# RoboSim5G - Developer Documentation

**Version:** 1.0
**Date:** March 2026
**Project:** RoboSim5G - 5G Network Simulation for Robotics Research
**Organization:** phine.tech GmbH

---

## Table of Contents



---

## 1. What is RoboSim5G?

RoboSim5G is an **open-source framework for simulating 5G network connectivity in robotic applications** within the Gazebo Ignition Fortress simulation environment. It bridges the gap between robotics simulation and realistic 5G network behavior by providing:

- **Gazebo plugins** that simulate 5G User Equipment (UE) and gNodeB (gNB) components
- **Integration with real 5G Core Network implementations** (OAI, free5GC, Open5GS)
- **Docker-based deployment** for reproducible, isolated network environments
- **ROS 2 Humble integration** for seamless robotics workflow

The framework enables developers to test and validate robotic applications under realistic 5G network conditions without requiring physical 5G hardware infrastructure.

**Key Technical Features:**

- C++17 Gazebo plugins with ROS 2 integration
- Container-orchestrated 5G Core Network functions
- Support for multiple 5G Core Network implementations
- Dynamic network configuration through SDF world files
- Real 5G protocol simulation (authentication, registration, PDU sessions)

---

# 2. Who is it for? / Who does it help and how?

## Target Developers

RoboSim5G is designed for:

1. **Robotics Researchers**
    - Testing autonomous systems under realistic wireless network conditions
    - Evaluating network-dependent robot behaviors (teleoperation, cloud processing, fleet coordination)
    - Publishing research on 5G-enabled robotics
2. **5G Network Engineers**
    - Validating network configurations before physical deployment
    - Testing edge computing scenarios with robotic endpoints
    - Developing network slicing strategies for robotic applications
3. **Software Developers**
    - Building distributed robotic systems with 5G connectivity
    - Integrating cloud-based AI/ML pipelines with robots
    - Developing multi-robot coordination systems
4. **Education and Training**
    - Teaching 5G networking concepts through practical robotics examples
    - Demonstrating end-to-end 5G system integration

## How it helps

**For Roboticists:**

- No deep 5G networking knowledge required

- Focus on robot behavior, not network infrastructure
- Test network failure scenarios safely

**For Network Engineers:**

- Validate configurations with mobile robotic endpoints
- Simulate realistic traffic patterns and mobility
- Rapid prototyping without hardware costs

