Nama : Cathleen Gracia
NRP : 5025231018

Laporan Penugasan 3

1. Langkah pertama dalam pengerjaan penugasan ini adalah menginstall turtlesim
    - sudo apt install ros-humble-turtlesim
2. membuat workspace dengan command mkdir -p (nama-workspace)/src
    - mkdir -p PenugasanBayucaraka/src
3. Setelah itu navigasikan ke workspace tersebut dengan command cd (nama-workspace)/src
    - cd PenugasanBayucaraka/src
4. Clone repository dengan command git clone (link)
   git clone https://github.com/cthleen/MagangBayu24-ROS2.git
5. Buat package dengan command ros2 pkg create --build-type ament_python {nama-package} untuk python
    - ros2 pkg create --build-type ament_python tugas2
6. Tambahkan publisher node pada direktori tugas2/tugas2
    - cd tugas2/tugas2
    - wget https://raw.githubusercontent.com/ros2/examples/humble/rclpy/topics/minimal_publisher/examples_rclpy_minimal_publisher/publisher_member_function.py
7. Tambahkan dependencies pada package.xml
    - code . (untuk membuka di vscode)
    - tambahkan ini di package.xml
      <exec_depend>rclpy</exec_depend>
      <exec_depend>std_msgs</exec_depend>
8. Tambahkan entry point pada setup.py
    - entry_points={
        'console_scripts': [
            'talker = tugas2.publisher_member_function:main',
        ],
      },
9. Buat codingan pada publisher_member_function.py
10. Lakukan command colcon build untuk build package tersebut
11. Lakukan command source install/setup.bash
12. Untuk menjalankan talker node menggunakan command ros2 run py_pubsub talker
13. Pada terminal lain, jalankan turtlesim dengan command ros2 run turtlesim turtlesim_node

Penjelasan code

```
def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher = self.create_publisher(Twist, '/turtle1/cmd_vel', 10)
        self.timer = self.create_timer(2.0, self.timer_callback)
        self.count = 0
```

Pada bagian *self.publisher = self.create_publisher(Twist, '/turtle1/cmd_vel', 10)*, dapat terlihat saya membuat publisher dengan tipe Twist dan topic ‘/turtle/cmd_vel’.

Selain itu, saya juga melakukan inisialisasi untuk variabel self.count.

```
def teleport_absolute(self, x, y, theta):
        teleport_service = self.create_client(TeleportAbsolute, '/turtle1/teleport_absolute')
        while not teleport_service.wait_for_service(timeout_sec=1.0):
        self.get_logger().info('Waiting for the teleport_absolute service...')

        request = TeleportAbsolute.Request()
        request.x = x
        request.y = y
        request.theta = theta

        future = teleport_service.call_async(request)
        rclpy.spin_until_future_complete(self, future)
        self.get_logger().info('Teleportation complete.')
```

Bagian ini digunakan untuk teleport turtle ke titik (7.5, 5.0, 0.0),

```
def clear_turtlesim(self):
        clear_service = self.create_client(Empty, '/clear')
        while not clear_service.wait_for_service(timeout_sec=1.0):
        self.get_logger().info('Waiting for the clear service...')

        empty_request = Empty.Request()
        clear_future = clear_service.call_async(empty_request)
        rclpy.spin_until_future_complete(self, clear_future)
        self.get_logger().info('Clearing complete.')
```

Bagian ini untuk clearing turtlesim.

```
def timer_callback(self):
        if self.count < 6:
                if self.count % 2 ==0:
                        msg = Twist()
                        msg.angular.z = 2.07774
                        self.get_logger().info('Rotating...')
                else:
                        msg = Twist()
                        msg.linear.x = 4.0
                        self.get_logger().info('Moving Forward...')
        elif self.count >= 6 and self.count < 12:
                if self.count % 2 == 1:
                        msg = Twist()
                        msg.angular.z = math.radians(177.5)
                        msg.linear.x = 2 * math.pi - 0.1
                        self.get_logger().info('Moving...')
                elif self.count == 6:
                        msg=Twist()
                        msg.angular.z = math.radians(30)
                        self.get_logger().info('Rotating...')
```

```
            else:
                msg=Twist()
                msg.angular.z = math.radians(-60)
                self.get_logger().info('Rotating...')
        elif self.count == 12:
        return

        self.count += 1
        self.publisher.publish(msg)
```

