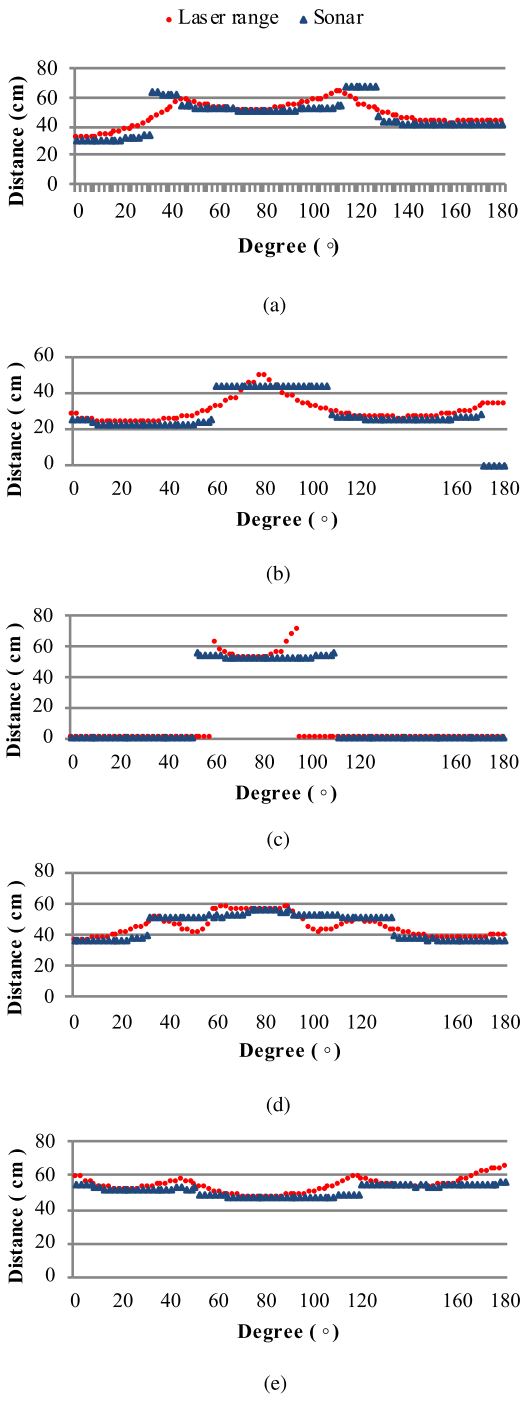


**FIGURE 13.** Distance measurements from 0° to 180° in 2° intervals with the ultrasonic sensor (blue triangles) and the laser range finder (red circles) for five different experimental environments in Fig. 12. (a) Environment 1. (b) Environment 2. (c) Environment 3. (d) Environment 4. (e) Environment 5.

position predicted by the particles. The position of the mobile robot changes to a predicted position every four cycles, where each cycle includes movement, measurement, and resampling of all particles. If the error in movement or measurement caused by the encoder or sensors increases, the number of cycles should be reduced to avoid unexpected situations,

