ROS2 Humble + Gazebo

无人机仿真配置完整教程

ArduPilot(APM)固件版 - 自动飞行代码验证

适用于ArduPilot/ArduCopter自动驾驶仪

Ubuntu 22.04 LTS + Gazebo Classic 11

2025年2月（修订版V2）

目录

一、概述与环境准备

1.1 教程目标与适用范围

本教程专为使用ArduPilot（APM）固件的无人机开发者设计，提供一套完整的ROS2 Humble与Gazebo仿真环境配置指南。教程将帮助您搭建可靠的仿真测试平台，用于验证自动飞行算法和控制代码。通过本教程的学习，您将能够建立一个功能完备的仿真环境，实现无人机模型的精确物理仿真、传感器数据模拟以及飞行控制代码的全面测试验证。

ArduPilot是一款功能强大、应用广泛的开源自动驾驶仪软件，支持多种飞行器类型包括多旋翼（ArduCopter）、固定翼（ArduPlane）、垂直起降飞行器（ArduVTOL）以及地面车辆（Rover）。本教程将以ArduCopter四旋翼为例，详细介绍仿真环境的搭建过程，并确保所有软件版本之间的兼容性。

1.2 系统硬件要求

无人机仿真环境对计算机硬件有一定要求，合理的硬件配置能够显著提升仿真体验和开发效率。以下是运行ROS2 Humble与Gazebo无人机仿真的推荐硬件配置。

硬件组件	最低配置	推荐配置	说明
CPU	Intel i5-8代 / AMD Ryzen 5	Intel i7-10代+ / AMD Ryzen 7	多核处理器有助于提升仿真计算效率
内存	16 GB DDR4	32 GB DDR4	Gazebo仿真消耗大量内存资源
显卡	NVIDIA GTX 1060 6GB	NVIDIA RTX 3060 12GB	独立显卡对Gazebo渲染至关重要
存储	256 GB SSD	512 GB NVMe SSD	SSD显著提升编译和加载速度

表1-1 硬件配置要求对照表

1.3 软件环境总览与版本兼容性

完整的无人机仿真环境涉及多个软件组件的协同工作，版本兼容性是搭建成功的关键。以下表格详细列出了各软件组件的版本要求及其兼容性说明，请务必按照指定版本进行安装，避免因版本不匹配导致的问题。

软件组件	推荐版本	兼容性说明
操作系统	Ubuntu 22.04 LTS	ROS2 Humble官方唯一支持的Ubuntu版本
ROS2	Humble Hawksbill	LTS版本，支持至2027年
Gazebo	Gazebo Classic 11.14.x	ArduPilot_Gazebo官方支持版本，不要使用新版Gazebo(Ignition/Harmonic)
ArduPilot	ArduCopter 4.5.x (Copter-4.5)	稳定版本，与Gazebo Classic 11兼容良好

ArduPilot_Gazebo	master分支或最新release	必须与Gazebo Classic 11.x配合使用
MAVROS	2.x（ROS2版本）	通过apt安装ros-humble-mavros
MAVProxy	1.8.x+	ArduPilot推荐的地面站工具

表1-2 软件环境组件清单与版本兼容性

重要提示：ArduPilot_Gazebo插件目前仅支持Gazebo Classic 11.x版本，不支持新一代Gazebo（原Ignition Gazebo，现称Gazebo Harmonic/Sonic等）。如果您错误安装了新版Gazebo，需要先卸载再安装Gazebo Classic 11。版本不匹配是导致仿真启动失败的最常见原因之一。

二、ROS2 Humble 安装配置

2.1 系统准备与依赖安装

在开始安装ROS2 Humble之前，需要确保系统环境满足安装要求，并完成必要的系统配置。以下命令将更新系统并安装必要的依赖工具。

```
# 步骤1：更新系统软件包
sudo apt update && sudo apt upgrade -y

# 步骤2：安装基础依赖工具
sudo apt install -y curl gnupg2 lsb-release software-properties-common

# 步骤3：配置区域设置（确保字符编码正确）
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

# 步骤4：验证区域设置
locale  # 应显示 LANG=en_US.UTF-8
```

