Development of a multi-robot web interface for applied learning of industrial robotics in Vocational Education and Training

Yuri Unamuno Arriola[1]*, Iban Arruti[2]

[1] *CIFP Repelega LHII*

[2] *CPIFP Salesianos Urnieta LHIPI*

**Correspondence: yuri.unamuno@cifprepelega.eus*

DOI: https://doi.org/10.56007/arrivet.v1i3.62

ABSTRACT:

This article presents the development of FP Robotik Interface, an educational web interface for programming, simulating and verifying industrial robots in Vocational Education and Training. The project was created to help students understand the logic of a robotic task before working directly with each manufacturer's specific environment. The tool enables visual block-based programming, generates a neutral representation of the program, displays a simplified 3D simulation of the TCP path and produces industrial code, especially for Universal Robots, with possible extension to ABB and Fanuc. It also includes a telemetry-based verification workflow: a Siemens IOT2040 gateway running Node-RED receives robot data through RTDE, Modbus TCP and a STEP socket, normalises the information and publishes it to the web interface using MQTT. The interface can compare planned steps with points reached by the real robot, introducing concepts such as traceability, diagnosis and error analysis. The system has been developed as an educational prototype over one academic year and should not be considered a certified industrial product or a complete digital twin.

Keywords: industrial robotics; Vocational Education and Training; visual programming; telemetry; educational IoT

1. Introduction

Teaching industrial robotics in Vocational Education and Training (VET) requires connecting abstract concepts with workshop practice. Students must learn to interpret coordinates, poses, TCPs, trajectories, tools, speeds, reference frames, automatic cycles and safety procedures. However, a recurring difficulty appears in the classroom: understanding a work sequence visually does not always mean being able to translate it into the real robot language or to anticipate its physical behaviour in an industrial cell.

This difficulty increases when schools have robots from different manufacturers. Universal Robots, ABB and Fanuc share general programming principles, but they use different languages, environments and conventions. As a result, part of the learning time is spent adapting to a specific software package before students have internalised the common logic of the robotic task.

FP Robotik Interface has been developed to cover this intermediate space between a purely educational tool and a professional industrial environment. The proposal does not aim to replace PolyScope, RobotStudio or any other manufacturer-specific system. Its purpose is to offer a prior educational layer that allows students to create programs with blocks, review the structure of the program, simulate the TCP path in a simplified way, generate industrial code and then verify the real execution using telemetry data.

The central idea of the project is to close a complete cycle of applied learning: program design, simulation, code generation, controlled execution on a real robot and comparison between what was planned and what was executed. In this way, students do not only check whether the robot moves; they analyse which point should have been reached, which point was actually reached and what technical hypotheses may explain the difference.

2. Educational context and need

In many VET centres, the industrial robot is a shared, expensive resource subject to strict safety rules. It is not always possible for all students to program directly on the controller for enough time to test, make mistakes and correct them. For this reason, it is useful to have a preliminary tool that prepares the work before the real cell is used.

Visual programming provides an accessible entry point, but it can be insufficient if it is not connected to industrial concepts. Therefore, the blocks in FP Robotik Interface represent real robotics elements: tool, TCP, point, pose, joint movement, linear movement, gripper action, approach, retreat, palletising and step verification.

The project is framed as applied innovation in VET. It combines web development, visual programming, industrial robotics, IoT communications, data gateways and results analysis. It also enables challenge-based learning activities in which students design a solution, simulate it, generate code, execute it under supervision and evaluate the data obtained.

3. Objectives

The general objective is to develop a multi-robot web interface that supports applied learning of industrial robotics in Vocational Education and Training. The tool must be simple enough to introduce students to robot programming, but close enough to industrial practice to generate programs that can be reviewed and transferred to a real environment.

The specific objectives are to separate the logic of the program from the specific robot by means of a neutral model; generate industrial code for different robot families; visualise the programmed sequence in a simplified 3D simulation; incorporate a program versioning system; receive telemetry from a real robot; compare planned points with reached points; and facilitate transfer between VET centres.

