Proceeding of System Engineering & Sports Engineering, 1 (x) (2019) ISBN 978-967-XXX

COLLOQUIUM ON SYSTEM & SPORTS ENGINEERING (COSSE 2019)

Website: https://uitm-sea-ic.com/
doi:
Research paper, Short communication, Review, Technical paper

Deployment Of Autonomous Navigation Feature On An Unmanned Ground Vehicle (UGV)

Mohamad Haziq Ahmad Yusri

Member of SEA-IC
Student of Mechanical Engineering, UiTM Shah Alam
E-mail: haziqahmad38@gmail.com

Abstract

This project presents a study about the deployment of autonomous navigation feature on an unmanned ground vehicle (UGV). Specifically, the study focuses on the use of a robotic operating system (ROS) to manage and control the motion characteristics. To realize autonomous navigation, an advanced light detection and ranging (LIDAR) sensor is integrated into the UGV. The core hardware Raspberry Pi and Arduino are used for sensing and actuating mechanisms. The aforementioned core hardware enables the UGV to sense and decide the tasks with better accuracies, e.g., avoid objects and obstacles while performing prescribed navigation tasks. In order to verify the effectiveness of the proposed solution, several assessments are designed and put to the test. The outcome of this project is will be beneficial in the manufacturing industry for example in material handling section in assembly plant and managing inventory of goods in warehouse such as Amazon, Lazada companies, .

Keywords: Arduino , *LIDAR , Raspberry Pi , ROS , UGV*

1. Introduction

It can be seen that technology advancement is so rapid and many complex applications are realized to facilitate and simplify routine of human activities, production plants, agricultural sector and so forth. The advent of fourth revolution industry (IR4.0) catalyze the growth of digital technology which has solved many industrial problems in production, manufacturing and logistics.

In production plant for example, IR4.0 is heavily used for a number processes, including maintaining the raw material's inventory, predictive maintenance of machines and logistic of finished parts. In this project, we explore and conduct a study on logistic of finished parts in much detailed context pertaining to hardware and software components in order to effectively transport a finished part from one point to another. In this regard, we employ light detection and ranging technology (LiDAR) in managing the logistic of finished parts. Specifically the LiDAR is used as a sensor for deciding the best route option when transporting the finished parts from point A to point B without colliding with any object. This is done through the deployment of robot operating system (ROS) framework on the logistic management system. This method smoothens the work flow of the transportation process [1]. ROS has been widely used in many countries to control automation robot and has a smooth communication link. Wang Rui who is the researcher in laser scanning on ROS framework has done a number of experiments in point cloud where it shows that the data received is very precise even for small parts [2]. As a result, it effectively increases the output of production rate. The abovementioned logistic management system normally implemented and applied on mobile robot which is commonly known as unmanned ground vehicle (UGV). By and large, it comes with the ability to maneuver autonomously by following prescribed lines and reroute or stop if there is obstacle blocking its pathway.

Indeed, the line tracking system is a good approach for navigating along the pre-defined tracks or lines, still, there is possibility for UGV to move out of the track. When the UGV cannot find the prescribed line or accidentally follow the wrong line, it will disturb the whole work flow process. Thus, the technician needs to reset the UGV to ensure it operates properly. To effectively solve this problem, in this study, we propose to replace the line tracking sensor with LiDAR sensor where it has the ability to detect surrounding objects at all directions [3]. The proposed LiDAR has maximum range of 8m wide of detection and can detect small object such as wire. The long-range detection feature of this sensor increases the sensitivity and awareness of the UGV about its environments and the obstacles around its surroundings.

2. Hardware-Software Integration

The integration starts with mechanical design of an UGV that could support and carry all essential components including LiDAR sensor, a high discharge rate battery pack, a micro-controller and mini-computer (raspberry Pi), the design must be strong and can sustain heavy loads and abrupt changes when accelerating and braking. The overview of the deployment of the logistic management system is illustrated by the following stages.