2.2 ROS2 Humble 安装

```
# 步骤1：添加ROS2 GPG密钥
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key \
  -o /usr/share/keyrings/ros-archive-keyring.gpg

# 步骤2：添加ROS2软件源
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] \
  http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" \
  | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

# 步骤3：更新软件包列表
sudo apt update

# 步骤4：安装ROS2 Humble桌面完整版（推荐）
sudo apt install -y ros-humble-desktop

# 步骤5：安装开发工具
sudo apt install -y ros-dev-tools python3-colcon-common-extensions python3-rosdep

# 步骤6：验证安装
ros2 --version
echo $ROS_DISTRO  # 应输出：humble
```

2.3 环境变量配置

```
# 步骤1：将ROS2环境配置添加到bashrc
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc

# 步骤2：重新加载bashrc
source ~/.bashrc

# 步骤3：初始化rosdep（依赖管理工具）
sudo rosdep init
rosdep update

# 步骤4：验证环境配置
printenv | grep -i ros
```

```
# 应看到 ROS_DISTRO=humble 等环境变量
```

2.4 ROS2 工作空间创建

ROS2工作空间是存放功能包源码和编译产物的目录结构。以下命令将创建一个专门用于无人机仿真的工作空间，并建立推荐的目录结构。

```
# 步骤1：创建工作空间根目录
mkdir -p ~/drone_sim_ws/src
cd ~/drone_sim_ws

# 步骤2：编译空工作空间（生成install目录）
colcon build

# 步骤3：添加工作空间环境到bashrc
echo "source ~/drone_sim_ws/install/setup.bash" >> ~/.bashrc
source ~/.bashrc

# 步骤4：创建子目录结构（稍后使用）
mkdir -p ~/drone_sim_ws/src/drone_description
mkdir -p ~/drone_sim_ws/src/drone_gazebo
mkdir -p ~/drone_sim_ws/src/drone_control
```

三、Gazebo 仿真环境搭建

3.1 Gazebo 版本选择与安装

ArduPilot_Gazebo插件目前仅支持Gazebo Classic 11.x版本。这是选择Gazebo版本的关键依据，请务必安装正确版本。以下命令将安装Gazebo Classic 11及其ROS2集成包。

```
# 步骤1：安装Gazebo Classic 11及ROS2集成包
# 注意：这会自动安装Gazebo 11.x版本
sudo apt install -y ros-humble-gazebo-ros-pkgs

# 步骤2：安装Gazebo开发工具（编译插件需要）
sudo apt install -y gazebo11 libgazebo11-dev

# 步骤3：验证Gazebo版本
gazebo --version
# 必须显示 Gazebo 11.x.x（如11.14.0）
# 如果显示其他版本，说明安装错误

# 步骤4：测试Gazebo启动
gazebo --verbose
# 应打开Gazebo窗口，显示空世界
# 按Ctrl+C关闭
```

重要：如果您的系统安装了新版Gazebo（如Gazebo Garden/Harmonic），需要先卸载再安装Gazebo Classic 11。可以使用以下命令检查已安装的Gazebo版本：

```
# 检查已安装的Gazebo相关包
dpkg -l | grep gazebo

# 如果看到 gazebo12 或 ign-gazebo 等包，需要卸载
# sudo apt remove gazebo12 libgazebo12-dev
# sudo apt remove 'ignition-*' 'libignition-*'

# 然后重新安装Gazebo 11
# sudo apt install -y gazebo11 libgazebo11-dev ros-humble-gazebo-ros-pkgs
```

3.2 Gazebo 与 ROS2 集成

```
# 测试Gazebo与ROS2集成
# 终端1：启动Gazebo空世界
ros2 launch gazebo_ros gazebo.launch.py

# 终端2：查看ROS2话题列表
ros2 topic list
# 应看到以下话题：
# /clock
# /gazebo/world_stats
# /gazebo/model_states
# 等等...

# 终端3：生成测试模型验证
ros2 run gazebo_ros spawn_entity.py -entity test_box -database cube_20k -x 0 -y 0 -z 2
# 应在Gazebo窗口中看到一个立方体
```

