IstiABot, an Open Source Mobile Robot for Education and Research

Remy Guyonneau and Franck Mercier[1]

***Abstract*— IstiABot is a mobile robot made for education and research purposes. This robot aims to be modular, easy to modify and used by first year students as well as last year students and researchers. To achieve those requirements, the robot is built on the top of a CANBus Network.**

This paper presents the robot and the approach behind its building. It also presents an educational application of the platform (tuning a PID controller) and a research application (Simultaneous Localization And Mapping - SLAM - experimentations). Finally it concludes about this experience and introduce two other robots that were built based on the IstiABot.

As it was wanted for the robot to be as free as possible, all the source code and 3D modeling are available.

I. INTRODUCTION

A. Context

Dealing with mobile robots requires knowledge in electronics, mechanics and computer sciences. This is an interesting field to apply theoretical notions. Adding that students are usually well motivated when dealing with robots, robotics has became a widely considered tool for education of undergraduate and graduate students [1], [2], [3].

Polytech Angers (formally named "IstiA") is the engineer school of the University of Angers (France). Among other, it trains students to have a master degree in engineering for industrial systems. As mobile robotics is nowadays a commonly considered industrial tool [4], [5], [6], the need of having a platform to train the students to work on/with mobile robots became urgent.

In addition to the education applications, an other goal was to be able to use the robots for research applications. Indeed, most of the teachers in the school are also researchers and some of them deal with mobile robotics [7], [8]. Having a platform that can be used by students and researchers helps to raise students awareness of the world of research.

B. Developing a New Robot

Despite the number of commercial robots, it was decided to build "home made" robots. This was motivated by several reasons.

First the following requirements were defined for the platform:

- The platform has to be used by bachelor's level students and master's level students. That is, it has to be easy to use for beginners but at the same time it has to allow complex developments for high-level students and researchers;
- The platform has to be easy to modify. Students must be able to develop their own modules (mechanic and electronic add-on) to add sensors/actuators to the robot;
- The platform hardware/software has to be as Free/Open Source as possible.

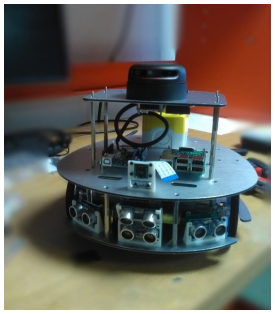


Fig. 1: The IstiABot.

Secondly, this was a perfect opportunity to use some of the in-house production capabilities and to put forward what can be done at the school.

And last but not least, designing and building a robot is a good way to increase robotic skills and to involve the students in the making process.

For all those reasons it was decided to develop the robots instead of buying commercial ones. Here is a quick overview regarding this choice:

- Drawback:
  - The main drawback of the "developing our robot" solution is that it is time consuming. It took a full year to have the first moving robots (conception, realization, debug...).
- Advantages:
  - The robot well suits the needs. It meets all the previously defined requirements;
  - This is a mastered platform, so it can be modified, repaired and evolved as needed;
  - All the hardware design and source code are freely available. Some tutorials were also written to help the students. Thanks to those tutorials, they are able to build their own robots[1] but also to program them (there is one tutorial to set up and use the onboard Rapsberry Pi for "high level" programming [2] and

[1]Polytech Angers, University of Angers, France. name.surname@univ-angers.fr

[1]https://github.com/PolytechAngersMecatroniqueClub/Tutorials/tree/master/Tuto_assembling

[2]https://github.com/PolytechAngersMecatroniqueClub/Tutorials/wiki

one tutorial about the CANBus programming[3] to deal with the other boards.). Those tutorials help the students to be more autonomous working with the robots.

This paper presents the produced platform (depicted in Figure 1) , explaining the choices made and introducing the robot's current applications. Section II details the platform and the home made boards, Section III presents applications in education and research. Finally Section IV concludes this paper.

II. ROBOT PRESENTATION

A. General Architecture

The IstiABot is a differential two wheeled mobile robot with an idler wheel. Considering a differential two wheeled robot allows simple modelization and command which is well suited for beginner students.

