CSSR for

Culturally Sensitive Social Robotics for Africa

D5.1 Actuator Test

Due date: **1/10/2023**
Submission Date: **26/03/2024**
Revision Date: **n/a**

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

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

Responsible Person: **Yohannes Haile** Revision: **1.0**

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 D5.1 is designed to establish a comprehensive suite of unit tests ensuring the accurate and reliable functioning of the actuators. To achieve this, the test will consist of a series of tests for the actuators namely head, arms, hands, legs, and wheels. The test will be conducted on both the physical robot and the simulator. The central elements of this deliverable comprise the creation of a ROS node named `actuatorTest`, the generation of a detailed report outlining the developmental procedures, refinement of requirements, and explicit definition of functional characteristics.

Additionally, a user manual will be provided to guide users through the construction and launch of the module. The interface design will cover input, output, and control data, while also specifying appropriate data structures. All coding activities will strictly adhere to established software engineering standards as set out in Deliverable D3.2.

Contents

1 Introduction

This document examines the actuator functionality within the Pepper robot, highlighting the pivotal role these components play in facilitating its physical interactions. Figure 1 showcases the distribution of 20 joint actuators across Pepper—spanning the head, arms, hands, and wheels—enabling a broad spectrum of movements and behaviors.

The actuators embedded in Pepper's head enable in executing turn and nod motions, thereby fostering engaging interactions through nuanced head movements. The arm, and hand actuators empower Pepper with the capability to mimic human gestures, significantly enhancing its non verbal communicative and interactive potential. The actuators in the hands further permit the opening and closing motions. Such functionality is vital for projects emphasizing non-verbal communication through various gestures and movements.

Moreover, the inclusion of actuators in the torso and hips extends Pepper's mobility, allowing it to perform bends and twists. This flexibility is crucial for adapting Pepper's movements to reflect various cultural norms of body language, thereby augmenting its ability to interact in a culturally sensitive manner.

The outcomes of the Actuator Test (Deliverable 5.1) are crucial for informing the development of the `Animate Behavior Subsystem` (Deliverable 5.2) by establishing an essential link between the evaluation of actuator performance and the enhancement of the Pepper robot's interactive behaviors.

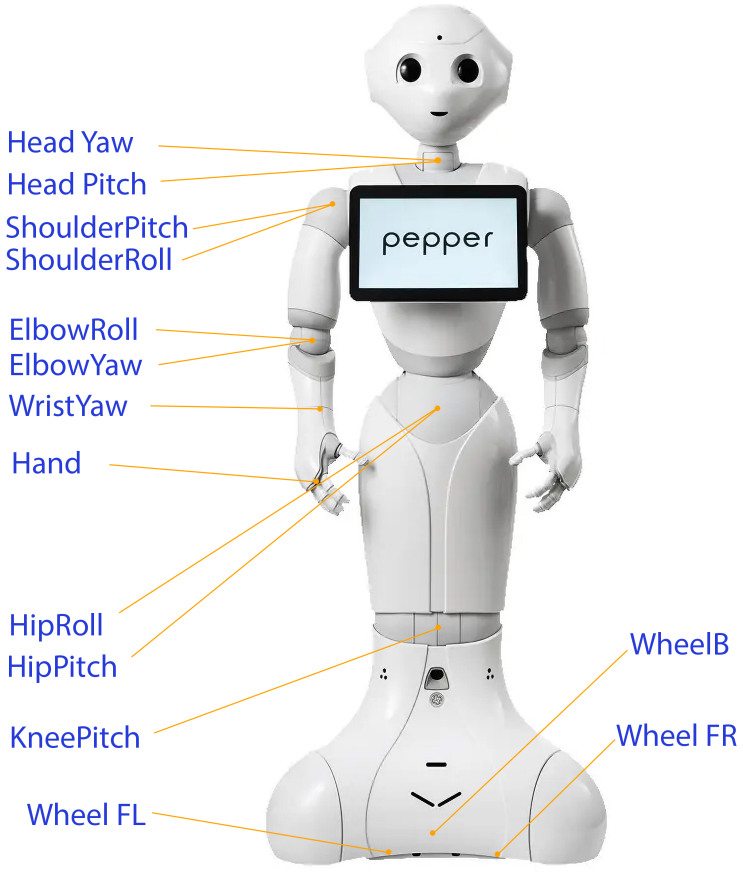

Figure 1: Pepper robot actuators

This deliverable consists of a comprehensive report divided into distinct sections that detail the software development lifecycle. These sections encompass the definition of requirements, aligning functional necessities with project goals, module specifications detailing the testing methodologies for joint and wheel actuators, and the design of interfaces, specifying data exchange via the ROS middleware and file inputs.

