CEA Comité Español de Automática
XLVI JORNADAS AUTOMÁTICA CARTAGENA 2025
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Espresso macchiato, por favore: collaborative robotic coffee-making for education

Bartek Łukawski*, Miquel Martín, Carmen Menchén, Edwin D. Oña, Juan G. Victores, Alberto Jardón

RoboticsLab, Department of Systems Engineering and Automation, Universidad Carlos III de Madrid, Avda. Universidad, 30, 28911, Leganés, Spain.

To cite this article: Bartek Łukawski, Miquel Martín, Carmen Menchén, Edwin D. Oña, Juan G. Victores, Alberto Jardón. 2025. Espresso macchiato, por favore: collaborative robotic coffee-making for education. Jornadas de Automática, 46. https://doi.org/10.17979/ja-cea.2025.46.12274

Resumen

Presentamos un ejercicio práctico que involucra el brazo colaborativo robótico ABB "GoFa" CRB 15000-5: una estación de café para una cafetera doméstica convencional con cápsulas. El objetivo es proporcionar un caso de uso práctico e intuitivo para su adopción en sesiones prácticas con estudiantes de grado o máster. Esperamos enriquecer de este modo dichas sesiones, mostrando algunos mecanismos que ofrece el entorno de programación de ABB de cara al desarrollo de interfaces gráficas personalizadas. Exploramos AppStudio, la herramienta software de reciente introducción (enero de 2025) para crear aplicaciones gráficas, así como la más consolidada OmniCore App SDK. En este trabajo, describimos dos nuevas aplicaciones que demuestran la validez de ambas herramientas a través de experimentos tanto sobre el simulador RobotStudio como en un montaje real.

Palabras clave: robótica colaborativa, robótica en educación, interfaces gráficas, interfaces humano-robot.

Abstract

We present a practical exercise involving the ABB "GoFa" CRB 15000-5 collaborative robotic arm: a coffee-making station for a domestic-like conventional coffee pod machine with capsules. It is aimed at providing an intuitive and hands-on use case to be adopted in practice sessions with students, framed within undergraduate or graduate studies. We hope to enrich those sessions by showcasing some mechanisms the ABB programming ecosystem provides in terms of developing custom graphical interfaces. We explore the newly introduced (Jan'25) AppStudio software for graphical applications, along with the well-established OmniCore App SDK framework. A pair of new applications are introduced in this work, showcasing both tools through experiments in the RobotStudio simulator and on a real setup.

Keywords: collaborative robotics, educational robotics, graphical interfaces, human-robot interfaces.

1. Introduction

Late-hour study, continuous stress, final exams, endless projects and deliverables... there are many elements in student life that put a strain on young people seeking their successful graduation. Many find a relief in unhealthy energetic beverages. A traditional solution steps in: coffee, in large amounts. We have combined this basic need for many with a practical use-case that can be enjoyed (and learned from) in laboratory sessions for students in robotics. Figure 1 depicts the setup we have developed and programmed using the available professional tools and a coffee machine we found at home.

Figure 1: Coffee-making work station.

*Corresponding author: blukawsk@ing.uc3m.es

This document is structured as follows. Section 2 explores previous works on this topic. Section 3 describes the robot platform and its custom tool. In Section 4, the proposed architecture is described along with the software used to implement the solutions presented in Section 5. Experiments are shown in Section 6, and conclusions are drawn in Section 7.

2. Background

Collaborative robots, or "cobots", found their way into industry in recent years to achieve safe and effective interaction with human workers while performing industrial tasks. These robots no longer operate in their own designated space in which human presence is to be avoided, as in classical industrial setups. Intrinsic safety measures are implemented, thus avoiding additional external sensors: monitored stop upon workspace violation; hand guiding or teaching (also referred to as "lead-through" control) to physically guide the robot through the desired positions; constant speed and workspace monitoring to react whenever the operator gets close to the robot; and limitation of power and force. Several collaborative scenarios can be drawn according to the relation between the cobot, the operator, and the process: independent (the cobot is aware of the operator, but both perform their own task), simultaneous (concurrent work on the same piece), sequential (tasks are accomplished separately, but in an ordered manner), and supportive (both actors need each other in completing the task) (El Zaatari et al., 2019).

