Arduino Set-up and the first micro-ros code

Overview: We need to now upload our code to ESP32 for publishing or subscribing to any message or data. We need to have the following three things installed in our system for ESP part.

- Install Arduino IDE from https://www.arduino.cc/en/software/
- Install ESP32 board in Arduino IDE
- Install Micro-ros Arduino library for your ROS version from https://github.com/micro-ROS/micro_ros_arduino

First Micro-ROS Program: This program will simply publish a welcome message using micro-ros and ESP32. Step by step brief explanation of the example is given below.

Include the following libraries including micro ros, ros client, string msg and so on.

```
#include <micro_ros_arduino.h>
#include <Arduino.h>
#include <rcl/rcl.h>
#include <rclc/rclc.h>
#include <rclc/executor.h>
#include <std_msgs/msg/string.h>
```

Declaring node, Hello publisher and other ROS resources.

```
// ROS node and publisher
rcl_node_t node;
rcl_publisher_t pub_hello;
rclc_executor_t executor;
rcl_allocator_t allocator;
rclc_support_t support;
```

Declaring String message and checking agent connection.

```
std_msgs__msg__String hello_msg;
// Check agent connection function
bool check_agent_connection() {
  return rmw_uros_ping_agent(100, 1) == RMW_RET_OK; }
```

Setting up setup function for defining transport, checking connection, allocation, support, node initialization, publisher and executor.

```
void setup() {
  // Initialize serial transport for micro-ROS
  set_microros_transports();
  // Wait for agent
  while (!check_agent_connection()) {
    delay(1000);
  }
  allocator = rcl_get_default_allocator();
  // Initialize ROS support
  rclc_support_init(&support, 0, NULL, &allocator);
  // Create node
  rclc_node_init_default(&node, "esp32_hello_node", "", &support);

  // Create publisher for LDR sensor
  rclc_publisher_init_default(
    &pub_hello,
    &node,
    ROSIDL_GET_MSG_TYPE_SUPPORT(std_msgs, msg,  String),
    "message"
  );
  // Create executor
  rclc_executor_init(&executor, &support.context, 1, &allocator);
}
```

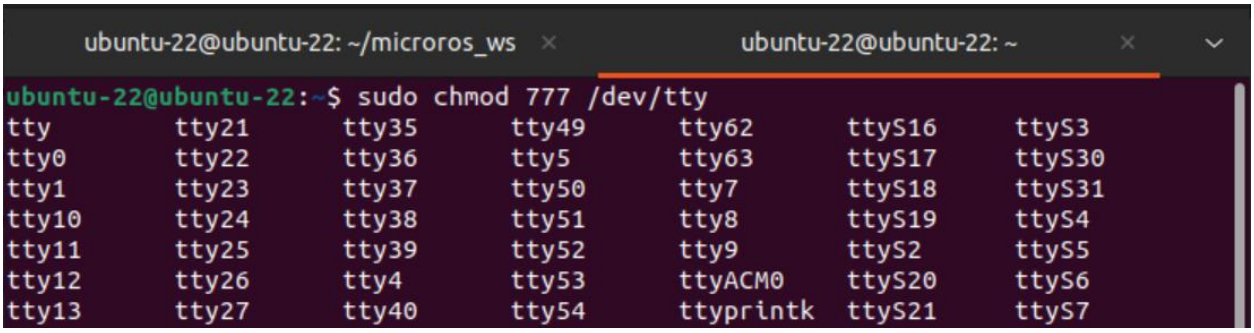
Publishing the char* message

```
void loop() {
  if (check_agent_connection()) {
    char* msg = "Hello from ESP32 using Micro-ROS";
    hello_msg.data.data = msg;
    rcl_publish(&pub_hello, &hello_msg, NULL);
  }
  delay(500);
}
```

Uploading Code: After writing the above code upload it to ESP32 by selecting appropriate port. Remember the port for the future. If there is any permission issue, give permission by following command. In my case, the port is /dev/ACMO

