Mediterranean Journal of Basic and Applied Sciences (MJBAS)
Volume 8, Issue 3, Pages 70-81, July-September 2024

Educational Robotics: The Implementation of Karel in 3D

Ioannis Giachos[1], Evangelos-Diomidis Sagias[1], Christos Papakitsos[2], Evangelos Papakitsos[1*] & Nikolaos Laskaris[1]

[1]*Department of Industrial Design & Production Engineering, University of West Attica, Egaleo, Athens, Greece.* [2]*Department of Mechanical Engineering, University of West Attica, Egaleo, Athens, Greece. Corresponding Author (Evangelos Papakitsos) Email: papakitsev@uniwa.gr*[*]

DOI: https://doi.org/10.46382/MJBAS.2024.8307

Article Received: 21 May 2024 *Article Accepted: 28 July 2024* *Article Published: 30 July 2024*

ABSTRACT

This paper explores the use of educational robots in teaching programming, with a focus on the virtual robot Karel. Developed in 1981, Karel is one of the earliest educational tools designed to teach programming and algorithmic thinking. Through simple commands and a grid environment, Karel provides students with an opportunity to grasp the fundamental principles of programming in a visual and interactive manner. The paper outlines the evolution of Karel from its initial form to modern 3D implementations, as well as the integration of physical robots that allow students to see their commands executed in real-time. This combination of theoretical and practical education bridges the gap between learning and application, enhancing students' understanding and problem-solving abilities. The robot's functional specifications include basic movement commands, turns, gripper control, lifting, and obstacle detection and handling. These functionalities enable the robot to interact with its environment accurately and efficiently, offering students a rich field for experimentation and learning. The application design encompasses the development of movement control units, servomechanisms, ultrasonic sensors, and communication modules, ensuring the smooth operation and interactivity of the robot.

Keywords: Educational robotics; Virtual robot; Karel robot; Robotic applications; Teaching programming; 3D implementation; Fundamental programming; Object-oriented programming; STEM; Experimentation learning.

1. Introduction

The idea of using educational robots as tools for teaching computer programming and algorithmic thinking has a rich history, dating back to the early 1980s. One of the pioneering projects in this area was the development of the Karel robot, created by Richard E. Pattis in 1981 [1]. The original Karel was a simple, grid-based robot that could be programmed to move, follow commands and manipulate objects. This simple approach provided an accessible entry point for novices to understand the basics of programming, using a tangible and visual method that demystified abstract concepts. Richard E. Pattis, during his tenure at Stanford University, developed Karel as a teaching tool for the Introduction to Programming course. The robot's name was derived from Karel Čapek, the Czech author who coined the word "robot" [2]. Karel's robot was designed to operate within a grid environment, where it could execute simple commands such as "move", "turnLeft", "putBeeper" and "pickBeeper". Through these commands, students could understand the basics of programming logic and algorithms without having to tackle the complexity of real programming. Over the years, Karel's idea has been revisited and redesigned many times, with each iteration seeking to leverage the latest technological advances to enhance the educational experience. The evolution from 2D to 3D environments is a significant leap forward in this regard. Karel's 3D application introduces a new dimension of interactivity and engagement, making the learning process more immersive and intuitive for students [3]. Today, students are able to program Karel in more sophisticated environments using programming languages such as Python and Java, which further enhances their understanding of more advanced programming concepts [3].

The virtual educational robot Karel, implemented as a free software program, operates on a personal computer (PC) screen in a microcosm environment. The primary goal of this project is to train primary and secondary school students in computer programming and algorithmic thinking through an attractive and interactive platform [1].

Karel serves as an introductory tool to help students understand basic programming concepts such as control structures (e.g., loops and conditions), variables, and procedures [2]. Through this process, students gain a foundation that will help them understand more complex programming projects in the future [3]. Because of this, Karel has been introduced in more educational systems, like the Greek one [4], especially in secondary education [5], starting from ICT teachers' training in using Karel [6]. Regarding pupils, all major aspects of teaching computer programming can be introduced, namely, procedural programming [7], structured programming [8] and object-oriented programming [9].

