ISBN: 978-628-95207-4-3. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2023.1.1.887

Development of a Multimodal Robot Controlled by Raspberry Pi Using Python-Based Graphical Programming Environment

Luis Segura, Student[1], Emilio Higuchi, Student[2], Leonardo Vinces, Magister[3] and José Oliden, Magister[4]
[1,2,3,4] Universidad Peruana de Ciencias Aplicadas, Perú, u201815453@upc.edu.pe, u201719027@upc.edu.pe, leonardo.vinces@upc.pe, pceljoli@upc.edu.pe

***Abstract*– In the fields of computer science and electronics, resources for artificial intelligence and robotics tend to be aimed at experts, making it challenging for students to enter these fields without specialist assistance. To address this issue, this project proposes a cost-effective and educational development environment for graphical programming utilizing node-based programming and a Raspberry Pi-controlled robot. The software architecture, developed in Python, incorporates graphical resources to enable signal processing, remote connection, and artificial intelligence libraries under the functional programming paradigm. The resulting programming environment is intuitive and user-friendly, offering artificial intelligence capabilities to a multimodal robotic system controlled by Raspberry Pi.**

***Keywords*— STEAM, reactive programming, GUI, educational robotics, artificial intelligence, multimodal communication, computer vision, teaching programming, Learning artificial intelligence, Programming language, Raspberry Pi, graphical programming, programming by nodes.**

I. Introduction

According to previous research, teaching robotics in STEM education has positive impacts on students by providing hands-on experience with technology, facilitating the understanding of abstract concepts, promoting logical thinking, and enabling practical methodologies for teachers [1][2][3][4]. However, the complexity of available tools and a lack of confidence among mentors regarding the management of new technologies hinder the efficient dissemination of knowledge in these fields [5][6][7].

To address this issue, a need arises for a system that integrates robotics and artificial intelligence algorithms in a simple, didactic, and affordable way. Raspberry Pi is a suitable ecosystem due to its compatibility with affordable components and programming complexity. Various educational robotics projects have used Raspberry Pi as the central controller.

Multiple educational robotics projects have utilized different versions of Raspberry Pi as the central controller. For instance, Bindu, Alam, and Neloy [8] proposed a multipurpose robotic arm with six degrees of freedom that transmits real-time camera images via an HTTP server. However, control is carried out via radio frequency, and the programming environment is unspecified.

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

Educational robotics proposals mostly use existing development environments, with Scratch being the best-known platform for its simplicity in block programming and integration with ROS. For robots that use Arduino controllers, the generally used environment is the Arduino IDE. Vega and Cañas [9] present an educational robot programmed through an API accessible from Python, or generated through Scratch, while Arvin et al. [10] proposes a remote-controlled robot programmed through Arduino IDE or graphical programming of mBlock. However, neither of these proposals includes support for running artificial intelligence algorithms or multimodal control.

After excluding projects concerning specific robot hardware development, Liang et al. [11] proposed a framework for robot action teaching, reuse, and refinement, where a graphical interface permits action details calibration to optimize robot performance, applying a mathematical model for process execution. Their solution requires a degree of automatic planning knowledge related to AI models. Pacheco and Macedo [12] propose a language to simplify complex instructions for ROS-based robot control using the reactive paradigm. However, their solution lacks a graphical programming environment and artificial intelligence support. Finally, Berenz and Schaal [13] present a decision tree-based scripting language for reactive robot control that allows runtime node modification but requires programming knowledge and lacks a specialized development environment.

To address these limitations, this research proposes a graphical development environment using reactive nodes to program multimodal robot control, with Raspberry Pi as the primary controller.

