



NSWS SpiderPi Hexapod Robot Documentation

Written by: Teh Wei Sheng

Last Updated: 31/7/2024



Table of Contents

Chapter 1 Introduction.....	2
Chapter 2 Software Installations.....	3
2.1 Installing Ubuntu Linux - Jammy Jellyfish (22.04).....	3
2.2 Installing ROS 2 Humble Hawksbill.....	4
2.3 Installing Git.....	4
2.4 VNC Installations (Optional).....	5
Chapter 3 Controlling the SpiderPi bot.....	6
3.1 Connecting to the Spiderpi.....	6
3.2 ROS 2 Commands.....	6
3.3 Rteleop.py.....	7
Chapter 4 Adjusting the Parameters.....	9
4.1 Move.....	9
4.2 Stand.....	10
Chapter 5 Conclusions & Areas for Improvement.....	12
Chapter 6 Frequently Asked Questions (FAQs).....	13
Chapter 7 References.....	14
Chapter 8 NSW Journal.....	15

Chapter 1 | Introduction

This report outlines the control of the HiWonder SpiderPi hexapod robot using the “SpiderTeleop” ROS 2 package from [here](#). It includes a step-by-step guide for setting up the ROS 2 environment and using the package to operate the SpiderPi robot.

The ROS 2 package includes two main functions: ‘rteleop.py’ (remote control of the SpiderPi from a terminal) and ‘navigation_demo.py’ (demonstrating how various movement commands can be called in a script for the robot to perform specific tasks).

The HiWonder SpiderPi hexapod robot originally came with a phone application called “WonderPi,” available on both the App Store and Play Store. We discovered that the phone application utilizes JSON-RPC calls to the RPC server built into the SpiderPi robot to perform tasks. We adopted the same method for controlling the SpiderPi from a laptop terminal, sending JSON-RPC commands to the robot. This is done by connecting the laptop to the local network generated by the SpiderPi robot during startup.


The SpiderPi robot includes its own kinematics module for movements. Thus, in our JSON-RPC commands, we use the same structure as the phone application calls. Currently, we have identified three different action group commands: Move, Stand, and Transport, each with various adjustable parameters except Transport (explained in the sections below). These commands allow users to control the robot’s movements and perform specific tasks. Additionally, users can customize and create their own action groups.

Users are encouraged to install Ubuntu 22.04 (Jammy Jellyfish) and ROS2 Humble Hawksbill to run the ROS 2 package. Installation guides are provided in Chapter 2.

Chapter 2 | Software Installations

Installation Requirements:

- Operating System: Ubuntu Linux - Jammy Jellyfish (22.04)
- Robot Operating System (ROS): ROS 2 Humble Hawksbill
- Programming Language: Python 3.12
- Version Control System: Git
- Optional: VNC (Virtual Network Computing)

Note: Users are advised to read through the SpiderPi RPI 4B Version user documentation  SpiderPi (2024) provided by HiWonder before beginning the installations. The ROS 2 package has been tested and verified on the configuration listed above. Different operating systems, software, or hardware may require modifications.

2.1 Installing Ubuntu Linux - Jammy Jellyfish (22.04)

Ubuntu is a free and open-source Linux distribution based on Debian. If you are new to Ubuntu, here are some helpful links to get you started:

- [Ubuntu 22.04 LTS | Jammy Jellyfish - GeeksforGeeks](#)
- [Ubuntu 22.04 Guide - Linux Tutorials - Learn Linux Configuration](#)

Now, let's begin with the installation:

1. Ensure that your system meets the installation requirements mentioned below
 - a. RAM: 4 GB or more
 - b. Storage: 25 GB free hard disk space
 - c. Processor: Dual Core Processor (2 GHz)
 - d. Installation Media: Bootable Installation Media
 - e. Internet: Stable Internet Connection (Optional)
2. Follow the instructions in this blog post step by step to install Ubuntu 22.04 LTS (Jammy Jellyfish) on your laptop or PC.
 - a. [How to Download and Install Ubuntu 22.04 \[Step by Step\] - Linux Genie](#)

2.2 Installing ROS 2 Humble Hawksbill

The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications. We use ROS 2 Humble for this package as it is a relatively new release as of July 2024 and more stable than the latest release, Jazzy Jalisco.

