An Open-Source Solution for Mobile Robot Based Environmental Sensing

J. Grönman*, J. Viljanen*, J. Vihervaara*, and M. Saari*
* Tampere University/Pori Campus, Pori, Finland
jere.gronman@tuni.fi

***Abstract* - Internet of Things-based devices are equipped with sensors and network connections to collect data and store this data to cloud services. This collected data allows operational decision-making processes to be based on a more accurate picture of the current state of physical environment than before. Existing data collection methods often rely on sensors in fixed locations to obtain environmental measurements. Such a solution is not very scalable in terms of the density of the measuring points. In a dynamic environment, the cost of maintaining such fixed-based solutions may also be high. A more flexible solution will be achieved by equipping an autonomous mobile robot with sensors. Mobile robots are increasingly being used for data-gathering in a wide range of environmental applications. However, a robotic solution can be expensive. In this paper, we present a low-cost solution based on open-source components. The solution utilizes a robot operating system, sensor and IoT-board based on open-source implementation. By implementing a pilot project, we concretize the potential of our approach to environmental monitoring. Potential application areas of this solution include the microclimate control of greenhouses and warehouses, for example.**

Keywords – environmental sensing; mobile robot; open-source

I. INTRODUCTION

Environmental sensing includes the processes and activities to characterize and monitor the quality of the environment. Recent advances in environmental sensing technologies have introduced a wide range of new environmental sensors with reduced cost and size. This opens new possibilities to put into service a wide variety of smart applications which improve the efficiency of production processes and human well-being.

The Internet of Things [1][2] connects environmental sensors to the Internet allowing interaction between the physical and digital world. The computer-based digital world includes cloud services, data analytics and other software-based intelligent services. This interaction enables real-time data collection from the physical environment and to store the collected data to cloud services. Therefore, environmental sensing with IoT approach enables a continuous view of the current state of the environment. The collected real-time data allows decision-making processes to be based on a more accurate picture of the current state of physical environment than before.

Existing environmental sensing methods often rely on sensors in fixed locations to obtain measurements. For example, Wireless Sensor Network have been utilized to implement such measurement solutions [3][4]. However, such solutions are not very scalable in terms of the density of the measuring points. In a dynamic environment, the cost of maintaining such fixed-based solutions may also be high. A more flexible solution will be achieved by equipping an autonomous mobile robot with sensors. Mobile robots are increasingly being used for data-gathering in a wide range of environmental applications, because robotics is seen as a promising tool to enhance environmental data collection [5].

Unfortunately, a fine robot-based implementation can sometimes be too expensive. Large-scale utilization of open-source components can offer a solution to this problem. Open-source software and components are often cost-competitive. In addition, open-source is a development model that has shown many other significant benefits in many application areas [6][7][8]. The functions contained in an open-source implementation are manageable. It can be ensured that the code contains only the functionalities required for implementation. Open-source applications can be developed with an iterative approach. This allows implementation to be modified during development to meet new requirements. The small-scale development process remains flexible enough to accept case-by-case changes. Open-source implementations are also often reliable because of the motivated team of experts involved in development and testing. Obviously, the open-source approach also has its drawbacks, as disclosed in the papers [6][7].

This paper presents a low-cost and flexible solution for environmental sensing. The implementation is based on open-source components. The solution utilizes a ROS robot operating system, RuuviTag sensor and Raspberry Pi IoT-board to achieve a low-cost solution. By implementing a pilot project, we concretize the potential of our approach to environmental sensing. Potential application areas of this solution include the microclimate control of greenhouses and warehouses, for example. Robots can also be used in environments where it is unsafe for humans to enter. Compact robots may also reach places that are inaccessible to humans.

The rest of the paper is organized as follows. Section 2 introduces the backgrounds techniques of our pilot project. It also briefly presents some related works.

Section 3 presents the pilot project. Section 4 presents the main conclusions of the project.

II. BACKGROUNDS

A. Components of the pilot project

The mobile robot used in the pilot project is Turtlebot 2. The robot moves by adjusting the speed of the drive wheels. The robot's movement and positioning utilize odometry, which analyzes the distance traveled by the drive wheels [9]. In addition, the robot has sensors to detect obstacles. It is powered by the battery. The battery is recharged at a docking station where the robot can dock independently when the battery charge level is too low. Turtlebot2 is provided with different kinds of sensors, including Microsoft Kinect 360 3D depth camera.

