

# ROS1 Use Cases

Aurora is a newly developed sensor for integrated positioning and mapping created by SLAMTEC that integrates laser, vision, inertial navigation and deep learning technologies. The sensor does not require external dependencies and can provide six-degree-of-freedom positioning capabilities for indoor and outdoor three-dimensional high-precision mapping machines as soon as it is turned on.

## 1. Development environment requirements

Based on the Ubuntu 20.04 operating system, with ROS1 software package and compiler Gcc9.

## 2. Download radar SDK

Download link: [SLAMTEC Product Documents Download and Technical Support](#)

Two versions of the SDK have been placed in the attachment. Use the corresponding version according to the gcc version of your environment.



Aurora ROS1 SDK contains resources and codes that you may use during development. Its directory structure is organized as follows:

Directory	Description
docs	Reference Documents
src	Source Code
--slamware_ros_sdk	ROS SDK Source Code Package
--slamware_sdk	SDK Related Header Files and Library Files
--aurora_remote_public	Aurora Related Header Files and Library Files

Terminal input, the following content can be used to view the gcc version

```
gcc -v
```

```

yahboom@yahboom-virtual-machine:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/9/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:hsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 9.4.0-1ubuntu1~20.04.2' --with-bugurl=file:///usr/share/doc/gcc-9/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,gm2 --prefix=/usr --with-gcc-major-version-only --program-suffix=-9 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --libdir=/usr/lib --enable-nls --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-plugin --enable-default-pie --with-system-zlib --with-target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --disable-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none=/build/gcc-9-9QD0t0/gcc-9-9.4.0/debian/tmp-nvptx/usr,hsa --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.2)
yahboom@yahboom-virtual-machine:~$

```

### 3. Hardware requirements

To use ROS1 SDK, you need an Auraro-based spatial mapping device, turn it on and configure the appropriate IP address, and the virtual machine needs to connect to the hotspot of the aurora radar. The slamware\_ros\_sdk\_server\_node node will try to connect to the device after it is started.

## 4. Environment setup

### 4.1. Create a workspace first

```

mkdir -p ~/ros_ws/src
cd ~/ros_ws/src

```

Put the sdk source code you just downloaded in the src working directory where the source code is stored,

### 4.2 Compile

```

cd ..
catkin_make

```

It is normal to have warnings when compiling slamware\_sdk,

```

[13.419s] WARNING:colcon.colcon_cmake.task.cmake.build:Could not run installation step for package 'slamware_sdk' because it has no 'install' target

```

### 4.3 Configure the workspace system environment

```

source devel/setup.bash

```

In order to automatically load the environment every time the terminal is started: add the following command to ~/.bashrc

```

echo "source ~/ros_ws/devel/setup.bash" >> ~/.bashrc

```

## 4.4 Configure the system environment for dynamic libraries for image building

Since `aurora_remote_public_lib` is a dynamic library, you need to add the platform path to `LD_LIBRARY_PATH`. For example, if you place `slamware_ros1_sdk_linux-x86_64-gcc9` in the `~/ros_ws/sec` folder, you need to add the following command to `~/bashrc`

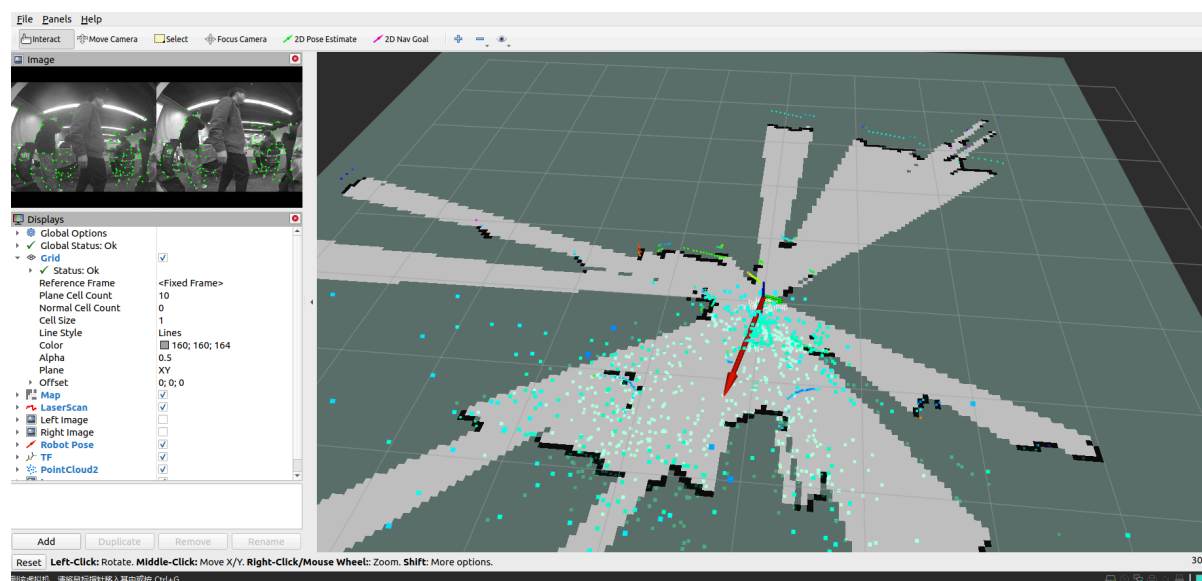
**Note: If you download the SDK for the aarch64 environment, the corresponding name should be changed to `slamware_ros1_sdk_linux-aarch64-gcc9`.**

```
export
LD_LIBRARY_PATH=~/.ros_ws/src/aurora_ros1_sdk_linux_x86_gcc9/src/aurora_remote_public/lib/linux_x86_64:$LD_LIBRARY_PATH
```

