

# Dofbot\_Pro Jetson-Nano Version Startup Instructions

From 【5. 2D Visual Tracking Course】 to 【14. Voice Control for 3D Grasping Course (Sorting & Tracking)】 can run on the main board, **the main board runs on ubuntu18.04+ros1-melodic environment**. If you need to learn the ROS2 version feature code and run AI large model programs, you need to enter the docker container (**container environment is Ubuntu22.04+ros2-humble**) to run them. For first-time entry into the docker container, you can use the following command:

```
sh Docker_Dofbot_Pro-nano.sh
```

```
-----  
MY_IP: 192.168.8.88  
ROS_MASTER_URI: http://192.168.8.88:11311  
-----  
jetson@yahboom:~$ sh Docker_Dofbot_Pro-nano.sh  
Docker service has been started  
access control disabled, clients can connect from any host  
-----  
ROS_DOMAIN_ID: 62 | ROS:  
-----  
root@yahboom:/# [ ]
```

All subsequent commands in the ROS2 version tutorials need to be entered in the terminal of this container. After entering docker, the system will generate a container ID. You can check the container ID using the following command. Open a terminal on the main board and input:

```
docker ps
```

```
jetson@yahboom: ~  
-----  
MY_IP: 192.168.8.88  
ROS_MASTER_URI: http://192.168.8.88:11311  
-----  
jetson@yahboom:~$ docker ps -a  
CONTAINER ID        IMAGE               COMMAND      CREATED          NAMES  
STATUS              PORTS              NAMES  
9ca3b86aac13      192.168.2.51:5000/nano-ai:1.4.0   "/bin/bash"   16 seconds ago
```

Based on the image name and startup time, you can find the container ID. Use this ID to enter the same container each time. Open a terminal on the main board and enter the following command to enter the container:

```
#Replace the container ID with the queried ID  
docker exec -it containerID /bin/bash
```

The screenshot shows a terminal window with the following session:

```
jetson@yahboom:~  
MY_IP: 192.168.8.88  
ROS_MASTER_URI: http://192.168.8.88:11311  
-----  
jetson@yahboom:~$ docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED NAMES  
STATUS PORTS  
9ca3b86aac13 192.168.2.51:5000/nano-ai:1.4.0 "/bin/bash" 16 seconds ago  
-----  
root@yahboom:~  
----- ago  
MY_IP: 192.168.8.88 192.168.2.51:5000/nano-ai:1.3.2 "/bin/bash" 3 days ago  
ROS_MASTER_URI: http://192.168.8.88:11311 gracioso_joliot  
----- 2 days ago "/bin/bash" 3 days ago  
jetson@yahboom:~$ docker exec -it 9ca3b86aac13 /bin/bash relaxed_williamson  
----- /bin/bash" 3 days ago  
ROS_DOMAIN_ID: 62 | ROS: stoic_brown  
----- /bin/bash" 4 days ago  
cdroot@yahboom:/# cd ago agitated_chandrasek  
root@yahboom:~# source .bashrc  
----- /bin/bash" 4 days ago  
ROS_DOMAIN_ID: 62 | ROS: humble sad_bhabha  
----- /bin/bash" 6 days ago  
root@yahboom:~# jupyter-lab --allow-root exciting_hamilton
```

A red arrow points from the command `docker exec -it 9ca3b86aac13 /bin/bash` to the container ID `9ca3b86aac13` in the `docker ps -a` output.

After entering the container, we are not in the `/root` directory. You need to execute the following commands to enter the `/root` directory and refresh the environment variables to be able to find the functions. Enter these commands in the docker container terminal:

```
cd  
source .bashrc
```

If you want to view the code in the container, you need to enter the following command in the docker container terminal to start jupyter:

```
jupyter-lab --allow-root
```

```
ROS_DOMAIN_ID: 62 | ROS: humble          gifted_meninsky
root@yahboom:~# jupyter-lab --allow-root
[1 2025-11-21 18:40:24.186 ServerApp] jupyter_lsp | extension was successfully linked.
[1 2025-11-21 18:40:24.203 ServerApp] jupyter_server_terminals | extension was successfully linked.
[1 2025-11-21 18:40:24.229 ServerApp] jupyterlab | extension was successfully linked.
[1 2025-11-21 18:40:24.249 ServerApp] notebook | extension was successfully linked.
[1 2025-11-21 18:40:26.000 ServerApp] notebook_shim | extension was successfully linked.
[1 2025-11-21 18:40:26.137 ServerApp] notebook_shim | extension was successfully loaded.
[1 2025-11-21 18:40:26.147 ServerApp] jupyter_lsp | extension was successfully loaded.
[1 2025-11-21 18:40:26.152 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[1 2025-11-21 18:40:26.179 LabApp] JupyterLab extension loaded from /root/.local/lib/python3.10/site-packages/jupyterlab
[1 2025-11-21 18:40:26.179 LabApp] JupyterLab application directory is /root/.local/share/jupyter/lab
[1 2025-11-21 18:40:26.181 LabApp] Extension Manager is 'pypi'.
[1 2025-11-21 18:40:26.606 ServerApp] jupyterlab | extension was successfully loaded.
[1 2025-11-21 18:40:26.621 ServerApp] notebook | extension was successfully loaded.
[1 2025-11-21 18:40:26.625 ServerApp] Serving notebooks from local directory: /root
[1 2025-11-21 18:40:26.625 ServerApp] Jupyter Server 2.17.0 is running at:
[1 2025-11-21 18:40:26.625 ServerApp] http://yahboom:9999/lab
[1 2025-11-21 18:40:26.625 ServerApp] http://127.0.0.1:9999/lab
[1 2025-11-21 18:40:26.625 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[1 2025-11-21 18:40:26.809 ServerApp] Skipped non-installed server(s): basedpyright, bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyrefly, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
```

