CSSR for

Culturally Sensitive Social Robotics
for Africa

D3.3 Software Installation Manual

Due date: **1/10/2023**
Submission Date: **07/09/2023**

Start date of project: **01/07/2023**

Duration: **36 months**

Lead organisation for this deliverable: **Carnegie Mellon University Africa**

Responsible Person: **CSSR4Africa Team**

Revision: **1.3**

Contents

Executive Summary

Deliverable D3.3 serves as a comprehensive installation manual for the Culturally Sensitive Social Robotics for Africa (CSSR4Africa) project. The manual will be continually updated as a living document to ensure it reflects the current capabilities of the system. The objective of this task is to document the process of installing and executing the required software components to instantiate the CSSR4Africa system. It provides step-by-step instructions for setting up the development environment and controlling the Pepper robot in both physical and simulated environments.

1   Introduction

This Deliverable, D3.3: Software Installation Manual documents the process for the installation and execution of the software required to instantiate all or part of the CSSR4Africa system and run the unit, integration, and system tests. This installation manual assumes the system has a native Ubuntu 18.04 operating system. It was tested on a computer device with a storage of 512GB, 16GB of RAM, and 4 Intel CPU cores. In addition to that it was tested on a dedicated server (see Appendix A).

The installation is divided into 2 main sections. Section 2, is for setting up the development environment for the physical robot and the Gazebo simulator. This section first goes through a prerequisite setup that is needed for both the physical robot and Gazebo simulator environments and proceeds with corresponding installation steps. Section 3, will discuss installing and running of the CSSR4Africa software.

2   Setting up the Development Environment

This section provides step-by-step instructions for installing a set of software packages required to control the physical Pepper robot and the robot in the Gazebo simulator. Open a new terminal *(ctrl + shift + t)* and type the following commands carefully.

2.1   Prerequisite

The prerequisite for setting up the development environment for both the physical robot and the simulator is installing ROS Melodic. The following instructions are followed to install ROS melodic.

1. Setup the computer to accept software from packages.ros.org.

   ```
   sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu \
   $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
   ```

2. Install curl.

   ```
   sudo apt install curl
   ```

3. Setup your keys.

   ```
   curl -s https://raw.githubusercontent.com/ros/rosdistro/\
   master/ros.asc | sudo apt-key add -
   ```

4. Update Debian package index.

   ```
   sudo apt update
   ```

5. Install ROS Melodic with the default configurations.

   ```
   sudo apt install ros-melodic-desktop-full
   ```

6. Make ROS environment variables automatically added every time a new shell is launched.

   ```
   1 echo "source /opt/ros/melodic/setup.bash" >> $HOME/.bashrc
   2 source $HOME/.bashrc
   ```

7. Install Git

   ```
   sudo apt install git
   ```

2.2   Setting up the Development Environment for the Physical Robot

2.2.1   Installing and Configuring the C++ NAOqi SDK

NAOqi serves as the main software responsible for running and controlling the Pepper robot. To control Pepper with ROS, NAOqi needs to be installed, and a suitable ROS driver must be found to enable communication between the two software platforms. The NAOqi Framework, being a cross-language programming framework utilized for programming Pepper, allows programming the robot using both C++ and Python languages. For the development of the bridge, the NAOqi C++ SDK needs to be installed, and the following steps detail the installation process.

1. Open a new terminal and check the compiler version.
   A GCC compiler version 4.8.2 or higher is required. On most Ubuntu distributions, it is installed by default. Check the GCC version by typing the following command:

   ```
   gcc -v
   ```

2. Install pip.
   If you don't have pip installed on your machine, please run the code below.

   ```
   sudo apt install python-pip
   ```

3. Install qbuild.
   qbuild is used to create a cross-platform executable.

   ```
   pip install qibuild --user
   ```

4. Update the path environment variable.

   ```
   # to load qBuild when your shell starts
   PATH=$PATH:$HOME/.local/bin
   ```

5. Configure qBuild.

   ```
   qibuild config --wizard
   # Input 1 and hit enter (1 Unix Makefiles (default))
   # Input 1 and enter (1 None (default))
   ```

6. Download the NAOqi-SDK.

   ```
   wget -P $HOME/Downloads/ https://community-static.aldebaran.com/\
   resources/2.5.10/NAOqi%20SDK/naoqi-sdk-2.5.7.1-linux64.tar.gz
   ```