ROS (Robot Operating System) [10] is an open-source operating system developed for robot control and management. ROS is a meta operating system that runs on top of the Linux operating system. It has a modular structure and is supported by a large number of special software packages. Its role is to manage software packages and control communication between different processes. Software packages can be utilized in autonomous navigation and in the design of motion paths, for example. Its programming is not tied to a specific programming language.

Raspberry Pi is an open-source IoT board for providing a complete Linux computer at a very low cost [11]. It is a card-sized minicomputer that can operate also with the power of a battery. Raspberry Pi provides an interface for sensors and actuators through the general purpose I/O pins. Sensors can also be connected to it via wireless connections such as Bluetooth. For control purposes, it can be connected wirelessly via a WLAN port by using a secure shell (SSH) session, for example.

RuuviTag is an IoT-sensor including a circuit board that comes with a waterproof and breathable plastic housing. The circuit board contains a sensor for measuring ambient temperature, relative humidity and barometric pressure. In addition, the circuit board includes its own sensor for measuring acceleration. It is powered by a battery that is promised to function for a few years. For communication purposes, the circuit board includes a Bluetooth chip that enables communication with other devices, such as smartphones or Raspberry Pi. RuuviTag does not have its own IP address. If its data must be read via the Internet, it must be connected via Bluetooth to a Raspberry Pi-based Internet server, for example, RuuviTag is able to communicate with Raspberry Pi with the help of an open-source Python library. Above all, RuuviTag has an open-source firmware. This way it can be customized to suit the needs of the application area, as is done in the paper [12].

InfluxDB [13] is an open-source time series database that provides storage of automatic measurements of Ruuvitag parameters, including the results of calculations as well as assigning a timestamp to each measurement.

Grafana is a program that can be used to easily create graphical control screens [14]. Grafana is an open-source and free program. Grafana is designed for use with time series databases and it supports many different time series databases, such as InfluxDB. Grafana was originally made for monitoring data centers, but its use has spread to other industries.

B. Related studies

There are several studies on mobile robots [15][16]. Several studies also exist to deploy mobile robots for environmental sensing both indoor and outdoor. Here are some of them briefly outlined. However, a few studies focus on the research area from an open-source perspective.

The study [17] demonstrated the mobile platform in a laboratory experiment of measuring air-change effectiveness. By comparing the measurements from the mobile platform and those from a standard dense sensor network, they showed that the automated mobile sensing approach was able to determine the air-change effectiveness with high spatial granularity and accuracy. The paper [18] presets a study where three prototype robots were used to monitor pollution. The robots were capable to autonomously navigate in real urban environments. Different kinds of sensors were integrated into these robots for environmental sensing. Because the solution used sophisticated technology, it could not be considered a low-cost solution.

The study [19] presents a robotics platform for performing environmental monitoring in data centers. They present a platform based on the Robot Operating System, in which a mobile robot is able to autonomously navigate in a data center room for executing measurements at different locations. They observed that the robot avoided fixed and dynamic obstacles successfully. The paper [20] proposes a robot for human rescue operation in an environment which is unsuitable for any human intervention. The robot is capable of autonomous operation by being guided using sensors. The robot is mounted with sensors that help it navigate while operating autonomously. The Sensors also enables the robot to detect a human in need of rescue. The robot was designed for rescue operations by allowing a minimal threat to human life.

III. IMPLEMENTATION

Initially, the robotics solution was configured for a healthcare pilot. The pilot focused on an autonomous mobile robot that could assist nurses in their work. In this study, a modified version was made outlining a solution to various practical cases where a robot can operate when a human is not capable of those conditions, for example oxygen-free warehouses, hazard or hot/cold areas of the industry.

The goal was to create a robotics solution with an autonomously operating mobile robot capable of utilizing IoT devices and sensors. The solution supports browser-based implementation and the robot is remotely controllable. It is possible to connect different sensors to the unit, including temperature, humidity, motion or light sensors. The implementation follows open-source principles and the robotics solution utilizes software found

on the official ROS website. The user interface is implemented according to the current web design trends by using HTML5, CSS and JavaScript.

