
TURTLEBOT 3 AS A ROBOTICS EDUCATION PLATFORM

Robin Amsters

KU Leuven,
Department of Mechanical Engineering,
3000 Leuven, Belgium
robin.amsters@kuleuven.be

Peter Slaets

KU Leuven,
Department of Mechanical Engineering,
3000 Leuven, Belgium
peter.slaets@kuleuven.be

October 25, 2019

ABSTRACT

Teaching robotics to engineering students can be a challenging endeavor. In order to provide hands-on experiences, physical robot platforms are required. Previously, obtaining these platforms could be expensive, and required a lot of technical expertise from teaching staff. However, more recent models address these issues, therefore providing more opportunities for hands-on sessions. In this paper, we describe how we used the Turtlebot 3 mobile robot in master courses at KU Leuven. We provide an overview of the main functionalities, and suggest a number of improvements to further lower the learning curve for students. Additionally, we elaborate on the curriculum and learning outcomes of two courses that utilized Turtlebots in practically oriented sessions..

Keywords Turtlebot · Robotic Operating System · Educational robotics · Mobile robotics · Project-based learning

1 Introduction

When designing the curriculum for a particular course, one can include a variety of different learning activities. Lectures are often used to convey theoretical concepts to large groups of students, while practically oriented lab sessions or projects can encourage students to apply their knowledge to real-world problems. Standardized test scores were found to be correlated to the frequency of hands-on learning [1]. In science education specifically, these practically oriented learning activities can take the form of simulations, remote experiments or hands-on experiments. An extensive literature survey by Ma and Nickerson [2] concluded that there are proponents and opponents to all types. However, some form of experimental work may offer benefits over a pure simulation approach. Sauter et al. [3] found that students who conducted remote labs were more engaged in the experiment they were performing than those who used computer simulations. Additionally, remote lab users wrote higher-quality research questions and were more open to performing repeated experiments. In robotics education, a physical robot is required for hands-on or remote experiments. However, selection of an educational robotics platform is not straightforward. In the past, one often had to choose between purchasing expensive models, or building a custom platform from scratch. Table 1 provides an overview of several wheeled robotics platforms, with release dates ranging from 1995 up to 2018. This list is not meant to be exhaustive, rather, it is intended to show a trend. One can observe that while robotics education has become significantly more affordable, platforms with sensors that enable functionalities such as mapping and autonomous navigation are still relatively expensive.

When selecting a platform for introductory robotics courses at KU Leuven Campus group T, we set forward the following requirements:

- **Wheeled robot:** Most of the envisioned practical sessions focus on wheeled robotics, rather than humanoids or robot arms.
- **Low cost:** Due to the large number of students, the cost per robot needs to be limited.
- **Low learning curve:** The introductory mobile robotics course that is to make use of the platforms, only has three lab sessions (see Section 4). The time required to get familiar with the platforms should therefore be minimized, in order to maximize the available learning time.

Table 1: Overview of wheeled robot platforms

Robot	Year of release	Cost [€]	Target audience	Sensors
Pioneer 3DX [4]	1995	3233.00	Higher education	sonar / encoder
Khepera [5]	1999	2999.00	Higher education	sonar / IR
Amigobot [6]	2001	3150.00	Higher education	sonar
Scribbler 3 [7]	2010	220.86	Secondary education	IR / ambient light / encoder
Turtlebot 2 [8]	2012	1673.00	Higher education	RGB-D / gyroscope / encoder
Edison [9]	2014	51.58	Secondary education	IR / sound / ambient light
Cozmo [10]	2016	199.99	Secondary education	camera / IMU / IR
Turtlebot 3 (burger) [11]	2017	585.95	Higher education	LIDAR / IMU / encoder
Turtlebot 3 (waffle) [11]	2017	1399	Higher education	LIDAR / RGB / IMU / encoder
Rosbot 2.0 [12]	2018	1661.00	Higher education	RGB-D / LIDAR / encoder / IR

- **Advanced perception:** While simpler robots are capable of tasks like line-following, this is not sufficient for the envisioned courses and projects. In master level courses, more advanced perception is required. Some form of depth perception should at least be available (e.g., in the form of an RGB-D camera, or a laser scanner).
- **Standard software framework:** It is imperative that the skills students acquire are as broadly applicable as possible. By using a standard software framework, students are more likely to use the same software again in their later career. Additionally, such a standard framework usually has more third party software available, broadening the possibilities of the platform.