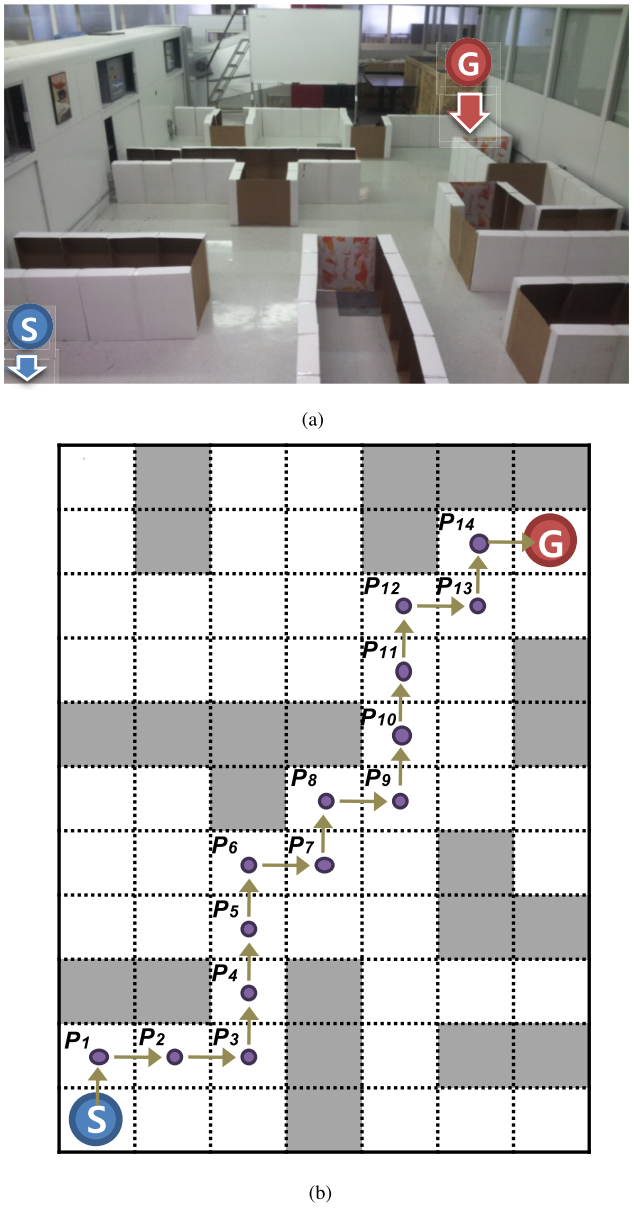


**FIGURE 14.** Map for navigating and localizing the mobile robot. (a) Actual map. (b) Grid map.

such as collision with a wall or other obstacles. An actual environment and its two-dimensional occupancy grid map generated with the particle filter routine in this work are shown in Fig. 14. The size of the environment is 4.69 m × 7.37 m. The space is divided into a 0.67 m × 0.67 m grid, which requires 7 grids in each row and 11 grids in each column. To navigate the mobile robot, a starting point and a goal point are given to the mobile robot.

Fig. 15 shows that the mobile robot smoothly navigates to the goal point with the shortest path and localizes its own position using the particle filter. The mobile robot localizes its own position to an ideal position when the variances of particles are smaller than a threshold (set to 0.1 in this study). This means that most particles congregate at one cluster.