7. Go to the download folder and extract the downloaded file.

   ```
   1 cd $HOME/Downloads
   2 tar -xvzf naoqi-sdk-2.5.7.1-linux64.tar.gz
   ```

8. Create a toolchain.

   ```
   qitoolchain create mytoolchain \
   $HOME/Downloads/naoqi-sdk-2.5.7.1-linux64/toolchain.xml
   ```

9. Configure the SDK.

   ```
   1 qibuild init
   2 qibuild add-config myconfig -t mytoolchain --default
   ```

2.2.2 Installing NAOqi Driver and ROS Packages

The NAOqi driver is a module that provides some bridge capabilities. It publishes sensory data, and the robot position (Pepper in our case) and enables ROS to call part of the NAOqi API. The easiest way is to install it with the apt package tool used to install software on Ubuntu. Installing the driver through the apt will not provide all the packages needed to communicate with the robot through ROS. For example, the **naoqi_dcm_driver**, is the package controlling the robot's actuators. Thus, the remaining packages were compiled from their sources through catkin make. This requires cloning the official sources code from GitHub. Below are the steps to follow.

```
# Install the NAOqi driver
sudo apt-get install ros-.*-naoqi-driver

# Create ROS workspace
mkdir -p $HOME/workspace/pepper_rob_ws/src

# Move to workspace directory
cd $HOME/workspace/pepper_rob_ws/src

# Clone NAOqi DCM driver repository
git clone https://github.com/ros-naoqi/naoqi_dcm_driver.git

# Clone Pepper DCM driver repository
git clone https://github.com/ros-naoqi/pepper_dcm_robot.git

# Clone Pepper virtual repository
git clone https://github.com/ros-naoqi/pepper_virtual.git

# Clone NAOqi driver repository
git clone https://github.com/ros-naoqi/naoqi_driver.git

# Clone Pepper robot repository
git clone https://github.com/ros-naoqi/pepper_robot.git

# Clone Pepper moveit config repository
git clone https://github.com/ros-naoqi/pepper_moveit_config.git

# Build the repository
cd .. && catkin_make
# Add the workspace to your ROS environment by sourcing the setup file in
    devel folder
source devel/setup.bash

# Add the setup to your .bashrc file so that you don't have to do this
    every time you open a new terminal
echo "source $HOME/workspace/pepper_rob_ws/devel/setup.bash" >> \
$HOME/.bashrc

# Install additional packages
sudo apt-get install ros-melodic-joint-trajectory-controller

sudo apt-get install ros-melodic-ros-controllers

```

```
42  sudo apt-get install ros-melodic-pepper-meshes
43  # When configuring window opens up, you may agree to the license terms
        using the right/left arrow key and select <ok> and hit enter, and then
        select <Yes> to accept the terms and press enter
44
45  # Install rosdep
46  sudo pip install -U rosdep
47  sudo rosdep init
48  rosdep update
49
50  rosdep install --from-paths src --ignore-src -r -y
```

You might encounter the following error when launching the driver in the section 2.2.3.

> 🛑 **Error**
>
> terminate called after throwing an instance of 'qi::FutureUserException'what():
> Can't find service: ROS-Driver-Audio

Currently, the cause of the issue and its solution remain unidentified. As a temporary workaround, disable the audio service. To deactivate the Audio Service, access the boot config file using the below command and set the audio key to "false". After making the change, remember to save the file.

```
sudo nano $HOME/workspace/pepper_rob_ws/src/naoqi_driver/share/\
    boot_config.json
```

The audio service is being deactivated as a temporary workaround to ensure the functionality of the NAOqi driver. It is important to note that this action will result in the microphone not working. Despite exploring various alternatives, the problem persisted, leaving with no other viable solutions currently covered in this manual. The upcoming version of the software installation manual will address this issue and provide alternative solutions to resolve the microphone functionality along with the NAOqi driver's operation.

2.2.3 Bring up Pepper

To establish communication between the computer device and the Pepper robot, the IP address of the computer device and its network interface name, and the IP address of the Pepper robot are needed. Below are the steps to find this information. First, make sure the Pepper robot and computer device are on the same network.

1. **IP address identification of the computer**
   In order to find the IP address of the computer device, open a new terminal and execute the following commands.

   (a) Install network tool (if not installed).

   ```
   sudo apt install net-tools
   ```

   (b) Identify network ip and interface name.

   ```
   ifconfig
   ```

Figure 1: "ifconfig" command output: Network interface name and IP address.