Based on our requirements, the Turtlebot 3 burger stands out from the others, as it offers advanced sensors for a significantly lower price. Moreover, it makes use of the "Robotic Operating System" (ROS), which is an open source framework. In the rest of this work, we will discuss our experiences in using this platform for robotics education. The paper is structured as follows; Sections 2 and 3 provide an overview of the hardware and software, and the additions that were made. We made use of these platforms in several educational projects, two case studies are described in Sections 4 and 5. Section 6 discusses the learning outcomes, before and after the introduction of new robot platforms. Finally, a conclusion is presented in Section 7.

2 Hardware Overview

Turtlebots are sold in 2 main configurations, referred to as the "burger" and "waffle" models (see Fig. 1). The waffle model is larger, has an additional camera sensor and is significantly more expensive than the burger model. As we had no need for a larger robot or the camera, we opted for the burger model. The main hardware components of the platform are:

- **LIDAR:** a 360° laser scanner is mounted on top of the robot. This sensor transmits a rotating laser, which reflects of nearby obstacles. By using the reflection time, distances to nearby objects can be obtained, which can be used for mapping or obstacle avoidance.
- **Raspberry Pi:** one level below the LIDAR is a Raspberry Pi 3 (RPI) single-board computer. Since its introduction a few years ago, the Raspberry Pi has gained incredible popularity among hobbyists and researchers due to its small size, low cost and power use. It is most useful in applications that require a computer, but not a large amount of processing power. On the Turtlebot, it reads data from the sensors and communicates with a 'master' pc, which does the actual computations.
- **OpenCR:** one level below the RPi, the OpenCR hardware control board is located, which is based around an Arduino Uno. The board contains an Inertial Measurement Unit (IMU) with a three axis accelerometer, gyroscope and magnetometer. Its main function is to connect the Raspberry Pi to the motors and sensors, such that data can be read and commands can be sent. Additionally, the board provides power connections for all components.
- Finally, on the lowest level, we find the **motors** and the 11.1V Lithium Polymer (LiPo) **battery**. Both the burger and the waffle models have a differential drive configuration, which means that they have 2 independently

driven motors and 1 caster wheel for stability. Dynamixel XL430-W250 servo motors are used to drive the wheels.

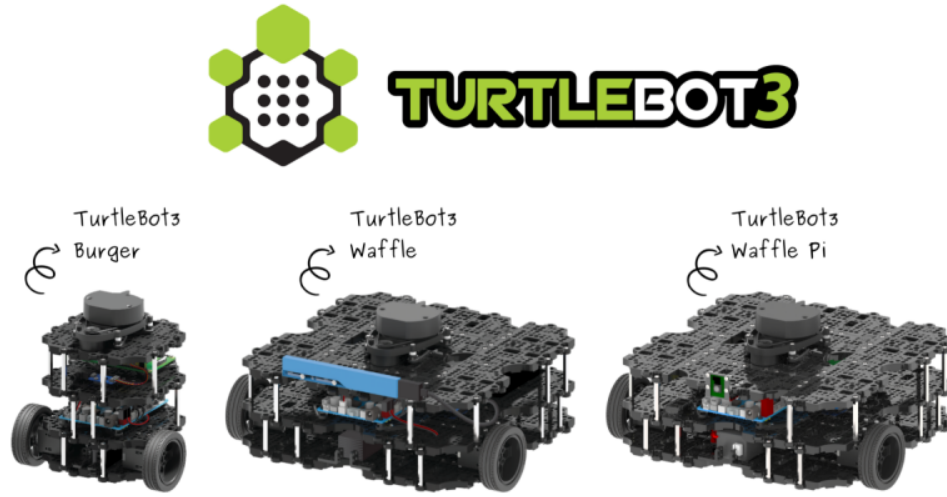


Figure 1: Overview of the Turtlebot 3 models [13]