**For Researchers:**

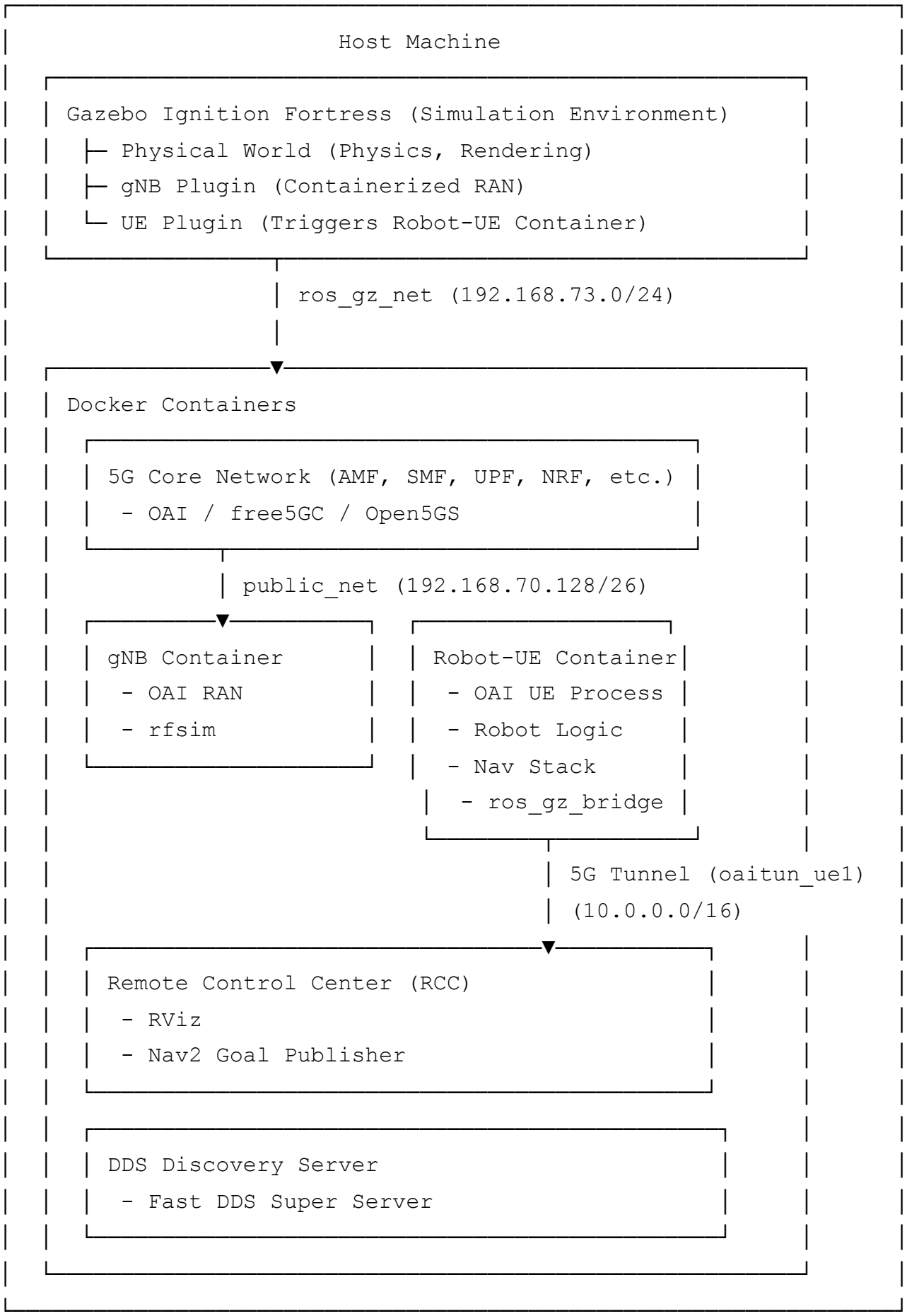
- Reproducible experiments
- Open-source, extensible platform
- Integration with standard ROS 2 workflows

---

# 3. How does it work?

## 3.1 System Architecture Overview

RoboSim5G implements a **layered architecture** that separates simulation, robotics logic, and network functions:


Host Machine
Gazebo Ignition Fortress (Simulation Environment)
├─ Physical World (Physics, Rendering)
├─ gNB Plugin (Containerized RAN)
└─ UE Plugin (Triggers Robot-UE Container)
ros_gz_net (192.168.73.0/24)
Docker Containers
5G Core Network (AMF, SMF, UPF, NRF, etc.)
- OAI / free5GC / Open5GS
public_net (192.168.70.128/26)
gNB Container
- OAI RAN
- rfsim
Robot-UE Container
- OAI UE Process
- Robot Logic
- Nav Stack
- ros_gz_bridge
5G Tunnel (oaitun_ue1)
(10.0.0.0/16)
Remote Control Center (RCC)
- RViz
- Nav2 Goal Publisher
DDS Discovery Server
- Fast DDS Super Server

**Key Components:**

1. **Gazebo Simulation Layer**
    - Runs on host machine (demo mode) or container (containerized mode)
    - Hosts gNB and UE plugins
    - Simulates physical robot and environment
2. **5G Network Layer**
    - Core Network functions in containers
    - gNB container (Radio Access Network)
    - Authentication, session management, routing
3. **Robot Application Layer**
    - Robot-UE container with dual function:
        - 5G modem simulation (OAI UE)
        - Robot control logic (Nav2, SLAM, etc.)
4. **Remote Control Layer**
    - RCC container for teleoperation/monitoring
    - Communicates via 5G tunnel
5. **Service Discovery Layer**
    - DDS Discovery Server for ROS 2 node discovery
    - Enables unicast communication over 5G