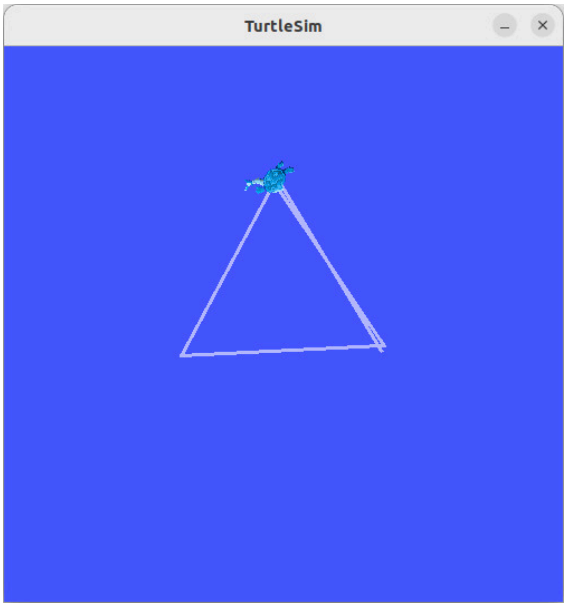
Bagian ini untuk membuat pola.
Saya membuat program tersebut agar saat self.count bernilai genap turtle berotasi dan saat bernilai ganjil turtle bergerak maju atau membentuk setengah lingkaran.

Sintaks dibawah ini berguna untuk membuat bagian segitiganya.

```
if self.count < 6:
        if self.count % 2 ==0:
            msg = Twist()
            msg.angular.z = 2.07774
            self.get_logger().info('Rotating...')
        else:
            msg = Twist()
            msg.linear.x = 4.0
            self.get_logger().info('Moving Forward...')
```

Sintaks *msg.angular.z* menyatakan kecepatan sudut dan *msg.linear.x* menyatakan kecepatan linear. Jadi, *msg.angular.z* berguna untuk mengendalikan gerakan rotasi turtle dan *msg.linear.x* mengendalikan gerakan linear maju atau mundurnya robot.
Variabel self.count digunakan untuk menghitung berapa gerakan yang telah dilakukan robot dan menentukan pada saat urutan tersebut gerakan apa yang harus dilakukan.

Seharusnya segitiga sama sisi memiliki sudut masing-masing 60 derajat, jadi gerak rotasi sudut yang diperlukan untuk membentuk sisi segitiga adalah 120 derajat atau 2.0944. Namun, ketika mencoba dengan sudut tersebut, segitiga yang saya buat malah menjadi tidak pas seperti gambar berikut.



Jadi, saya mencoba mengubah kecepatan sudutnya agar bisa pas dan akhirnya menemukan yang paling pas ketika kecepatan sudutnya 2.07774.

Sintaks dibawah ini berguna untuk membuat setengah lingkaran.

```
elif self.count >= 6 and self.count < 12:
        if self.count % 2 == 1:
                msg = Twist()
                msg.angular.z = math.radians(177.5)
                msg.linear.x = 2 * math.pi - 0.1
                self.get_logger().info('Moving...')
        elif self.count == 6:
                msg=Twist()
                msg.angular.z = math.radians(30)
                self.get_logger().info('Rotating...')
        else:
                msg=Twist()
                msg.angular.z = math.radians(-60)
                self.get_logger().info('Rotating...')
```

Karena saya ingin membuat setengah lingkaran dengan radius 2, jadi saya membuat kecepatan sudutnya $\frac{\pi}{2}$. Dari sana, saya menggunakan rumus $v = \omega \times r$ sehingga mendapatkan nilai kecepatan linear π.

$v = \omega \times r$

$v = \frac{\pi}{2} \times 2$

$v = \pi$

Namun, saya juga mengalami kendala dimana gerak dalam membentuk lingkarannya juga tidak pas, jadi saya mencoba dan memodifikasi angkanya agar bisa lebih pas.

Saya melakukan rotasi dengan mengatur kecepatan sudut. Untuk yang pertama, saya merotasikan 30 derajat searah jarum jam dari titik awal. Untuk yang kedua dan ketiga, saya merotasikan 60 derajat melawan arah jarum jam dari titik sebelumnya.

```
def main(args=None):
    rclpy.init(args=args)
    minimal_publisher = MinimalPublisher()
    try:
        minimal_publisher.teleport_absolute(7.5, 5.0, 0.0)
        minimal_publisher.clear_turtlesim()
        rclpy.spin(minimal_publisher)

    finally:
        minimal_publisher.destroy_node()
        rclpy.shutdown()
```

Sintaks *minimal_publisher.teleport_absolute(7.5, 5.0, 0.0)* berguna untuk menjalankan fungsi teleport_absolute dengan kondisi x = 7.5, y = 5.0, dan θ = 0.0.
Sintaks *minimal_publisher.clear_turtlesim()* berguna untuk menjalankan fungsi clear_turtlesim. Fungsi ini berguna untuk membersihkan jalan atau path yang sudah dilalui turtle.
Sintaks *rclpy.spin(minimal_publisher)* berguna untuk menjalankan node.