# 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 стандартных сообщений  
  
class MinimalPublisher(Node): # Создаём класс, наследуемый от Node  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_('minimal\_publisher') # Инициализируем узел с именем 'minimal\_publisher'  
 self.publisher\_ = self.create\_publisher(String, 'topic', 10) # Создаём 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 консоль (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 сообщения  
 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 |
| self.create\_subscription(...) | Создаём подписку на тему topic с типом String. |
| self.listener\_callback | Эта функция будет вызываться каждый раз, когда приходит новое сообщение. |
| self.subscription | Просто упоминаем переменную, чтобы Python не удалил её как “неиспользуемую”. |
| listener\_callback(self, msg) | Получает сообщение msg 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

1. **Загрузите окружение**:

source install/local\_setup.bash

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

ros2 run my\_py\_pubsub talker

1. **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.