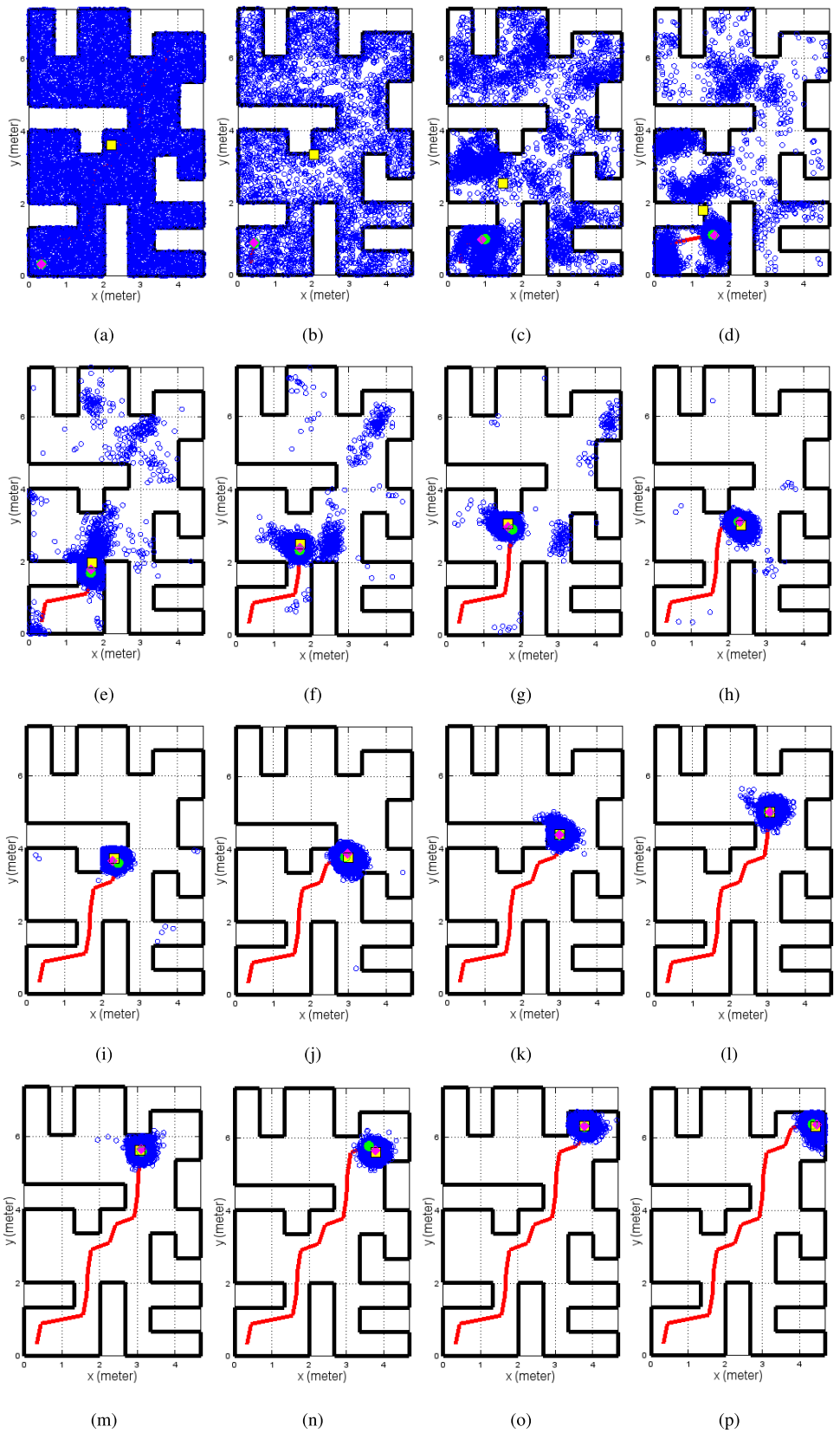


**FIGURE 15.** Experimental localization results using random particle filters with the actual trajectory (red line), particles (blue circles), ideal position (green circle), actual robot position (magenta diamond), and robot position predicted from particles (yellow square). (a) $P_0$-Start. (b) $P_1$. (c) $P_2$. (d) $P_3$. (e) $P_4$. (f) $P_5$. (g) $P_6$. (h) $P_7$. (i) $P_8$. (j) $P_9$. (k) $P_{10}$. (l) $P_{11}$. (m) $P_{12}$. (n) $P_{13}$ (o) $P_{14}$. (p) $P_{15}$.

**TABLE 2.** LG nexus 4 and mac mini resolution.

| | LG Nexus 4 E 960 | Mac Mini (Mid 2010) |
|---|---|---|
| OS | Android OS, v4.4.2 (KitKat) | Mac OS |
| CPU | Quad-core 1.5 GHz Krait | 2.4 GHz Intel Core 2 Duo |
| RAM | 2 GB | 2 GB |
| Storage | 16 GB | 500 GB |
| Weight | 139 g (0.306 pounds) | 1370 g (3.0 pounds) |

At first, a number of particles are spread in the map, as shown in Fig. 15 (a). Each particle has a different weight after motion and measurement update of the ultrasonic sensors and movements of the mobile robot. New particles are subsequently resampled with weights. All particles converge to one cluster after multiple iterations of motion, measurement, and resampling updates, as shown in Fig. 15 (p). Consequently, the mobile robot can safely reach the goal position with continuous localization.

### C. NETWORK

When the smartphone is used as a computational processor, the mobile robot spends a lot of time performing complex computations, such as the particle filters. Consequently, the mobile robot seems to have stopped for a long time. To solve this problem, desktop computers with higher performance than a smartphone are used as computational processors. In this case, the smartphone acts as a bridge to communicate between the mobile robot and the desktop computers using RabbitMQ. A smartphone and Mac mini were used to compare the calculation times between the devices, as listed in Table 2.

