Teleoperation Works

- Installing ROSBRIDGE on Machine
- Creation of a React Based weB application on Remote Computer

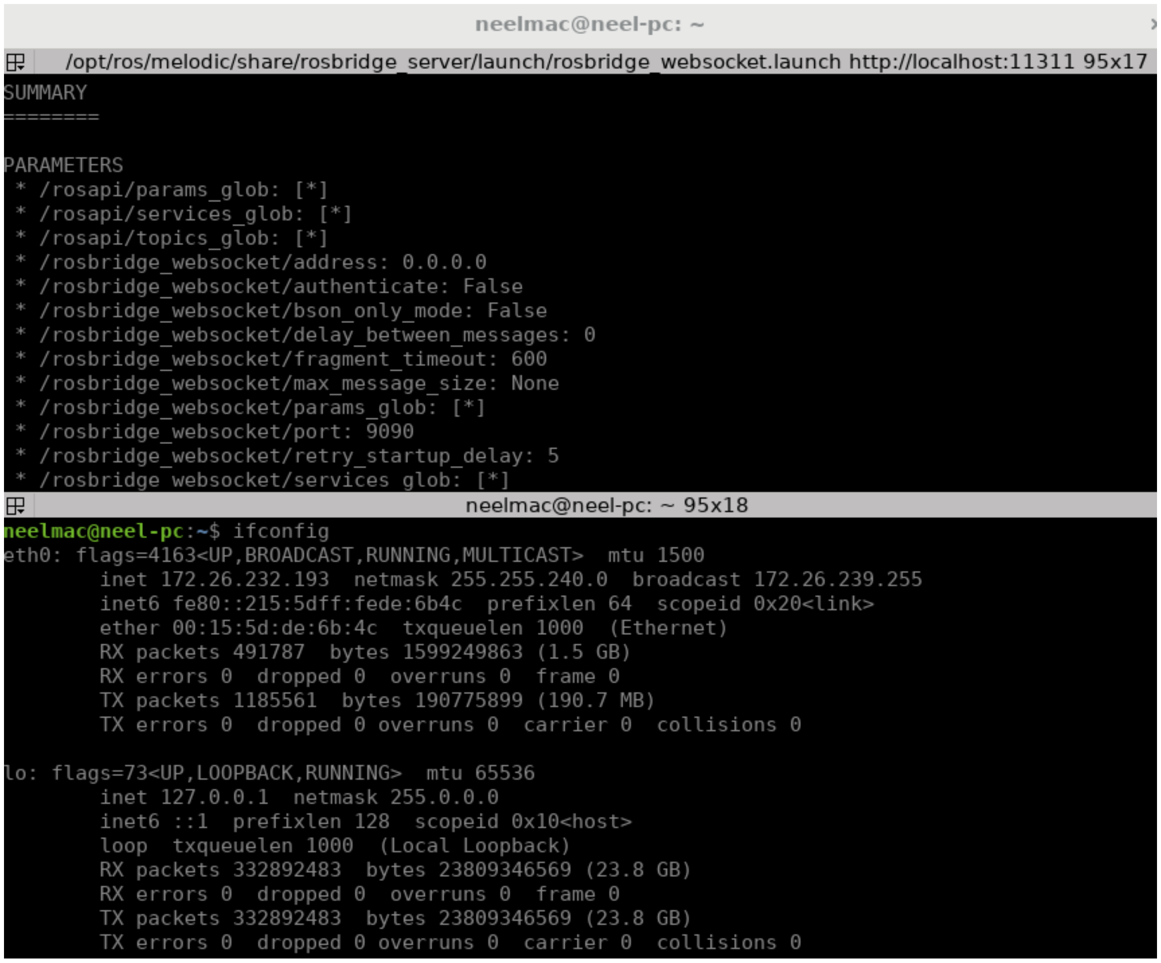
1. Installation of Rosbridge

   ```
   sudo apt-get install ros-melodic-rosbridge-server
   ```

   Here we are using melodic distro on Keylo

2. Running Rosbridge
   ```
   source /opt/ros/<rosdistro>/setup.bash
   ```
3. Start the ROSBRIDGE Server
   ```
   roslaunch rosbridge_server rosbridge_websocket.launch
   ```

