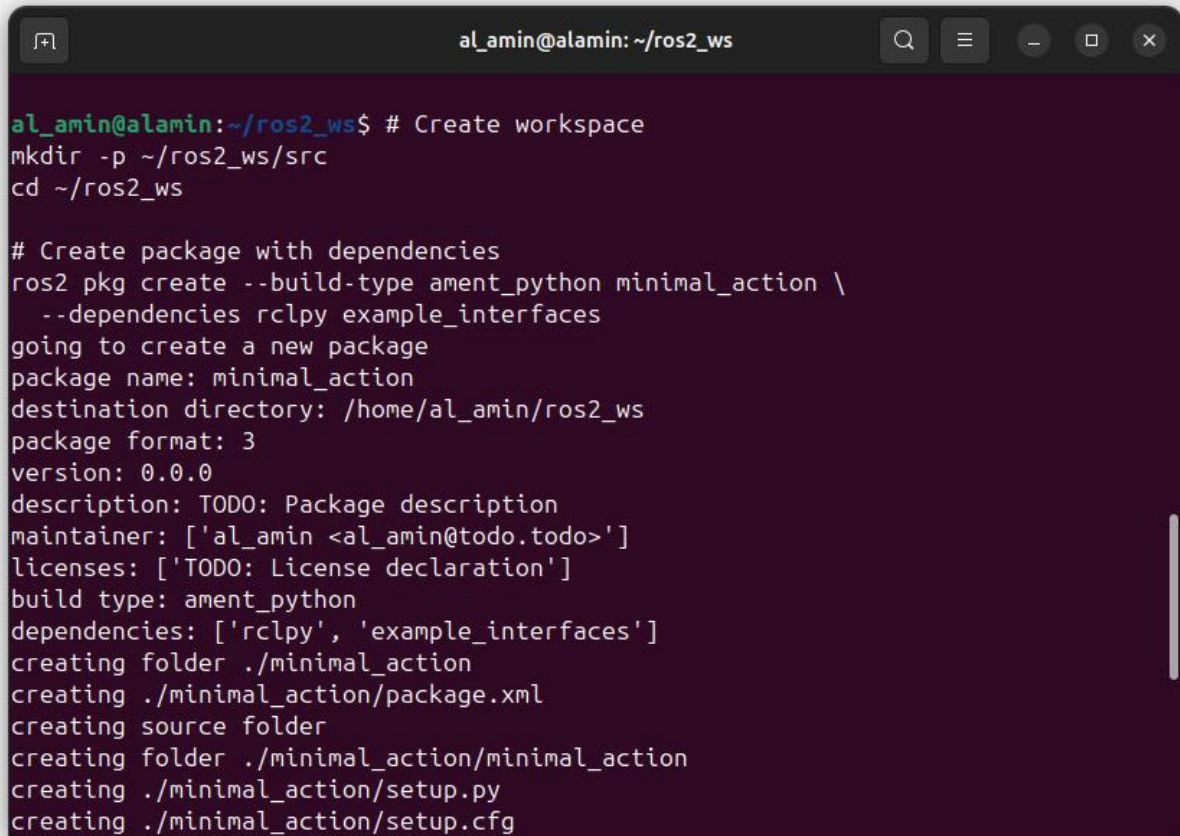


Name : Al Amin Hossain Nayem
Labwork 3

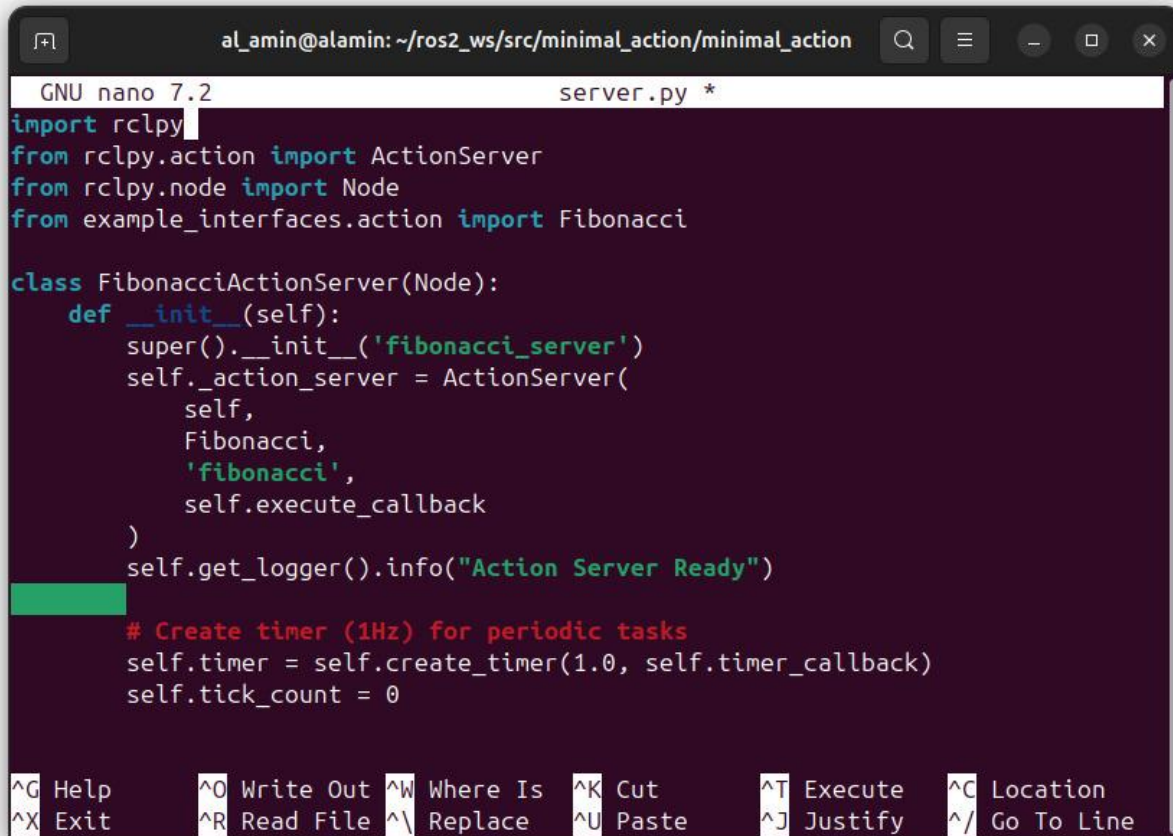
1)Using existing action

A terminal window with a dark purple background and white text. The window title is 'al_amin@alamin: ~/ros2_ws'. The terminal shows the following commands and output:

```
al_amin@alamin:~/ros2_ws$ # Create workspace
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws

# Create package with dependencies
ros2 pkg create --build-type ament_python minimal_action \
  --dependencies rclpy example_interfaces
going to create a new package
package name: minimal_action
destination directory: /home/al_amin/ros2_ws
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['al_amin <al_amin@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_python
dependencies: ['rclpy', 'example_interfaces']
creating folder ./minimal_action
creating ./minimal_action/package.xml
creating source folder
creating folder ./minimal_action/minimal_action
creating ./minimal_action/setup.py
creating ./minimal_action/setup.cfg
```