四、ArduPilot(APM)仿真平台搭建

4.1 ArduPilot固件安装配置

ArduPilot源码需要从GitHub克隆，并使用waf构建系统进行编译。以下步骤将引导您完成ArduPilot的安装和编译过程。建议使用稳定分支（如Copter-4.5）而非master分支，以确保稳定性。

```
# 步骤1：进入工作空间src目录
cd ~/drone_sim_ws/src

# 步骤2：克隆ArduPilot仓库（使用稳定分支）
git clone https://github.com/ArduPilot/ardupilot.git
cd ardupilot

# 步骤3：切换到稳定分支（推荐Copter-4.5）
git checkout Copter-4.5
# 或使用最新稳定版：git checkout Copter-4.5.7

# 步骤4：更新子模块（重要！）
git submodule update --init --recursive

# 步骤5：安装ArduPilot依赖
Tools/environment_install/install-prereqs-ubuntu.sh -y

# 步骤6：配置编译目标为SITL（软件在环仿真）
./waf configure --board sitl

# 步骤7：编译ArduCopter（四旋翼）
./waf copter

# 编译过程需要5-15分钟，取决于CPU性能
```

编译成功后，可执行文件位于 build/sitl/bin/arducopter。接下来需要配置环境变量，以便系统能够找到ArduPilot的工具脚本和可执行文件。

```
# 步骤8：配置环境变量
# 将以下内容添加到 ~/.bashrc 文件末尾：

# ===== ArduPilot 环境变量 =====
export ARDUPILOT_ROOT=~/drone_sim_ws/src/ardupilot
export PATH=$PATH:$ARDUPILOT_ROOT/Tools/autotest
export PATH=$PATH:$ARDUPILOT_ROOT/build/sitl/bin
# ============================

# 步骤9：重新加载bashrc
source ~/.bashrc

# 步骤10：验证环境变量
echo $ARDUPILOT_ROOT
# 应输出：/home/您的用户名/drone_sim_ws/src/ardupilot

# 步骤11：验证arducopter可执行
which arducopter
# 应输出：/home/您的用户名/drone_sim_ws/src/ardupilot/build/sitl/bin/arducopter
```

4.2 ArduPilot_Gazebo插件安装

ArduPilot_Gazebo是ArduPilot官方提供的Gazebo仿真插件，用于实现飞控与Gazebo仿真环境之间的通信。该插件包含传感器模型、电机模型和必要的通信接口。以下步骤将安装该插件。

```
# 步骤1：进入工作空间src目录
cd ~/drone_sim_ws/src

# 步骤2：克隆ArduPilot_Gazebo仓库
git clone https://github.com/ArduPilot/ArduPilot_Gazebo.git
cd ArduPilot_Gazebo

# 步骤3：检查是否为兼容版本
git log -1 --oneline
# 确保是最新版本

# 步骤4：创建构建目录
mkdir -p build && cd build

# 步骤5：配置CMake（指定Gazebo 11）
cmake .. -DGAZEBO_VERSION=11

# 步骤6：编译
make -j$(nproc)

# 步骤7：安装插件
sudo make install

# 安装完成后，插件库文件位于：
# /usr/lib/x86_64-linux-gnu/gazebo-11/plugins/
# 或
# /usr/local/lib/
```

插件安装完成后，需要配置Gazebo环境变量，使其能够找到模型文件和插件库。这是仿真启动成功的关键步骤。

```
# 步骤8：配置Gazebo环境变量
# 将以下内容添加到 ~/.bashrc 文件末尾：

# ===== Gazebo 环境变量 =====
# ArduPilot_Gazebo模型路径
export GAZEBO_MODEL_PATH=$ARDUPILOT_ROOT/Tools/simulation/gazebo-classic/models:$HOME/drone_sim_ws/src/ArduPilot_Gazebo/models:$GAZ

# ArduPilot_Gazebo插件路径
export GAZEBO_PLUGIN_PATH=$HOME/drone_sim_ws/src/ArduPilot_Gazebo/build:$GAZEBO_PLUGIN_PATH

# Gazebo资源路径
export GAZEBO_RESOURCE_PATH=$HOME/drone_sim_ws/src/ArduPilot_Gazebo/models:$GAZEBO_RESOURCE_PATH
# ============================

# 步骤9：重新加载bashrc
source ~/.bashrc

# 步骤10：验证环境变量
echo $GAZEBO_MODEL_PATH
echo $GAZEBO_PLUGIN_PATH
# 应包含ArduPilot_Gazebo相关路径
```

