CSSR for

Culturally Sensitive Social Robotics for Africa

D3.3 Software Installation Manual

Due date: **1/10/2023**
Submission Date: **26/07/2023**
Revision Date: **27/04/2025**

Start date of project: **01/07/2023** Duration: **36 months**

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

Responsible Person: **CSSR4Africa Team** Revision: **1.9**

Project funded by the African Engineering and Technology Network (Afretec) Inclusive Digital Transformation Research Grant Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including Afretec Administration)	
RE	Restricted to a group specified by the consortium (including Afretec Administration)	
CO	Confidential, only for members of the consortium (including Afretec Administration)	

Executive Summary

Deliverable D3.3 is 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 system's current capabilities. This task aims 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.

Contents

1 Introduction

Software Installation Manual(D3.3) outlines the detailed steps necessary for installing and executing the software critical for deploying parts or the entirety of the CSSR4Africa system, as well as conducting the requisite unit, integration, and system tests. It is designed for systems running a native Ubuntu 20.04 OS and equipped with ROS Noetic. The manual also specifies the NAOqi OS version for the robot as 2.5.10.7.

The manual is divided into several key sections. The initial section, referred to as "Setting up Pepper Robot, delineates the preliminary configurations required for the Pepper robot before any software installations. Following this, the "Setting up the Development Environment" section provides comprehensive instructions for preparing the development environments tailored for both the physical robot and the Gazebo simulator. This includes a walkthrough of the prerequisite setup common to both environments, followed by specific installation procedures. Lastly, the "Installing and Running the CSSR4Africa Software" section discusses installing and initiating the CSSR4Africa software, ensuring a smooth deployment.

2 Setting up Pepper robot

To configure the Pepper robot for the first time, reference the Pepper Robot Pocket Guide included with your robot when shipped. This guide provides detailed instructions for unboxing and setting up your robot. For additional support, access the online configuration guide at Pepper User Guide. We are utilizing version 1.8 of the Pepper robot. For comprehensive technical details, including its sensor and actuator capabilities, please refer to the Pepper Technical Detail [1]. This resource offers a thorough overview of the robot's technical specifications

Next, establish a Wi-Fi connection between the robot and the user's computer. It's advisable to use a dedicated Wi-Fi router for this purpose, rather than relying on a campus wireless network (CMU), to avoid the complications of logging in with a school ID (Andrew ID). To achieve this, connect a router to the campus's network via an Ethernet cable. Upon powering your laptop, instructions on the tablet will guide you to connect to this router. Once connected, the Pepper robot will vocalize its IP address when you press the power button once. This IP address is crucial, as it will be used to establish a connection through ROS (Robot Operating System), enabling communication and control of the robot from your computer. Furthermore, we set up the router for the robot to have a static IP address using the robot's MAC address.

The MAC address can be found when accessing the Pepper web page as shown in Figure 1 by opening a web browser and entering the IP address. Then log in by setting the username and password you configured when setting up the robot. Then using the Wi-Fi MAC address configure the router to have the pepper robot IP address permanent. This can be done by looking in the DHCP or static IP Assignment. Then assign a static IP address by adding a new reservation, enter the MAC address, and save the setting. Restart the robot and the router for the changes to take effect.

