การจำลองการทำงานของหุ่นยนต์ด้วย Webots: วิธีการประหยัดต้นทุนสำหรับการศึกษาในระดับมหาวิทยาลัย

Simulating Robot Operations with Webots: A Cost-Effective Approach for University Education

Shutchon Premchaisawatt[1] Krit Tashoo[2] and Narong Boonsaner[3]

Department of Industrial Technology, Faculty of Industrial Education,

Rajamangala University of Technology Isan, Khon Kaen Campus[1, 2, 3]

E-mail: shutchon.pr@rmuti.ac.th[1], krit.ts@rmuti.ac.th[2], narong.bo@rmuti.ac.th[3]

บทคัดย่อ

บทความนี้กล่าวถึงความท้าทายในการจัดการศึกษาด้านหุ่นยนต์อย่างครอบคลุมในระดับมหาวิทยาลัย โดยเฉพาะอย่างยิ่งเมื่อคำนึงถึงต้นทุนสูงที่เกี่ยวข้องกับแขนกลระดับอุตสาหกรรมและการบำรุงรักษา บทความนี้เสนอ Webots ซึ่งเป็นแพลตฟอร์มจำลองหุ่นยนต์ที่มีประสิทธิภาพและเข้าถึงได้ง่าย เป็นทางออกที่เป็นไปได้ บทความเปรียบเทียบ Webots กับซอฟต์แวร์จำลองอื่น ๆ เช่น Gazebo Coppeliasim OpenRoberta และ MATLAB Robotics Toolbox โดยเน้นจุดแข็งในด้านความเป็นมิตรต่อผู้ใช้ คุณสมบัติที่ครอบคลุม และความคุ้มค่า งานวิจัยนี้เน้นย้ำข้อได้เปรียบของ Webots เหนือแขนกลราคาประหยัด ซึ่งช่วยให้นักศึกษาสามารถทดลองกับระบบหุ่นยนต์ที่หลากหลายและงานที่ซับซ้อนได้มากขึ้น บทความให้รายละเอียดเกี่ยวกับการใช้งานการควบคุมหุ่นยนต์ใน Webots โดยใช้ Python พร้อมนำเสนอกรอบการทำงานพื้นฐานและตัวอย่างโค้ด มีการนำเสนอการทดลองที่เกี่ยวข้องกับนักศึกษาสายอุตสาหกรรม ซึ่งแสดงให้เห็นถึงประสิทธิภาพของ Webots ในการเพิ่มความเข้าใจของนักศึกษาเกี่ยวกับแนวคิดพื้นฐานด้านหุ่นยนต์ ผลลัพธ์จากการทดสอบทางสถิติ t-statistic ชี้ให้เห็นถึงการพัฒนาอย่างมีนัยสำคัญในความเข้าใจของนักศึกษาเกี่ยวกับส่วนประกอบของหุ่นยนต์และหลักการทำงาน หลังจากเข้าร่วมกิจกรรมการเรียนรู้ที่ใช้ Webots ทางผู้วิจัยจึงการสนับสนุนให้ใช้ Webots เป็นเครื่องมือที่มีคุณค่าในการเชื่อมช่องว่างระหว่างความรู้ทางทฤษฎีและทักษะปฏิบัติในการศึกษาด้านหุ่นยนต์ ซึ่งในที่สุดจะช่วยเตรียมความพร้อมให้นักศึกษาสำหรับการศึกษาด้านหุ่นยนต์

คำสำคัญ: Webots, การจำลองหุ่นยนต์, แขนกล, Python

Abstract

This article addresses the challenge of providing comprehensive robotics education at the university level, particularly considering the high costs associated with industrial-grade robot arms and maintenance. It proposes Webots, a powerful and accessible robot simulation platform, as a viable solution. The article compares Webots with other simulators like Gazebo, Coppeliasim, Open Roberta, and

MATLAB Robotics Toolbox, highlighting its strengths in terms of user-friendliness, comprehensive features, and cost-effectiveness. It emphasizes the advantages of Webots over budget robot arms, enabling students to experiment with a wider range of robotic systems and complex tasks. The article details the implementation of robot control in Webots using Python, providing a basic framework and code snippets. An experiment involving industrial education students is presented, demonstrating the effectiveness of Webots in enhancing students' understanding of fundamental robotics concepts. The results of t-statistic highlight significant improvements in students' comprehension of robot components and working principles after engaging with Webots-based learning activities. The article concludes by advocating for Webots as a valuable tool for bridging the gap between theoretical knowledge and practical skills in robotics education, ultimately preparing students for the evolving robotics landscape.