3 Software Overview

The operating system on the Raspberry Pi is Ubuntu MATE 16.04, a lightweight version of the Linux distribution Ubuntu. This version is better suited for single-board computers due to the lower hardware requirements compared to the complete desktop version [14]. A number of modifications were made to the default version to make it more suited to our needs:

- The Raspberry Pi functions as a wireless access point (AP). Previously, we utilized a central AP, to which all robots and control computers were connected. However, this infrastructure resulted in unstable connections due to the large number of devices connected to the router. If the robots function as their own AP, less devices need to be connected to each network. Therefore, the requirements of such a network are lower. We found that the individual networks did not interfere with each other significantly. Even when 10 robots were used in the same room, connections remained stable. Another advantage of this approach is that students do not need to stay in the same room to test their programs. They can temporarily move to a more open space if needed. Turning a Raspberry Pi into a wireless AP was achieved with the help of a few online tutorials [15][16]. When connected to the robot WiFi, a Secure Shell (SSH) connection can be established on a fixed IP address. The first connected device will also be assigned a fixed IP address.
- Superfluous software such as LibreOffice, Thunderbird, etc. was removed. Additionally, the Pi was set to boot without a graphical interface.
- A number of Bash aliases were defined, which replace frequently used lengthy commands with shorter versions. For example, the complete command to initialize sensor communication on the robot is:

```
roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

which we replaced by:

```
bringup
```

Besides a Linux distribution, the Robotic Operating System was also installed on the SD card of the RPi. ROS is an open-source, meta-operating system used in robotics research and industry [17]. It provides functionality like hardware abstraction, localization, navigation, visualization, etc. in a standard communication structure. ROS works via a publisher-subscriber interface. Programs are broken up into smaller units called 'nodes'. Nodes communicate with each other through 'topics'. A node can make information available by 'publishing' it on a topic, in the form of a 'message' (ROS data formats). Another node can then 'subscribe' to receive this information. This approach enables modularly built programs and abstraction. While ROS simplifies getting started in robotics education and research, the learning

curve can still be quite steep. Therefore, we also make use of the Robotics System Toolbox, which enables publishing and subscribing to ROS topics in MATLAB [18]. At our campus, students already complete several projects with MATLAB before starting their Masters degree. Therefore, it is preferred over other ROS interfaces such as C++ and Python. A custom toolbox was developed as a final abstraction level for the mobile robotics exercise sessions (Section 4) and the control systems project (Section 5). The focus of these courses is not to learn ROS, therefore the internal software processes of the robot are abstracted away. The custom code, the documentation and the exercises can be found on our GitHub page¹.

The resulting software architecture is shown in Fig. 2. The robot acts as a slave to a master PC on which the program is executed. A 'roscore' (central ROS process that connects nodes to each other) needs to be started manually by executing a command on the RPi over an SSH connection. The Robotics System Toolbox starts a similar process in the background on the master PC. Following initialization, the two processes share data, which can be used on either the PC or the RPi. The use of the MATLAB toolboxes is optional, as the robot can also be directly controlled via a roscore on the master PC. This is the case in, for example, master theses that make use of these robot platforms. These students will work on their project much longer and therefore have more time to gain a better understanding of ROS.

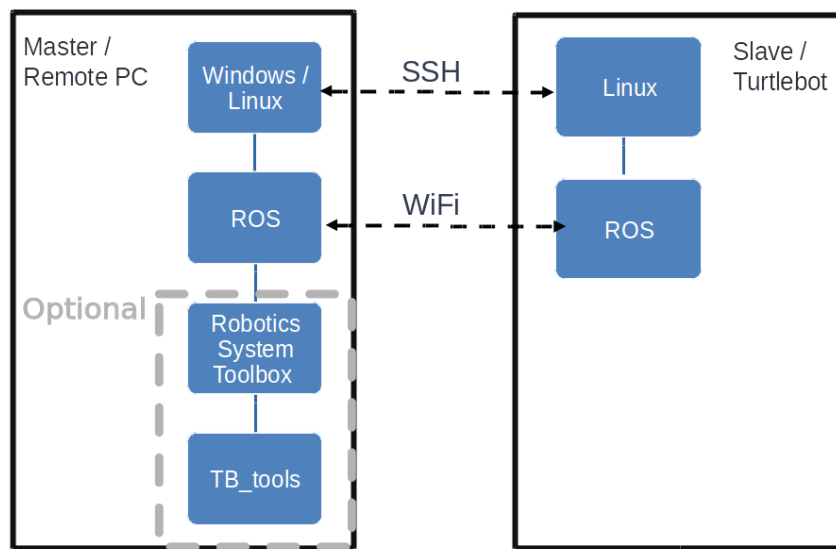


Figure 2: Software architecture

4 Example Use Case: Mobile Robotics Exercise Sessions

As a first use case, we will discuss the autonomous vehicles course offered at KU Leuven Campus Group T. This course consists of an introduction to robotics, and an introduction to mobile robotics. In the introduction to robotics chapter, concepts such as workspace analysis, forward and reverse kinematics are discussed, while the mobile robotics course focuses on autonomous navigation. Course material is based on the book "Robotics, Vision and Control" by Peter Corke [19]. The associated MATLAB toolbox is also used [20].

To enable autonomous navigation, a robot needs to know where it is. Therefore, localization is the first chapter of the course and is discussed in depth. Dead-reckoning is introduced first as it is arguably the simplest the simplest form of localization. By integrating displacement measurements, one is able to acquire a position estimate relative to the initial position. However, every measurement has a slight error, therefore the overall error only grows over time. Nonetheless, it is a useful source of information, as it is generally always available due to the use of proprioceptive sensors (such as IMU's and encoders). To combat drift, additional sources of information are needed. The Kalman filter is discussed as a sensor fusion algorithm. Additionally, the non-linear Extended Kalman Filter is introduced. Besides localization, an autonomous robot needs to be able to plan a trajectory between the starting location and the destination. Therefore,

¹https://github.com/RobinAmsters/GT_mobile_robotics