Now, let's begin with the installation:

1. Ensure that you have correctly installed Ubuntu 22.04 LTS (Jammy Jellyfish) before beginning the installation of ROS 2 Humble.
2. Follow the instructions on this [page](#) up to the “Environment Setup” section to install ROS 2 Humble.
3. You can test the ROS 2 environment on your PC by trying the talker-listener examples in the “Try Some Examples” section to ensure ROS 2 is correctly installed.
4. Once ROS 2 Humble is correctly installed, proceed to configure the environment by following the instructions [here](#).

The complete documentation of ROS 2 Humble Hawksbill can be found [here](#).

2.3 Installing Git

To clone the repository and use the ROS 2 package, users will need to install Git by following the instructions below:

1. Open a terminal and run the following command to install Git:

```
sudo apt update
```

```
sudo apt install git
```
2. Ensure Git is installed correctly by checking its version:

```
git --version
```
3. Set your Git username and email address:

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your.email@example.com"
```
4. Use the git clone command to clone the spiderPi repository:

```
git clone https://github.com/edic-nus/spiderPI.git
```
5. After running the “git clone” command, a ‘spiderPI’ directory should be created in your current directory. List the contents of the current directory to see if the new directory is present:

```
ls
```

2.4 VNC Installations (Optional)

VNC (Virtual Network Computing) is a graphical desktop-sharing system for remotely controlling a Raspberry Pi, allowing users to access its desktop environment from another computer or mobile device. This enables easy management, troubleshooting, and use of applications without needing a physical monitor or keyboard connected to the Raspberry Pi.

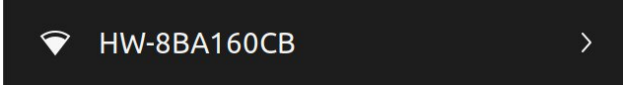
Users can follow the instructions provided [here](#) to install VNC and connect to the Raspberry Pi on the SpiderPi robot.

Chapter 3 | Controlling the SpiderPi bot

This chapter provides a step-by-step guide to using the ROS 2 package to control the SpiderPi hexapod robot's movements from a laptop terminal.

3.1 Connecting to the Spiderpi

Follow the instructions below:

1. Prepare the SpiderPi:
 - a. Ensure the SpiderPi is in an upright position.
 - b. Switch on the Raspberry Pi expansion board located on the back. The LED1 and LED2 on the Raspberry Pi will light up.
2. Wait for Initialization:
 - a. Wait approximately 30 seconds. The buzzer will emit a “beep” sound, and the robot will assume its initial standing posture.
 - b. LED1 on the expansion board at the back of the robot will remain on, while LED2 will blink once every second, indicating that the device has started up successfully.
3. Connect to Wi-Fi:
 - a. The Raspberry Pi will enter AP (Access Point) mode and create a Wi-Fi hotspot named with the prefix “HW”.
 - b. 
 - c. Connect your laptop to this network to control the SpiderPi.

3.2 ROS 2 Commands

Once the network connection is established, follow these instructions to use Rteleop:

1. Navigate into the SpiderTeleop directory:

```
cd ~/your_path/SpiderTeleop
```
2. Source the “rpc_commands” ROS 2 package. Use one of the following commands to source the package:

```
source install/local_setup.bash
source install/setup.bash
```

3. Verify the package is correctly sourced:

```
echo $AMENT_PREFIX_PATH
```

The result should give something similar to this:

```
wshengggg@wshengggg-Legion-Slim-7-16IRH8:~/spiderPI/SpiderTeleop$ source install/local_setup.bash
wshengggg@wshengggg-Legion-Slim-7-16IRH8:~/spiderPI/SpiderTeleop$ echo $AMENT_PREFIX_PATH
/home/wshengggg/spiderPI/SpiderTeleop/install/rpc_commands:/opt/ros/humble
wshengggg@wshengggg-Legion-Slim-7-16IRH8:~/spiderPI/SpiderTeleop$
```

4. Run 'rteleop.py':

```
ros2 run rpc_commands rteleop
```

You should see this as the result:

```
wshengggg@wshengggg-Legion-Slim-7-16IRH8:~/spiderPI/SpiderTeleop$ ros2 run rpc_commands rteleop
Press the corresponding key to control the spider robot:

Hexapod Mode:
  q w e
  a s d

Note: Switch to Quadrupled Mode (input 'S') before using Quadrupled Mode commands (u, i, o, j, k, l).