II. Methodology

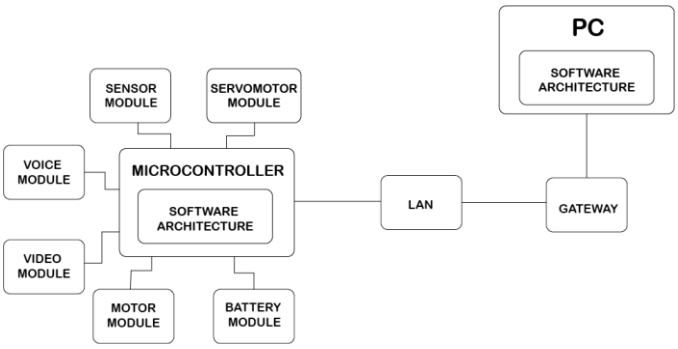

Fig. 1 Modular robot block diagram

A. *Programming paradigm*

To harness the full capabilities of the Raspberry Pi as a controller and enable remote instruction execution, software for both the server-side (robot) and client-side (computer with the programming environment) was developed using Python as the programming language and its external libraries.

Table I. Comparison of features between existing controllers on the market

Controllers	*Features*				
	Processor	*RAM*	*Frequency*	*I/O*	*Price*
Raspberry Pi 4B	ARM Cortex-A72 64-bit quad-core	2GB	1.5GHz	28	45$
Banana Pi BPI-M4	ARM Cortex-A53 Quad-Core 64 Bit	1GB	921MHz	40	38$
NVIDIA Jetson Nano	Quad-core ARM® A57 @ 1.43 GHz	2 GB	1.35GHz	56	250$
Beaglebone Black	AM3358BZCZ100 ARM Cortex-A8	512 MB	1GHz	46	$55

A graphical environment was created to locate nodes that represent complex functions and interconnect them to form a program for robot controller execution.

The node design comprises two or more attributes that may possess inputs, outputs, or static values as parameters of their representative functions. As depicted in Fig 2, the node attributes specify the type of data they handle, ensuring correct information entry and exit from the node.

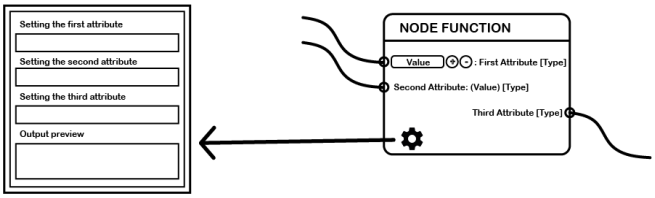

Fig. 2. Basic node model

Each node features a configuration option that reveals a window for specifying static attribute values, configuring input details, or presenting real-time changes to the processed information. By leveraging the node design as depicted in solutions proposed by [13] and [14], the graphical instructions are transformed into reactive paradigm functions that run on the Raspberry Pi. In this paradigm, the system receives an asynchronous data stream, to which it responds when a change occurs and executes a specific action.

B. *Architecture of the development environment*

The proposed solution comprises two interconnected applications in a local area network: one runs on the computer where the robot will be programmed graphically, while the other runs on the Raspberry Pi controller, which controls the actuators of the robot.

During the development of the programming environment, external libraries outlined in Table II were utilized, alongside Python language modules for executing calculation functions, processing data structures, and managing information on LAN networks.

Table II. External Python libraries used in the development of the software

Library	*Function*
Dear PyGUI	Control the execution, design and updating of the various components of the graphical interface of the software
Paramiko	Establish remote connection to controller via SSH protocol
OpenCV	Process the images received from the controller in real time

Visual Studio Code was used as the code editor for its versatile extensions and multipurpose functions. Development commenced by creating a virtual Python environment, as library version differences could result in compatibility errors with other available projects on the same computer. Fig 3 presents the proposed architecture, where the Main file initializes the construction of the graphical user interface (GUI) and the runtime functions that remain active during software deployment.