2)Implementing custom action



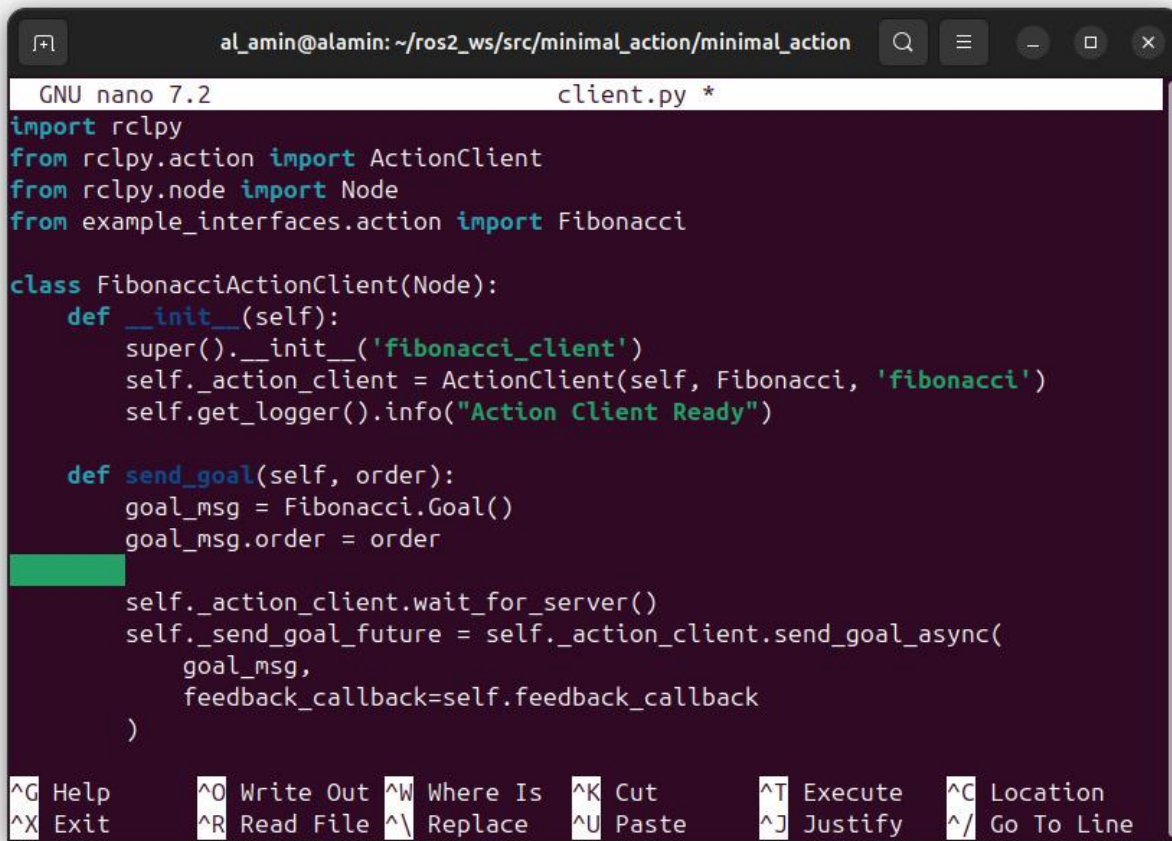
```
al_amin@alamin: ~/ros2_ws/src/minimal_action/minimal_action
GNU nano 7.2 server.py *
import rclpy
from rclpy.action import ActionServer
from rclpy.node import Node
from example_interfaces.action import Fibonacci

class FibonacciActionServer(Node):
    def __init__(self):
        super().__init__('fibonacci_server')
        self._action_server = ActionServer(
            self,
            Fibonacci,
            'fibonacci',
            self.execute_callback
        )
        self.get_logger().info("Action Server Ready")

        # Create timer (1Hz) for periodic tasks
        self.timer = self.create_timer(1.0, self.timer_callback)
        self.tick_count = 0

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

Name : Al Amin Hossain Nayem
Labwork 3



```
al_amin@alamin: ~/ros2_ws/src/minimal_action/minimal_action
GNU nano 7.2 client.py *
import rclpy
from rclpy.action import ActionClient
from rclpy.node import Node
from example_interfaces.action import Fibonacci

class FibonacciActionClient(Node):
    def __init__(self):
        super().__init__('fibonacci_client')
        self._action_client = ActionClient(self, Fibonacci, 'fibonacci')
        self.get_logger().info("Action Client Ready")

    def send_goal(self, order):
        goal_msg = Fibonacci.Goal()
        goal_msg.order = order

        self._action_client.wait_for_server()
        self._send_goal_future = self._action_client.send_goal_async(
            goal_msg,
            feedback_callback=self.feedback_callback
        )