Due to the success and high adoption rate of cobots, learning programs have been introduced in academia. The Wayne State University experience in 2017 derived in creating dedicated teaching modules on their mechanics, safety considerations, programming, kinematics and dynamics (Djuric et al., 2017). "Learning factories" aim to step further into this area by providing a realistic manufacturing environment that fosters learning and teaching. A related pilot program in Vienna stressed on the importance of user interfaces (UIs), categorizing them in accordance to four levels of interaction regarding the human operator: bystander, modifier, programmer and integrator. In terms of usability, it was concluded that the quality of UI design helps increase acceptance and adoption of cobots at the workplace. Small knowledge entities called "learning nuggets" were proposed to set an educational platform that democratizes collaborative robotics in the form of sets of learning concepts and targets that can be consumed in 5–30 minutes (Schmidbauer et al., 2020).

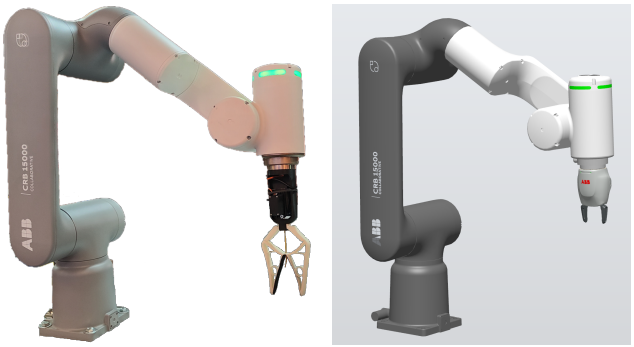
In 2020, a survey conducted in the context of a workshop at the Lawrence Technological University towards studies in Mechatronics and Robotics Engineering (MRE) concluded that system integration skills need to be prioritized over more specialized skills. In particular, general skills such as extraction of system requirements, design to meet specifications, integration of subsystems and preparing documentation have been highlighted (Berry et al., 2020). Another survey in 2021 followed an iterative process comprised of exploration, design and evaluation phases to collect the knowledge, skills, abilities and other requirements (KSAOs) required to prepare production workers and engineers towards human-cobot collaboration (Wolffgramm et al., 2021).

In the RoboticsLab environment the authors of this work collaborate with, previous studies have been conducted involving the "GoFa" cobot described in the following section. Low-cost equipment such as webcams has been explored to showcase visual servoing pick-and-place applications in practice sessions belonging to postgraduate studies in robotics (Oña et al., 2024). Besides, the Externally Guided Motion (EGM) features of ABB controllers has been leveraged to propose additional use cases: path correction of a predefined trajectory using force-torque sensor feedback, 3D surface reconstruction of objects with a depth sensor, teleoperation using a low-cost 3D mouse joystick, and teleoperation through a virtual reality headset (Łukawski et al., 2025; Yepez-Figueroa et al., 2025).

3. Robot Platform

The robotic platform devoted to the coffee-making task was the ABB "GoFa" CRB 15000-5 model. This collaborative robot arm features six degrees of freedom (DoF) in which the last three axes do not share a common origin, as usual in most established industrial robots. This allows to mitigate the effect of kinematic singularities. On each joint, a torque sensor is attached on the shaft to measure external forces and detect collisions. This robot also features a pair of external buttons on the last link for target teaching and enabling the lead-through mode. On the software level, a safety controller monitors the tool center point (TCP) of the robot at all times along with the applied velocity and exerted forces. If the predefined kinematic, dynamic or workspace limits are exceeded, an emergency halt is triggered.

Figure 2 represents the real robot model and its virtual counterpart loaded in the RobotStudio simulator.



(a) Real robot (b) RobotStudio model

Figure 2: ABB "GoFa" CRB 15000-5.

For the purpose of the practice sessions with students developed in the context of this work, a custom gripper tool was designed and 3D-printed, see Figure 3. A single high-efficiency DS3235 servomotor is combined with a fishing line to pull an elastic structure printed with TPU filament. Its elasticity allows to recover the initial position when the string is gradually released. A standard input-output interface is wired to the embedded Arduino Nano v3 and connected to the robot's end-effector port; electrical current is also drawn from the same place to power the board. The RobotStudio model is articulated, i.e. the open-close motion can be simulated.

(a) Real implementation (b) RobotStudio model

Figure 3: Custom 3D-printed flexible gripper tool.