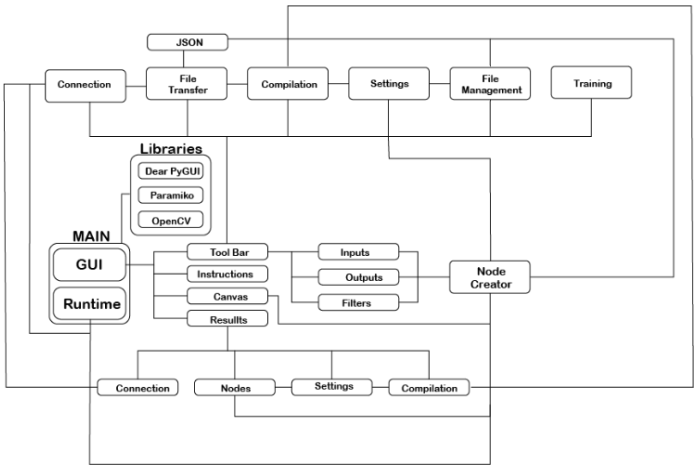

Fig. 3. Software architecture

The software's interface comprises four major fields that perform crucial tasks. The toolbar serves as the primary field, containing configuration options, file management features for data preservation, connection to the controller, and model compilation and training. The instructions window classifies each function that creates a different node into three categories: input for data input through peripherals or sensors, output for actuator control, filtering, and pre-processing of information, which enables decision-making. The canvas is where nodes are created, managed, and connected. Finally, the results window displays errors, warnings, or updates of every process carried out in the software.

It is important to note that the build process involves retrieving information from the nodes stored in the canvas and converting each node into a respective function call statement that accepts attribute information as parameters. The resultant

file is transmitted to the controller, where it is executed based on the functions stored in it.

Regarding the interface shown in Fig. 4, it can be observed that the interface is divided into four main fields that serve as the most significant functionalities of the software. The toolbar is responsible for various options, such as file management for data retention, configuring settings, connecting to the controller, and training models. The instructions window provides a categorization of each function into three categories, input for information input through sensors or peripherals, output for controlling actuators, filters, or pre-processing that classifies information and makes decisions about it. The canvas field enables the creation, management, and connection of nodes. Finally, the results window displays any errors, warnings, or updates related to the software's processes.

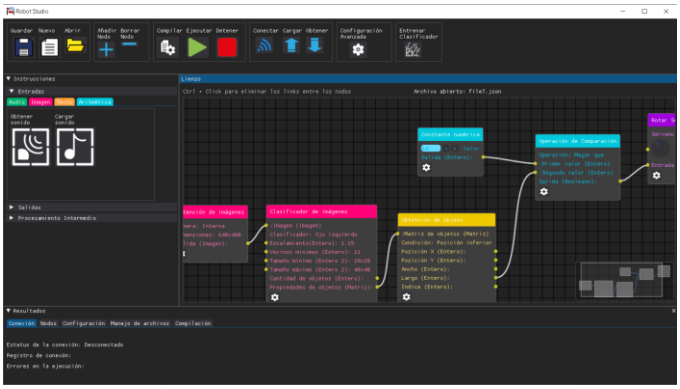

Fig. 4. Graphical user interface of the software that controls the robot

An essential aspect to note is that during the build process, the canvas nodes are utilized to convert each node into a respective function call statement that accepts the attribute information as parameters. The resulting file is then transmitted to the controller, which executes it based on the stored functions.

C. *Controller software architecture*

The Python file generated by the development environment is saved in a specific directory and sent to the controller for execution through operating system instructions. The file contains function calls that are part of the driver software architecture and are categorized similarly to nodes for easy access.

During the file execution process, the input and preprocessing functions' values are initialized, indicating the devices, models, protocols, and other characteristics used during execution. The proposed paradigm being reactive, the functions are executed in an infinite loop where incoming data is received from devices, and actions are performed with the actuators based on the information received.