4.3 仿真环境启动与验证

完成以上配置后，可以启动ArduPilot SITL仿真进行验证。以下命令将启动一个完整的仿真环境，包括Gazebo、ArduPilot飞控和MAVProxy控制台。

重要说明：sim_vehicle.py脚本必须从ArduPilot根目录执行，并使用python3命令。以下是正确的启动方式：

```
# 正确的启动方式
# 步骤1: 进入ArduPilot目录
cd ~/drone_sim_ws/src/ardupilot

# 步骤2: 启动仿真（完整命令）
python3 Tools/autotest/sim_vehicle.py -v ArduCopter -f gazebo-iris --console --map

# 参数说明:
# -v ArduCopter  : 使用ArduCopter固件（四旋翼）
# -f gazebo-iris : 使用iris四旋翼Gazebo模型
# --console      : 启动MAVProxy控制台
# --map          : 启动地图显示窗口
```

启动过程需要约30-60秒，以下是成功启动时应看到的输出信息：

```
# ========== 预期输出信息 ==========

# 1. Gazebo启动信息:
# [Msg] Gazebo multi-robot simulator, version 11.14.0
# [Msg] Loading model iris...
# [Msg] Loaded model iris

# 2. ArduPilot启动信息:
# ArduCopter V4.5.7 (c0c0c0c0)
# Frame: QUAD
# Gazebo model iris found!
# APM: Gazebo model iris found

# 3. EKF初始化信息:
# EKF2 IMU0 initialised
# EKF2 IMU1 initialised

# 4. GPS锁定信息:
# GPS lock acquired
# EKF2 GPS checks passed

# 5. MAVProxy控制台提示:
# Mode STABILIZE
# APM: Ready to FLY

# ========== 成功启动标志 ==========
# - Gazebo窗口显示iris四旋翼模型
# - MAVProxy控制台显示 "Ready to FLY"
# - 地图窗口显示无人机位置
# - 没有红色错误信息
```

如果启动失败，请检查以下几点：1）Gazebo版本是否为11.x；2）GAZEBO_MODEL_PATH和GAZEBO_PLUGIN_PATH是否正确设置；3）是否在ardupilot目录下执行命令；4）网络是否正常（首次启动需要下载模型资源）。