4. This is how terminal looks like when the rosbridge is started
   - In the first terminal it shows the parameters of rosbridge running on server
   - In second terminal we ran `ifconfig` to check the ip of machine



```
neelmac@neel-pc: ~
/opt/ros/melodic/share/rosbridge_server/launch/rosbridge_websocket.launch http://localhost:11311 95x17
SUMMARY
========

PARAMETERS
 * /rosapi/params_glob: [*]
 * /rosapi/services_glob: [*]
 * /rosapi/topics_glob: [*]
 * /rosbridge_websocket/address: 0.0.0.0
 * /rosbridge_websocket/authenticate: False
 * /rosbridge_websocket/bson_only_mode: False
 * /rosbridge_websocket/delay_between_messages: 0
 * /rosbridge_websocket/fragment_timeout: 600
 * /rosbridge_websocket/max_message_size: None
 * /rosbridge_websocket/params_glob: [*]
 * /rosbridge_websocket/port: 9090
 * /rosbridge_websocket/retry_startup_delay: 5
 * /rosbridge_websocket/services_glob: [*]
neelmac@neel-pc: ~ 95x18
neelmac@neel-pc:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.26.232.193  netmask 255.255.240.0  broadcast 172.26.239.255
        inet6 fe80::215:5dff:fede:6b4c  prefixlen 64  scopeid 0x20<link>
        ether 00:15:5d:de:6b:4c  txqueuelen 1000  (Ethernet)
        RX packets 491787  bytes 1599249863 (1.5 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1185561  bytes 190775899 (190.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 332892483  bytes 23809346569 (23.8 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 332892483  bytes 23809346569 (23.8 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

React Application: Now in this part we are going to develop a web application to achieve teleoperation.

From the link given below you can download the node js application. This will allow us to use node package manager to get and use different libraries in our project.

https://nodejs.org/en/download

After installing the node application we can start using the node package manager (npm).

3. Create a React Application

Open terminal or command prompt on your machine. And navigate to your project directory.
And create your application directory.

```
npx create-react-app my-app
```

Navigate to the directory you have created

```
cd my-app
```

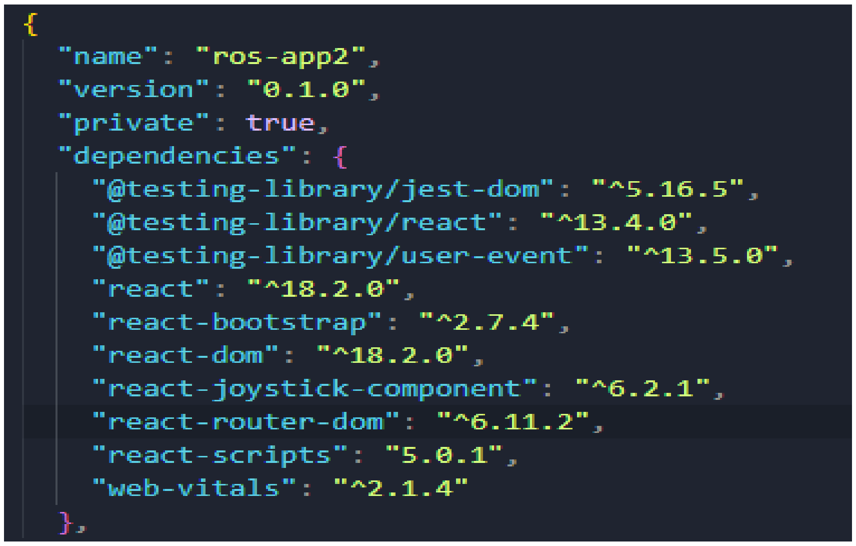
Now you can start the server by following command to your code editor's terminal or any other terminal

```
npm start
```

When your application is developed you can use following command and get deployment ready application