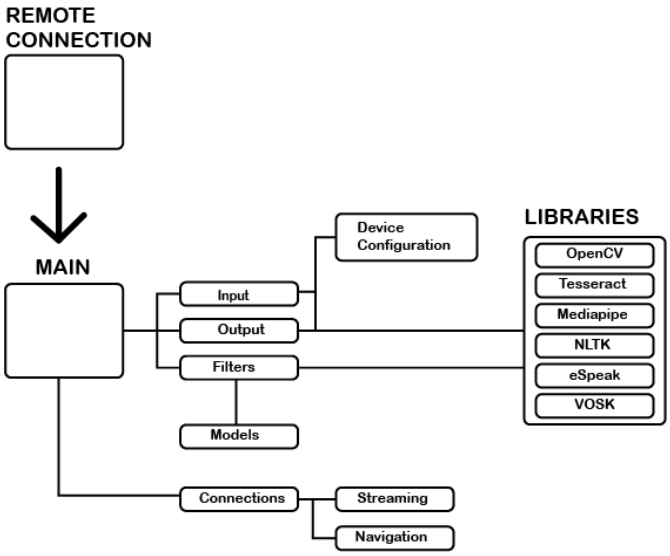

Fig. 5. Software architecture in the controller

Libraries listed in Table III, required for artificial intelligence models, computer vision algorithms, and processing, are installed on the controller, enabling any function that requires them to access them easily.

The software on the controller has remote connection instructions, acting as a server to transmit real-time information from input devices over a local network to monitor the successful execution of the desired program.

Table III. External python libraries used in the controller

Library	***Function***
OpenCV	Acquire and process images in real time and apply artificial intelligence models for pattern recognition
Tesseract	Process the obtained images for text recognition
Mediapipe	Process the obtained images for gesture recognition
NLTK	Process text as natural language received in files or through speech recognition
eSpeak	Convert text to speech emitted by the robot's speaker
VOSK	Convert the audio signal obtained from the microphone into text rendered in natural language

D. *Communications scheme*

As mentioned earlier, the robot is remotely controlled from the programming environment, provided both are part of the same LAN, and both the client and server can access the Internet independently. Fig 6 illustrates the SSH protocol for instruction execution, HTTP for real-time multimedia transmission, and SFTP for secure file transfer.

Standardized protocols enable only one-way instruction, but proposals such as Perzanowski et al [9] suggest multimodal communications, allowing for more intuitive and didactic programming by considering multiple means of interacting with the robot.

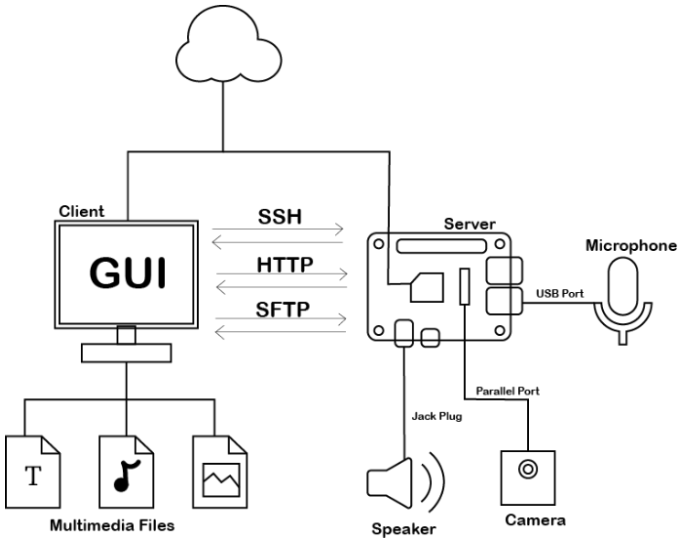

Fig. 6. Scheme of communications and multimodal interaction

In order for the robot to interact with its surroundings, it relies on communication devices such as a microphone, horn, and camera. Additionally, the robot can access multimedia files from a controller or cloud to obtain information from the virtual environment. By processing data from both environments, the robot is able to perform actions that have been programmed by the user.

