Single publisher and subscriber

Run in terminal
mkdir -p ~/trying/src
cd trying
cd src
ros2 pkg create --build-type ament_python py_pubsub

This files were created in trying\src\py_pubsub-

#I mistakenly put the same code in init and setup files and thus it shows me colcon.colcon core error.

init.py

```
from setuptools import setup

package_name= 'py_pubsub'

setup(
        name=package_name,
        version='0,0,0',
        packages=[package_name],
        data_files=[
        ('share/ament_index/resource_index/packages',
        ['resoure/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        ],
        install_requires=['setuptools'],
        zip_safe=True,
        maintainer='tirth',
        maintainer_email='tirth@todo.todo',
        description='TODO: Package description',
        license='TODO: License declaration',
        tests_require=['pytest'],
        entry_points={
        'console_scripts': [
                'talker = py_pubsub.publisher_member_function:main',
                'listener = py_pubsub.subscriber_member_function:main',
        ],
        },
)
```

---

publisher_member_function.py

```
from std_msgs.msg import String


class MinimalPublisher(Node):

        def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 0.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

        def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1


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

        minimal_publisher = MinimalPublisher()

        rclpy.spin(minimal_publisher)

        # Destroy the node explicitly
        # (optional - otherwise it will be done automatically
        # when the garbage collector destroys the node object)
        minimal_publisher.destroy_node()
        rclpy.shutdown()


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

---

subscriber_member_function

```
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,
        'topic',
        self.listener_callback,
        10)
        self.subscription  # prevent unused variable warning

        def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)


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

        minimal_subscriber = MinimalSubscriber()

        rclpy.spin(minimal_subscriber)

        # Destroy the node explicitly
        # (optional - otherwise it will be done automatically
        # when the garbage collector destroys the node object)
        minimal_subscriber.destroy_node()
        rclpy.shutdown()


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

---

package .xml

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>py_pubsub</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="tirth@todo.todo">tirth</maintainer>
  <license>TODO: License declaration</license>
  <exec_depend>rclpy</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

  <export>
        <build_type>ament_python</build_type>
  </export>
</package>
```

---

setup.cfg

```
[develop]
script-dir=$base/lib/py_pubsub
[install]
install-scripts=$base/lib/py_pubsub
```

---

setup.py

```
from setuptools import setup

package_name = 'py_pubsub'

setup(
        name=package_name,
        version='0.0.0',
        packages=[package_name],
        data_files=[
```

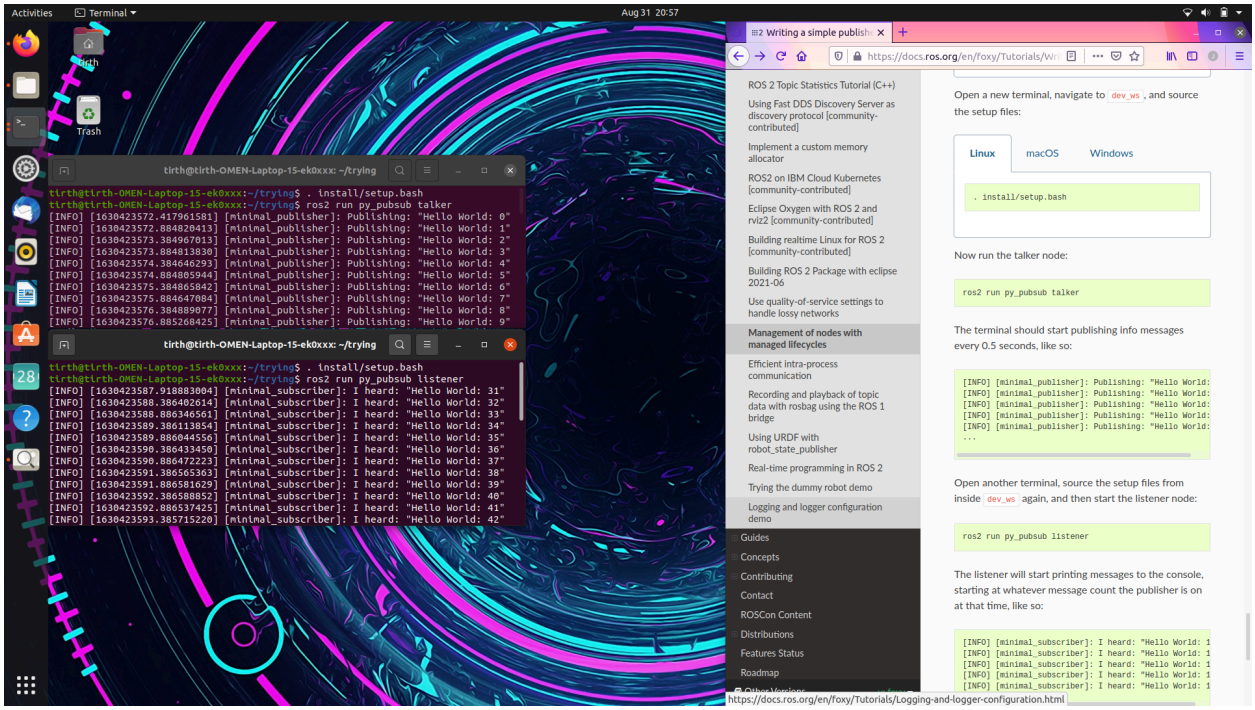
```
        ('share/ament_index/resource_index/packages',
        ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        ],
        install_requires=['setuptools'],
        zip_safe=True,
        maintainer='tirth',
        maintainer_email='tirth@todo.todo',
        description='TODO: Package description',
        license='TODO: License declaration',
        tests_require=['pytest'],
        entry_points={
        'console_scripts': [
                'talker = py_pubsub.publisher_member_function:main',
                'listener = py_pubsub.subscriber_member_function:main',
        ],
        },

