

Setting up our ROS2 “workspace”

1. ROS2 workspaces are typically created in the Users home directory(folder). Within the “*workspaces*” folder a project folder is created. Typically named after the Robot being worked on (eg *my_robot_ws*).

```
mkdir ~/workspaces
cd ~/workspaces
mkdir my_robot_ws
```

2. Create a “*src*” folder that will be the “root” of all the packages for the “*my_robot_ws*” project. NB: packages cannot be nested within each other!

```
cd my_robot_ws
mkdir src
```

3. All the ROS2 packages for the project, “*my_robot_ws*”, will be created in folder *~/workspaces/my_robot_ws/src*. This is performed using the ROS2 package creation command as follows:

```
ros2 pkg create hri_stt.pkg --build-type ament_cmake
```

4. A description of the package is written to the screen. Navigate into the newly created package folder:

```
cd hri_stt.pkg
```

5. Within this package folder there will be 2 new folders and two new files:

```
Folders:    include & src
Files:      CMakeLists.txt & package.xml
```

6. Now create a “*scripts*” folder for our Python3 code. This will then give us 3 folders within the folder: *~/workspaces/my_robot_ws/src* Also create a launch folder which will later contain our launch files.

```
mkdir scripts
mkdir launch
```

7. Add an empty file to the scripts folder

```
cd scripts
touch __init__.py
```

The purpose of the “*__init__.py*” file is to mark the **scripts** folder as a Python Package folder. This is then referenced by the CMakeList.txt file

8. We can now build our workspace using the ROS2 “**colcon**” build tool. Navigate to the “*my_robot_ws*” workspace.

```
cd ~/workspaces/my_robot_ws
```

colcon build

This should report "package successfully built" and list the files in the current folder:
`~/workspaces/my_robot_ws.`

9. There should now be 3 additional folders making 4 in total with the existing **src** folder.
 New folders are: *build, install & log*

10. Now to include our package in the current terminal environment execute:
source install/setup.bash

11. Now check the terminal session is aware of our newly created ROS2 package being, *hri_stt.pkg*, by listing the packages:

ros2 pkg list

OR to filter package list:

ros2 pkg list | grep -i hri_stt.pkg

Adding our Python code to our Package

1. **Copy your Python modules to the scripts folder in our package:**

```
cd ~/workspaces/my_robot_ws/src/hri_stt.pkg/scripts
cp <yourPython files>.py .
```

for this example, the python files are *stt.py* and *listen.py*

2. **Update package.xml**

Unlike ROS1 we need to update the package.xml file as shown in Figure 1. All lines not highlighted are generated in the **colcon build** process.

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>hri_stt.pkg</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="ray@todo.todo">ray</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>
  <buildtool_depend>ament_cmake_python</buildtool_depend>

  <depend>roscpp</depend>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

These 4 lines are auto generated and need to be populated with your information prior to distributing the package.

The 2 Highlighted lines are added manually

Figure 1 - package.xml file

3. Update CMakeLists.txt file

Unlike ROS1 we also need to update the CMakeLists.txt file as shown in Figure 2. All lines not highlighted are generated in the **colcon build** process.

```

cmake_minimum_required(VERSION 3.5)
project(hri_stt.pkg)

# Default to C99
if(NOT CMAKE_C_STANDARD)
  set(CMAKE_C_STANDARD 99)
endif()

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
find_package(ament_cmake_python REQUIRED)
find_package(rclpy REQUIRED)

ament_python_install_package(scripts/)

install(PROGRAMS
  scripts/stt_v2.py
  scripts/listen.py
  DESTINATION lib/${PROJECT_NAME}
)

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

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git repo)
  # uncomment the line when this package is not in a git repo
  #set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

ament_package()

```

The 7 Highlighted lines - added manually.

All python code must be referenced here in the PROGRAMS section.

These 3 lines are all that is necessary when using any number of launch files – added manually.

Figure 2 - CMakeLists.txt file

4. ROS2 Parameters

In ROS2 the parameters are attributes of the ROS2 node. They can be declared with default values in the `__init__` of the node class. These can be overwritten in the launch file.

5. ROS2 Launch files

Launch files must follow the following format:

`<your_launch_name>.launch.py`

And be stored in:

`~/workspaces/my_robot_ws/src/hri_stt.pkg/launch`

In ROS2 the launch files are written in python3. The file in Figure 3 shows a single node with ROS2 parameters defined. The `LaunchDescription` can contain many nodes to be launch at the same time, as the `ld` variable is a python list.

```
# Name : sst.launch.py
# Author: Derek Ripper
# Date : 12 Jan 2022
#
# Purpose: To launch Speech to text node
#####
from launch          import LaunchDescription
from launch_ros.actions import Node
from launch.actions   import ExecuteProcess

def generate_launch_description():
    ld = LaunchDescription ([
        Node(
            package   ="hri_stt_pkg",
            executable ="stt_v2.py",
            name      ="stt_node", #Takes priority over node name in package code
            output    ="screen",
            emulate_tty = True,
            parameters =[
                {"SR_SPEECH_ENGINE" : "google"},
                {"SR_ENERGY_THRESHOLD" : 300 },
                {"SR_PAUSE_THRESHOLD" : 1.01 },
            ]
        )
    ])
    return ld
```

Figure 3- Typical launch.py file

The node can then be run by:

ros2 launch hri_stt.pkg stt.launch.py