Guide to MicroRos Usage

Daniele Genovese

August 9, 2024

1 Prerequisites

1.1 ROS

MicroRos is a version of ROS2 designed to be used on microcontrollers. Its aim is to bring ROS capabilities to systems with a limited processing power, memory and storage. Since MicroRos allows the communication between ROS networks and embedded systems, a basic knowledge about ROS/ROS2 is required. In this document, it is assumed that a ROS2 distro is already installed on the computer.

1.2 Required HW/SW

1.2.1 HW

- Microcontroller (MC) (in this case, an ESP32 MC will be used)
- USB-MicroUSB cable

1.2.2 SW

- Visual Studio Code (VSCode)
- Integrated Development Environment extension for VSCode (in this case, PlatformIO)
- MicroRos agent
- MicroRos application

2 PlatformIO Setup

2.1 VSCode

Open the terminal and write the following line:

sudo apt install code --classic

Once the installation has ended, open VSCode by writing:

code .

2.2 PlatformIO extension

When VSCode opened, go to the top left bar and open "Extensions"

Figure 1: Extensions submenu

Search for PlatformIO extension and install it.

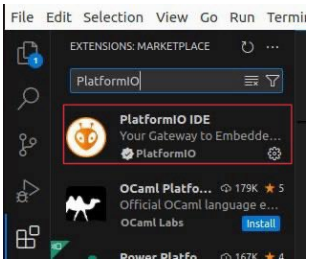

Figure 2: PlatfotmIO installation

2.3 PlatformIO new project

Once installed, the submenu will appear on the left of the VSCode window:

Figure 3: PlatfotmIO submenu

Click on it, select *Create New Project* and then, *+ New Project*.
Now, name your project, select your board and the framework. After that, deselect the option *Use default location* and choose your ROS2 workspace. This is an example of configuration:

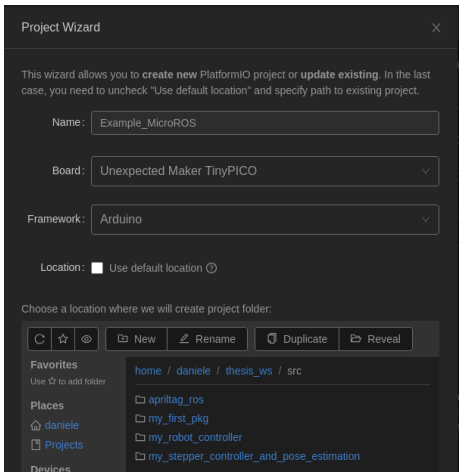

Figure 4: PlatformIO project configuration

The two most important files are main.cpp (the application file) and platformio.ini (the configuration file). Now, open the main file inside the directory /path/to/your/workspace/Example_MicroROS/src. The aim of this example is to run a simple publisher on the MC. To achieve this, we can copy the publisher example from Here into main.cpp.

After that, the file platformio.ini must be configured. The field platform, board and framework are configured by default. The library dependencies must be added. In this case, only micro_ros_platformio library must be included:

https://github.com/micro-ROS/micro_ros_platformio

Moreover, the type of transport must be chosen. In this case, the serial transport is used. The file must look like this:

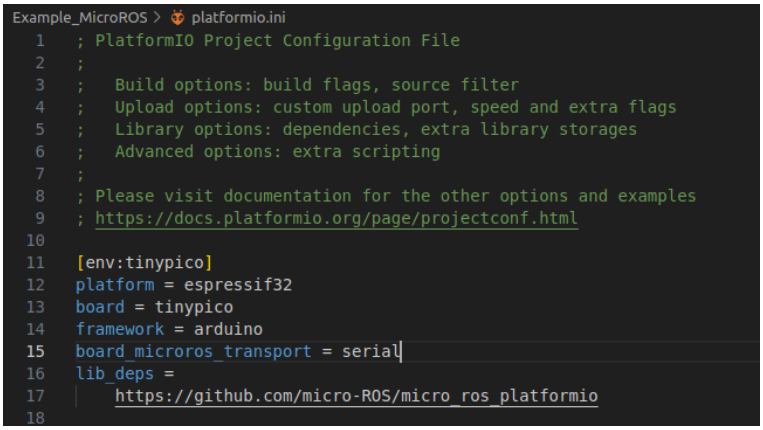

Figure 5: Platformio.ini file

Pay attention, in this case, ROS2 iron is being using, that is the default distro for MicroRos. In case of using another distro, the following line must be included inside the Platformio.ini file:

$$\text{board\_microros\_distro} = \prec your\_distro \succ$$

Then, save the file and wait for the update of metadata.
Once the metadata are updated, the application is ready to be flashed onto the MC. Using the USB-microUSB cable, connect the board to the computer. Now, compile the file using the bottom-left menu.

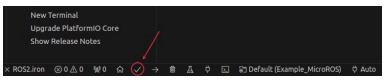

Figure 6: Compilation button

If the compilation went well, the output should look like this:

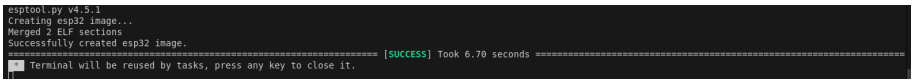

Figure 7: Correct compilation output

Finally, upload the application onto the board using the button on the lower left menu.

