Smart Home Simulation in CoppeliaSim Using C# Through WebSocket

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Vice Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Subject Area: Computer Technology.
Type of the Paper: Simulation-based Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.8075717
Google Scholar Citation: IJAEML

How to Cite this Paper:
Chakraborty, S., & Aithal, P. S. (2023). Smart Home Simulation in CoppeliaSim Using C# Through WebSocket. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *7*(2), 134-143. DOI: https://doi.org/10.5281/zenodo.8075717

International Journal of Applied Engineering and Management Letters (IJAEML)
A Refereed International Journal of Srinivas University, India.

Crossref DOI: https://doi.org/10.47992/IJAEML.2581.7000.0178

Received on: 04/06/2023
Published on: 24/06/2023

Smart Home Simulation in CoppeliaSim Using C# Through WebSocket

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information Sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Vice Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *The increasing integration of intelligent devices and automation technologies in our daily lives has led to the development of smart homes, where various devices and systems are interconnected to provide enhanced comfort, convenience, and energy efficiency. Simulation tools are vital in designing, testing, and validating innovative home systems before their physical implementation. This research paper presents an intelligent home simulation framework implemented in CoppeliaSim. The simulation framework utilizes the C# programming language and leverages WebSocket communication to establish real-time interactions between the simulation environment and external applications. It discusses the architecture and implementation details of the simulation framework, highlighting the integration of CoppeliaSim, C#, and WebSocket.*

Design/Methodology/Approach: *CoppeliaSim was chosen as the simulation platform due to its robust features, flexibility, and support for robotics and automation simulations. The simulation framework was developed using the C# programming language. C# provides a rich set of libraries and tools for efficient development, and its integration with CoppeliaSim allowed seamless communication and control of the simulated intelligent home environment. WebSocket communication established real-time interactions between the simulation environment and external applications.*

Findings/Result: *The simulation framework successfully created a realistic, intelligent home environment, accurately representing various components and systems in typical smart homes. It effectively replicated real-world behaviors and interactions of intelligent home devices. The innovative home simulation framework highlighted its ability to accurately simulate an intelligent home environment, facilitate real-time interactions, and evaluate system performance.*

Originality/Value: *The integration of CoppeliaSim, C#, and WebSocket communication in developing an intelligent home simulation framework presents a novel approach. While CoppeliaSim is a widely used simulation platform, combining C# and WebSocket communication provides an innovative and efficient way to create a brilliant home environment and enable real-time interactions with external applications.*

Paper Type: *Simulation-based Research.*

Keywords: Smart home simulation, automation simulation, CoppeliaSim model creation. Home automation in CoppeliaSim.

1. INTRODUCTION :

The rapid proliferation of intelligent devices and the Internet of Things (IoT) has paved the way for the emergence of smart homes, transforming traditional living spaces into intelligent and interconnected environments. Smart homes offer enhanced comfort, convenience, energy efficiency, and security by seamlessly integrating various devices, systems, and technologies. However, designing, testing, and validating intelligent home systems can be complex and costly, requiring careful consideration of device interactions, user behaviour, and environmental conditions. Simulation tools provide an invaluable means of addressing these challenges by offering a virtual platform for

modeling and evaluating innovative home systems before their physical implementation. Among the available simulation platforms, CoppeliaSim has gained prominence due to its extensive features, support for robotics and automation simulations, and flexible application programming interface (API). Leveraging the capabilities of CoppeliaSim, this research paper presents an intelligent home simulation framework developed using the C# programming language and WebSocket communication. The primary objective of this research is to create a realistic, intellectual home environment within CoppeliaSim, enabling the evaluation and analysis of different innovative home configurations, algorithms, and scenarios. C# and WebSocket communication integration provide real-time interactions between the simulation environment and external applications, enhancing responsiveness and enabling seamless integration with monitoring or control systems. Through this simulation framework, researchers, developers, and practitioners can assess system performance, energy consumption, and user experience, facilitating efficient optimization and decision-making processes in innovative home development.