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

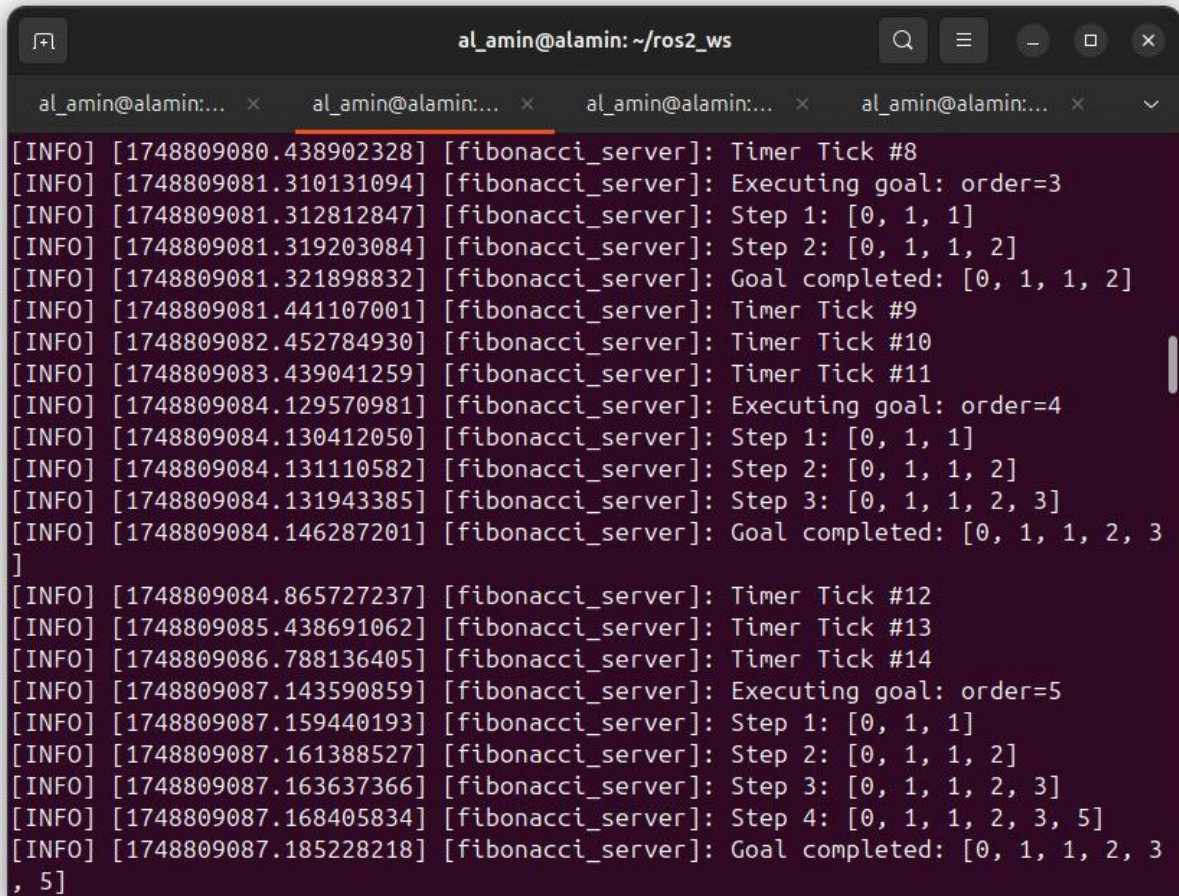
Name : Al Amin Hossain Nayem
Labwork 3

```
al_amin@alamin: ~/ros2_ws
al_amin@alamin:~/ros2_ws/src/minimal_action$ ls
minimal_action package.xml resource setup.cfg setup.py test
al_amin@alamin:~/ros2_ws/src/minimal_action$ cd minimal_action/
al_amin@alamin:~/ros2_ws/src/minimal_action/minimal_action$ ls
client.py __init__.py
al_amin@alamin:~/ros2_ws/src/minimal_action/minimal_action$ nano server.py
al_amin@alamin:~/ros2_ws/src/minimal_action/minimal_action$ nano client.py
al_amin@alamin:~/ros2_ws/src/minimal_action/minimal_action$ cd ~/ros2_ws
colcon build --packages-select minimal_action
source install/setup.bash
[0.243s] WARNING:colcon.colcon_ros.prefix_path.ament:The path '/home/al_amin/ros2_ws/install/custom_action_interfaces' in the environment variable AMENT_PREFIX_PATH doesn't exist
[0.243s] WARNING:colcon.colcon_ros.prefix_path.ament:The path '/home/al_amin/ros2_ws/install/temp_converter' in the environment variable AMENT_PREFIX_PATH doesn't exist
[0.243s] WARNING:colcon.colcon_ros.prefix_path.catkin:The path '/home/al_amin/ros2_ws/install/custom_action_interfaces' in the environment variable CMAKE_PREFIX_PATH doesn't exist
Starting >>> minimal_action
Finished <<< minimal_action [1.62s]

Summary: 1 package finished [1.82s]
al_amin@alamin:~/ros2_ws$
```