Stage 1: Design prototype model of UGV
Stage 2: ROS environment setup in Raspbery Pi on Ubuntu OS
Stage 3: ROS communication application (Publisher, Subscriber, Topics)
Stage 4: Electronics and power distribution setup on UGV
Stage 5: Apply programming in avoiding obstacle to the UGV with appropriate calculation

2.1. Design prototype model of UGV

The design process is done by a CATIA Computer-Aided-Design (CAD) software package. First, a heavy-duty chassis is designed to replace the old chassis RC car as the torque for the mounted DC motor is quite small. The new chassis will support a bigger brushless DC motor which has greater output torque than common DC RC motor. Figure 2.1.1(a) illustrates the finished and printed chassis prototype (the preliminary design). The overall dimension of the chassis is shown in Figure 2.1.1 (b). Meanwhile, Figure 2.1.1(c) shows the final design of the chassis upon a series of test done on the preliminary chassis prototype. The overall dimension of the final chassis is shown in Figure 2.1.1(d).

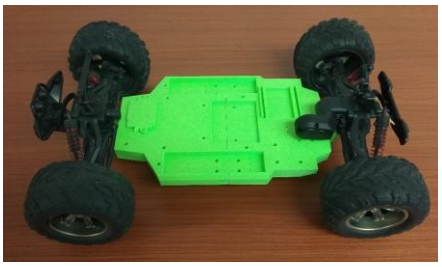


Figure 2.1.1(a): The preliminary design of UGV chassis.

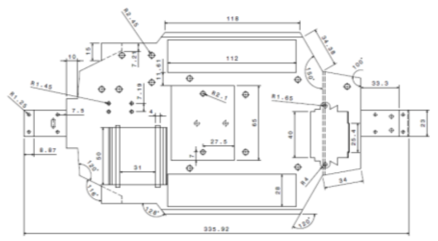


Figure 2.1.1(b): Dimensions for the preliminary design

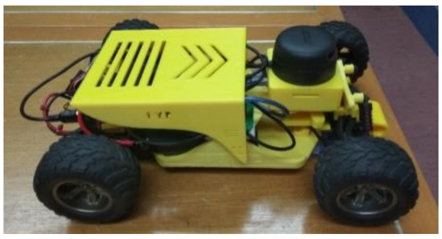


Figure 2.1.1(c): The final UGV design

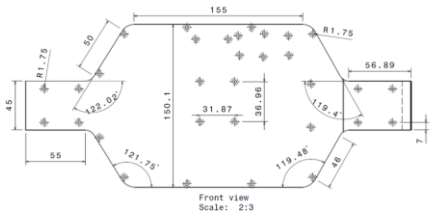


Figure 2.1.1(d): Dimensions for the final chassis design

The prototyping process is facilitated by 3D printing technology and almost 80 percent of the UGV parts are printed via 3D printer including gears. In the beginning various infill percentage are tested to determine the best density that could survive the sudden acceleration and braking. All the parts are printed using cornstarch-based material (Material properties is Polylactic Acid (PLA)) to promote green technology. This material properties behave according to typical trend of plastic behavior. However, PLA is less elastic in comparison to plastic ABS material. The melting point of PLA is about 180 degree Celsius and the maximum yield strength is 70MPa [4]. For electrical and electronic parts, the material is made up of a non-conductor characteristic which can be used to protect the electrical components from harm [5].

2.2. Configuring ROS environment on Ubuntu OS

ROS is a computing platform for the robotic and artificial intelligent and it provides various tools and software that are commonly used for automation tasks. The beauty of the system is it can reduce the development and maintenance time. Example of application is in NASA Space where is uses ROS framework on Robonaut 2 that has the ability to help humans work chores and explore in space for discovery. This shows that the ROS environment is a good communication based program.

For this project, ROS is used as a main core to manage decisions and commands the UGV for autonomous navigation. The ROS environment setup is mandatory to launch the robotic system on this framework. The ROS application development environment used for the UGV is as follows.

- Hardware: Raspberry Pi 3B
- Operating System (OS): Ubuntu version 16.04
- ROS: Kinetic Kame

Figure 2.2.1: UGV main desktop framework

ROS package need to be installed on Raspberry Pi in order to use the specific tools and software inside the ROS ecosystem. The installation also requires some other inquiries to startup the configuration such as network time protocol (NTP). This protocol is a compulsory configuration in order to reduce ROS time difference in the communication between multiple PCs. Then, initialization of the operating system, ROS dedicated build system called catkin workspace is created to let all the working programming to work and communicate each other in this ecosystem.

2.3. ROS communication

In ROS environment there is a meta-operating system where it links all the sensors and devices to communicate each other called ecosystem. This refers to the structure that connects hardware manufacturers, operating system developing companies, app developers and end users. There are several main types of communication that been used for this UGV.