## 3.2 Software Structure

```
RoboSim5G/
├── phine-plugins/              # Core Plugin Source Code
│   └── src/phine_plugins/
│       ├── CMakeLists.txt      # Build configuration
│       ├── package.xml         # ROS 2 package manifest
│       └── src/
│           ├── RS5G_plugins.hh     # Plugin class definitions
│           ├── gNB_plugin.cc       # gNB plugin implementation
│           ├── UE_plugin.cc        # UE plugin implementation
│           ├── aux_functions.cc    # Shared utility functions
│           └── aux_functions.hh    # Utility function headers
```

```
│
├── demo/                          # Host-based Simulation Demo
│   ├── launch_robot_5G*.sh        # Launch scripts per CN type
│   ├── kill_all*.sh               # Cleanup scripts per CN type
│   ├── build_images.sh            # Image building automation
│   ├── oai_setup/                 # OAI Core Network configs
│   ├── free5gc_setup/             # free5GC Core Network configs
│   ├── open5gs_setup/             # Open5GS Core Network configs
│   ├── gazebo_launch/             # Gazebo world and launch files
│   ├── nav_stack/                 # Example robot application
│   └── images/                    # Dockerfiles for containers
│       ├── ue_amr/                # Robot-UE image
│       ├── rcc/                   # Remote Control Center image
│       └── dds_discovery_server/  # DDS server image
│
├── demo_containerized/            # Fully Containerized Demo
│   └── [similar structure to demo, Gazebo in container]
│
├── model_folder/                  # Gazebo Models
│   └── phine_gNB/                 # gNB visual model
│       ├── model.config
│       └── model.sdf
│
└── doc/                           # Documentation
    ├── 5G_CORE_NETWORKS.md        # Core Network comparison guide
    └── images/
```

## 3.3 Plugin System

**gNB Plugin Architecture**

**Purpose:** Simulates a 5G base station (gNodeB) by launching and configuring a containerized RAN implementation.

**Load-Time Behavior:**

1. Reads SDF parameters ( `<cn_type>` , `<IP_GNB>` , `<IP_AMF>` , etc.)
2. Selects configuration method based on `cn_type` (OAI/free5GC/Open5GS)
3. Modifies Core Network-specific configuration files:
    - Docker Compose files ( `docker-compose-gNB.yml` )
    - gNB configuration files (e.g., `gnb.sa.bandn78.fr1.106PRB.rfsim.conf` )
4. Launches gNB container with Docker Compose

5. Initializes ROS 2 node for pose publishing

**Runtime Behavior:**

- Publishes gNB pose to `/pose_of_<gNB_name>` topic
- Container runs OAI gNB in rfsim mode (software-defined radio simulation)

**Key Code Structure:**

```
class gNB_plugin : public gz::sim::System,
                   public gz::sim::ISystemConfigure,
                   public gz::sim::ISystemPostUpdate {
public:
    void Configure(...) override;  // Load-time configuration
    void PostUpdate(...) override; // Runtime updates

private:
    void configureOAI(const char *project_path);
    void configureFree5gc(const char *project_path);
    void configureOpen5gs(const char *project_path);

    rclcpp::Node::SharedPtr node;
    rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher;
    std::string model_name, linkName, netName, ip_gnb, ip_amf;
};
```

**UE Plugin Architecture**

**Purpose:** Simulates a 5G User Equipment (UE) integrated with robot logic by building and launching a containerized robot-UE system.

**Load-Time Behavior:**

1. Reads extensive SDF parameters (IMSI, keys, DNN, robot paths, etc.)
2. Selects Core Network-specific configuration
3. **Copies user's robot workspace** into container image directory
4. Modifies UE configuration files:
    - `docker-compose-ue.yml`
    - `conf/UE<robot_ID>.conf`
5. **Builds Docker image** containing:
    - OAI UE software
    - User's robot application
    - ros_gz_bridge

    - DDS profile for 5G tunnel interface
6. Launches robot-UE container
7. Automatically starts ros_gz_bridge
8. Optionally starts robot launch file

**Runtime Behavior:**

- OAI UE process continuously runs, maintaining 5G connection
- Robot logic executes within same container
- All ROS 2 communication routed through 5G tunnel interface

**Key Code Structure:**

```
class UE_plugin : public gz::sim::System,
                  public gz::sim::ISystemConfigure {
public:
    void Configure(...) override;

private:
    void configureOAI(const char *project_path);
    void configureFree5gc(const char *project_path);
    void configureOpen5gs(const char *project_path);

    // Extensive parameter set for UE and robot configuration
    std::string robot_container_name, imsi, key, opc, dnn;
    std::string robot_project_path, robot_launch_file_name;
    std::string ros_discovery_server;
    // ... many more parameters
};
```

**Auxiliary Functions**

Located in `aux_functions.cc/hh`, these provide shared utilities:

- File modification (replacing specific strings in config files)
- Docker command execution
- Configuration validation
- Debug logging

## 3.4 Network Architecture

**Docker Networks**

**1. public_net (phine-net)**

- **Subnet:** 192.168.70.128/26 (configurable)
- **Purpose:** Control plane communication
- **Connected containers:**
  - All 5G Core Network functions (AMF, SMF, UPF, NRF, UDR, etc.)
  - gNB container
  - Robot-UE container
  - RCC container
  - DDS Discovery Server

**2. ros_gz_net**

- **Subnet:** 192.168.73.0/24 (fixed)
- **Purpose:** Direct Gazebo ↔ Robot-UE communication (mimics physical connection)
- **Static IP assignment:**
  - Host Gazebo: 192.168.73.129
  - Robot-UE container: 192.168.73.150 (MUST NOT CHANGE)
- **Used for:** ros_gz_bridge topic transport

**3. 5G Tunnel (oaitun_ue1)**

- **Subnet:** Configured in Core Network (e.g., 10.0.0.0/16)
- **Purpose:** Simulated 5G user plane data
- **Created by:** OAI UE process upon successful registration
- **Used for:** ROS 2 communication between Robot-UE and RCC

**4. n3_net (free5GC only)**

- **Purpose:** gNB ↔ UPF GTP-U tunnel (N3 interface)
- **Note:** OAI and Open5GS use public_net for N3

**Routing Configuration**

**Host Machine:**

```
# Route 5G subnet via UPF for RCC communication
sudo ip route add <5G_subnet> via <UPF_IP> dev <public_net_interface>

# Example for OAI:
sudo ip route add 10.0.0.0/16 via 192.168.70.134 dev br-<bridge_id>
```

**RCC Container:**

- Must have route to 5G subnet via UPF or ext-dn

- Configured in docker-compose-ue.yml

**DDS Discovery Server:**

- Must be reachable from both public_net and 5G tunnel
- Requires routing through UPF

## 3.5 Containerization Strategy

**Two Deployment Modes**

**Demo Mode (demo/):**

- Gazebo runs natively on host
- Better GUI performance
- Direct GPU access
- Requires local Gazebo Fortress installation

**Containerized Mode (demo_containerized/):**

- Gazebo runs in container
- Fully reproducible environment
- Easier dependency management
- Requires X11 forwarding for GUI

**Container Roles**

**1. Core Network Containers**

- Based on official OAI/free5GC/Open5GS images
- Configured via mounted volumes
- Stateful (database persistence)

**2. gNB Container**

- Based on OAI RAN image
- Runs in rfsim mode (software radio simulation)
- Configured by gNB plugin at runtime

**3. Robot-UE Container**

- **Built dynamically** by UE plugin
- Multi-stage Dockerfile:
  - Base: OAI UE image
  - Adds: ROS 2 Humble, user's robot workspace

- Entry point script starts:
    1. OAI UE process (background)
    2. ros_gz_bridge
    3. Robot launch file

**4. RCC Container**

- ROS 2 Humble + RViz + Nav2
- DDS client configuration for Discovery Server
- Pre-configured route to 5G subnet

**5. DDS Discovery Server Container**

- Fast DDS super server
- Persistent ID for network-wide discovery
- Listens on port 11811

**Image Building Process**

The `build_images.sh` script:

1. Builds base images (dds_discovery_server, rcc)
2. Builds Gazebo simulation image (containerized mode only)
3. UE image is built dynamically by UE plugin
4. Core Network images pulled from registries

---

# 4. Getting Started as a Developer

## Prerequisites

**Required Software:**

- Ubuntu 22.04
- ROS 2 Humble
- Gazebo Ignition Fortress 2.6.9
- Docker 27.5.1+
- Docker Compose v2.33.1+
- Git

**Optional:**

- C++17 compiler (GCC 11+)
- clang-format 17 (for code formatting)

- ROS 2 development tools ( `ros-humble-desktop-full` )

## Cloning the Repository

```
git clone https://github.com/phinetech/RoboSim5G.git
cd RoboSim5G
```

## Environment Setup

```
# Set project path
export PROJECT_PATH=$(pwd)

# Add plugin path
export IGN_GAZEBO_SYSTEM_PLUGIN_PATH=$PROJECT_PATH/phine-plugins/build/ph

# Add model path
export IGN_GAZEBO_RESOURCE_PATH=$PROJECT_PATH/model_folder:$IGN_GAZEBO_RE

# Set ROS 2 domain ID (avoid conflicts)
export ROS_DOMAIN_ID=42

# Set Gazebo partition (avoid conflicts)
export IGN_PARTITION=robosim5g

# Set Gazebo transport interface (for demo mode)
export IGN_TRANSPORT_INTERFACE=192.168.73.129
```

Add these to `~/.bashrc` for persistence.