Figure 2 presents the general architecture of the IstiABot. As it can be noticed, the robot entirely relies on a CANBus network. This allows a great modularity. Indeed in the CANBus network, only the messages are identified, not the devices. That allows for instance the robot to be controlled by an Arduino board or a Raspberry Pi board without the need of any modification. Basically, it is possible to define a message that says "move forward" and it does not matter if the Raspberry Pi or the Arduino board send it. This allows a large scale of programmer level: beginer will prefer Arduino while others prefer Rapsberry Pi.

B. The Hardware

Two 12V brushed DC motors are used to motorize the robot. Those have a 30:1 metal gearbox and an integrated quadrature encoder that provides a resolution of 64 counts per revolution of the motor shaft. The robot is powered by a 14V lithium-ion battery. A tutorial of how to assemble the robot can be found at the following github repository[4].

In Figure 2 it can be noticed that all the images correspond to elements that are available on the shelf (US sensor boards, Arduino...) but the non graphical boards (power board, motorboard...) are home made (design and fabrication). Some of those boards are presented in the later.

C. The Motor Board

This board allows controlling a DC motor according to a speed order (Figure 3). It requires two different power supplies: 5V for the electronics and 12V for the motor. It receives the speed order through the CANBus. Then it generates a PWM[5] signal that is connected to the motors. In order to control the motor speed, the encoder values are processed and a speed regulation is done using a PID[6] controller. This controller is detailed in Section III-A.

This board has also a current sensor. Its data can be processed to detect current peaks that could happen when the robot is stuck while trying to move for instance.

It was originally designed for brushless technology. It has three output channels corresponding to each motor coil and three hall sensors are wired for feedback. This board is electronically optimized for Maxon EC45 brushless[7], but it was also planned the possibility to use DC technology with coder feedback via LS7366R[8] quadrature counter.

All the schematics and source codes of this board can be downloaded from the following github repository[9].

D. CANBus Shields

A drawback of having a modular robot based on a CAN-Bus Network is that each component needs to be associated to a CANBus interface (Sensors, HMI...).

The key point of the development is the reliability and robustness of CAN communication on each sub-system of the robot. Electronically speaking, except the Raspberry Pi and the Arduino shields, all electronic board are based on the same architecture. They have an ATmega32M1 microcontroller coupled with a high-speed CAN transceiver (MCP2562) for interfacing CAN controller to physical two-wire CANBus.

The ATmega32M1 has an embedded CAN controller, and it exists C and C++ libraries that implement the CAN functions. Those libraries have the benefit to ease the use of the CAN controller low level registers by "hiding them". Otherwise, this set of registers can be complex to manage. For instance the ATmega32M1 proposes six "MOB"[10] that is possible to pre-configure to send messages, to listen a range of addresses or to listen a specific address. But you do not need to know that when using the libraries.

The main drawback of using those libraries is that they consider blocking functions: functions that wait for a flag through a loop, which does not allow to release the CPU for other things. A more efficient way to use the CAN controller would be to use the microcontroller's interruptions instead of having blocking loops. But configuring the interruptions means dealing with the low level microcontroller registers which is not possible with the libraries.

That is, it was decided not to use the availale libraries but instead to directly deal with the CAN registers. To ease the CAN register configurations, a set of non blocking functions was written[11], a tutorial was made[12], and some

[3] https://github.com/PolytechAngersMecatroniqueClub/Tutorials/tree/master/Tuto_CAN

[4] https://github.com/PolytechAngersMecatroniqueClub/Tutorials/blob/master/Tuto_assembling/tutoIstiaBot_montage.pdf

[5] Pulse Width Modulation

[6] Proportional Integral Derivative

[7] https://www.maxonmotor.com/maxon/view/product/397172

[8] https://lsicsi.com/datasheets/LS7366R.pdf

[9] https://github.com/PolytechAngersMecatroniqueClub/Motor_Board

[10] Message OBjects

[11] https://github.com/PolytechAngersMecatroniqueClub/Motor_Board/blob/master/code/include/CanISR.h

[12] https://github.com/PolytechAngersMecatroniqueClub/Tutorials/blob/master/Tuto_CAN/tuto_CAN_ATMEGA.pdf

