The rclpy library will be imported for the methodology provided here for creating a custom ros2 teleop keyboard. Rclpy is this official ROS Client Library for Python.

Message type Twist will be imported from geometry_msgs.msg. Geometry_msgs provides messages for common geometric primitives such as points, vectors, and poses. These primitives are designed to provide a common data type and facilitate interoperability throughout the system. And within geometry_msgs, twist message type expresses velocity in free space broken into its linear and angular parts.

Message definations:

geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular

A variable named settings will be defined with termios.tcgetattr(fd), which will return a list containing the tty *attributes for file descriptor* fd. And tty is a terminal (it stands for teletype - the original terminals used a line printer for output and a keyboard for input!).

Python sys module stdin will be used by the interpreter for standard input.

This ros2 node will be used by the px4 companion computer to receive inputs in the form of keyboard keys, which will result in movement of the drone within the simulation environment.

The selected keys for movement is as follows:

```
Moving around:

  u   i   o

  j   k   l

  m   ,   .


For Holonomic mode (strafing), hold down the shift key:

---------------------------

  U   I   O

  J   K   L

  M   <   >


t : up (+z)

b : down (-z)

anything else : stop


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%

To assign the keys with its respective movement instructions, dictionaries will be created defining each individual options. The proposed solution is going to include two separate dictionaries for movement and speed bindings.

A custom function will be created called getKey(). The tty.setraw will be set to sys.stdin.fileno(). This will change the mode of the file descriptor fd to raw. A variable Key will be assigned to ‘stdin. readline()’. Stdin stands for standard input which is a stream from which the program reads its input data. So, this function will be responsible for taking the keyboard input for drone movement.

Another custom function will be created with the name vels() for returning the values for speed and turn in real time.

Next step would be calling of the function rclpy.init() for initializing the ros2 node. It will be named as 'teleop_twist_keyboard' through rclpy.create_node() function.

We will declare that our node is publishing to the topic 'cmd_vel' using the message type Twist, with the name ‘qos_profile_default’. Twist here is actually the class geometry_msgs.msg.Twist.

Variables will be declared for each coordinate values with an initial value. An if-else condition will be implemented to check whether the given input key belongs to SpeedBindings or MoveBindings dictionary. If keys are present in SpeedBindings, the key values will be assigned to x,y,z and th. If they are present in the MoveBindings, the values would be assigned to speed and turn.

Accordingly the drone will be instructed to move in the designated directions with the help of Twist.Angular and Twist.linear values. And at the end the Twist message will be published to px4 through the rtps_bridge with the help of px4_msgs and px4_ros_com packages.

To execute the node, we have to add the teleop_keyboard.py file in the px4_ros_com workspace under a package name.

Next step would be to source two bash files called clean_all.bash and build_ros2.bash under the scripts folder.

Then execute the following command in the terminal after launching px4 and micro rtps bridge:

```
Ros2 run teleop_keyboard ros2_teleop_keyboard.py
```