



bilibili 照祥同学

16-Comprehensive_Test_2

URL: https://github.com/ZhaoXiangBox/esp32_ros2_robot

Videos from Bilibili 照祥同学: [第十六节：整车综合开发与测试（二）](#)

目录:

1. 更新粉丝反馈的代码工程编译不成功的问题; (更换舵机库, 无感使用 / 更换ros2 wifi 初始化代码)
2. 在 15-Comprehensive_Test_1 的基础上, 新增发布3个传感器的 Topic: GPS IMU Ultrasonic;
3. 新增 ROS2 Foxy 下如何编写 像 ROS1 中的 launch 文件; (两种方式 两个模板)

1.修复仓库中代码编译不成功的问题, 具体描述文件细节见该路径下:

esp32_ros2_robot/10-esp32_foxy_ackermann_control_wifi/chapter_10.pdf
esp32_ros2_robot/14-PS4_ROS2_ESP32_2_Dof_Servo/chapter_14.pdf

2.采集 IMU 超声波 GPS 传感器的数据并发布相应的 Topic

a. 新增发布3个传感器的 Topic: GPS IMU Ultrasonic

超声波测距 Topic name: **ultrasonar** (测距值)

IMU 姿态 Topic name: **mpu6050** (三轴姿态 以四元素发布) 目前使用的 DMP 解算之后的输出, 这部分可有很大的优化空间！！！

GPS 定位 Topic name: **gps** (GPS 模组的定位数据 经纬度)

b.具体的解析代码可看对应的文件:

hc_sr04.h

mpu6050.h (还有很大优化空间)

ATGM336H_GPS.h

c.在 ros_driver.h 文件中新增 三种对应的 ros 代码如下:

```
...
// 头文件
#include <sensor_msgs/msg/nav_sat_fix.h>
#include <sensor_msgs/msg/range.h>
#include <sensor_msgs/msg/imu.h>

...
// 创建发布者
rcl_publisher_t imu6050_publisher;
rcl_publisher_t gps_publisher;
rcl_publisher_t hc_sr04_publisher;

// 创建定时器发布, ROS2 自带的定时触发发布话题, 固定频率发送
rcl_timer_t Ultra_timer;
rcl_timer_t Imu_timer;
rcl_timer_t Gps_timer;

// 定义发布的话题: gps 、 超声波 、 姿态传感器
sensor_msgs__msg__NavSatFix gps_msg;
sensor_msgs__msg__Range ultrasonar_msg;
sensor_msgs__msg__Imu Imu_msg;

// 定时器回调函数执行的间隔时间 ms
const unsigned int Ultra_timer_timeout = 100;      // 10 Hz
const unsigned int Imu_timer_timeout = 50;           // 20 Hz
const unsigned int Gps_timer_timeout = 200;          // 5 Hz

...
// 定时器 回调函数 , 可根据设置的时间 定时发布topic数据
void Ultra_timer_callback(rcl_timer_t * Ultra_timer, int64_t last_call_time)
{
    RCLC_UNUSED(last_call_time);
    if (Ultra_timer != NULL) {
        // update HC_SR04 data && publish.
        ultrasonar_msg.header.frame_id.data = (char*)"ultrasonar";
        ultrasonar_msg.min_range = 0.01; // 测距范围
        ultrasonar_msg.max_range = 1.02;
        unsigned long Temp = Read_HC_SR04_Data(); // 实际的测距值
        ultrasonar_msg.range = (double)(Temp / 100.0); // 单位 m
        RCSOFTCHECK(rcl_publish(&hc_sr04_publisher, &ultrasonar_msg, NULL));
        // Serial.print(" Ultrasonar Dis :");
        // Serial.println(Read_HC_SR04_Data());
    }
}

void Imu_timer_callback(rcl_timer_t * Imu_timer, int64_t last_call_time)
{
    RCLC_UNUSED(last_call_time);
```

```

if (Imu_timer != NULL) {
    Imu_msg.header.frame_id.data = (char*)"mpu6050";

    Imu_msg.orientation.x = q[1];      // 这里为了适配 ROS 中 IMU 姿态的表示方式 用的 四
元素
    Imu_msg.orientation.y = q[2];
    Imu_msg.orientation.z = q[3];
    Imu_msg.orientation.w = q[0];

    Imu_msg.angular_velocity.x = 0.0;
    Imu_msg.angular_velocity.y = 0.0;
    Imu_msg.angular_velocity.z = 0.0;

    Imu_msg.linear_acceleration.x = 0.0;
    Imu_msg.linear_acceleration.y = 0.0;
    Imu_msg.linear_acceleration.z = 0.0;

    RCSOFTCHECK(rcl_publish(&imu6050_publisher, &Imu_msg, NULL));
}

}

void Gps_timer_callback(rcl_timer_t * Gps_timer, int64_t last_call_time)
{
    RCLC_UNUSED(last_call_time);
    if ((Gps_timer != NULL) && (Save_Data.isUsefull)) {
        double latitude_temp = (Save_Data.N_S[0] == 'N') ?
atof(Save_Data.latitude) : -atof(Save_Data.latitude) ;
        double longitude_temp = (Save_Data.E_W[0] == 'E') ?
atof(Save_Data.longitude) : -atof(Save_Data.longitude) ;

        gps_msg.header.frame_id.data = (char*)"gps";
        gps_msg.latitude = latitude_temp / 100.0 ;      // 纬度 默认 北纬是正 (度.分秒)
        gps_msg.longitude = longitude_temp / 100.0;      // 经度 默认 东经是正 (度.分秒)

        RCSOFTCHECK(rcl_publish(&gps_publisher, &gps_msg, NULL));
    }
}

...
}

void Init_ROS()
{
    ...

    RCCHECK(rclc_publisher_init_best_effort(&imu6050_publisher, &node, ROSIDL_GET_M
SG_TYPE_SUPPORT(sensor_msgs, msg, Imu), "mpu6050"));
    RCCHECK(rclc_publisher_init_best_effort(&gps_publisher,
&node, ROSIDL_GET_MSG_TYPE_SUPPORT(sensor_msgs, msg, NavSatFix), "gps"));
    RCCHECK(rclc_publisher_init_best_effort(&hc_sr04_publisher, &node, ROSIDL_GET_M
SG_TYPE_SUPPORT(sensor_msgs, msg, Range), "ultrasonar"));

    ...
    // init excuter. 1 esp32_state + 1 twist + 1 gps + 1 hc_sr04 + 1 imu_ + 1
Ultra_timer + 1 Imu_timer + 1 Gps_timer = 8
    // Serial.println(" Init ROS 4! ");
    RCCHECK(rclc_executor_init(&executor, &support.context, 8, &allocator));
    ...

    RCCHECK(rclc_executor_add_timer(&executor, &Ultra_timer));
    RCCHECK(rclc_executor_add_timer(&executor, &Imu_timer));
    RCCHECK(rclc_executor_add_timer(&executor, &Gps_timer));
}