五、关键文件准备与配置

5.1 完整目录结构说明

为了确保仿真环境能够正常工作，所有文件必须放置在正确的位置。以下是完整的目录结构说明，包括每个目录和文件的用途。请按照此结构组织您的文件。

```
# 完整目录结构
# 假设工作空间位于 ~/drone_sim_ws

~/drone_sim_ws/                          # 工作空间根目录
|-- src/                                 # 源码目录
|   |-- ardupilot/                       # ArduPilot固件（从GitHub克隆）
|   |   |-- Tools/
|   |   |   |-- autotest/
|   |   |   |   |-- sim_vehicle.py       # 仿真启动脚本
|   |   |   |-- simulation/
|   |   |       |-- gazebo-classic/
|   |   |           |-- models/          # ArduPilot自带Gazebo模型
|   |   |-- build/
|   |   |   |-- sitl/
|   |   |       |-- bin/
|   |   |           |-- arducopter       # 编译后的可执行文件
|   |
|   |-- ArduPilot_Gazebo/                # Gazebo插件（从GitHub克隆）
|   |   |-- models/                      # Gazebo模型文件
|   |   |   |-- iris/                    # iris四旋翼模型
|   |   |       |-- model.sdf
|   |   |-- build/                       # 编译输出目录
|   |       |-- libArduPilotPlugin.so    # 编译后的插件库
|   |
|   |-- drone_description/               # 【自定义】无人机模型描述包
|   |   |-- CMakeLists.txt
|   |   |-- package.xml
|   |   |-- urdf/                        # URDF模型文件（可选）
|   |   |   |-- drone.urdf
|   |   |-- meshes/                      # 3D网格文件
|   |   |   |-- drone.stl
|   |   |-- launch/                      # 模型显示launch文件
|   |       |-- display.launch.py
|   |
|   |-- drone_gazebo/                    # 【自定义】Gazebo仿真配置包
|   |   |-- CMakeLists.txt
|   |   |-- package.xml
|   |   |-- launch/                      # Launch文件
|   |   |   |-- drone_sim.launch.py      # 主启动文件
|   |   |   |-- spawn_drone.launch.py    # 模型生成文件
|   |   |-- worlds/                      # 世界文件
|   |   |   |-- empty.world
|   |   |-- models/                      # 自定义模型
|   |   |   |-- my_drone/
|   |   |       |-- model.sdf
|   |   |-- config/                      # 配置文件
|   |       |-- mavros_config.yaml
|   |
|   |-- drone_control/                   # 【自定义】飞行控制功能包
|       |-- CMakeLists.txt
|       |-- package.xml
|       |-- src/                         # C++源码
|       |   |-- flight_controller.cpp
|       |-- scripts/                     # Python脚本
```

```
|       |   |-- flight_control.py
|       |-- launch/
|           |-- control.launch.py
|
|-- build/                              # 编译中间文件
|-- install/                            # 安装文件
|-- log/                                # 编译日志
```

5.2 Launch 文件编写与放置

Launch文件用于启动仿真环境的各个组件。以下是一个完整的Launch文件示例，以及如何创建和放置这些文件。

步骤1：创建drone_gazebo功能包

```
# 创建功能包
cd ~/drone_sim_ws/src
ros2 pkg create drone_gazebo --build-type ament_cmake --dependencies gazebo_ros

# 创建必要的目录
cd drone_gazebo
mkdir -p launch worlds models config
```

步骤2：创建Launch文件

文件路径：~/drone_sim_ws/src/drone_gazebo/launch/drone_sim.launch.py

```
# drone_sim.launch.py
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import ExecuteProcess, TimerAction
from launch_ros.actions import Node

def generate_launch_description():
    pkg_gazebo = get_package_share_directory('drone_gazebo')
    world_file = os.path.join(pkg_gazebo, 'worlds', 'empty.world')

    gzserver = ExecuteProcess(
        cmd=['gzserver', '--verbose', '-s', 'libgazebo_ros_init.so',
             '-s', 'libgazebo_ros_factory.so', world_file],
        output='screen'
    )

    gzclient = ExecuteProcess(
        cmd=['gzclient'],
        output='screen'
    )

    mavros_node = TimerAction(
        period=8.0,
        actions=[
            Node(
                package='mavros',
                executable='mavros_node',
                parameters=[{
                    'fcu_url': 'udp://:14550@127.0.0.1:14555',
                    'system_id': 1,
                    'target_system_id': 1,
                }],
                output='screen'
            )
        ]
```

```
    )

    return LaunchDescription([gzserver, gzclient, mavros_node])
```

步骤3：创建世界文件

文件路径：~/drone_sim_ws/src/drone_gazebo/worlds/empty.world

```
# 世界文件内容（XML格式，注意转义）
# 创建文件：~/drone_sim_ws/src/drone_gazebo/worlds/empty.world

# 文件内容如下（复制时去掉注释符号）：
# <?xml version="1.0" ?>
# <sdf version="1.6">
#   <world name="drone_world">
#     <physics type="ode">
#       <max_step_size>0.001</max_step_size>
#       <real_time_factor>1</real_time_factor>
#     </physics>
#     <light name="sun" type="directional">
#       <pose>0 0 10 0 0 0</pose>
#       <diffuse>0.8 0.8 0.8 1</diffuse>
#     </light>
#     <model name="ground_plane">
#       <static>true</static>
#       <link name="link">
#         <collision name="collision">
#           <geometry><plane><normal>0 0 1</normal></plane></geometry>
#         </collision>
#       </link>
#     </model>
#   </world>
# </sdf>

# 或直接使用ArduPilot自带的世界文件
```

