

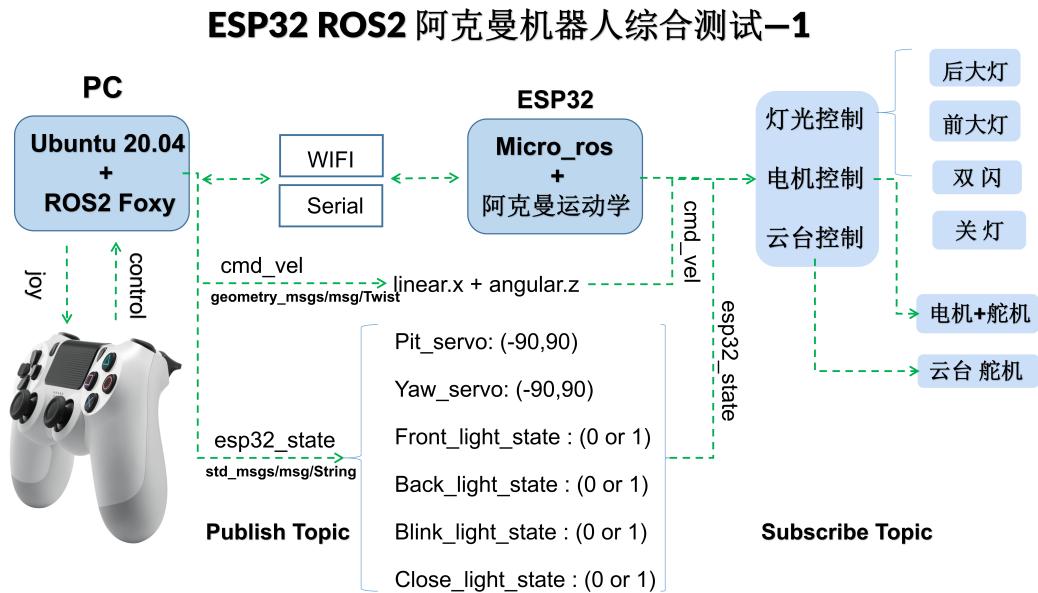


bilibili 照祥同学

15-Comprehensive_Test_1

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

Videos from Bilibili 照祥同学: [第十五节：整车综合开发与测试\(一\)](#)

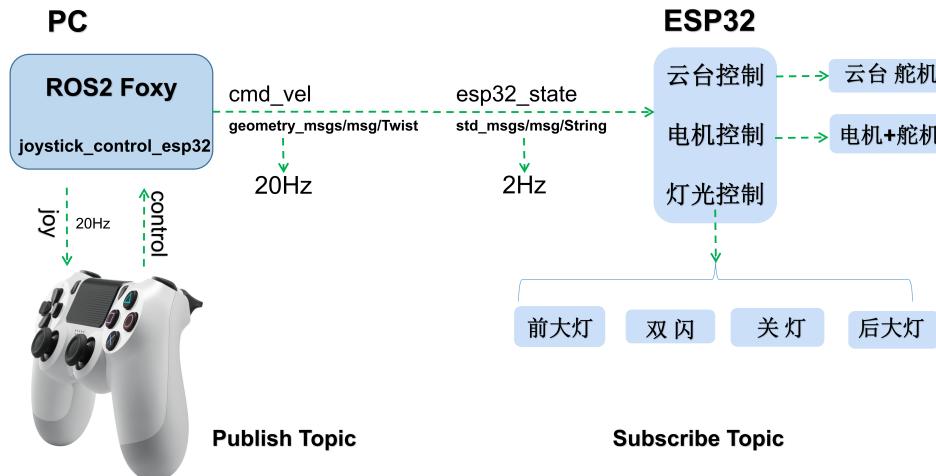


这里将前面章节中的灯光控制和云台舵机控制（ $4+2=6$ 个topic）合并成为了：**esp32_state** 一个。该topic使用的数据类型是：**std_msgs/msg/String**（字符串类型），如上图中所示，该类型中包含了6种信息。