The robot platform used is the Turtlebot 2 mobile robot with Kinect Xbox 360 3D sensor. The mobile robot is controlled by a Raspberry Pi 3 Model B+ computer that has an open-source ROS operating system developed for robot control. The mobile robot operates in a wireless network, moves autonomously within a defined area, and can be tracked and controlled remotely. The robotics solution is also capable of utilizing the information transmitted by IoT devices and sensors. The user interface is designed to work in a web browser and can be accessed also from a touch screen mounted on the robot. Information content can be displayed in the user interface. RuuviTag is connected via Bluetooth to the Raspberry Pi and the collected data is transferred to the Influx database. The stored data can be visualized by the Grafana software. The Grafana software runs on a website, which is accessible to all devices of the same wireless LAN on a browser basis.

When configurating the robotic solution, a base map must first be created for the area. The map is created by using the RViz (ROS Visualization) software that uses the robot's sensor data to visualize the area. With the RViz software the user can monitor the robot's movements on the map in real time. By combining this with the live image produced by the camera, the new area can be mapped entirely from a remote device via a wireless network. Because the RViz software is constantly updating the map image, one option was to use the program on a VNC (Virtual Network Computing) connection.

ROS consists of processes represented by nodes that are programs that communicate with each other. The program can send or receive messages belonging to a specific topic. ROS is designed to be modular and the robot control system consists of several nodes, each with its own function. For example, one node controls the electric motor and another node controls the route.

There are many different example programs available for Turtlebot2 that can be used as explained above. The example programs can be seen in Mark Silliman's Turtlebot2 github-page [21. The example programs are not always ready for specified needs. Therefore, the code must be modified. This robotics solution uses modified version of the follow the_route.py program. In our solution, each defined point on the map must be capable of performing a specific function. Therefore, the arrival of the robot at the spesific point of the map must be detectable by the program code. For this purpose, we created a program that sends a unique message with its own topic when the robot reaches the map point. To listen to these messages, a separate Python program was created. This program executes the actions defined by the code when the program detects the message associated with the map point. Figure 1 shows the files and programs associated with the customized follow_the_route.py program.

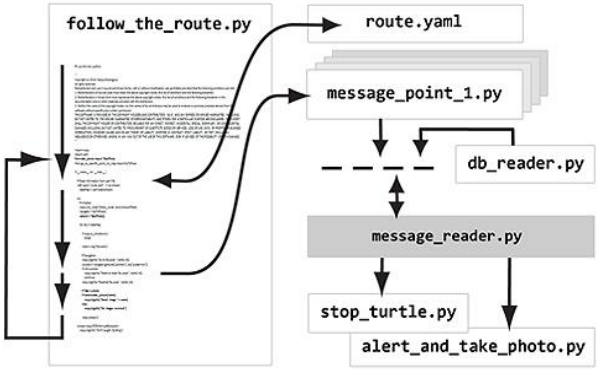

Figure 1. The program files of the implementation

Follow_the_route.py controls the robot in the desired area delimited by predefined map points. The route.yaml file contains the x and y coordinates of predefined points.

In the original follow_the_route.py program, the robot took a picture at each point, but it was modified for this implementation to send a unique message. This will keep track of the points that have been written in the log file and, at the same time, use the message_reader.py program to launch new programs at each point as needed.

At the same time, the RuuviTag sensor sends temperature and other data to the database that is read actively by db reader.py. If a predetermined value of the RuuviTag sensor is exceeded in the database, the db_reader.py program will send a message such as "Alert, temperature 44."

Message_reader.py also reads messages continuously. If the message contains "Alert," the program starts the stop_turtle.py program that stops the robot. In addition, the alert_and_take_photo.py program is launched. The program sends an alert to the user interface and takes a picture of the place.

Because almost every mobile device has a web browser, it was decided to implement a web user interface. The user interface and its development do not require installation of separate development environments. Because the ROS operating system does not support the web browser based communication model, the browser is not capable of communicating with the ROS operating system. A rosbridge interface has been developed to enable communication between the ROS operating system and web applications. Figure 2 shows a typical web application that uses rosbridge.

