Embedded system environment self-awareness using LIDAR technologies for robotics applications

S G Bogdanov¹, D S Chikurtev² and N R Spasova³

¹Technical University of Sofia, Faculty of Computer Systems and Technologies, Department of Computer Systems, Bulgaria
²Institute of Information and Communication Technologies, Bulgarian Academy of Science, acad. G Bonchev str., block 2, Bulgaria
³Technical University of Sofia, Department of Microelectronics, Faculty of Electronic Engineering and Technologies

e-mail: dchikurtev@gmail.com

Abstract. LIDAR technology is a major part of autonomous robots. That technology provides the necessary data for autonomous navigation, localization, surface scanning and obstacle detection. This paper presents a modular embedded system architecture, which is a part of a self-sufficient AI. The LIDAR software module of the proposed system is described. That module will provide 2D information about the surrounding environment. It is developed like a loadable module with options for self-compiling in the system. This research is the first step of self-awareness embedded system. In addition, we illustrate the application of the module in localization and navigation of mobile robots. The results achieved show that the developed software module is applicable for robotics localization and navigation.

1. Introduction
Light detection and ranging (LiDAR) technology is part of the active remote sensing systems. These types of systems emit radiation toward the target using their own energy source and detect the radiation reflected from that target. An important advantage for active sensors is their ability to obtain measurements independently of sun illumination conditions and largely independent of weather conditions [1]. This advantage is the main reason for the wide applications of the LiDAR technologies. Nowadays LiDAR is applied in many different areas like autonomous vehicles, robotics, UAVs, agriculture, forestry, archaeology and others [2-6].

The main function of the LiDAR is to provide data of the environment by applying 2D or 3D scanning. Then the output data of the LiDAR have to be processed by applying software algorithms. The final result of the scan process is 3D or 2D map or object [7, 10].

In the advanced mobile robot systems, the LiDAR technology is a must system. It is the main system for mobile robot localization and navigation applications. Thanks to the LiDAR, mobile robots are capable of self-localization and self-movement [8]. Usually 2D LiDARs are embedded into mobile robots, because they are less energy consumable and require less computing performance [9]. In addition, most of the mobile robot applications are indoor in a static or slightly changing environment, so there are no needs for 3D scanning. For example, in the factories and warehouses is known the location of all the objects and the participants in the production and transportation processes, so all the information that the robot needs is its location and the locations of the rest of the environment.
In order to reduce the requirements of computing power and to provide easy to use LiDAR systems we propose a model of a LiDAR based embedded system, that autonomously processes the LiDAR data and outputs ready to use data. This system provides a modular approach for mobile robot architecture design. Another advantage of the system is the universal communication with other devices and systems. The paper describes a part of modular embedded system architecture which is a control of a self-sufficient artificial intelligence (AI). This research will cover the RPLidar A3 software module of this system which provides to the system 2D information about the surrounding environment [11]. This will be the first step of self-awareness. It is developed like a loadable module with options for self-compiling in system.

2. Requirements for the modular embedded system

Part of our goal is to apply the embedded module of RPLidar A3 in the Robot Operating System (ROS). In the current state when we want to use RPLidar in ROS we have to install sensor SDK and ROS packages, then compile the packages and process the sensor data with running the RPLidar node. The node processes the sensor data and outputs messages in the topic named sensor_msgs/LaserScan.

In ROS the standard methods for autonomous navigation with robots are: Simultaneous Localization and Mapping (SLAM) gmapping and Autonomous Monte Carlo localization approach (AMCL). Both methods are using laser scan data for mapping and localization [12, 13]. So, based on their requirements we plan to include in the proposed embedded module, a method that outputs directly the message with the necessary data.

To provide an accurate and well working system we should know in detail the parameters of the RPLidar A3 sensor. The sensor characteristics provided by the manufacturer are represented in table 1.

| Item               | Enhanced mode                                                                 | Outdoor Mode                                                                 |
|--------------------|-------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| Application        | Extreme performance Ideal for indoor environments with maximum ranging distance and sampling frequency. | Extreme reliability Ideal for both outdoor and indoor environments with reliable resistance to daylight. |
| Scenarios          | White object: 25 meters                                                        | White object: 20 meters                                                      |
| Operating Range    | Black object: 10 meters                                                        | Black object: TBD                                                             |
| Minimum Operating range | 0.2m                            | 0.2m                                                                          |
| Sample Rate        | 16 kHz                                                                         | 16 kHz or 10 kHz                                                            |
| Scan Rate          | Typical value: 10 Hz (adjustable between 5 Hz-15 Hz)                          |                                                                               |
| Angular Resolution | 0.225°                                                                         | 0.225° or 0.36°                                                             |
| Scan Field Flatness| ±1.5                                                                            |                                                                               |
| Communication Interface | TTL UART                      |                                                                               |
| Communication Speed | 256000 bps                       |                                                                               |

In addition, we need information about the output data from the sensor. In table 2 are described the data output of the sensor.
Table 2. RPLidar A3 data output.

| Data Type   | Unit | Description                                                                 |
|-------------|------|-----------------------------------------------------------------------------|
| Distance    | mm   | Current measured distance value between the rotating core of the RPLIDAR and the sampling point |
| Heading     | degree | Current heading angle of the measurement                                   |
| Start Flag  | (Bool) | Flag of a new scan                                                          |
| Checksum    |      | The Checksum of RPLIDAR return data                                          |

The RPLIDAR outputs sampling data continuously and it contains the sample point data frames in the figure 1. Host systems can configure output format and stop RPLIDAR by sending a ‘stop’ command.