Quadrupled Mode:
  u i o
  j k l

Posture Controls:
1: Stand Low
2: Stand Middle
3: Stand High
4: Stand Super High
5: Quadrupled Stand
6: Transport
7: Posture Detect
8: Run Action
9: Stop

```

5. Similarly, user can run the navigation_demo.py with:

```
ros2 run rpc_commands navigation_demo
```

Note that the SpiderPi will start moving as soon as the command is run.

To stop it, press “Ctrl” + “C” to cancel.

3.3 Rteleop.py

The 'rteleop.py' script calls the Python scripts in the same directory to send JSON-RPC commands to the RPC server on the SpiderPi. There is one Python script for each movement, for example, "moveforward.py" for the move forward function. Once the rteleop.py is run, users can start controlling the SpiderPi movement using the keyboard.

Here are detailed labels for the key functions:

- Use the following keys to control the hexapod mode:
 - Move Forward: **w**

- Move Left: **a**
- Move Backward: **s**
- Move Right: **d**
- Rotate Anticlockwise: **q**
- Rotate Clockwise: **e**
- Switch to quadrupled mode (input **5**) before using quadrupled mode commands:
 - Quadrupled Move Forward: **i**
 - Quadrupled Move Left: **j**
 - Quadrupled Move Backward: **k**
 - Quadrupled Move Right: **l**
 - Quadrupled Rotate Anticlockwise: **u**
 - Quadrupled Rotate Clockwise: **o**
- Posture controls:
 - Stand Low: **1**
 - Stand Middle: **2**
 - Stand High: **3**
 - Stand Super High: **4**
 - Quadrupled Stand: **5**
 - Transport: **6**
 - Posture Detect: **7**
 - Run Action: **8**
 - Stop: **9**

Important Note:

- By default, all movement commands (e.g., move left, right, forward, backward, rotate clockwise, anticlockwise) are set to move indefinitely. If you want to run another command, you must press **9** (stop) to stop the previous movement before pressing another movement key.
- If you do not wish for the movement commands to execute indefinitely, read the next chapter to learn how to adjust the parameters.

Chapter 4 | Adjusting the Parameters

This chapter provides details about the movement command parameters and serves as a guide for adjusting them. We have identified three action group commands: Move, Stand, and Transport. We will explain how to adjust the parameters for each command to suit different purposes.

4.1 Move

We will start with the "Move" action group command. This action group has five parameters: <mode>, <movement_direction>, <rotation>, <speed>, and <times>. We use this action group to create most of the movement commands like "Move Forward", "Rotate", and "Stop" by adjusting these parameters.

Here is an example code of the "moveforward.py" that utilizes the "Move" action group command:

```
1  import requests
2  import json
3
4  def send_rpc_request(method, params, rpc_id=5):
5      url = "http://192.168.149.1:9030"
6      headers = {"Content-Type": "application/json"}
7      payload = {
8          "method": "Move",
9          "params": [2, 0, 0, 200, 0],
10         #<mode>, <movement_direction>, <rotation>, <speed>, <times>
11         "jsonrpc" : "2.0",
12         "id": 5
13     }
14
15     response = requests.post(url, headers=headers, data=json.dumps(payload))
16     return response.json()
17
18     # Example usage
19     response = send_rpc_request("Move", [2, 0, 0, 200, 0], rpc_id=5)
20     print(response)
```

moveforward.py

We can adjust the parameters in the “params” section (line 9). Below are the details of the "Move" command parameters:

- **mode:** Determines the mode of movement.
 - 2 = hexapod mode
 - 4 = quadrupled mode
- **movement_direction:** Sets the direction of movement (e.g., forward, backward).
 - 0 = front
 - 90 = left
 - 180 = right
 - 270 = back
- **rotation:** Controls the rotation angle or direction.
 - < 0 = anticlockwise
 - 0 = no rotation
 - > 0 = clockwise

Note: The magnitude of the value denotes how much the servo motor rotates each time (in degrees).

- **speed:** Specifies the speed of the movement.
 - ranges from 50 to 250
- **times:** Indicates the number of repetitions.
 - 0 = continuous movement until the stop command is executed
 - Minimum number of times: 2

Users can edit or customize different movement scripts to carry out specific tasks. For example, to create a clockwise rotation movement of exactly 180 degrees, the parameters of the “Move” command can be set to:

- "params": [2, 0, 45, 200, 4] (Rotate clockwise 45 degrees 4 times)
- "params": [2, 0, 30, 200, 6] (Rotate clockwise 30 degrees 6 times)

4.2 Stand

The "Stand" action group command allows you to adjust the posture of the SpiderPi robot. This action group has three parameters: <Height>, <mode>, and <time>. These parameters can be adjusted to create various standing postures.

Here is an example code of the "standmiddle.py" that utilizes the "Stand" action group command:

```
1  import requests
2  import json
3
4  def send_rpc_request(method, params, rpc_id=13):
5      url = "http://192.168.149.1:9030"
6      headers = {"Content-Type": "application/json"}
7      payload = {
8          "method": "Stand",
9          "params": [100, 2, 1],
10         # <Height>, <mode>, <time>
11         "jsonrpc" : "2.0",
12         "id": 13
13     }
14
15     response = requests.post(url, headers=headers, data=json.dumps(payload))
16     return response.json()
17
18 # Example usage
19 response = send_rpc_request("Stand", [100, 2, 1], rpc_id=13)
20 print(response)
```

standmiddle.py

We can adjust the parameters in the “params” section (line 9). Below are the details of the "Stand" command parameters:

- **Height:** Determines the height of the stand.
 - 50 to 200 (50 being the lowest and 200 the highest)
- **mode:** Specifies the mode of standing.
 - 2 = hexapod mode
 - 4 = quadrupled mode
- **time:** Indicates the time duration to reach the specified height.
 - Time in seconds

Users can edit or customize different stand scripts to carry out specific postures. For example, to make the SpiderPi stand at a low height in quadrupled mode quickly, the parameters of the “Stand” command can be set to:

- **"params": [50, 4, 1]** (Stand at a height of 50 in quadrupled mode in 1 second)

Chapter 5 | Conclusions & Areas for Improvement

In this report, we have outlined the process of setting up and controlling the HiWonder SpiderPi hexapod robot using the “SpiderTeleop” ROS 2 package. This package allows users to send JSON-RPC commands from a laptop terminal to the SpiderPi robot. We covered the installation of necessary software, including Ubuntu 22.04, ROS 2 Humble Hawksbill, Python 3.12, and Git. Additionally, we provided step-by-step guidance on using the `rteleop.py` and `navigation_demo.py` scripts for remote control and scripted movements.

The detailed explanation of the action group commands—Move, Stand, and Transport—enables users to customize the robot's behavior for specific tasks. By adjusting the parameters of these commands, users can tailor the robot's movements and postures to suit their needs.

There are several areas for improvement that could further refine the "SpiderTeleop" ROS 2 package:

1. **Explore Additional Action Group Commands:** Look for any additional action group commands that can be used in addition to the three that are now known (Move, Stand, and Transport). This could unlock new functionalities and enhance the robot's versatility.
2. **Camera Data Streaming:** Develop a method to stream camera data from the SpiderPi to a laptop, similar to the phone application's capability. This would improve the user's ability to monitor and control the robot remotely.
3. **ROS 2 Network for Sensors and Actuators:** Build a ROS 2 network that integrates sensors such as the gyroscope, camera, and ultrasonic sensors installed on SpiderPi with the actuators (servo motors). This would allow users to carry out autonomous navigation on the SpiderPi robot using ROS 2.
4. **Custom Action Group Commands:** Utilize the SpiderPi PC software to create or customize new action group commands and implement them in the ROS 2 package. This customization would allow users to tailor the robot's actions to specific tasks and needs.

By addressing these areas for improvement, we can significantly enhance the functionality, usability, and reliability of the "SpiderTeleop" ROS 2 package.

Chapter 6 | Frequently Asked Questions (FAQs)

- Where can I find additional guidance for using the HiWonder SpiderPi robot?
 - ◆ You can find the user manual provided by HiWonder here: [SpiderPi \(2024\)](#)

- In rteleop mode, I pressed a movement command and the bot didn't change the movement although I pressed another movement command, why?
 - ◆ You need to press "9" (Stop) to halt the current movement first before pressing another movement key.

- How can I ensure that the movement commands do not run indefinitely?
