International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Interaction with Universal Robots

Lalit Yadav S M[1], Dr Pavitra G[2], Dr Swapnil SN[3], Rajarajan[4]

[1]Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bengaluru
[2] Internal Guide, Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bengaluru
[3] Internal Co Guide, Department of Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bengaluru
[4]Software Engineer Raad Systems Pvt. Ltd, Bengaluru

Abstract
This paper presents the design and development of a software system to control and monitor a Universal Robots UR10e collaborative robot. A user-friendly front-end was developed using Windows Presentation Foundation (WPF) while C# served as the back-end logic handler. The Model-View-ViewModel (MVVM) design pattern was employed to ensure a clean separation of concerns and scalability. The system leverages the UR10e's available client interfaces, notably the Dashboard Server and Primary/Secondary Interfaces, for remote operation and status monitoring. We discuss system architecture, implementation details, and the challenges encountered when integrating industrial robotic systems with modern desktop applications.

Keywords: Universal Robots UR10e, WPF, MVVM, C#, Communication Interface, Robot Control

1. Introduction

In recent years, the adoption of collaborative robots (cobots) such as the Universal Robots UR10e has transformed industries ranging from manufacturing to healthcare. These robots combine flexibility, safety, and ease of programming, making them highly attractive for automation tasks that require interaction with human workers. Despite their inherent flexibility, creating a fully customized, intuitive, and robust human-machine interface (HMI) for industrial robots often remains a technical challenge, particularly when native robot interfaces offer limited visual customization options.

The Universal Robots UR10e, part of the e-Series lineup, offers a range of network-based communication interfaces enabling external systems to remotely control and monitor the robot. Among these, the Dashboard Server and the Primary/Secondary Interfaces stand out as essential tools for programmatic robot management and data acquisition. The Dashboard Server (port 29999) provides command-driven control to manage robot operations such as loading, playing, pausing, stopping programs, and querying robot states. The Secondary Interface (port 30002) streams robot state information at a moderate update rate (10 Hz), making it particularly useful for applications that require periodic monitoring of the robot's condition without overwhelming network bandwidth.

In this project, a customized HMI was developed specifically for the UR10e robot using Windows Presentation Foundation (WPF) for the front-end and C# for the back-end logic. The architecture followed the Model-View-ViewModel (MVVM) design pattern, ensuring a clean separation between the user interface,

application logic, and data access layers.The primary communication channels used to interact with the UR10e were the Dashboard Server for executing robot control commands and managing protective stops, and the Secondary Interface for receiving robot status updates at a reliable and manageable rate.

By leveraging these interfaces, the developed system allows users to control the robot remotely, monitor its operational state in real-time, and handle error recovery without direct interaction with the teach pendant. The use of WPF provides a modern, responsive, and highly customizable graphical interface, while the underlying MVVM structure ensures that the system remains maintainable and scalable as additional features are integrated.

This paper details the design methodology, communication protocols, and software architecture used to realize this integration. Furthermore, it discusses practical challenges encountered during development and integration, and outlines best practices for building similar systems in future industrial automation projects

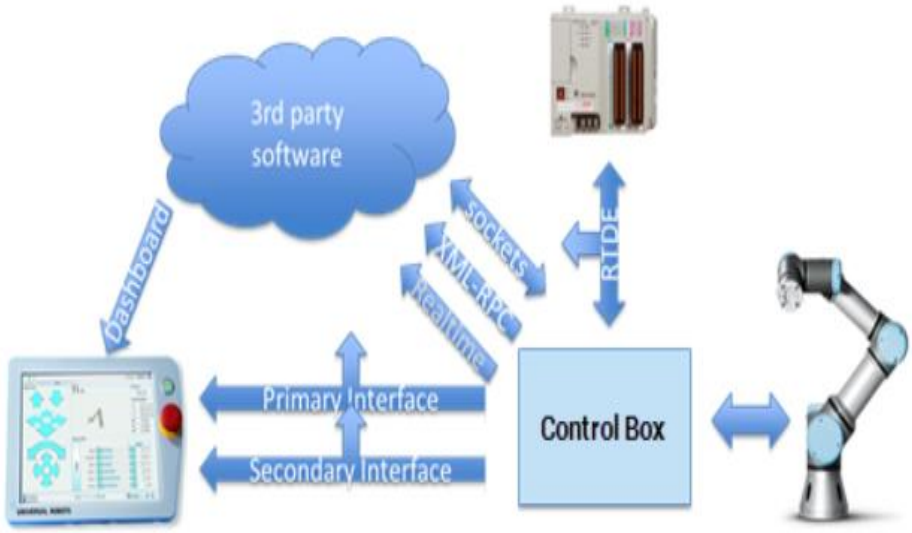
2. Universal Robots UR10e Client Interfaces Overview