The introduction of 3D graphics in the Karel virtual robot offers several advantages over traditional 2D implementations. It allows for a more dynamic and visually appealing environment, which can help to maintain student interest and motivation [3]. The 3D environment also provides a more realistic simulation of real-world scenarios, making it easier for students to understand complex concepts [3]. Through this approach, students can gain a more comprehensive understanding of programming principles and develop skills that will be useful in both their academic and professional lives [3].

The 4WD Mechanical Robot Arm Smart Car serves as the physical implementation of the virtual robot Karel. This hardware component is flexible and programmable, making it an ideal tool to bring virtual courses to life [10]. By integrating software and hardware components, the project bridges the gap between theoretical learning and practical application. Students can see the direct impact of their code on the physical robot, which helps solidify their understanding of programming principles and their real-world applications [10]. This hands-on approach encourages students to experiment and develop problem-solving skills, promoting creativity and innovation [10]. Overall, Karel's story and its evolution reflect the continuous effort to improve the educational process through technology. From its original form as a simple grid robot to modern 3D implementations and the integration of physical robots, Karel continues to be an important tool for teaching programming, offering students the opportunity to learn through action and interaction [11].

1.1. Study Objectives

Considering the above abilities and potential of Karel, the objectives of the herein study were:

(i) To enquire the feasibility of a 3D implementation of virtual Karel to a physical robot; (ii) To study the degree of transforming the performance of virtual Karel in three dimensions; (iii) To deal with robotic problems such as movement in space, obstacle detection and avoidance; (iv) To bridge the educational gap between theoretical learning and practical applications; and (v) To facilitate experiential learning and develop problem-solving skills, creativity and innovation.

2. Design of the application

The design of the 3D implementation of the Karel-type virtual robot with diverse capabilities includes several key modules. Each module serves a distinct purpose and manages specific data and processes to ensure the efficient operation of the robot. The main units include the motion control unit, the servo control unit, the ultrasonic sensor unit, and the communication unit.

2.1. Motion control unit

The motion control unit is responsible for the directed movement of the robotic car, including forward and backward movements, as well as turning to the left or right. This unit ensures that the robotic car can navigate its environment accurately and efficiently.

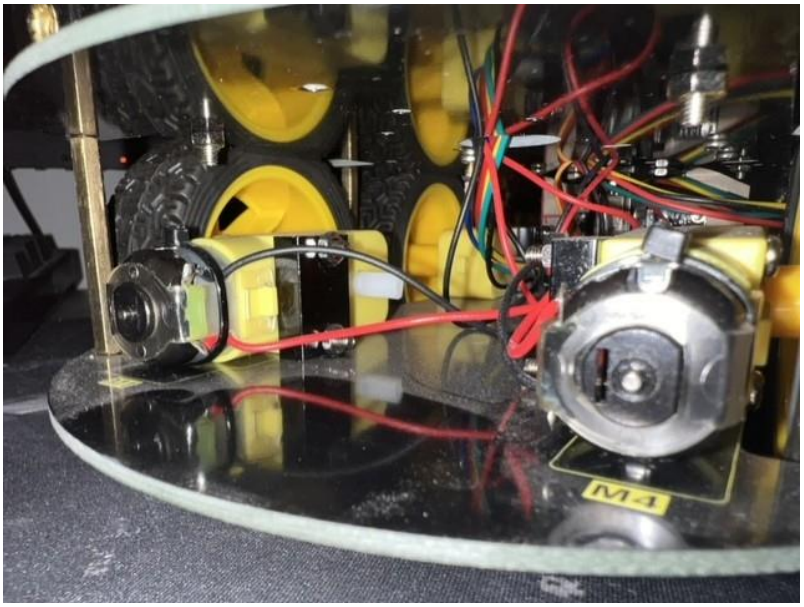

Figure 1. The motors in construction [11]

The managed data include:

- Speed settings for each motor.
- The duration for which each movement is performed.