 - ◆ To prevent movement commands from running indefinitely, you need to specify the <times> parameter in the movement command scripts. Setting it to a specific number will limit the repetitions. Note that the <times> parameter starts from “2” (the minimum number of repetitions is 2).

- When I run the rteleop or navigation_demo, there is an error of “rpc_commands package not found”, what should I do?
 - ◆ Make sure you have correctly sourced the ROS 2 package by following the commands from chapter 3.2.

- How do I customize new action group commands?
 - ◆ New action group commands can be customized using the SpiderPi PC software following this guide [5. SpiderPi PC Software&Programming](#) .

- When I run rteleop.py, I got this error: ModuleNotFoundError: No module named 'pynput'. What should I do to fix it?
 - ◆ Run this command in the terminal: “pip install pynput” to install the “pynput” library.

Chapter 7 | References

edic-nus. (2024). SpiderPi Repository. Available at: <https://github.com/edic-nus/spiderPI>

"HiWonder SpiderPi Robot User Manual." Available at:
https://drive.google.com/drive/folders/1xR_VRBoANUaOvXoV2JYvCHucj53v0oZq

S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, May 2022.
<https://www.science.org/doi/10.1126/scirobotics.abm6074>

"ROS 2 Documentation — ROS 2 Documentation: Humble documentation." Available at:
<https://docs.ros.org/en/humble/index.html>

"Ubuntu Linux - The Jammy Jellyfish (22.04 LTS)," *GeeksforGeeks*. Available at:
<https://www.geeksforgeeks.org/ubuntu-linux-the-jammy-jellyfish-22-04-lts/>

"How to Download and Install Ubuntu 22.04," *Linux Genie*. Available at:
<https://linuxgenie.net/how-to-download-and-install-ubuntu-22-04/#:~:text=Download%20the%20Ubuntu%2022.04.2%20LTS%20ISO%20file%20from,the%20bootable%20USB%3A%20Step%203%3A%20Install%20Ubuntu%2022.04>

Ubuntu 22.04 Guide. LinuxConfig.org. Available at: <https://linuxconfig.org/ubuntu-22-04-guide>

Chapter 8 | NSWS Journal

This chapter details my 7-week journey in my NSWS job, where I developed the ROS 2 Humble package for the HiWonder SpiderPi hexapod robot.

Week 1

- Set up ROS 2 Humble workspace on SpiderPi's Raspberry Pi.
- Created ROS 2 Humble package for development.
- Integrated HiWonder's source codes for SpiderPi into the ROS 2 package.
- Installed ROS 2 package dependencies, and set up 'package.xml' and 'setup.py'.

Week 2

- Started debugging problems with integrating all the source codes into ROS 2 Humble.
- Tried to debug the kinematics.so file version incompatibility issue (original kinematics.so file is 32-bit, but we are using a 64-bit Raspberry Pi to run ROS 2).
- Got a 64-bit kinematics.so file from HiWonder, but it was compiled for Python 3.11, which isn't compatible with our Python 3.10 environment (ROS 2 Humble's Python version).

Week 3

- Tried isolating the Python environments into Python 3.10 and Python 3.11, using data piping to call kinematics.so from the ROS 2 environment.
- Used sockets (1 server and 1 client) to connect the two Python environments and attempted to run the kinematics.so file.

Week 4

- Debugged the Python environment isolation and TCP sockets method; could import kinematics.so but had issues calling the functions in it.
- Discovered that the Python version mismatch problem was unsolvable because the new kinematics.so file from HiWonder isn't compatible with our SpiderPi robot (it seems they added new stuff).
- Couldn't get the correct version of kinematics.so file from HiWonder.

Week 5

- Abandoned the idea of developing ROS 2 Humble on the SpiderPi itself.
- Restarted the project using the RPC server method (sending commands from a remote terminal to the SpiderPi using JSON-RPC calls).
- Developed all movement commands using JSON-RPC calls.
- Created 'rteleop.py' for the SpiderPi.

Week 6

- Researched more commands for the package.
- Created 'navigation_demo.py' for the SpiderPi.
- Created a ROS 2 package and integrated the previous code into it.

Week 7

- Completed documentation for the ROS 2 package developed.
- Refined the movement commands and continued to explore more commands.