Основы ROS2

2. 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)

```
def main(args=None):
    rclpy.init(args=args)
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"
```

> ⚠ Обрати 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
```