Name : Al Amin Hossain Nayem
Labwork 3

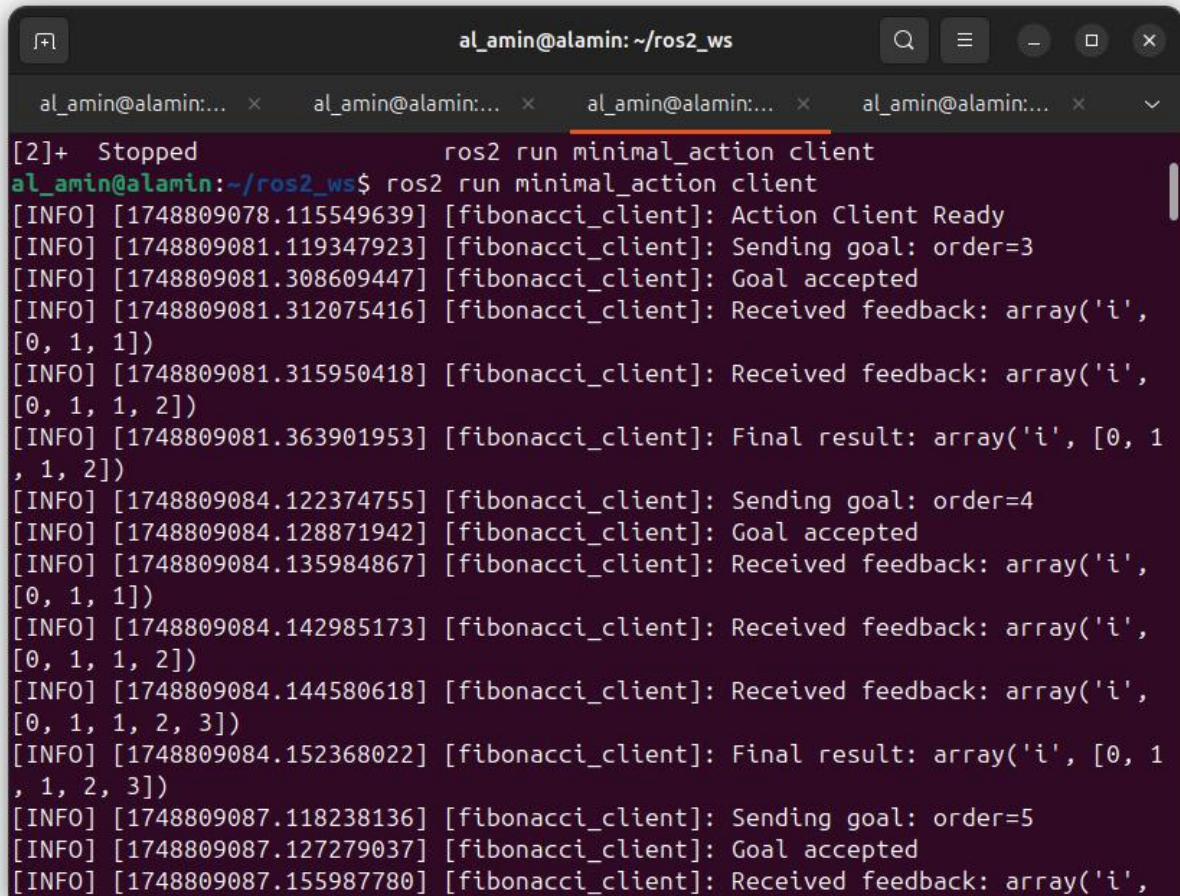
3)Using timer in service node



```
al_amin@alamin: ~/ros2_ws
[INFO] [1748809080.438902328] [fibonacci_server]: Timer Tick #8
[INFO] [1748809081.310131094] [fibonacci_server]: Executing goal: order=3
[INFO] [1748809081.312812847] [fibonacci_server]: Step 1: [0, 1, 1]
[INFO] [1748809081.319203084] [fibonacci_server]: Step 2: [0, 1, 1, 2]
[INFO] [1748809081.321898832] [fibonacci_server]: Goal completed: [0, 1, 1, 2]
[INFO] [1748809081.441107001] [fibonacci_server]: Timer Tick #9
[INFO] [1748809082.452784930] [fibonacci_server]: Timer Tick #10
[INFO] [1748809083.439041259] [fibonacci_server]: Timer Tick #11
[INFO] [1748809084.129570981] [fibonacci_server]: Executing goal: order=4
[INFO] [1748809084.130412050] [fibonacci_server]: Step 1: [0, 1, 1]
[INFO] [1748809084.131110582] [fibonacci_server]: Step 2: [0, 1, 1, 2]
[INFO] [1748809084.131943385] [fibonacci_server]: Step 3: [0, 1, 1, 2, 3]
[INFO] [1748809084.146287201] [fibonacci_server]: Goal completed: [0, 1, 1, 2, 3]
]
[INFO] [1748809084.865727237] [fibonacci_server]: Timer Tick #12
[INFO] [1748809085.438691062] [fibonacci_server]: Timer Tick #13
[INFO] [1748809086.788136405] [fibonacci_server]: Timer Tick #14
[INFO] [1748809087.143590859] [fibonacci_server]: Executing goal: order=5
[INFO] [1748809087.159440193] [fibonacci_server]: Step 1: [0, 1, 1]
[INFO] [1748809087.161388527] [fibonacci_server]: Step 2: [0, 1, 1, 2]
[INFO] [1748809087.163637366] [fibonacci_server]: Step 3: [0, 1, 1, 2, 3]
[INFO] [1748809087.168405834] [fibonacci_server]: Step 4: [0, 1, 1, 2, 3, 5]
[INFO] [1748809087.185228218] [fibonacci_server]: Goal completed: [0, 1, 1, 2, 3, 5]
```