several of the most used path planning algorithms (A*, D*, distance transform, etc.) are covered in the lectures and the exercise sessions. The special case of a robot that does not need a specific plan, is also discussed. These robots make use of so called 'reactive navigation', and react directly to sensor input (such as driving towards a light, or following a line) [19]. Finally, an autonomous robot needs to be able to execute its plan. Path following falls into the category of control theory, and is therefore not a part of this course.

The Turtlebot platforms are used during the lab sessions of the mobile robotics course. Students receive three such lab sessions. In order to make the most of the available time, students are asked to prepare each session beforehand. And receive a few questions to turn in at the beginning of each session. The majority of their grade is based on these preparations. Labs are organized as follows:

1. **Introduction and dead reckoning:** In the very first session, students get acquainted with the robots and software. Basic tasks such as driving in a straight line or an ellipse are to be performed by the robots. Additionally, the concept of dead-reckoning is treated in a number of simulation and practical exercises. The disadvantages of drift are discussed. Finally, students gain experience with so called 'error ellipses', which are visual representations of uncertainty on a position estimate.
2. **Kalman filter:** Drift is one of the main disadvantages of dead-reckoning based localization. The Kalman filter and its derivatives can be used to include more sensor data into the position estimate, and thus compensate for this drift. In the second session, students design a linear Kalman filter, derive state transition and observation models for an Extended Kalman Filter (EKF) and experiment with the effects of different types of sensor noise on pose estimation.
3. **Path planning:** In this session, students test different path planning algorithms that were discussed in the lectures. They are then asked to compare their advantages and disadvantages for certain situations. Finally, students design a program that makes the Turtlebot follow a wall. Braitenberg vehicle type behavior is used to achieve this task [21].

5 Example Use Case: Embedded Control Systems Experience

In this second use case, we discuss how we made use of the robots in a guided project that was part of an Embedded Control Systems (ECS) course. We refer to this project as an 'experience'. Students learn the theoretical concepts in a lecture format. Several control strategies are covered in these lectures. The more classical approach of PID controllers and design with the Ziegler-Nichols method is discussed. Additionally, more modern control schemes in the Z-domain or in state space are also covered [22][23]. In exercise sessions, students practice designing the controllers discussed in the lectures on paper or in a simulation environment. MATLAB is used for these simulations. Finally, the experiences require students to put concepts from the lectures and exercise sessions into practice. The aim of these experiences is to independently design a controller for a physical system. In groups of 2 or 3 students, different parts of the control loop are built according to a milestone system. This means that the overall assignment is broken up into parts, each of which can grant a percentage of the total grade. After completing one of these parts, teaching staff checks progress, and gives a pass/fail on the current milestone. A pass allows students to progress to the next milestone, a fail means that current work is not sufficient, and more time on the current milestone is needed. This system provides students with regular feedback, and allows major flaws to be detected early. At the end of the project, students present their work with a demonstration and turn in a written report.

The campus building is constructed with a spiral walkway in the center (see Fig. 3), the overall assignment of the experience sessions is to follow this spiral with the Turtlebot. The robot should keep a constant distance relative to the wall whilst driving. The overall control loop is shown in Fig. 4, the following sections will explain the different parts in further detail.