另外，为了方便控制和使用，这里仍然保留了 **ESP32** 端订阅 **cmd_vel** 速度话题。

所以，按照最初的设想，**ESP32 ROS2** 小车，其需要订阅的消息一共就两个：**cmd_vel + esp32_state**。

ESP32 ROS2 阿克曼机器人综合测试-1



为了节约ESP32的处理器资源，提高ESP32通过WIFI订阅Topic的响应能力，对之前的joystick_control节点程序做了更改，其中，需要按下手柄的L2按键之后，摇杆控制的车速才会被使能，否则一直发布的速度指令为0，且速度话题：“cmd_vel”的发布频率和手柄的信号的采集频率一致，为20Hz；小车状态话题：“esp32_state”，通过ROS2的定时器发布，发布的时间间隔是500ms，即2Hz。

First: New joystick_control_esp32 ROS2 node

ESP32 micro_ros 之字符串合并

std_msgs/msg/String

位置	1	2	3	4	5	6	7	8	9	10	11
内容	pit_servo	空格	yaw_servo	空格	Front_light_stat	空格	Back_light_state	空格	Blink_light_state	空格	Close_light_state

ROS2 字符串合并:

joystick_control_esp32/src/joy_control.cpp :

使用空格字符作为间隔

```
auto esp32_state_msg = std::msg::String();
esp32_state_msg.data = std::to_string(esp32_state.pit_servo) + " " + std::to_string(esp32_state.yaw_servo) + " " +
    std::to_string(esp32_state.Front_light_state) + " " + std::to_string(esp32_state.Back_light_state) + " " +
    std::to_string(esp32_state.Blink_light_state) + " " + std::to_string(esp32_state.Close_light_state);
```

esp32_state话题中的内容为6个字符串，字符串之间使用字符空格作为间隔符号，在之前的基础上，更改了ROS2功能包joystick_control节点中joy话题回调函数的内容。如上图所示，在将手柄的值重映射之后，通过使用标准库中的：std::to_string()函数将值转换成字符串，相邻两个字符串使用字符空格' '间隔，依次连接组成一个完整的字符串，然后通过ROS2 topic的形式发布出去。

Tips: 看了前面章节的同学，记得从本章节文件夹中的 ROS_node 替换之前的 joystick_control。

Second : Split the String

ESP32 micro_ros 之字符串拆分

std_msgs/msg/String

位置	1	2	3	4	5	6	7	8	9	10	11
内容	pit_servo	空格	yaw_servo	空格	Front_light_stat	空格	Back_light_state	空格	Blink_light_state	空格	Close_light_state

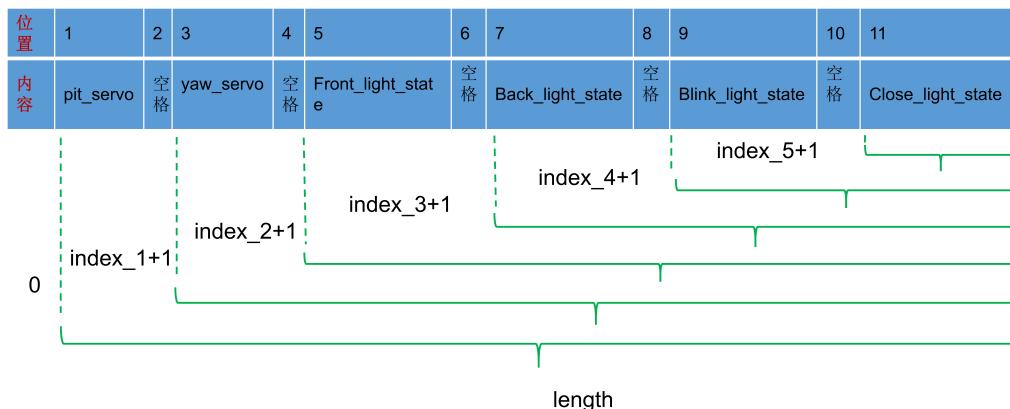
ESP32 字符串拆分示例：

```
String str // 使用ESP32 的字符串对象来操作订阅的字符串 std_msgs/msg/String  
str.length() // 求解整个字符串的长度  
str.indexOf(' ') //求解字符' ' 在字符串str中的位置  
String str2 = str.substring(0,index_1) //将str字符串的第0 - index_1 的字符拷贝到str2中  
str.toInt() // 字符串转整型值
```