4. Architecture and Software Tools

The overall proposed component architecture is represented in Figure 4. The user is the main actor and initiator of the task the robot system is devoted to complete. Key functions such as low- and high-level motion control and command execution, kinematic calculations, input/output signal handling and safety monitoring are powered by RobotWare, ABB's robot control software. This controller block executes instructions written in RAPID, ABB's proprietary programming language. Most robot capabilities are exposed as RAPID commands and variables, either in vanilla RobotWare controllers or through the activation of Add-Ins such as EGM.

The Robot Web Services (RWS) platform is leveraged to introduce a communications layer in RobotWare controllers, thus allowing to expose motion commands to external devices. RWS introduces the Representation State Transfer (REST) paradigm in ABB robot programming, thus adhering to a standardized and widely adopted architecture used in and beyond robotics. This allows to implement a client-server setup that extends the usual programming tools to other languages (e.g. JavaScript, Python, C++) and frameworks. In the proposed application, a RESTful server managed by the robot controller and a compatible JavaScript client library leverage the HTTPS protocol to enable additional motion control interfaces.

In this work, graphical user interfaces (GUI) have been developed targeting two platforms: user devices that allow web browsing (such as PCs, laptops, tablets or smartphones), and the integrated ABB FlexPendant tablet.

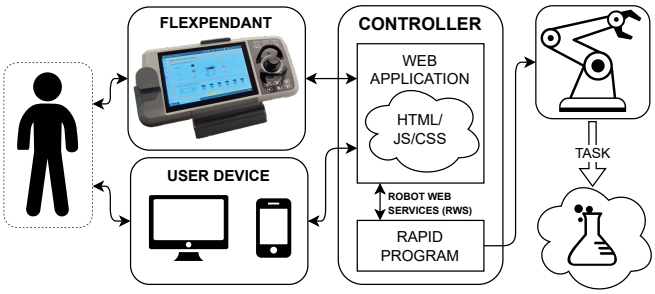


Figure 4: Component architecture.

Figure 5 depicts the coffee-making work station as simulated in RobotStudio. It features a coffee pod machine, three platforms (represented in white color) for storing the cups, capsules and the prepared coffee, and the cups and capsules themselves. All elements have been 3D-modeled and imported into RobotStudio for their visual matching with the real counterparts. The station also includes a table-like platform on which all these elements are placed, which also accurately matches the real setup, both visually and dimensionally. The cup and capsule platforms have been 3D-printed for the real scenario, as previously shown in Figure 1.

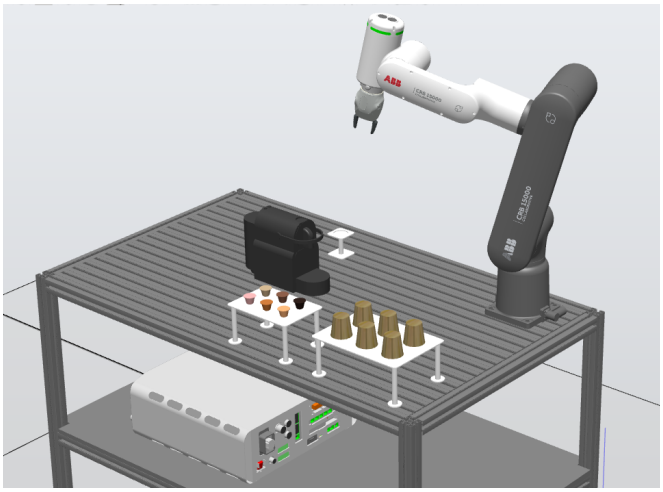


Figure 5: RobotStudio-simulated coffee-making work station.