Technical objective	Educational value
Neutral model	It helps students understand the common logic of a robotic program before studying a manufacturer-specific syntax.
3D simulation	It helps review the sequence, approach movements and the conceptual TCP path.
Industrial code	It connects visual blocks with real languages such as URScript or RAPID.
Telemetry	It transforms robot execution into analyzable data.
Planned-versus-real comparison	It introduces traceability, error, diagnosis and technical validation.

Table 1 Mapping of technical objectives to educational outcomes.

4. Interface design and neutral model

The web interface is organised around Blockly, a visual editor that allows programs to be built using blocks. Students select blocks from different categories and combine them to create a robot sequence. This approach lowers the initial barrier to entry, because the first task is not to memorise syntax, but to understand the logic of the process.

The key architectural element is the neutral model. Instead of generating code directly from the visual blocks for a specific robot, the system first transforms the program into a common internal representation. This model includes information about the program name, the selected robot, the active tool, defined points and poses, movements, actions, planned steps and verification identifiers.

An important didactic decision has been to distinguish clearly between a point and a pose. A point contains only X, Y and Z coordinates. A pose also adds orientation through Rx, Ry and Rz. This difference is essential in industrial robotics, because reaching a spatial position is not enough; many tasks require the tool to reach the point with a specific orientation.

The use of a planned-steps list, or plannedSteps, makes it possible to expand high-level instructions into verifiable actions. For example, a pick-and-place operation can be decomposed into approach, descent, gripper closing, retreat, movement to the place point, descent, opening and retreat. This structure is later used both for simulation and for comparison with the real robot.

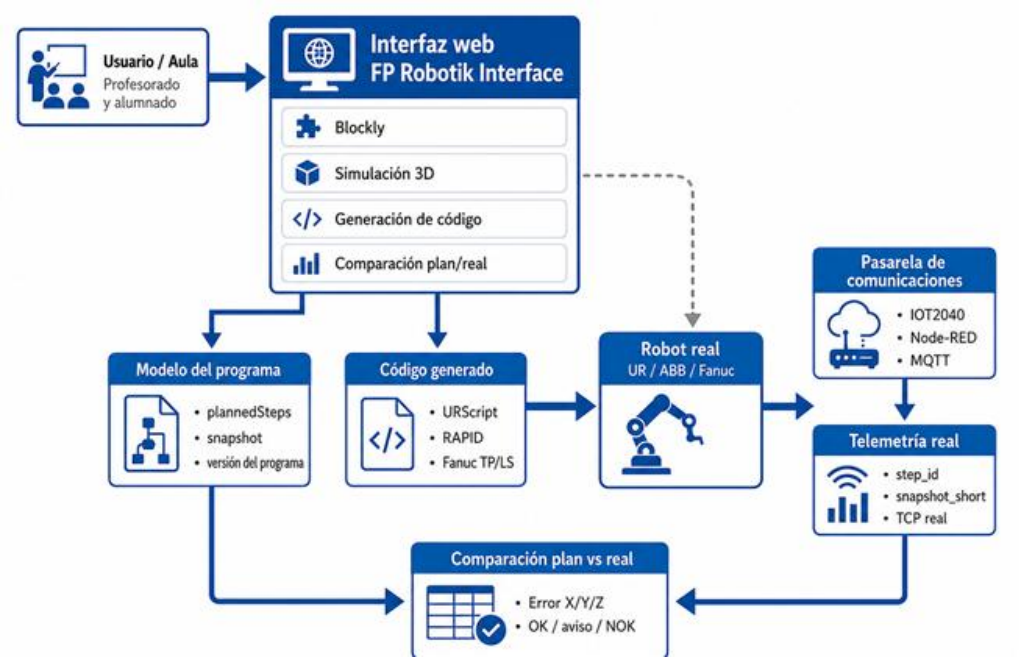

Figure 1 General architecture of the interface and its relationship with the communication gateway.