![RPLidar Sample Point Data Frames](image)

Figure 1. The RPLIDAR Sample Point Data Frames.

3. Research and development of the embedded module

We are looking at the RPLidar A3 as one of the affordable consumer products, other Lidar solutions are way more expensive and too much automotive targeted. Looking at the RPLidar A3 as chosen solution for LiDAR sensor, one of the main problems is lack of some specific details in documentation which will make it easier to implement and use from developer. Documentation is almost full, there are missing some details which push you to try and use software which is provided from vendor. There are some not very convenient practices, one of the main problems is lack of response for some of the commands and lack of error responses messages for bad syntax of wrong commands. For example, some “basic” commands do not have response codes and if there is no visual confirmation if LiDAR is spinning things become confusing. There is a pretty good description about most of the commands in the provided documentation, but there is a just little part about starting that communication and no information about spinning the motor.

The figure 2 illustrates a high-profile overview of the working sequence of the scanning command which is the core of the product, the command/functionality which gives us the needed information about the surrounding environment.
How we can notice from image above, even if communication is set how it should be 256000 BAUD 8n1 (115200 is not working stable enough), proper serial port is selected, for example /dev/ttyUSB0 and start command is sent. In theory lidar should start spinning and data should flow back to our host program, but lidar still stays freeze. In conclusion from the scheme above, there is no indication that lidar is waiting to reach some specific stable RPM speed or it is damaged or something else is wrong.

There is something very important which is not mentioned in the documentation, is the RPM speed set command, which is used for setting targeted RPM speed of the motor and is presented in RobotOS implementation and in RobotStudio, obviously because they are working. For robot studio there is no source core. The proper sequence to make all this working in short is:

- Power on the LiDAR;
- Initialize Serial port;
- Spin the motor (and wait for stable RPM);
- Start capture;
- Get data, process and analyze it or redistribute it.

4. Experiments and results
As part of the experiments, sniff communication between the USB to serial converter which is bridge between Host machine and the RPLidar itself. On one side is used Robot Studio as software which is known to work with this hardware and communication is sniffed in both directions (Figure 5). Initiating same sequence of commands which to the sensor does not lead to the same behavior, so continue with looking for the same logical fragments of code in ROS part and python library. Driving the motor is possible applying PWM signal directly on the PWM pin or with commands. Setting PWM with command to, for example 660 RPM (which value is used for default) cause the motor to spin. This will work only if before sending this command DTR bit is cleared (Figure 3 and Figure 4).

Half of the steps are pretty standard except serial port initialization and spin command for the motor. In a python implementation which is an identical copy of RobotOS logic for LiDAR, we assume that RobotStudio is using almost the same implementation. As can be seen from implementation in RobotOS source code:
Important part is \_chanDev->clearDTR(); which clear the DTR bit of the serial interface, it become zero, this means to CP2102 next command should not be transferred to the RPLidar A3 module. The same logic can be found in python library which is clone of the RobotOS implementation and can control RPLidar A3 sensor:

\begin{verbatim}
    def start_motor(self):
        'Starts sensor motor'
        self.logger.debug ('Starting motor')
        # For A1
        self._serial_port.setDTR(False)
        # For A2
        self.set_pwm(DEFAULT_MOTOR_PWM)
        self.motor_running = True
\end{verbatim}

Again self.serial_port.setDTR(False) which set the DTR bit to zero. So, the conclusion about motor spin is that as a valuable part of custom implementation of the RPLidar A3 there is no official information about that. If we clear DTR bit (drive it low) command which is sent right after it is implemented from CP2102 in other case commands are directly passed to RPLidar A3 without any attempt of implementation.

Here we can see a working setup with sniffing tools for both RX and TX lines (figure 5). Sniffing tools is FT4232 breakout custom board or in short USB to 4 UARTs convertor.
After analyses we can see that there is no RPM command from RobotStudio to Sensor which means that this command is stopped in CP2102. This experiment confirms that if someone wants to use RPLidar A3 with provided USB to UART converter it is pretty hard to understand how.

For non-standard serial port implementation, because of 256000 bit per second which is not listed for standard speed, is used termios2 which provides ability to setup custom nonstandard baud rates for serial communication.

![Image](image.png)

**Figure 6.** Starting RPLidar. ‘start_rplidar’ spins the motor with default RPM and after that sends RPLIDAR_START_CAPTURE command to initiate data capturing.