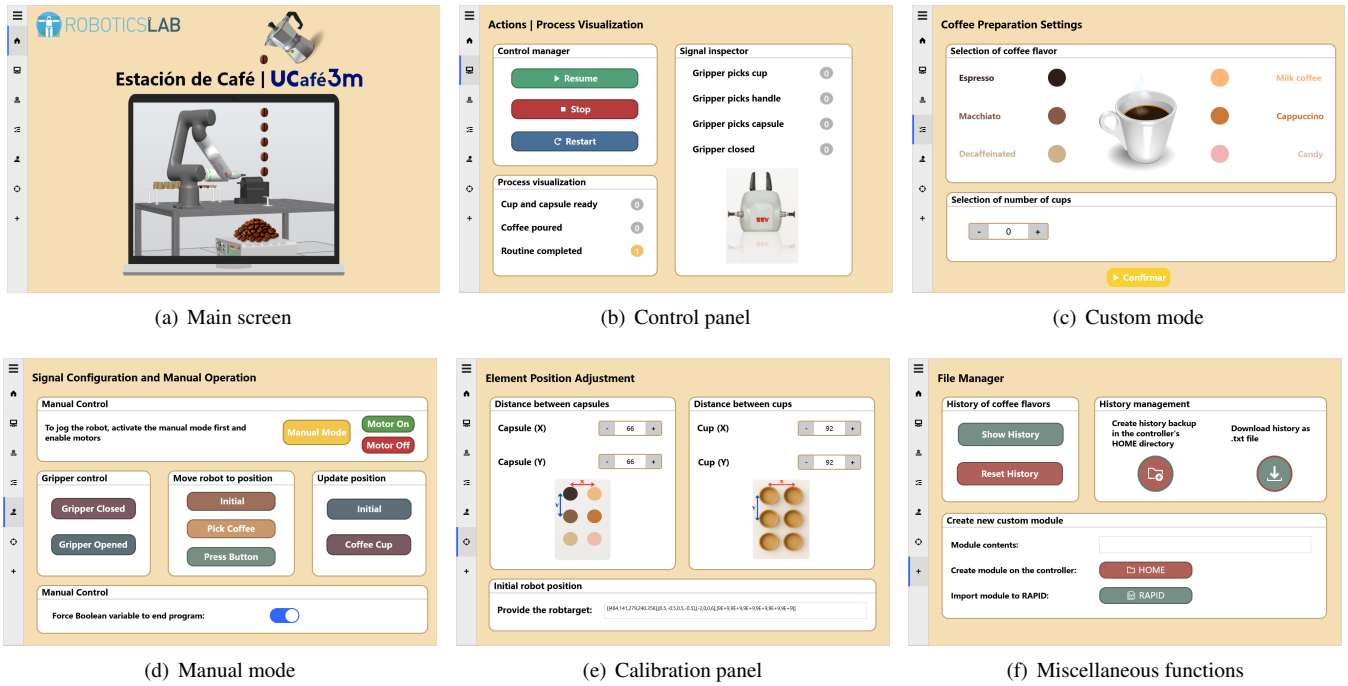
This station was introduced in a previous work to investigate the implementation of safety measures in the RAPID code (Coloma, 2023): TCP speed and distance monitoring between the operator and the robot, and speed and force limits imposed on the TCP. Convenient RAPID constructs, such as TRAP routines (equivalent to interrupt service routines, or ISR, in other programming contexts), contributed to materialize those requirements in the final application.

The software tools involved in the making of the graphical interfaces are publicly distributed and documented in the official ABB developer website[1].

4.1. ABB AppStudio

In January 2025, ABB presented a new software tool for developing GUIs on the FlexPendant, PC, tablets and smartphones. It has been engineered as a no-code editor linked to a RobotWare controller running in RobotStudio. It enables users to add graphical components and widgets to compose a GUI application without resorting to any programming languages, using an intuitive drag-and-drop methodology. In addition, translation of interface elements is supported and simplified, and a screen size selection menu is shown at project start to target the desired device. Projects can be exported in compressed .asppag packages similarly to the Pack&Go files supported by RobotStudio (.rspag extension). Available project templates usually avoid the need to start from scratch.

[1] https://developercenter.robotstudio.com



(a) Main screen (b) Control panel (c) Custom mode

(d) Manual mode (e) Calibration panel (f) Miscellaneous functions

Figure 6: Graphical interface of the application developed with ABB AppStudio.

In our application, we have made extensive use of the supplied library of graphical components ("TComponents"). Among the most relevant, we identified buttons, checkboxes, text inputs, radio and select buttons, toggles and sliders. Many of those elements have associated custom actions defined by JavaScript event handlers. Even though it is devised as a no-code editor, code blocks can be supplied with custom logic in specific action callbacks, e.g. upon clicking a button.

On the programming level, the associated JavaScript library provides access to the RWS client, exposing controller status and configuration, robot signal and RAPID modules. For instance, it is possible to enable or disable motors, start and stop the robot controller, access and modify digital inputs and outputs, extract the value of a RAPID variable and invoke custom RAPID modules.