- M1: Motor model identification:** The servo motors of the Turtlebot have built-in PID controllers, which regulate the speed of each individual wheel. In the very first sub-assignment, a model for the DC motors with PID controllers is identified. From the step response, students can derive the transfer function.
- M2: Speed control:** Using the transfer function (**M1**), students are asked to design a controller, which takes wheel speed as an input, and outputs an actuator command such that the speed follows a given reference. Next, individual wheel speeds are translated to a forward (v) and rotational (w) speed of the robot. The overall controller therefore takes v and w as an input, and internally translates these to velocity commands for the left and right wheel. By writing the controller inputs this way, we found it easier to design the distance controller



Figure 3: Spiral walkway inside the Group T campus

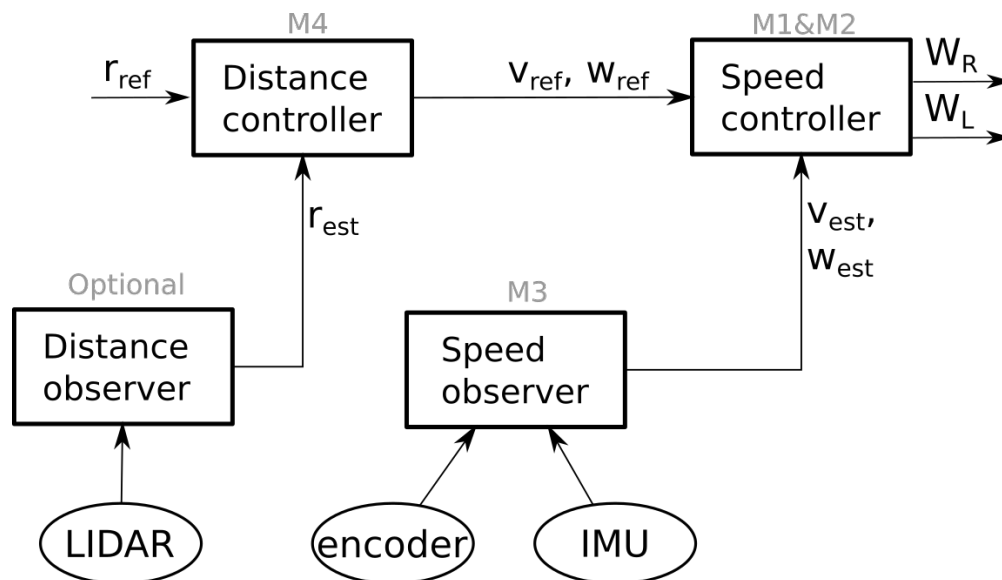


Figure 4: Overview of the control loop and milestones of the project

(M3). Students are allowed to use any control scheme they know of. However, all groups designed some form of PID controller.

M3: Distance control The final part of the overall control loop is a controller that allows the robot to drive at a constant distance relative to the wall of the spiral. It will take LIDAR measurements as an input (which may have to be processed first) and return a forward and a rotational speed as an output. These speeds are then used as an input for the speed controller (M2). In this stage, students are asked to use a state feedback controller, such that they gain hands-on experience in designing this type of controller as well.

M4: State estimator It is possible to control the speed of the motors based on just encoder measurements. However, sensors are not perfect. In order to reduce inaccuracies, students are asked to include a state estimator into the control loop. An estimator can filter noisy sensor values and include multiple sources of information, thus improving accuracy and robustness.

6 Evaluation of learning outcomes

Table 2 summarizes performance indicators from the academic years before (2017) and after (2018) the introduction of the Turtlebots. In the academic year 2017 and before, robot platforms were already used. Therefore, the change between 2017 and 2018 is not the introduction of physical platforms, rather, the platforms used are now much better suited for teaching. For the Embedded Control systems course, other factors changed as well between the academic years (different teacher, course content and project). Therefore, only data for the mobile robotics course is included. Table 2 shows the median final scores per academic year. These scores are based on the exercises of the exam, as the theory questions should not be influenced by the use of robot platforms. Scores for Mobile robotics increased by 2%. However, the correlation between the lab scores and the exam scores is very low. While the correlation increases in 2018, the low value may indicate that better lab performance may not necessarily influence exam scores. Therefore, we can not make any definitive conclusions about the impact of these new robot platforms on students' grades.

Table 2 also shows the passing rate of students. That is, the percentage of students who pass the course on their first examination attempt. For both courses, the passing rate increased by approximately 4%, reaching almost 90% for the Mobile Robotics course. However, this increased passing rate might not be completely attributed to the introduction of new robots. Small fluctuations are always possible. Therefore, a longer time period should be observed before drawing conclusions.

Table 2: Learning outcomes before and after Turtlebots

Academic year	2017	2018
Number of students	60	65
Median score	60%	62%
Exam participation	100.00%	96.92%
Correlation lab/exam	0.13	0.20
Success rate	85.00%	89.23%

6.1 Mobile Robotics

