2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC

Host Sponsor UNC Charlotte

ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #52570

Work-In-Progress: MATLAB/Simulink-Based Open-Source Robot Control Framework for Undergraduate Robotics Laboratory Curriculum

Ryan Cullen, San Francisco State University
Dr. David Quintero, San Francisco State University

Dr. David Quintero received B.S. degree from Texas A&M University, a M.S. degree from Stanford University, and a Ph.D. from the University of Texas at Dallas all in mechanical engineering. He is now an Assistant Professor of Mechanical Engineering at San Francisco State University representing as a Hispanic-Serving Institution with research focus on design and control of wearable robotic systems, and engineering education in the field areas of mechatronics/robotics.

Work-In-Progress: MATLAB-Based Open-Source Robot Control Framework for Undergraduate Robotics Laboratory Curriculum

Abstract

Providing engineering students with hands-on experience using industrial-grade robot hardware is vital for developing practical competencies for students seeking a career in robotics. However, many instructional robotics teaching labs face barriers due to software dependencies, Linux-based configurations, and the steep learning curve associated with robotic software platforms such as Robot Operating System. To address this challenge, we use MATLAB, a platform widely taught in engineering undergraduate programs, to develop a control pipeline for robotic teaching laboratories. The framework targets Universal Robots' open-source e-series hardware and runs entirely in a Windows environment. The proposed workflow leverages the Robotics System Toolbox in MATLAB to enable live communication via the Real Time Data Exchange protocol over TCP/IP. Explanation of the operation workflow is presented, and preliminary validation demonstrates that the workflow is stable, reproducible, and requires no additional hardware beyond a Windows PC and a standard Ethernet connection. This MATLAB-based infrastructure not only enhances accessibility and safety but also establishes a scalable foundation for laboratory modules in topics such as forward kinematics to efficiently onboard students in robot fundamentals. By lowering technical barriers and leveraging widely adopted educational software with a Windows-compatible, MATLAB-centered workflow, this framework enhances robotics education across a broad range of undergraduate engineering programs and promotes deeper student engagement in robotics and automation.

Introduction

Motivation and Context

Robotics and automation have increased across several industry sectors such as manufacturing, healthcare, logistics, and autonomous systems that demand the next generation engineering workforce to be skilled in these technologies. Undergraduate mechanical and electrical/computer engineering programs are increasingly incorporating hands-on robotics laboratory experience to adequately prepare students for the modern workforce [7, 8]. However, many programs lack easily accessible robotic learning curricula due to the steep learning curves associated with common industrial automation toolchains and complicated proprietary tools that accompany many industrial grade manipulators. When courses do use open-source stacks such as Robot Operating System (ROS), while it offers a vast framework of tools and add-ons for robot development, students can spend substantial time on environment setup and debugging rather than core robotics concepts [2, 9, 12]. Recent surveys of robotics instructors also report heavy reliance on locally developed teaching materials, suggesting a need for shareable, reproducible lab infrastructure [7]. Prior work has extended MathWorks MATLAB with ROS and Gazebo to create teaching toolkits, but these approaches still depend on a full ROS installation and are typically demonstrated in simulation-centric contexts [10]. Low-cost educational robotics platforms have also been adapted to ROS, but these solutions do not directly address the

challenge of teaching with industry standard collaborative manipulators [11]. Thus, there is a need for an approachable yet robust control pipeline for industrial grade robots that is conducive to an undergraduate lab environment.

MATLAB is already a core tool in many undergraduate engineering programs for modeling, simulation, and control [7]. Using familiar software reduces the learning curve and allows students to quickly start building intuition and getting results. At the same time, Universal Robots (UR) is at the forefront of collaborative robot (i.e., cobot) technology, boasting 44 to 47% of the cobot market share as of 2022 [1]. These systems are becoming increasingly popular in industry and education because they are open source platforms that come with built-in safety features and a rich ecosystem of hardware and software accessories [13]. In collaboration with UR, MathWorks provides Robotics System Toolbox for UR Series Manipulators [4], an addon that allows the user to connect a PC to a UR robot via the Real Time Data Exchange (RTDE) protocol developed by UR [14]. This paper aims to conjoin familiar MATLAB based software stacks with cutting edge robotics hardware to provide a basis for comprehensive robotics lab curriculum for undergraduate robotics students with a low barrier to entry.