## Building the Plugins

```
cd phine-plugins
colcon build
source install/setup.bash
```

Expected output: Plugins in `build/phine_plugins/` :

- `libgNB_plugin.so`
- `libUE_plugin.so`

## Building Container Images

```
cd $PROJECT_PATH/demo  # or demo_containerized
./build_images.sh
```

This builds all required Docker images.

## Running the Demo

**Choose a Core Network:**

```
# For OAI:
source launch_robot_5G.sh

# For free5GC:
source launch_robot_5G_free5gc.sh

# For Open5GS:
source launch_robot_5G_open5gs.sh
```

**Expected Behavior:**

1. Core Network containers start (~10 seconds)
2. gNB container starts and registers with AMF
3. Gazebo opens with world loaded
4. Robot-UE container builds (first run: ~5 minutes) and starts
5. UE registers with Core Network
6. RViz opens with navigation interface

## Verifying the Setup

**Check Core Network:**

```
cd demo/oai_setup  # or free5gc_setup
docker compose ps  # All services should be "running" or "healthy"
```

**Check gNB Registration:**

```
docker logs oai-amf | grep -A 5 "gNBs' Information"
# Should show gNB listed as "Connected"
```

**Check UE Registration:**

```
docker logs oai-amf | grep -A 10 "UEs' Information"
# Should show UE with IMSI and "5GMM-REGISTERED"
```

**Check 5G Tunnel:**

```
docker exec -it ue_turtlebot ip addr show oaitun_ue1
# Should show 5G tunnel interface with IP from configured subnet
```

---

# 5. Development Guidelines

## Code Style

Follow the project's coding guidelines in CODING_GUIDELINES.md :

- **Indentation:** Tabs (not spaces)
- **Naming:**
  - Variables: `snake_case`
  - Functions: `camelCase()`
  - Classes: `CamelCase`
  - Constants: `ALL_CAPS`
- **Style:** ROS 2 C++ Style Guide (Google C++ as fallback)

## Formatting Code

```
# Format a single file
clang-format -i path/to/file.cpp

# Format all plugin sources
cd phine-plugins/src/phine_plugins/src
clang-format -i *.cc *.hh
```

Use clang-format 17 for consistency.

## Version Control Workflow

**Branching Strategy:**

- `main` : Stable releases (protected)
- `dev` : Development branch (protected)
- `feature/<name>` : New features
- `bugfix/<name>` : Bug fixes

**Commit Message Format:**

```
<type>: <short summary>

[optional body: detailed explanation]

Closes #<issue_number>
```

**Types:** `feat`, `fix`, `docs`, `refactor`, `test`, `chore`

**Example:**

```
feat: add Open5GS support to UE plugin

Implements configureOpen5gs() method with proper IMSI format
and DNN configuration for Open5GS core network.

Closes #45
```

## Merge Request Process

1. Create feature branch from `dev`
2. Make changes, commit with clear messages
3. Format code with clang-format
4. Build and test locally
5. Push branch to remote
6. Create Merge Request targeting `dev`
7. Assign reviewer
8. Address review comments
9. Merge after approval

## Testing

**Manual Testing Checklist:**

- [ ] Plugins build without errors
- [ ] Core Network containers start successfully
- [ ] gNB registers with AMF
- [ ] UE registers and obtains 5G tunnel IP
- [ ] Robot can navigate with Nav2 goals
- [ ] RViz shows robot position updates
- [ ] Switching gNB/UE off disrupts communication
- [ ] Switching back on restores communication

**Automated Testing:**
Currently, the project relies on manual testing. Future work should include:

- Unit tests for auxiliary functions
- Integration tests for plugin configuration
- CI/CD pipeline for build verification

