Donnie Robot: Towards an Accessible And Educational Robot for Visually Impaired People

Guilherme H. M. Marques, Daniel C. Einloft, Augusto C. P. Bergamin, Joice A. Marek, Renan G. Maidana
Marcia B. Campos, Isabel H. Manssour, Alexandre M. Amory
PUCRS University, Faculdade de Informática, Porto Alegre, RS, Brazil
{guilherme.marques.002, daniel.einloft, augusto.bergamin, joice.marek, renan.maidana}@acad.pucrs.br
{marcia.campos, isabel.manssour, alexandre.amory}@pucrs.br

***Abstract*—Robotics has been used to draw the attention of young students to computing and engineering. Unfortunately, most hardware and software resources used to teach programming and robotics are not adequate for students with visual impairment (VI). For instance, most programming environments for young students are highly visual and the commercial robotics kits were not built for people with VI. This paper describes the main design requirements so that both the proposed robot hardware and the software can also be used by people with VI. The hardware part is the focus of this paper, a robot called Donnie. It consists of open hardware, a design with low-cost and easy to find parts, and a 3D printed structure. Our main contribution is the presentation of a low-cost Arduino-based robot, compatible with Player robotic framework and integrated with sound feedback and text-to-speech.**

Keywords*—*Educational Robotics, Assistive Robotics, Blind Programmers, Audio-based navigation.

I. Introduction

Robotics have been used as a tool to teach multidisciplinary topics such as mathematics, physics, mechanics, computing and engineering. Robot programming is a useful tool for the development of computational thinking, as well as for solving challenges in the field of motor skills, orientation and mobility (O&M) [1].

Despite these benefits of robotics for O&M and learning, usually there is no focus on people with visual impairment (VI). For example, there are several programming languages for educational purposes based on command blocks and Drag & Drop features, such as Scratch and Gameblox. However, these programming environments are highly graphical and do not provide an accessible interface for visually impaired people.

There are few attempts to use existing robotics programming environments for students with VI [2], [3]. Most of these prior works use Lego Mindstorms with some encouraging results. On the other hand, they also present some shortcomings and deficiencies in terms of usage because the programming environment was mainly built for sighted people and then adapted for visually impaired people.

Thus, the main goal of this paper is to present a robotic programming environment for the practice of computational thinking and training of O&M skills which is inclusive to people with VI. This environment considers assistive and inclusive features for visual impaired people as the number one priority. The proposed system consists of a new educational/assistive robot called Donnie and a software environment that encourages students to study beyond their initial programming skills. Both the hardware and the software architectures are completely based on open technology and they are freely available at Donnie's website . The software architecture, called GoDonnie, is a Logo-based programming language built on top of the Player/Stage [4] robotic framework, running on Linux operating system.

The focus of this paper is on the robot's hardware architecture, which is based on resources commonly used for educational purposes such as Arduino, Raspberry Pi 2 and 3D printed parts. The main unique contributions of this paper can be summarized as:

- the first Arduino-based robot compatible with the Player robotic framework, including extensive documentation and manuals;
- lower hardware cost compared to the Lego Mindstorms;
- an integration of sound feedback and text-to-speech as alternative feedback modes for visually impaired people;
- both hardware and software design based on open technology and 3D printing.

This paper is organized as follows. Section II reviews prior works on the use of robotics for people with visual disabilities. Section III describes the entire system architecture (both hardware and software designs). Section IV demonstrates the functionalities of the robot and its integration with the proposed environment. Section V presents final remarks and future work.

II. Related Work

Accessible programming environments for users with VI have already been subjects of research [5], [6], specially using robots [2], [7], [3], [8]. There are several resources currently

978-1-5386-0956-9/17/$31.00 ©2017 IEEE
http://scratch.mit.edu/
http://education.mit.edu/portfolio_page/gameblox/
https://github.com/lsa-pucrs/donnie-assistive-robot-sw

available to be used for this purpose, such as Lego Mindstorms NXT robots, Bricx Command Center (BricxCC) programming environment, Wiimote for tactile feedback, screen reader and Text Magnifier/Reader such as JAWS and Zoomtext, respectively. Ludi et al. [2] describe the results of the robotics programming workshops, using these tools, developed for participants with different degrees of vision.