Limitations of Current Approaches

A common control platform is Robot Operating System (ROS) that enables powerful workflows, but it is widely recognized as challenging to teach at the undergraduate level because it requires significant coding skills as a prerequisite and introduces major configuration and debugging overhead on top of the robotics content [9, 12]. Recent engineering education has noted that ROS has a steep learning curve and high barriers to entry, even when the course design is explicitly built to support it [2]. In practice, ROS-based lab work often assumes comfort with Python, command line tooling, and an understanding of distributed software stacks that undergraduates are often lacking [2, 12]. Platform support can also complicate onboarding. ROS distributions typically target a specific Ubuntu LTS as a primary Tier 1 platform, and while other platforms exist, support level and installation paths vary by distribution [3]. MATLAB, on the other hand, does not suffer from the same dependency issues and runs on operating systems commonly found in school settings, namely Windows 11 and MacOS.

In industry, there are several other major robot manipulator vendors besides UR such as FANUC, KUKA, and ABB. These robots are generally programmed using either a Teach Pendant (TP) or vendor-specific programming tools. TPs are consoles, usually a touch-screen tablet attached to the robot, that allow the user to configure safety settings, networking, limits, and add-ons, as well as to create quick programs using the vendor's interface [13]. They are essential for commissioning and safe operation, but they are not designed for students to implement their own controllers or explore lower-level behavior, and they often obscure the underlying robot mechanics. By providing an external and generalized control scheme, students will be able to apply what they learn to a broad range of robots and applications.

Teach pendant programming can be approachable, but it can hide the underlying control concepts, make complex tasks cumbersome, and lock students into vendor-specific toolchains [7]. ROS can be general and scalable, but it can overwhelm students with prerequisites and

configuration costs [2, 12]. A Windows-native framework that controls a UR robot using MATLAB addresses this gap by offering an approachable but still powerful platform for teaching robotics fundamentals on a real industrial robot. Additionally, it expands the horizon of usability by giving the user access to the plethora of open source tools available online to expand the functionality of the robot beyond what the alone TP can offer. This is especially important in contexts where peripherals such as third party cameras or sensors are involved. Finally, it also enables an "apples-to-apples" simulation-to-hardware workflow, since students can validate models in simulation environments before running on the physical system [10].

Contributions

The contributions of this work are: (1) A Windows-native, MATLAB centered control framework for UR cobots with TPs running Polyscope X (the newest version of UR's proprietary TP software), using RTDE for robot communication [6, 14]. (2) A design rationale explaining why MATLAB and RTDE were chosen over teach pendant-only and ROS-based approaches for an undergraduate lab context [2-4], [12]. (3) Scaffolded student lab modules that begin with safety and basic motion, progress through simulation-based verification, and culminate in forward kinematics derivation and validation for hands-on robot hardware [5, 7, 10].

Methods

System Architecture and Design Rationale

The decision to center the lab framework on MATLAB is based on alignment with typical engineering curricula and the practical constraints of undergraduate lab during a semester or quarter academic system. Since engineering students often learn and use MATLAB in prerequisite courses [15, 16], the programming language is familiar that the lab can start quickly and remain focused on robotics concepts. This also keeps the host development environment consistent with the rest of an engineering program that may already rely on Windows-based machines for coursework, licensing, and existing lab infrastructure. ROS-based pipelines were also explored but eventually rejected due to Linux dependency and previous work suggesting a high barrier of entry for undergraduates [2]. Fig. 1 is a flowchart that outlines the selection process.