E. Artificial intelligence models

Information obtained from input devices is processed using artificial intelligence models, which extract characteristics and patterns to make decisions. Each model is simplified into a node, with properties, inputs, and outputs configured to run in a function on the controller. The specific models used, along with the library that provides tools to manage them, are listed in Table IV.

Table IV. Models used to identify patterns in signals

Model	***Description***	***Library***
Haar Cascade Classifiers	High-speed classifiers optimized for real-time recognition of objects under pre-trained models	OpenCV
NLP	Natural language processing algorithms focused on evaluating text data to recognize human language patterns	NTLK
Neural networks LSTM	Short-term memory algorithms focused on the detection of text patterns in images, automatic translation or speech recognition	Tesseract
DNN-HMM	Hybrid model of deep neural networks combined with the hidden Markov model for real-time speech recognition	VOSK

F. Compilation

The development environment allows for the creation and storage of nodes in a canvas, which can be saved as a JSON file containing the necessary information for later use. This file can also be used to generate the main file to be executed in the controller, according to the structure shown in Figure 7. The compiler uses a JSON configuration file to translate each node to its respective Python function and generate the final file for execution.

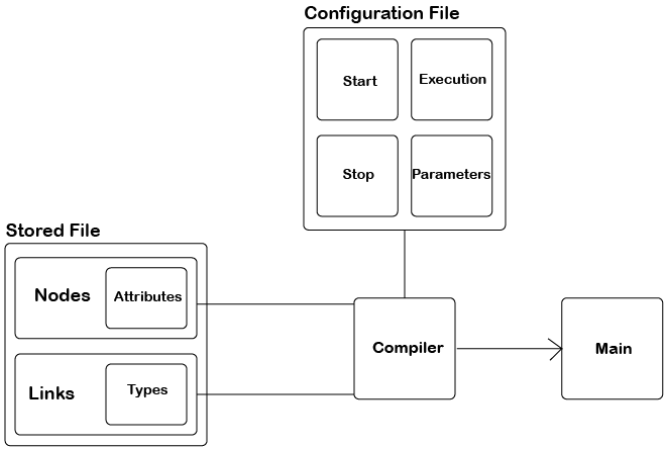

Fig. 7. Compilation scheme

III. EXPERIMENTATION

During the experimentation, the programming environment was presented to 3rd, 4th and 5th grade students of an educational institution that did not offer a robotics course in its curriculum. First, the students took a test of prior knowledge, which provided data to compare at the end of the activity. They then participated in an introductory theoretical talk on "Educational Robotics", which provided them with the necessary information to develop a program using the provided interface.

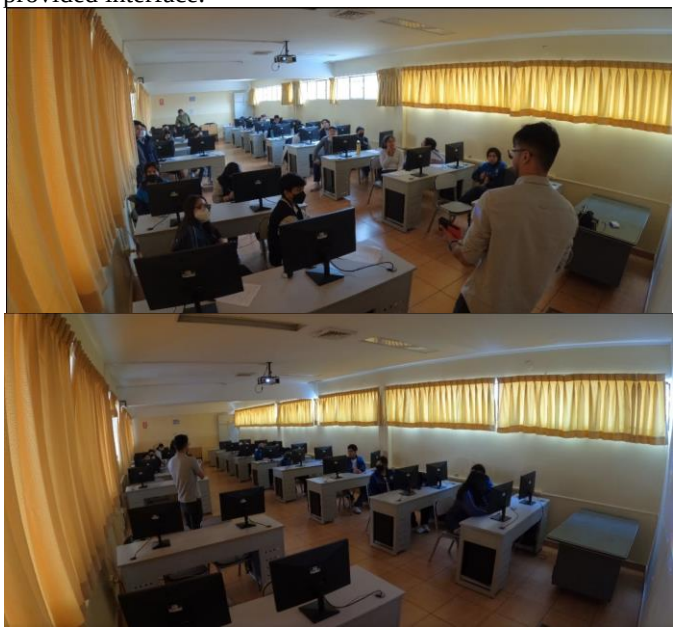

Fig. 8. (a) Upper: High School 5th Class, (b) Lower: High School 3rd Class

They were then given a self-made modular robot controlled by the Raspberry Pi to test the algorithms they created with the knowledge they gained. Finally, the students evaluated the class and the programming environment