```
npm run build
```

We are not running any build command at this stage. Now navigate to the my-app directory and check the package.json. It has a record of installed libraries:

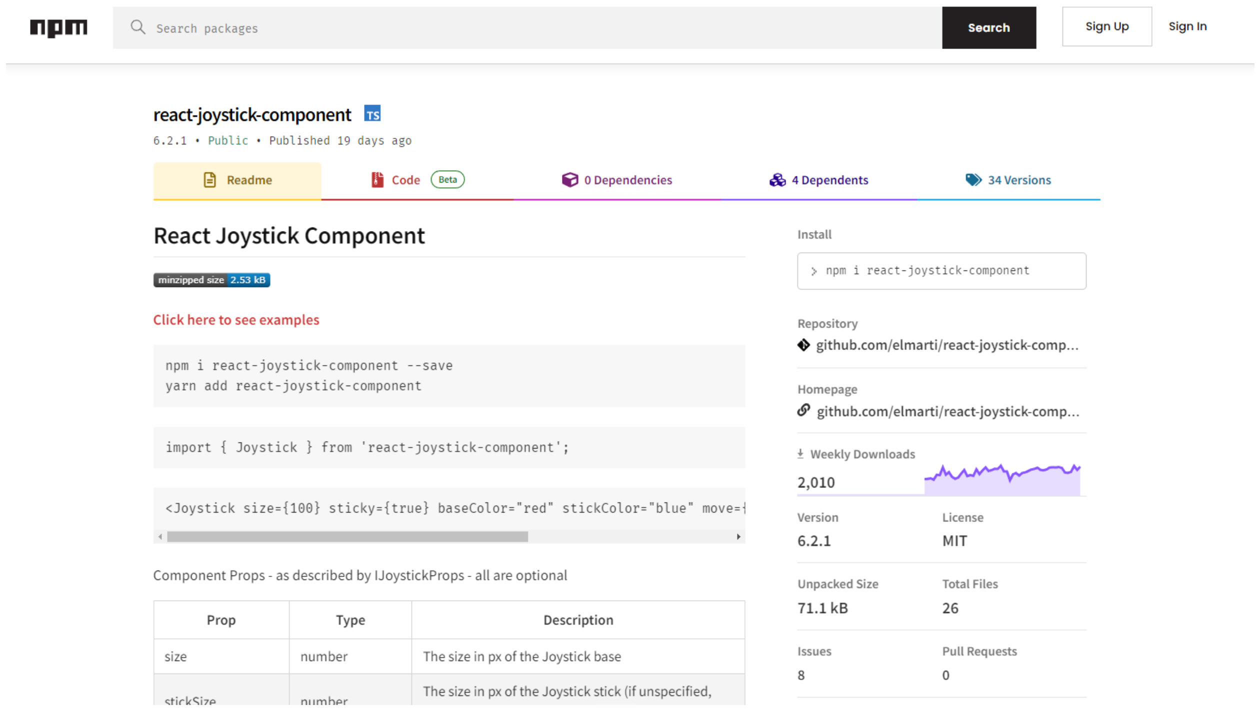


```
{
  "name": "ros-app2",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.7.4",
    "react-dom": "^18.2.0",
    "react-joystick-component": "^6.2.1",
    "react-router-dom": "^6.11.2",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
```

If you see this package of json most of the files and libraries are preinstalled when you build your application. However, you need to install additional libraries such as react-dom, react-joystick-component and react-router-dom.
You can install this libraries by npm install <library>.

For example, using a virtual joystick on our webapp's screen we will use react-joystick-component. We can visit their website or search those libraries on https://www.npmjs.com/

The joystick component is available at this site
https://www.npmjs.com/package/react-joystick-component



react-joystick-component TS
6.2.1 • Public • Published 19 days ago

React Joystick Component

minzipped size 2.53 kB

Click here to see examples