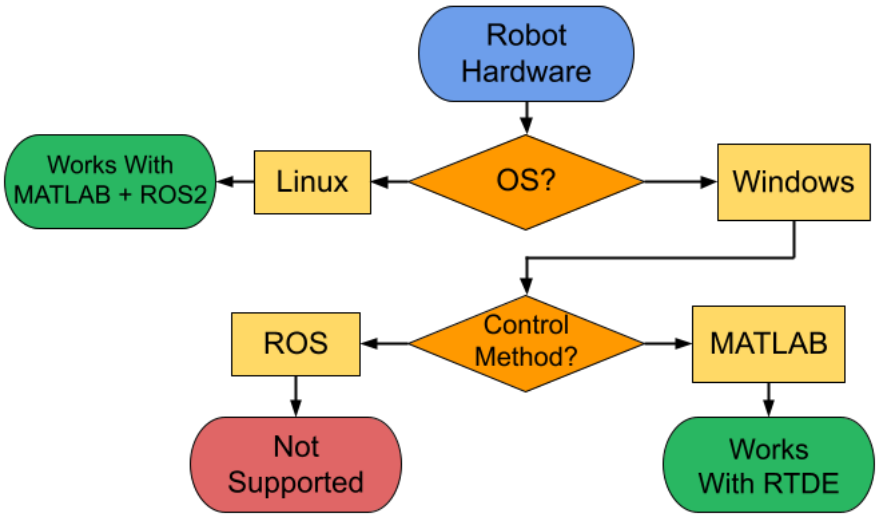

Figure 1: Flowchart detailing the narrowing and selection process of the integrated OS and control platform. Linux is required to use ROS for direct hardware control of a UR robot. RTDE works natively on Windows and has compatibility with MATLAB.

Within MATLAB, the Robotics System Toolbox provides built-in support for common robotics workflows as well as specific integration paths for Universal Robots. RTDE is used as the primary communication layer to support real-time state feedback and control-related data exchange over TCP/IP via an Ethernet connection from the PC to the robot. The intended outcome is a lab environment where students can connect, read robot state, and execute structured control actions without needing external middleware to bridge basic functionality. This framework also provides the ability for users to control the robot in a simulated environment prior to deploying to the actual hardware using UR provided software tools like URSim (a one-to-one physics robot simulator) and URStudio (a browser-based Polyscope X emulator). Fig. 2 below is a flowchart that details the different paths explored for the purpose of simulation.

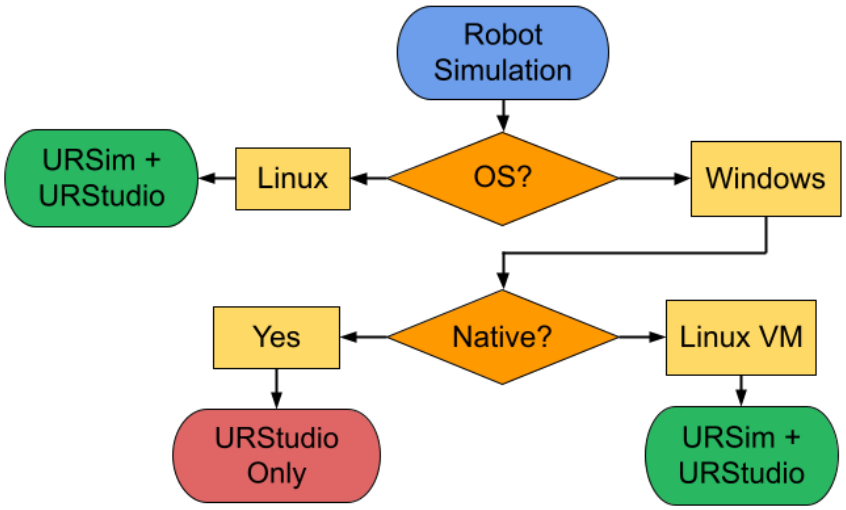

Figure 2: Flowchart detailing the different methods of creating simulation environments. URStudio is a browser-based version of Polyscope X which provides users with all the functionality that TP provides. However, it does not allow you to simulate external control methods like ROS or MATLAB; for that you need URSim and a Linux Virtual Machine (VM). "Native" refers to the base OS without any virtual machine (VM) sublayers.

Through this iterative evaluation process, a Windows-native MATLAB-RTDE configuration emerged as the optimal architecture for the undergraduate lab setting. MathWorks provides documentation for the general RTDE setup applicable to both the UR hardware and URSim [6]; Fig. 3 illustrates this reference configuration. The specific validated implementation developed in this study is presented in the Results section.