Ludi et al. [7] also developed a software called JBrick, for students who are visually impaired. It provides accessibility features for the Lego Mindstorms NXT and was tested with ten visually impaired participants, which demonstrated that they were satisfied in using it. Howard et al. [3] also used a Lego Mindstorms NXT robot kit that was integrated with JAWS screen reader and MAGIC magnifier. The results of a pilot study with nine participants with different degrees of visually impairment showed that the use of multimodes of interaction enabled these students to participate in traditional robot programming processes. Dorsey et al. [8] used BricxCC as a programming interface for the Lego Mindstorms NXT that was also integrated with JAWS and MAGIC and tested with visually impaired students.

Another approach for teaching mobile robot programming users with VI is to use cubic blocks and RFID tags, allowing operation via tactile information [5], [6]. Thus, students can control a mobile robot by positioning wooden blocks on a mat. This system, called P-CUBE, was used to operate a robot composed by an Arduino UNO microcontroller board, a wireless microSD shield, a buzzer, two motors, a gearbox and batteries. It was tested with four visually impaired users and the system was improved with their feedbacks.

Tran et al. [9] proposed an audio programming tool to help visually impaired people. It is based on text-to-speech technology and used for teaching programming, but it was not designed for children or teenage students. There are other works involving programming education for visually impaired students [10], [11], [12], however they are not a robotics-based curriculum.

It is possible to see that Lego Mindstorms NXT is used in several works, mainly for its affordable cost, but usually associated with others tools to provide accessibility. However, accessibility will depend on the associated programming environment, and several of them have limitations that make usage difficult for visually impaired people. P-CUBE's tactile approach is interesting, however the programming commands are very limited. The availability of an audio programming tool with voice commands helps in accessibility, but it does not address the use of robots and it needs to be tested with children and younger people without programming experience.

III. System Architecture

The overall system architecture is illustrated in Figure 1. The student uses a computer (Figure 1(a)) with WiFi connectivity to remotely connect to either the physical robot (Figure 1(c)) or the virtual simulation environment (Figure 1(b)). Both physical and virtual robots are equivalent, except that the physical robot also uses vibration as an assistive aid. The student's computer has the proposed assistive IDE that includes textual interfaces, screen readers and magnification software. The students build the programs in a Logo-based language which translates the commands/sensors to the Player robotic framework. Player sends the commands to the robot and receives robot sensor data to display at the console or in log files. The next sections details the robot's hardware (Section III-A) and software (Section III-B) designs.

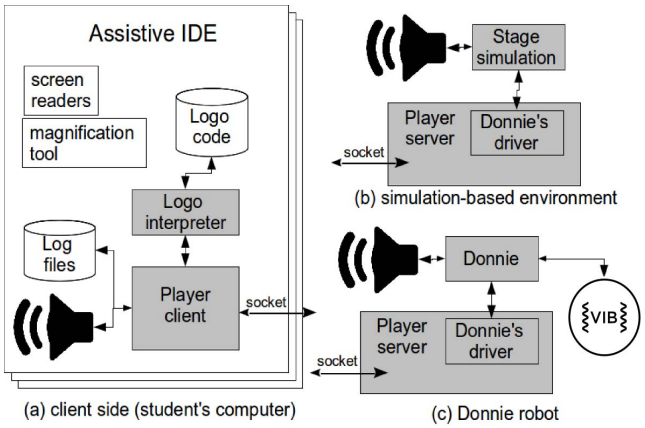


Fig. 1: Donnie's system architecture. Client-side software (a), simulation-based robot (b), physical robot (c).