Figure 8: Upload button

The obtained output should look like this:

Figure 9: Correct upload output

If this type of error is encountered:

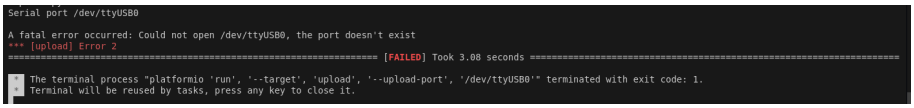

Figure 10: Common error

Type the following line on the terminal to ensure that the user is in the dialout group and try to upload again the application (reboot the system before uploading):

sudo usermod -aG dialout $USER

3 MicroROS agent

3.1 MicroROS installation

Once the application has been uploaded, the microROS build system must be installed.

- Create a workspace for microROS and download the MicroRos tools

  source /opt/ros/$ROS_DISTRO/setup.bash

  mkdir microros_ws

  cd microros_ws

  git clone -b $ROS_DISTRO https://github.com/micro-ROS/micro_ros_setup.git src/micro_ros_setup

- Update the dependencies

  sudo apt update && rosdep update

  rosdep install –from-paths src –ignore-src -y

- Install pip

  sudo apt-get install python3-pip

- Build and source MicroROS tools

  colcon build

  source install/local_setup.bash

3.2 MicroROS agent run

Now, the microROS agent must be created. To do this, follow the next steps:

- Download the package

  ros2 run micro_ros_setup create_agent_ws.sh

- Build the package

  ros2 run micro_ros_setup build_agent.sh

  source install/local_setup.bash

- Find the serial device name

  ls /dev/serial/by-id/

- Run the agent

  ros2 run micro_ros_agent micro_ros_agent serial –dev [device]

  (In this case the serial name is :

  /dev/serial/by-id/usb-Silicon_Labs_CP2104_USB_to_UART_Bridge

  _Controller_01C84D77-if00-port0

  so the command becomes :

  ros2 run micro_ros_agent micro_ros_agent serial –dev /dev/serial/by-id/usb-Silicon_Labs

  _CP2104_USB_to_UART_Bridge_Controller_01C84D77-if00-port0)

After that, you will obtain something like this:

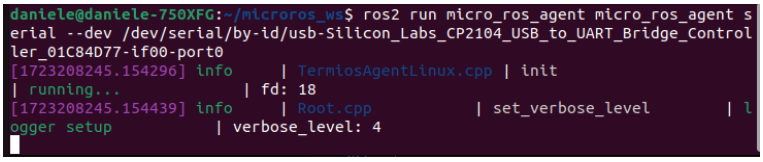

Figure 11: Agent starting output

At this point, remove and insert USB cable of MC again and the agent has been created:

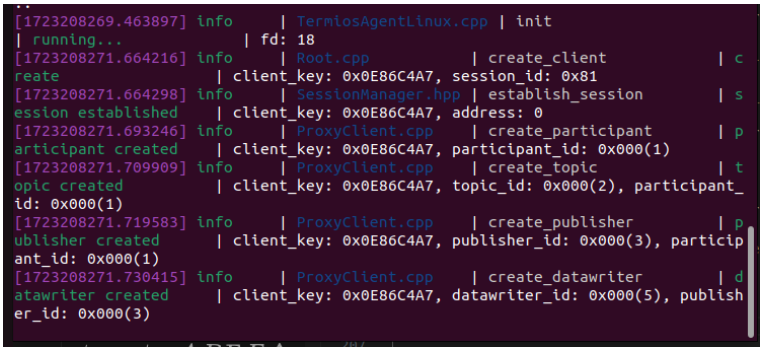

Figure 12: Agent final output

And this is what the node is publishing:

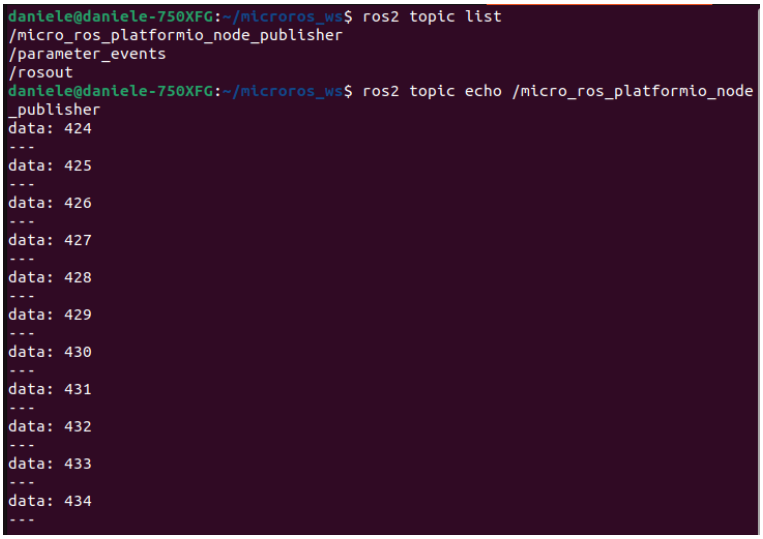

Figure 13: Application output