The included procedures are as follows:

- `advance(int seconds)`: This procedure moves the robot forward for a specified duration; sets the PWM (pulse width modulation) values for the motors to move the robot.
- `back(int seconds)`: This procedure moves the robot backwards for a specified duration; it adjusts the PWM values to reverse the motors.
- `turnL(int seconds)`: This procedure turns the robot to the left for a specified duration; controls the motors on each side differently to achieve the turn.
- `turnR(int seconds)`: This procedure turns the robot to the right for a specified duration; similar to the left turn, it controls the motors to achieve the right turn.
- `stopp()`: This procedure stops all motion by resetting the speeds of the motors to zero, ensuring that the robot stops its motion safely.

The drive control unit relies heavily on the Adafruit_PWMServoDriver library to control the motors via PWM signals, which provide the necessary control of motor speeds and directions.

2.2. Servo control unit

The servo control unit manages the position and movement of the robot's servo motors, including the caliper, the rotary mechanism and the lifting mechanism. This module allows precise control of the robot's mechanical components, enabling it to interact with its environment through gripping, rotation and lifting actions.

The managed data include angles for each servo motor (caliper, rotary, lift).

The included procedures are as follows:

- `myservo1.write(angle)`: Sets the angle for the claw servo motor, allowing it to open or close as needed.
- `myservo2.write(angle)`: Sets the angle for the rotation servo motor, allowing rotary motion.
- `myservo3.write(angle)`: Sets the angle for the lift servo motor, controlling vertical movement.

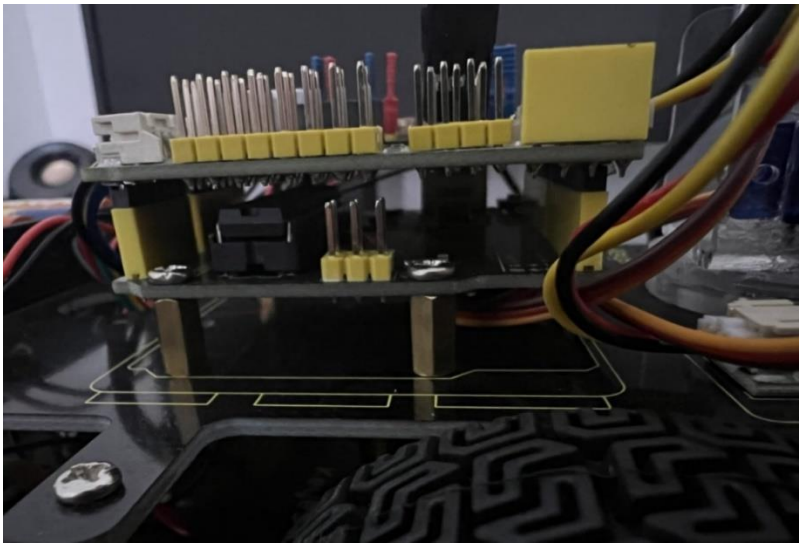

Figure 2. The Arduino UNO-type board on the base under the shield in the construction [11]

2.3. Ultrasonic sensor unit

The ultrasonic sensor unit is designed to measure the distances from obstacles in front of the robot using an ultrasonic sensor. This unit is vital for obstacle avoidance, ensuring that the robot can navigate its environment safely and interact effectively with objects.

The managed data include distance measurements from the ultrasonic sensor.

The included procedures are:

- `Ultrasonic_Ranging()`: This procedure measures and returns the distance to the nearest obstacle. It sends a pulse and measures the time it takes for the echo to return, calculating the distance based on this duration.
- `grab_obstacle()`: this complex procedure uses ultrasonic sensor data to control the robot's movement based on proximity to obstacles. It adjusts the robot's position to ensure safe and efficient object grabbing. If the obstacle is too close, the robot moves backwards; if it is at an optimal distance, the robot stops; if it is further away, the robot moves forward until the object is within grabbing range.