Additionally, the deliverable outlines the module's operation, governed by configurations detailed in `actuatorTestConfiguration.ini`, and delineates the structure for message data handling. Furthermore, a user manual for effective deployment and configuration, ensuring a smooth integration into the testing framework for both physical and simulated Pepper robot platforms. The coding section provides the implemented program code, adhering to standards specified in Deliverable D3.2.

2 Requirements Definition

The actuator test provides a systematic process aimed at understanding and documenting the functional and non-functional needs that the module must fulfill. This deliverable is important in identifying the specific user expectations, ensuring that the module will be capable of performing the precise actuator tests in various scenarios, including different operations and environments (physical robot and simulator). The test encompasses a thorough examination of the intended functionality, such as actuator identification, movement control, and the ability to conduct tests in sequence or parallel.

As seen from the diagram in Figure 1, the Pepper robot has 20 joint actuators and 3 wheel actuator. For joint actuators, the module must be able to move each joint to its minimum, maximum, and mid-range positions at specified average angular velocities. For wheel actuators, the module should conduct tests at designated positive and negative angular and linear velocities.

Misalignment of the Module

In addition to joint and wheel actuators, the robot features loudspeakers capable of generating speech from text. However, this phase does not encompass loudspeaker testing due to the absence of the required rostopic in the NAOqi DCM driver. Therefore, loudspeaker evaluation is incorporated into the `Sensor Test` deliverable.

While the physical robot is equipped with hand actuators, this feature is missing in the simulator. Consequently, hand actuator testing is exclusive to the physical robot.

Another noted limitation concerns the robot's performance during parallel actuator testing. The robot struggles with simultaneous activation of multiple actuators, particularly affecting performance when wheel actuators are tested. This issue necessitates conducting wheel actuator tests sequentially to prevent the delay encountered during parallel testing.

3 Module Specifications

Joint actuator

To validate the range of motion and control accuracy of each joint actuator by moving it through its minimum, mid-range, and maximum positions.

During the initialization phase, an average angular velocity ($\dot{\theta}$) is selected for the movement of each actuator. In the test execution phase, each joint is moved to its minimum position at the predetermined $\dot{\theta}$, and the duration required for this movement (Δt) is calculated using the formula $\Delta t = \Delta\theta/\dot{\theta}$, where $\Delta\theta$ represents the total change in the actuator's angle. This procedure is then systematically repeated to position the joint at its mid-range and maximum values. The testing procedure will have the following order: `home position` $\rightarrow$ `minimum position` $\rightarrow$ `maximum position` $\rightarrow$ `mid-range position` $\rightarrow$ `home position`. In the verification phase, it is crucial to confirm that each joint accurately reaches these designated positions.

Wheel actuator

To assess the wheel actuator's capability to execute controlled rotational and linear movements.

The test procedures initiate with the evaluation of the robot's linear motion capabilities, subsequently moving on to its rotational dynamics. In the linear movement test, the robot moves at a specified positive linear velocity, for instance, 1 m/s, for a pre-determined time. This procedure is then mirrored using a negative linear velocity, such as -1 m/s, to assess the robot's proficiency in reverse movements. Following the linear evaluation, the rotational movement test is conducted. Here, the robot is commanded to rotate at a chosen positive angular velocity, such as 90°/s, for a fixed interval. The fidelity of this rotation is thoroughly inspected, with any deviations from expected behavior being noted. This is again performed with a negative angular velocity, for example, -90°/s, to confirm the robot's rotational consistency in both directions and to log any variations observed.

4 Interface Design

Source Code

The source code for conducting actuator tests is structured into two primary components: `actuatorTestApplication` and `actuatorTestImplementation`. The actuatorTestImplementation component encapsulates all the essential functionality required for executing comprehensive actuator tests. This includes all of the tests for the actuators that consist of the head, arms, hands, legs, and wheels. Similar to the sensor test, the actuator test is also equipped with the functionality to process various files critical for the testing process, such as configuration files, input files, and topic files.

On the other hand, the actuatorTestApplication invokes those functions for the testing process. It is tasked with the execution of functions defined within the actuatorTestImplementation, effectively managing the actuator test operations.

Here is the file structure of the pepper interface tests package:

```
pepper_interface_tests
├── config
│   ├── actuatorTestConfiguration.ini
│   ├── actuatorTestInput.ini
│   ├── sensorTestInput.ini
│   └── sensorTestConfiguration.ini
├── data
│   ├── pepperTopics.dat
│   ├── sensorTestOutput.dat
│   └── simulatorTopics.dat
├── include
│   └── pepper_interface_tests
│       ├── actuatorTestInterface.h
│       └── sensorTestInterface.h
├── launch
│   ├── actuatorTestLaunchRobot.launch
│   ├── sensorTestLaunchRobot.launch
│   └── interfaceTestLaunchSimulator.launch
├── src
│   ├── actuatorTestApplication.cpp
│   ├── actuatorTestImplementation.cpp
│   ├── sensorTestApplication.cpp
│   └── sensorTestImplementation.cpp
├── README.md
├── CMakeLists.txt
└── package.xml
```

Configuration File

The operation of the actuatorTest node is determined by the contents of the configuration file that contains a list of key-value pairs as shown below.