```
npm i react-joystick-component --save
yarn add react-joystick-component
```

```
import { Joystick } from 'react-joystick-component';
```

```
<Joystick size={100} sticky={true} baseColor="red" stickColor="blue" move={
```

Component Props - as described by IJoystickProps - all are optional

Prop	Type	Description
size	number	The size in px of the Joystick base
stickSize	number	The size in px of the Joystick stick (if unspecified,

Install
```
> npm i react-joystick-component
```

Repository
github.com/elmarti/react-joystick-comp...

Homepage
github.com/elmarti/react-joystick-comp...

Weekly Downloads
2,010

Version	License
6.2.1	MIT
Unpacked Size	Total Files
71.1 kB	26
Issues	Pull Requests
8	0

That gives us information about how to install and its different parameters.

We will also need libraries such as ROSLIBJS that is available at the link below:
https://github.com/RobotWebTools/roslibjs/tree/develop/build

However, we can use roslib js from any reliable sources or npm install <library> also. But if the installed version doesn't work we have to uninstall that. And we can download the files from the link above. And keep those in the separate folder in the public directory of our project.

Now come to the src directory and you can create another subdirectory named components that will contain all the react components that we are going to use in our front end web app. Here we are writing a very basic hello world app in react js and after we can customize it for other applications.

React Components and Hello World App Tutorial
(This is a react tutorial if you haven’t used react before)

In this tutorial, we will explore React components and create a simple "Hello World" application using React. React is a JavaScript library used for building user interfaces, and it follows a component-based architecture.

Prerequisites

Before getting started, make sure you have the following prerequisites:
- Basic knowledge of HTML, CSS, and JavaScript
- Node.js and npm (Node Package Manager) installed on your system

(Note that we have already accomplish this step)
Step 1: Setting Up the Project

1. Create a new directory for your project and navigate into it using the command line.
2. Initialize a new Node.js project by running the command:

   ```
   npm init -y
   ```
   This will create a `package.json` file for your project.
3. Install React and React DOM by running the following command:

   ```
   npm install react react-dom
   ```

Step 2: Creating a React Component
1. Inside your project directory, create a new file called `App.js`.
2. Open `App.js` in a text editor and add the following code:

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Hello, World!</h1>
      </div>
    );
  }
}

export default App;
```

Step 3: Setting Up the HTML File

1. Create a new file called `index.html` in your project directory.
2. Open `index.html` in a text editor and add the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>React Hello World App</title>
  </head>
  <body>
    <div id="root"></div>
    <script src="dist/bundle.js"></script>
  </body>
</html>
```

Step 4: Creating the Entry Point

1. Create a new file called `index.js` in your project directory.
2. Open `index.js` in a text editor and add the following code:

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));
```

Step 5: Building the React App

1. Open the command line and navigate to your project directory.
2. Run the following command to build your React app:

```
npx babel src --out-dir dist --presets react-app/prod
```

This command transpiles the JSX syntax used in your React components into plain JavaScript code.

Step 6: Running the App
1. Open the command line and navigate to your project directory.
2. Start a local development server by running the following command:

```
npx serve
```

3. Open your web browser and visit http://localhost:5000. You should see the "Hello, World!" message displayed on the page.

Here you have developed a hello world app on react. However, you aren't required to do steps 5 and 6 while you are still developing. You can use npm start command in project directory and you can start this application on http://localhost:3000

Since we have developed a hello world application, We can use different components such as connection, Home, Teleoperation, RobotState and Map to add different features to our application.
I have posted that repository on my github account. You can download the files and build the project on top of that.

The Version I have developed is fully compatible with TurtleSim and Gazebo Simulator:

For your personal use you can tell operate a robot through the following procedure:

1. Start Rosbridge on Robot
    a. Check IP of Robot
    b. Check Rosbridge Port Number

2. Make sure both robot and machine are connected on the same network.
    a. I have used WSL for ROS and a Windows PC for Web Application that was Hosting WSL

    (You can skip step 2 Since I have already)
3. Download the Repository I have mentioned
    a. Download ROSLIBJS from the link I have mentioned earlier and put that in the Subdirectory inside the Public folder and Import to your application files and component files
4. Go to Config.js in scripts folder and put your robot’s IP address and port address
5. Congratulations! Your ROS TELEOPERATION IS WORKING!