A. Hardware Architecture

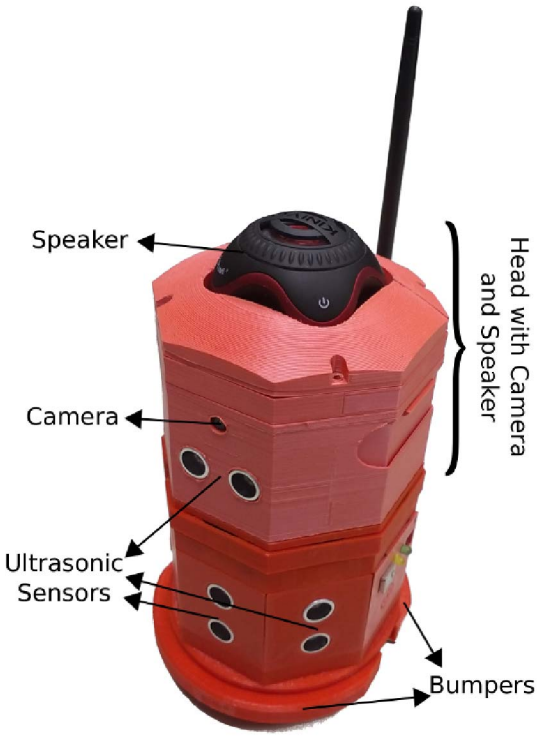
The hardware architecture consists of an open mechanical model, which can be built with a 3D printer, and an open electronic system based on Raspberry Pi 2 (RPi 2) and Arduino. Both boards were chosen due to their availability in the market, low cost, and the size of their developer communities, which eases the addition of new capabilities in the future. Moreover, both boards are used in several educational projects, and thus the schools might already have some of these boards. The rest of this section details Donnie's mechanical model, electronics, and concludes with a cost comparison with Lego Mindstorms.

1) The Mechanical Model: Figure 2 illustrates Donnie's mechanical model and an example of the 3D printed robot. Its main mechanical features include: (*i*) it is a differential robot with two parallel wheels; (*ii*) it has one bumper in the front and another in the rear; (*iii*) it has a head that can turn 180 degrees to scan the surroundings with a camera; (*iv*) it has seven sonar sensors (six in the base and one in the head) for obstacle avoidance; (*v*) it has a modular design, divided into stacks, such that an additional stack can be added on demand.

Currently Donnie has three stacks. The lower stack (the feet) is for the wheels, the motors, the motor encoders and the bumpers. The second stack (the body) has the battery, six ultrasonic sensors and the Arduino board. The third one (the head) has one ultrasonic sensor, the camera, the speaker and the RPi 2 board. Additional stacks can be mounted in between the existing stacks to add new functionalities.

2) The Electronic System: Figure 3 illustrates Donnie's electronic system, which is divided in two main parts: Arduino and RPi 2. While the RPi 2 board does the interface to the robot's peripherals, which are similar to computer peripherals (e.g. camera, speaker, WiFi), the Arduino board does the interface to the IO resources exclusive for robots (e.g. range sensors, motors, encoders, etc).

(a) 3D model.



(b) 3D printed robot.

Fig. 2: Donnie's mechanical model. CAD 3D model (a) and 3D printed robot (b).

The design decision to use two boards (Arduino and RPi 2) instead of only one of them is discussed as follows. It would be possible to implement all peripherals using only the RPi 2 board. However, this would lead to a reduced number of interfaces for future expansions and add-ons. By using an Arduino Mega, there are enough interfaces for several new features, such as a gripper, line following sensors, among other

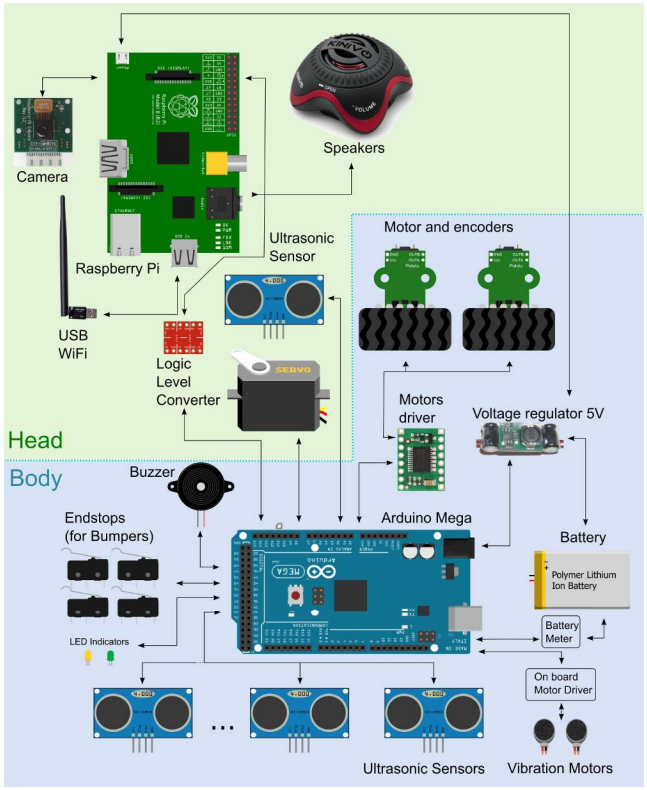


Fig. 3: Donnie's Electronic System divided in two parts: head and body.

possible extensions. Alternatively, there are a few expansion boards for RPi 2 for robotics that could also be used instead of the Arduino Mega. The advantage of Arduino is its availability, large user community, and that it can be removed from the robot and used alone to teach programming [13]. This way, Arduino is a more versatile option than the RPi expansion boards.