In addition, a wrapping client layer is provided on top of RWS to provide fine-grained access to other controller characteristics and attributes. Among them, we found particularly useful the ability to access and manipulate the controller's file system, e.g. to create files and directories, write into files, and read from files. Besides, it is also possible to query network status, and manage user access and restrictions. Connection between the AppStudio editor and RobotStudio is seamless: if there are running instances of both, the former can access and act upon the virtual robot controller running on the latter.

4.2. *OmniCore App SDK*

The OmniCore App Software Development Kit (SDK) provides a library of web-related components to develop graphical applications on the OmniCore FlexPendant, that is, the FlexPendant tablet device shipped with RobotWare 7.x ABB controllers. It comprises a JavaScript (JS) client library to interface with the robot controller, and the front-end elements shown in the GUI, backed by HTML and CSS code.

The library of graphical components is slightly richer than AppStudio's, although we have seen that the latter is being gradually improved to support more elements already present in the former. In addition to the ones already mentioned in the previous section, the OmniCore SDK also allows to insert line and pie charts, menus, dialogs and on-screen keyboards. The RWS client exposes an analogous application programming interface (API), which connects to the robot controller through asynchronous JavaScript calls.

AppStudio projects also compile into JS/HTML/CSS code. However, the OmniCore App SDK does require programming knowledge and is therefore less user-friendly in the creation of new GUI apps, although it doesn't impose as many restriction as AppStudio does at this stage of development.

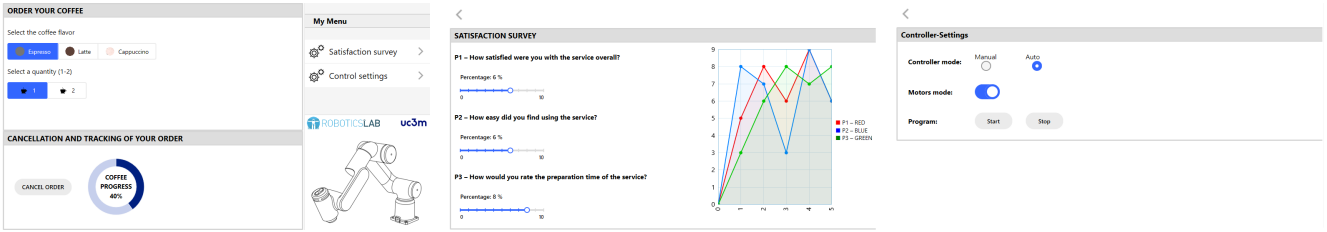
5. Proposed Graphical Interfaces

We propose two graphical applications to showcase the previous tools, both oriented at providing graphical user interfaces for the coffee-making work station.

5.1. *ABB AppStudio*

Figure 6 illustrates the interface developed using AppStudio. It features a side panel with a screen selector to cycle across the different views and modes, namely:

- Main screen (6(a)): splash screen.
- Control panel (6(b)): provides access to main start, stop and restart actions, and allows to inspect the current state of the process in a glimpse. Through additional digital signals defined on the controller, the operator is shown the progress of the coffee preparation through Boolean flags: when the cup and capsules have been placed by the robot, when the coffee is being poured in



(a) Main screen (b) Satisfaction survey (c) Control panel

Figure 7: Graphical interface of the application developed with OmniCore App SDK.

the cup, and whenever the routine has been fully completed. Four signals complement this panel showing the last action executed by the gripper.

- Custom mode (6(c)): selection of the desired coffee flavor (espresso, macchiato, decaffeinated, milk coffee, cappuccino and candy) and total number of cups to be poured the coffee in.
- Default mode: same as custom mode, but the same coffee flavor is poured for all six cups in a continuous fashion, for debugging purposes or targeting automatic demonstrations.
- Manual mode (6(d)): predefined robot arm and gripper motions. To initiate jogging, the operator must switch mode and manually enable motors first, for which convenient buttons are provided. Additional buttons allow to open or close the gripper on demand, move the TCP to a predefined pose (initial, grab cup after pouring coffee, and press the button to start pouring), or update the internal target of start and end poses.
- Calibration panel (6(e)): adjust the distance along axes X and Y between capsules and cups on their respective platforms. On this screen, a text input field allows to insert the `robtarget` definition for the initial robot pose.
- Miscellaneous functions (6(f)): a history of orders is kept on the controller file system. This screen allows to show it using a popup window, clear it, create a backup file, or download it as a text file. Besides, a new RAPID module can be defined and loaded in the controller through another text input field.