Upon executing the "ifconfig" command, the output will display information similar to the illustration shown in Figure 1. Take note of the network interface name along with its corresponding IP address as shown in the output. This IP address will be passed as a value for roscore_ip argument later in the document.

2. **IP address identification of the Pepper robot**
   To find the IP address of the Pepper Robot, ensure the robot is powered on and connected to the network. Usually when the Pepper Robot is booting it says, "Hello, I'm Pepper, my internet address is $< robot\_ip >$ (for example, 172.29.111.230)" or in case you missed it, press the chest button once and take a note of that IP address. This IP address will be passed as a value for robot_ip argument.

Using the above IP address of the robot, the IP address and the network interface name of the computer device, the following instruction will bring Pepper up over ROS and the NAOqi driver will be launched. Below are the instructions to follow:

1. Bring up the Pepper robot, assuming the robot is turned on.

```
roslaunch pepper_dcm_bringup pepper_bringup.launch \
robot_ip:=172.29.111.230 roscore_ip:=172.29.111.229 \
network_interface:=ens160
```

2. Launch the NAOqi driver.

```
roslaunch naoqi_driver naoqi_driver.launch nao_ip:=172.29.111.230 \
roscore_ip:=172.29.111.249 network_interface:=ens160
```

2.3 Setting up the Development Environment for the Gazebo Simulator

Section 2.2 assumes access to a physical Pepper robot. However, circumstances may arise where testing software without the physical robot becomes necessary. In such cases, a simulator can serve as an alternative. This section explains how to control the Pepper robot within a Gazebo simulator. The simulation is based on an open-source Pepper ROS environment developed by Sam Pfeiffer.[1] The installation of the Gazebo simulator is presented below.

2.3.1 Installation of the Gazebo Simulator

To install the Gazebo simulator, follow the instructions below.

1. **Install the Pepper Gazebo Simulator**

   (a) Make a separate workspace for the simulator installation.

   ```
   mkdir -p $HOME/workspace/pepper_sim_ws/src
   cd $HOME/workspace/pepper_sim_ws/src
   ```

   (b) Clone the necessary packages from the GitHub repository.

   ```
   git clone -b correct_chain_model_and_gazebo_enabled \
   https://github.com/awesomebytes/pepper_robot
   ```

   ```
   git clone -b simulation_that_works \
   https://github.com/awesomebytes/pepper_virtual
   ```

   ```
   git clone https://github.com/cssr4africa/\
       gazebo_model_velocity_plugin
   ```

---

[1]The Pepper ROS simulator is available on the following links: https://github.com/awesomebytes/pepper_virtual

(c) Install additional libraries.

```
sudo apt-get install ros-melodic-tf2-sensor-msgs \
ros-melodic-ros-control ros-melodic-ros-controllers \
ros-melodic-gazebo-ros ros-melodic-gazebo-ros-control \
ros-melodic-gazebo-plugins ros-melodic-controller-manager \
ros-melodic-ddynamic-reconfigure-python
```

(d) Install pepper mesh.

```
sudo apt-get install ros-melodic-pepper-meshes
# When the configuring window opens up, you may agree to the
   license terms using the right/left arrow key and select <ok>
   and press enter, and then select <Yes> to accept the terms
   and press enter
```

(e) Build the packages in the workspace.

```
cd .. && catkin_make
```

(f) Make ROS environment variables automatically added every time a new shell is launched.

```
echo "source $HOME/workspace/pepper_sim_ws/devel/setup.bash" >> \
$HOME/.bashrc
```

```
source $HOME/.bashrc
```

2. **Run the Pepper Gazebo Simulator**

(a) Start the Gazebo simulation.

```
roslaunch pepper_gazebo_plugin \
pepper_gazebo_plugin_in_office_CPU.launch
```

Once you've successfully launched the simulator, your simulation environment should be similar to the Figure 2.