On the other hand, there are also several robot proposals using only Arduino boards. The main drawback of these approaches is that they provide limited computing capability, limiting the learning experience. The robot would become obsolete as soon as the student acquired the basic programming skills enabled by an Arduino board. This way, by combining both Arduino and RPi 2 boards the robot becomes modular, expandable, and also with good computing capabilities to enable advanced programming and robotics learning. Both boards are also well documented, with several books about robotics such as [14] and [13].

As illustrated in Figure 3, the Arduino board communicates with the Raspberry Pi 2 board using the serial interface via a level shifter. As illustrated in Figure 3, the Arduino board controls the following Donnie's resources:

1) The DC driver and two motors to move the robot;
2) The motor's magnetic encoders (resolution of 900 counts per revolution) that evaluate the distance traveled by each wheel;

https://www.adafruit.com/products/1940
https://www.piborg.org/picoborgrev

3) The buzzer and the vibrating motor used for assistive interface in case of robot collision, obstacle detection, low battery alert;
4) The LEDs to provide feedback similar as the ones mentioned before, for students with normal vision;
5) The two tactile bump sensors (front and rear bumpers), each one using two switches to detect collisions;
6) The battery sensor used to indicate low battery;
7) The servo motor to implement the head movement for scanning the surroundings with head's sonar and the camera;
8) Seven sonar sensors (including the one in the head) for navigation and obstacle avoidance;
9) The level shifter to adapt the serial voltage between the Arduino and the RPi 2 boards.

The RPi 2 board runs a Linux operating system with the Raspbian 8.0 (Jessie) distribution.

This board runs the Donnie's device driver compatible with Player middleware and it also controls the camera (and its image processing software based on OpenCV), the sound at the speaker and the WiFi connectivity.

Two additional boards were developed to connect the robot's electrical components. The first board is a 3.6 by 4.2 cm board with two layers, used to connect the RPi 2 board with the electronics in the head (ultrasonic sensor and servo motor) and with the Arduino board. The second board is attached to the Arduino board. It is 6.3 by 10.6 cm and has two layers. It contains six ultrasonic connectors, two vibration motor drivers, one battery voltage meter, four digital bumper inputs, two LEDs, one buzzer output, two encoder inputs and a motor driver to control two DC motors. This board has a connector to communicate with the first board via serial interface.

3) The Hardware Cost: The estimated cost to build a Donnie robot is about US$ 285. Table I shows the price of each component used to build Donnie, compared with the price of the equivalent parts of Lego's Mindstorms. The 'X' represents that the Lego does not have an equivalent official part and the '-' represents the item is not needed because it is included into the main part. For example, the Brick Mindstorm, built with an ARM9 microprocessor, has an embedded speaker and WiFi support, thus it is not required to buy these peripherals separately. The amount of plastic required for 3D printing is about 0.44 kg. Assuming the plastic costs US$48 per kilo, the 3D printing cost is US$ 21.

A few accessories would have to be acquired to make Lego's Mindstorms more comparable with Donnie. Even though Donnie still has advantages such as a better processor (able to run more complex codes and image processing algorithms), it has more sensors than Lego's robot and lower cost. The costs from Donnie's parts are based on Pololu's and Adafruit's webstore and the cost for Lego is based on Amazon's webstore.

TABLE I: The price table of Donnie's parts in comparison with Lego's Mindstorms equivalent parts.