- Topic - The main subject that want to discuss
- Publish – Action of transmit relative message
- Subscribe – Action of receiving relative message

These type of communication is the link between LiDAR sensors and the motor to the main server. The ROS master will collects all the information from the subscriber and publisher to link the node together. The master and node communicate using extensible markup language remote procedure call (XMLRPC)[6].

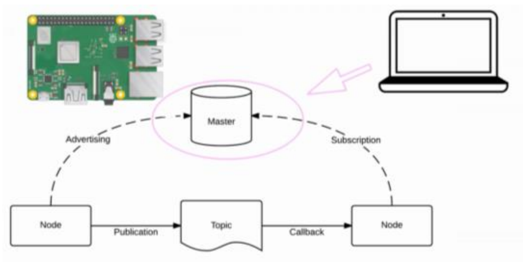


Figure 2.3.1: The communication web between hardware

From the figure, Raspberry pi will be the slave while the laptop is the running master. The message from the laptop will transfer to the slave and execute the command given. This will have smooth system that will increase the data transfer efficiency from LiDAR sensor to the motor action.

2.4. Electronics and power distribution setup

Power supply is the heart of UGV which used to power up the main board and brushless motor. The battery is used that have higher discharged current and contain 3 cells [7]. Each of the cells contain 3.7V output voltage and it is connected in series that will give 11.1V for the UGV. This voltage is then separate to other devices such as electronic speed controller (ESC), LiDAR sensor, servo motor, and Raspberry Pi.

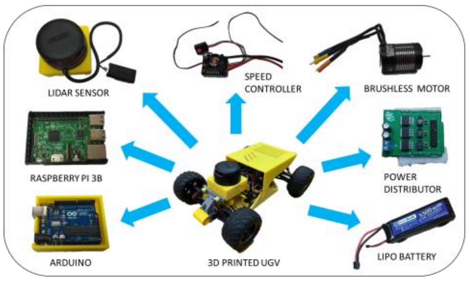


Figure 2.4.1: UGV main devices

The power supply distributor will distribute the power from battery to the other devices. This distributor will give constant current supply to all devices equally that makes the system stable when running. These are the operate voltage that needed to run the whole system:

• Raspberry Pi 3B	: Operates at 5.2V and 1.35A
• RP LiDAR A2	: Operates at 5.0V and 1.50A
• ESC	: Operates at 11.1V and 45A (max)
• Arduino	: Power supply from Raspberry Pi 3

2.5. Programming ecosystem of the UGV

In this system, programming is the core to maneuver the UGV and obstacles avoidance. This can be called the decision maker that will calculate the steering angle and decide to avoid from the blocks. There have 3 types of languages are used in this system; there are C, C++, and python. Each of them are used for a difference command. For C++, the coding is to extract data from 0 degree to 180 degree wide area to get front view of the UGV from LiDAR sensor. Then, python is used to calculate the steer angle and brake system to avoid the obstacles. Lastly, the calculated data will be send to the Arduino for moving the motor [8]. These 3 combination coding will give the smooth data transfer from LiDAR to motor control.

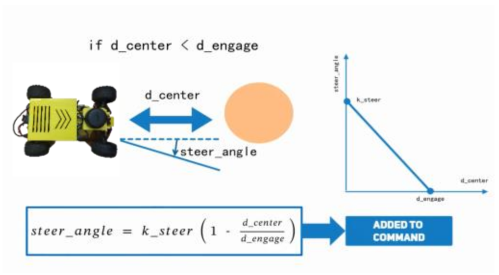


Figure 2.5.1: Steering formula in avoiding the obstacle

The figure above shown that the motor speed control where the speed of the UGV will slowed down if the obstacle is getting closer from it [9]. Then, the steer angle will engage if the distance is less than the threshold. This formula is inspired from Ackermann steering geometry. The LiDAR rotates and detect every angles and will locates the minimum distance from the UGV to avoid from being crash with the obstacles. The minimum data will be checked for the steering to steer.

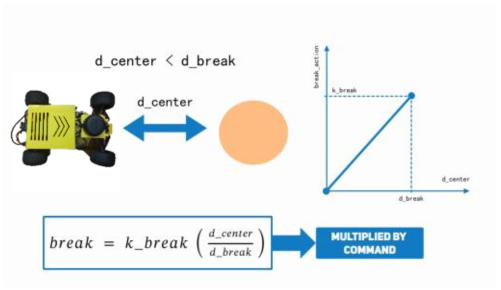


Figure 2.5.2: Formula for break action

For breaking system, the UGV will apply full break action if the distance is less than distance break value. This will ensure that the UGV will avoid hitting the obstacles with full speed. The value of zero means stop the motor while value one will move with certain amount of speed limit for this UGV.