To visualize the simulation on rviz, open a new terminal and type the below command.

```
rosrun rviz rviz -d `rospack find \
pepper_gazebo_plugin`/config/pepper_sensors.rviz
```

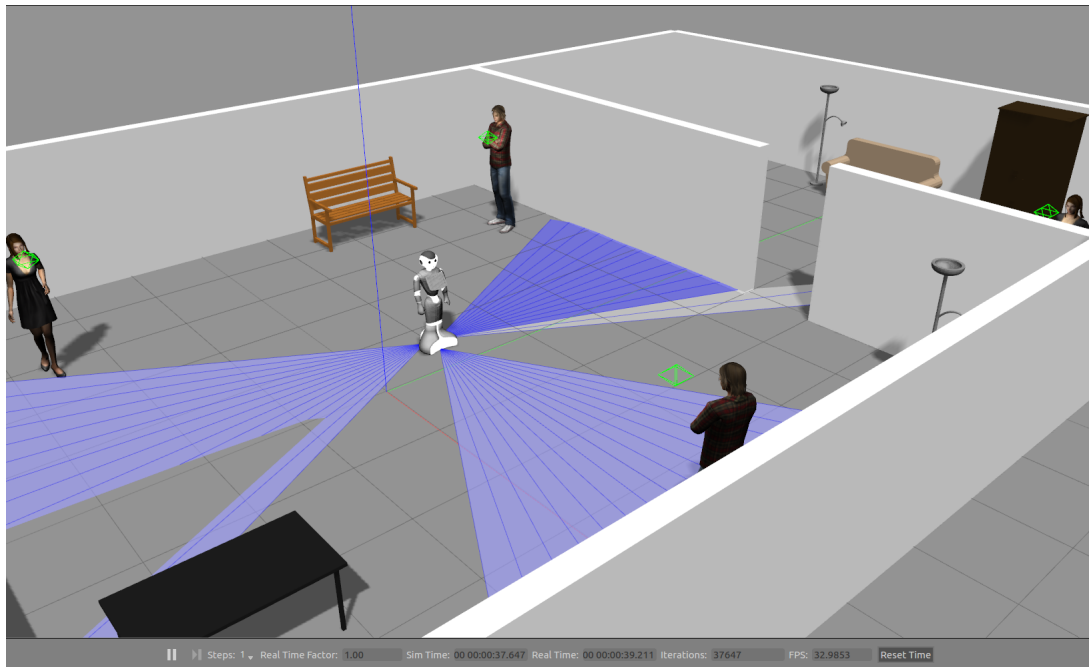The visualization that appears on rviz after launching will have a similar appearance to Figure 3.

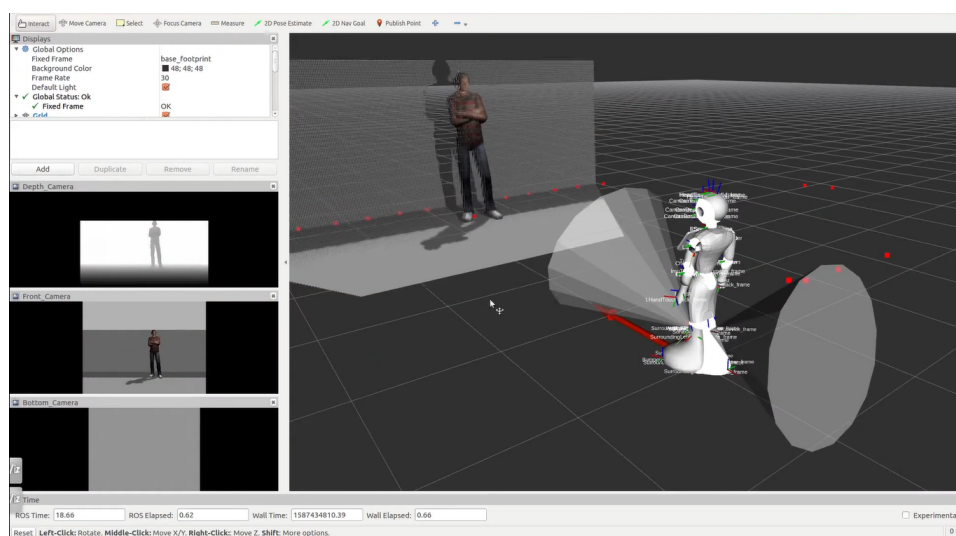Figure 2: Pepper Gazebo simulation environment.

Figure 3: Visualization of the simulation on rviz.

3 Installing and Running the CSSR4Africa Software

3.1 Installing the CSSR4Africa Software

This section will guide you through the installation process of the CSSR4Africa software in the source directory of the workspace. The installation of the software for the physical and simulator environment will be discussed below.