The configuration file is named `actuatorTestConfiguration.ini`

Table 1: Configuration file for the actuator test.

Key	Value	Description
`platform`	`simulator` or `robot`	Specifies the platform to be tested. The platform can be set to either `simulator` or `robot`.
`robotTopics`	`robotTopics.dat`	Specifies the name of the robot topics file. The robot topics file contains the list of topics for the robot.
`simulatorTopics`	`simulatorTopics.dat`	Specifies the name to the simulator topics file. The simulator topics file contains the list of topics for the simulator.
`mode`	`parallel` or `sequential`	Specifies the mode of the test. The mode can be either `parallel` or `sequential`. The parallel mode runs all the tests in parallel. The sequential mode runs all the tests sequentially.

Input File

The input file is used to specify which actuators to test by using the actuator name as the key and `True` or `False` as the value. The actuator name must be the same as the actuator name in the topics file.

The input file is named `actuatorTestInput.ini`

Output Data File

There is no output data file for the actuator test. The result of the test is printed on the terminal.

Topics File

For the test, a selected list of the topics for the robot and simulator is stored in the topics file. The topic files are written in the .dat file format. The data file is written in key-value pairs where the key is the actuator name and the value is the topic

The topics file for the robot is named `robotTopics.dat` and the topics file for the simulator is named `simulatorTopics.dat`.

Topics Subscribed

There is no topics subscribed by the actuatorTest node.

Topics Published

The actuatorTest node publishes the following topics

Table 2: Topics Published by the actuatorTest node.

Topic	Actuator	Platform
/pepper_dcm/Head_controller/follow_joint_trajectory	Head	robot
/pepper_dcm/LeftArm_controller/follow_joint_trajectory	Left Arm	robot
/pepper_dcm/RightArm_controller/follow_joint_trajectory	Right Arm	robot
/pepper_dcm/LeftHand_controller/follow_joint_trajectory	Left Hand	robot
/pepper_dcm/RightHand_controller/follow_joint_trajectory	Right Hand	robot
/pepper_dcm/Pelvis_controller/follow_joint_trajectory	Leg	robot
/cmd_vel	Wheels	robot
/pepper/Head_controller/follow_joint_trajectory	Head	simulator
/pepper/LeftArm_controller/follow_joint_trajectory	Left Arm	simulator
/pepper/RightArm_controller/follow_joint_trajectory	Right Arm	simulator
/pepper/LeftHand_controller/follow_joint_trajectory	Left Hand	simulator
/pepper/RightHand_controller/follow_joint_trajectory	Right Hand	simulator
/pepper/Pelvis_controller/follow_joint_trajectory	Leg	simulator
/pepper/cmd_vel	Wheels	simulator

Launch File

The `actuatorTestLaunchRobot.launch` launch file is used to launch the actuator test. It is used for the robot and `actuatorTestLaunchSimulator.launch` for the simulator. The robot has launch file has the following parameters:

- `robot_ip`: specifies the IP address of the robot.
- `roscore_ip`: specifies the IP address of the roscore.
- `network_interface`: specifies the network interface name.

A default value is provided for each parameter in the launch file. For the robot ip, the default value is `172.29.111.230` and for the network_interface, the default value is `eth0`. The roscore ip does not have a default value and must be provided when launching the file.

The diagram below shows the data flow of the actuator test. The actuator test node reads the configuration file, input file, and topics file. The robot or simulator receives the topics and executes the tests. The robot or simulator then publishes the results to the actuator test node.