)
```

Now building the package

colcon build --packages-select py_pubsub

RUN THIS IN TWO TERMINALS-
. install/setup.bash
ros2 run py_pubsub talker

. install/setup.bash
ros2 run py_pubsub listener



++

30/09/21

Multiple (Publisher and subscriber)-

Run in terminal

mkdir -p ~/trying/src

cd trying

cd src

ros2 pkg create --build-type ament_python py_pubsub

This filer were created in trying\src\py_pubsub-

---

publisherb_member_function.py

```
import rclpy
from rclpy.node import Node

from std_msgs.msg import String


class MinimalPublisherAA(Node):

        def __init__(self):
        super().__init__('minimal_publisherAA')
        self.publisherAA_ = self.create_publisher(String, 'topicb', 10)
        timer_period = 1.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

        def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisherAA_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1


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

        minimal_publisherAA = MinimalPublisherAA()

        rclpy.spin(minimal_publisherAA)

        # Destroy the node explicitly
        # (optional - otherwise it will be done automatically
        # when the garbage collector destroys the node object)
        minimal_publisherAA.destroy_node()
        rclpy.shutdown()


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

---

publisher_member_function.py

```
import rclpy
from rclpy.node import Node

from std_msgs.msg import String

#publisher1
class MinimalPublisher(Node):

        def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 1.0  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

        def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1

        # subscriber2
class MinimalSubscriberAA(Node):

        def __init__(self):
        super().__init__('minimal_subscriberAA')
        self.subscription = self.create_subscription(
        String,
        'topicb',
        self.listener_callback,
        10)
        self.subscription  # prevent unused variable warning

        def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)


def main(args=None):
        rclpy.init(args=args)
        minimal_publisher = MinimalPublisher()
        minimal_subscriberAA = MinimalSubscriberAA()
```

```
        while(2):
        rclpy.spin_once(minimal_publisher)
        #timer_period = 0.5
        rclpy.spin_once(minimal_subscriberAA)


        minimal_publisher.destroy_node()
        rclpy.shutdown()
        minimal_subscriberAA.destroy_node()
        rclpy.shutdown()

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

---

package.xml

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>py_pubsub</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="tirth@todo.todo">tirth</maintainer>
  <license>TODO: License declaration</license>
  <exec_depend>rclpy</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

  <export>
        <build_type>ament_python</build_type>
  </export>
</package>
```

---

setup.cfg

```
[develop]
script-dir=$base/lib/py_pubsub
[install]
install-scripts=$base/lib/py_pubsub
```

---

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,
        'topic',
        self.listener_callback,
        10)
        self.subscription  # prevent unused variable warning

        def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)


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

        minimal_subscriber = MinimalSubscriber()

        rclpy.spin(minimal_subscriber)