步骤4：修改CMakeLists.txt（添加文件安装规则）

```
# 在 ~/drone_sim_ws/src/drone_gazebo/CMakeLists.txt 中添加：

# 安装launch文件
install(DIRECTORY launch/
  DESTINATION share/${PROJECT_NAME}/launch
)

# 安装worlds文件
install(DIRECTORY worlds/
  DESTINATION share/${PROJECT_NAME}/worlds
)

# 安装models文件
install(DIRECTORY models/
  DESTINATION share/${PROJECT_NAME}/models
)

# 安装config文件
install(DIRECTORY config/
  DESTINATION share/${PROJECT_NAME}/config
)
```

5.3 SDF模型文件配置

SDF模型文件定义了无人机的物理属性和传感器配置。ArduPilot_Gazebo已经提供了预配置的iris模型，您可以直接使用或根据需要修改。

```
# 自定义模型目录结构
~/drone_sim_ws/src/drone_gazebo/models/
`-- my_quadcopter/                    # 您的自定义模型
    |-- model.config                  # 模型配置文件
    `-- model.sdf                     # SDF模型定义文件
```

model.config文件内容：

```
# 文件路径: ~/drone_sim_ws/src/drone_gazebo/models/my_quadcopter/model.config
# 内容如下:

# <?xml version="1.0"?>
# <model>
#   <name>My Quadcopter</name>
#   <version>1.0</version>
#   <sdf version="1.6">model.sdf</sdf>
#   <author>
#     <name>Your Name</name>
#     <email>your@email.com</email>
#   </author>
#   <description>A custom quadcopter for ArduPilot simulation</description>
# </model>
```

model.sdf文件内容：

```
# 文件路径: ~/drone_sim_ws/src/drone_gazebo/models/my_quadcopter/model.sdf
# 内容如下（注意: 这是简化版本，完整版本请参考ArduPilot_Gazebo/models/iris）:

# <?xml version="1.0" ?>
# <sdf version="1.6">
#   <model name="my_quadcopter">
#     <pose>0 0 0.1 0 0 0</pose>
#
#     <link name="base_link">
#       <inertial>
#         <mass>1.5</mass>
#         <inertia>
#           <ixx>0.01</ixx> <ixy>0</ixy> <ixz>0</ixz>
#           <iyy>0.01</iyy> <iyz>0</iyz>
#           <izz>0.02</izz>
#         </inertia>
#       </inertial>
#
#       <collision name="collision">
#         <geometry>
#           <cylinder><radius>0.2</radius><length>0.1</length></cylinder>
#         </geometry>
#       </collision>
#
#       <visual name="visual">
#         <geometry>
#           <cylinder><radius>0.2</radius><length>0.1</length></cylinder>
#         </geometry>
#       </visual>
#     </link>
#
#     <!-- ArduPilot插件配置 -->
#     <plugin name="ArduPilotPlugin" filename="libArduPilotPlugin.so">
#       <fdm_addr>127.0.0.1</fdm_addr>
```

```
#       <fdm_port_in>9002</fdm_port_in>
#       <fdm_port_out>9003</fdm_port_out>
#       <imu_name>imu_sensor</imu_name>
#     </plugin>
#
#     <!-- IMU传感器 -->
#     <sensor name="imu_sensor" type="imu">
#       <topic>/ap/imu</topic>
#       <update_rate>100</update_rate>
#     </sensor>
#   </model>
# </sdf>
```

5.4 MAVROS配置文件

```
# MAVROS配置文件
# 文件路径: ~/drone_sim_ws/src/drone_gazebo/config/mavros_config.yaml

mavros:
  fcu_url: "udp://:14550@127.0.0.1:14555"
  gcs_url: ""

  conn:
    heartbeat_rate: 1.0
    timeout: 10.0
    timesync_rate: 10.0

  system_id: 1
  component_id: 240
  target_system_id: 1
  target_component_id: 1

plugin:
  global_position:
    frame_id: "map"
  local_position:
    frame_id: "map"
  imu:
    frame_id: "base_link"
  setpoint_position:
    frame_id: "map"
    mav_frame: "LOCAL_NED"
```