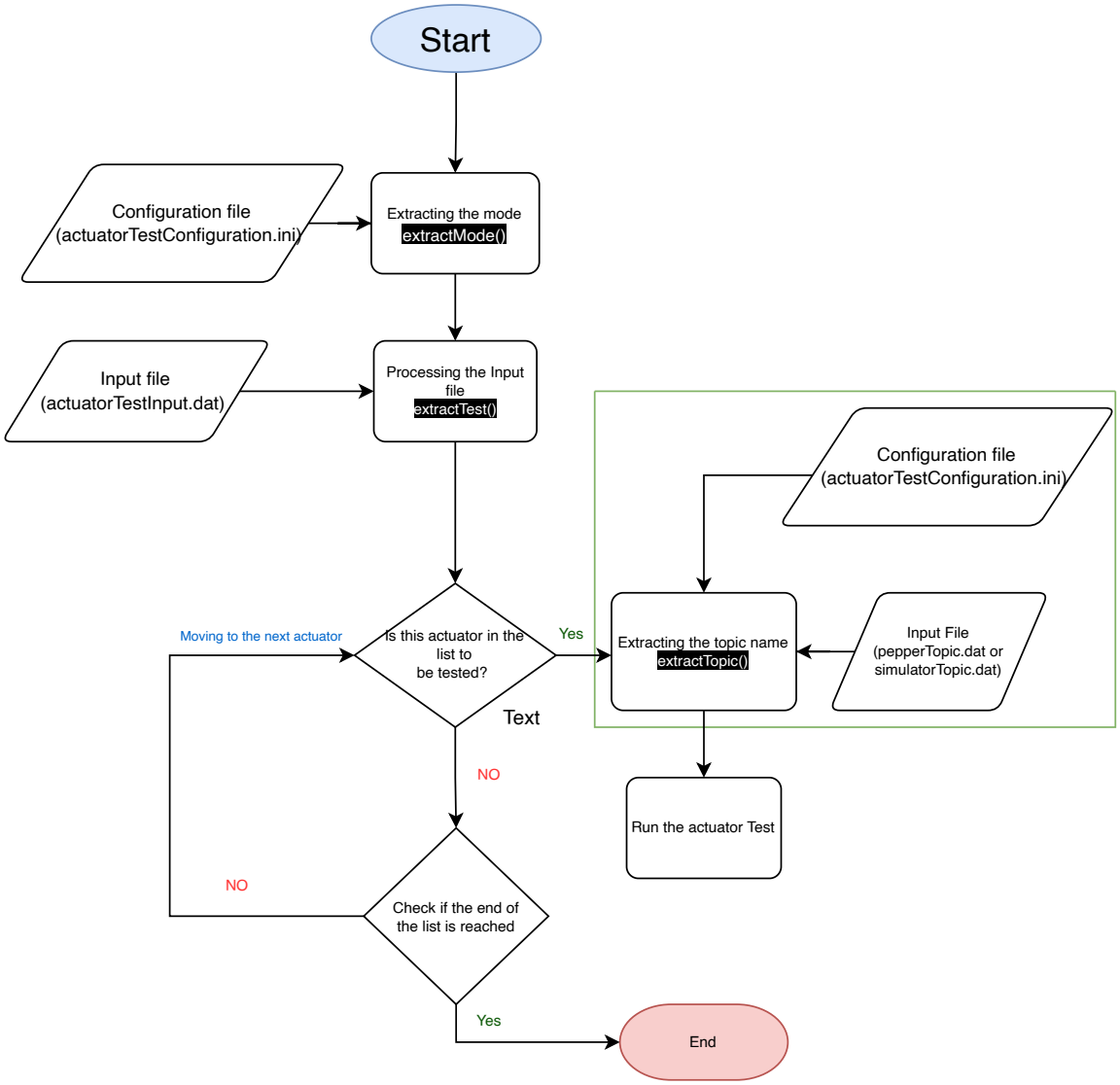

Figure 2: Data Flow of the Actuator Test

5 Module Design

For testing the physical robot, the NAOqi DCM (Device Communication Manager) driver is used to control the robot's actuators. The driver provides a hardware interface to connect to Alderban's robot Nao, Romeo, and Pepper robots. There are two ways to control the robot: DCM commands or ALMotion(default). DCM commands control the robot at a faster frequency but it sometimes results in the robot shaking. Hence, the test for the actuator is done using ALMotion.

The module is designed using to move the joint actuator of robot to a specified position by defining trajectory goal and sending it to a control server via a ROS(Robot Operating System) topic. First, the module initializes a client for interacting with ROS actions server. The function is structured attempt to attempt a connection to the server. If the connection to the server multiple times before giving up and throwing an error. The function `ControlClientPtr createClient(const std::string& topicName)` takes in the the topic name and returns actionClient pointer.

After the client is created, the module commands the robot to move to a specified position by defining a trajectory goal and sending it to the control server via ROS action client. First, the module will define a goal message for a joint trajectory action, which is part of the `control_msgs` package. The `follow_joint_trajectory` action is used to generate a more complex motion control mechanism, often used for executing predefined paths or trajectories for a set of joints.