This paper proceeds as follows: Section 2 provides an overview of related work in intelligent home simulation and highlights this research's unique contributions. Section 3 demonstrates the objective of the research work. Section 4 presents the approach and methodology of the simulated intelligent home system. Section 5 discusses the experiment of the research work. In Section 6, we added Recommendations, and finally, Section 7 summarizes the key contributions and suggests future research directions. Through this research, we aim to advance the field of intelligent home development by providing a comprehensive simulation framework that accurately replicates real-world behaviours and interactions. The integration of CoppeliaSim, C#, and WebSocket communication offers a valuable tool for designing and optimizing intelligent home systems, ultimately contributing to the realization of future efficient, sustainable, and user-centric smart homes.

2. RELATED WORKS :

Szlęg et al. [1] proposed a simulation environment in Unity3D specifically designed for testing and training underwater vehicles. The study focuses on developing realistic underwater scenarios and integrating physics-based simulations. Chen et al. [2] surveyed to evaluate open-source simulation platforms suitable for multi-copter UAV swarm simulations. Cavalcanti et al. [3] present the RoboStar technology, which offers a comprehensive toolbox for combined proof, simulation, and testing in robotics. Buerkle et al. [4] propose an EEG-based approach for recognizing arm movement intentions in symbiotic human-robot collaboration. Faria et al. [5] discuss path generation, control, and monitoring techniques in additive manufacturing hybrid processes for composites. Márquez [6] investigates the socioeconomic impact of robotics on unemployment in the industry, with a specific focus on Mexico. Verma [7] discusses integrating robotic applications using the Internet of Robotics Things (IoT) for advanced research. Zhu et al. [8] present the development of a low-cost data glove using flex sensors for the teleoperation of robot hands. Zhu et al. [9] show the result of a low-cost data glove using flex sensors for the teleoperation of robot hands. The study focuses on providing an affordable and intuitive interface for controlling robotic manipulators. Audonnet et al. [10] systematically compare simulation software for robotic arm manipulation using ROS2. The study evaluates various simulation frameworks [10].

3. OBJECTIVES :

The research aims to guide those willing to create home automation inside the virtual environment through the simulator. Sometimes researchers are unable to experiment with real scenarios. Because to experiment with the actual system, we need several instruments and devices, which is so costly. Not only that, only just for experiments need to purchase. And also need expert people to fit the electrical device in the home, which incurs costs and is sometimes unavailable to the researcher. The novice researcher is in danger of doing itself. So the better option is to get a simulated environment so that no hazards come in the way. It will replicate the same as a natural environment. So this research work help researcher how we can set up a virtual environment to test their intelligent home algorithm and research work. This does not require hardware or equipment; the working pc/laptop can execute the job. As an example, We added here a couple of typical loads. More loads can be connected in the same way.

4. APPROACH AND METHODOLOGY :

We install the CoppeliaSim simulator in our working system. We create a couple of models unavailable inside the CoppeliaSim IDE. We added some loads from IDE itself, like light and door, which are available in the simulator. Then we add nonthreaded Lua scripts. Inside the Lua scripts, we first added a websocket server into the code. This server receives commands from the C# application running the C# websocket client. Then we assign a load object id for every available piece of equipment. Now our scripts id ready to receive commands from the client. We install visual studio 2022 community edition. Using C# language, we create a graphical user interface (GUI) to send the command to the simulator. Before running the simulator, we need to run the server. Run the CoppeliaSim. After that, run the C# application. Click connect button. The client will connect with the server using the 8088 port. Once the connection is established, we can send a command to the server by clicking the specific button. The server receives the command, parses the command string, and triggers the typically connected load.

5. EXPERIMENT :

To strengthen our acquired knowledge, we can do some practical experiments. To do that need to follow the below procedure.

1) Download CoppeliaSim from https://www.coppeliarobotics.com/downloads and install.
2) Create a model in CoppeliSim. The documents are readily available over the net.
3) Some valuable papers regarding coppeliasim are available at: https://scholar.google.com/citations?hl=en&user=QzNd9y0AAAAJ&view_op=list_works&gmla=AHoSzlW7fmECMQwyPU367k8X8EiaTkk7hbIFNrrKV4jQ7xXscccan6Q3AQBuET4v1CPtsMLv-1-Rbb0pN7xIkL3i
4) Project download from Github Link: https://github.com/sudipchakraborty/Smart-Home-Simulation-In-CoppeliaSim-Using-Csharp-Through-WebSocket.git
5) under the CoppeliaSim folder. Open the “Home_Automation” file. It will look like Figure 1.