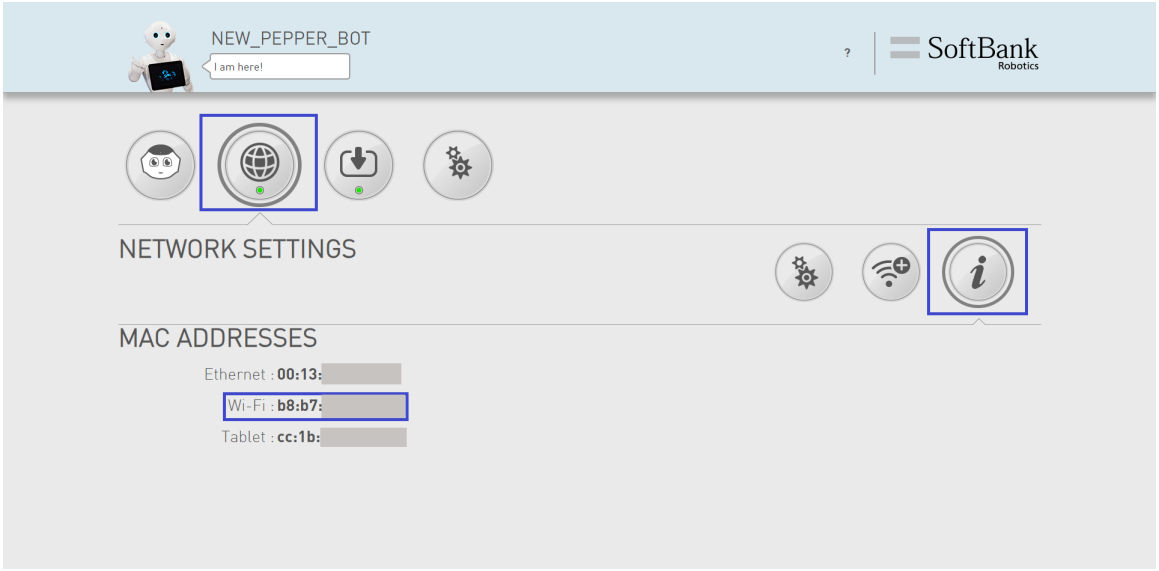

Figure 1: Pepper Web Page

3 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.

3.1 Prerequisite

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

Note: There are two alternatives below to set up the development environment for the CSSR4Africa project. One is to follow the step-by-step guide to install ROS Noetic and the required dependencies or use the alternative method to install the required software using the provided shell scripts

3.1.1 Installing from the shell scripts.

1. First clone the GitHub repository and navigate to the directory.

```
git clone https://github.com/cssr4africa/software-Installation-
    scripts.git
```

2. Run the shell script for installing ROS-Noetic.

```
./install_ros_noetic.sh
```

3. Run the shell script for installing the necessary packages and setting up the workspace

```
./install_pepper_ws.sh
```

3.1.2 Installation Steps of the ROS-Noetic

1. Update and Upgrade the system.

```
sudo apt update && sudo apt upgrade -y
```

2. Install Curl, Git, and Python3-pip.

```
sudo apt install -y curl git python3-pip net-tools
```

3. 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'
```

4. Setup your keys.

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

5. Update Debian package index.

```
sudo apt update
```

6. Install ROS Noetic with the default configurations.

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

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

```
echo "source /opt/ros/noetic/setup.bash" >> $HOME/.bashrc
```

8. Sourcing the .bashrc file.

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

9. Install additional ROS packages.

```
sudo apt install -y python3-rosdep python3-rosinstall \
python3-rosinstall-generator python3-wstool build-essential
```

10. Initialize rosdep.

```
sudo rosdep init
rosdep update
```

3.2 Setting up the Development Environment for the Physical Robot

3.2.1 Installing NAOqi Driver and ROS Packages

The NAOqi driver is a module that provides bridge capabilities. It publishes sensory data and the pepper position, enabling ROS to call part of the NAOqi API. The driver is dependent on `naoqi_libqi`, `naoqi_libqicore`, and `naoqi_bridge_msgs` packages. Those packages could be installed using apt-get. The rest of the packages are installed from the source by cloning them from GitHub. The three packages `naoqi_dcm_driver`, `naoqi_driver`, and `pepper_dcm_robot` have been modified compared to the official repository on GitHub. Hence, these updated packages are under the CSSR4Africa repository.

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

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

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

# Clone NAOqi DCM driver repository
git clone https://github.com/cssr4africa/naoqi_dcm_driver.git

# Clone NAOqi driver repository
git clone https://github.com/cssr4africa/naoqi_driver.git
```

```
# Clone Pepper DCM driver repository
git clone https://github.com/cssr4africa/pepper_dcm_robot.git

# Clone Pepper virtual repository
git clone https://github.com/ros-naoqi/pepper_virtual.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-noetic-joint-trajectory-controller
sudo apt-get install ros-noetic-ros-controllers
sudo apt-get install ros-noetic-pepper-meshes
```

