Install ESP32-microros components

Note: Since the micro_ros components are hosted on GitHub, downloading the file may fail due to network issues. The supporting virtual machine has already downloaded and compiled the microros component. There is no need to reinstall the ESP32-microros component unless necessary.

1、 Download microros components

Download the microros component to ~/esp/Samples/extra_component. Here we take downloading the humble version as an example.

```
mkdir -p ~/esp/Samples/extra_components
cd ~/esp/Samples/extra_components
git clone -b humble https://github.com/micro-ROS/micro_ros_espidf_component.git
cd micro_ros_espidf_component
```

micro_ros component source code address：https://github.com/micro-ROS/micro_ros_espidf_component

2、 Activate the ESP-IDF development environment

Run the following command in the terminal

```
source ~/esp/esp-idf/export.sh
```

Note: Every time you open a new terminal, you need to activate the ESP-IDF development environment before you can compile the ESP-IDF project.

3.、 Install dependencies

Run the following command in the terminal

```
pip3 install catkin_pkg lark-parser empy colcon-common-extensions
```

Note: Be sure to activate the ESP-IDF development environment before running the above command.

4、 Modify microros configuration

The default microros configuration only supports 1 node, 2 publishers, 2 subscribers, 1 service, 1 client, and 1 history record.

If you feel it is not enough, you can modify the microros configuration. If you do not modify it now and need to recompile if it does not meet the requirements for subsequent use, you can enter the following command to clear the files generated by microros compilation, and then recompile to generate a microros static library.

```
cd
~/esp/Samples/extra_components/micro_ros_espidf_component/examples/int32_publish
er
idf.py clean-microros
```

Modify the colcon.meta file in the micro_ros_espidf_component directory, find the "rmw_microxrcedds" column, and then modify it according to the actual situation. For ease of use, the publisher, subscriber, and history records are all modified to 3.

NOTE: Since the performance of the microcontroller is limited, it can just meet the needs to avoid wasting resources and affecting the communication speed.

```
        "rmw_microxrcedds": {
            "cmake-args": [
                "-DRMW_UXRCE_XML_BUFFER_LENGTH=400",
                "-DRMW_UXRCE_TRANSPORT=udp",
                "-DRMW_UXRCE_MAX_NODES=1",
                "-DRMW_UXRCE_MAX_PUBLISHERS=3",
                "-DRMW_UXRCE_MAX_SUBSCRIPTIONS=3",
                "-DRMW_UXRCE_MAX_SERVICES=1",
                "-DRMW_UXRCE_MAX_CLIENTS=1",
                "-DRMW_UXRCE_MAX_HISTORY=3"
            ]
        },
```

5、 Compile and generate microros static library

Then run the following command in the terminal.

```
cd
~/esp/Samples/extra_components/micro_ros_espidf_component/examples/int32_publish
er
idf.py set-target esp32s3
```

After compilation is completed, "Configuring done" and "Generating done" are prompted.

At this time, the system will automatically download the required files and compile and generate the libmicroros.a static library.

```
ls ../../*.a
```


6、Instructions for use

After compiling and generating the libmicroros.a static library, it means that the assembly and installation are complete. Next, you only need to add extra_components to the external components of the project.

Add the following content in the root CMakeLists.txt file of your project, where "../../extra_components" is replaced according to the actual path.

```
set (EXTRA_COMPONENT_DIRS "../../extra_components")
```

If you need to modify the compilation parameters of microros, you can run the following command to clear the cache of microros, and then recompile to generate the libmicroros.a static library.

```
idf.py clean-microros
```