The ultrasonic sensing module utilizes the 'pulseIn' function to measure the return time of the ultrasonic pulse, converting this time into distance measurements.

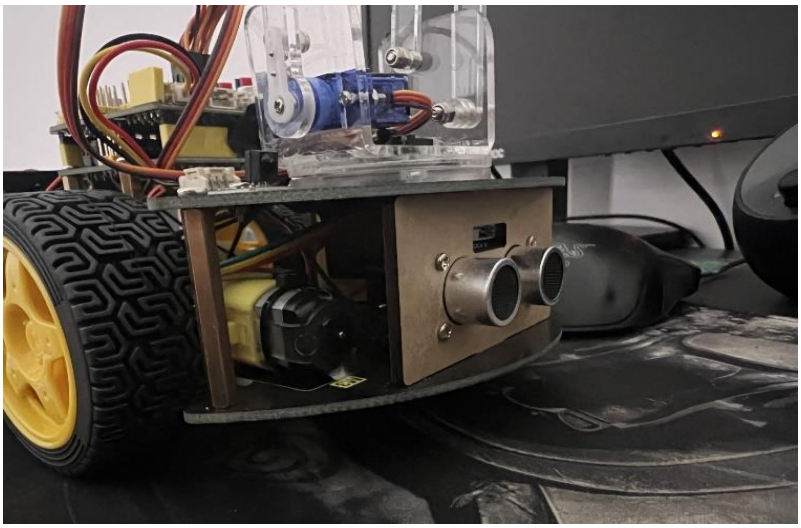

Figure 3. The ultrasonic sensor in construction [11]

3. Implementation - Construction

The following components were used for the construction of the robot:

- The 4.5 V 200 RPM (revolutions per minute) motor for Arduino [12] is a small and efficient direct current (DC) motor that offers the possibility of precise speed and direction control. Its 4.5 V operating voltage ensures low power consumption, making it ideal for applications with limited energy requirements. Its maximum rotational speed is 200 RPM, making it suitable for applications requiring moderate speed and motion stability, such as robotic vehicles and automatic control systems.

- The HC-SR04 ultrasonic sensor is a reliable and widely used distance sensor that operates using ultrasonic waves. This sensor consists of an ultrasonic transmitter and an ultrasonic receiver. The transmitter emits an ultrasonic pulse at a frequency of 40 kHz, which travels through the air until it encounters an object. When the pulse is reflected by the object, it is returned to the receiver, which measures the time taken for the signal to return.

The sensor has four terminals: VCC, Trig, Echo and GND. The VCC terminal is connected to the power supply voltage (usually 5V), while GND is connected to ground. The Trig terminal is used to send a trigger pulse from the microcontroller (such as the Arduino), while the Echo terminal returns the signal to the microcontroller, with the pulse duration corresponding to the distance measured.

The HC-SR04 can measure distances from 2 cm to 4 m with an accuracy of up to 3 mm. Its easy integration into platforms such as Arduino makes the sensor ideal for robotics, automation and obstacle detection applications. The HC-SR04 sensor offers an affordable and reliable solution for projects, requiring accurate distance measurements in various environments.

Keyestudio V4.0 [13] features a large number of input/output (I/O) pins, including 14 digital pins (6 of which can be used as PWM outputs) and 6 analog input pins, allowing a variety of sensors and components to be connected.

The board also supports serial communication via a USB port, facilitating the programming process and data transfer with the PC. An integrated linear voltage stabilizer ensures that a stable operating voltage of 5V is provided, even when the board is powered by an external voltage source.

Keyestudio V4.0 includes additional built-in features that enhance its functionality, such as LED indicators for terminal status and power supply, and a reset button to restart the program. The board can be programmed through the Arduino development environment (Arduino IDE), making it accessible for both novice and experienced users.