Experiments were performed in five different environments to compare the calculation time for the particle filters algorithm as follows:

- Using a smartphone as a computing resource with SL4A;
- Using a smartphone as a broker and one task server as a main processor with RabbitMQ;
- Using a smartphone as a broker, one task server, and two task clusters as sub-processors;
- Using a smartphone as a broker, one task server, and four task clusters as sub-processors;
- Using a smartphone as a broker, one task server, and eight task clusters as sub-processors.

10,000 particles were used in each environment. Although a large number of particles were intentionally used to see the difference clearly in heavy calculation tasks, using 1,000 particles would be a good choice in such a real environment.

In the second experiment, one smartphone and one task server were utilized as a broker and a main processor, respectively. The main processor runs the complex computations, such as the movement and particle measurement updates, and sends simple commands to the MQS, such as the movement of a mobile robot or the signal for measuring distance. Then, the smartphone mounted on a mobile robot receives this command from the MQS and passes it to the Arduino through Bluetooth such that the mobile robot can move or measure

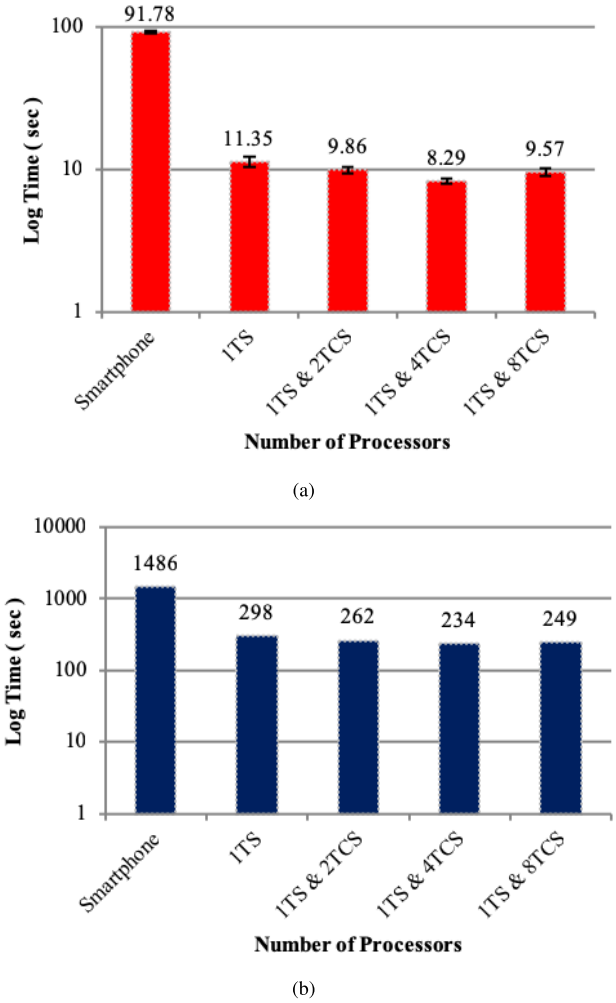


**FIGURE 16.** Comparison of the average step time and total travel time in five cases. (a) Average step time. (b) Total travel time.

distance. Afterward, the Arduino sends the distance data and the movement completion signal to the smartphone, which passes them to the MQS. Finally, the main processor in the task server receives data from the MQS and starts executing complex computations. This cycle is repeated until the mobile robot reaches the destination.

In the third to fifth experiments, task clusters were added to aid computation. For instance, when using two task clusters, 5000 particles were assigned to each task cluster, 2500 particles were assigned in the fourth case, and 1250 particles were assigned in the fifth case. After receiving particles, the sub-processors conduct measurement and movement update of the particles. Meanwhile, the task server sends data to the smartphone as mentioned in the second experiment. Updated particles are sent to the main processor through the MQS. Once the task server receives all values of the particles, the main processor starts resampling particles. This cycle is repeated until the mobile robot reaches its destination.