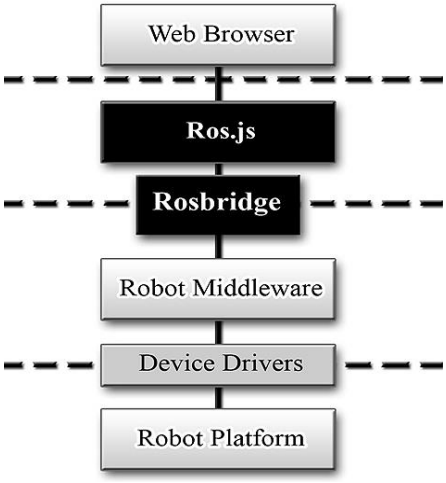

Figure 2. The rosbridge interface between the ROS operating system and web applications

There is a 3rd party Javascript library known as ros.js that has been developed to help creating web applications. The library is designed to integrate seamlessly with the ROS web application. Although it is versatile and extensive, ros.js can be used with a very simple code.

Figure 3 shows the complete layout of the user interface in the browser window. This view acts as the front page of the user interface. It displays temperature, humidity, barometric pressure, and acceleration data as measured by the RuuviTag sensor. With browser-based implementation, the same action logic can be utilized both locally on the touch screen of the robot and in the interfaces of remote devices.

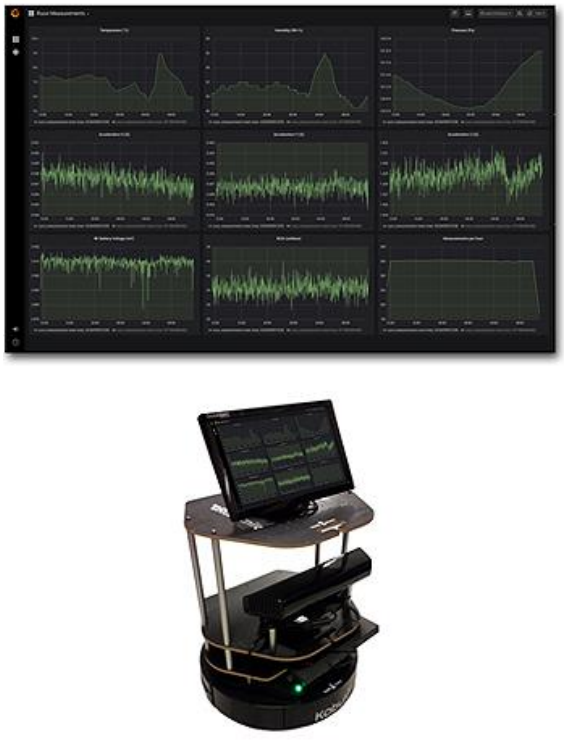

Figure 3. The layout of the user interface

IV. Results and Discussion

The robot created a sufficiently accurate map of the area, but the mapping could not detect all the obstacles in the area to be mapped. When creating the map, it was found that the robot detects thin obstacles poorly. Such objects should either be marked on the map manually or removed from the environment. However, the objects that are manually marked on the map can cause problems because the robot may not be able to locate itself if it cannot find the object shown on the map. In addition, the sensor could not detect very low obstacles. These results are consistent with the findings of other studies and our previous research findings.

When environmental measurements were taken, the robot was able to locate and find the desired measurement points with sufficient accuracy. This was facilitated by the limited size of the area used for piloting, which was about 20X20 meters. A good enough map from a small space can be created with a reasonable amount of work compared to a large space. In addition, errors due to odometry do not become a big problem in a map made of a small space. The area contained a sufficient number of detectable obstacles to ensure reliable testing.

The used mobile robot Turtlebot 2 is inexpensive and well suited for testing. Because it is intended only for indoor use and flat floor surfaces, demanding conditions require a more robust and expensive physical robot implementation than the one used in this study. However, the pilot project showed that the low-cost robot can also provide advanced functions with the help of ROS and sophisticated software components.

It was also found that the processing power of the Raspberry Pi 3 computer is very limited. Its graphics processing unit (GPU) will not be suitable if there is a lot of graphics content to take care of. An example if there are a lot of obstacles that need to be identified from the video or image. Much better results can be produced using a variety of low-cost computers such as the NVIDIA Jetson Nano or Jetson Xavier NX. These computers include the more efficient GPU needed for video and image processing. More processing power will enable the use of neural networks in the future studies.

The sensor used in the pilot project was cheap and easy to deploy. However, the sensor was able to measure the environmental parameters with sufficient accuracy. In addition, the sensor can be customized to better fit the needs of the application area because its implementation is based on the open-source approach. The Bluetooth data connection offered by the sensor met the requirements of the application, because the mapped area was compact.