This development board is an ideal solution for developing a variety of electronic projects and experiments, from simple educational examples to more complex automated systems and robotics.

The Keyestudio TB6612 motor driver shield is a highly functional and easy-to-use component designed for controlling DC motors and stepper motors in applications with Arduino platforms and other microcontrollers. Based on the TB6612FNG integrated circuit (IC), this shield offers reliable and efficient motor control, providing dual H-bridge channels that allow simultaneous driving of two DC motors or one stepper motor.

The TB6612 shield supports operation over a wide range of voltages (from 4.5V to 13.5V), making it suitable for various applications with different power requirements. It can deliver up to 1.2A current per channel with a maximum peak current of 3.2A, ensuring adequate power supply for even the most demanding motors. The shield also has built-in protection against overheating and overcurrent, ensuring safe operation of the connected motors and the system as a whole.

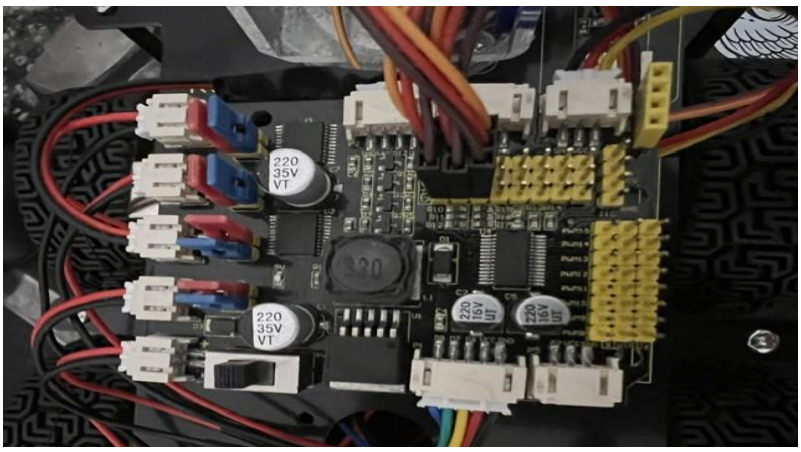

Figure 4. Keyestudio TB6612 motor driver shield in construction [11]

The connectivity of the shield to the Arduino board is simple, thanks to its shield configuration, which allows it to be placed directly on the board. It provides additional terminals for external powering of the motors, if more power is required than the Arduino board can provide. The TB6612's control inputs can be connected to digital terminals on the Arduino board, allowing easy programming of the motors' motion through the Arduino development environment (Arduino IDE).

The Keyestudio TB6612 motor driver shield is ideal for robotics, automation applications and any kind of project that requires accurate and reliable motor control. Its flexibility and ease of use make it ideal for both novice and experienced users, looking to develop complex and functional electronic systems.

The Keyestudio MG90S 14g is a small and powerful servo for use in Arduino projectors and other electronics applications. Here is a description of its functions:

The Keyestudio MG90S is a digitally controlled servo motor, designed for precise and reliable motion. Weighing only 14 grams, it is a lightweight and portable motor that provides power and precision in applications. The MG90S provides angular motion up to 180 degrees, making it suitable for a variety of applications where precise position control is required. The motor is compatible with platforms such as Arduino and Raspberry Pi, and can be easily integrated into robotic projects, RC cars, airplanes, etc. It is extremely durable and provides smooth motion without vibration, and its tolerance to high loading makes it suitable for applications that require reliable and continuous operation. Applications for the MG90S include moving mechanisms, moving robotic arms, moving heads and cameras in robotic systems, and other applications where precise and fast motion with small fluctuations is required.

4. Operational assessment

4.1. Description of the results of the operation of each component

4.1.1. Microcontroller

Function: The microcontroller is the "brain" of the robotic car. It is responsible for processing the commands and controlling the other components.

Inputs: Programmed commands via USB or Bluetooth (often Arduino is used as a base).

Outputs: Control signals to the motors and the robotic arm.