Component	Function
Web interface	Main working screen for teachers and students.
Blockly	Visual editor for building programs with blocks.
Internal compiler	Converts blocks into a neutral program model.
Generators	Produce code for UR, ABB or Fanuc robots.
3D simulator	Displays a simplified visual representation of the sequence.
Comparator	Calculates differences between the plan and the real execution.

Table 2 Relationship between technical objectives and educational values.

5. Code generation and 3D simulation

The interface includes a lightweight 3D simulation focused on the expected TCP path. This simulation is intentionally simplified. It does not reproduce the full kinematics of each robot, joint limits, collisions or dynamic behaviour. Its educational function is to show the work sequence, the conceptual trajectory and the order in which actions are executed.

This limitation is not a weakness if it is made explicit. In the classroom, the simulator helps students detect basic errors before going to the real robot: inverted pick and place points,

inconsistent approach heights, incorrect gripper sequences or unexpected palletising directions. At the same time, it allows the teacher to explain that a visual simulation does not replace risk assessment or validation in the real cell.

Code generation links visual programming with real industrial languages. The most advanced development has been carried out for Universal Robots through URScript. The architecture also includes outputs for ABB using RAPID and for Fanuc using a representation close to its programming logic. From an educational point of view, the value is not only obtaining the code but being able to compare how the same task is expressed in different languages.

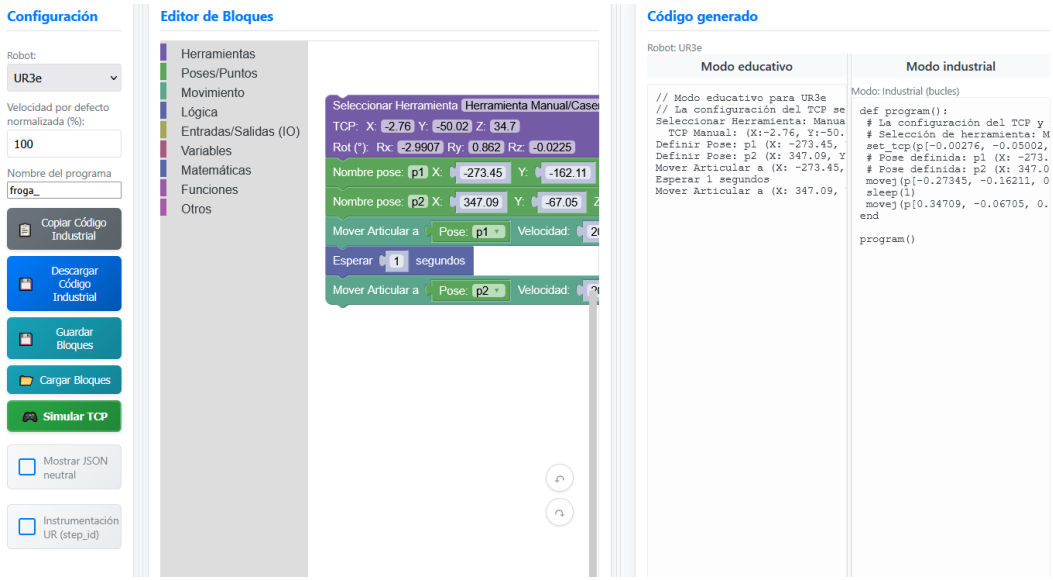

Figure 2 General view of the interface: block editor, generated code and working area.

6. Telemetry and planned-versus-real comparison

Telemetry-based verification is one of the main contributions of the project. The system can generate instrumented code so that the robot reports when it finishes certain steps. This information is received by the communication gateway and published to the web interface, where it can be associated with the previously generated plan.