3.2.2 Installing and Configuring the Python NAOqi SDK

The NAOqi Framework, a cross-language programming framework for programming Pepper, supports both C++ and Python. The Python NAOqi SDK was used when the microphone sensor failed to function with the naoqi_driver, the Python SDK was employed to access the microphone sensors' API. This method allowed the modified naoqi_driver to publish the captured audio data successfully. Furthermore, since the tablet was not supported by the ROS-NAOqi drivers, the Python SDK was used to provide an interface through ROS.

rospy (a python client library for ROS(Robot Operating System) which enables the development of ROS-based applications was used mainly to define publishers, subscribers, and services. However, one of the limitations of Python SDK is that it runs on Python 2 whereas rospy uses Python 3 (ROS-Noetic). To overcome this problem, a UDP socket was used that will send the data from Python 2 to the python3 script that receives data and then publishes or provides a ros service using the rospy.

1. Change directory to home

   ```
   cd $HOME
   ```

2. Installing Python 2.7

   ```
   sudo apt install python2

   # check the version of the Python
   python2 --version
   ```

3. Installing Pip2

```
curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py
```

4. Installing the necessary packages

```
sudo apt-get install libpython2.7
sudo apt-get install libatlas3-base
sudo python2 get-pip.py
```

5. Checking the pip2 version

```
pip2 --version
```

6. Installing NumPy

```
pip2 install numpy
```

7. Download the Python SDK by using the following command.

```
wget -S -L https://community-static.aldebaran.com/resources/2.5.5/\
sdk-python/pynaoqi-python2.7-2.5.5.5-linux64.tar.gz
```

8. Extract the SDK

```
tar -xvf pynaoqi-python2.7-2.5.5.5-linux64.tar.gz
```

9. To ensure that the `PYTHONPATH` environment variable is automatically set every time a new shell session starts.

```
# If you extracted it in another directory you can set it up by
   putting the correct path here.

In addition, Write the entire code in one line (avoid using two
lines)

echo "export PYTHONPATH=${PYTHONPATH}:$HOME/pynaoqi-python2
   .7-2.5.5.5-linux64/lib/python2.7/site-packages" >> $HOME/.bashrc
```

To apply the changes immediately, you can source your

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

3.2.3 Bring up Pepper

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

1. **IP address identification of the computer** To find the IP address of the computer device, open a new terminal and execute `ifconfig` commands.

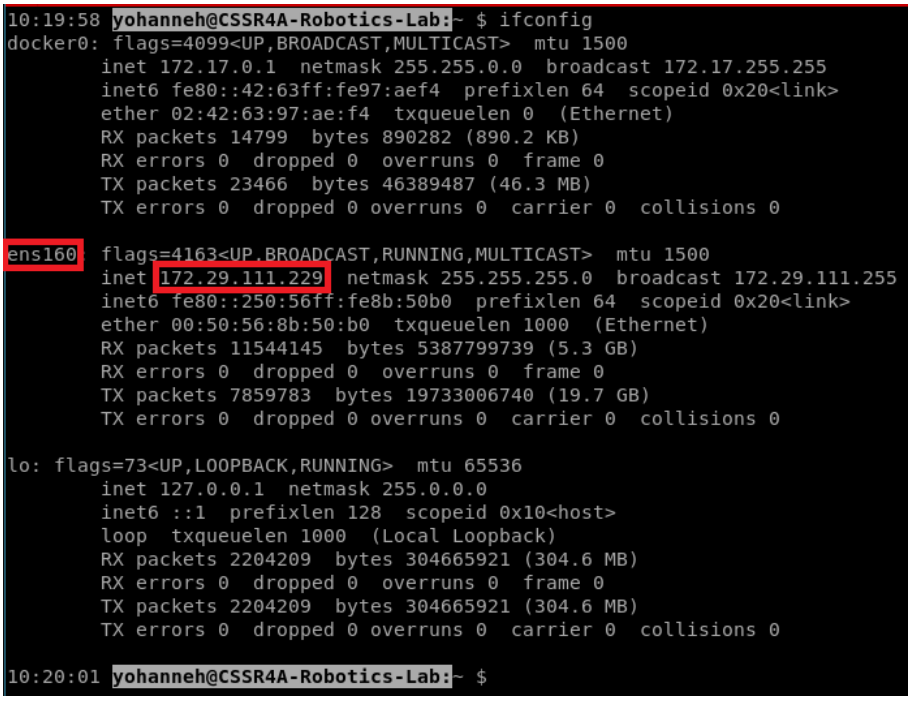

Figure 2: `ifconfig` command output: Network interface name and IP address.