Typical values:

- Voltage: 5V (usually via USB or battery).
- Signal values: 0-5V digital signals for PWM (Pulse Width Modulation).

4.1.2. Drive motors (DC Motors)

Function: These motors are responsible for driving the 4 wheels of the robot.

Inputs: PWM signals from the microcontroller.

Outputs: Mechanical movement of the wheels.

Typical values:

- Voltage: 6-12V (depending on motor type).
- Current: 0.5-1A per motor.

4.1.3. Robotic arm

Function: The robotic arm allows the execution of movements and the grasping of objects.

Inputs: Control signals for the servo motors from the microcontroller.

Outputs: Movement of the joints and fingers of the arm.

Standard values:

- Voltage: 4.8-6V for servo motors.
- Current: 100-500mA per servo.

4.1.4. Sensors

Function: The sensors are used for obstacle detection, line tracking or light detection.

Inputs: Power from the microcontroller.

Outputs: Analog or digital signals back to the microcontroller.

Typical values:

- Voltage: 3.3-5V.
- Signal values: Analog (0-1023) or digital (0 or 1).

4.1.5. Batteries and Power Supply

Function: They provide the energy needed to operate the robotic car and arm.

Inputs: Charging from an external source (if rechargeable).

Outputs: Powering the motors, microcontroller and servos.

Typical values:

- Voltage: 7.4-12V (LiPo batteries are often used).
- Current: Depends on the requirements of the components.

4.1.6. Motor Drivers

Function: They manage the power supplied to the motors based on commands from the microcontroller.

Inputs: PWM signals from the microcontroller.

Outputs: current to the motors.

Typical values:

- Voltage: Compatible with the voltage of the motors (6-12V).
- Current: Maximum 2A per channel (depending on controller model).

4.2. Potential problems in construction and operation

During the process of manufacturing and operation of the robot, several problems were identified that affected the performance and reliability of the system.

First, during the assembly of the robotic arm, it was found that the screws did not fit properly on the servo motor of the base. This incompatibility between the screws and the servo motor sockets created problems in securely

fastening the components, resulting in instability of the arm during operation and a reduction in the accuracy and performance of the robot.

In addition, when using the Bluetooth wireless connection to send commands to the robot, it was observed that after repeated commands, the system became stuck and would not accept new commands. This problem is speculated to be due to memory management errors or an overload of the communication protocol, requiring further investigation and possible software upgrades to properly manage commands via Bluetooth. For this reason, it was not used for the construction.

During the testing of the servo motors used to drive the robotic arm, it was found that one of the issues that arose was insufficient servomotor strength when the screws securing the servomotors were over-tightened. This is probably caused by increased friction or deformation of the servo motor, combined with the possible lack of adequate power supply. To avoid this problem, it is necessary to properly adjust the tightening of the screws and to ensure that the servomotors are supplied with adequate power.

In addition, the lifting capacity of the robotic arm was limited, as it can only lift very light objects. This limiting feature is due to the limited power of the servo motors used in the arm. To enhance the lifting capacity, the use of more powerful servo motors or the optimisation of the structure of the robotic arm may be required.

Finally, inaccuracy was observed in the distance sensor when the robot is in motion, as it cannot accurately calculate the distance. This inaccuracy can be caused by a delay in data processing or by external interference. To improve the performance of the sensor, it may be necessary to review its configuration and positioning, and to incorporate data filtering algorithms to eliminate interference.

The above issues highlight the challenges faced in the design and operation of the cart and require careful analysis and optimization to improve the overall performance of the system.

4.3. Applications and research directions for the future

The robot offers many possibilities for improvements and extensions, both at the level of applications and in research directions. The following suggestions highlight some of the most interesting directions for future development.

