

# 基于T-bot底盘实验指导书---话题订阅

## 1.考核目标

### 1.1 实验原理

本实验主要考核消息的订阅功能的理解与应用。在 ROS（机器人操作系统）中，机器人通过消息传递进行节点之间的通信。通过小乌龟实验，我们了解到通过键盘控制小乌龟的运动，其核心在于 ROS 系统中存在一个节点 `teleop_twist_keyboard`，产生了话题 `/cmd_vel`。小乌龟订阅了 `/cmd_vel` 话题后，能够根据接收到的速度指令进行运动。因此，通过订阅 `/cmd_vel` 消息，我们可以实现机器人的运动控制。

- **结点运行**

输入如下指令

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

启动键盘控制结点，该结点是系统默认自带的，不需要我们去编写这个结点。

- **消息类型**

单独打开终端，通过查看 `/cmd_vel`，终端中输入 `ros2 topic info /cmd_vel` 获得该话题的类型

```
Type: geometry_msgs/msg/Twist
```

```
Publisher count: 1
```

```
Subscription count: 0
```

关于twist指令可以参考[twist官方解释](#)

[🏠](#) / [Message Definitions](#) / [Twist](#)

[View page source](#)

## Twist 🔗

This is a ROS message definition.

### Source

```
# This expresses velocity in free space broken into its linear and angular parts.
```

```
Vector3 linear
```

```
Vector3 angular
```

## 1.2 实验目标

- **创建消息订阅者**：制作一个 ROS 节点，订阅 `/cmd_vel` 消息。
- **实现小车移动控制**：根据接收到的速度和角度指令，实现小车的移动控制。
- **验证响应能力**：测试小车在不同速度和方向命令下的响应能力，确保其能够按预期进行移动。
- **数据记录与分析**：记录实验过程中小车的运动数据，以便进行后续分析。

## 2.实验步骤

### 2.1创建包

首先在/home/pi目录下创建一个文件夹，名称自拟，打开终端，输入如下指令

```
mkdir -p xxx/src  
cd xxx/src
```

其中xxx为自建文件夹

在home/pi/xxx/src 目录创建一个名为 my\_base 的包：

```
ros2 pkg create --build-type ament_cmake my_base
```

### 2.2编写订阅者节点代码

新建一个cpp文件，命名为my\_base.cpp,文件引用头文件bot\_serial.h, my\_base.cpp和bot\_serial.h文件内容可以参考~/wangwei/src/my\_base/src文件夹下的文件，当然你也可以直接复制过去。

接下来分析相关文件的具体功能：

该头文件中定义了下发指令的结构体和基本类

```

#define SEND_DATA_CHECK    1
#define READ_DATA_CHECK    0
#define FRAME_HEADER        0X7B        //Frame head
#define FRAME_TAIL          0X7D        //Frame tail
#define RECEIVE_DATA_SIZE  24
#define SEND_DATA_SIZE      11
#define PI                    3.1415926f //PI //圆周率
typedef struct _SEND_DATA_
{
    uint8_t tx[SEND_DATA_SIZE];
    float X_speed;
    float Y_speed;
    float Z_speed;
    unsigned char Frame_Tail;
}SEND_DATA;

```

该结构体描述了ROS系统下发指令的详细细节，分析可知，控制X轴、y轴、Z轴速度的下发便可以完成移动底盘运动的控制。

```

class turn_on_robot : public rclcpp::Node
{
public:
    turn_on_robot(); //Constructor //构造函数
    ~turn_on_robot(); //Destructor //析构函数
    void Control(); //Loop control code //循环控制代码
    serial::Serial Stm32_Serial; //Declare a serial object //声明串口对象
private:
    rclcpp::Time _Now, _Last_Time; //Time dependent, used for
    float Sampling_Time;
    RECEIVE_DATA Receive_Data;
    SEND_DATA Send_Data;
    bool Get_Sensor_Data_New();
    unsigned char Check_Sum(unsigned char Count_Number,unsigned char mode);
    //check function //校验函数
    short IMU_Trans(uint8_t Data_High,uint8_t Data_Low);
    //IMU data conversion read //IMU数据转化读取
    float Odom_Trans(uint8_t Data_High,uint8_t Data_Low);
    //Odometer data is converted to read //里程计数据转化读取
};