Keywords: Webot, Robot Simulation, Robot Arm, Python

1. Introduction

Robotics is rapidly evolving, demanding a skilled workforce equipped to handle complex challenges [1]. Educational institutions face increasing pressure to provide students with practical robotics experience [2]. However, the high cost of industrial-grade robot arms and maintenance often presents a significant barrier [3]. This article advocates for Webots [4], a powerful yet accessible robot simulation platform, as a solution to bridge this gap and enable comprehensive robotics education at the university level.

The robot simulators are nothing new. There are several robot simulation software. The comparison of software was done in terms of the accuracy of motion in mobile robots [5] which can be able the guideline including Gazebo [6], Coppeliasim [7], MORSE [8], Open Roberta [9], and MATLAB Robotics Toolbox [10], each possesses strengths and limitations. Gazebo and MORSE, known for their realism and complex physics engine, can be resource-intensive and challenging for beginners [11]. Coppeliasim, with its integrated development environment, may be more suitable for advanced users [12]. Open Roberta, primarily targeting educational robotics, simplifies programming but lacks the depth required for advanced applications [13]. MATLAB Robotics Toolbox excels in control systems design but often necessitates a strong foundation in MATLAB and Simulink [14]. The comparison of robot simulation software is summarized in Table 1.

Table 1: Comparative overview of robotics simulation platforms.

Feature	Gazebo	CoppeliaSim	MORSE	Open Roberta	MATLAB Robotics Toolbox	Webots
Cost	Free	Free & Paid versions	Free	Free	Paid	Free for education
User Interface	Steeper learning curve	Moderate learning curve	Steeper learning curve	Very user-friendly	MATLAB-based interface	User-friendly
Programming	C++, Python	Lua, C++	Python	Visual programming language	MATLAB	C++, Python, Java, etc.
Robot Library	Broad range	Moderate	Broad range	Limited, mostly educational robots	Customizable but limited library	Extensive, commercially available robots
Sensor Simulation	Detailed sensor models	Realistic sensor models	Detailed sensor models	Simplified sensor models	Customizable sensor models	Realistic sensor models
Ease of Use for Beginners	Low	Moderate	Low	Very High	Moderate	High
Advanced Features	Supports ROS, complex physics engine	Physics engine, API for extensions	Supports many middlewares	Limited advanced features	Strong for control systems design	Supports ROS, advanced control algorithms
Target Users	Researchers, Developers	Researchers, Educators, Professionals	Researchers, Educators	Students, Beginners	Researchers, Educators	Researchers, Educators, Professionals, Researchers,
Strengths	Large community, Wide range of supported robots, Cloud simulation	Easy to use, Versatile robot modeling, Fast simulation	Modular design, Focus on AI and machine learning	Beginner-friendly, Educational focus	Integration with MATLAB, Wide range of algorithms and tools	Wide range of supported sensors and actuators, Realistic rendering, Integration with MATLAB
Weaknesses	Less user-friendly, Less realistic rendering	Limited physics simulation capabilities	Smaller community, Less user-friendly	Limited to educational robots, Less flexibility	Limited to MATLAB users, Less focus on robot simulation	Steep learning curve, Resource intensive

Webots strikes a balance, offering a user-friendly interface coupled with comprehensive features. It boasts a vast library of commercially available robot models, sensors, actuators, and environments, providing a realistic simulation experience. Its support for multiple programming languages like C++, Python, and Java caters to diverse learning environments. Moreover, its open-source nature and availability of a free educational license make it a cost-effective solution compared to commercial alternatives and even budget robot arms like Dobot.

On the other side, compared to using budget robot arms such as Dobot Magician which is usually used in education [15], Webots enables students to experiment with a wider range of robotic systems, sensors, and environments without the limitations of physical hardware. Budget arms, while valuable for introductory purposes, often lack the robustness, precision, and advanced sensing capabilities of their industrial counterparts. Figure 1 illustrates this disparity, showcasing a typical budget arm alongside a sophisticated industrial robot. This discrepancy limits the complexity of tasks students can program, hindering the exploration of advanced robotics concepts like industrial automation, computer vision, and collaborative robotics. In addition, the programming paradigm of the budget robot is different from the industrial robot [16]. The students have to learn to program the industrial robot again as before.