The components of this topic include:

- `/goal`: used to send a "FollowJointTrajectoryGoal", which includes a trajectory comprising multiple points (position, velocities, acceleration, and/or efforts for each joint) and the time at which those points should be reached.
- `/cancel`: can cancel a currently executing trajectory.
- `/feedback`: provides real-time feedback about the current state of the trajectory execution.
- `/result`: provides the outcome of the trajectory execution after completion.
- `/status`: provides status information about the goal, such as if it's active, succeeded, or aborted

The trajectory goal will be send to the action server through the client. The server, presumably a part of a motion control system, will interpret this goal and command the robotic arm to move accordingly. The system then waits for a fixed duration for the movement to complete before proceeding.

The wheel test is done by publishing on the `cmd_vel` topic. The `cmd_vel` topic is used to control the robot's wheels. The message type used for the `cmd_vel` topic is `geometry_msgs/Twist`. The `Twist` message type is used to send velocity commands to the robot. The `Twist` message contains two `Vector3` messages, one for linear velocity and one for angular velocity. The `Vector3` message type is used to send three-dimensional linear and angular velocity commands.

6 Executing the Actuator Test

To start the actuator test, the user must first install the necessary software packages as outlined in Deliverable D3.3. Referring to the interface section, the user must set the platform to be tested, and specify which mode to run the test. Then using the key-value pairs, the users must specify which actuators to test by using the actuator name as the key and `True` or `False` as the value. To launch the actuator, the user must run the following command:

```
# Launch the Actuator Test for the physical robot
roslaunch pepper_interface_tests actuatorTestLaunchRobot.launch \
robot_ip:=<robot_ip> roscore_ip:=<roscore_ip> \
network_interface:=<network_interface>
```

```
# Launch the Actuator Test for the simulator
roslaunch pepper_interface_tests actuatorTestLaunchSimulator.launch
```

The above command will launch the actuator test for the robot and simulator respectively. After launching the actuators test, the user will run the tests by using the following command:

```
rosrun pepper_interface_tests actuatorTest
```

Note:

A noted limitation occurs when operating the robot in parallel mode: the robot's joint actuator cannot function concurrently while the robot is moving. Specifically, when the robot is executing movement commands via the `cmd_vel` topic, the joint actuator cannot be engaged simultaneously. To prevent any operational conflicts, it is essential to conduct a few test on the joint actuator only when the robot remains stationary.

Some technical details of the joints will be in the sections below. Refer to the Pepper Technical Documentation for more information on the joints.

6.1 Head Actuator Test

As shown in Figure 3 there are two joints in the head of the Pepper robot which are the `HeadYaw` and `HeadPitch`. The head Yaw joint is responsible for the left and right movement of the head while the head pitch joint is responsible for the up and down movement of the head. The test is conducted by moving both joints in the following sequence: `Home pos` → `Minimum pos` → `Maximum pos` → `Mid-range pos` → `Home pos`. The terminal output will show the result of the test. The head joint range is shown in the table 3.

Table 3: Head Joint Range. [1]

Joint	Min (°)	Max (°)	Min (rad)	Max (rad)	Home (rad)
`HeadYaw`	-119.5	119.5	-2.0857	2.0857	-0.2
`HeadPitch`	-40.5	25.5	-0.7068	0.4451	0.01

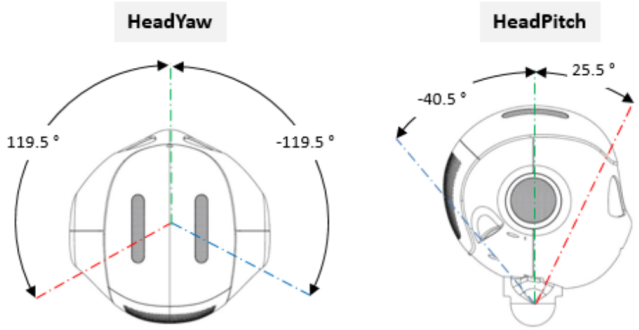

Figure 3: Pepper Head Joint [1]

The rostopic `/pepper_dcm/Head_controller/follow_joint_trajectory` moves the head joint.

Due to the collision with the robot's body, some head joint positions are not reachable. So when specifying the position, the user must be mindful of the limits especially when planning to move the head in a certain trajectory. The table 4 shows the limits of the head joint.

Table 4: Head joint limits. [1]

HeadYaw (°)	**HeadPitch - (°)**	**HeadPitch + (°)**
-119.5	-35.0	13.5
-91.4	-35.1	13.5
-61.6	-35.2	20.9
-33.33	-40.5	36.5
33.33	-40.5	36.5
61.6	-35.2	20.9
91.4	-35.1	13.5
119.5	-35.0	13.5

6.2 Arm Actuator Test

Left Arm

The left arm of the Pepper robot is articulated with five distinct joints: `LShoulderPitch`, `LShoulderRoll`, `LElbowYaw`, `LElbowRoll`, and `LWristYaw`. These joints are tested sequentially in the listed order to ensure proper functionality and range of motion. The testing involves moving each joint through a specific sequence of positions to verify the joint's operational range and functionality as follows:

`Home pos` → `Minimum pos` → `Maximum pos` → `Mid-range pos` → `Home pos`.

