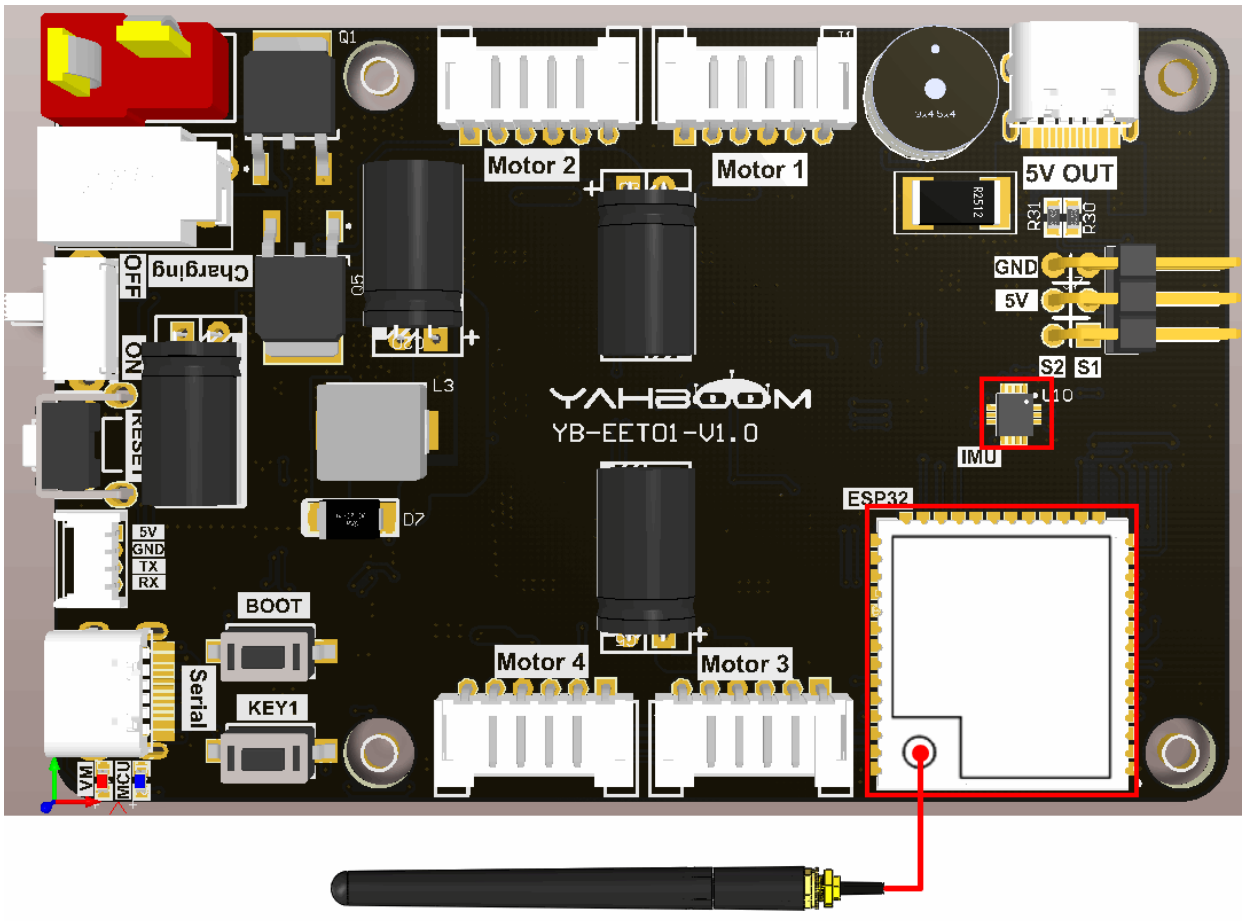
Release IMU data topic

1. Experimental purpose

Learn ESP32-microROS components, access the ROS2 environment, and publish IMU data topics.

2. Hardware connection

As shown in the figure below, the microROS control board integrates the IMU six-axis attitude sensor and the ESP32-S3-WROOM core module. It has its own wireless WiFi function. The ESP32-S3 core module needs to be connected to the antenna and the type-C data cable. The computer and microROS control board function as firmware burning.