The application has been tested both on the OmniCore FlexPendant and on a laptop using a web browser. For the latter case, the IP address, port and the path of the web app on the controller's file system must be provided in the address bar of the selected web browser. User credentials are requested prior to accessing the interface. File storage can be leveraged in a twofold manner: either the history of orders is stored on the controller if running on a FlexPendant, or downloaded via JavaScript onto the user's device (PC, laptop, tablet, smartphone) if running via web browser.

During the development of this application, AppStudio 1.1.0 was released. This version solved a fatal error upon invoking the "closeApplication" method. Besides, JavaScript classes can now be instantiated from the code editor, and several new components and widgets have been added.

5.2. *OmniCore App SDK*

The implementation of our coffee-making app using the OmniCore App SDK is presented in Figure 7. Its main screen (7(a)) features three coffee flavors to be selected from when ordering a coffee, and the quantity has been limited to one or two cups per flavor, in both cases using a toggle widget. On the same screen, a cancellation button is placed, and order tracking has been implemented using a pie chart which shows the progress of the overall coffee-making process.

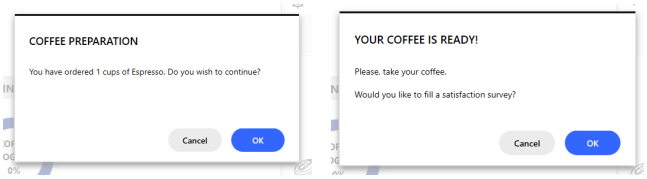
Two additional screens can be browsed from within the main one. Figure 7(b) depicts a satisfaction survey users are asked to complete after each order. The updated results are shown as soon as all questions have been asked. A draggable bar scaled from zero to ten is used as input for the users to provide an answer on said scale:

- How satisfied were you with the service overall?
- How easy did you find using the service?
- How would you rate the preparation time of the service?

Responses are displayed on a colored line graph. All responses are kept on the graph, therefore the history of users' satisfaction is shown until the application is restarted. In addition, these responses are also logged to a text file on the controller's file system, in order to be downloaded later.

The last screen (7(c)) provides access to the robot controller. Through radio buttons and toggles, the mode can be switched from manual to automatic, the motors can be enabled or disabled, and the program can be started or stopped.

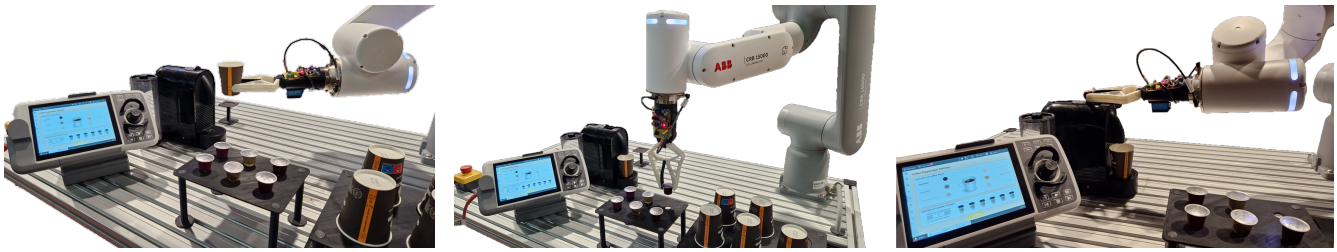
Notification popups are shown on specific events. Figure 8 illustrates the popups shown to the user upon initiating the order and at its conclusion.



(a) Process starts (b) Process ends

Figure 8: Notification popups.

This application was tested on the OmniCore FlexPendant only due to the compatibility this framework offers. Similarly to AppStudio, live updates of RAPID signals and variables can be tracked and represented on this interface.



(a) Pick cup (b) Pick capsule (c) Insert capsule

Figure 9: Stages of the coffee-making experiment conducted on the real robot.

6. Experiments

The graphical applications have been integrated with the coffee-making station. The underlying RAPID code that runs on the robot controller has been adjusted for each application, and signals have been defined accordingly if needed.