The rostopic `/pepper_dcm/LeftArm_controller/follow_joint_trajectory` is used to move the left arm joint.

Table 5 shows the range of the left arm joint.

Table 5: Left Arm Joint Range.[1]

Joint	MIn (°)	Max (°)	Min (rad)	Max (rad)	Home (rad)
LShoulderPitch	-119.5	119.5	-2.0857	2.0857	1.7625
LShoulderRoll	0.5	89.5	0.0087	1.5620	0.09970
LElbowYaw	-119.5	119.5	-2.0857	2.0857	-1.7150
LElbowRoll	-89.5	-0.5	-1.5620	-0.0087	-0.1334
LWristYaw	-104.5	104.5	-1.8239	1.8239	0.06592

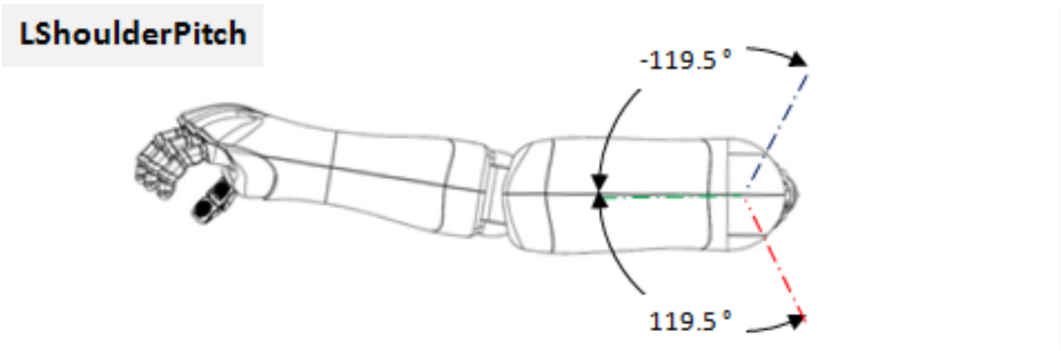

Figure 4: Left Shoulder Pitch Joint [1]

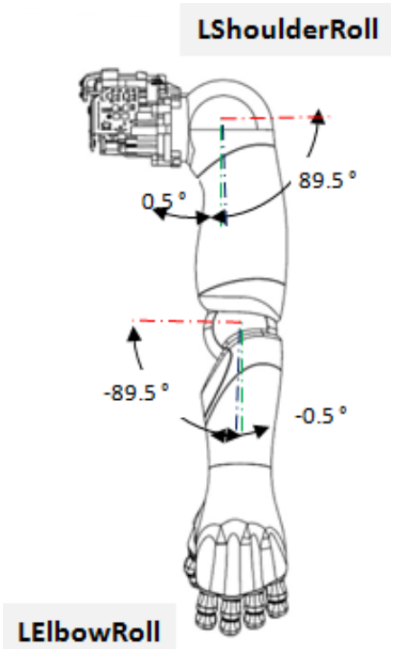

Figure 5: Left Shoulder Roll and Elbow Roll Joint [1]

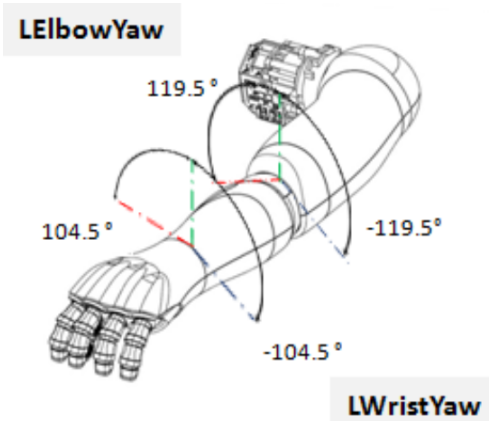

Figure 6: Left Elbow and Wrist Joint [1]

Due to collision, the LElbowRoll is limited. The table 6 shows the limits of the left Elbow Roll joint.

Table 6: Left Arm Joint Limits.[1]

LElbowYaw (°)	LElbowRoll Min (°)	LElbowRoll Max (°)
-119.5	-78.0	-0.5
-60.0	-78.0	-0.5
0.0	-89.5	-0.5
99.5	-89.5	-0.5
119.5	-83.0	-0.5

Right Arm

The right arm of the Pepper robot is articulated with five distinct joints: `RShoulderPitch`, `RShoulderRoll`, `RElbowYaw`, `RElbowRoll`, and `RWristYaw`. These joints are tested sequentially in the listed order to ensure proper functionality and range of motion. The testing involves moving each joint through a specific sequence of positions to verify the joint's operational range and functionality as follows:
`Home pos` → `Minimum pos` → `Maximum pos` → `Mid-range pos` → `Home pos`.