3. Results and discussions

The UGV model has been tested with variety of speed to check the strength of the gear structure. Gears are the important parts that used to move the UGV forward. Then, it has achieved the main goal where it needs to avoid the obstacles in front. The data for these tests has been recorded to improve the UGV in the future.

3.1. Fine Tuning the speed

The brushless motor used an ESC to control the speed of the motor. This type of motor can produce a constant torque while having high movement speed. In programming the ESC, the calibration must be done to initiate the speed values. Maximum RPM that can be achieved is 50000 that consumed maximum current of 40A. This high current value will protect the ESC from burning.

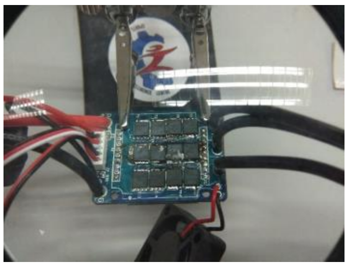


Figure 3.1.1: The burning ESC

In testing the UGV, the sudden rise in current lasting about three nanoseconds that caused the motor to move at the maximum speed and burn the electronic main board. This phenomena is called voltage surges that always happened at all high current supply devices. The strike current cause the ESC to overheat and melt the electrical components. In preventing it from overheat, the manufacturer has attached together with heat sink and high speed cooling fan to cool down the temperature.

Even though the cooling devices have been attached, this ESC does not sustain the surge voltage supplied from the 3 cells LiPO batteries. From observation, the speed of the motor affect the ESC and motor temperature. The temperature keeps increasing against time. The best running time for the ESC is 15 minutes per testing. This speed controller need to control in order to get smooth motor movement in longer period of time. Therefore, the programming is applied to the Arduino in controlling the speed.

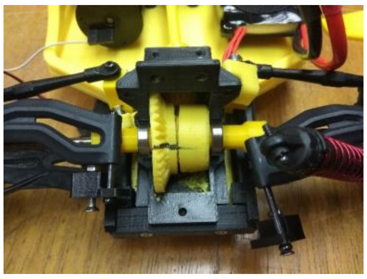


Figure 3.1.2: Broken rear gear due to high speed

Before starting the programming, ESC calibration need to be setup. The calibration process caused the gear to broken down. This requirement is to identify the maximum and minimum speed for the brushless motor before operating it. Therefore, in calibration state, the motor went to maximum speed for about 8000 microseconds and stop for about 8000 microseconds. As a result, the impact cause shock to the transmission gear. The gear sustain too much force where all the loads carried were focused on the gear. Thus, the motor need to be unmounts during the calibration process as to ensure the safety of the UGV parts and devices.

3.2. Road Testing

This UGV has been tested on the side way with two obstacles block the path. In this project, the UGV has to maneuver automatically when the command is received from the main server, laptop. The LiDAR sensor will be rotates and start scanning on the surrounding area. Then, this sensor will update the distance data for each angle and begin to calculate for the output speed and steering angle.

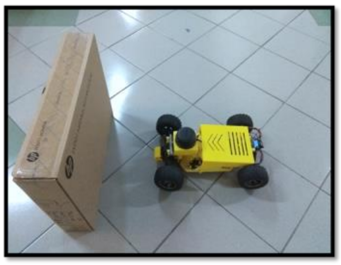


Figure 3.2.1: The UGV fully brakes when the distance is less than 0.4m

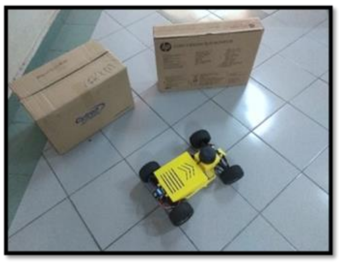


Figure 3.2.2: UGV succesfuly avoid the obstacles

In this situation, the two blocks have been setup about 10m from the UGV on the hallway street. The distance travel will counter the motor speed and make it avoidable. It took many testing parameters to get a better steering angle in avoiding the obstacles. The UGV will slowed down when the distance to obstacles is getting nearer. When the UGV is less than 2 meter, the throttle will fully stop.