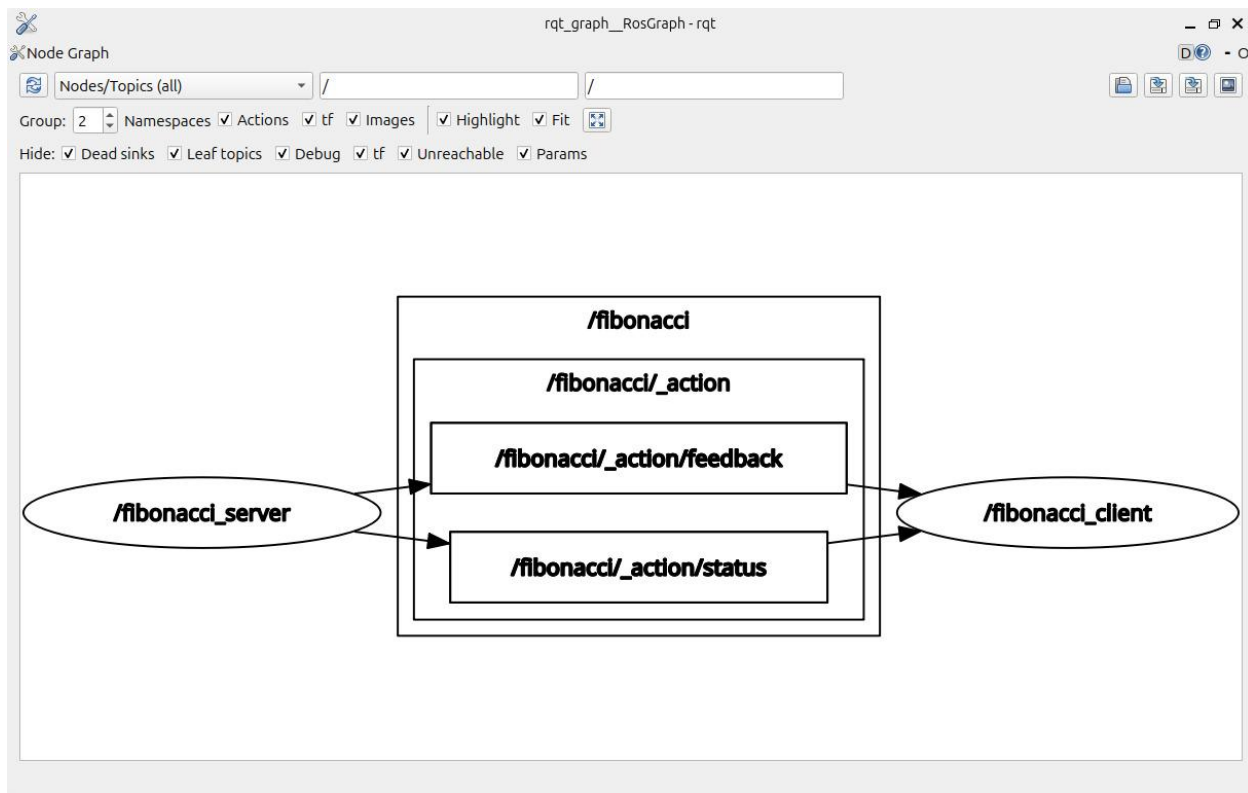
Name : Al Amin Hossain Nayem
Labwork 3

Results.



```
al_amin@alamin: ~/ros2_ws
[2]+  Stopped                  ros2 run minimal_action client
al_amin@alamin:~/ros2_ws$ ros2 run minimal_action client
[INFO] [1748809078.115549639] [fibonacci_client]: Action Client Ready
[INFO] [1748809081.119347923] [fibonacci_client]: Sending goal: order=3
[INFO] [1748809081.308609447] [fibonacci_client]: Goal accepted
[INFO] [1748809081.312075416] [fibonacci_client]: Received feedback: array('i',
[0, 1, 1])
[INFO] [1748809081.315950418] [fibonacci_client]: Received feedback: array('i',
[0, 1, 1, 2])
[INFO] [1748809081.363901953] [fibonacci_client]: Final result: array('i', [0, 1
, 1, 2])
[INFO] [1748809084.122374755] [fibonacci_client]: Sending goal: order=4
[INFO] [1748809084.128871942] [fibonacci_client]: Goal accepted
[INFO] [1748809084.135984867] [fibonacci_client]: Received feedback: array('i',
[0, 1, 1])
[INFO] [1748809084.142985173] [fibonacci_client]: Received feedback: array('i',
[0, 1, 1, 2])
[INFO] [1748809084.144580618] [fibonacci_client]: Received feedback: array('i',
[0, 1, 1, 2, 3])
[INFO] [1748809084.152368022] [fibonacci_client]: Final result: array('i', [0, 1
, 1, 2, 3])
[INFO] [1748809087.118238136] [fibonacci_client]: Sending goal: order=5
[INFO] [1748809087.127279037] [fibonacci_client]: Goal accepted
[INFO] [1748809087.155987780] [fibonacci_client]: Received feedback: array('i',
```

4) graph.



My Understanding of ROS2 Actions

ROS2 Actions are used for long tasks where feedback and result are both needed. Unlike simple services, actions let me send a goal (like generating a Fibonacci sequence) and keep getting updates while the task is running.