Industrial robot arms are built for precision, speed, and endurance in demanding industrial environments [17]. They boast advanced control systems, powerful actuators, and a wide range of specialized end-effectors. This allows for complex tasks like high-speed assembly, intricate welding, and heavy material handling. In contrast, budget robot arms often prioritize affordability over high performance [15]. They typically have limited payload capacity, reduced accuracy, and simpler control systems that may not be suitable for intricate tasks or high-speed operations.

Figure 1. a budget robotic arm eg., DOBOT magician (left) and an industrial robotic arm (right) e.g., KUKA.

This article advocates for Webots, a powerful yet accessible robot simulation platform, as a solution to bridge this gap and enable comprehensive robotics education at the university level. While budget-friendly robot arms have their place in introductory exercises,

Webots provides a cost-effective way to expose students to the complexities of industrial-grade robotics.

While several simulators exist, including Gazebo, Coppeliasim, Open Roberta, and MATLAB Robotics Toolbox, each possesses strengths and limitations. From the comparison of simulation software as in Table 1 This article will delve into the specific advantages of using Webots for university-level robotics education, demonstrating its effectiveness as a tool to equip students with practical skills and prepare them for the demands of the evolving robotics landscape.

2. Implementing Robot Control in Webots using Python

Webots provides a straightforward Python API that allows users to control simulated robots directly from Python scripts. This section outlines the fundamental steps and code snippets to illustrate how to control an industrial robot arm within the Webots environment using Python.

2.1 Setting up the Environment:

2.1.1 Install Webots: Download and install the appropriate version of Webots for your operating system from the official website (https://cyberbotics.com/).

2.1.2 Create a World File: Open Webots and create a new world file. Import a suitable industrial robot model from the extensive Webots library or import your custom model.

2.1.3 Add Sensors and Objects (optional): Depending on your simulation goals, add relevant sensors (e.g., distance sensors, cameras) to the robot and objects to interact with within the simulation environment.

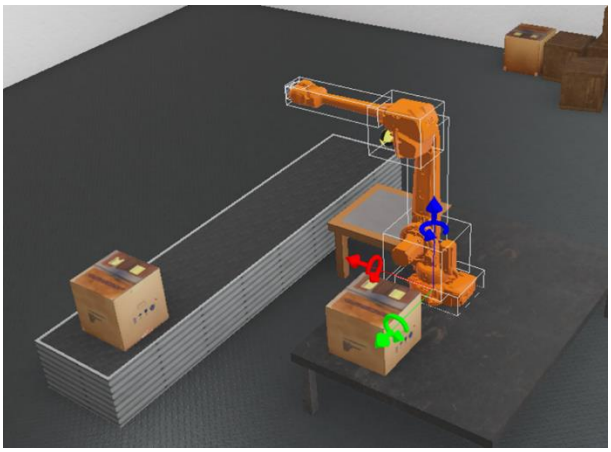

Figure 2. The Webots environment which contains a robot and a conveyor.

2.2 Python Script Structure:

A sample Python code for robot control in Webots follows this structure as in Figure 2

2.3 Example code explanation:

2.3.1 Import Necessary Classes: Begin by importing relevant classes from the controller module, such as Robot, Motor, PositionSensor, etc.

2.3.2 Robot Initialization: Create a class that inherits from the Robot class. In the constructor (__init__),

Table 2: Example robot control code

```
from controller import Robot, Motor, PositionSensor

# Import necessary classes

# Time step for the simulation (milliseconds)
TIME_STEP = 32

# Robot initialization
class MyRobot(Robot):
    def __init__(self):
        super(MyRobot, self).__init__()
        # Get robot nodes (motors, sensors)
        # Replace with actual set motor and sensor name
        self.motor1 = self.getDevice("motor1_name")
        self.motor2 = self.getDevice("motor2_name")
        # ... other motors and sensors

        # Optional: Enable position sensors if required
        self.sensor1 = self.getDevice("sensor1_name")
        self.sensor1.enable(TIME_STEP)
        # ... other sensors

    def run(self):
        while self.step(TIME_STEP) != -1:
            # Control logic here
            # Read sensor data (if applicable)
            sensor_value = self.sensor1.getValue()

            # Implement control algorithms, motion planning, etc.
            self.motor1.setPosition(target_position1)
            self.motor2.setVelocity(target_velocity2)
            # ... control other motors

# Create an instance of the robot
robot = MyRobot()

# Run the main control loop
robot.run()
```

obtain references to the robot's motors and sensors using their corresponding names defined in the Webots model.