In the architecture developed for Universal Robots, a Siemens SIMATIC IOT2040 running Node-RED acts as a gateway between the robot and the web. Node-RED receives data through Dashboard Server, RTDE, Modbus TCP and a STEP socket. It then normalises the messages and publishes them via MQTT. The web interface does not communicate directly with the robot; it receives already processed data from the broker.

The STEP socket allows the URScript program itself to send a step capture when it reaches a specific point. The data line includes the program version identifier, the step identifier, the event counter and the actual TCP position. The interface can then compare planned X, Y and Z values with real X, Y and Z values and calculate the error in millimetres.

To avoid incorrect comparisons, the system includes the concept of a snapshot, or program version. A real execution should only be compared with the plan of the same version. If students modify the blocks after executing the robot, previous data should no longer be interpreted as equivalent to the open program. This traceability introduces a valuable educational practice: it is not enough to receive data; its origin, version and relation to the plan must also be known.

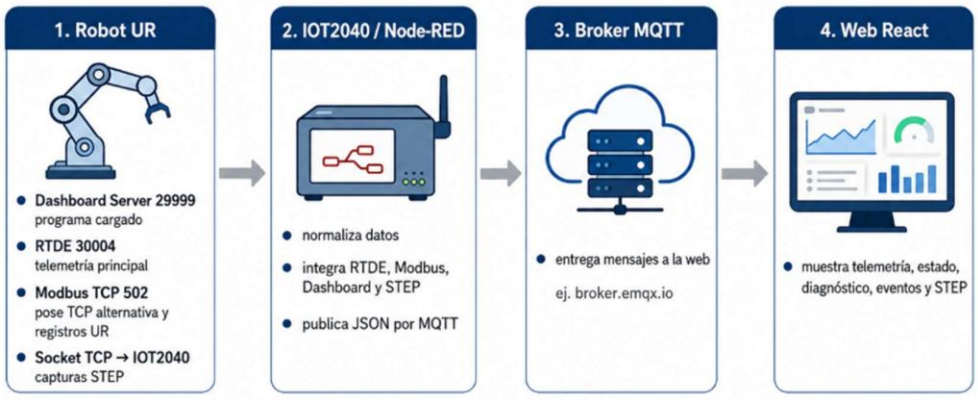

Figure 3 Communication architecture between the UR robot, IOT2040/Node-RED, MQTT broker and web interface.

Condition for comparison	Reason
Telemetry from the same program	Avoids mixing data from different executions.
Matching version identifier	Ensures traceability.
step_id existing in the plan	Links each capture with its planned point.
Correct TCP	Avoids systematic position errors.
Capture at the correct point	Ensures that the end of the intended movement is measured.

Table 3 Validation conditions for telemetry data comparison.