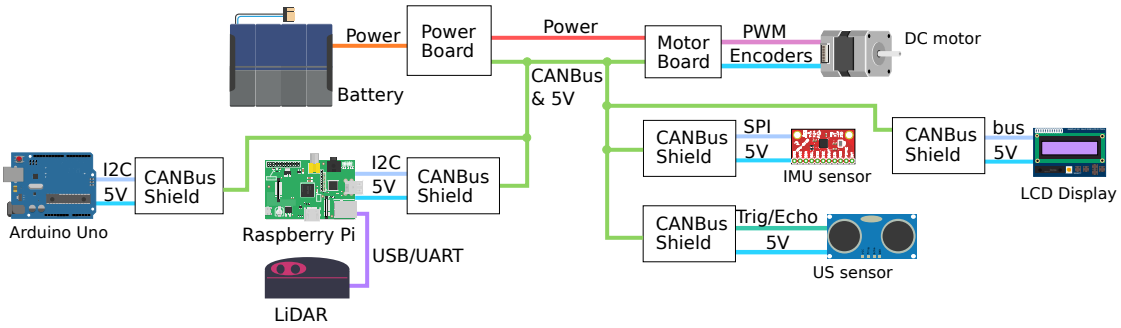


Fig. 2: Architecture of the IstiABot.

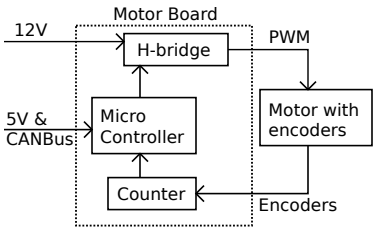


Fig. 3: Summary of the motor board. Through the CANBus is sent messages containing speed command for the motor (values are in $rad.s^{-1}$). A PID controller is implemented in the micro-controller to regulate the motor speed.

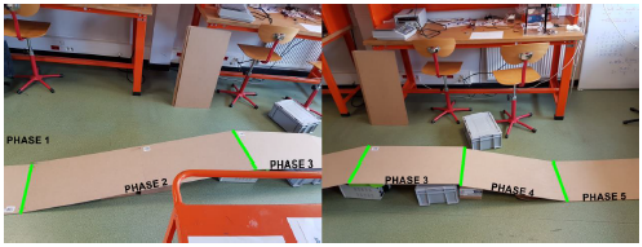


Fig. 4: Experimental setup : the expected trajectory is a straight line going through five phases with different slopes. The robot must move over the path at constant speed.

code examples are available[13][14]. The general idea of those ressources is to ease the understanding of the CAN registers.

III. ROBOT APPLICATIONS

As mentioned earlier, the robot is used by beginners and more experimented students. The idea is that the students can progress with the same platform throughout their learning process. This is mainly allowed by the CANBus Network and the fact that an Arduino board and a Raspberry Pi board can be used to control the robot.

Here is a progression example:

- Once the robot is properly configured, it is simple to make it move with Arduino programming. Considering Scratch for Arduino makes it possible to control the robot even for students that never learned how to code. Based on the same idea, it is possible to easily get values from the sensors and start developing simple moving algorithms;
- The introduction of control loops and embedded programming can be done via low level programming for board's micro-controllers. For instance by configuring the CANBus Network or by setting up a speed regulation on the motor board;
- Considering the Raspberry Pi and a LiDAR sensor, it is then possible to deal with more complex algorithms as localization, mapping, path planning... At this step it is possible to introduce research problems and developments.

In the following part of this document two examples of robot use are detailed:

- Implementation of a PID regulation.
- Implementation of a SLAM algorithm.

A. Setting up a PID Controller

1) Problem Presentation: Figure 4 shows a simple experimentation that illustrates the problem: the robot must move through the path (straight line) at constant speed. The path is divided into five phases with different slopes. First, the experimentation is done using an open loop command: a constant PWM is applied to each motor. By using the Arduino and the wheel encoder measurements (sent by the motor board to the CANBus) it is quite easy to log the speed of each motors. The results of this first experimentation are presented in Figure 5.

Those curves depict two problems:

- The speed is not constant over the trajectory. It decreases when the slope is positive and increases when the slope is negative;
- The left wheel and the right wheel do not move at the same speed. Because of this difference, the robot does not move straight forward.

This simple experimentation allows presenting the problem to the student and intuiting the ideal behavior: The curves must be identical and equal to the setpoint. The students are now able to explain the goal of the project, and later, compare their results with this reference to quantify the gain of the added correction.