六、自动飞行代码验证流程

6.1 MAVROS安装与配置

```
# 安装MAVROS（ROS2版本）
sudo apt install -y ros-humble-mavros ros-humble-mavros-extras

# 安装MAVLink地理信息依赖
sudo apt install -y geographiclib-tools libgeographic-dev

# 安装地理数据（首次安装需要，约100MB）
sudo geographiclib-get-geoids egm2008-1

# 验证MAVROS安装
ros2 pkg list | grep mavros
# 应显示：
# mavros
# mavros_extras
# mavros_msgs
```

6.2 仿真环境完整启动流程

以下是在ArduPilot仿真环境中验证自动飞行代码的完整启动流程。请按照顺序在多个终端中执行命令。

```
# ========== 终端1：启动ArduPilot SITL仿真 ==========
# 进入ArduPilot目录
cd ~/drone_sim_ws/src/ardupilot

# 启动仿真（正确命令）
python3 Tools/autotest/sim_vehicle.py -v ArduCopter -f gazebo-iris --console --map

# 等待启动完成（约30-60秒）
# 成功启动的标志：
# - Gazebo窗口打开，显示iris四旋翼
# - MAVProxy控制台显示 "Ready to FLY"
# - 地图窗口显示无人机位置

# ========== 终端2：启动MAVROS节点 ==========
# 等待终端1完全启动后执行
source /opt/ros/humble/setup.bash
source ~/drone_sim_ws/install/setup.bash

ros2 run mavros mavros_node --ros-args \
  -p fcu_url:=udp://:14550@127.0.0.1:14555 \
  -p system_id:=1 \
  -p target_system_id:=1

# 成功连接的输出：
# [INFO] [mavros]: FCU URL: udp://:14550@127.0.0.1:14555
# [INFO] [mavros]: MAVROS started. MY ID 1.240, TARGET ID 1.1
# [INFO] [mavros]: CON: Got HEARTBEAT, connected.

# ========== 终端3：验证连接状态 ==========
source /opt/ros/humble/setup.bash

# 查看MAVROS状态
ros2 topic echo /mavros/state --once

# 预期输出：
```

```
# connected: true
# armed: false
# guided: false
# manual_input: true
# mode: "STABILIZE"

# 查看位置信息
ros2 topic echo /mavros/local_position/pose --once

# 查看话题列表
ros2 topic list | grep mavros
```

MAVROS常用话题列表：

话题名称	消息类型	用途
/mavros/state	mavros_msgs/State	飞控状态（连接、解锁、模式）
/mavros/local_position/pose	geometry_msgs/PoseStamped	本地位置（读取）
/mavros/global_position/global	sensor_msgs/NavSatFix	GPS位置（读取）
/mavros/imu/data	sensor_msgs/Imu	IMU数据（读取）
/mavros/setpoint_position/local	geometry_msgs/PoseStamped	位置设定点（控制）
/mavros/setpoint_velocity/cmd_vel	geometry_msgs/Twist	速度设定点（控制）

表6-1 MAVROS常用话题列表

6.3 飞行代码测试方法

```
# ========== 阶段1：解锁测试 ==========
# 在MAVProxy控制台（终端1）中执行：
mode GUIDED        # 切换到GUIDED模式
arm throttle       # 解锁

# 或通过ROS2服务：
ros2 service call /mavros/cmd/arming mavros_msgs/srv/CommandBool "{value: true}"

# 验证解锁状态：
ros2 topic echo /mavros/state --once
# 应显示 armed: true

# ========== 阶段2：起飞测试 ==========
# 在MAVProxy控制台中执行：
takeoff 3          # 起飞到3米高度

# 或通过ROS2发布位置设定点：
ros2 topic pub /mavros/setpoint_position/local geometry_msgs/msg/PoseStamped \
  "{header: {frame_id: 'map'}, pose: {position: {x: 0.0, y: 0.0, z: 3.0}, orientation: {w: 1.0}}}" \
  --rate 10

# 监控位置：
ros2 topic echo /mavros/local_position/pose

# ========== 阶段3：降落测试 ==========
```

```
# 在MAVProxy中：
mode LAND              # 切换到LAND模式

# 或通过ROS2服务：
ros2 service call /mavros/cmd/land mavros_msgs/srv/CommandTOL \
  "{min_pitch: 0.0, yaw: 0.0, latitude: 0.0, longitude: 0.0, altitude: 0.0}"

# ========== 阶段4：上锁测试 ==========
# 确保无人机已着陆后上锁：
ros2 service call /mavros/cmd/arming mavros_msgs/srv/CommandBool "{value: false}"
```