At this point we will use a sensor with a provided adapter board, so we need to make the “magic” command before start receiving data. Implementation provides condition compilation for part with PWM control (Figure 6) so after removing the middle board all will work again; all of the other logic is the same. At the beginning of development is used standard console application without any communication and unified interfaces for AI core. After logic which will deal with RPLidar A3 is stable enough all that will be shipped in kind of package to AI core.

Package will include source code and “Makefile” for building that source code. Architecture is as simple as possible, because simple solutions are more reliable for long term use without user interaction (fig. 7). After serial port initialization and after the motor is spined, we can start to “talk” to the sensor (fig. 8).
For communication we are using two independent threads which are synchronized between them, one thread is dialing with Receiving information - RXthread another with Transmitting information TXthread (see fig. 9). After initialization of the configuration of the sensor and sending initial RPM and start capturing command, data flow is pretty intensive, so we should have a big enough buffer to handle it, or periodically push it to dynamic lined list.
Figure 9. Represent implementation of both thread which are responsible for transmitting and receiving via serial port for RPLidar.

This article is logical intro to AICore which will interface this module using unified interface and will be able to exist on its own and recompile part of itself, but first step is giving it ability to know where it is (at least basic). The RPLidar A3 will provide information about big object around the system in 2D models like point in space. These information will be used for localization of mobile robots and obstacle avoidance.

5. Conclusion
It is developed software module which allow developer to use RPLidar out of ROS and Robot Studio which is vendor provided. Currently is used independently from ROS and Robot Studio on both embedded and desktop Linux based systems. This give opportunity to incorporate this RPLidar sensor with different types of neural network and other flavor of embedded systems. As future work we plan is to incorporate this module for RPLidar with different sensors and technologies which will allow to AICore as main self-driving software to be able to navigate robot autonomously. Important part of this work will be to give option to the embedded system to learn about surrounding environment using different kind of sensor, scanners and cameras. Which will logically lead to option like mission control for given robot.

Acknowledgement
The research is supported by the Bulgarian National Science Fund under the contract № KP-06-M27/1-04.12.2018.

References
[1] Toskova A T, Doychev E H, Toskov B P An Idea for Extension of the Virtual Educational Space for Lifelong Learning Computer science and communications 5 (4) pp 35-38
[2] Dong Pinliang and Qi Chen 2017 LiDAR remote sensing and applications (CRC Press)
[3] Hecht Jeff 2018 Lidar for self-driving cars Optics and Photonics News 29 no 1 pp 26-33
[4] Cheng Yi and Gong Ye Wang 2018 Mobile robot navigation based on lidar In 2018 Chinese
Control and Decision Conference (CCDC) pp 1243-1246

[5] Kumar G Ajay, Ashok Kumar Patil, Rekha Patil, Seong Sill Park and Young Ho Chai 2017 A LiDAR and IMU integrated indoor navigation system for UAVs and its application in real-time pipeline classification Sensors 17 no 6 1268

[6] Khosravipour, Anahita, Andrew K. Skidmore, and Martin Isenburg. 2016 Generating spike-free digital surface models using LiDAR raw point clouds: A new approach for forestry applications International journal of applied earth observation and geoinformation 52 pp 104-114

[7] Inomata, Takeshi, Daniela Triadan, Flory Pinzón, Melissa Burham, José Luis Ranchos, Kazuo Aoyama, and Tsuyoshi Haraguchi 2018 Archaeological application of airborne LiDAR to examine social changes in the Ceibal region of the Maya lowlands PloS one 13 no 2 e0191619.

[8] Behroozpour Behnam, Phillip AM Sandborn, Ming C Wu, and Bernhard E Boser 217 Lidar system architectures and circuits IEEE Communications Magazine 55 no 10 pp 135-142

[9] Hess Wolfgang, Damon Kohler, Holger Rapp, and Daniel Andor 2016 Real-time loop closure in 2D LIDAR SLAM In 2016 IEEE International Conference on Robotics and Automation (ICRA) pp 1271-1278

[10] Csaba György, László Somlyai and Zoltán Vámossy 2018 Mobile robot navigation using 2D LiDAR In 2018 IEEE 16th world symposium on applied machine intelligence and informatics (SAMI) pp 000143-000148

[11] Li Xuyou, Shitong Du, Guangchun Li and Haoyu Li 2020 Integrate Point-Cloud Segmentation with 3D LiDAR Scan-Matching for Mobile Robot Localization and Mapping Sensors 20 no 1 237

[12] Chen Liang, Jian Yang and Hui Kong 2017 Lidar-histogram for fast road and obstacle detection.” In 2017 IEEE International Conference on Robotics and Automation (ICRA) pp 1343-1348

[13] Xu Yuan, Yuriy S Shmaliy, Yueyang Li, Xiyuan Chen and Hang Guo 2017 Indoor INS/LiDAR-based robot localization with improved robustness using cascaded FIR filter IEEE Access 7 34189-34197

[14] Bresson Guillaume, Zayed Alsayed, Li Yu, and Sébastien Glaser 2017 Simultaneous localization and mapping: A survey of current trends in autonomous driving IEEE Transactions on Intelligent Vehicles 2 no 3 194-220