```

为了建立底盘与ROS系统之间的联系，有必要建立一个类，并在内中定义串口

接 serial::Serial Stm32\_SerialStm32\_Serial ,其中 Get\_senosr\_Data\_New() 函数与 Check\_Sum() 函数不属于考核点，老师已经实现了，直接调用即可。学生需要完成类的构造函数

数 turn\_on\_robot() 和 Control() 函数

my\_base.c主要由四个函数，请将以下四个函数拷贝到文件中，并实现缺失的代码。

- **函数1**

```
#include "bot_serial.h"
int main(int argc, char **argv) {
    rclcpp::init(argc, argv);
    turn_on_robot Robot_Control;//Instantiate an object //实例化一个对象
    Robot_Control.Control();
    rclcpp::shutdown();
    return 0;
}
```

- **函数2**

可见，我们同时需要在my\_base.c文件中需要实现Robot\_Control.Control()函数，该函数的模板如下：

```
void turn_on_robot::Control()
{
    while(rclcpp::ok())
    {
        try
        {
            rclcpp::spin_some(this->get_node_base_interface());
        }
        catch (const rclcpp::exceptions::RCLError & e )
        {
            RCLCPP_ERROR(this->get_logger(),"unexpectedly failed with %s",e.what());
        }
    }
}
```

- **函数3**

我们需要重点完成的是订阅完消息后的回调函数如何实现

```

void turn_on_robot::Cmd_Vel_Callback(const geometry_msgs::msg::Twist::SharedPtr twist_aux)
{
    Send_Data.tx[0]=FRAME_HEADER; //frame head 0x7B
    Send_Data.tx[2] = 0; //set aside //预留位
    printf("%f:%f:%f", twist_aux->linear.x,twist_aux->linear.x,twist_aux->angular.z);
    RCLCPP_INFO(this->get_logger(),"cmd is ready");
    //The target velocity of the X-axis of the robot
    //机器人x轴的目标线速度,请填写代码

    //The target velocity of the Y-axis of the robot
    //机器人y轴的目标线速度, 请填写代码

    //The target angular velocity of the robot's Z axis
    //机器人z轴的目标角速度请填写代码

    // Send_Data.tx[9]=Check_Sum(9,SEND_DATA_CHECK);
    // Send_Data.tx[10]=FRAME_TAIL; //frame tail 0x7D
    // try
    // {
    //     Stm32_Serial.write(Send_Data.tx,sizeof (Send_Data.tx));
    // }
    // catch (serial::IOException& e)
    // {
    //     RCLCPP_ERROR(this->get_logger(),"Unable to send data through serial port");
    // }
}

```

#### • 函数4

```

turn_on_robot::turn_on_robot():rclcpp::Node ("wheeltec_robot")
{

    Cmd_Vel_Sub = create_subscription<geometry_msgs::msg::Twist>(
    "cmd_vel", 2, std::bind(&turn_on_robot::Cmd_Vel_Callback, this, _1));
    RCLCPP_INFO(this->get_logger(),"wheeltec_robot Data ready"); //Prompt message //提示信息
    // try
    // {
    //     Stm32_Serial.setPort("/dev/ttyACM0"); //Select the serial port number to enable //选择
    //     Stm32_Serial.setBaudrate(115200); //Set the baud rate //设置波特率
    //     serial::Timeout _time = serial::Timeout::simpleTimeout(2000); //Timeout //超时等待
    //     Stm32_Serial.setTimeout(_time);
    //     Stm32_Serial.open(); //Open the serial port //开启串口
    // }
    // catch (serial::IOException& e)
    // {
    //     RCLCPP_ERROR(this->get_logger(),"wheeltec_robot can not open serial port,Please check
    // }
    // if(Stm32_Serial.isOpen())
    // {
    //     RCLCPP_INFO(this->get_logger(),"wheeltec_robot serial port opened"); //Serial port opened
    // }
}

```

## 2.3修改package.xml

进入ros2\_ws/src/my\_base目录并打开package.xml，按照之前教程要求填写description，maintainer和license。如果你并不想开源你的代码，可以忽略此步。

```

<description>TODO: Package description</description>
<maintainer email="nxrobo@todo.todo">nxrobo</maintainer>
<license>TODO: License declaration</license>

```

在编译工具依赖ament\_cmake后

```

<buildtool_depend>ament_cmake</buildtool_depend>

```

添加下列依赖项：

```
<depend>rclcpp</depend>  
<depend>std_msgs</depend>
```

改写完毕后注意记得保存文档!