1. 上图中提到的是几个 ESP32 Arduino IDE 所支持的字符串处理的 API

ESP32 micro_ros 之字符串拆分

std_msgs/msg/String



2. 上图展示了整个字符串拆分的逻辑和顺序，其中每一项所代表的意义分别如下

pit_servo: 云台 pitch 方向的目标角度, int类型, 范围: 【-90, 90】 ;

yaw_servo: 云台 yaw 方向的目标角度, int类型, 范围: 【-90, 90】 ;

Front_light_state: 前大灯是否打开, bool 类型, 范围: true or false ;

Back_light_state: 后大灯是否打开, bool 类型, 范围: true or false ;

Blink_light_state: 前后双闪是否打开, bool 类型, 范围: true or false ;

Close_light_state: 是否关闭所有灯光, bool类型, 范围: true or false ;

3. ESP32 端的字符串拆分代码如下

```
void subs_Esp32_state_callback(const void *msgin){
    const std_msgs_msg_String * msg = (const std_msgs_msg_String *)msgin;
    // Serial.println(msg->data.data);
    String Servos_str;
    String Sub_temp_str;
    String Sub_End_str;

    // 字符串按照 空格 拆分
    Servos_str = msg->data.data;
    int length = Servos_str.length(); // 记录字符串的总长度 字节数目

    int index_1 = Servos_str.indexOf(' '); // 第一个空格字符的 标号
    Sub_temp_str = Servos_str.substring(0, index_1);
    esp32_state.Yaw_Servo_Data = 90 + Sub_temp_str.toInt();
    Sub_End_str = Servos_str.substring(index_1+1, length); // 将空格之后的字符串 拷贝给
    sub_End_str

    int index_2 = Sub_End_str.indexOf(' ');
    Sub_temp_str = Sub_End_str.substring(0, index_2);
    esp32_state.Pitch_Servo_Data = 90 + Sub_temp_str.toInt();
    Sub_End_str = Servos_str.substring(index_1 + 1 + index_2 + 1 ,length);

    int index_3 = Sub_End_str.indexOf(' ');
    Sub_temp_str = Sub_End_str.substring(0, index_3);
    esp32_state.Front_light_state = Sub_temp_str.toInt();
    Sub_End_str = Servos_str.substring(index_1 + 1 + index_2 + 1 + index_3 + 1
    ,length);

    int index_4 = Sub_End_str.indexOf(' ');
    Sub_temp_str = Sub_End_str.substring(0, index_4);
    esp32_state.Back_light_state = Sub_temp_str.toInt();
    Sub_End_str = Servos_str.substring(index_1 + 1 + index_2 + 1 + index_3 + 1
    + index_4 + 1 ,length);

    int index_5 = Sub_End_str.indexOf(' ');
    Sub_temp_str = Sub_End_str.substring(0, index_5);
    esp32_state.Blink_light_state = Sub_temp_str.toInt();
    Sub_End_str = Servos_str.substring(index_1 + 1 + index_2 + 1 + index_3 + 1
    + index_4 + 1 + index_5 + 1 ,length);

    esp32_state.Close_light_state = Sub_End_str.toInt();

    ...
}
```

4. 定义了一个结构体类型用于存储 esp32_state 被解析的数据

```
typedef struct {
    int Yaw_Servo_Data = 90;
    int Pitch_Servo_Data = 90;

    bool Front_light_state = false;
    bool Back_light_state = false;
    bool Blink_light_state = false;
    bool Close_light_state = false;
}ESP32_STATE;
ESP32_STATE esp32_state;
```

本实验的整个架构图如上，详细的操作视频请看文档开头附带的视频链接，代码已开源，链接如视频详情。