        # Destroy the node explicitly
        # (optional - otherwise it will be done automatically
        # when the garbage collector destroys the node object)
        minimal_subscriber.destroy_node()
        rclpy.shutdown()


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

---

setup.py

```
from setuptools import setup

package_name = '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,
        maintainer='tirth',
        maintainer_email='tirth@todo.todo',
        description='TODO: Package description',
        license='TODO: License declaration',
        tests_require=['pytest'],
        entry_points={
        'console_scripts': [
                'both = py_pubsub.publisher_member_function:main',
                'listener = py_pubsub.subscriber_member_function:main',
                'talker = py_pubsub.publisherb_member_function:main',
        ],
        },
)
```

AFTER colcon build
Sir code ban gaya hai 2 publisher and 2 subscriber ka

Publisher 1---subscriber1
Publisher 2---subscriber2

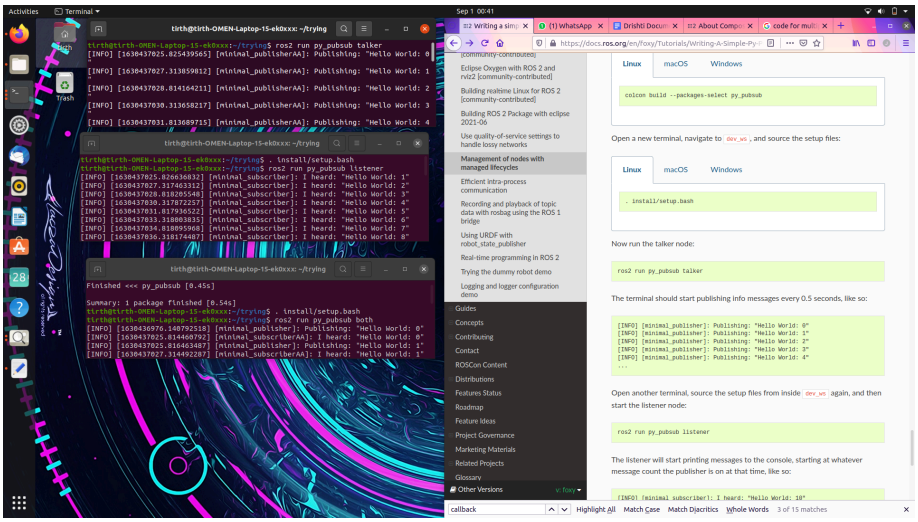
Publisher 1 subscriber2 jo hai wo same xx.py file ma hai
Publisher 2 alag sa hai xy.py file ma hai
And subscriber1 alag sa yy.py file ma hai

minimal_publisher is publisher1

minimal_subscriber is subscriber1

minimal_publisherAA is publisher2

minimal_subscriberAA is subscriber2



++
01/10/21

Multiple (Publisher and subscriber)-

One of the publishers should be publishing fibonacci series and the topics for both the pairs should be different.

Run in terminal
mkdir -p ~/trying/src
cd trying
cd src
ros2 pkg create --build-type ament_python py_pubsub

This filer were created in trying\src\py_pubsub

Publisher 2 code

```
import rclpy
from rclpy.node import Node

from std_msgs.msg import Float64
```

```
class MinimalPublisherAA(Node):

        def __init__(self):
        super().__init__('minimal_publisherAA')
        self.publisherAA_ = self.create_publisher(Float64, 'topicb', 10)
        timer_period = 1.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        print('*********publisherB*********')
        self.x = 0.0
        self.y = 0.0
        self.a = 0.0
        self.b = 1.0
        self.i = 0.0


        def timer_callback(self):

        if self.x == 0.0:
        self.i = 0.0
        self.x = self.x + 1.0
        self.y = self.y + 1.0

        elif self.y == 1.0:
        self.i = 1.0
        self.y = self.y + 1.0

        else:
        self.i = self.a + self.b
        self.a = self.b
        self.b = self.i
        #'Terms in Fibonacci series: %d' %
        msg = Float64()
        msg.data = self.i
        self.publisherAA_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)


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

        minimal_publisherAA = MinimalPublisherAA()

        rclpy.spin(minimal_publisherAA)
```

```
        minimal_publisherAA.destroy_node()
        rclpy.shutdown()


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

Publisher1 & Subscriber2