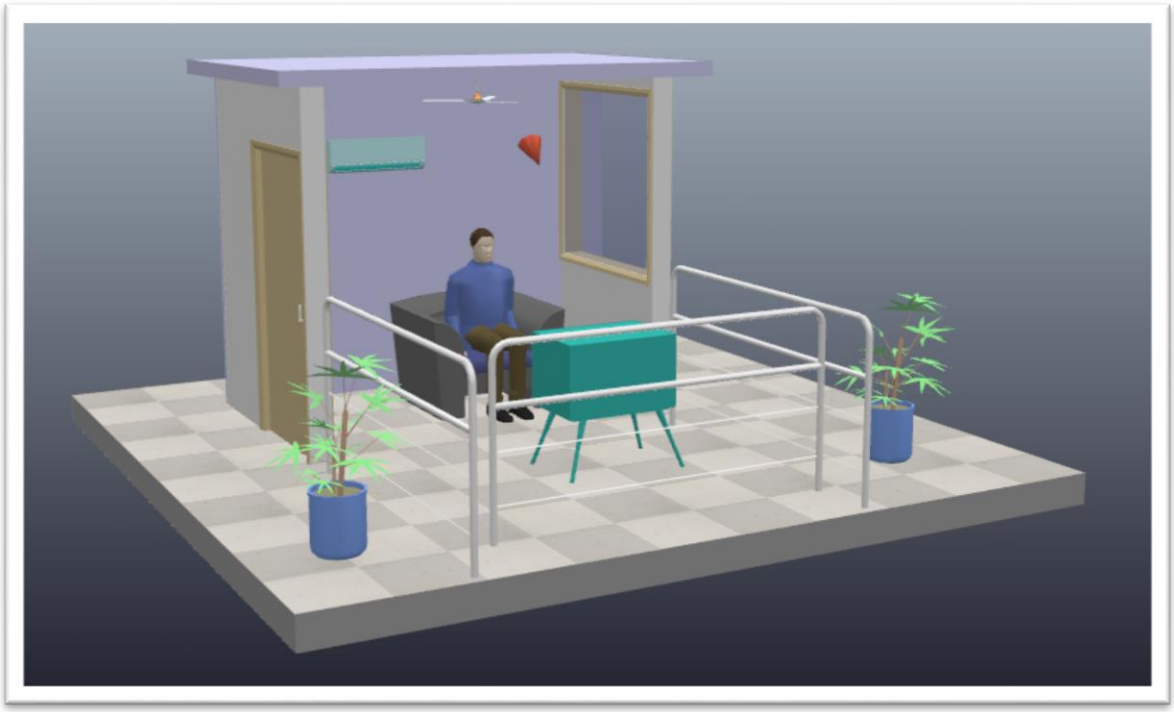

Fig. 1: CoppeliaSim Scene file [Source: Authors]

6) Download the visual studio community edition from: https://visualstudio.microsoft.com/downloads/

7) From the download project folder under the c# application, open the "HomeAutomation_Simulation.sln" file.
8) Try to compile the project. It might be necessary to install some packages using the Nuget package manager.
9) Now build and run. The interface looks like the figure depicted in Figure 2.