3.1.1 Installing the CSSR4Africa Software for the Physical Robot

The outlined steps for installing the CSSR4Africa software for the physical robot are as follows:

```
1  # Move to the source directory of the workspace
2  cd $HOME/workspace/pepper_rob_ws/src
3
4  # clone the CSSR4Africa software from the GitHub repository
5  git clone https://github.com/cssr4africa/cssr4africa.git
6
7  # build the source files
8  cd .. && catkin_make
```

Upon completing the installation above, the **src** folder within the development environment's workspace will contain the CSS4Africa software meta-package and all the packages used to enable communication between ROS and NAOqi. Below is the directory structure of the workspace for reference.

```
workspace
└── ros
    ├── build
    ├── devel
    ├── src
    │   ├── cssr4africa
    │   │   └── pepper_interface_tests
    │   ├── naoqi_dcm_driver
    │   ├── naoqi_driver
    │   ├── pepper_dcm_robot
    │   ├── pepper_moveit_config
    │   ├── pepper_robot
    │   └── pepper_virtual
```

3.1.2 Installing the CSSR4Africa Software for the Robot Simulator

The steps for installing the CSSR4Africa software for the simulated environment will be presented as follows:

```
1  # move to the source directory of the workspace
2  cd $HOME/workspace/pepper_sim_ws/src
3
4  # clone the CSSR4Africa software from the GitHub repository
5  git clone https://github.com/cssr4africa/cssr4africa.git
6
7  # build the source files
8  cd .. && catkin_make
```

3.2　Running the CSSR4Africa Software

In this section, the execution of the unit, integration, and system tests of the CSSR4Africa software packages for the physical robot and the robot simulator will be presented.

3.2.1　The Pepper Interface Tests Package

The pepper_interface_test package contains the source files that have been written to perform the unit tests to evaluate the functionality and performance of sensors and actuators on the physical robot and the robot simulator. The actuator test includes testing the head, right arm, left arm, leg and wheels. The sensor test includes testing the front sonar, back sonar, front camera, bottom camera, depth camera and laser sensor. During the execution of each test, you will be prompted to enter any key to continue to the next test after each actuator or sensor is tested. The directory structure of the pepper_interface_test is as follows:

```
Pepper_interface_tests
├── config
│   ├── actuatorTestConfiguration.ini
│   ├── actuatorTestInput.ini
│   ├── sensorTestInput.ini
│   └── sensorTestConfiguration.ini
├── data
│   ├── pepperTopics.dat
│   └── simulatorTopics.dat
├── include
│   └── pepper_interface_tests
│       ├── actuatorTest.h
│       └── sensorTest.h
├── launch
│   └── interface.launch
├── src
│   ├── actuatorTest.cpp
│   └── sensorTest.cpp
├── README.md
├── CMakeLists.txt
└── package.xml
```

To run the test on the physical robot, change the first line of the actuatorTestConfiguration.ini and sensorTestConfiguration.ini file to "platform robot". On the other hand, in order to run the test on the simulator platform, change the first line of the actuatorTestConfiguration.ini and sensorTestConfiguration.ini file to "platform simulator".

Before starting the test on the physical robot, run the launch file of "pepper_interface_tests" package and NAOqi driver on two different terminals. Please replace the variables robot_ip, roscore_ip, and network_interface_name with the values obtained from the section 2.2.3. Then proceed to run the actuator and simulator test.

```
roslaunch pepper_interface_tests interface.launch \
robot_ip:=<robot_ip> roscore_ip:=<roscore_ip> \
network_interface:=<network_interface_name>
```

```
roslaunch naoqi_driver naoqi_driver.launch nao_ip:=<robot_ip> \
roscore_ip:=<roscore_ip> network_interface:=<network_interface_name>
```

Before starting the test on the robot simulator, open a new terminal and launch the simulator. Then proceed to run the actuator and simulator test.

```
roslaunch pepper_gazebo_plugin \
pepper_gazebo_plugin_in_office_CPU.launch
```

To begin the tests open a new terminal and perform the following tests.

1. Test the sensors

   ```
   rosrun pepper_interface_tests sensorTest
   ```

   During each sensor test, the sensor's numerical output data will be logged to a 'sensorTestOutput.dat' file. Additionally, real-time sensor values will be displayed in the terminal for a 10-second duration. Meanwhile, sensor data images will be showcased in a separate window, also for a 10-second interval.

   Expected output for the sonar sensor (front and back sonar): The test outputs the back sonar sensor frame ID, its field view, and the minimum and maximum distance range it can measure.

   Expected output for the cameras (front and bottom camera): The test displays an RGB image of the scene captured by the camera.

   Expected output for the depth camera: The test displays a gray-scale image of the scene captured by the camera.