	Donnie (U$)	Lego Mindstorms EVE3 (U$)
Motor driver	8.95	-
Motor, encoder and wheel	39.95	-
Micro controller	45.00	349.95
Raspberry pi 2	39.95	X
Servo motor	19.95	27.90
Buzzer	1.49	X
Ultrasonic sensors	17.50	39.95
Ubec	5.98	-
Camera	29.95	X
Usb WiFi*	19.05	-
Two bumpers	7.96	X
Battery	14.86	-
Speaker	12.99	-
Plastic	21.00	-
TOTAL (U$)	**284.58**	**417.8**

B. Software Architecture

This section presents a brief overview of the software architecture of the proposed system in order to understand some key aspects of the project. Thus, since this paper focus on the robot constructional aspects, only the keys features of the software architecture are presented due to the limited number of pages.

A crucial design decision was whether we would use a robotic software framework or not, and which framework to adopt. Most educational robots do not support those frameworks because they are complex for educational purposes, especially for unexperienced users, because they rely on more advanced concepts such as distributed/parallel computing, device drivers, among others. However, professional robotic designs are typically compatible with one or more robotic software frameworks, as a matter of software best practices and compatibility with useful robotics resources (e.g. sensors, drivers, etc). So the challenge was to make Donnie compatible with a software framework while at the same time guaranteeing ease of use by novice users.

The adopted approach is to add an easy-to-use frontend on top of the framework so that the new users do not have direct access to the framework. This frontend is a terminal-like interpreter of a new programming language called GoDonnie. Details of this programming language is out of this paper's scope, but it is inspired on Logo and designed to be easy to learn. The advantage of using this frontend approach is that, although the framework is hidden, it is still available for modification when the user reaches the required coding experience. In fact, this layered software architecture is built such that it invites the users, as they evolve in their programming skills, to gradually start coding also for the lower layers, improving their understanding of a complete computing software stack. This includes, for instance, multiple concepts of Computer Science, such as embedded programming, serial protocol, hardware/software interface, networked communication, topics of artificial intelligence (e.g. path planning, autonomous behavior), computer vision (e.g. blob finder, tag recognition, pattern recognition), and advanced robotics (e.g. visual odometry, localization, mapping, multi-agent systems). In conclusion, although at this stage of Donnie's research project we focus on teaching programming to young users, the proposed system could also be used for high-school and graduate courses.

Current survey papers [15], [16] list dozens of robotic frameworks such as ROS, Orocos, Player, among others. Currently, ROS is the most used, documented, and with the biggest developer community [17]. However, ROS is complex for educational purposes, specially for users with little or none programming background. Moreover, it demands updated/-modern desktop computers and embedded processors, which might not be available in most schools.

Player, on the other hand, is much simpler in several aspects. First, it is based on a simpler client-server software design approach, compared to peer-to-peer, enabling simpler software design and hardware abstraction. The computational requirements to run Player are low, which would fit nicely on older embedded processor, such as RPi 1. The source code is small (about 5 MBytes), the compilation process is straight-forward (few dependencies), there are several examples of robot drivers (there are dozens of robots compatible with Player), ready-to-use drivers for most common robotic resources (e.g. laser rangers, cameras, etc), and it uses a simple textual configuration file to describe the robot drivers, which is a relevant feature for assistive purposes. Player supports several programming languages such as Ruby, Python, C, C++, Java, Matlab, Escala, which means that the programmer is not restricted to a single programming language. Player is also integrated to Stage, a lightweight robotic simulator which we extended to give sound feedbacks for visually impaired users. Every action of the robot is associated with a different sound. This way, when the physical robot is not available, the Stage simulator can be used by both visually impaired and sighted users. Finally, Player/Stage have been very active in the research community with currently more than 1500 citations just to the first paper [4]. Details about Donnie's integration with Player/Stage are out of the scope of this paper, once again, due to the limited number of pages.

Listing 1 presents a simplified view of the firmware structure running in the Arduino board. The main initial loop (lines 1 to 5) waits for an initial configuration packet sent from the Player driver to the Arduino via the serial port. The robot does not start until this configuration packet is received. However, the power status is checked to alert the battery status.