The UR10e controller provides multiple communication interfaces for external device integration:

- Dashboard Server (Port 29999):Allows program control (load, play, stop, pause), robot state inquiries (robotmode, safetystatus), and protective stop management. Commands are plain-text based over TCP sockets.
- Primary/Secondary Interfaces (Ports 30001/30002):Transmit robot state data and accept URScript commands at a frequency of 10 Hz.
- Real-Time Data Exchange (RTDE) (Port 30004):Facilitates structured and customized data exchange with higher reliability.
- Socket Communication and XML-RPC:Allow the robot to act as a TCP client to communicate with external servers for specific actions and computations.

For this project, primary interaction was performed via the Dashboard Server, supplemented with monitoring via the Primary Interface.

UR robot can interact with external devices by different types of communication interfaces.

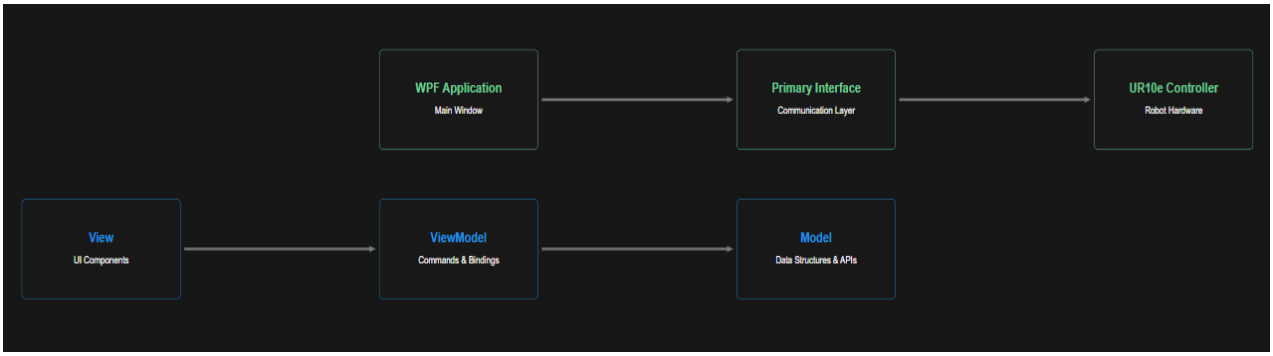


3. System Architecture

The software was divided into three main layers based on the MVVM model:

Layer	Description
Model	Encapsulates robot communication logic (e.g., TCP/IP socket handling, command serialization)
View-Model	Mediates between the Model and the View, exposes commands and observable properties for UI binding
View	WPF XAML interface presenting controls for robot operation (buttons, indicators, status feedback)

Figure 1: System Overview



Hardware Setup

The system comprises a Universal Robots UR10e robotic arm connected to a host PC via Ethernet. The robot's controller communicates with the PC using the primary client interface, which supports real-time data streaming and URScript command execution.

Software Components

Front-End: A WPF application developed in C# serves as the graphical user interface (GUI). The MVVM pattern is employed to separate the UI (View), business logic (ViewModel), and data (Model), promoting testability and maintainability.