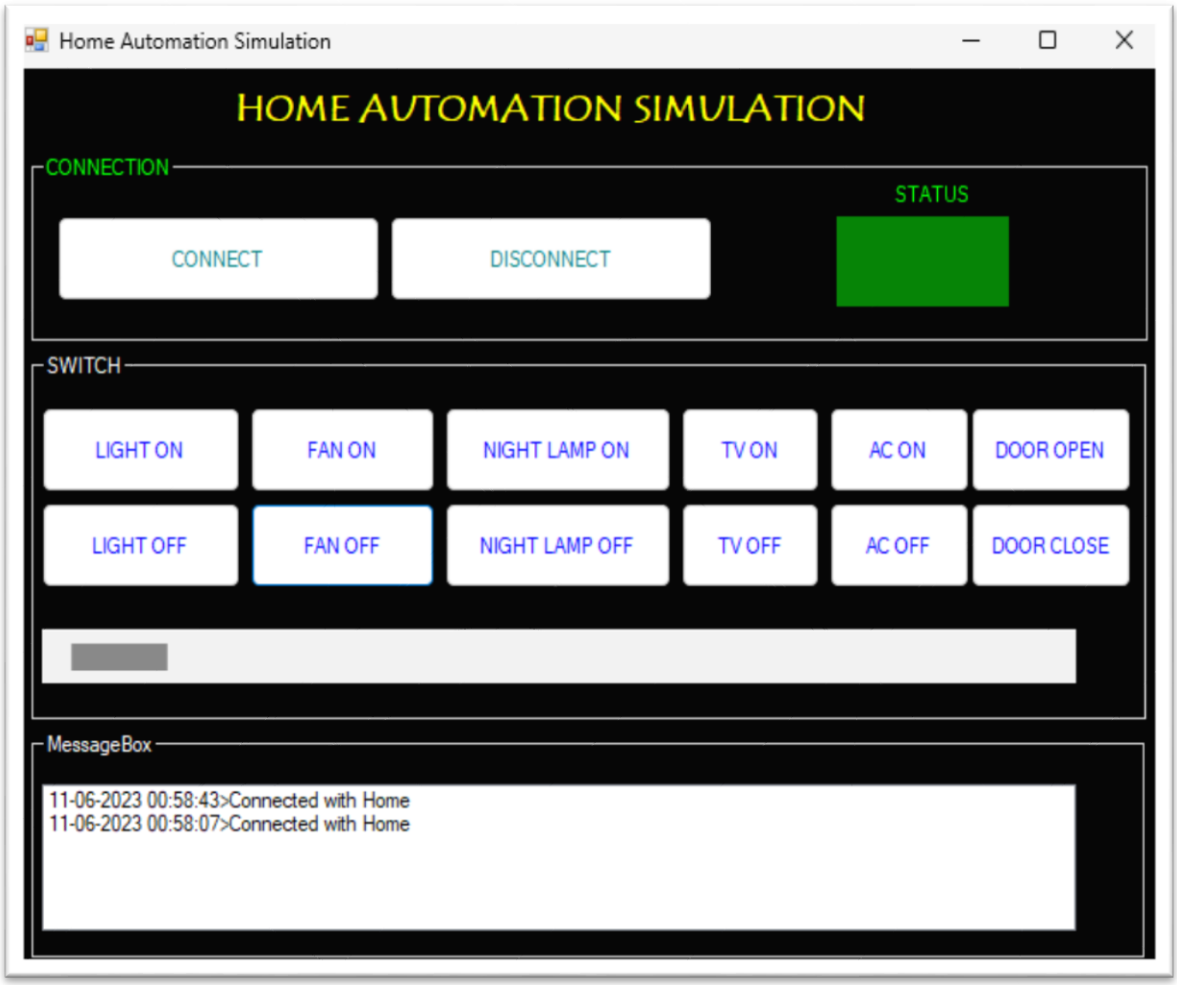

Fig. 2: C# Graphical User Interface [Source: Authors]