Fig. 9. Test of a speech instruction recognition algorithm

During the experimentation, one of the main complications encountered was the difficulty in addressing the individual knowledge biases of each student within the limited number of sessions. Additionally, the sample size was a limitation for the tests conducted, although the experimental procedure can be easily replicated using the designed programming environment.

Fig. 10. Self-made Modular Robot

IV. Results

After comparing the results of the "prior knowledge test" and the "final test," it was observed that there was an increase in students' knowledge and understanding of robotics and educational programming. This finding is supported by Fig. 11, which illustrates the difference in students' knowledge levels before and after the session in response to test questions.

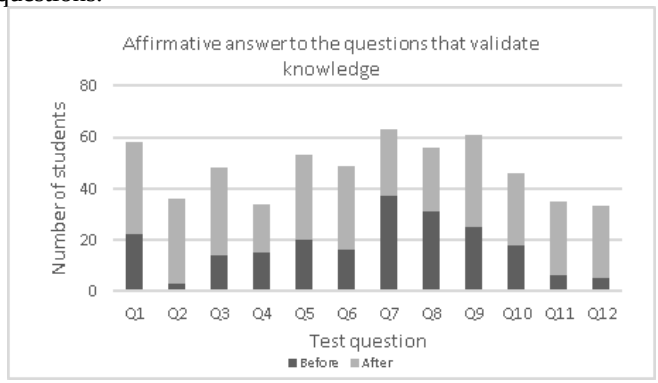

Fig. 11. Contrast graph between prior knowledge and acquired knowledge

Furthermore, the programming interface validations were analyzed and categorized, as depicted in Fig. 12. The results indicated that 93% of students characterized the interface as "friendly" or "very friendly," while 61% reported learning "too much" in the 80-minute class session, and only 7% believed that they learned "little." Additionally, the data showed that 32% of the students were "very willing" to pursue a career in STEM-related fields.

The study reveals that the educational activity positively impacted students' knowledge of robotics and programming, as well as their perception of the user-friendliness of the programming interface. The results suggest that the short 80-minute class session was effective in promoting students' interest in STEM careers.