The calculation time results in the five different cases are shown in Fig. 16. One calculation step is defined as one cycle

involving movement, measurement, and resampling update as the mobile robot navigates. When only the smartphone is utilized as the main processor, the calculation time is significantly longer than that with the other methods. For example, the average step time in using a smartphone is 80.41s longer than using one task server, which means the task server can process nearly eight steps while the smartphone processes one step. Note that the computation time decreases as the number of task clusters increases. However, the calculation time with eight task clusters was longer than that with four task clusters. The reason is that network communication requires more time than the complex computations. Hence, one can see that the calculation time depends on the number of particles and task clusters. Thus, according to the circumstances, the number of task clusters should be carefully chosen.

The following sites show the calculation time results when one smartphone, one PC, three PCs, five PCs, and nine PCs were used, respectively.

- with a smartphone and without MQS: https://www.youtube.com/watch?v=HBgq8bRK140
- with one PC (one task server): https://www.youtube.com/watch?v=ZZd1gfv_ANw
- with three PCs (one task server and two task clusters): https://www.youtube.com/watch?v=qQuSzpsTS_0
- with five PCs (one task server and four task clusters): https://www.youtube.com/watch?v=HwsYpSrMy90
- with nine PCs (one task server and eight task clusters): https://www.youtube.com/watch?v=fgEMLI0LVpw

The following site shows an autonomous mobile robot with 1000 particles.

- A networked smartphone-based mobile robot: https://www.youtube.com/watch?v=GKOWgfqClsA

## IV. CONCLUSIONS

A networked mobile robot with a smartphone-based robotic platform was presented in this paper. Rotating the ultrasonic range sensor package is quite cost-effective in that sufficient sensor readings can be obtained for navigating the robot and the number of required sensors can be reduced. It goes without saying that the scanning time also decreases. We showed that the orientation sensor in a smartphone provided better performance for rotating the robot than using an encoder. This approach demonstrates robustness in conditions where the friction of a surface is inconsistent. Furthermore, a load on the mobile robot can be distributed by mounting a smartphone on it. We propose using a smartphone as a bridge and PCs as a robots' brain, in which all heavy-duty computations are executed through the MQS. The practicality and effectiveness of the proposed approaches were thoroughly verified in multiple experiments with the corresponding real mobile robot platform. The concept and design of the proposed network robotic platform in this work can be widely adopted in indoor autonomous mobile robots.

## REFERENCES


[1] S. Thrun, "Particle filters in robotics," in *Proc. Uncertainty Artif. Intell.*, Edmonton, AB, Canada, Aug. 2002, pp. 511–518.

[2] G. T. McKee and P. S. Schenker, "Networked robotics," *Proc. SPIE*, vol. 4196, pp. 197–210, Oct. 2000.

[3] G. T. McKee, "What is networked robotics?" in *Informatics in Control Automation and Robotics*, J. A. Cetto, J.-L. Ferrier, J. M. C. D. Pereira, and J. Filipe, Eds. Berlin, Germany: Springer, 2008, pp. 35–45.

[4] B. Koken, "Cloud robotics platforms," *Interdiscipl. Description Complex Syst.*, vol. 13, no. 1, pp. 26–33, Jan. 2015.

[5] E. Cardozo, E. Guimarães, L. Rocha, R. Souza, F. Paolieri, and F. Pinho, "A platform for networked robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Taipei, Taiwan, Oct. 2010, pp. 1000–1005.

[6] Y.-G. Ha, J.-C. Sohn, and Y.-J. Cho, "Service-oriented integration of networked robots with ubiquitous sensors and devices using the semantic Web services technology," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Edmonton, AB, Canada, Aug. 2005, pp. 3947–3952.

[7] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, "Cloud-based robot grasping with the Google object recognition engine," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Karlsruhe, Germany, May 2013, pp. 4263–4270.

[8] M. Waibel *et al.*, "RoboEarth," *IEEE Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, Jun. 2011.

[9] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: Architecture, challenges and applications," *IEEE Netw.*, vol. 26, no. 3, pp. 21–28, May 2012.

[10] V. Kumar, D. Rus, and G. S. Sukhatme, "Networked robots," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer, 2008, pp. 943–958.