Listing 1: Arduino-based firmaware.

```
1  do{
2    sendRequestConfig() //request a robot config from driver
3    cmd = readCommand() //receive robot config
4    power_update() //check battery status
5  }while(cmd != CONFIGPACK) // wait until robot config received
6
7  // main loop
8  while(TRUE) {
9    cmd = readCommand() //Read incoming command from serial port
10   processCommand(cmd) //Execute requested command
11   updateSensors() //Read each sensor and update variables
12   updateIndicators() //Update leds, vibs and buzzer indicators
13   sendData() //Send data to serial port
14   updateTicks() //Increment the tickCnt each 1ms (1000us)
15 }
```

The main loop in Listing 1 (lines 8 to 15) performs the robot control. It initially reads incoming packets from the serial port (line 9), executes the commands (e.g. move commands, line 10), updates the sensor readings into the internal memory (line 11), updates the indicators (LEDs, buzzer, vibration motors) based on the command and sensor readings (line 12), and sends the new data via serial port to the Player Driver (line 13). The last line updates counters that control the frequency to send the serial messages.

The Enlace-level of the serial messages presented in Figure 4 has two constants bytes of header, one byte of packet length, one byte for message types, variable number of bytes for the payload, and a final byte with checksum. Each functionality in the Arduino board has a corresponding message type.

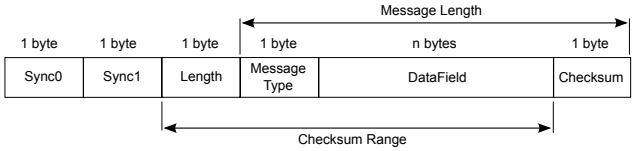


Fig. 4: Enlace-level protocol packet format.

When the user adds a new functionality to the robot, he/she has to define a new message type and adapt both the firmware and the driver to handle this new message. The firmware and driver codes have comments to give clues to the user as in where to change.

IV. Results

The Donnie's odometry error was evaluated while it traveled a 30x30 cm square course. Six separate runs where evaluated and the odometry error (O_e) of each run is calculated as:

$$O_e = \frac{\sum_{i=1}^{N} \sqrt{(x_{ei} - x_{ri})^2 + (y_{ei} - y_{ri})^2}}{N} \quad (1)$$

where N is number of recorded points, x_{ei} is the estimated X, y_{ei} is the estimated Y, x_{ri} is the real (Ground-Truth) X, and y_{ri} is the real (Ground-Truth) Y.

In figure 5 we show Donnie's movement. The best (lowest error) and worst (highest error) are depicted in red and green, and it's ground truth is depicted in blue. The O_e of the desired trajectory against the actual trajectories is between 2.28% and 3.98%. Donnie's odometry tests returned satisfactory results since its current version does not include low level control algorithms that controls the motors velocity. We expect better odometry results once the control algorithms are fully integrated into Donnie's next firmware. As future work, we are also including inertial sensors and visual odometry algorithms to obtain a better position estimation.

The origin of the odometry error is that Donnie's wheels have irregular shapes, having only a part of its tires touching the ground. This can be considered as an uncertainty in the system, making the distance between the center of the wheels different than it should be, inducing an error on the odometry calculations [18].

The second test evaluated the reaction time required for Donnie to stop moving when an obstacle is detected within 15 cm. Our latency analysis method consists of the time the system takes to send the distance value to the Raspberry until the Arduino gives the command to stop the motors. This

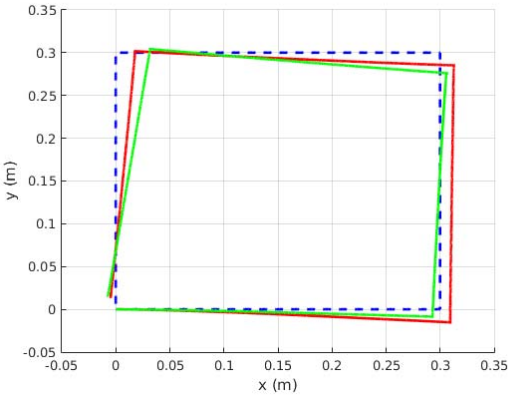


Fig. 5: Donnie's trajectory as recorded by the best odometry (red), worst odometry (green) and the Ground Truth (blue).

latency varies from 81 to 183 ms. This delay causes an error of about 1 cm in the distance from the robot to the obstacle.

A similar method was used to evaluate the latency to detect impact with the bumpers until the motors stop. The delay of this detection varies from 75 to 178 ms over 20 executions.

V. Conclusions and Future Work

This paper presented a robotic programming environment that has been designed for both sighted and visually impaired people. This environment (both hardware and software) has sound/vibration feedback, a low hardware cost, use open technology, and is well documented. The paper focused on the hardware design and the firmware, while the remaining of the environment is just briefly introduced.