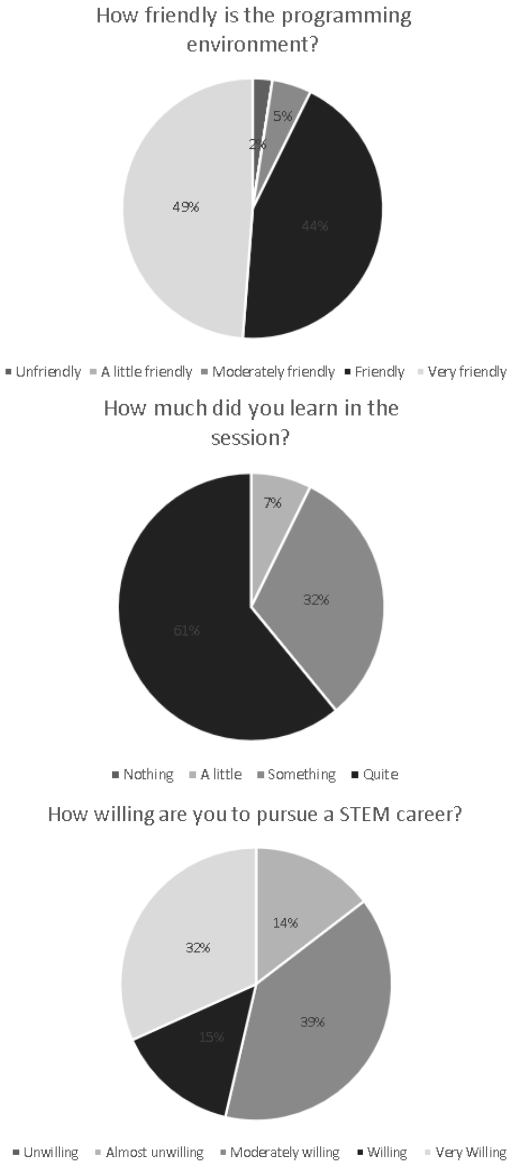

Fig. 12. *Pie charts of validations. (a) Top: "Programming Environment", (b) Center: "How much did you learn in the session?", (c) Bottom: Pursuing a "STEM" career* (c) Bottom: Pursuing a "STEM" career

IV. CONCLUSIONS

This paper describes the design and development of a graphical programming environment that utilizes the reactive paradigm to enable multimodal control of an intelligent robot powered by Raspberry Pi. The paper includes a discussion of programming paradigms, development environment architecture, and controller software. Additionally, the communication scheme and artificial intelligence models used in the development process are presented.

To validate the software's acceptance, a didactic class was conducted in an educational institution, where 93% of the students characterized the software as "friendly" or "very friendly," and 61% reported learning "too much" in a short 80-minute class session.

Overall, this research resulted in the creation of a user-friendly, intuitive, and didactic programming environment that integrates artificial intelligence functions into a multimodal robotic system controlled by Raspberry Pi..

V. ACKNOWLEDGMENT

To the Research Directorate of the Peruvian University of Applied Sciences for the nuptial support for the realization of this research work UPC-EXPOST-2022-2

VI REFERENCES

[1] Anwar, S., Bascou, N. A., Menekse, M., & Kardgar, A. (2019). A systematic review of studies on educational robotics. Journal of Pre-College Engineering Education Research (J-PEER), 9(2), 2.

[2] Tiryaki, A., & Adigüzel, S. (2021). The Effect of STEM-Based Robotic Applications on the Creativity and Attitude of Students. Journal of science learning, 4(3), 288-297.

[3] Sakulkueakulsuk, B., Witoon, S., Ngarmkajornwiwat, P., Pataranutaporn, P., Surareungchai, W., Pataranutaporn, P., & Subsoontorn, P. (2018, December). Kids making AI: Integrating machine learning, gamification, and social context in STEM education. In 2018 IEEE international conference on teaching, assessment, and learning for engineering (TALE) (pp. 1005-1010). IEEE.

[4] Goh, H., & Ali, M. B. B. (2014). Robotics as a tool to stem learning. International Journal for Innovation Education and Research, 2(10), 66-78.

[5] Savard, A., & Freiman, V. (2016). Investigating complexity to assess student learning from a robotics-based task. Digital experiences in mathematics education, 2(2), 93-114.

[6] Mallik, A., Rahman, S. M., Rajguru, S. B., & Kapila, V. (2018, June). Fundamental: examining the variations in the tpack framework for teaching robotics-aided STEM lessons of varying difficulty. In 2018 ASEE Annual Conference & Exposition

[7] Scaradozzi, D., Screpanti, L., Cesaretti, L., Storti, M., & Mazzieri, E. (2019). Implementation and assessment methodologies of teachers' training courses for STEM activities. Technology, Knowledge and Learning, 24(2), 247-268.

[8] Bindu, R. A., Alam, S., & Neloy, A. A. (2019). A Cost-Efficient Multipurpose Service Robot using Raspberry Pi and 6 DOF Robotic Arm. Proceedings of the 2019 2nd International Conference on Service Robotics Technologies - ICSRT 2019. doi:10.1145/3325693.3325701

[9] Vega, J., & Cañas, J. M. (2018). PiBot: An open low-cost robotic platform with camera for STEM education. Electronics, 7(12), 430.