[11] Y.-H. Seo, S.-S. Kwak, and T.-K. Yang, "Mobile robot control using smart phone and its performance evaluation," in *Advanced Communication and Networking*, T.-H. Kim, H. Adeli, R. J. Robles, and M. Balitanas, Eds. Berlin, Germany: Springer, 2011, pp. 362–369.

[12] S. J. Lee, J. Lim, G. Tewolde, and J. Kwon, "Autonomous tour guide robot by using ultrasonic range sensors and QR code recognition in indoor environment," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, Milwaukee, WI, USA, Jun. 2014, pp. 410–415.

[13] Y.-S. Chiou, F. Tsai, S.-C. Yeh, and W.-H. Hsu, "An IMU-based positioning system using QR-code assisting for indoor navigation," in *Computer Science and Its Applications*, S.-S. Yeo, Y. Pan, Y. S. Lee, and H. B. Chang, Eds. Dordrecht, The Netherlands: Springer, 2012, pp. 655–665.

[14] H. Jung, Y. Kim, and D. H. Kim, "Communication quality analysis for remote control of a differential drive robot based on iPhone interface," in *Convergence and Hybrid Information Technology*, G. Lee, D. Howard, and D. Slezak, Eds. Berlin, Germany: Springer, 2011, pp. 278–285.

[15] N. Oros and J. L. Krichmar, "Smartphone based robotics: Powerful, flexible and inexpensive robots for hobbyists, educators, students and researchers," Center Embedded Comput. Syst., Univ. California, Irvine, Irvine, CA, USA, Tech. Rep. 13-16, Nov. 2013.

[16] *Arduino*. Accessed: 2016. [Online]. Available: http://www.arduino.cc/

[17] *Raspberry Pi*. Accessed: 2016. [Online]. Available: http://www.raspberrypi.org/

[18] *Beagle Board*. Accessed: 2016. [Online]. Available: http://beagleboard.org/

[19] *Rover5*. Accessed: 2016. [Online]. Available: https://www.sparkfun.com/datasheets/Robotics/Rover%205%20Introduction.pdf

[20] *Rob-11593 Motor Controller*. Accessed: 2016. [Online]. Available: http://cdn.sparkfun.com/datasheets/Robotics/4%20Channel%20 instruction%20manual.pdf

[21] *333 CPR Quadrature Encoder*. Accessed: 2016. [Online]. Available: http://letsmakerobots.com/

[22] *LG Nexus 4 Smartphone*. Accessed: 2017. [Online]. Available: http://www.gsmarena.com/lg_nexus_4_e960-5048.php

[23] *Product User's Manual—HC-SR04 Ultrasonic sensor*, Cytron Technol., Malaysia, 2013.

[24] *HC-06 Bluetooth Module*. Accessed: 2017. [Online]. Available: https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf

[25] *SL4A (Scripting Layer for Android)*. Accessed: 2017. [Online]. Available: https://github.com/damonkohler/sl4a

[26] I. M. Rekleitis, "A particle filter tutorial for mobile robot localization," Centre for Intell. Mach., McGill Univ., Montreal, QC, Canada, Tech. Rep. TR-CIM-04-02, Jan. 2004.