[13] https://github.com/PolytechAngersMecatroniqueClub/Motor_Board/blob/master/code/main.cpp

[14] https://github.com/PolytechAngersMecatroniqueClub/US_Board/blob/master/code/main.cpp

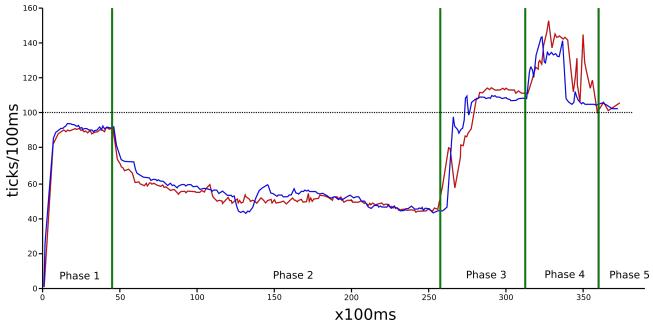


Fig. 5: Results for an open loop command. The graph gives the evolution of the motor speeds (number of encoder ticks each 100 ms) when moving through the different phases. The red curve corresponds to the right motor while the blue curve corresponds to the left one. The setpoint for each motor is 100 ticks/100 ms (the doted line), which corresponds to 0.52 rotation/s in this case.

2) PID Implementation: The implementation of a regulation loop with PID controller is needed to maintain both speed and direction. It is added in a feedback loop and works on an error signal (the difference between the setpoint and the measured signal). There are several possible structures for PID controllers. The numerical one considered for the IstiABot is defined by Algorithm 1.

Algorithm 1 Numerical PID

Require: target, state
error = target - state
sum_error = sum_error+error
correction = correction + K_p×error + K_i×sum_error + K_d×(error - previous_error)
previous_error = error
return correction

For a PID implementation, the main difficulty is to tune the value of the coefficients K_p, K_i and K_d. There is a lot of methods to evaluate those parameters.

As we did not have access to the motor constants we evaluated the PID parameters "by hand" according to the following procedure depicted in Figure 6:

- Set all the parameters to zero and get the step response of the system;
- Then set K_p, then K_i and finally K_d to obtain the behaviors depicted in the figure.

3) Results of PID implementation: In comparison to the curves in Figure 5, the same experimentation was done and results are given in Figure 7.

As it was foreseen, curves are greatly better: the setpoint is reached for each phase and the behavior of the left and right motor is the same. In other words, while not considering the transition steps, the speed remains constant and the robot move straight forward.

4) Project conclusion: This project allows students to work on a recurring automatic problem (regulation) through a practical application. Thanks to the robot they have a good understanding of the problem, the consequences and the expected results. This project allows mixing theoretical approach with experimentation and programming. As the students understand why they are doing the mathematics, they are more motivated and thorough. Furthermore, having at the end, the robot behaving as expected seems to be a reward as important as the given grade...

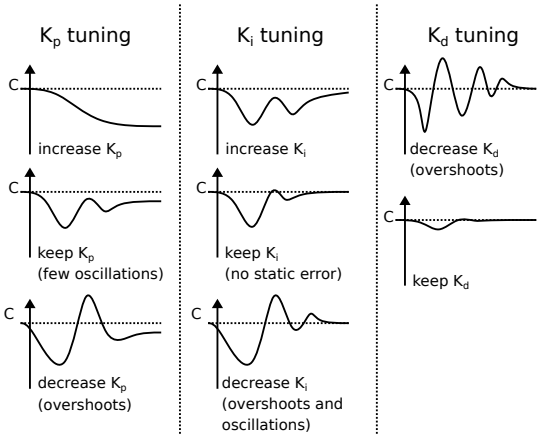


Fig. 6: Experimental tuning. The graphs correspond to examples of step responses, C being the setpoint.