- 1.The server sends progress using feedback.
- 2.At the end, it returns the full result.
- 3.If needed, I can also cancel the task.

Internally, actions use services to send/receive goals and results, and topics to send feedback. This makes them useful for things like robot movement or other tasks that take time and need updates.

My System Structure & Purpose

I made a simple ROS2 system with 2 nodes — one client and one server — using actions and a timer.

Structure Overview

- 1.Client Node sends a goal (e.g. "Give me Fibonacci of 10")
- 2.Server Node calculates it and keeps sending feedback
- 3.A 1Hz timer is used to simulate periodic tasks

Node Details

1.fibonacci_server.py (Server)

- Accepts a goal with an order number
- Uses a timer to wait 1 second before sending each feedback
- Sends final sequence as result

2.fibonacci_client.py (Client)

- Sends goal
- Gets live feedback (like progress)
- Shows result when done

3.Visualization

- I used rqt_graph to see how nodes and topics connect.
- Timer is inside the server and doesn't show up directly.

4. Key Steps

- Defined custom action: Fibonacci.action
- Wrote server with execute_callback and timer
- Made client that sends goal and handles feedback
- Built and ran it using ROS 2 commands

Conclusion

- I learned how ROS2 Actions help with tasks that need progress updates
- The timer shows how background tasks can run at a fixed rate
- This system helped me understand how ROS2 nodes work together for async tasks
- It's a basic example, but the same logic applies in bigger robotics apps