2.3.3 Enable Sensors: If using sensors, enable them and specify the TIME_STEP for data acquisition.

2.3.4 Control Loop: The run() method contains the main control loop. The step(TIME_STEP) function needs to be called regularly to advance the simulation. Inside the loop:

Read sensor data (if applicable) using methods like getValue().

Implement control algorithms, motion planning, or any desired robot behavior.

Set motor positions, velocities, or torques using methods like setPosition(), setVelocity(), setTorque(), etc.

2.3.5 Robot Instance: Create an instance of your robot class.

2.3.6 Run Simulation: Execute the run() method to start the simulation and robot control loop.

3. Experiment

To assess the effectiveness of Webots in enhancing students' understanding of fundamental robotics concepts, an experiment was conducted with 20 industrial education students. The experiment focused on two key

areas: Components of a Robot and Working Principles of Robots.

3.1 Participants:

Twenty students enrolled in an introductory robotics course at the university level participated in the study.

The participants had limited prior experience with robotics.

3.2 Methodology:

The experiment followed a pre-test, intervention, and post-test design:

Pre-test: Before using Webots, students were given a pre-test to evaluate their baseline understanding of robot components (e.g., actuators, sensors, end-effectors) and working principles (e.g., degrees of freedom, coordinate systems, basic kinematics).

Intervention: Students engaged in a series of structured Webots-based learning activities designed to reinforce their understanding of the target concepts.

Components: Students worked with different robot models in Webots, exploring various sensors (distance sensors, cameras) and actuators (rotational motors, linear actuators). They were tasked with modifying robot designs by adding, removing, or substituting components to observe their effects on functionality.

Working Principles: Exercises were designed to illustrate concepts like degrees of freedom by manipulating robot joints in Webots. Students programmed basic robot motions to reach specific coordinates, reinforcing their understanding of coordinate systems and kinematics. Visualizations in Webots, such as displaying coordinate frames, aided in comprehending these abstract concepts.

Post-test: After completing the Webots-based learning modules, students took a post-test identical in format to the pre-test to measure their learning gains.

3.3 Data Collection and Analysis:

Scores from the pre-test and post-test were recorded for each student.

Paired sample t-tests were used to compare the mean pre-test and post-test scores and determine statistically significant improvements in understanding.

3.4 Expected Results and Discussion:

We anticipate that the post-test scores will be significantly higher than the pre-test scores, indicating that the Webots-based learning activities effectively enhanced students' comprehension of robot components and working principles. The interactive and visual nature of Webots is expected to facilitate a deeper understanding of these concepts compared to traditional textbook-based learning. Further qualitative data, such as student feedback and observations during the Webots-

based learning activities, will be collected to provide richer insights into the perceived educational value of the simulation platform. The results of this experiment will contribute to the growing body of evidence supporting the use of Webots as an effective tool for robotics education.

4. Result

After the experiment, Figure 3 is the result of the pre-test and post-test of understanding robot components tests by learning with Webots based. The graph shows the results of pre-test and post-tests for 20 students on a robot component test, likely after learning with the Webots software. The results can be summarized as follow:

Significant Improvement: Almost everyone scored higher on the post-test than on the pre-test, indicating that learning with Webots is effective.

High Achievers: Many students achieved a perfect score of 10 points on the post-test, demonstrating a strong understanding of the content.

Few Low Scores: Only a small number of students scored below 5 points on the post-test.