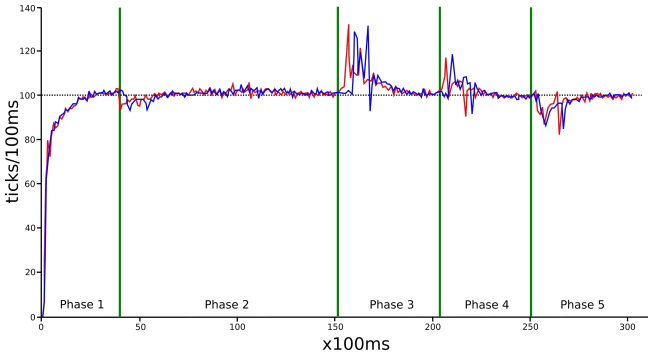


Fig. 7: Results of the experimentation after the implementation and tuning of the PID controller.

B. The SLAM problem

For a mobile robot to be fully autonomous, it needs to be able to evaluate its pose (locate itself) at any time. In order to be able to define its pose, it needs to have, most of the time, a known representation of its environment, i.e. a map. In several situations, it appears that the map is not known and can not be *a priori*. In this case, the robot needs to build the map and locate itself at the same time (The difficulty being: it needs its location to build the map, and it needs a map to locate itself...). This problem is known as SLAM (Simultaneous Localization and Mapping). A SLAM algorithm, developed in our laboratory few years ago [13], was recently implemented and tested with an IstiABot.

1) Overview of the approach: The developped SLAM, named PaSLAM in the later, is a LiDAR[15] based SLAM. A LiDAR measurement set is defined by:

$$Z(t) = \{d_i(t), \theta_i(t)\}, i = 1, ..., n, \quad (1)$$

with $Z(t)$ the measurement set at time t, d_i the distance and θ_i the angle of the i^{th} measurement of the set, and n the number of measurements for one LiDAR set. In the later the notation (t) will be omit to ease the reading.

From this LiDAR set and according to the robot's pose[16] at time t, $\mathbf{x}_r = (x_r, y_r, \theta_r)$, it is possible to compute a point cloud X_Z corresponding to all the detected obstacles from the robot's pose:

$$X_Z = \{x_i, y_i\}, i = 1, ..., n, \quad (2)$$

with

$$x_i = d_i \cos(\theta_i + \theta_r) + x_r, \quad (3)$$
$$y_i = d_i \sin(\theta_i + \theta_r) + y_r. \quad (4)$$

From that, localizing the robot means solving the following minimization problem:

$$\min_{x_r, y_r, \theta_r} \sum_{i=1}^{n} f_M(x_i, y_i), \quad (5)$$

with $f_M(.)$ the cost function, parametrized by the map $M = \{(o_x^j, o_y^j)\}, j = 1, ..., m$ considered as a set of m obstacles defined by their positions (o_x^j, o_y^j). This cost function is defined by

$$f_M(x_i, y_i) = \sqrt{(x_i - o_x^{min,i})^2 + (y_i - o_y^{min,i})^2}, \quad (6)$$

with $(o_x^{min,i}, o_y^{min,i}) \in M$ the closest obstacle to the point (x_i, y_i). That is, $(o_x^{min,i}, o_y^{min,i})$ is defined by

$$(o_x^{min,i}, o_y^{min,i}) = \underset{(o_x^j, o_y^j) \in M}{\arg\min} \sqrt{(x_i - o_x^j)^2 + (y_i - o_y^j)^2}. \quad (7)$$

Each time the robot is localized, the new detected obstacles are added to the current map:

$$M = M \bigcup \{(x_i, y_i)\}. \quad (8)$$

Note that to initialize the SLAM we consider:

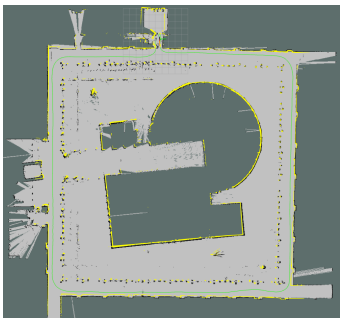
$$M(0) = \{(x_i(0), y_i(0))\}, \quad (9)$$

with $\mathbf{x}_r(0) = (0, 0, 0)$ and $Z(0)$ being the first measurement set.