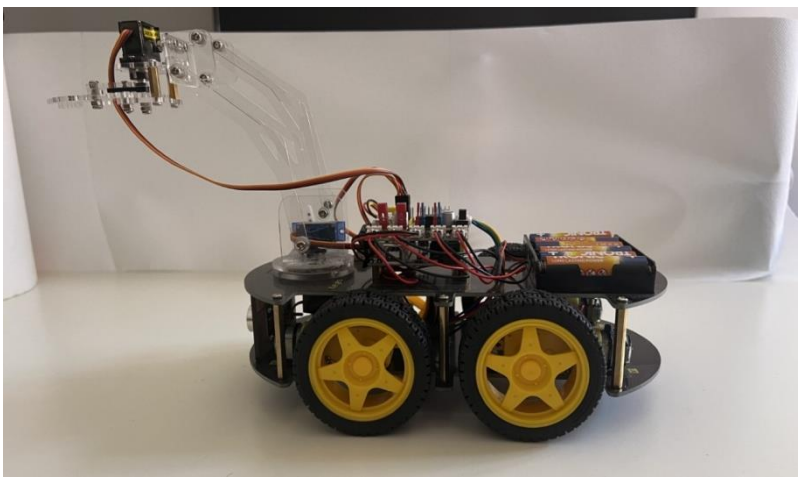

Figure 5. The robot in side-view [11]

The autonomy of the robot is critical for its practical use in real-world conditions. One of the ways to improve range is to integrate more efficient batteries or develop charging systems on the robot, such as using solar panels. In addition, optimizing the software to reduce energy consumption while performing daily tasks could significantly increase the robot's uptime. The durability of the robot's construction materials is crucial for its long-term use. An important research direction is the investigation and application of new, light but strong materials that can withstand mechanical stresses and environmental conditions. Materials such as composites, special polymers and carbon fibers could be used to improve the durability and lifespan of the robot. The open-source nature of the robot allows developers and researchers to contribute to the development of new features and improvements. The community can develop and share new algorithms, optimize existing programs, and add new features, such as advanced artificial intelligence and machine learning systems, that can improve the robot's autonomous operation [14].

The design of the robot allows for easy changing and modification of components, making upgrades and customizations easier. This capability provides the basis for a number of applications, such as testing new sensors, replacing motors with more powerful models, and adding additional components such as cameras and other sensors for more specialized functions [15].

Researchers can leverage the robot to develop and test new technologies and applications. Some potential research directions include developing advanced navigation algorithms for autonomous movement in complex environments and exploring ways to better interact between humans and robots through voice commands [16] and gesture recognition. Using the robot as an educational tool to teach the basics of robotics and programming in schools and universities is also an important direction. Continued research and development in these areas will enhance the functionality and efficiency of the robot, making it an even more useful and versatile tool for both educational and professional applications.

5. Conclusion

In summary, this work dealt with the development of an educational robot that aims to enhance the learning of programming and algorithmic thinking through an attractive and interactive platform. Both the theoretical foundations and the technical details of the implementation of a virtual and physical Karel-type robot were analyzed, highlighting the importance of the transition from 2D to 3D operation to improve the educational experience.

The evolution of Karel's 3D implementation allows pupils to more accurately grasp programming concepts through hands-on application and interactivity. The functional specifications and descriptions of the individual sub-modules provide a detailed picture of how pupils can operate the robot to understand the fundamentals of programming, such as motion control, servo management and obstacle detection.

This project bridges the gap between theory and practice, enabling pupils to see the effects of their programming commands directly in action. This approach not only enhances understanding of programming concepts but also increases pupils' interest and involvement in the learning process.

Finally, the development and application of this educational robot contribute to the advancement of educational technology and opens new avenues for the integration of similar tools in the educational process. This project is an approach that can be a model for future developments in education, combining technology with education in a way that facilitates the understanding and application of complex concepts in real-world scenarios.

Furthermore and besides educational applications, the present project can be extended in future, towards: (a) more efficient batteries and charging systems for increasing autonomy, (b) the usage of new materials for improved durability, and (c) artificial intelligence enhanced performance, including natural language understanding and communication, machine learning, decision making and object recognition.