## 2.4修改CmakeLists.txt

打开 CMakeLists.txt , 在 find\_package(ament\_cmake REQUIRED) 下添加两行:

```
find_package(rclcpp REQUIRED)  
find_package(geometry_msgs REQUIRED)  
find_package(tf2_geometry_msgs REQUIRED)  
find_package(serial REQUIRED)  
  
add_executable(my_base src/my_base.cpp)  
ament_target_dependencies(my_base rclcpp serial tf2_geometry_msgs)
```

## 3.编译运行

打开终端, 执行如下命令:

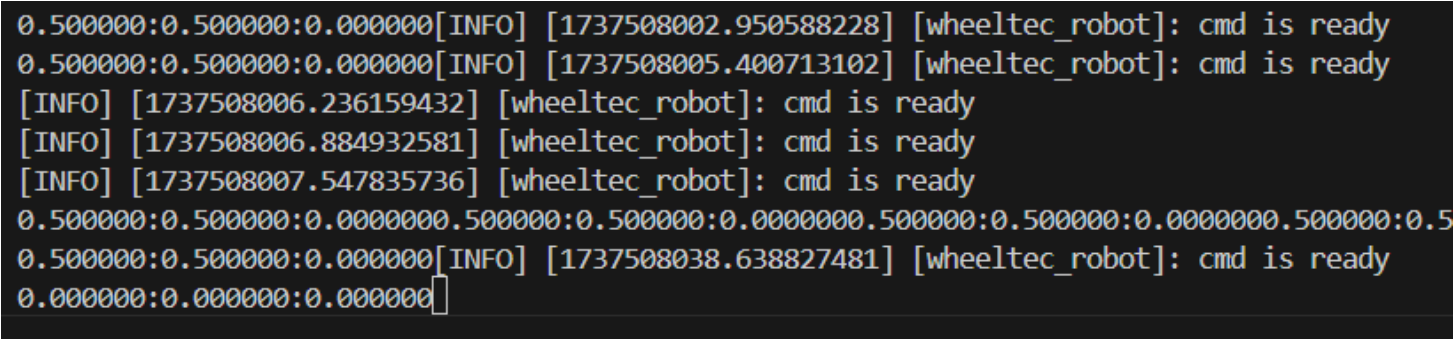
执行 colcon build

执行 source install/setup.bash

执行 ros2 run my\_base my\_base

## 4.测试验证

上述步骤正确, 再打开终端, 启动键盘节点, 输入指令, 则my\_base节点会响应, 如图所示:



```
0.500000:0.500000:0.000000[INFO] [1737508002.950588228] [wheeltec_robot]: cmd is ready  
0.500000:0.500000:0.000000[INFO] [1737508005.400713102] [wheeltec_robot]: cmd is ready  
[INFO] [1737508006.236159432] [wheeltec_robot]: cmd is ready  
[INFO] [1737508006.884932581] [wheeltec_robot]: cmd is ready  
[INFO] [1737508007.547835736] [wheeltec_robot]: cmd is ready  
0.500000:0.500000:0.000000.500000:0.500000:0.000000.500000:0.500000:0.000000.500000:0.5  
0.500000:0.500000:0.000000[INFO] [1737508038.638827481] [wheeltec_robot]: cmd is ready  
0.000000:0.000000:0.000000
```

## 5.回调函数中完成代码整合

如图所示，完成代码编写

```
void turn_on_robot::Cmd_Vel_Callback(const geometry_msgs::r
{
    Send_Data.tx[0]=FRAME_HEADER; //frame head 0x7B
    Send_Data.tx[2] = 0; //set aside //预留位
    printf("%f:%f:%f", twist_aux->linear.x,twist_aux->linear.
        RCLCPP_INFO(this->get_logger(),"cmd is ready");
    //The target velocity of the X-axis of the robot
    //机器人x轴的目标线速度,请填写代码

    //The target velocity of the Y-axis of the robot
    //机器人y轴的目标线速度,请填写代码

    //The target angular velocity of the robot's Z axis
    //机器人z轴的目标角速度请填写代码
```

## 6.实现通过键盘操作小车的目的