V. Conclusions

The main objective of this study was to provide a low cost and flexible solution for environmental sensing. The study introduced the low-cost environmental sensing solution based on a mobile robot and open-source components. We implemented a pilot project to ensure that the proposed solution met our design target. The pilot project showed that it is possible to create mobile solutions for environmental sensing using open source

components, but the development project requires a reasonably large amount of work.

REFERENCES

[1] A. Whitmore, A. Agarwal, and L. Da Xu, The Internet of Things-A survey of topics and trends, Information Systems Frontiers, 2014, pp. 1-14.

[2] H. Saha, A. Mandal, and A. Sinha, Recent trends in the Internet of Things, IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017, 4 pages.

[3] T. Alhmiedat, A survey on environmental monitoring systems using wireless sensor networks, Journal of Networks, 2015 10(11), pp. 606-615.

[4] E. Ganesh, IOT based environment monitoring using wireless sensor network, International Journal of Advanced Research, 2015, vol. 5, no. 2, pp. 964-970.

[5] M. Dunbabin and L. Marques, Robots for environmental monitoring: Significant advancements and applications, IEEE Robotics & Automation Magazine, 2012, 19(1), pp. 24-39.

[6] J. Đurković, V. Vuković, and L. Raković, Open source approach in software development - advantages and disadvantages, Management Information Systems, 2008, Vol. 3 No. 2, pp.29-33.

[7] M. Heron, V. Hanson, and I. Ricketts, Open source and accessibility: advantages and limitations, Journal of interaction Science, SpringerOpen, 2013.

[8] G. Signorini, Open Source and Sustainability: the Role of Universities, 2019, https://arxiv.org/abs/1910.06073, 20 pages.

[9] S. Mohamed, M. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila, A survey on odometry for autonomous navigation systems. IEEE Access, 7, 2019, pp. 97466-97486.

[10] M. Quigley, B. Gerkey, and W. D. Smart, Programming robots with ROS: a practical introduction to the Robot Operating System, O'Reilly Media, 2015.

[11] A. Kyaw, H. Truong, and J. Joseph, Low-cost computing using Raspberry Pi 2 Model B, Journal of Computers, 2018, 13(3), pp. 287-299.

[12] F. Álvarez, L. Almon, H. Radtki, and M. Hollick, Bluemergency: Mediating post-disaster communication systems using the Internet of Things and Bluetooth mesh, arXiv preprint arXiv:1909.08094, 2019.

[13] S. Naqvi, S. Yfantidou, and E. Zimányi, Time series databases and influxdb, Studienarbeit, Université Libre de Bruxelles, 2017, 44 pages.

[14] E. Betke and J. Kunkel, Real-time I/O-monitoring of HPC applications with SIOX, elasticsearch, Grafana and FUSE, In International Conference on High Performance Computing, 2017 , pp. 174-186.

[15] G. Nagymate, Android based autonomous mobile robot, Recent Innovations in Mechatronics, 2015, 4 pages.

[16] J. Martins, MRSLAM - Multi-Robot Simultaneous Localization and Mapping, 2013, https://ap.isr.uc.pt/archive/MRSLAM_dissertacao_Joao_Martins-vfinal-040913_235208.pdf, 51 pages.

[17] M. Jin, S. Liu, S. Schiavon, and C. Spanos, Automated mobile sensing: Towards high-granularity agile indoor environmental quality monitoring, Building and Environment, 2018, 127, pp. 268-276.

[18] M. Reggente, A. Mondini, G. Ferri, B. Mazzolai, A. Manzi, M. Gabelletti, P. Dario, and A. Lilienthal, The dustbot system: using mobile robots to monitor pollution in pedestrian area, 3rd Biannual International Conference on Environmental Odour Monitoring and Control, 2010, pp. 273-278.

[19] L. Terrissa, S. Ayad, and N. Zerhouni, Robotics based Solution for data center e-monitoring, In International Conference on Advanced Systems and Emergent Technologies, 2019, pp. 201-208.

[20] A. Maurya, M. Sonkusare, A. Raut, D. Tamhane, and D. Palase, Surveillance robot with human detection. In Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 107-110.

[21] M. Silliman, Mark Silliman's Turtlebot github-page for developers interested in robotics, https://github.com/markwsilliman/turtlebot.