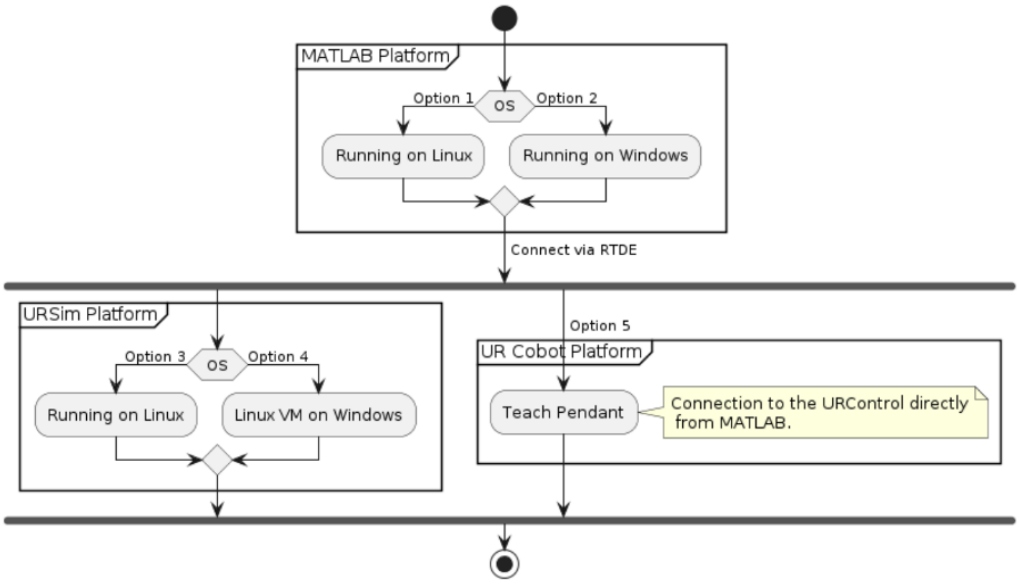

Figure 3: System architecture for a Windows 11 based MATLAB + RTDE control pipeline for UR e-series cobots and the URSim physics simulator

Laboratory Curriculum

The lab curriculum workflow is structured in four modules intended to build student capability progressively while keeping each milestone concrete and allow assessment. The UR3e is used as UR's e-series benchtop robot tailored for classroom environment.

Lab 01: The first module is focused on installation, basic setup, and introduction to RTDE. Students set up the connection, confirm state feedback, and verify that the system can reliably exchange data. Additionally, students will explore Polyscope X within URStudio, familiarizing themselves with the UR's Graphical User Interface (GUI) and some basic robot features. This stage is deliberately small in scope to establish confidence in the environment and reduce the chance that later labs fail due to basic setup issues.

Lab 02: The second module transitions into TP-based joint and end-effector control and safety features. Students interact with motion execution in a controlled way, learn the difference between reading state versus commanding motion, and begin connecting abstract robotics concepts to real robot behavior. Additionally, they will explore the safety features and guardrails that UR platforms expose to their users including safety planes, speed and force limits, emergency stops, and UR's "reduced mode" feature which slows the robot automatically when operating in certain selected areas. The goal of this module is twofold: to provide students with a satisfying and exciting experience by physically commanding the hardware, and to ensure a safe and controlled lab environment going forward.

Lab 03: The third module introduces the concept of Forward Kinematics (FK), a fundamental aspect of robot theory. Students will be asked to derive the FK equations that describes the location of the end effector of their UR robot based on a specific set of joint angles with respect to its robot base frame. A schematic diagram of the UR3e robot will be provided for the purpose of derivation (Fig. 4). To validate their equation, students will first pick and deploy an arbitrary robot pose using URStudio. They will read and record the angles for each joint in this pose, as well as the cartesian coordinates XYZ position of the robot's end effector. They will then validate their derived FK equation by plugging in the recorded joint angles and comparing the result to the recorded position of the end effector.

Lab 04: The final lab will be a practical pick-and-place demonstration on the actual hardware using RTDE commands through MATLAB. Students will first manually record a sequence of joint and gripper positions on the TP, then cycle through these waypoints to successfully pick up and place a cube from one location on the lab bench to another. This will be done using built-in TP commands. Students will then try to replicate this sequence using the same waypoints with analogous RTDE commands. This is meant to demonstrate the differences in workflow between programming the robot using the TP versus externally with MATLAB. The overall goal of this lab is to introduce students to basic TP programming and RTDE functions and give them hands-on experience controlling a real industrial grade robot manipulator.