---

# 6. Extending and Customizing

## Adding a New Core Network

**Steps:**

1. Create setup directory: `demo/<new_cn>_setup/`
2. Add Docker Compose files for CN, gNB, UE
3. Add CN-specific configuration files
4. Implement configuration methods in plugins:

```
// In RS5G_plugins.hh
void configureNewCN(const char *project_path);

// In gNB_plugin.cc and UE_plugin.cc
void gNB_plugin::configureNewCN(const char *project_path) {
    // Modify CN-specific config files
    // Launch gNB container
}
```

5. Update SDF world files with new CN parameters
6. Create launch and kill scripts
7. Document in `doc/5G_CORE_NETWORKS.md`

## Customizing Robot Application

**Replace the demo robot with your own:**

1. **Prepare your robot workspace:**

```
# Structure:
my_robot_ws/
├── src/
│   └── my_robot_pkg/
│       ├── launch/
│       │   ├── robot_logic.launch.py
│       │   └── ros_gz_bridge.launch.py
│       ├── config/
│       └── ...
└── install/  # After colcon build
```

2. **Update UE plugin parameters in world file:**

```
<robot_project_path>${PROJECT_PATH}/my_robot_ws</robot_project_path>
<robot_project_name>my_robot_ws</robot_project_name>
<robot_package_name>my_robot_pkg</robot_package_name>
<robot_launch_file_name>robot_logic.launch.py</robot_launch_file_name
<ros_gz_bridge_name>ros_gz_bridge.launch.py</ros_gz_bridge_name>
```

3. **Customize UE container Dockerfile:**

   - Edit `demo/images/ue_amr/Dockerfile`
   - Add your dependencies:

```
# Install additional ROS packages
RUN apt-get update && apt-get install -y \
    ros-humble-my-dependency \
    && rm -rf /var/lib/apt/lists/*
```

4. **Configure DDS profile for your interfaces:**

   - Edit `demo/images/ue_amr/dds.xml`
   - Whitelist 5G tunnel interface
   - Adjust message size limits if needed

## Adding New Plugin Features

**Example: Adding Signal Strength Simulation**

1. **Extend gNB plugin class:**

```
// In RS5G_plugins.hh
class gNB_plugin : public gz::sim::System, ... {
private:
    double calculateSignalStrength(const gz::math::Pose3d &ue_pose,
                                   const gz::math::Pose3d &gnb_pose);
    rclcpp::Publisher<std_msgs::msg::Float64>::SharedPtr signal_pub_;
};
```

2. **Implement in PostUpdate:**

```
// In gNB_plugin.cc
void gNB_plugin::PostUpdate(const UpdateInfo &_info, const ECM &_ecm)
    // Get UE pose from entity
    auto ue_pose = getUEPose(_ecm);
    auto gnb_pose = getGNBPose(_ecm);

    // Calculate signal strength (simple distance model)
    double signal = calculateSignalStrength(ue_pose, gnb_pose);

    // Publish
    auto msg = std_msgs::msg::Float64();
    msg.data = signal;
    signal_pub_->publish(msg);
}
```

3. **Add SDF parameter:**

```
<plugin name="phine_plugins::gNB_plugin" filename="gNB_plugin">
    <!-- existing parameters -->
    <enable_signal_strength>true</enable_signal_strength>
    <signal_publish_rate>10.0</signal_publish_rate>
</plugin>
```

## Modifying Network Parameters

**Changing 5G Subnet:**

1. Edit Core Network config:

```
# demo/oai_setup/conf/config.yaml
upf:
  interfaces:
    n3:
      ipv4_range: "10.0.0.0/16"  # Change this
```

2. Update UE plugin SDF parameter:

```
<subnet_5G>10.0.0.0/16</subnet_5G>
```

3. Update DDS profile whitelist:

```
<!-- demo/images/ue_amr/dds.xml -->
<interfaceWhiteList>
    <address>10.0.0.1</address>
    <address>10.0.0.2</address>
    <!-- ... add all IPs in new range -->
</interfaceWhiteList>
```