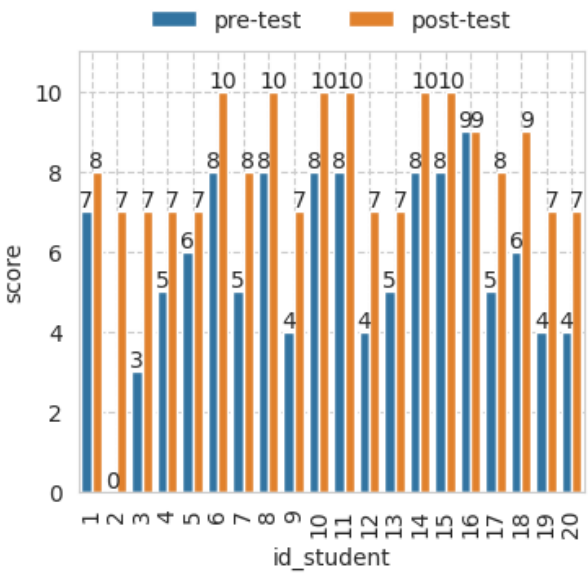

Figure 3. The understanding robot components pre-test and post-test scores.

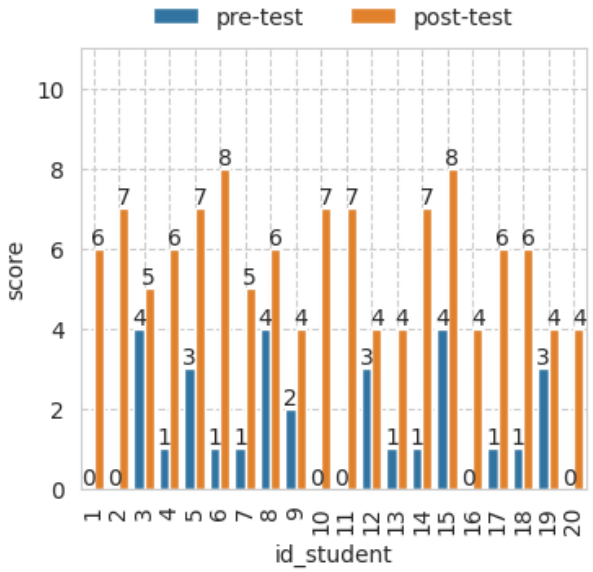

Figure 4. The understanding working principles pre-test and post-test scores.

Figure 4 is the result of pre-test and post-test of understanding robot working principles tests by learning with Webots based. The graph illustrates the results of pre- and post-tests on robot working principles, likely after a learning

period with the Webots simulator. The result can be summarized as follow:

Substantial Improvement: Almost everyone scored significantly higher on the post-test compared to the pre-test, indicating that learning with Webots is highly effective.

High Performance: Many students achieved high scores (6 points or above) on the post-test, demonstrating a good understanding of the principles.

Few Low Scores: Only a small number of students scored below 4 points on the post-test.

Both sets of results show extremely low p-values, indicating that the observed improvements are highly unlikely to be due to chance. The large negative t-statistics as in Table 3 further emphasize the substantial differences between pre-test and post-test performance, confirming the positive impact of Webots learning on the robot.

Table 3: T-statistic and p-value of the tests

Test	t-statistic	p-value
Robot Component Tests	-8.0118	0.000
Robot Working Principles	-9.163	0.000

Overall, these results strongly suggest that Webots learning significantly enhances both the robot's technical capabilities (component performance) and its theoretical understanding (working principles). This provides compelling evidence for the effectiveness of Webots as a learning tool for robots.

5. Conclusion

The experiment's results underscore Webots' potential as a valuable tool for robotics education. The significant improvement in students' understanding of robot components and working principles after engaging with Webots-based learning activities highlights the platform's effectiveness in conveying complex concepts. The interactive nature of Webots, coupled with its ability to visualize abstract concepts, appears to foster a deeper understanding compared to traditional teaching methods.

However, it's important to acknowledge the limitations of this study. The sample size was relatively small, and the participants were from a specific academic background. Further research with larger and more diverse groups of students would strengthen the generalizability of these findings. Additionally, the experiment focused on foundational robotics concepts. Investigating Webots' impact on more advanced topics, such as control systems design or robot programming, would provide a more comprehensive assessment of its educational value.

Despite these limitations, the results are promising and suggest that Webots can be a

valuable asset in bridging the gap between theoretical knowledge and practical skills in robotics education. By providing students with a risk-free environment to experiment, explore, and learn, Webots empowers them to develop the competencies needed to thrive in the rapidly evolving field of robotics.