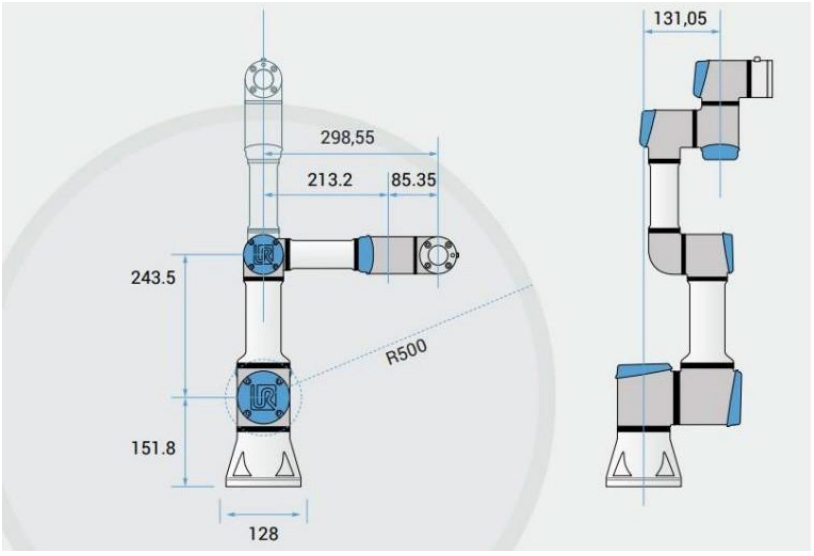

Figure 4: UR3e Schematic Diagram [17]

Evaluation

Student feedback and self-assessment are collected through a structured questionnaire designed to measure learning outcomes and lab usability. The complete instrument is provided in Appendix A.

Results

System Architecture

The primary result of this work is the identification and validation of an optimal system architecture for connecting MATLAB to a UR cobot in an undergraduate lab setting. Fig. 5 presents the finalized architecture, which emerged from the iterative evaluation of operating systems, communication protocols, and simulation environments described in the Methods section. The host machine is a standard Windows 11 PC running MATLAB with the Robotics System Toolbox for Universal Robots. Communication to the UR3e controller (running Polyscope X) is established over Ethernet using the RTDE protocol via TCP port 30004. Notably, only the robot requires a static IP address (e.g., 192.168.1.10); the host PC does not. The Robotiq gripper is controlled via the ToolComm Forwarder URCapX through RS485 over the robot's tool flange, with gripper commands issued as function calls embedded within a modified urserver.script file running on the Teach Pendant. This architecture requires no additional hardware controllers, Linux environment, or specialized middleware: only a standard Ethernet cable, a Windows PC, MATLAB, and the Robotics System Toolbox support package.