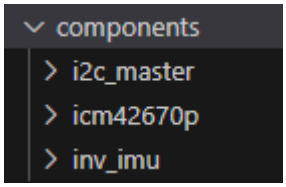


3. Core code analysis

The virtual machine path corresponding to the program source code is as follows

```
~/esp/Samples/microros_samples/imu_publisher
```

Since the IMU six-axis attitude sensor ICM42670P is used this time, and the components of ICM42670P have been made in the previous routine, the components of ICM42670P need to be copied to the components directory of the project. Call Icm42670p_Init at the beginning of the program to initialize ICM42670P.



Initialize the release of IMU information, set frame_id to "imu_frame", and then decide whether to add the ROS_NAMESPACE prefix based on whether ROS_NAMESPACE is empty.

```
void imu_ros_init(void)
{
    msg_imu.angular_velocity.x = 0;
    msg_imu.angular_velocity.y = 0;
    msg_imu.angular_velocity.z = 0;

    msg_imu.linear_acceleration.x = 0;
    msg_imu.linear_acceleration.y = 0;
    msg_imu.linear_acceleration.z = 0;

    msg_imu.orientation.x = 0;
    msg_imu.orientation.y = 0;
    msg_imu.orientation.z = 0;
    msg_imu.orientation.w = 1;

    char* content_frame_id = "imu_frame";
    int len_namespace = strlen(ROS_NAMESPACE);
    int len_frame_id_max = len_namespace + strlen(content_frame_id) + 2;
    // ESP_LOGI(TAG, "imu frame len:%d", len_frame_id_max);
    char* frame_id = malloc(len_frame_id_max);
    if (len_namespace == 0)
    {
        // ROS命名空间为空字符
        // The ROS namespace is empty characters
        sprintf(frame_id, "%s", content_frame_id);
    }
    else
    {
        // 拼接命名空间和frame id
        // Concatenate the namespace and frame id
        sprintf(frame_id, "%s/%s", ROS_NAMESPACE, content_frame_id);
```

```
    }
    msg_imu.header.frame_id =
micro_ros_string_utilities_set(msg_imu.header.frame_id, frame_id);
    free(frame_id);
}
```

Create a new IMU data update task to update the IMU data every 10 milliseconds.

```
void imu_update_data_task(void *arg)
{
    int16_t gyro_raw[3] = {0};
    int16_t accel_raw[3] = {0};
    float imu_accel_g[3] = {0};
    float imu_gyro_dps[3] = {0};
    while (1)
    {
        Icm42670p_Get_Gyro_RawData(gyro_raw);
        Icm42670p_Get_Accel_RawData(accel_raw);
        Icm42670p_Get_Accel_g(imu_accel_g);
        Icm42670p_Get_Gyro_dps(imu_gyro_dps);
        msg_imu.angular_velocity.x = imu_gyro_dps[0];
        msg_imu.angular_velocity.y = imu_gyro_dps[1];
        msg_imu.angular_velocity.z = imu_gyro_dps[2];
        msg_imu.linear_acceleration.x = imu_accel_g[0];
        msg_imu.linear_acceleration.y = imu_accel_g[1];
        msg_imu.linear_acceleration.z = imu_accel_g[2];
        vTaskDelay(pdMS_TO_TICKS(10));
    }
    vTaskDelete(NULL);
}
```

Get the WiFi name and password to connect from the IDF configuration tool.

```
#define ESP_WIFI_SSID      CONFIG_ESP_WIFI_SSID
#define ESP_WIFI_PASS      CONFIG_ESP_WIFI_PASSWORD
#define ESP_MAXIMUM_RETRY  CONFIG_ESP_MAXIMUM_RETRY
```

The uros_network_interface_initialize function will connect to WiFi hotspots based on the WiFi configuration in IDF.

```
ESP_ERROR_CHECK(uros_network_interface_initialize());
```

Then obtain ROS_NAMESPACE, ROS_DOMAIN_ID, ROS_AGENT_IP and ROS_AGENT_PORT from the IDF configuration tool.

```
#define ROS_NAMESPACE      CONFIG_MICRO_ROS_NAMESPACE
#define ROS_DOMAIN_ID      CONFIG_MICRO_ROS_DOMAIN_ID
#define ROS_AGENT_IP       CONFIG_MICRO_ROS_AGENT_IP
#define ROS_AGENT_PORT     CONFIG_MICRO_ROS_AGENT_PORT
```