10) Figure 3 depicts the Lua scripts which are responsible for handling the equipment. We created each module for each piece of equipment to make the code easy to understand. Lua starts in nonthreaded code. It forms a web socket server with port 8088. It waits for a request to trigger the load from the WebSocket client. When a request receives, it parses the command, matches the code block, and starts the load.
11) Figures 4 and 5 depict the C# code to send the command to the server. When the application starts, it starts a WebSocket client with port number 8088.
12) First, Run the CoppeliaSim scene file, i.e., the web socket server.
13) Now run the c# application. Click on connect button. It will connect with the CoppeliaSim WebSocket server. If the connection is established, it can send any command to the server.
14) A couple of buttons are added to the GUI. We send the string for the particular load when we press the button.
15) Receiving the string, the server parses the command and triggers the load.

```
Child script "/Cuboid[0]"

local ws=require "Websocket_Server"
local ac=require "AC_Split"
local fan=require "FAN_Ceiling"
local lt=require "Light_Tube"
local nl=require "Light_Night"
door=sim.getObjectHandle("./../doorJoint")
-----------------------------------------------------------------------------------------
function sysCall_init()
        lt.init("tube_light")
        nl.init("night_lamp")
        fan.init("bearing")
        ac.init("swing_motor")
        ws.Init(8088)
end
-----------------------------------------------------------------------------------------
function sysCall_actuation()
   if(ws.received==true) then
        ws.received=false
        print(ws.cmd)
        ws.send("OK")
        ----------------------------------------
        if(ws.cmd=="Light on")          then lt.on()        end
        if(ws.cmd=="Light off")         then lt.off()       end
        ----------------------------------------
        if(ws.cmd=="Fan on")            then   fan.on()     end
        if(ws.cmd=="Fan off")           then  fan.off()     end
        ----------------------------------------
        if(ws.cmd=="Night Lamp on")     then    nl.on()     end
        if(ws.cmd=="Night Lamp off")    then   nl.off()     end
        ----------------------------------------
        if(ws.cmd=="TV on")             then                end
        if(ws.cmd=="TV off")            then                end
        ----------------------------------------
        if(ws.cmd=="AC on")             then    ac.on()     end
        if(ws.cmd=="AC off")            then   ac.off()     end
        ----------------------------------------
         if(ws.cmd=="Door Open")        then   sim.setJointTargetPosition(door,0)       end
        if(ws.cmd=="Door Closed")       then  sim.setJointTargetPosition(door,0.75)     end
        ----------------------------------------
   end
        ac.process()
        fan.process()
end
-----------------------------------------------------------------------------------------
function sysCall_sensing()
end
-----------------------------------------------------------------------------------------
function sysCall_cleanup()
end
-----------------------------------------------------------------------------------------
```

Fig. 3: Lua script to drive equipment [Source: Authors]

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using Tools;

namespace HomeAutomation_Simulation
{
    public partial class frm_main : Form
    {
        Websocket_Client client;
        MSG msg;

        public frm_main()
        {
            InitializeComponent();
            client=new Websocket_Client();
            msg=new MSG(lst: lst_msg);
        }

        private void btn_connect_with_home_Click(object sender, EventArgs e)
        {
            if(client.open())
            {
                msg.push(str: "Connected with Home");
            }
            else
            {
                msg.push(str: "Unable to connect with Home");
            }
        }

        private void frm_main_Load(object sender, EventArgs e)
        {

        }
```

Fig. 4: C# code to trigger the equipment [Source: Authors]

```
        private void btn_light_on_Click(object sender, EventArgs e)
        {
            client.send(message: "Light on");
        }

        private void btn_light_off_Click(object sender, EventArgs e)
        {
            client.send(message: "Light off");
        }

        private void btn_disconnect_Click(object sender, EventArgs e)
        {
             client.close();
        }

        private void btn_fan_on_Click(object sender, EventArgs e)
        {
            client.send(message: "Fan on");
        }

        private void btn_fan_off_Click(object sender, EventArgs e)
        {
            client.send(message: "Fan off");
        }

        private void btn_night_lamp_on_Click(object sender, EventArgs e)
        {
            client.send(message: "Night Lamp on");
        }

        private void btn_night_lamp_off_Click(object sender, EventArgs e)
        {
            client.send(message: "Night Lamp off");
        }

        private void btn_tv_on_Click(object sender, EventArgs e)
        {
            client.send(message: "TV on");
        }

        private void btn_tv_off_Click(object sender, EventArgs e)
        {
            client.send(message: "TV off");
        }

        private void btn_ac_on_Click(object sender, EventArgs e)
        {
            client.send(message: "AC on");
        }

        private void btn_ac_off_Click(object sender, EventArgs e)
        {
            client.send(message: "AC off");
        }

        private void btn_door_open_Click(object sender, EventArgs e)
        {
            client.send(message: "Door Open");
        }