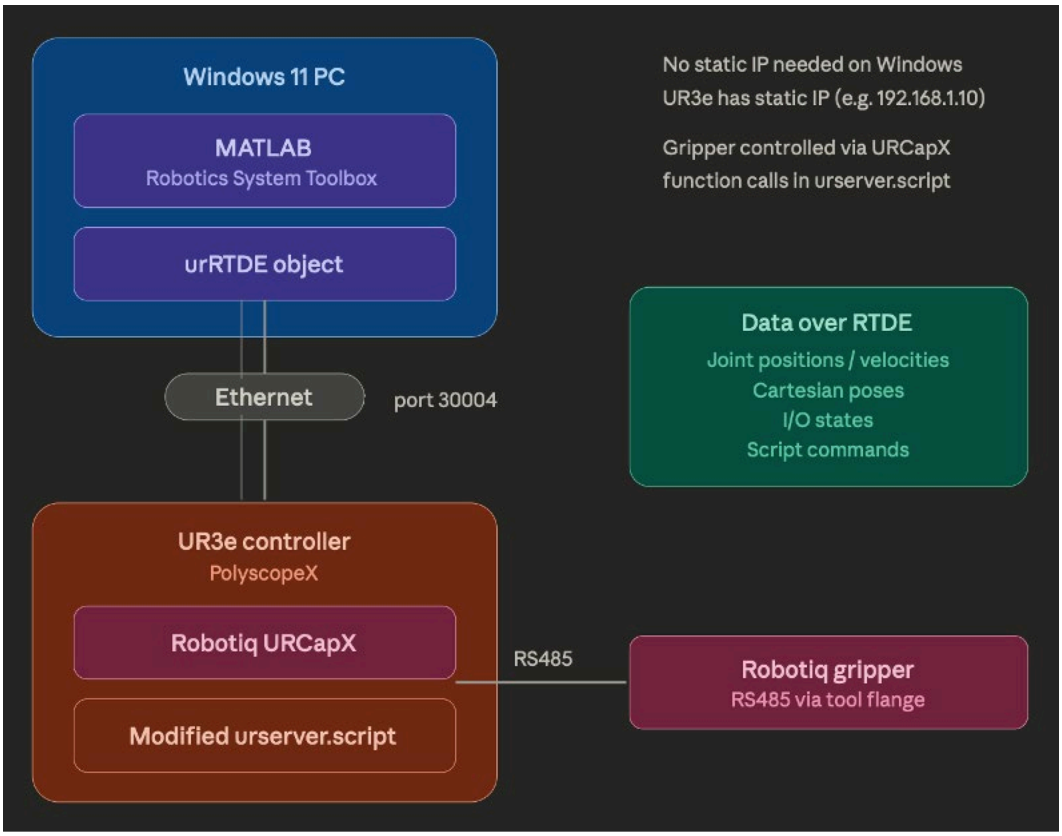

Figure 5: Finalized system architecture for the MATLAB-based UR robot control framework. The Windows 11 host PC communicates with the UR3e controller over Ethernet via RTDE (TCP port 30004). Only the UR3e requires a static IP (e.g., 192.168.1.10); the host PC does not. The Robotiq gripper is controlled via RS485 through URCapX embedded in a modified urserver.script. Data exchanged over RTDE includes joint positions/velocities, Cartesian poses, I/O states, and script commands.

This preliminary validation focused on stability and reproducibility of the MATLAB to UR robot connection via RTDE. Early testing indicates that the RTDE link is stable and that students can rapidly connect to the UR3e from a Windows MATLAB environment and begin development. This directly supports the initial goal of reducing setup complexity while enabling real robot interaction. Below (Fig. 6) is a MATLAB code snippet that demonstrates the simplicity of using RTDE; less than 10 lines of code are needed to start sending movement commands to the robot.

```
% Specify the Robot's IP Address
ROBOT_IP = '192.168.1.10';

% Define the urRTDEClient object
ur = urRTDEClient(ROBOT_IP, ...
    'CobotName', 'universalUR3e', ...
    'RTDEFrequency', 125, ...
    'ConnectionTimeOut', 10);

% Read the robot joint states
joints = ur.readJointConfiguration();

% Store current position
startJoints = joints;

% Move wrist +5 degrees
targetJoints = startJoints;
targetJoints(6) = targetJoints(6) + deg2rad(5);
fprintf('Moving wrist +5 deg...\n');
ur.sendJointConfigurationAndWait(targetJoints, 'EndTime', 2);
```

Figure 6: Snippet of MATLAB code demonstrating the ease of use of the MATLAB RTDE support package for UR e-series manipulators. In less than 10 lines of code the user can establish an RTDE connection with the robot, read joint states, and send movement commands.

Initial testing of the URStudio as a platform for testing FK derivations has proved successful. It is a browser-based emulator of the Polyscope X software, allowing users to quickly pick poses read joint angles. Fig. 7 is a screenshot of the URStudio 3D view, which provides the user with manual joint controls and the ability to read the angle values.