sudo chmod 777 /dev/ttyACM0

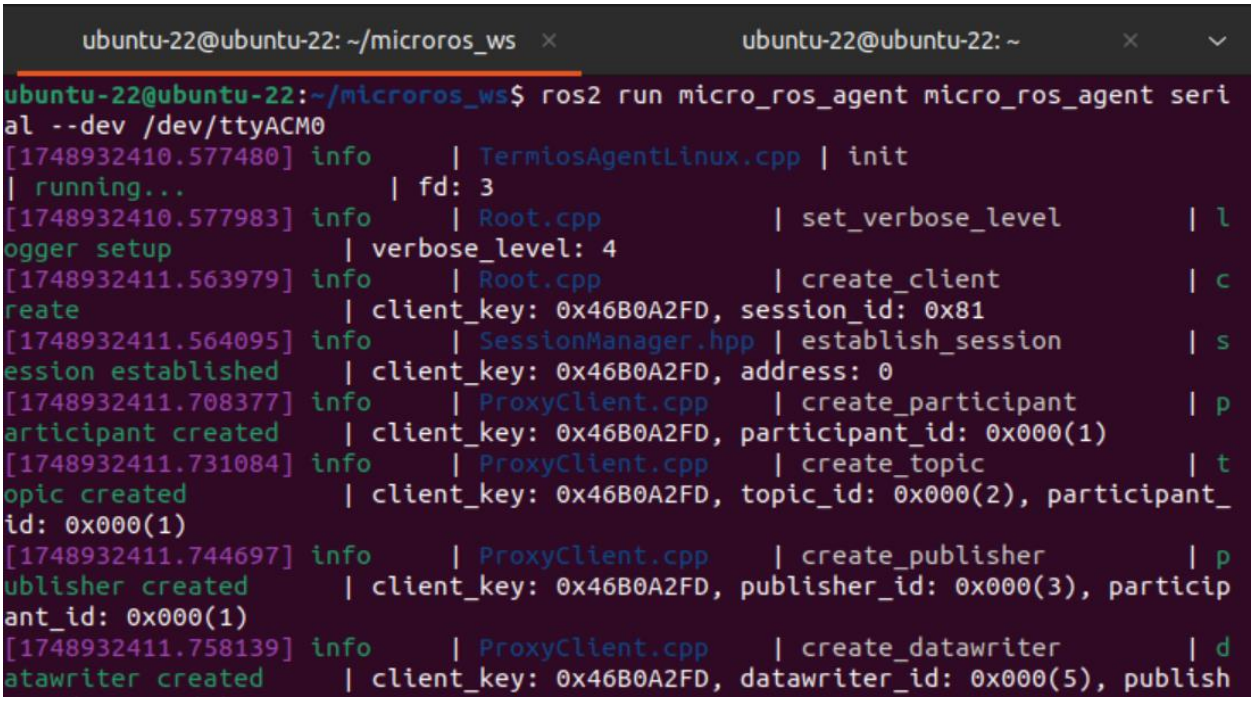
Note: In most cases the port is ACM0 or USB0, but you can check by double tapping Tab after writing below.



Running micro-ros agent: Now we need to run the micro-ros agent by executing the following command with serial argument in which we will pass our port path.

ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyUSB0

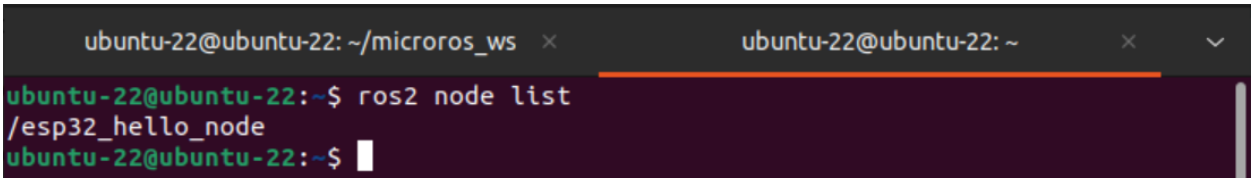
If everything works fine we will have the following output.



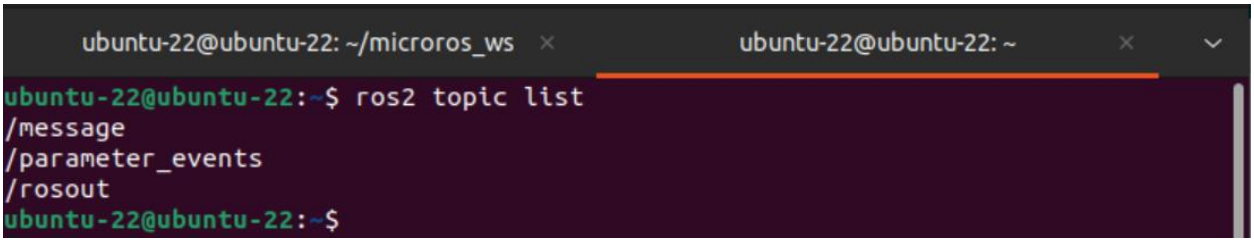
Checking the data in ROS: In our Arduino code we have the following ROS information.

- **Node:** esp32_hello_node
- **Topic:** message
- **Message:** Hello from ESP32 using Micro-ROS

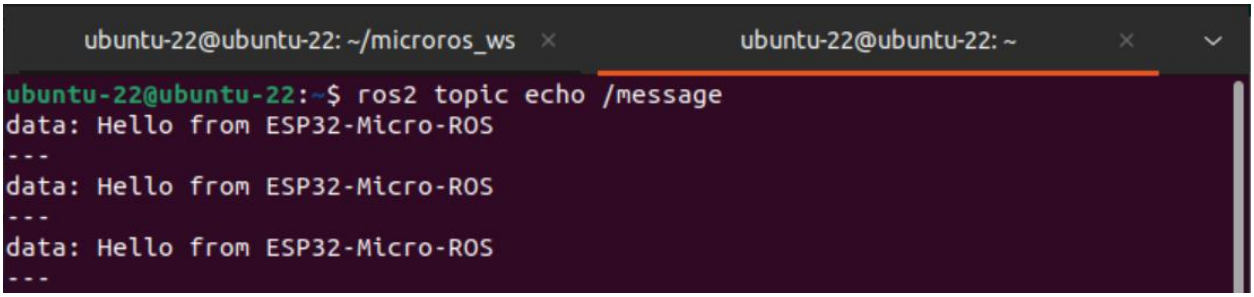
Checking Node.



Checking Topic.



Checking Message.



Task 1: Publish an integer, float and char message in single program.

Task 2: Publish the hello message using WiFi transport. Use the following link for help: https://github.com/micro-ROS/micro_ros_arduino/blob/jazzy/examples/micro-ros_publisher_wifi/micro-ros_publisher_wifi.ino