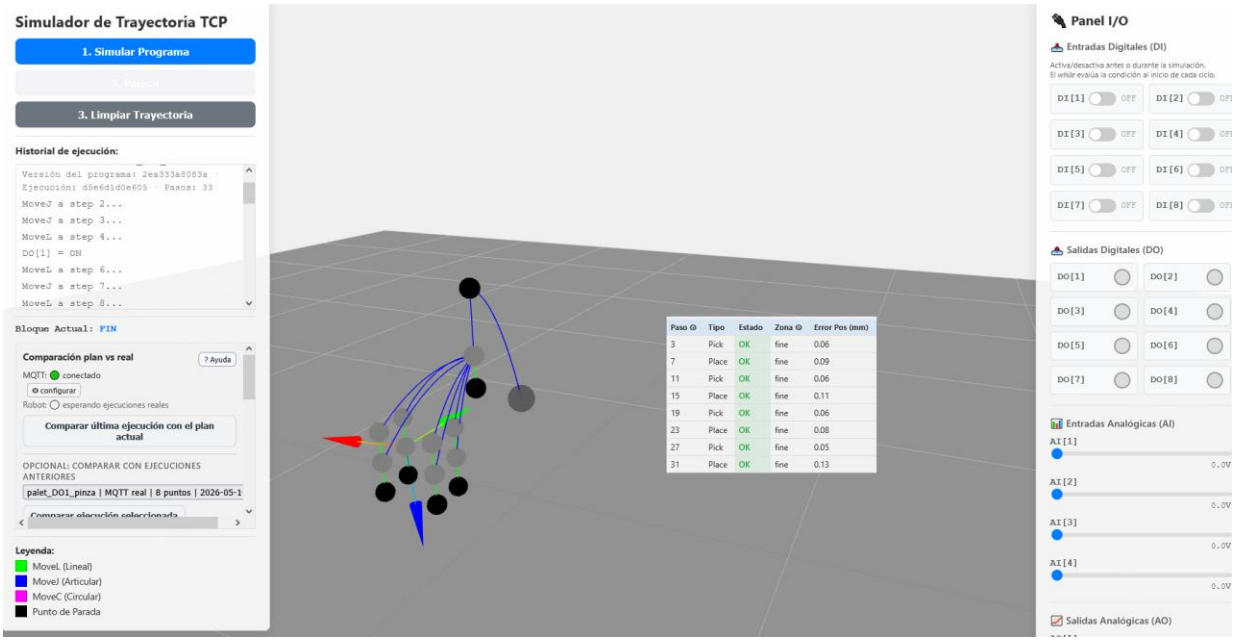

Figure 4 TCP trajectory simulator and planned-vs-real comparison in 3D Simulation.

The figure shows the simulated TCP path, the execution history, MQTT connectivity status, and the comparison table between planned steps and real robot captures. In this example, the pick-and-place steps are matched successfully, with position errors below 0.15 mm.

7. Transfer between VET centres

The project has been documented with a clear transfer-oriented approach. In addition to the interface user manual, a specific guide has been prepared to import and configure the Node-RED template on an IOT2040 gateway. This separation allows another centre to adapt the solution without having to understand or reprogram the entire internal logic.

The Node-RED template concentrates the centre configuration, robot IP address, gateway IP address, MQTT topics and main physical connections. The idea is that the receiving teacher only modifies the values marked as editable, checks the connections and tests communication step by step. This approach improves replicability and reduces the risk of errors when the system is transferred to another robot or local network.

In the medium term, this architecture opens a line of collaboration between VET centres. Different schools could share exercises, programs, configurations and telemetry results. In a later phase, a network of connected educational robots could be explored, always under supervision, with local safety control and without turning the web into a remote-control system.

8. Results and discussion

The development has produced a functional prototype with visual programming, a neutral model, simplified 3D simulation, URScript code generation, version management, telemetry reception through MQTT and comparison between planned steps and real positions. A communication architecture based on IOT2040, Node-RED, RTDE, Modbus, STEP socket and MQTT has also been documented.

The tests have shown the importance of aspects that may appear secondary in the classroom, such as correct TCP definition, tool orientation, consistency between points and poses, and correspondence between the executed program and the saved version. These problems are not merely technical failures; they become didactic opportunities to explain how an apparently correct program can produce unexpected behaviour in a real machine.

The project has also reinforced a methodological idea: planned-versus-real comparison transforms robotics into an analytical activity. Students can discuss why a deviation appears, whether the error is systematic, whether it comes from the TCP, pose definition, communication, the tool or an incorrect program version. This type of reasoning is closer to real technical practice than the simple execution of a closed routine.

9. Limitations and safety

FP Robotik Interface should be understood as an advanced educational prototype, not as a finished industrial product. Although functional tests have been performed on the main workflows, the interface has not been validated in 100% of the possible combinations of robot, blocks, configurations and telemetry.

The 3D simulation is not a complete digital twin. It does not perform exhaustive validation of collisions, joint limits, reachability or functional safety. Therefore, execution on a real robot must always be carried out under teacher supervision and following the centre's safety procedures. Before loading a program, positions, orientation, TCP, tool, payload, speeds, work limits and possible physical interferences must be reviewed.

