

ЧТО ТАКОЕ ПУБЛИКАТОР И ПОДПИСЧИК?

- **Публикатор (Publisher)** — это узел (node), который отправляет сообщения в определённую тему (**topic**).
- **Подписчик (Subscriber)** — это узел, который **слушает** определённую тему и получает сообщения, которые туда публикуются.

Структура проекта

Все проекты в ROS2 лежат в рабочих областях - папках с постфиксом "_ws".

Создадим такую папку и перейдем в неё.

```
mkdir my_project_ws  
cd my_project_ws
```

Созданные Вами файлы всегда обязательно лежат в папке **src**, создадим её и перейдем в неё. Теперь мы можем создать наш ROS2 пакет внутри.

```
mkdir src  
cd src
```

Создадим простой ROS 2 пакет:

```
ros2 pkg create --build-type ament_python my_py_pubsub
```

Это создаст папку **my_py_pubsub** с базовой структурой Python-пакета ROS

Затем создадим два файла в папке **my_py_pubsub/my_py_pubsub/**:

- **publisher_member_function.py**
- **subscriber_member_function.py**

Публикатор (Publisher)

Код: publisher_member_function.py

```
import rclpy                                     # Основная библиотека ROS 2 для Python  
from rclpy.node import Node                     # Базовый класс для создания узлов
```

```

from std_msgs.msg import String # Импортируем тип сообщения String из станд
артных сообщений

class MinimalPublisher(Node):    # Создаём класс, наследуемый от Node

    def __init__(self):
        super().__init__('minimal_publisher') # Инициализируем узел с имен
ем 'minimal_publisher'
        self.publisher_ = self.create_publisher(String, 'topic', 10) # Соз
даём публикатор
        timer_period = 0.5 # секунды
        self.timer = self.create_timer(timer_period, self.timer_callback)
# Таймер вызывает callback каждые 0.5 сек
        self.i = 0 # Счётчик сообщений

    def timer_callback(self):
        msg = String() # Создаём объект сообщения типа
String
        msg.data = f'Hello World: {self.i}' # Заполняем поле data
        self.publisher_.publish(msg) # Публикуем сообщение
        self.get_logger().info(f'Publishing: "{msg.data}"') # Выводим в ко
нsole (лог)
        self.i += 1 # Увеличиваем счётчик

def main(args=None):
    rclpy.init(args=args) # Инициализация ROS 2
    minimal_publisher = MinimalPublisher() # Создаём экземпляр нашего узла
    rclpy.spin(minimal_publisher) # Запускаем "цикл обработки" узла
    minimal_publisher.destroy_node() # Очищаем ресурсы
    rclpy.shutdown() # Завершаем работу ROS 2

```

```
if __name__ == '__main__':
    main()
```

Пояснение по строкам:

Строка	Объяснение
import rclpy	Подключаем основную библиотеку ROS 2 для Python.
from rclpy.node import Node	Импортируем базовый класс узла.
from std_msgs.msg import String	Используем стандартный тип сообщения — строку.
class MinimalPublisher(Node):	Наш узел будет классом, унаследованным от Node.
super().__init__('minimal_publisher')	Инициализируем узел с именем minimal_publisher (это имя будет видно в системе).
self.create_publisher(String, 'topic', 10)	Создаём публикатор: тип сообщения, имя темы (topic), размер очереди (10 сообщений).
self.create_timer(...)	Создаём таймер, который каждые 0.5 сек вызывает timer_callback.
msg = String()	Создаём новое сообщение.
msg.data = ...	Заполняем содержимое сообщения.
self.publisher_.publish(msg)	Отправляем сообщение в тему.
self.get_logger().info(...)	Выводим сообщение в терминал (как print, но с логированием ROS).
rclpy.spin(...)	Блокирует выполнение и обрабатывает входящие/исходящие сообщения.

Подписчик (Subscriber)

Код: subscriber_member_function.py

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription(
            String,          # Тип сообщения
            'topic',         # Имя темы (должно совпадать с публикатором!)
            self.listener_callback, # Функция, вызываемая при получении со
                                # общения
            10)              # Размер очереди
        self.subscription # Предотвращает предупреждение "переменная не используется"

    def listener_callback(self, msg):
        self.get_logger().info(f'I heard: "{msg.data}"')

def main(args=None):
    rclpy.init(args=args)
    minimal_subscriber = MinimalSubscriber()
    rclpy.spin(minimal_subscriber)
    minimal_subscriber.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Пояснение:

Строка	Объяснение
<code>self.create_subscription(...)</code>	Создаём подписку на тему <code>topic</code> с типом <code>String</code> .
<code>self.listener_callback</code>	Эта функция будет вызываться каждый раз, когда приходит новое сообщение.
<code>self.subscription</code>	Просто упоминаем переменную, чтобы Python не удалил её как “неиспользуемую”.
<code>listener_callback(self, msg)</code>	Получает сообщение <code>msg</code> и выводит его содержимое.

(ОБЯЗАТЕЛЬНО!) прописать точки входа в `setup.py`

```
Все узлы должны быть “зарегистрированы” в setup.py, чтобы ros2 run их находил. Пример регистрации в setup.py (в корне пакета):  
from setuptools import setup  
  
package_name = 'my_py_pubsub'  
  
setup(  
    name=package_name,  
    version='0.0.0',  
    packages=[package_name],  
    data_files=[  
        ('share/ament_index/resource_index/packages', ['resource/' + package_name]),  
        ('share/' + package_name, ['package.xml']),  
    ],  
    install_requires=['setuptools'],  
    zip_safe=True,
```

```
entry_points={
    'console_scripts': [
        'talker = my_py_pubsub.publisher_member_function:main',
        'listener = my_py_pubsub.subscriber_member_function:main',
    ],
},
)
```

Как запустить?

1. **Соберите пакет** (нужно быть в папке `my_project_ws`, на одном уровне с папкой `src`. Вы увидите `src` при вызове команды `ls`):

```
colcon build --packages-select my_py_pubsub
```

2. **Загрузите окружение:**

```
source install/local_setup.bash
```

3. **Запустите публикатор в одном терминале:**

```
ros2 run my_py_pubsub talker
```

4. **Запустите подписчик в другом терминале (не забудьте загрузить окружение во второй терминал):**

```
ros2 run my_py_pubsub listener
```

Вы увидите, как подписчик выводит сообщения, которые публикует первый узел.