Future research could explore the long-term impact of Webots-based learning on students' career trajectories and success in robotics-related fields. Additionally, investigating the optimal integration of Webots into existing curricula and developing best practices for its use would further enhance its educational impact.

In conclusion, this study provides compelling evidence for the effectiveness of Webots as a tool for enhancing students' understanding of fundamental robotics concepts. While further research is needed, the results suggest that Webots has the potential to revolutionize robotics education by making it more accessible, engaging, and effective. As the demand for skilled robotics professionals continues to grow, platforms like Webots will play an increasingly crucial role in preparing the next generation of innovators and problem solvers.

References

[1] J. Arents and M. Greitans, "Smart Industrial Robot Control Trends, Challenges and Opportunities within Manufacturing," Applied Sciences, vol. 12, no. 2, p. 937, Jan. 2022, doi: https://doi.org/10.3390/app12020937.

[2] C. Hyunjin and K. Tongjin, "A Study on the Development of Robot Education in the Fourth Industrial Revolution," Journal of Physics: Conference Series, vol. 1642, p. 012026, Sep. 2020, doi: https:// doi.org/10.1088/1742-6596/1642/1/012026.

[3] S. Chookaew, S. Howimanporn, S. Hutamarn, and T. Thongkerd, "Perceptions of Vocational Education and Training Teachers with regard to an Industrial Robot Training," TEM Journal, pp. 1149–1154, Aug. 2021, doi: https://doi.org/10.18421/tem103-19.

[4] "Webots: robot simulator," Cyberbotics.com, 2019. https://cyberbotics.com/

[5] A. Farley, J. Wang, and J. A. Marshall, "How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion," Simulation Modelling Practice and Theory, vol. 120, pp. 102629–102629, Nov. 2022, doi: https://doi.org/10.1016/j.simpat.2022.102629.

[6] "Gazebo," gazebosim.org. https:// gazebosim.org/home

[7] “Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics,” coppeliarobotics.com. https://coppeliarobotics.com/

[8] “Modular Open Robots Simulation Engine by morse-simulator,” morse-simulator.github.io. https://morse-simulator.github.io/

[9] “Open Roberta | Ein Projekt der Roberta-Initiative des Fraunhofer IAIS.” https://www.open-roberta.org/

[10] P. I. Corke, “A robotics toolbox for MATLAB,” IEEE Robotics & Automation Magazine, vol. 3, no. 1, pp. 24–32, Mar. 1996, doi: https://doi.org/10.1109/100.486658.

[11] Farzan Majeed Noori, D. Portugal, R. P. Rocha, and M. S. Couceiro, “On 3D simulators for multi-robot systems in ROS: MORSE or Gazebo?,” IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR) , Oct. 2017, doi: https://doi.org/10.1109/ssrr.2017.8088134.

[12] Y. Zhang, B. Feng, G. Zhang, and X. Bai, “The Application of CoppeliaSim EDU in Robot Course Teaching,” Journal of education and educational research, vol. 7, no. 1, pp. 167–170, Jan. 2024, doi: https://doi.org/10.54097/as82qn67.

[13] B. Jost, M. Ketterl, R. Budde, and T. Leimbach, “Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One?,” IEEE International Symposium on Multimedia, Dec. 2014, doi: https://doi. org/10.1109/ism.2014.24.

[14] M. G. Krishnan, A. T. Vijayan, and S. Ashok, “Interfacing an industrial robot and MATLAB for predictive visual servoing,” Industrial Robot-an International Journal, vol. 48, no. 1, pp. 110–120, Jul. 2020, doi: https://doi.org/10.1108/ir-05-2020-0100.

[15] T. Brito, J. Lima, J. Braun, L. Piardi, and P. Costa, “A DOBOT Manipulator Simulation Environment for Teaching Aim with Forward and Inverse Kinematics,” Lecture Notes in Electrical Engineering, pp. 303–312, Sep. 2020, doi: https://doi.org/10.1007/978-3-030-58653-9_29.

[16] Å. Fast-Berglund, O. Salunkhe, and M. Åkerman, “Low-cost Automation – changing the traditional view on automation strategies using collaborative applications,” IFAC-PapersOnLine, vol. 53, no. 2, pp. 10285–10290, 2020, doi: https://doi. org/10.1016/j.ifacol.2020.12.2762.