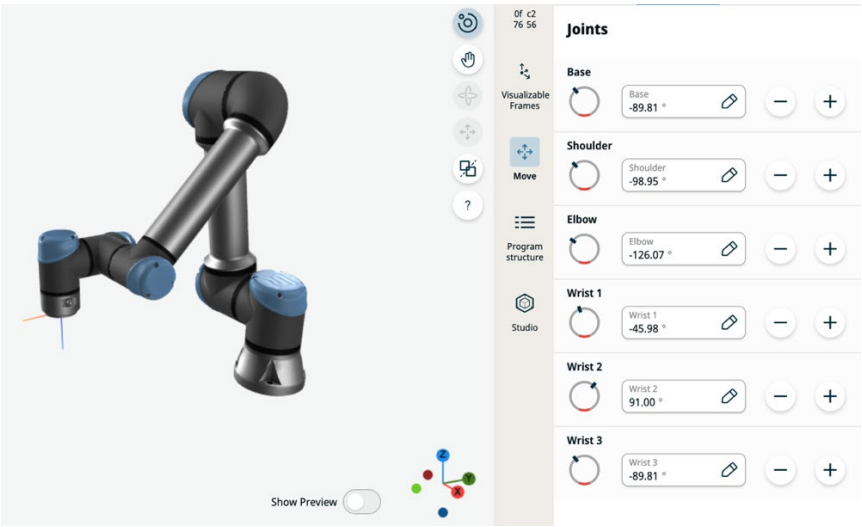

Figure 7: A screenshot demonstrating the 3D view from within URStudio [13]. This screen provides the user with manual joint controls as well as the ability to read the value for each joint angle. Students will use this to pick arbitrary poses to validate against their derived FK equations.

Discussion

The primary strength of the MATLAB-based workflow is that it utilizes skills that students will have already developed in previous coursework. Rather than forcing students to learn an entirely new ecosystem before they can meaningfully interact with a robot, this approach treats MATLAB as the main interface and uses RTDE as a clean bridge to the hardware. This preserves pedagogical value while removing configuration overhead that can derail early robotics experiences. Additionally, MATLAB's rich toolbox ecosystem allows students to immediately connect control theory to observable robot behavior. The simulation-to-hardware pathway enabled by URSim and URStudio further reinforces learning by allowing students to validate programs in a safe virtual environment before deploying to the physical robot [10].

Alternative approaches come with various limitations. Teach pendant programming obfuscates the underlying mechanics of robot control, and ROS-based frameworks require significant prerequisite knowledge and complex Linux configurations. The approach outlined in this paper offers accessibility comparable to TP programming while also exposing students to programmable, external control systems.

Limitations remain. A MATLAB-only approach may reduce early exposure to ROS (Robot Operating System) that students may encounter in industry and other research environments. There is also an inherent dependency on MathWorks licensing and toolbox availability, which may not be uniform across institutions, and the framework is currently validated only for UR e-series hardware.

Conclusion & Future Work

This paper presents a Windows-native, MATLAB-based laboratory infrastructure for controlling Universal Robots e-series manipulators using RTDE. A central contribution of this work is the iterative design process that converged on an optimal system architecture (Fig. 7) for undergraduate lab use. The framework is designed to lower the practical and cognitive barriers that often limit undergraduate access to industrial grade robotics, while preserving the conceptual visibility needed to link theory to hardware. By documenting the software stack selection process and offering scaffolded lab modules culminating in forward kinematics verification and a hardware pick-and-place demonstration, this work provides a replicable template for institutions seeking to modernize robotics instruction while minimizing barriers to entry.

Formal assessment of student outcomes is ongoing; future analysis of the feedback questionnaire (Appendix A), setup times, and lab milestone completion rates will evaluate the validity of this approach and its alignment with ABET Student Outcomes related to experimental design and data analysis [5, 8]. This data will also be used to inform future design decisions involving extending the lab curriculum to MATLAB+ROS-based control frameworks, with the goal of introducing students to industry standard tooling while maintaining the same approachable MATLAB-first architecture.

References

[1] R. 24/7 Staff, “Universal Robots Reports $85M in Revenue for Q1 2022,” Robotics 24/7, Apr. 28,2022. https://www.robotics247.com/article/universal_robots_reports_85_million_in_revenue_for_q1 (accessed Jan. 07, 2026).

[2] Oca, Siobhan Rigby, and Blake Hament. "Implementing and Using ROS in Undergraduate Robotics Curricula." 2024 ASEE Annual Conference & Exposition. 2024.

[3] “REP 2000 -- ROS 2 Releases and Target Platforms (ROS.org),” Ros.org, 2018. https://www.ros.org/reps/rep-2000.html (accessed Jan. 07, 2026).

[4] “Robotics System Toolbox Support for Universal Robots UR Series Manipulators,” Mathworks.com, 2022. https://www.mathworks.com/hardware-support/universal-robots.html (accessed Jan. 07, 2026).

[5] Rahman, SM Mizanoor. "Assessing and benchmarking learning outcomes of robotics-enabled stem education." Education Sciences 11.2 (2021): 84.