七、常见问题与解决方案

7.1 Gazebo相关问题

问题现象	可能原因	解决方案
Gazebo版本错误	安装了新版Gazebo	卸载新版，安装Gazebo 11：sudo apt install gazebo11 libgazebo11-dev
模型加载失败	GAZEBO_MODEL_PATH未设置	检查并设置环境变量，确保包含ArduPilot_Gazebo/models路径
插件加载失败	GAZEBO_PLUGIN_PATH未设置	检查并设置环境变量，确保包含ArduPilot_Gazebo/build路径
Gazebo黑屏	显卡驱动问题	检查NVIDIA驱动：nvidia-smi，确保正确安装

表7-1 Gazebo常见问题

7.2 ArduPilot启动问题

```
# 问题1：sim_vehicle.py命令找不到
# 解决：确保在ardupilot目录下执行，并使用python3
cd ~/drone_sim_ws/src/ardupilot
python3 Tools/autotest/sim_vehicle.py -v ArduCopter -f gazebo-iris --console --map

# 问题2：找不到iris模型
# 解决：检查GAZEBO_MODEL_PATH环境变量
echo $GAZEBO_MODEL_PATH
# 应包含：.../ArduPilot_Gazebo/models

# 问题3：飞控无法解锁
# 解决：检查EKF状态，在MAVProxy中执行：
status          # 查看系统状态
ekfstatus       # 查看EKF状态
# 确保GPS已锁定，EKF已初始化

# 问题4：MAVROS无法连接
# 解决：检查端口配置
# ArduPilot默认使用UDP 14550/14555
ros2 run mavros mavros_node --ros-args -p fcu_url:=udp://:14550@127.0.0.1:14555

# 问题5：编译ArduPilot失败
# 解决：确保子模块已更新
cd ~/drone_sim_ws/src/ardupilot
git submodule update --init --recursive
./waf distclean
./waf configure --board sitl
./waf copter
```

7.3 环境变量检查清单

```
# 环境变量检查脚本
# 将以下内容保存为 check_env.sh 并执行

#!/bin/bash
```

```
echo "===== 环境变量检查 ====="

echo "1. ROS_DISTRO: $ROS_DISTRO"
[ "$ROS_DISTRO" = "humble" ] && echo "   [OK]" || echo "   [ERROR]"

echo "2. ARDUPILOT_ROOT: $ARDUPILOT_ROOT"
[ -d "$ARDUPILOT_ROOT" ] && echo "   [OK]" || echo "   [ERROR]"

echo "3. GAZEBO_MODEL_PATH:"
echo "   $GAZEBO_MODEL_PATH"
[[ "$GAZEBO_MODEL_PATH" == *"ArduPilot_Gazebo"* ]] && echo "   [OK]" || echo "   [ERROR]"

echo "4. GAZEBO_PLUGIN_PATH:"
echo "   $GAZEBO_PLUGIN_PATH"
[[ "$GAZEBO_PLUGIN_PATH" == *"ArduPilot_Gazebo"* ]] && echo "   [OK]" || echo "   [ERROR]"

echo "5. Gazebo版本: "
gazebo --version | head -1

echo "6. arducopter可执行文件: "
which arducopter && echo "   [OK]" || echo "   [ERROR]"

echo "===== 检查完成 ====="
```

通过本教程的学习，您应该已经掌握了ROS2 Humble与Gazebo ArduPilot仿真环境的完整搭建流程。如果在配置过程中遇到问题，请首先检查环境变量设置和软件版本兼容性。建议将本教程中的环境变量配置保存到bashrc文件中，以便每次打开终端时自动加载。