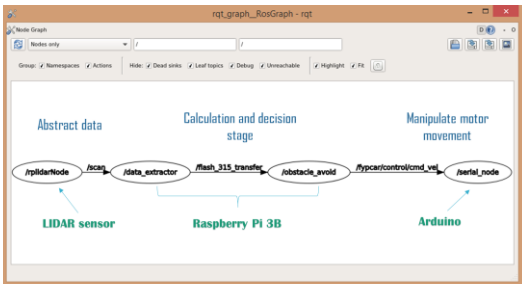
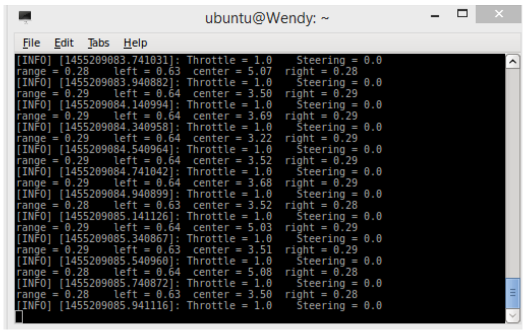


Figure 3.2.3: Programming structure in ROS environment



```
[INFO] [1455209083.741031]: Throttle = 1.0    Steering = 0.0
range = 0.28    left = 0.63  center = 5.07  right = 0.28
[INFO] [1455209083.940882]: Throttle = 1.0    Steering = 0.0
range = 0.29    left = 0.64  center = 3.50  right = 0.29
[INFO] [1455209084.140994]: Throttle = 1.0    Steering = 0.0
range = 0.29    left = 0.64  center = 3.69  right = 0.29
[INFO] [1455209084.340958]: Throttle = 1.0    Steering = 0.0
range = 0.29    left = 0.64  center = 3.22  right = 0.29
[INFO] [1455209084.540964]: Throttle = 1.0    Steering = 0.0
range = 0.29    left = 0.64  center = 3.52  right = 0.29
[INFO] [1455209084.741042]: Throttle = 1.0    Steering = 0.0
range = 0.29    left = 0.64  center = 3.68  right = 0.29
[INFO] [1455209084.940899]: Throttle = 1.0    Steering = 0.0
range = 0.28    left = 0.63  center = 3.52  right = 0.28
[INFO] [1455209085.141126]: Throttle = 1.0    Steering = 0.0
range = 0.29    left = 0.64  center = 5.03  right = 0.29
[INFO] [1455209085.340867]: Throttle = 1.0    Steering = 0.0
range = 0.29    left = 0.63  center = 3.51  right = 0.29
[INFO] [1455209085.540960]: Throttle = 1.0    Steering = 0.0
range = 0.28    left = 0.64  center = 5.08  right = 0.28
[INFO] [1455209085.740872]: Throttle = 1.0    Steering = 0.0
range = 0.28    left = 0.63  center = 3.50  right = 0.28
[INFO] [1455209085.941116]: Throttle = 1.0    Steering = 0.0
```
Figure 3.2.4: Output value from the calculation stage when the UGV is applied to the obstacles

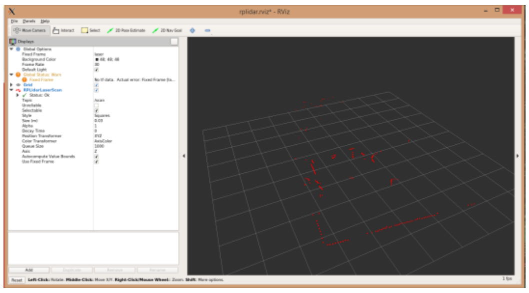


Figure 3.2.5: RViz framework

The figure 3.2.3 shows the overall of ROS environment of this system. The calculation data from the LiDAR sensor is applied to navigate the UGV. The output scan data are from 0 to 180 degree in angle range which scanning in front of the UGV. Then, the minimum distance will be taken for calculation process. The throttle range is from 0.0 to 1.0 values that will drive the UGV forward. Value 1.0 means full throttle, while 0.0 will be fully stop. For steering, the maximum reading value is 1.5 and minimum value is -1.5. The middle of the steering value indicates no turning angle. In navigating the UGV to the left, the negative value will be applied to the command but for turning to the left the command will be vice versa.