```

```
    ...  
}
```

3.ROS2 中 launch 文件的写法

joystick_control_esp32/launch/joystick_control.launch.py

```
<launch>  
  <node pkg="joy" exec="joy_node" name="joy_node" />  
  <node pkg="joystick_control_esp32" exec="joystick_node" name="joystick_node" />  
  <node pkg="rviz2" exec="rviz2" name="rviz2" />  
  <node pkg="tf2_ros" exec="static_transform_publisher"  
    name="base_link_to_ultrasonar" args="0.15 0 0 0 0 0 base_link ultrasonar" />  
  <node pkg="tf2_ros" exec="static_transform_publisher"  
    name="base_link_to_mpu6050" args="0.1 -0.03 0.1 0 0 0 base_link mpu6050" />  
  <node pkg="tf2_ros" exec="static_transform_publisher" name="base_link_to_gps"  
    args="0.12 0 0.1 0 0 0 base_link gps" />  
</launch>
```

joystick_control_esp32/launch/esp32_ros2.launch.py

```
from launch import LaunchDescription  
from launch_ros.actions import Node  
  
def generate_launch_description():  
    return LaunchDescription([  
        Node(  
            package='rviz2',  
            executable='rviz2',  
            name='rviz2'  
        ),  
  
        Node(  
            package='joystick_control_esp32',  
            executable='joystick_node',  
            name='joystick_node'  
        ),  
  
        Node(  
            package='joy',  
            executable='joy_node',  
            name='joy_node'  
        ),  
  
        Node(  
            package='tf2_ros',  
            executable='static_transform_publisher',  
            arguments = ["0.15", "0", "0", "0", "0", "0", "base_link",  
"ultrasonar"],  
            name='ultrasonar_tf'
```

```

        ) ,

        Node(
            package='tf2_ros',
            executable='static_transform_publisher',
            arguments = ["0.1", "-0.03", "0.1", "0", "0", "0", "base_link",
"mpu6050"],
            name='mpu6050_tf'
        ) ,
        Node(
            package='tf2_ros',
            executable='static_transform_publisher',
            arguments = ["0.12", "0", "0.1", "0", "0", "0", "base_link",
"gps"],
            name='gps_tf'
        ) ,
    ]
)

```

CMakeLists.xml 文件需要增加如下内容：

```

...
find_package(tf2_ros REQUIRED)
find_package(tf2 REQUIRED)
...
install(DIRECTORY launch DESTINATION share/${PROJECT_NAME}/)

```

最后，**切记**：每次修改完毕 launch 文件之后，都需要在终端中进入你的工作空间 重新编译 并 source 之后，才生效！！！

如我的工作空间目录如下：

```

micro_ros_ws
└── src
    ├── joystick_control_esp32
    ├── micro_ros_setup
    ├── ros2.repos
    └── uros

```

那么 我打开终端之后：

```

cd microros_ws
colcon build
source install/setup.bash

```

具体的细节请看代码和对应的展示视频！