## 4.5 Start the graph building node

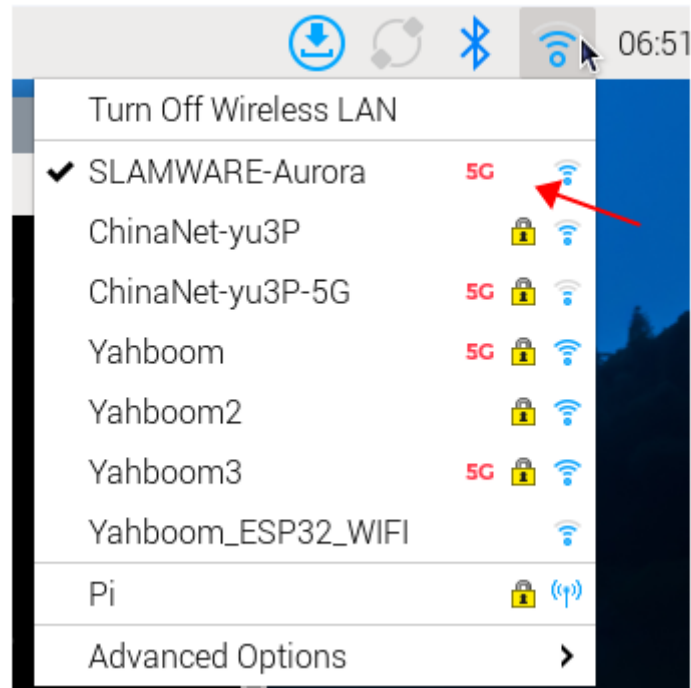
If the Aurora device is in AP mode, connect to the Aurora WIFI and start the node

```
roslaunch slamware_ros_sdk slamware_ros_sdk_server_and_view.launch
ip_address:=192.168.11.1
```



### Raspberry Pi 5, X5, Jetson nano version:

If it is the Raspberry Pi desktop version and Jetson nano, x5 desktop version, after successfully booting up, connect the external monitor to the hotspot of Aurora radar to the motherboard, as shown below,



Then enter `ifconfig` in the terminal to view the ip address under the current hotspot, and then use this address for vnc remote desktop connection, **(the premise is that the computer must also be connected to the hotspot emitted by Aurora radar)**

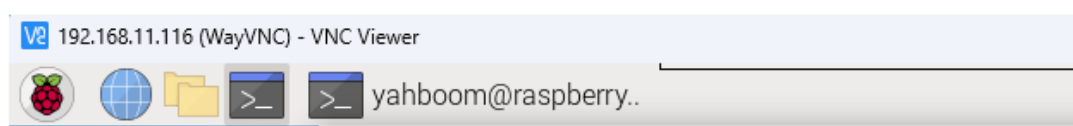
```
yahboom@raspberrypi: ~
File Edit Tabs Help

RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
device interrupt 107

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 523 bytes 39751 (38.8 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 523 bytes 39751 (38.8 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.11.116 netmask 255.255.255.0 broadcast 192.168.11.255
inet6 fe80::311a:7dd5:59ca:1384 prefixlen 64 scopeid 0x20<link>
ether d8:3a:dd:f4:6c:87 txqueuelen 1000 (Ethernet)
RX packets 1291013 bytes 1908798215 (1.7 GiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 485054 bytes 33477628 (31.9 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

yahboom@raspberrypi:~$
```



Enter `docker` in advance and enter in the terminal

```
sh ros1_noetic.sh
```

If the following interface appears, it means that you have successfully entered docker.

```
access control disabled, clients can connect from any host
WARNING: Published ports are discarded when using host network mode
root@raspberrypi:/#
```

Then enter in the docker terminal,

```
ros2 launch slamware_ros_sdk slamware_ros_sdk_server_and_view.launch
ip_address:=192.168.11.1
```

If successful, you can view the map building screen.