The main flow of the process is common to both applications, and is depicted in Figure 9:

1. Upon receiving an order, the robot translates to a safe initial pose.
2. It picks the first available cup and places it on the cup tray of the coffee machine (9(a)).
3. The capsule tray is opened with a horizontal slide motion by pressing the gripper against a wide lid. In simulation, this tray is opened with a handle that must be rotated with a circular motion.
4. Several capsule types are available, according to the flavors the graphical application was designed for. The robot picks the one selected by the user (9(b)), and inserts it into the capsule tray of the coffee machine (9(c)).
5. The robot activates the coffee machine by pressing a button, coffee is poured.
6. When pouring is finished, the robot picks the filled cup and places it on the disposal platform.
7. The robot is ready to process the next order. Internal variables keep track of available cups and capsules.

The process has been validated first in simulation, and then translated into the real scenario. No real coffee was poured, though, since the flexibility of the custom gripper would make it bend when holding the filled cup, thus spilling coffee all over the workspace. Future work is recommended towards improving the gripper design with rigid elements.

7. Conclusions

In this work, a practical application has been presented involving a collaborative ABB robot, custom 3D-printed pieces and tools, and a coffee machine. The original intention has been accomplished: a mundane task (regarded by many) such as preparing coffee was automated and further improved upon with intuitive and practical graphical interfaces. This experience helped us explore and assimilate available tools, including a recent addition to the ABB ecosystem, AppStudio.

We aim to introduce these use cases in academia, targeting undergraduate and graduate studies on industrial and collaborative robotics. Our students can learn from a hands-on experience with collaborative robots performing interesting tasks inspired by the daily experience (of coffee drinkers).

Acknowledgments

The research leading to these results received funding from “iRoboCity2030-CM” (TEC-2024/TEC-62) of Comunidad de Madrid, Dirección General de Investigación e Innovación Tecnológica, 5696/2024; “iREHAB” (DTS22/00105) of Instituto de Salud Carlos III; and EU structural funds.

References

Berry, C., Gennert, M., Reck, R., 2020. Practical skills for students in mechatronics and robotics education. In: ASEE annual conference exposition proceedings.
DOI: 10.18260/1-2--35066

Coloma, J., 2023. Analysis and safety configuration of the ABB CRB 15000 (GoFa) collaborative robot (*Análisis y configuración de seguridad de robot colaborativo ABB CRB 15000 (GoFa)*). Bachelor’s Thesis. Universidad Carlos III de Madrid, Escuela Politécnica Superior.

Djuric, A., Rickli, J. L., Jovanovic, V. M., Foster, D., 2017. Hands-on learning environment and educational curriculum on collaborative robotics. In: ASEE Annual Conference & Exposition.
DOI: 10.18260/1-2--28428

El Zaatari, S., Marei, M., Li, W., Usman, Z., 2019. Cobot programming for collaborative industrial tasks: An overview. Robotics and Autonomous Systems 116, 162–180.
DOI: 10.1016/j.robot.2019.03.003

Łukawski, B., Oña, E. D., Jardón, A., Victores, J. G., Balaguer, C., 2025. Development of educational applications with ABB GoFa collaborative robot using Externally Guided Motion. In: IEEE Int. Conf. on Control, Automation and Diagnosis (ICCAD).
DOI: 10.1109/ICCAD64771.2025.11099395

Oña, E. D., Fraile, S., Balaguer, C., Jardón, A., 2024. Implementation of a visual environment for teleoperation with the ABB GOFA collaborative robot using Externally Guided Motion (EGM) (*Implementación de un entorno virtual para teleoperación de robot colaborativo ABB GOFA usando movimiento guiado externamente (EGM)*). In: XLV Jornadas de Automática. CEA.
DOI: 10.17979/ja-cea.2024.45.10977

Schmidbauer, C., Komenda, T., Schlund, S., 2020. Teaching cobots in learning factories – User and usability-driven implications. Procedia manufacturing 45, 398–404.
DOI: 10.1016/j.promfg.2020.04.043

Wolffgramm, M., Tijink, T., Van Geloven, M. D., Corporaal, S., 2021. A collaborative robot in the classroom: Designing 21st Century engineering education together. Journal of Higher Education Theory and Practice 21 (16), 177–187.
DOI: 10.33423/jhetp.v21i16.4924

Yepez-Figueroa, J. J., Oña, E. D., Balaguer, C., Jardón, A., 2025. Implementation of a computer vision system using a low-cost webcam for robotic manipulation in educational contexts (*Implementación de sistema de visión artificial usando webcam de bajo coste para manipulación robótica en aplicaciones educativas*). In: Simposio CEA de Robótica, Bioingeniería, Visión Artificial y Automática Marina.