Upon executing the `ifconfig` command, the output will display information similar to the illustration shown in Figure 2. Please take note of the network interface name and its corresponding IP address as shown in the output. This IP address will be passed as a value for the `roscore_ip` argument. Similarly, the network interface must be passed as a value for the `network_interface` argument.

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)". If you missed it, press the chest button once and take note of that IP address. This IP address will be passed as a value for the 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 for the sample shown above:

1. Open a new terminal and 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. Open a new terminal and launch the NAOqi driver.

```
roslaunch naoqi_driver naoqi_driver.launch nao_ip:=172.29.111.230 \
roscore_ip:=172.29.111.229 network_interface:=ens160
```

3.3 Setting up the Development Environment for the Gazebo Simulator

Section 3.2 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.

3.3.1 Installation of the Gazebo Simulator

To set up the Gazebo simulator for the Pepper robot, follow the instructions below.

1. **Setup Pepper in the 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
```

   (c) Install additional libraries.

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

   (d) Install pepper mesh.

```
sudo apt-get install ros-noetic-pepper-meshes
```

   (e) Build the packages in the workspace.

```
cd .. && catkin_make -DSIMULATOR=ON
```

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

Note: After setting up the physical robot's environment, follow step f to modify the `.bashrc` will result in the simulator environment being sourced last, overriding the physical robot's settings. If your main work is with the physical robot, consider skipping this step. You can manually `source devel/setup.bash` when necessary for simulator tasks to maintain the correct environment setup

(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
```

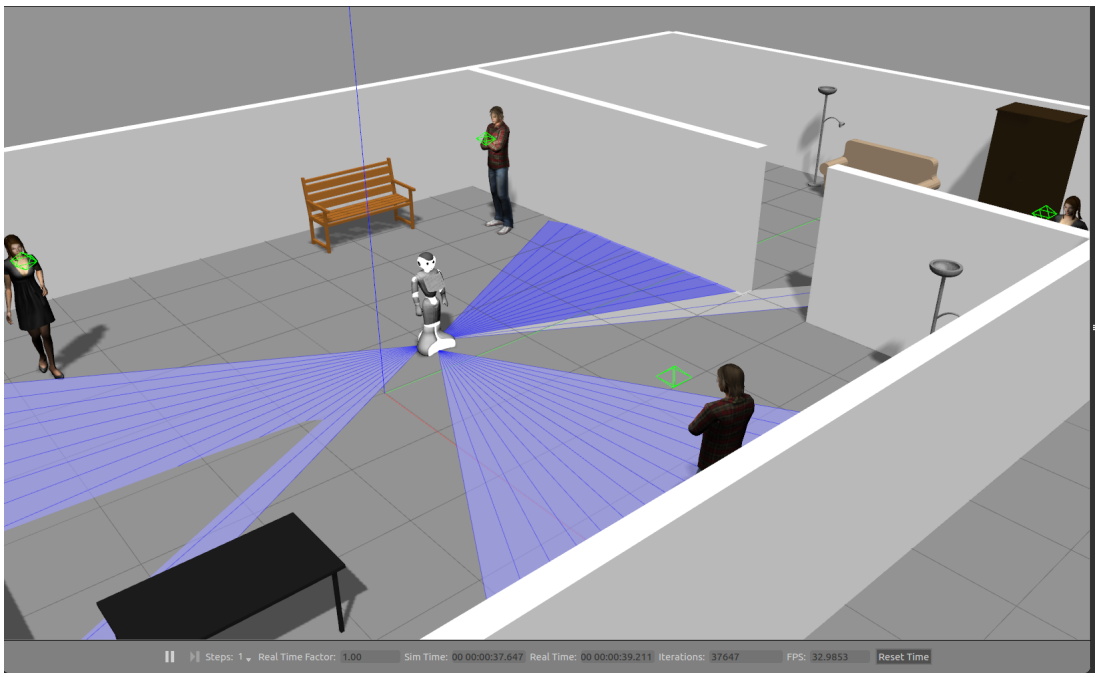

Figure 3: Pepper Gazebo simulation environment.