The telemetry comparison does not turn the interface into a remote-control system either. Its purpose is supervision, step validation and learning through data analysis. This limitation must be explained explicitly to students, because it helps distinguish between simulation, verification, industrial control and certified safety.

10.Methods and development process

The project has been developed through an iterative methodology of prototyping, testing and improvement. First, the main programming blocks and the interface structure were defined. Then the compilation, code generation and simulation functions were separated. Finally, versioning, telemetry and planned-versus-real comparison mechanisms were added.

Validation has been carried out progressively: browser-based interface tests, review of generated programs, trajectory simulation, MQTT communication tests and partial tests with a UR robot. Documentation has been produced in parallel so that other centres can understand the system and replicate it.

AI-based assistance tools were used during development and writing to support document structuring, text revision, improvement proposals and programming tasks. Authorship, technical decisions, functional validation, interpretation of results and final responsibility remain with the authors.

11.Conclusions

FP Robotik Interface demonstrates that a multi-robot web interface can act as a bridge between visual programming and real industrial robotics in Vocational Education and Training. The tool allows students to design programs using blocks, understand their structure, visualise a simplified simulation, generate industrial code and compare real execution with the planned path.

The main contribution of the project is the integration of several learning layers in a single workflow: programming, simulation, code generation, industrial communications, telemetry and results analysis. This integration promotes advanced technical skills and enables interdisciplinary activities linked to industrial digitalisation.

Although the system still requires full validation and improvement in several modules, the prototype developed provides a solid basis for future work: extending compatibility with ABB and Fanuc, improving simulation, incorporating orientation analysis, strengthening user management, documenting didactic exercises and moving towards shared experiences between VET centres.

References

ABB Robotics. (2024). *RAPID overview*. ABB Robotics documentation.

Arruti, I., & Unamuno Arriola, Y. (2026a). *FP Robotik Interface multi-robot interface user manual* [Unpublished internal project document]. CIFP Repelega LHII / Salesianos Urnieta.

Arruti, I., & Unamuno Arriola, Y. (2026b*). Import, configuration and commissioning of the Node-RED template: UR - IOT2040 - Node-RED - RTDE - Modbus - MQTT - Web React* [Unpublished internal project document]. CIFP Repelega LHII / Salesianos Urnieta.

Google Developers. (2024). *Blockly documentation*. https://developers.google.com/blockly

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Prentice Hall.

Node-RED. (2024). *Node-RED documentation*. https://nodered.org/docs/

OASIS. (2019). *MQTT Version 5.0*. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). *Scratch: Programming for all. Communications of the ACM, 52*(11), 60-67.

Siemens. (2024). *SIMATIC IOT2040 documentation*. Siemens Industry Online Support.

Universal Robots. (2024a). *Real-Time Data Exchange (RTDE) guide*. Universal Robots documentation.

Universal Robots. (2024b). *URScript manual*. Universal Robots documentation.

Figures

Figure 1 General architecture of the interface and its relationship with the communication gateway. 5

Figure 2 General view of the interface: block editor, generated code and working area. 6

Figure 3 Communication architecture between the UR robot, IOT2040/Node-RED, MQTT broker and web interface. 7

Figure 4 TCP trajectory simulator and planned-vs-real comparison in 3D Simulation. 8

Tables

Table 1 Mapping of technical objectives to educational outcomes. 4

Table 2 Relationship between technical objectives and educational values. 5

Table 3 Validation conditions for telemetry data comparison. 7

Author contribution

Y.U.A. and I.A. contributed to project conceptualisation, technical documentation, testing and manuscript preparation. The final version was reviewed and approved by the authors.

Declaration of interests

The authors declare that they have no conflicts of interest related to this work.

Acknowledgements

The authors thank the participating centres and the teachers involved in testing, review and documentation of the project, as well as the contributions made during the educational development and transfer process.