   Expected output for the laser sensor: The test outputs a set of values wrapped in the message published by the laser sensor. These values include respectively: the laser sensor frame ID, the start and end angles of the scan, the angular distance between measurements, the time between measurements and scans and the sensor's minimum and maximum range values.

2. Test the actuator

   ```
   rosrun pepper_interface_tests actuatorTest
   ```

   Expected output for the head actuator: The test includes testing the head pitch and head roll. The head pitch test enables the robot's head to tilt along its vertical axis. The head yaw test turns the robot's head from side to side. In each joint test, the maximum, minimum and mid-range position of the joints will be tested after the position of each joint is set to a zero position value.

   Expected output for the left and right arm : The test includes testing the left/right arm shoulder pitch, shoulder roll, elbow roll, elbow yaw, and wrist yaw. The shoulder pitch test moves the left/right arm of the robot up and down with respect to the shoulder joint. The shoulder roll test moves the left/right arm of the robot from side to side with respect to the shoulder

joint. The elbow roll test moves the left/right arm of the robot from side to side with respect to the elbow joint. The elbow yaw test moves the left/right forearm of the robot in a rotational movement around the elbow axis. The wrist yaw test moves the left/right wrist of the robot in a rotational movement around the wrist axis. In each joint test, the maximum, minimum and mid-range position of the joints will be tested after the position of each joint were set to a zero position value. The zero position value will set the arm to point forward.

Expected output for the leg: The test includes testing the hip pitch, hip roll and knee pitch. The hip pitch test moves the robot to the front and back with respect to the hip joint. The hip roll test moves the robot to the left and right side with respect to the hip joint. The knee pitch moves the robot to the front and back with respect to the knee joint. In each joint test, the maximum, minimum and mid-range position of the joints will be tested after the position of each joint were set to a zero position value.

Expected output for the wheels: The test includes testing the linear and angular movement of the robot. The linear movement will enable the robot to move forward and backward. The angular movement will enable the robot to move clockwise and anticlockwise.

A   Appendix   Server Specification and Accessibility

To support the software development process for CSSR4Africa, a ROS-based environment is necessary, requiring a Ubuntu host for writing software. While it is possible to install your own Ubuntu OS and set up the development environment independently, a dedicated server has been provisioned specifically for those affiliated with Carnegie Mellon University Africa (CMU-Africa). Users can conveniently create profiles on a Ubuntu 18.04 virtual machine, which functions as the designated server. Ubuntu 18.04 is the recommended version due to its compatibility with ROS Melodic, the meta-operating system utilized for the CSSR4Africa project. By leveraging this server, users can streamline the setup process and ensure a standardized environment for development tasks.

A.1   Technical Specifications & Accessibility

This section will provide a specification of the server, and the accessibility of the server from different operating systems.

A.1.1   Specifications

The server has the following specifications:

- Storage: 1 TB

- RAM: 16 GB

- CPU: 12 Cores

A.1.2   Accessibility

Users can connect to the server in two different modes: non-graphical user interface or the command line mode (Non-GUI) and a graphical user interface (GUI).

1. **Non-Graphical User Interface -Windows, Linux, and Mac**
   Connecting to the server via the command line entails using the Secure Shell Protocol (SSH). You can do this using the terminals available on the three operating systems (OS) supported by this manual, namely Windows, Linux, and Mac. Additionally, you can use PuTTY, one of the most popular applications for SSH connection on Windows. Below is the bash command line template to connect to the server.

```
1 # step 1: enter the command line below to request a connection
2 # eg: ssh cssr4a-server@192.168.1.1
3 ssh <username>@<server_ip>
4 # step 2: enter your password based on the message prompted below
5 # eg: cssr4a-server@192.168.1.1 password: enter_the_password_here
6 <username>@<ip>`s password: _
```

2. **Graphical User Interface**

   For Windows Users

   To establish a connection with the server's GUI, you'll need a remote desktop application. Windows users can select from two applications for this purpose: Remote Desktop Connection and Microsoft Remote Desktop. Both of these applications are accessible through the Windows store.

   For Mac OS Users

   If you're using Mac OS, you'll find Microsoft Remote Desktop available for download in the Mac OS store. This application facilitates connecting to the server's GUI remotely.

   For Ubuntu Users

   Ubuntu users can set up a remote desktop connection by installing Remmina, a remote desktop application tailored for Ubuntu. This choice will enable you to access the server's GUI seamlessly.