        private void btn_door_close_Click(object sender, EventArgs e)
        {
            client.send(message: "Door Closed");
        }
    }
}
```

Fig. 5: C# code to trigger the equipment (continued) [Source: Authors]

6. RECOMMENDATIONS :

- The fan object adapted from https://free3d.com/

- Different fan models can get from https://free3d.com/3d-models/fan
- The complete project is for the research purpose only. To show the procedure to communicate and how to enable the virtual load
- The researchers are requested to customize for their purpose.
- There is much scope for improvement in the model as well code.
- The code is not made for professional grades due to keep it simple as possible to understand.
- An error handle need to add for production.

7. CONCLUSION :

This research paper has presented a comprehensive intelligent home simulation framework developed in CoppeliaSim using C# and WebSocket communication. Integrating C# and WebSocket communication enables real-time interactions between the simulation environment and external applications. It contributes to the advancement of intelligent home development by providing a comprehensive simulation framework that facilitates efficient evaluation, optimization, and decision-making processes. Integrating CoppeliaSim, C#, and WebSocket communication offers researchers, developers, and practitioners a powerful tool for designing, testing, and optimizing intelligent home systems, ultimately leading to enhanced comfort, convenience, energy efficiency, and user satisfaction in real-world smart home deployments.

REFERENCES :

[1] Szlęg, P., Barczyk, P., Maruszczak, B., Zieliński, S., & Szymańska, E. (2023, January). Simulation Environment for Underwater Vehicles Testing and Training in Unity3D. *In Intelligent Autonomous Systems 17: Proceedings of the 17th International Conference IAS-17* (pp. 844-853). Cham: Springer Nature Switzerland. Google Scholar↗

[2] Chen, Z., Yan, J., Ma, B., Shi, K., Yu, Q., & Yuan, W. (2023). A Survey on Open-Source Simulation Platforms for Multi-Copter UAV Swarms. *Robotics*, *12*(2), 53. Google Scholar↗

[3] Cavalcanti, A., Barnett, W., Baxter, J., Carvalho, G., Filho, M. C., Miyazawa, A., ... & Sampaio, A. (2021). RoboStar Technology: A roboticist's toolbox for combined proof, simulation, and testing. *Software Engineering for Robotics*, 249-293. Google Scholar↗

[4] Buerkle, A., Eaton, W., Lohse, N., Bamber, T., & Ferreira, P. (2021). EEG-based arm movement intention recognition towards enhanced safety in symbiotic Human-Robot Collaboration. *Robotics and Computer-Integrated Manufacturing*, *70*(1), 102137. Google Scholar↗

[5] Faria, C., Martins, D., Matos, M. A., Pinho, D., Ramos, B., Bicho, E., ... & Vaz, A. I. F. (2020). Path generation, control, and monitoring. *Additive Manufacturing Hybrid Processes for Composites Systems*, 203-236. Google Scholar↗

[6] Márquez, B. Y. (2022). Unemployment in the Industry with the Arrival of Robotics in Mexico. In Algorithms and Computational Techniques Applied to Industry (pp. 145-162). *Cham: Springer International Publishing.* Google Scholar↗

[7] Verma, H. M. (2021). Internet of Robotics Things (IoT) Based Integration of Robotic Applications for Advanced Research. *Wasit Journal of Computer and Mathematics Sciences*, 9-16. Google Scholar↗

[8] Tuluc, C., Verberne, F., Lasota, S., de Almeida, T., Malheiro, B., Justo, J., ... & Guedes, P. (2021). The MopBot Cleaning Robot–An EPS@ ISEP 2020 Project. In *Educating Engineers for Future Industrial Revolutions: Proceedings of the 23rd International Conference on Interactive Collaborative Learning (ICL2020), Volume 1 23* (pp. 79-90). Springer International Publishing. Google Scholar↗

[9] Zhu, S., Stuttaford-Fowler, A., Fahmy, A., Li, C., & Sienz, J. (2021, September). Development of a low-cost data glove using flex sensors for the robot hand teleoperation. *In 2021 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT)* (pp. 47-51). IEEE. Google Scholar↗

[10] Audonnet, F. P., Hamilton, A., & Aragon-Camarasa, G. (2022, November). A Systematic Comparison of Simulation Software for Robotic Arm Manipulation using ROS2. *In 2022 22nd International Conference on Control, Automation and Systems (ICCAS)* (pp. 755-762). IEEE. Google Scholar↗

****