Then, open a browser and enter the main board's IP address + :9999 in the browser's address bar. For example, if my main board's IP address is 192.168.8.88, then enter **192.168.8.88:9999** in the address bar and press Enter.



Enter password: **yahboom**, click **【Log in】** to enter the container to view the code.

```

1 from launch import LaunchDescription
2 from launch.actions import DeclareLaunchArgument
3 from launch.substitutions import LaunchConfiguration
4 from launch_ros.actions import PushRosNamespace
5 from launch.actions import GroupAction
6 from launch_ros.actions import ComposableNodeContainer
7 from launch_ros.descriptions import ComposableNode
8 from launch_ros.actions import Node
9 import os
10
11
12 def generate_launch_description():
13     # Declare arguments
14     args = [
15         DeclareLaunchArgument('camera_name', default_value='camera'),
16         DeclareLaunchArgument('depth_registration', default_value='false'),
17         DeclareLaunchArgument('serial_number', default_value=''),
18         DeclareLaunchArgument('usb_port', default_value=''),
19         DeclareLaunchArgument('device_num', default_value='1'),
20         DeclareLaunchArgument('vendor_id', default_value='0x2bc5'),
21         DeclareLaunchArgument('product_id', default_value=''),
22         DeclareLaunchArgument('enable_point_cloud', default_value='true'),
23         DeclareLaunchArgument('enable_colored_point_cloud', default_value='true'),
24         DeclareLaunchArgument('cloud_frame_id', default_value=''),
25         DeclareLaunchArgument('point_cloud_qos', default_value='default'),
26         DeclareLaunchArgument('connection_delay', default_value='100'),
27         DeclareLaunchArgument('color_width', default_value='640'),
28         DeclareLaunchArgument('color_height', default_value='480'),
29         DeclareLaunchArgument('color_fps', default_value='15'),
30         DeclareLaunchArgument('color_format', default_value='MJPG'),
31         DeclareLaunchArgument('enable_color', default_value='true'),
32         DeclareLaunchArgument('flip_color', default_value='false'),
33         DeclareLaunchArgument('color_qos', default_value='default'),
34         DeclareLaunchArgument('color_camera_info_qos', default_value='default'),
35         DeclareLaunchArgument('enable_color_auto_exposure', default_value='false'),
36         DeclareLaunchArgument('color_exposure', default_value='-1'),
37         DeclareLaunchArgument('color_gain', default_value='1'),
38         DeclareLaunchArgument('enable_color_auto_white_balance', default_value='true'),
39         DeclareLaunchArgument('color_white_balance', default_value='1'),
40         DeclareLaunchArgument('color_ae_max_exposure', default_value='1'),
41         DeclareLaunchArgument('color_brightness', default_value='1'),
42         DeclareLaunchArgument('color_sharpness', default_value='1'),
43         DeclareLaunchArgument('color_saturation', default_value='1'),
44         DeclareLaunchArgument('color_contrast', default_value='1'),
45         DeclareLaunchArgument('color_gamma', default_value='1'),
46         DeclareLaunchArgument('color_hue', default_value='1'),
47         DeclareLaunchArgument('depth_width', default_value='640'),
48         DeclareLaunchArgument('depth_height', default_value='400'),
49         DeclareLaunchArgument('depth_fps', default_value='10'),
50         DeclareLaunchArgument('depth_format', default_value='Y11'),
51         DeclareLaunchArgument('enable_depth', default_value='true'),

```

If you shut down and restart, and need to re-enter the previous container, you need to restart the container first, then enter it. Enter the following command to restart the closed container:

```
#Replace the container ID here with your own container ID, you can refer to above
and get it through docker ps -a
docker restart containerID
```

After restarting the container, you can enter the container using the following command:

```
#Replace the container ID with the queried ID
docker exec -it containerID /bin/bash
```