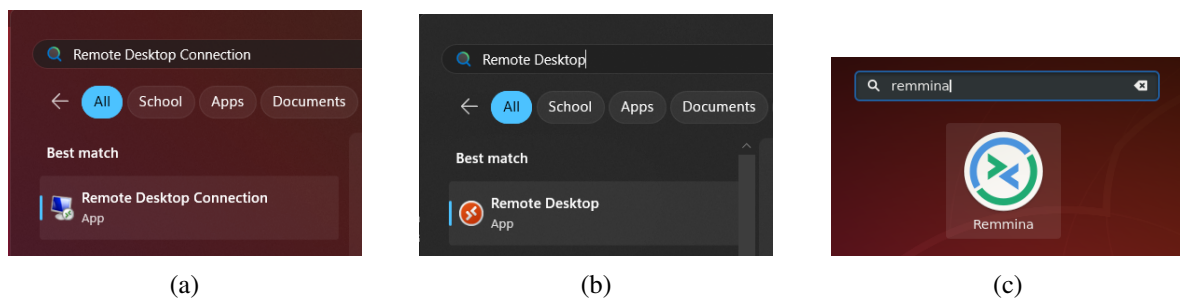
   (a)   (b)   (c)

   Figure 4: (a) Remote Desktop Connection Software. (b) Microsoft Remote Desktop application (c) Remmina.

   Installation of Remmina software can be done using either the command line or the Ubuntu Software.

   (a) **Installation of Remmina software using the command line**
      - **Install using snap**
        This is highly recommended since you are more likely to have access to the last update of the software.

        ```
        # Step 1: Get your system ready for installation
        sudo apt-get update
        # Step 2: Install snap if not installed or skip this step
        sudo apt-get install snapd
        # Step 3: install remmina
        sudo snap install remmina
        ```

      - **Install using PPA (Personal Package Archive)**
        While this works, the PPA is not actively maintained. Therefore, you are more likely to download an older version of the software.

```
1  # Step 1: add the ppa repository
2  sudo apt-add-repository ppa:remmina-ppa-team/remmina-next
3  # Step 2: get your system ready for installation
4  sudo apt update
5  # step 3: install remmina
6  sudo apt install remmina remmina-plugin-rdp \
7  remmina-plugin-secret
```

(b) **Installation using Ubuntu Software (GUI)**

This requires opening the Ubuntu Software application, searching for Remmina, and clicking on the install button at the right top of your screen to download the software.
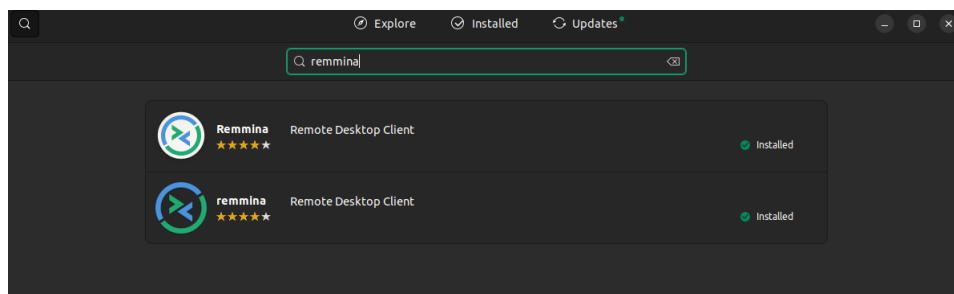

Figure 5: Remmina on the Ubuntu Software Application

After installing the remote desktop application of your preference, you can connect to the server. This implies that users must know the server's IP address as well as their credentials, and provide this information with the remote desktop application. It is important to note that users should be connected to the CMU-Secure network on campus or through the school's VPN. The server's IP address is: **172.29.111.229**.

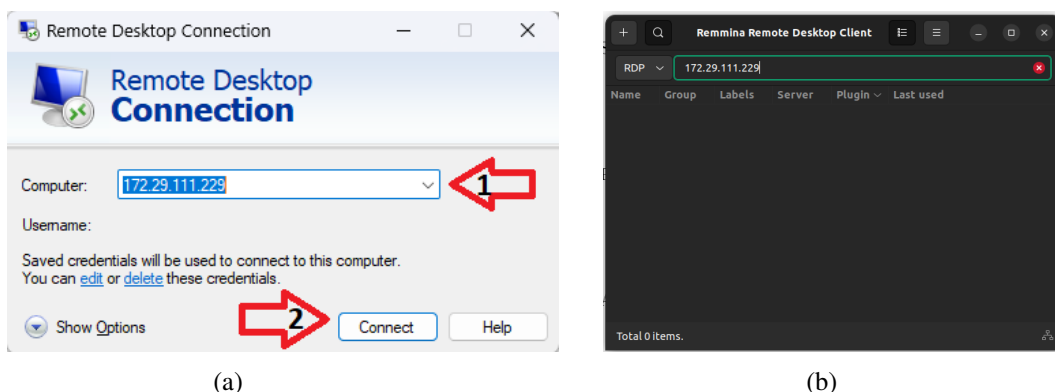(a)   (b)