Initialize the configuration of microROS, in which ROS_DOMAIN_ID, ROS_AGENT_IP and ROS_AGENT_PORT are modified in the IDF configuration tool according to actual needs.

```
    rcl_allocator_t allocator = rcl_get_default_allocator();
    rclc_support_t support;

    // 创建rcl初始化选项
    // Create init_options.
    rcl_init_options_t init_options = rcl_get_zero_initialized_init_options();
    RCCHECK(rcl_init_options_init(&init_options, allocator));
    // 修改ROS域ID
    // change ros domain id
    RCCHECK(rcl_init_options_set_domain_id(&init_options, ROS_DOMAIN_ID));

    // 初始化rmw选项
    // Initialize the rmw options
    rmw_init_options_t *rmw_options =
rcl_init_options_get_rmw_init_options(&init_options);

    // 设置静态代理IP和端口
    // Setup static agent IP and port
    RCCHECK(rmw_uros_options_set_udp_address(ROS_AGENT_IP, ROS_AGENT_PORT,
rmw_options));
```

Try to connect to the proxy. If the connection is successful, go to the next step. If the connection to the proxy is unsuccessful, you will always be in the connected state.

```
    while (1)
    {
        ESP_LOGI(TAG, "Connecting agent: %s:%s", ROS_AGENT_IP, ROS_AGENT_PORT);
        state_agent = rclc_support_init_with_options(&support, 0, NULL,
&init_options, &allocator);
        if (state_agent == ESP_OK)
        {
            ESP_LOGI(TAG, "Connected agent: %s:%s", ROS_AGENT_IP, ROS_AGENT_PORT);
            break;
        }
        vTaskDelay(pdMS_TO_TICKS(500));
    }
```

Create the node "imu_publisher", in which ROS_NAMESPACE is empty by default and can be modified in the IDF configuration tool according to actual conditions.

```
    rcl_node_t node;
    RCCHECK(rclc_node_init_default(&node, "imu_publisher", ROS_NAMESPACE,
&support));
```

To create publisher "imu", you need to specify the publisher information as sensor_msgs/msg/Imu type.

```
    RCCHECK(rclc_publisher_init_default(
        &publisher_imu,
        &node,
        ROSIDL_GET_MSG_TYPE_SUPPORT(sensor_msgs, msg, Imu),
        "imu"));
```

Create a timer for the publisher with a publishing frequency of 20HZ.

```
    const unsigned int timer_timeout = 50;
    RCCHECK(rclc_timer_init_default(
        &timer_imu,
        &support,
        RCL_MS_TO_NS(timer_timeout),
        timer_imu_callback));
```

Create an executor, where the three parameters are the numbers controlled by the executor, which should be greater than or equal to the number of subscribers and publishers added to the executor. and add the publisher's timer to the executor

```
    rclc_executor_t executor;
    int handle_num = 1;
    RCCHECK(rclc_executor_init(&executor, &support.context, handle_num,
&allocator));

    RCCHECK(rclc_executor_add_timer(&executor, &timer_imu));
```

The main function of IMU's timer callback function is to send Imu data.

```
void timer_imu_callback(rcl_timer_t *timer, int64_t last_call_time)
{
    RCLC_UNUSED(last_call_time);
    if (timer != NULL)
    {
        struct timespec time_stamp = get_timespec();
        msg_imu.header.stamp.sec = time_stamp.tv_sec;
        msg_imu.header.stamp.nanosec = time_stamp.tv_nsec;
        RCSOFTCHECK(rcl_publish(&publisher_imu, &msg_imu, NULL));
    }
}
```

Call rclc_executor_spin_some in the loop to make microros work normally.

```
    while (1)
    {
        rclc_executor_spin_some(&executor, RCL_MS_TO_NS(100));
        usleep(1000);
    }
```

4. Compile and download flash firmware

Use a Type-C data cable to connect the virtual machine/computer and the microROS control board. If the system pops up, choose to connect to the virtual machine.

Activate the ESP-IDF development environment. Note that every time you open a new terminal, you need to activate the ESP-IDF development environment before compiling the firmware.

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

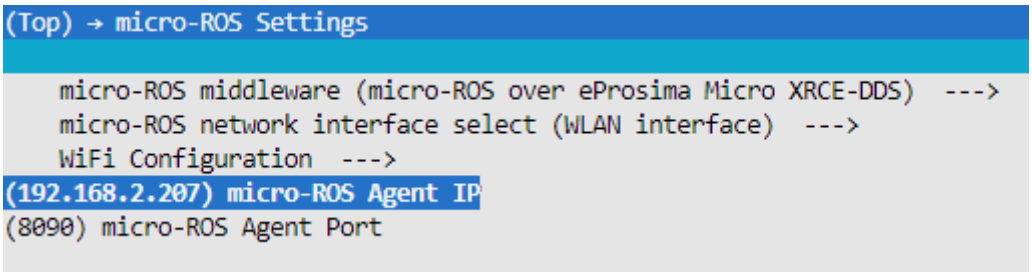
Enter the project directory

```
cd ~/esp/Samples/microros_samples/imu_publisher
```