In figure 3.2.5, the distance value from LiDAR to the obstacles have been recorded to the point cloud in RViz environment. The red dots show the obstacles at each 360 angle that can be used for mapping [10]. For this project, only front obstacles of the UGV will be evaluated to get smooth path movement.

4. Conclusions

This paper describes development of logistic management system that employs UGV as the main test bed and utilize a

LiDAR sensor to optimize autonomous navigation system. Then, the prototype model of the UGV is tested with high speed motor to get better maneuver flow. The work that need to be fulfil is to ensure that the UGV successfully avoid the obstacles smoothly. Therefore, programming is the main core in controlling the UGV movement behavior by extracting the data from LiDAR sensor. The results of this investigation show that, the speed control parameter affect the maneuver system of the UGV. The low speed movement will provide greater efficiency to the calculation values as the time to process the data will be much longer. Thus, the programming must be able to read all the data from the LiDAR sensor in calculating the path flow. This study has found that, ROS environment can run all programming efficiently without delay. As a result, the UGV achieved to maneuver automatically by avoiding the obstacles. The future improvements must be done in order to advance the goal achievement such as move the robot at a certain location with RViz mapping indicator. Besides, this method can also be apply to the other applications such as rockets, UAV, robots, and artificial intelligence system.

Acknowledgement

I would like to take this opportunity to express my utmost gratitude to my research supervisors Dr. Hanif Mohd Ramli and Dr. Zulkifli Mohamed. Without their assistance and dedicated involvement in every step throughout the process, this paper would have never been accomplished.

I would like to express my deep gratitude to Tiziano Fiorenzani whom creating a wonderful video on YouTube channel about ROS. His videos give me inspiration in doing my project where he shows step by step the installment of ROS environment on the Raspberry Pi.

I would also like to thank the experts who were involved in teaching me the ROS robot programming by developing a handbook written by YoonSeok Pyo, HanCheoi Cho, Ryuwoon Jung, and TaeHoon Lim. This handbook is the main guidance in creating my own ROS environment for this project. It makes me realize that there are a lot of things that need to learn for my future profession.

I feel I have learnt a lot from writing this thesis analyzing and realizing the concept of automation throughout the system. This will be the great treasure I will cherish not only in my future academic career but in my whole life.

References

[1] L. Wang, Y. Zhang, and J. Wang, "Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 276–281, 2017.

[2] W. Rui, L. I. Xiaoge, and W. Shuo, "A Laser Scanning Data Acquisition and Display System Based on ROS," pp. 8433–8437, 2014.

[3] F. B. P. Malavazi, R. Guyonneau, J. Fasquel, and S. Lagrange, "LiDAR-only based navigation algorithm for an autonomous agricultural robot," *Comput. Electron. Agric.*, vol. 154, no. September, pp. 71–79, 2018.

[4] Y. Ding, B. Lu, P. Wang, G. Wang, and J. Ji, "PLA-PBAT-PLA triblock copolymers : E ff ective compatibilizers for promotion of the mechanical and rheological properties of PLA / PBAT blends," vol. 147, no. November 2017, pp. 41–48, 2018.

[5] M. Kodal, A. A. Wis, and G. Ozkoc, "The mechanical , thermal and morphological properties of γ -irradiated PLA / TAIC and PLA / OvPOSS," *Radiat. Phys. Chem.*, vol. 153, no. October, pp. 214–225, 2018.

[6] Q. Xu, "Design and Implementation of an ROS based Autonomous Navigation System," pp. 2220–2225, 2015.

[7] Y. Lee and S. Park, "Rapid Charging Strategy in the Constant Voltage Mode for a High Power Li-Ion Battery," pp. 4725–4731, 2013.

[8] A. D. Deshmukh and U. B. Shinde, "A low cost environment monitoring system using raspberry pi and Arduino with Zigbee," *Proc. Int. Conf. Inven. Comput. Technol. ICICT 2016*, vol. 2016, 2016.

[9] A. A. El-samahy and M. A. Shamseldin, "Brushless DC motor tracking control using self-tuning fuzzy PID control and model reference adaptive control," *Ain Shams Eng. J.*, vol. 9, no. 3, pp. 341–352, 2018.

[10] B. Holý, "Registration of lines in 2D LIDAR scans via functions of angles," *Eng. Appl. Artif. Intell.*, vol. 67, no. July 2017, pp. 436–442, 2018.