Figure 6: (a) Connection page Remote Desktop Connection. (b) Connection page Remmina.

Figure 6a shows the connection page of the Remote Desktop Connection application, which features a connection button for users to establish a connection with the server. On the other hand, 6b illustrates the connection page of Remmina. In Remmina, users need to press the "Enter" key to initiate the connection to the server. It is important to note that in Remmina, the default connection type is RDP (Remote Desktop Protocol). This protocol enables users to access the server through a GUI. Refrain from changing the connection protocol to ensure

uninterrupted access to the server in GUI mode. However, if you opt for the SSH protocol, you will gain access to the server through a terminal interface.

After successfully logging in using your user profile information, your screen should resemble figure 7.
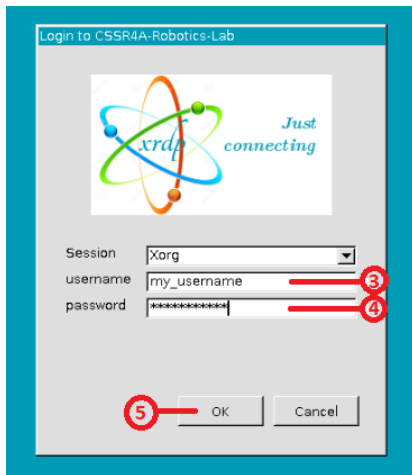
(a) Login with user credentials
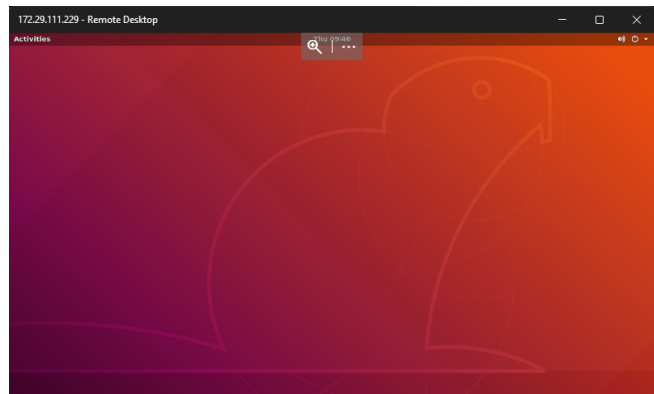
(b) Server's Ubuntu home page.

Figure 7: After logging in with the user credentials in (a), the home page of the Ubuntu server will resemble (b)

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Adedayo Akinade, CMU-Africa
Deogratias Amani, CMU-Africa
Yohannes Haile, CMU-Africa
Mihiretab Taye Hordofa, CMU-Africa
Kleber Kabanda, CMU-Africa
Natasha Mutangana, CMU-Africa
David Vernon, CMU-Africa
Pamely Zantou, CMU-Africa

Document History

Version 1.0

First draft.
CSSR4Africa Team.
26 July 2023.

Version 1.1

Fixed several typographical errors.
Updated the Pepper diagnostic routines test on the physical robot to include the expected output of the tests.
Added a new test for the wheels and removed the previous test.
CSSR4Africa Team.
31 July 2023.

Version 1.2

Added the Pepper diagnostic routine test for the robot simulator.
Added non-containerized installation of Pepper Gazebo simulator.
Split the containerized Pepper Gazebo simulator installation into installing and running sections.
Incorporated the updated software's tests into the Pepper diagnostic routine tests.
CSSR4Africa Team.
10 August 2023.

Version 1.3

Changed the test from Pepper diagnostic routines to Pepper interface tests.
Removed containerized docker installation.
Grouped the test section to sensorTest and actuatorTest.
Changed the extension of the configuration files from .txt to .ini. Changed the extension of data files from .txt to .dat.
Added appendix section for server specification and accessibility.
Yohannes Haile and Mihiretab Hordofa.
7 September 2023.