[27] J. Lim, S. J. Lee, G. Tewolde, and J. Kwon, "Ultrasonic-sensor deployment strategies and use of smartphone sensors for mobile robot navigation in indoor environment," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, Milwaukee, WI, USA, Jun. 2014, pp. 593–598.
[28] *RabbitMQ*. Accessed: 2018. [Online]. Available: http://www.rabbitmq.com/
[29] *AMQP (Advanced Message Queuing Protocol)*. Accessed: 2018. [Online]. Available: http://www.amqp.org/
[30] *Network Mobile Robot Using a Smartphone Based Robotics Platform*. Accessed: 2015. [Online]. Available: https://github.com/JongilLim
[31] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.
[32] *A* Algorithms*. Accessed: 2017. [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming/
[33] M. Yan. *Dijkstra's Algorithm*. Accessed: 2017. [Online]. Available: http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf
[34] *Greedy Best-First-Search Algorithm*. Accessed: 2017. [Online]. Available: http://en.wikipedia.org/wiki/Best-first_search/
[35] *Artificial Intelligence for Robotics*. Accessed: 2017. [Online]. Available: https://www.udacity.com/course/artificial-intelligence-for-robotics–cs373/
[36] B.-S. Choi, J.-W. Lee, J.-J. Lee, and K.-T. Park, "A hierarchical algorithm for indoor mobile robot localization using RFID sensor fusion," *IEEE Trans. Ind. Electron.*, vol. 58, no. 6, pp. 2226–2235, Jun. 2011.
[37] E. McCann, M. Medvedev, D. J. Brooks, and K. Saenko, "'Off the grid': Self-contained landmarks for improved indoor probabilistic localization," in *Proc. IEEE Conf. Technol. Pract. Robot Appl. (TePRA)*, Woburn, MA, USA, Apr. 2013, pp. 1–6.
[38] N. Ganganath and H. Leung, "Mobile robot localization using odometry and Kinect sensor," in *Proc. IEEE Int. Conf. Emerg. Signal Process. Appl.*, Las Vegas, NV, USA, Jan. 2012, pp. 91–94.
[39] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Saint Paul, MN, USA, May 2012, pp. 1697–1702.
[40] *Product User's Manual—HC-SR04 Ultrasonic Sensor*. Accessed: 2016. [Online]. Available: https://docs.google.com/document/d/1YyZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit
[41] *Parallax Laser Range Finder*. Accessed: 2016. [Online]. Available: https://www.parallax.com/sites/default/files/downloads/28044-Laser-Range-Finder-Guide-v2.0.pdf

**JONGIL LIM** received the B.S. degree in information and communication engineering from Wonkwang University, Jeonbuk, South Korea, in 2013, and the M.S. degree in electrical and computer engineering from Kettering University, Flint, MI, USA, in 2015. He is currently with the Software Reliability Technology Division, Agency for Defense Development, Daejeon, South Korea.

**GIRMA S. TEWOLDE** (M'00) received the B.Sc. degree in electrical engineering from Addis Ababa University, Addis Ababa, Ethiopia, in 1992, the M.Eng.Sc. degree in computer science and engineering from the University of New South Wales, Sydney, Australia, in 1995, and the Ph.D. degree in systems engineering, with a focus in computer engineering, from Oakland University, Rochester, MI, USA, in 2008. From 1995 to 2000, he was a Lecturer with Asmara University, Eritrea. He conducted a graduate study at the Royal Institute of Technology, Stockholm, from 2000 to 2001. In 2002, he joined Kettering University, as a Lecturer. He is currently an Associate Professor in computer engineering with the Electrical and Computer Engineering Department, Kettering University, Flint, MI. His research interests include mobile robotics, indoor and outdoor positioning and navigation, autonomous and connected vehicles, and sensor networks.

**JAEROCK KWON** (S'06–M'09) received the B.S. and M.S. degrees from Hanyang University, Seoul, South Korea, in 1992 and 1994, respectively, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, TX, USA, in 2009. From 1994 to 2004, he worked for various companies, such as LG Electronics, SK Teletech, and Qualcomm Internet Service. Since 2009, he has been a Professor with the Department of Electrical and Computer Engineering, Kettering University, Flint, MI, USA. His research interests include mobile robotics, autonomous vehicles, and artificial intelligence using neural networks.

**SEYEONG CHOI** (S'04–M'08–SM'15) received the B.S. and M.S. degrees from Hanyang University, Seoul, South Korea, in 1996 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from Texas A&M University, College Station, TX, USA, in 2007. From 2007 to 2008, he was with TAMU at Qatar (TAMUQ), as a Postdoctoral Research Associate and a Research Consultant. Then, he was a Leader of Advanced Communication Technology (ACT) part of Wireless Advanced Technology (WAT) Group in LG Electronics. Since 2010, he has been a Professor with the Department of Information and Communication Engineering, Wonkwang University, South Korea. His research interests include wireless communications, MIMO fading channels, diversity techniques, and system performance evaluation.