Once you've successfully launched the simulator, your simulation environment should be similar to Figure 3. 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 4.

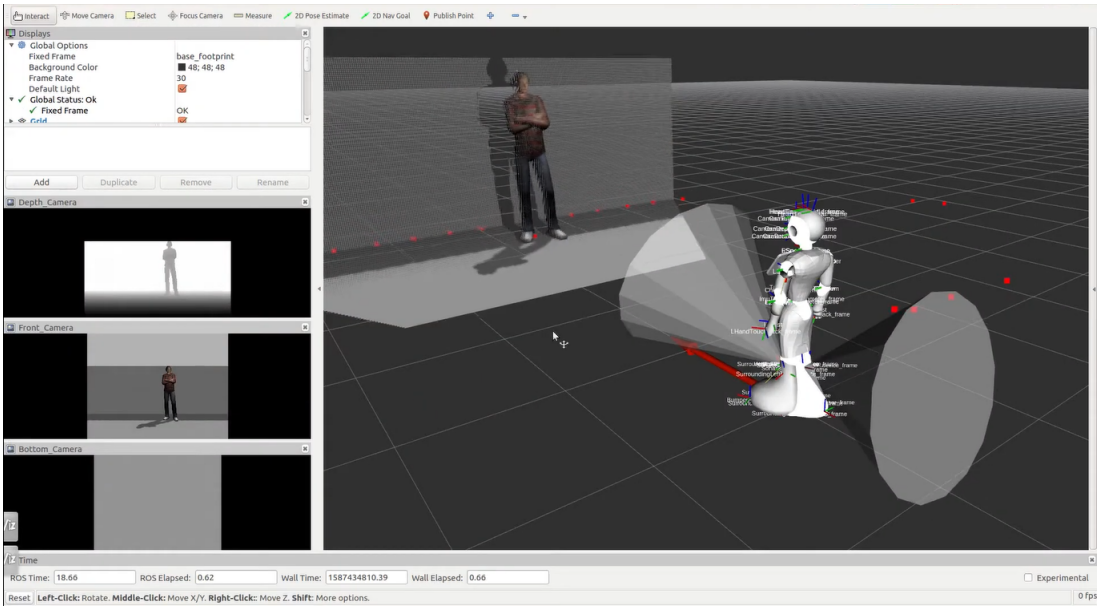

Figure 4: Visualization of the simulation on rviz.

4 Installing and Running the CSSR4Africa Software

4.1 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 software installation for the physical and simulator environment will be discussed below.

4.1.1 Installing the CSSR4Africa Software for the Physical Robot

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

```
# Move to the source directory of the workspace
cd $HOME/workspace/pepper_rob_ws/src

# Clone the CSSR4Africa software from the GitHub repository
git clone https://github.com/cssr4africa/cssr4africa.git

# Build the source files
cd .. && catkin_make

# Clone the models from HuggingFace:
cd && git lfs install
git clone https://huggingface.co/cssr4africa/cssr4africa_models

# Move the face detection models to the models directory:
mv cssr4africa_models/face_detection/models/* $HOME/workspace/
   pepper_rob_ws/src/cssr4africa/cssr_system/face_detection/models

# Move the person detection models to the models directory:
mv cssr4africa_models/person_detection/models/* $HOME/workspace/
   pepper_rob_ws/src/cssr4africa/cssr_system/person_detection/models

# Clone the unit test data from HuggingFace:
cd && git lfs install
git clone https://huggingface.co/cssr4africa/
   cssr4africa_unit_tests_data_files

# Move the face detection test data to the data directory:
mv cssr4africa_unit_tests_data_files/face_detection_test/data/* $HOME/
   workspace/pepper_rob_ws/src/unit_tests/face_detection_test/data

# Move the person detection test data to the data directory:
mv cssr4africa_unit_tests_data_files/person_detection_test/data/* $HOME/
   workspace/pepper_rob_ws/src/unit_tests/person_detection_test/data
```

Upon completing the installation above, the **src** folder within the development environment's workspace will contain the CSS4Africa software meta-package and all the necessary packages for the project. As of now, only five software have been uploaded in the `cssr_system` and `unit_tests` directory: `animateBehvaiour`, `faceDetection`, `gestureExecution`, `overtAttention`, and `personDetection`. Below is the directory structure of the workspace for reference.