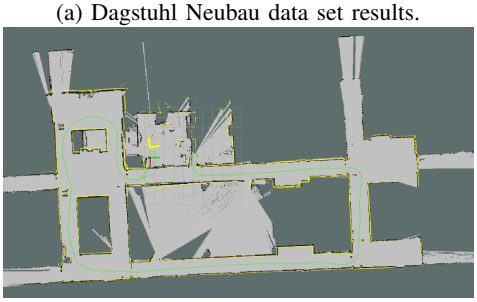
The PaSLAM algorithm considers a Nelder and Mead [14] approach to solve the minimization problem, Equation 5. This method is based on iterative transformations of a simplex defined in the search space.

[15]Light Detection And Ranging

[16]Position (x_r, y_r) and orientation (θ_r). Note that in this case only a two dimentional position is considered as the robot is supposed to move over a flat ground



(a) Dagstuhl Neubau data set results.



(b) L101 building data set results.

Fig. 8: Outputs of the PaSLAM and the Hector SLAM for two different datasets. The displayed maps correspond to the outputs of the Hector SLAM (white cells are free spaces, black cells are obstacles and grey cells are unknown) and the yellow lines correspond to the obstacles computed by the PaSLAM algorithm.

2) Results: The approach was implemented using ROS[17] middleware. ROS is open source and offers a lot of useful tools (implementation can be done in Python or C++, it has a large community and a lot of softwares are available...). To validate the implementation[18], the ROS version of PaSLAM has been compared to the ROS Hector SLAM [16], [15] software[19]. To compare those algorithms we used two data sets available online[20]: the "Dagstuhl Neubau" and the "L101 building" datasets.

Using those logged data we built a map with PaSLAM and a map with Hector SLAM. Then to compare the results we just put one map over the other one to display the differences.

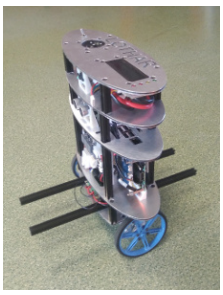
Figure 8 presents the results. It appears that the two approaches provide similar maps. This work was mainly done by a master student and the objective was multiple: to be comfortable with ROS and create a ROS implementation of PaSLAM, to evaluate the processing capability of the Raspberry Pi to onboard mapping tasks and to have a demonstrator used for communcation events.

[17]Robot Operating System

[18]A free/open-source version of the developed SLAM can be downloaded at the following repository :`https://github.com/PolytechAngersMecatroniqueClub/pa_slam`.

[19]`http://wiki.ros.org/hector_slam`

[20]`https://code.google.com/archive/p/tu-darmstadt-ros-pkg/downloads`



(a) Self Balancing Robot.



(b) Ball Balancing Robot.

Fig. 9: Robots based on the IstiABot components.

IV. CONCLUSION

As it was presented in this paper, the IstiABot platform made it possible to meet all the defined constraints about the platform: it is modular, we have a full understanding and documentation of the platform (hardware and software), it can be used by students from the first year to the last one as well as for research work. Thus, it has been considered for student projects (the PID implementation among others) and for research projects (implementation of the SLAM algorithm using ROS). The interest of the robot for research application does not lie into its hardware configuration neither its architecture. It is an interesting platform, mainly because it is mastered. Indeed, most of the time, research in robotics has to be validated with experimenations, and the robot can easilly be modified to fit needed experimental configurations.

CANbus Network is the key element of the versatility of the robots. With exactly the same boards and motors, it was for instance possible to design other robots with different abilities. Two of those robots are introduced in Figure 9. The first one (Figure 9a) is based on inverted pendulum principle. The objective of such a platform is to set up the command that will maintain the robot self balanced (it is naturally unstable). A video of the result can be found here[21], and a simulation of a self-balancing robot (used for primary results) can be downloaded here[22] (Python3 code using OpenGL for rendering.). The second robot (Figure 9b) is a robot that must be able to balance itself over a ball (that is free to move). Despite the change of wheels, the components of this robot remain the same as for the IstiABot and the self-balancing robot. Currently, only the hardware has been done and a simulation has been developped (Python3 code using OpenGL for rendering.) and is availbale here[23].

The IstiABot allows a large panel of student project (mechanics, electronics, computing...), and the possibility to test research algorithms is very comfortable, considering the possibility to customize this robot by adding and removing sensors as needed.