The odometry estimation error will be reduced in the next version when we add PID controllers and sensor fusion with inertial sensors and optical flow methods, with Donnie's camera. In the near future, we plan to evaluate whether the sound and vibration features in the robot are sufficient feedback for visually impaired users to estimate the robot state and pose. The proposed environment works both with virtual robots or with physical robots, and we plan to evaluate which one is more effective for the learning processes. We also plan to evaluate the effectiveness of both the front-end interpreter and the GoDonnie programming language, which were just briefly mentioned in this paper. Finally, we plan to evaluate if the proposed environment could also help the students to improve their orientation and mobility skills.

Acknowledgment

This work was partially supported by PUCRS via projects BPA-PEC-DES 11/2015, 11/2016, and 06/2017; FACIN's 1/2013; CNPq project 442126 and via PIBIC undergraduate student grants.

References

[1] S. Azevedo, A. Aglaé, and R. Pitta, "Minicurso: Introdução a robótica educacional," in *62o reunião anual da Sociedade Brasileira Para o Progresso da Ciência (SBPC)*, 2010.

[2] S. Ludi and T. Reichlmayr, "The use of robotics to promote computing to pre-college students with visual impairments," *ACM Transactions on Computing Education (TOCE)*, vol. 11, no. 3, p. 20, 2011.

[3] A. M. Howard, C. H. Park, and S. Remy, "Using haptic and auditory interaction tools to engage students with visual impairments in robot programming activities," *Learning Technologies, IEEE Transactions on*, vol. 5, no. 1, pp. 87–95, 2012.

[4] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, 2003, pp. 317–323.

[5] S. Kakehashi, T. Motoyoshi, K. Koyanagi, T. Ohshima, and H. Kawakami, "P-cube: Block type programming tool for visual impairments," in *Proceedings of the Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, 2013, pp. 294–299.

[6] S. Kakehashi, T. Motoyoshi, K. Koyanagi, T. Oshima, H. Masuta, and H. Kawakami, "Improvement of p-cube: Algorithm education tool for visually impaired persons," in *Proceedings of the IEEE Symposium on Robotic Intelligence In Informationally Structured Space (RiiSS)*. IEEE, 2014, pp. 1–6.

[7] S. Ludi, L. Ellis, and S. Jordan, "An accessible robotics programming environment for visually impaired users," in *Proceedings of the 16th international ACM SIGACCESS conference on Computers & accessibility*. ACM, 2014, pp. 237–238.

[8] R. Dorsey, C. H. Park, and A. M. Howard, "Developing the capabilities of blind and visually impaired youth to build and program robots," in *Proceedings of the CSUN Annual International Technology and Persons with Disabilities Conference*, 2013, pp. 55–67.

[9] D. Tran, P. Haines, W. Ma, and D. Sharma, "Text-to-speech technology-based programming tool," in *Proceedings of the 7th WSEAS International Conference on Signal, Speech and Image Processing*, 2007, pp. 173–176.

[10] K. G. Franqueiro and R. M. Siegfried, "Designing a scripting language to help the blind program visually," in *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, 2006, pp. 241–242.

[11] S. K. Kane and J. P. Bigham, "Tracking stemxcomet: teaching programming to blind students via 3d printing, crisis management, and twitter," in *ACM technical symposium on Computer science education*, 2014, pp. 247–252.

[12] M. Konecki, R. Kudelić, and H. Gjoreski, "Guidl ia: An intelligent assistant for aiding visually impaired in using guidl," in *Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1114–1117.

[13] R. Grimmett, *Arduino Robotic Projects*. Packt Publishing, 2014.

[14] ——, *Raspberry Pi Robotics Projects*, 2nd ed. Packt Publishing, 2015.

[15] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," *Autonomous Robots*, vol. 22, no. 2, pp. 101–132, 2007.

[16] A. Elkady and T. Sobh, "Robotics middleware: A comprehensive literature survey and attribute-based bibliography," *Journal of Robotics*, vol. 2012, 2012.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[18] R. Siegwart and I. R. Nourobakhsh, *Introduction to autonomous mobile robots*. The MIT Press, 2004, pg 53, Section 3.2.3.