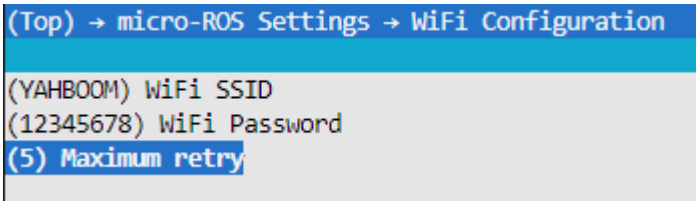
Open the ESP-IDF configuration tool.

```
idf.py menuconfig
```

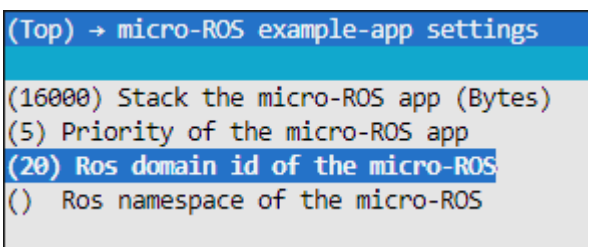
Open micro-ROS Settings, fill in the IP address of the agent host in micro-ROS Agent IP, and fill in the port number of the agent host in micro-ROS Agent Port.



Open micro-ROS Settings->WiFi Configuration in sequence, and fill in your own WiFi name and password in the WiFi SSID and WiFi Password fields.



Open the micro-ROS example-app settings. The Ros domain id of the micro-ROS defaults to 20. If multiple users are using it at the same time in the LAN, the parameters can be modified to avoid conflicts. Ros namespace of the micro-ROS is empty by default and does not need to be modified under normal circumstances. If non-empty characters (within 10 characters) are modified, the namespace parameter will be added before the node and topic.



After modification, press S to save, and then press Q to exit the configuration tool.

Compile, flash, and open the serial port simulator.

```
idf.py build flash monitor
```

If you need to exit the serial port simulator, press **Ctrl+]**.

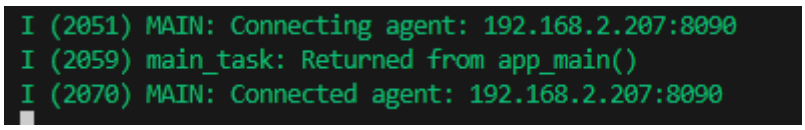
5. Experimental results

After powering on, ESP32 tries to connect to the WiFi hotspot, and then tries to connect to the proxy IP and port.

If the agent is not turned on in the virtual machine/computer terminal, please enter the following command to turn on the agent. If the agent is already started, there is no need to start the agent again.

```
docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --net=host
microros/micro-ros-agent:humble udp4 --port 8090 -v4
```

After the connection is successful, a node and a publisher are created.



At this time, you can open another terminal in the virtual machine/computer and view the /imu_publisher node.

```
ros2 node list
ros2 node info /imu_publisher
```

```
[redacted]:~$ ros2 node list
/imu_publisher
[redacted]:~$ ros2 node info /imu_publisher
/imu_publisher
  Subscribers:

  Publishers:
    /imu: sensor_msgs/msg/Imu
  Service Servers:

  Service Clients:

  Action Servers:

  Action Clients:
```

Subscribe to the data of the /odom_raw topic,

```
ros2 topic echo /imu
```

Press Ctrl+C to end the command

```
angular_velocity:
  x: -0.02343653328716755
  y: -0.0053264847956597805
  z: 0.0021305938716977835
angular_velocity_covariance:
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
- 0.0
linear_acceleration:
  x: 0.21653492748737335
  y: 0.1292031705379486
  z: 9.839777946472168
```

Check the frequency of the /imu topic. It is normal if it is about 20hz.

```
ros2 topic hz /imu
```

Press Ctrl+C to end the command

```
[redacted]:~$ ros2 topic hz /imu
average rate: 19.992
        min: 0.041s max: 0.060s std dev: 0.00505s window: 21
average rate: 19.905
        min: 0.037s max: 0.062s std dev: 0.00526s window: 41
average rate: 19.945
        min: 0.037s max: 0.062s std dev: 0.00496s window: 62
```