To make the movement symmetrical with the left arm, the `RShoulderPitch` joint is moved from the maximum position to the minimum position instead of the minimum position to the maximum position. While still having a home position as the starting and ending position. The terminal output will output each phase of the joint movement.

The rostopic `/pepper_dcm/RightArm_controller/follow_joint_trajectory` moves the right arm joint.

Table 7 shows the range of the right arm joint.

Table 7: Right Arm Joint Range.[1]

Joint	Min (°)	Max (°)	Min (rad)	Max (rad)	Home (rad)
`RShoulderPitch`	-119.5	119.5	-2.0857	2.0857	1.7410
`RShoulderRoll`	-89.5	-0.5	-1.5620	-0.0087	-0.09664
`RElbowYaw`	-119.5	119.5	-2.0857	2.0857	1.6981
`RElbowRoll`	0.5	89.5	0.0087	1.5620	0.09664
`RWristYaw`	-104.5	104.5	-1.8239	1.8239	-0.05679

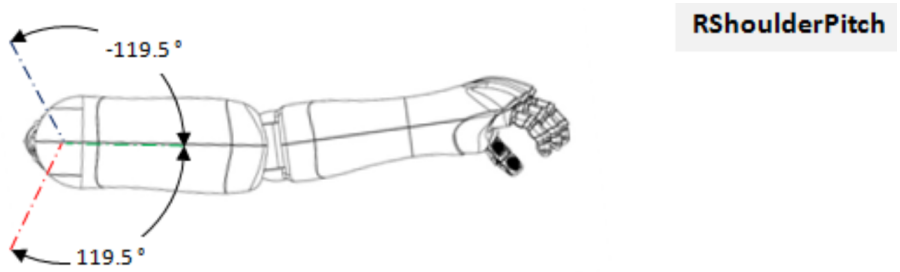

Figure 7: Right Shoulder Pitch Joint [1]

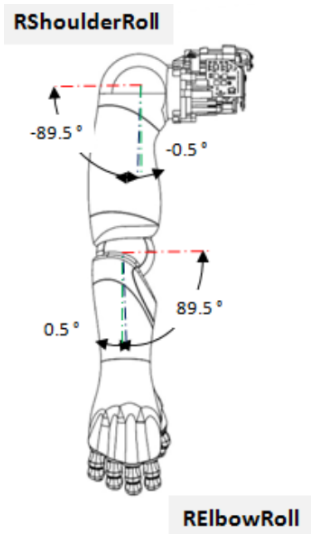

Figure 8: Right Shoulder and Elbow Roll Joint [1]

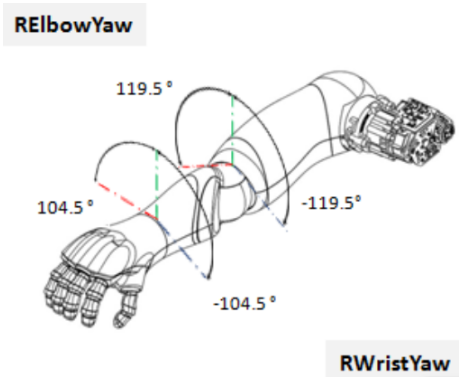

Figure 9: Right Elbow and Wrist Joint [1]

The joint limits for the RElbowRoll is shown in the table 8.

Table 8: Right arm joint limits. [1]

RElbowYaw (°)	RElbowRoll Min (°)	RElbowRoll Max (°)
-119.5	0.5	83.0
-99.5	0.5	89.5
0.0	0.5	89.5
60.0	0.5	78.0
119.5	0.5	78.0

6.3 Hand Actuator Test

Left Hand

The left hand of the pepper robot articulates a grasp opening and closing. To test the hand, the `LeftHand` joint is moved from the home position to the minimum position and then to the maximum position. Then back to its mid-range position. The terminal output will show the result of the test.

The rostopic `/pepper_dcm/LeftHand_controller/follow_joint_trajectory` is used to move the left-hand joint.

The range of the left-hand joint is range from `0.0 to 1.0`. Where `0.0` denotes the hand is fully `closed` and `1.0` denotes the hand is fully `open`.

Right Hand

The right hand of the pepper robot articulates a grasp opening and closing. To test the hand, the `RightHand` joint is moved from the home position to the minimum position and then to the maximum position. Then back to its mid-range position. The terminal output will show the result of the test.

