# КАК ПРИНЯТЬ И ОТПРАВИТЬ СООБЩЕНИЕ В ТОПИКЕ

## 1. Через команду ros2 topic pub (в терминале)

Это самый простой способ для тестирования — публикация сообщения из командной строки.

Синтаксис:

ros2 topic pub <топик> <тип\_сообщения> "<значения\_в\_yaml>"

Пример: публикация сообщения типа std\_msgs/msg/String

ros2 topic pub /chatter std\_msgs/msg/String "data: 'Hello from terminal!'"

ROS 2 автоматически опубликует одно сообщение и завершит работу. Чтобы публиковать постоянно с заданной частотой, добавьте флаг -r (rate):

ros2 topic pub -r 1 /chatter std\_msgs/msg/String "data: 'Hello every second!'" # публикует 1 раз в секунду

Пример: публикация geometry\_msgs/msg/Twist (часто используется для управления роботом)

ros2 topic pub /cmd\_vel geometry\_msgs/msg/Twist "linear:  
 x: 0.5  
 y: 0.0  
 z: 0.0  
angular:  
 x: 0.0  
 y: 0.0  
 z: 0.2"

Обратите внимание на отступы — используется **YAML-формат**, поэтому пробелы важны!

## 2. Из Python-ноды (с помощью rclpy)

Если нужно интегрировать в код:

import rclpy  
from rclpy.node import Node  
from std\_msgs.msg import String  
  
class MinimalPublisher(Node):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_('minimal\_publisher')  
 self.publisher\_ = self.create\_publisher(String, 'chatter', 10)  
 timer\_period = 1.0 # секунды  
 self.timer = self.create\_timer(timer\_period, self.timer\_callback)  
  
 def timer\_callback(self):  
 msg = String()  
 msg.data = 'Hello from Python node!'  
 self.publisher\_.publish(msg)  
 self.get\_logger().info(f'Publishing: "{msg.data}"')  
  
def main(args=None):  
 rclpy.init(args=args)  
 node = MinimalPublisher()  
 rclpy.spin(node)  
 node.destroy\_node()  
 rclpy.shutdown()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Запуск:

python3 publisher\_example.py