Back-End: The application communicates with the robot's controller using the primary client interface library, which facilitates sending URScript commands and receiving robot state data.

4. Implementation Details

4.1. Frontend (WPF)

- Developed interactive user controls including buttons for Power On, Brake Release, Load Program, Play, Pause, and Stop.
- Live robot status indicators (Program State, Robot Mode, Safety Status) using WPF (DataBinding) and (INotifyPropertyChanged).
- Asynchronous feedback updates using (DispatcherTimer) for UI thread safety.

4.2. Backend (C# Communication Library)

- Established TCP socket connections to UR10e on port 29999.
- Sent string-based commands and parsed textual responses.

- Implemented auto-reconnect and error handling strategies to ensure resilience against network fluctuations.

Communication Protocol

The communication between the WPF application and the UR10e robot is established over TCP/IP using the primary client interface. This interface allows the application to:

- Send URScript commands to the robot's controller.
- Receive real-time robot state data at a 10 Hz update rate.
- Monitor and control robot operations remotely without the need for a physical teach pendant.
  The primary interface transmits robot state data and additional messages, making it suitable for communication between the GUI and the controller.

4.3. MVVM Pattern Utilization

- ViewModel classes encapsulated all command logic and property updating, ensuring the View had no code-behind.
- Used RelayCommand to implement UI event handling.
- Models abstracted the socket logic completely, enabling ViewModels to remain purely logical.

5. Challenges and Solutions

Challenge	Solution
Robot controller refused commands during manual mode	Added mode check before sending sensitive commands
Sockets hanging if robot rebooted	Implemented watchdog timers and reconnect logic
UI freezing during network communication	Employed async-await and (Task Run) for socket communication
	Incorporated timing logic for retrying unlock commands after 5 seconds when necessary

6. Results and Discussion

The developed system successfully allowed real-time program control and robot status monitoring. Key observations:

- The robot response time via the Dashboard Server was typically < 100ms under a local network.
- MVVM architecture significantly simplified updates when adding new UI features like jogging or error logs.
- Asynchronous programming was crucial for maintaining a responsive and safe user interface.

7. Conclusion

This project demonstrates that industrial robotic systems like the Universal Robots UR10e can be efficiently integrated with modern desktop software using WPF and C#. The clean separation of concerns offered by MVVM ensures long-term maintainability and scalability, especially valuable in smart manufacturing environments where robots operate alongside dynamic systems. Future work includes

deeper RTDE integration for custom sensor feedback and enhancing security through authentication layers.

References

1. Universal Robots, "Dashboard Server e-Series Documentation," [Online]. Available: https://www.universal-robots.com/articles/ur/dashboard-server-e-series-port-29999/
2. Universal Robots, "Real-Time Data Exchange (RTDE) Guide," [Online].
3. Josh Smith, "WPF Apps With The Model-View-ViewModel Design Pattern," MSDN Magazine.
4. G. Ganesh, "Socket Programming in C#," [Online].
5. Dietrich, A., Ott, C., & Albu-Schaffer, A., "Efficient Human-Robot Interaction: Communication and Control in Collaborative Robots," *IEEE Transactions on Robotics*, 2017.
6. Microsoft Docs, "Data Binding Overview," Available: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/data/?view=netdesktop-6.0
7. J. Y. S. Luh, M. W. Walker, and R. P. Paul, "Resolved Acceleration Control of Mechanical Manipulators," *IEEE Transactions on Automatic Control*, vol. 25, no. 3, pp. 468–474, 1980.
8. KUKA Robotics, "Collaborative Robot Technology and Applications," Available: https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa
9. M. Quigley et al., "ROS: An Open-Source Robot Operating System," *ICRA Workshop on Open Source Software*, 2009.
10. M. Valente, P. Ribeiro, and F. T. Ferreira, "Design Patterns for MVC and MVVM in Mobile Applications," *Procedia Computer Science*, vol. 164, 2019, pp. 381-388.