[21] https://youtu.be/G2n9_4nhEaQ

[22] https://github.com/rguyonneau/Self-Balancing-Robot-Python-Simulator

[23] https://github.com/PolytechAngersMecatroniqueClub/IBallBBOT

Finally, an effort has been made to develop this robot (hardware and software) with a free license philosophy. We hope that this can permit to anyone to duplicate and customize this robot, or at least to be helpful for those who want to develop their own specific robots.

ACKNOWLEDGMENT

Thank you to Philippe Lucidarme who helped the robot conception and provided some board schematics. Thank you to Baptiste Hamard who works on the PID implementation for speed control and Vincent Cueille who implemented the first version of the mapping algorithm in the robots with ROS. Finally, thank you to the SAGI (automation and computer sciences) Department of Polytech Angers which subsidized the robots.

REFERENCES

[1] F. M. Lopez-Rodriguez, F. Cuesta, Andruino-A1: Low-Cost Educational Mobile Robot Based on Android and Arduino, Journal of Intelligent and Robotic Systems, 2015, 81:6376.

[2] B. Curto and V. Moreno, Robotics in Education, Journal of Intelligent and Robotic Systems, 2016, vol. 81, no 1, p. 3.

[3] J. M. Gomez-de-Gabriel , A. Mandow, J. Fernandez-Lozano and A. Garcia-Cerezo, Mobile robot lab project to introduce engineering students to fault diagnosis in mechatronic systems, 2015, IEEE Transactions on Education, 58(3), 187-193.

[4] L. Lindner, O. Sergiyenko, J. C. Rodrguez-Quionez, M. Rivas-Lopez, D. Hernandez-Balbuena, W. Flores-Fuentes, F. Natanael Murrieta-Rico and V. Tyrsa, Mobile robot vision system using continuous laser scanning for industrial application, Industrial Robot: An International Journal, 2016, Vol. 43 Issue: 4, pp.360-369.

[5] I. Nielsen, Q-V. DangEmail, G. Bocewicz and Z. Banaszak, A methodology for implementation of mobile robot in adaptive manufacturing environments, Journal of Intelligent Manufacturing, June 2017, Volume 28, Issue 5, pp 11711188.

[6] C. SprunkEmail, B. Lau, P. Pfaff and W. Burgard, An accurate and efficient navigation system for omnidirectional robots in industrial environments, Autonomous Robots, February 2017, Volume 41, Issue 2, pp 473493.

[7] M. Mustafa, A. Stancu, N. Delanoue and E. Codres, Guaranteed SLAMAn interval approach, Robotics and Autonomous Systems. 2018. Vol. 100 p. 160-170.

[8] R. Guyonneau, S. Lagranges, L. Hardouin and P. Lucidarme, Guaranteed Interval Analysis Localization for Mobile Robots, Journal on Advanced Robotics, 2014, Vol. 28 n16 p. 1067-1077.

[9] Robert Bosch, Specification, C.A.N. (1991). Bosch. GmbH, Postfach, 50.

[10] P. Jamieson and Jeff Herdtner, More missing the BoatArduino, Raspberry Pi, and small prototyping boards and engineering education needs them, 2015, IEEE Frontiers in Education Conference (FIE).

[11] J. Sobota et al., Raspberry Pi and Arduino boards in control education, IFAC Proceedings Volumes, 46.17 (2013): 7-12.

[12] K. J. Astrom and T. Hagglund, PID controllers: theory, design, and tuning. Research Triangle Park, NC : Instrument society of America, 1995.

[13] A. Bautin, P. Lucidarme, R. Guyonneau, O. Simonin, S. Lagrange, N. Delanoue and F. Charpillet, Cart-0-matic project : autonomous and collaborative multi-robot localization, exploration and mapping, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, 2013, Tokyo.

[14] J. E. Dennis and D. J. Woods, Optimization on microcomputers: The Nelder-Mead simplex algorithm, New computing environments: microcomputers in large-scale computing, 1987, vol. 11, p. 6-122.

[15] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, O. Von Stryk and U. Klingauf, Robocuprescue 2014-robot league team hector darmstadt (germany), RoboCupRescue 2014, 2014.

[16] S. Kohlbrecher, O. Von Stryk, J. Meyer and U. Klingauf, A flexible and scalable slam system with full 3d motion estimation, In 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (pp. 155-160), IEEE, 2011.