4. Update routing on host:

```
sudo ip route add 10.0.0.0/16 via <UPF_IP> dev <interface>
```

---

# 7. API Reference

## gNB Plugin API

**SDF Parameters:**

| Parameter | Type | Description | Default |
|---|---|---|---|
| `cn_type` | string | Core Network type ( `oai` , `free5gc` , `open5gs` ) | Required |

| Parameter | Type | Description | Default |
|---|---|---|---|
| `link_name` | string | Link name for pose publishing | `phine_cell` |
| `net_name` | string | Docker network name | `phine-net` |
| `IP_GNB` | string | Static IP for gNB container | Required |
| `IP_AMF` | string | AMF IP address | Required |
| `debug` | bool | Enable debug logging | `false` |
| `mobile_country_code` | string | MCC for PLMN | `001` |
| `mobile_network_code` | string | MNC for PLMN | `01` |

**ROS 2 Topics Published:**

- `/pose_of_<gNB_name>` (std_msgs/String): gNB pose as string

**Container Lifecycle:**

- Started by plugin during Configure()
- Stopped by user (kill scripts) or Gazebo shutdown

## UE Plugin API

**SDF Parameters:**

| Parameter | Type | Description | Default |
|---|---|---|---|
| `cn_type` | string | Core Network type | Required |
| `robot_container_name` | string | Name of robot-UE container | Required |
| `robot_id` | int | Robot ID (for multi-UE) | `1` |
| `ip_robotUE` | string | Static IP for container | Required |
| `net_name` | string | Docker network name | `phine-net` |
| `ip_gnb` | string | gNB IP address | Required |
| `debug` | bool | Enable debug logging | `false` |
| `subnet_5G` | string | 5G subnet (CIDR notation) | Required |
| `imsi` | string | UE IMSI | Required |
| `key` | string | UE authentication key (128-bit hex) | Required |

| Parameter | Type | Description | Default |
|---|---|---|---|
| `opc` | string | Operator code (128-bit hex) | Required |
| `dnn` | string | Data Network Name | Required |
| `nssai_sst` | int | Slice Service Type | `1` |
| `nssai_sd` | string | Slice Differentiator (or "no") | `no` |
| `robot_project_path` | string | Path to robot ROS 2 workspace | Required |
| `robot_project_name` | string | Workspace directory name | Required |
| `execute_robot_launch_file` | bool | Auto-start robot logic | `true` |
| `robot_package_name` | string | ROS 2 package name | Required |
| `ros_gz_bridge_name` | string | ros_gz_bridge launch file | Required |
| `robot_launch_file_name` | string | Robot logic launch file | Required |
| `ros_discovery_server` | string | DDS server IP:port | Required |

**Container Lifecycle:**

- Image built during first Configure()
- Container started by plugin
- Stopped by user (kill scripts) or Gazebo shutdown

## Auxiliary Functions API

**File Modification:**

```
void replaceInFile(const std::string &filePath,
                   const std::string &searchString,
                   const std::string &replaceString);
```

**Docker Execution:**

```
int executeDockerCompose(const std::string &composePath,
                         const std::string &command);
```

**Configuration Validation:**

```
bool validateCoreNetworkConfig(const std::string &cn_type,
                               const std::string &configPath);
```

# 8. Additional Resources

## Documentation Links

- **Main README:** README.md
- **Core Network Comparison:** doc/5G_CORE_NETWORKS.md
- **Contributing Guide:** CONTRIBUTING.md
- **Code of Conduct:** CODE_OF_CONDUCT.md
- **Coding Guidelines:** CODING_GUIDELINES.md

## External Resources

**5G Core Networks:**

- OAI GitLab
- free5GC Documentation
- Open5GS Documentation

**Gazebo:**

- Gazebo Fortress Documentation
- Gazebo Plugin Tutorial

**ROS 2:**

- ROS 2 Humble Documentation
- ros_gz Integration
- Fast DDS Discovery Server

**Docker:**

- Docker Documentation
- Docker Compose Specification

## Community Support

- **Slack Chat:** Join RoboSim Workspace
- **GitHub Issues:** RoboSim5G Issues
- **Email:** riccardo.belletti@phine.tech

## Citing RoboSim5G

If you use RoboSim5G in your research, please cite:

```
@software{robosim5g2026,
  title = {RoboSim5G: Open-Source 5G Network Simulation for Robotics},
  author = {phine.tech GmbH},
  year = {2026},
  url = {https://github.com/phinetech/RoboSim5G}
}
```

---

**Document Version:** 1.0
**Last Updated:** March 2026
**Maintainer:** phine.tech GmbH

For questions, contributions, or support, please refer to CONTRIBUTING.md or join our Slack community.