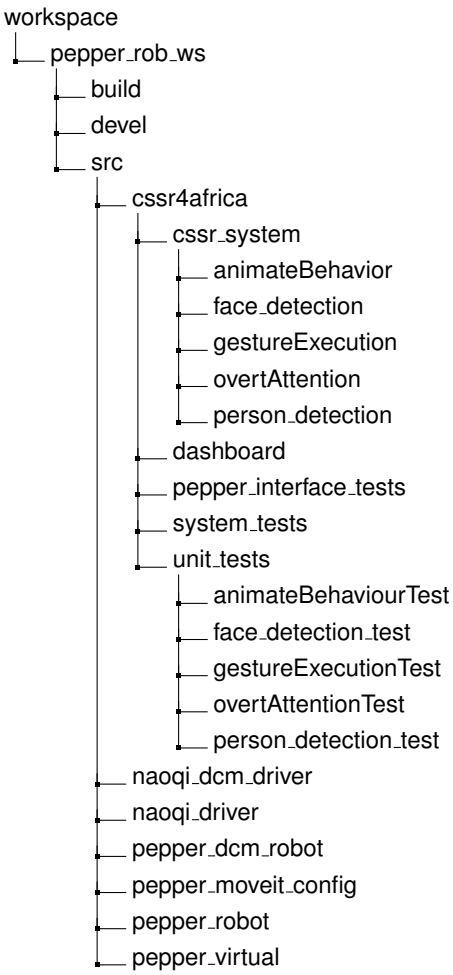

Figure 5: Directory structure for the CSSR4Africa software repository.

4.1.2 Installing the CSSR4Africa Software for the Simulator Robot

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

```
# Move to the source directory of the workspace
cd $HOME/workspace/pepper_sim_ws/src

# Clone the CSSR4Africa software from the GitHub repository
git clone https://github.com/cssr4africa/cssr4africa.git
# Build the source files
cd .. && catkin_make -DSIMULATOR=ON
```

4.2 Running the CSSR4Africa Software

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

4.2.1 The Pepper Interface Tests Package

The pepper_interface_test package contains the node 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 directory structure of the pepper_interface_test is as follows:

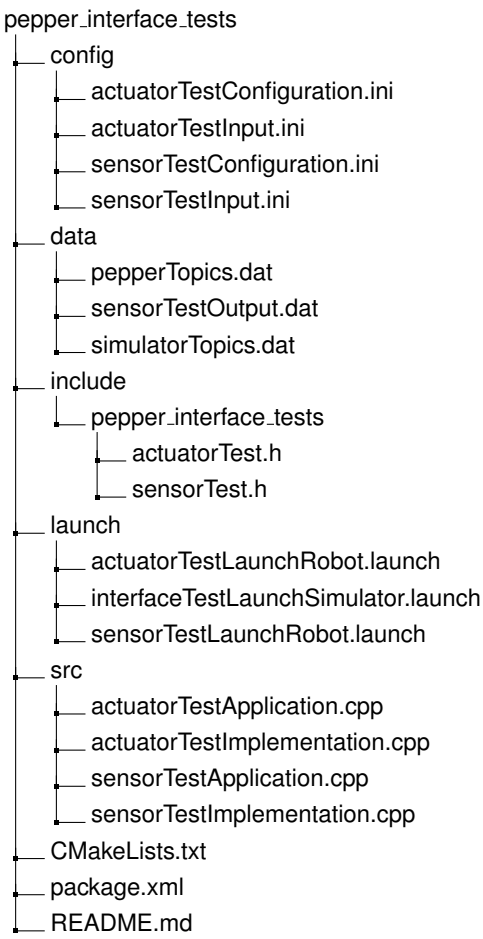

Figure 6: Directory structure for the `pepper_interface_tests` package. There are two nodes: `actuatorTest` and `sensorTest`.