Declarations

Source of Funding

This research does not benefit from grants from any non-profit, public or commercial funding agency.

Competing Interests Statement

The authors have declared that no competing financial, professional or personal interests exist.

Consent for publication

All authors contributed to the manuscript and consented to the publication of this research work.

Availability of data and material

Not applicable.

Authors' contributions

All the authors took part in literature review, analysis, and manuscript writing equally.

References

[1] Pattis, R.E. (1981). Karel the Robot: A Gentle Introduction to the Art of Programming. New York: Wiley.

[2] Pattis, R.E., Roberts, J., & Stehlik, M. (1995). Karel - The Robot, A Gentle Introduction to the Art of Programming (2nd Ed.). New York: Wiley.

[3] Ericson, B., Guzdial, M., Morrison, B., & Decker, A. (2013). Preparing secondary computer science teachers through an iterated attempt to provide real computing experiences. In Proceedings of the 44th ACM Technical Symposium on Computer Science Education, Pages 165–170.

[4] Xinogalos, S. (2011). Teaching Programming to Secondary Education Students with a Learning Environment Based on “Karel the Robot”: A Pilot Study in a Greek High School. In Thomas S. Clary (Eds.), “Horizons in Computer Science Research”, Pages 67–92, New York: Nova Science.

[5] Xinogalos, S. (2003). Scenarios for Teaching Programming in Secondary Education. In Proceedings of the 2nd Panhellenic Conference of Teachers on ICT “Utilization of Information and Communication Technologies in Teaching”, Pages 783–795, Syros, Greece.

[6] Xinogalos, S. (2009). Proposal for the Teaching of Programming in High School using the Robot Karel. In Proceedings of the 5th Panhellenic Conference of Teachers on ICT "Utilization of Information and Communication Technologies in Teaching", Pages 953–963, Syros, Greece.

[7] Xinogalos, S. (2010). The Teaching of the Procedure Concept Using the Karel Robot to High School Pupils: A Case Study. In Proceedings of the 5th Panhellenic Conference "Didactics of Informatics", Pages 105–114, Athens.

[8] Kotini, L. (2012). Structured Programming with the Karel Robot - Selection Structure. In Proceedings of the Workshop on Informatics "Informatics in the era of the New School", PAKE C, Macedonia.

[9] Xinogalos, S. (2002). "Educational Technology": A Teaching Microcosm for the Introduction to Object Oriented Programming. PhD Thesis, Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece.

[10] Cliburn, D.C. (2006). Experiences with the LEGO Mindstorms throughout the undergraduate computer science curriculum. In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, Pages 191–195.

[11] Sagias, E.D. (2024). 3D implementation of a virtual Karel type robot with alteration of its capabilities. Diploma Thesis, Department of Industrial Design and Production Engineering, School of Engineering, University of West Attica, Athens, Greece.

[12] https://www.arduino.cc/.

[13] https://wiki.keyestudio.com/ks0523_keyestudio_4wd_mechanical_robot_arm_smart_car.

[14] Giachos, I., Papakitsos, E.C., Savvidis, P., & Laskaris, N. (2023). Inquiring Natural Language Processing Capabilities on Robotic Systems through Virtual Assistants: A Systemic Approach. Journal of Computer Science Research, 5(2): 28–36. https://doi.org/10.30564/jcsr.v5i2.5537.

[15] Tsatsaris, A. (2024). Development of a Remote-Controlled Crawler Vehicle Control System. Master's Degree in Unmanned Autonomous and Remote Controlled Systems, Department of Industrial Design & Production Engineering, University of West Attica, Greece. https://polynoe.lib.uniwa.gr/xmlui/handle/11400/6807.

[16] Giachos, I., Papakitsos, E.C., & Chorozoglou, G. (2017). Exploring natural language understanding in robotic interfaces. International Journal of Advances in Intelligent Informatics, 3(1): 10–19. http://dx.doi.org/10.12928/ijain.v3i1.81.