[6] “Setup for Connecting UR Series Manipulators over RTDE - MATLAB & Simulink,” Mathworks.com, 2025. https://www.mathworks.com/help/robotics/setup-for-rtde.html (accessed Jan. 07, 2026).

[7] J. M. Esposito, “Proposed Goals for Positively Transforming Robotics Education at Postsecondary Institutions,” IEEE Robotics & Automation Magazine, vol. 24, no. 3, pp. 156-164, Sept. 2017, doi: 10.1109/MRA.2016.2636375.

[8] J. M. Esposito, “ABET’s New Accreditation Criteria for Robotics and Mechatronics Engineering: A Summary,” IEEE Robotics & Automation Magazine, vol. 32, no. 3, pp. 16-18, Sept. 2025, doi: 10.1109/MRA.2025.3584823.

[9] N. Lotfi et al., “Promoting Open-source Hardware and Software Platforms in Mechatronics and Robotics Engineering Education,” in Proc. ASEE Annual Conference & Exposition, 2020, Paper ID #29624.

[10] Y. Niu, H. Qazi, and Y. Liang, “Building a Flexible Mobile Robotics Teaching Toolkit by Extending MATLAB/Simulink with ROS and Gazebo,” in Proc. 2021 7th International Conference on Mechatronics and Robotics Engineering (ICMRE), 2021, doi: 10.1109/ICMRE51691.2021.9384836.

[11] J. M. Phillips, J. Schwartz, S. Shue, and J. M. Conrad, “Enabling ROS for Low-Cost Educational Robotics Platforms,” in Proc. 2023 IEEE 20th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET), 2023, doi: 10.1109/HONET59747.2023.10374981.

[12] T. M. B. Santos et al., “Introducing Robotic Operating System as a Project-Based Learning in an Undergraduate Research Project,” in Proc. 2023 Latin American Robotics Symposium (LARS), 2023, doi: 10.1109/LARS/SBR/WRE59448.2023.10332969.

[13] Universal Robots, “UR Developer Suite,” Universal Robots A/S. Accessed Jan. 14, 2026. https://www.universal-robots.com/developer/.

[14] Universal Robots, “Real-Time Data Exchange (RTDE) Guide,” Universal Robots documentation. Accessed Jan. 14, 2026. https://docs.universal-robots.com/tutorials/communication-protocol-tutorials/rtde-guide.html.

[15] Vazquez-Hurtado, Carlos, et al. "Development of Control Engineering Curriculum for Advanced Research and Undergraduate Education: A Practical Approach to Bring Theory Closer to Practice Using Quanser® Products." 2025 ASEE Annual Conference & Exposition. 2025.

[16] Yu, Bo, and Matthew J. Jensen. "A multi-course project for mechatronics, system dynamics, and control experimentation courses." 2025 ASEE Annual Conference & Exposition. 2025.

[17] Universal Robots, "UR3e Series," Universal Robots UR3e Ultra-lightweight, compact cobot Downloads Technical Datasheet. Accessed Jan. 14, 2026. https://www.universal-robots.com/products/ur3e/.

Appendix A: Evaluation Questionnaire

Student Feedback Questionnaire

Common evaluation methods for robotics education curricula usually center around assessment forms and questionnaires. The goal is to determine whether students can successfully complete core lab milestones with less friction compared to alternative stacks, while still learning the intended robotics concepts. A questionnaire to evaluate these outcomes is presented below, with a scale from 1 to 5 representing strongly disagree to strongly agree, respectively.

Table 1: Learning Outcome Questionnaire

#	Question	Answers (1 – 5, strongly disagree to strongly agree)
1	Were you able to sufficiently complete the lab objectives in the allotted time?	
2	Was the initial setup (opening software/files, connecting to the hardware/simulator) simple and straightforward?	
3	Was using the teaching pendant to program the robot more difficult than using MATLAB?	
4	Did programming the robot in MATLAB versus using the teaching pendant provide deeper understanding of robot mechanics?	
5	Do you feel the topics in this lab were presented effectively and added value to your overall learning?	
6	How well do you feel you learned the basics and use cases of robot simulation using URStudio?	
7	How well do you feel you learned the basics of RTDE communication and hardware control?	

8	Do you feel you were lacking knowledge that should be covered as a prerequisite coming into this lab?	
9	Would you recommend this lab to others as an introduction to robotics?	