```
import rclpy
from rclpy.node import Node

from std_msgs.msg import String
from std_msgs.msg import Float64

#publisher1
class MinimalPublisher(Node):

        def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 1.0  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0
        print('*********publisherA*********')

        def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1


#subscriber2
class MinimalSubscriberAA(Node):

        def __init__(self):
        super().__init__('minimal_subscriberAA')
        self.subscription = self.create_subscription(
        Float64,
        'topicb',
        self.listener_callback,
```

```
        10)
        self.subscription  # prevent unused variable warning
        print('*********subscriberB*********')

        def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)


def main(args=None):
        rclpy.init(args=args)
        minimal_publisher = MinimalPublisher()
        minimal_subscriberAA = MinimalSubscriberAA()

        while(2):
        rclpy.spin_once(minimal_publisher)
        #timer_period = 0.5
        rclpy.spin_once(minimal_subscriberAA)



        minimal_publisher.destroy_node()
        rclpy.shutdown()
        minimal_subscriberAA.destroy_node()
        rclpy.shutdown()


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

Subscriber1

```
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,
        'topic',
```

```
        self.listener_callback,
        10)
        self.subscription  # prevent unused variable warning
        print('*********subscriberA*********')

        def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)


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

        minimal_subscriber = MinimalSubscriber()

        rclpy.spin(minimal_subscriber)

        # Destroy the node explicitly
        # (optional - otherwise it will be done automatically
        # when the garbage collector destroys the node object)
        minimal_subscriber.destroy_node()
        rclpy.shutdown()


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

In setup.py file in console-

'both = py_pubsub.publisher_member_function:main',
'listener = py_pubsub.subscriber_member_function:main',
'talker = py_pubsub.publisherb_member_function:main',

Teleop_twist_keyboard code 1 copied from ros index

Link- https://github.com/ros2/teleop_twist_keyboard/blob/dashing/teleop_twist_keyboard.py
Topic is “cupcar0/cmd_vel” which is responsible to control movements of the car.

Teleop_twist_keyboard code 2 made by using blessed library

To install blessed library use command on virtual terminal “**sudo pip3 install blessed**”
Code

```
import sys
import geometry_msgs.msg
import rclpy
#library which is to be installed using command "sudo pip3 install
#blessed"
from blessed import Terminal
term = Terminal()

msg = """
*****************************************************
i for forward
u & o for left and right in forward
k for brake
j & l for turning wheels left and right
, for reverse
m & . for left and right in reverse
-----------------------------------------------------
q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
-----------------------------------------------------
Ctrl-C to QUIT
*****************************************************
"""

moveBindings = {
   'i': (1, 0, 0, 0),
   'o': (1, 0, 0, -1),
   'j': (0, 0, 0, 1),
   'l': (0, 0, 0, -1),
   'u': (1, 0, 0, 1),
   ',': (-1, 0, 0, 0),
   '.': (-1, 0, 0, 1),
   'm': (-1, 0, 0, -1),
   'k': (0, 0, 0, 0),
}

speedBindings = {
   'q': (1.1, 1.1),
   'z': (.9, .9),
```

```
    'w': (1.1, 1),
    'x': (.9, 1),
    'e': (1, 1.1),
    'c': (1, .9),
}

def vels(speed, turn):
    return 'currently:\tspeed %s\tturn %s ' % (speed, turn)

def main():
    rclpy.init()

    node = rclpy.create_node('teleop_twist_keyboard')
    pub = node.create_publisher(geometry_msgs.msg.Twist,
'/cupcar0/cmd_vel', 10)

    speed = 0.5
    turn = 1.0
    print(msg)
    print(vels(speed, turn))
    while True:
        with term.cbreak():
            key = term.inkey()  #taker input from terminal
        if key in moveBindings.keys():
            x = moveBindings[key][0]
            y = moveBindings[key][1]
            z = moveBindings[key][2]
            th = moveBindings[key][3]
        elif key in speedBindings.keys():
                speed = speed * speedBindings[key][0]
                turn = turn * speedBindings[key][1]
                print(vels(speed, turn))

        twist = geometry_msgs.msg.Twist()
        twist.linear.x = x * speed
        twist.linear.y = y * speed
        twist.linear.z = z * speed
        twist.angular.x = 0.0
        twist.angular.y = 0.0
        twist.angular.z = th * turn
```

```
        pub.publish(twist)

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

Do make the changes in setup.py and xml files of the package before colcon build.
We can also use curses library to take input, or stdin.read() (with import sys) function

---