[10] Arvin, F., Espinosa, J., Bird, B., West, A., Watson, S., & Lennox, B. (2019). Mona: an affordable open-source mobile robot for education and research. Journal of Intelligent & Robotic Systems, 94(3), 761-775.

[11] Liang, Y. S., Pellier, D., Fiorino, H., & Pesty, S. (2021). iRoPro: An interactive Robot Programming Framework. International Journal of Social Robotics. doi:10.1007/s12369-021-00775-9

[12] Pacheco, H., & Macedo, N. (2020, November). ROSY: An elegant language to teach the pure reactive nature of robot programming. In 2020 Fourth IEEE International Conference on Robotic Computing (IRC) (pp. 240-247). IEEE. doi: 10.1109/IRC.2020.00045.

[13] Berenz, V., & Schaal, S. (2018). Playful: Reactive Programming for Orchestrating Robotic Behavior. IEEE Robotics & Automation Magazine, 1–1. doi:10.1109/mra.2018.2803168

[14] Perzanowski, D., Schultz, A. C., Adams, W., Marsh, E., & Bugajska, M. (2001). Building a multimodal human-robot. IEEE Intelligent Systems, 16(1), 16–21. doi:10.1109/mis.2001.1183338

[15] Marques, L. S., Gresse von Wangenheim, C., & Hauck, J. C. (2020). Teaching machine learning in school: A systematic mapping of the state of the art. Informatics in Education, 19(2), 283-321.

[16] Milagros Loayza, Juan Alfaro, Leonardo Vinces, and Christian del Carpio, " A Novel Mechanical Design for a mobile robot that carries a load of 30 Kg," 17th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Industry, Innovation, And Infrastructure for Sustainable Cities and Communities" (LACCEI), 2019, doi: /10.18687/LACCEI2019.1.1.124

[17] Mijail Guerrero, Leonardo Vinces, Christian del Carpio, "Development of a Hand Prototype for Gripper Explosive Grenade of Pineapple Type by Applying Soft Robotic Elements and Embedded Sensors," 17th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Industry, Innovation, And Infrastructure for Sustainable Cities and Communities" (LACCEI), 2019, doi: 10.18687/laccei2019.1.1.274

[18] Julio Artica, Marco Klepatzky, Leonardo Vinces, Christian del Carpio, "Development of a positioning system using hybrid control to trace a fixed trajectory applied to a tracked mobile robot," 17th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Industry, Innovation, And Infrastructure for Sustainable Cities and Communities" (LACCEI), 2019, doi: 10.18687/laccei2019.1.1.357

[19] Chávez, L., Cortez, A., Vinces, L. (2022). A Strategy of Potential Fields and Neural Networks in the Control of an Autonomous Vehicle Within Dangerous Environments. In: Iano, Y., Saotome, O., Kemper Vásquez, G.L., Cotrim Pezzuto, C., Arthur, R., Gomes de Oliveira, G. (eds) Proceedings of the 7th Brazilian Technology Symposium (BTSym'21). BTSym 2021. Smart Innovation, Systems and Technologies, vol 295. Springer, Cham. https://doi.org/10.1007/978-3-031-08545-1_43

[20] J. Nuñez, N. Rivas and L. Vinces, "A design of an autonomous mobile robot base with omnidirectional wheels and plane-based navigation with Lidar sensor," 2022 Congreso Internacional de Innovación y Tendencias en Ingeniería (CONIITI), Bogota, Colombia, 2022, pp. 1-4, doi: 10.1109/CONIITI57704.2022.9953628.

[21] B. Díaz, N. Pacheco and L. Vinces, "Integration of a robotic arm Lynxmotion to a Robotino Festo through a Raspberry Pi 4," 2022 IEEE International Conference on Automation/XXV Congress of the Chilean Association of Automatic Control (ICA-ACCA), Curicó, Chile, 2022, pp. 1-5, doi: 10.1109/ICA-ACCA56767.2022.10005932.