Through an anonymous feedback form, students were asked to rate the course based on a number of questions. 10 students responded to this request. With the maximum score being 6, the question "The things I learned in this course are relevant for my education." scored 4.9. The question "I was satisfied with the quality of teaching in this course." received an average score of 4.8. Figure 5 shows the distribution of the received responses. When asked for feedback in person, many students requested more sessions to expand on the discussed topics. Contrary to previous years (when Turtlebots were not used), students had no negative comments on the platforms themselves.

6.2 Embedded Control Systems

No anonymous feedback forms are available for this course. The university organizes course evaluation, and not every course is evaluated every year. Nonetheless, students were still asked for their feedback in person. This feedback was also positive. Students appreciated the hands-on experience, as they found implementing a control loop on a physical system to be quite different from designing one based on a theoretical model. The project was, however, deemed more challenging than initially estimated. Particularly the design of the state feedback controller was the most difficult milestone.

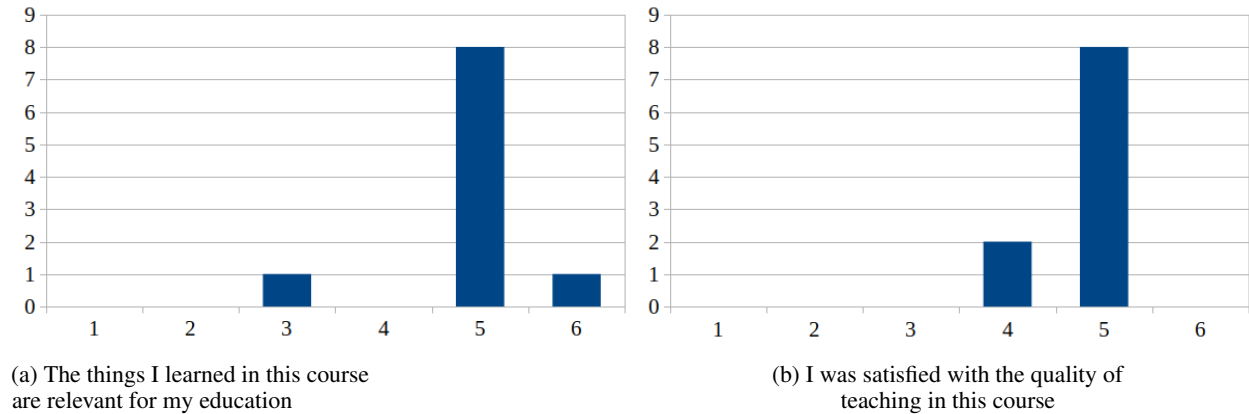


Figure 5: Student responses to anonymous survey

7 Conclusion

In this paper, we discussed our experiences in using the Turtlebot 3 as a robotics platform in higher education. We propose a number of improvements to the software, and discuss how we used the robots in two different courses. We present performance indicators before and after the introduction of the new platforms. It is challenging to draw a definitive conclusion about the effect of these platforms on student performance. Feedback from students and teaching staff has, however, been overwhelmingly positive. Their experience in using the robots has significantly improved compared to previous years. In the future, we would like to improve the platforms further by implementing a system to automatically update the software. Such a system would make the platforms easier to maintain. Additionally, for the control systems course, we would like to offer students different kinds of projects to choose from, instead of one common track.

Acknowledgments

Robin Amsters is an SB fellow of the Research Foundation Flanders (FWO) under grant agreement 1S57718N.

References

- [1] Patricia M Stohr-hunt. An Analysis of Frequency of Hands-on Experience and Science Achievement. *Journal of Research in Science Teaching*, 33(1):101–109, 1996.
- [2] Jing Ma and Jeffrey V. Nickerson. Hands-on, simulated, and remote laboratories. *ACM Computing Surveys*, 38(3):7–es, 2006.
- [3] Megan Sauter, David H. Uttal, David N. Rapp, Michael Downing, and Kemi Jona. Getting real: the authenticity of remote labs and simulations for science learning. *Distance Education*, 34(1):37–47, 2013.
- [4] CYBERBOTICS Ltd. Webots documentation: Adept’s Pioneer 3-DX, 2019.
- [5] F Mondada, E Franzi, and Guignard A. The Development of Khepera. In *Proceedings of the First International Khepera Workshop*, pages 1–21, Paderborn, 2015.
- [6] Sanco Middle East LLC. AmigoBot - Sanco Middle East LLC.
- [7] Parallax. Scribbler 3 (S3) Robot | 28333 | Parallax Inc, 2018.
- [8] Yujin robotics. TurtleBot2.
- [9] Microbric. Edison Programmable Robot - Ideal for school classroom education.
- [10] Anki. Cozmo | Meet Cozmo, 2018.
- [11] Robotis. TurtleBot3.
- [12] Husarion. Husarion - Robot development made simple, 2019.
- [13] ROBOTIS. TurtleBot 3.

- [14] Ubuntu Mate. About Ubuntu Mate, 2015.
- [15] Edo Scalafiotti. Turn a RaspBerryPi 3 into a WiFi router-hotspot – Edo Scalafiotti – Medium, 2016.
- [16] Phil Martin. Using your new Raspberry Pi 3 as a WiFi access point with hostapd.
- [17] Jason M O’Kane. A gentle introduction to ros, 2014.
- [18] Mathworks. Features - Robotics System Toolbox - MATLAB & Simulink.
- [19] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, volume 73. Springer Berlin Heidelberg, 2011.
- [20] P I Corke. A robotics toolbox for MATLAB. *IEEE Robotics Automation Magazine*, 3(1):24–32, 1996.
- [21] Valentino Braitenberg. *Vehicles: Experiments in synthetic psychology*. MIT press, 1986.
- [22] Karl Johan Aström and Richard M Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.
- [23] T Leondes Cornelius. *Digital control systems implementation and computational techniques*, 1996.