The rostopic `/pepper_dcm/RightHand_controller/follow_joint_trajectory` moves the right hand joint. The range of the right-hand joint is range from `0.0 to 1.0`. Where `0.0` denotes the hand is fully `closed` and `1.0` denotes the hand is fully `open`.

6.4 Leg Actuator Test

The leg of the pepper robot is articulated using three joints:`HipRoll`,`HipPitch`, and `KneePitch`. These joints are tested sequentially in the listed order to ensure proper functionality and range of motion. The testing involves moving each joint through a specific sequence of positions to verify the joint's operational range and functionality as follows:
`Home pos` → `Minimum pos` → `Maximum pos` → `Mid-range pos` → `Home pos`.

The rostopic `/pepper_dcm/Pelvis_controller/follow_joint_trajectory` moves the leg joint.

Table 9 shows the range of the leg joint.

Table 9: Leg joint range. [1]

Joint	Min (°)	Max (°)	Min (rad)	Max (rad)	Home (rad)
`HipRoll`	-29.5	29.5	-0.5149	0.5149	-0.00766
`HipPitch`	-59.5	59.5	-1.0385	1.0385	-0.0107
`KneePitch`	-29.5	29.5	-0.5149	0.5149	0.03221

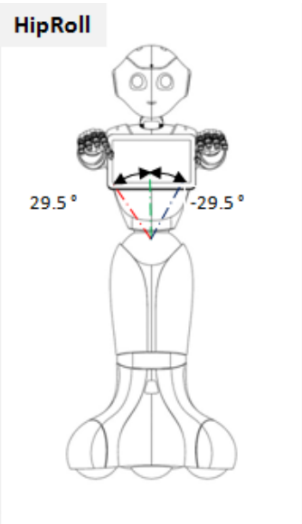

Figure 10: Hip Roll Joint [1]

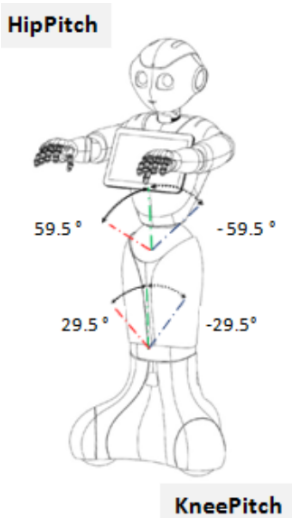

Figure 11: Hip Pitch Joint and Knee Pitch Joint [1]

6.5 Wheels Actuator Test

The pepper robot is equipped with three omnidirectional wheels as shown in Figure 12. The wheels are tested by moving the robot in a straight line 1 meter forward, and 1 meter backward. Then the robot is rotated 90 degrees to the left and 90 degrees to the right. The terminal output will show the result of the test.

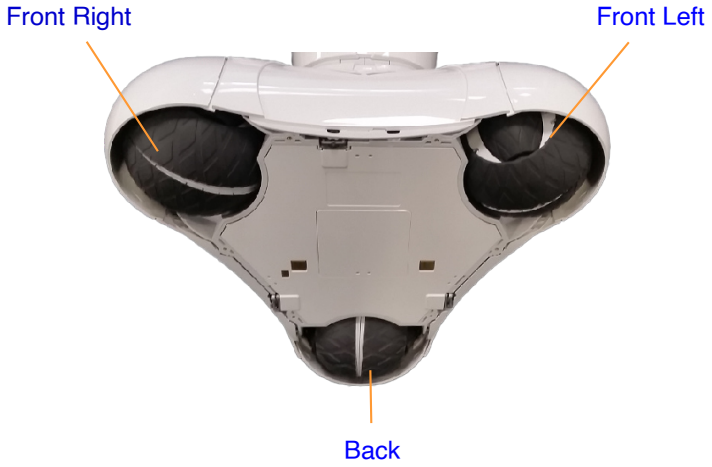

Figure 12: Pepper Wheels

Note: Before running the wheels, first move the robot out of the charging dock also close the power hatch before launching the actuator test. Furthermore, the user must ensure that the robot is in a safe environment to avoid collision with objects.

For testing the wheels, the rostopic `/cmd_vel` for the robot and `/pepper/cmd_vel` for the simulator is used to move the robot. The robot moves in the x-axis direction to move forward and

backward given the linear velocity and time duration. In addition, the robot rotates in the z-axis direction to rotate counterclockwise and clockwise given the angular velocity and time duration.

References

[1] Pepper joints. http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html.

Principal Contributors

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

Yohannes Haile, CMU-Africa
David Vernon, CMU-Africa

Document History

Version 1.0

First draft.
Yohannes Haile.
25 March 2024.