The actuator test includes testing the head, hand, arm, leg, and wheels. The sensor test comprehensively assesses multiple sensors: front and back sonar, front and bottom cameras, stereo and depth cameras, laser sensor, microphone, odometry, and joint states.
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, 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 the `pepper_interface_tests` package either the actuatorTestLaunchRobot.launch or sensorTestLaunchRobot.launch. Please replace the variables `robot_ip`, `roscore_ip`, and `network_interface_name` with the values obtained from the section 3.2.3. Then proceed to run the actuator and sensor test.

```
cd $HOME/workspace/pepper_rob_ws
source devel/setup.bash
```

Before starting the test on the robot, open a new terminal and launch either the sensor launch file or the actuator launch file.

Note: If you are switching to a new ROS workspace. Activate the workspace environment by sourcing the `devel/setup.bash`. This process ensures that your terminal session is configured to use the current workspace's ROS packages and environment settings.

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

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

Note: The launch file can be updated to incorporate default values for each parameter, simplifying the initiation process for the physical robot. To achieve this, navigate to either the `sensorTestLaunchRobot.launch` or `actuatorTestLaunchRobot.launch` file and modify the default parameter values to reflect those currently used.

If you are running the test for the simulator, start by sourcing the simulator workspace

```
cd $HOME/workspace/pepper_sim_ws
source devel/setup.bash
```

```
roslaunch pepper_interface_tests interfaceTestLaunchSimulator.launch
```

To begin the tests open a new terminal and run the following node.

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. Meanwhile, sensor data images will be showcased in a separate window for a 10-second interval.

   Please go through deliverables to Deliverable D4.1 for a detailed report for the sensor Test task.

2. Test the actuator

```
rosrun pepper_interface_tests actuatorTest
```

The acutator test is designed to unit test each joint by moving the joint to its minimum, maximum, and mid-range position at a selected angular velocity. The progress for testing each joint will be displayed in the terminal.

Please go through deliverables to Deliverable 5.1 for a detailed report for the actuator Test task.

References

[1] Pepper technical specifications. http://doc.aldebaran.com/2-5/family/pepper_technical/index_pep.html.

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 the appendix section for server specification and accessibility.
Yohannes Haile and Mihiretab Hordofa.
7 September 2023.

Version 1.4

Setting up Pepper Robot section added.
Changed the ROS-Version from ROS-Melodic to ROS-Noetic.
Removed Installation C++ NAOqi SDK.
Added Installation of Python NAOqi SDK.
Moved the NAOqi DCM driver, Pepper DCM, and NAOqi driver repository to CSSR4Africa due to code modification.
Fixed the Audio Service issue.
Modified the directory structure.
Changed the launch file for the pepper_interface_test for the physical and simulator.
Yohannes Haile .
25 March 2024.

Version 1.5

Added option for building the simulator workspace.
Removed "Package Installation of pepper tablet" section.
Moved the source of the workspace to a different section.
Page(16) removed the "Naoqi Driver"
Corrected the link for the sensor Test.
Joined multiple package installations into one.
Yohannes Haile .
20 August 2024.

Version 1.6

Modified the diretcory structure in Figure 5 to show the software already uploaded on the GitHub Repository. The software include `animateBehaviour`, `gestureExecution`, and `overtAttention`.
Corrected the name of the `pepper_interface_tests package` in the directory structure.
Adedayo Akinade .
7 February 2025.

Version 1.7

Added a `utilities` sub-directory that contains stand-alone software, such as the C++ helper classes that provide access to the culture knowledge base and the environment knowledge base. These classes are used by the ROS nodes that need to access this knowledge; see Deliverable D3.1 System Architecture.
David Vernon
17 February 2025.

Version 1.7

Added a `utilities` sub-directory that contains stand-alone software, such as the C++ helper classes that provide access to the culture knowledge base and the environment knowledge base. These classes are used by the ROS nodes that need to access this knowledge; see Deliverable D3.1 System Architecture.
David Vernon
17 February 2025.

Version 1.8

Removed the `utilities` sub-directory since the C++ helper classes will be integrated directly in the software for the `behaviorController` node and not be made available independently.
Added a `dashboard` sub-directory for the dashboard visualization software.
David Vernon
11 March 2025.

Version 1.9

Included steps to obtain the models and data files from HuggingFace for the `faceDetection` and `personDetection` software in the installation steps for the physical robot.
Modified the directory structure to include two new software modules:
`faceDetection` and `personDetection`.
Adedayo Akinade
27 April 2025.