**Test Plan: Subsystem 2 – TurtleBot Movement Logic**

**PASS: Basic Navigation**

Validate that the TurtleBot moves to a hardcoded table destination and returns to a hardcoded drinks location.

Steps:

1. Deploy the TurtleBot in a gazebo environment
2. Execute the movement logic to send the TurtleBot to the predefined goal
3. Confirm that the TurtleBot reaches the goal
4. Execute the movement logic to send the TurtleBot back to the drinks stand / home
5. Confirm that the TurtleBot successfully returns

**Expected Result:** TurtleBot reaches the published goal and returns to the drinks location without deviation

* Initially executed using existing Nav2Goal topic, no written logic
* Then designed written code to publish hardcoded one goal, TurtleBot travelled to goal, and after goal tolerance bool was true TurtleBot returned to drinks table
* - Provides status updates for locations/tasks completed

**CREDIT: Dynamic Goal Updates**

Validate that the TurtleBot can receive goal data dynamically from the host. Verify whether the goal updates overwrite previous ones and whether new goals are ignored during movement.

Steps:

1. Deploy the TurtleBot and provide an initial goal from the host – via terminal input
2. During execution, send a new goal update
3. Observe whether the new goal replaces the old one or is ignored
4. Check if the TurtleBot listens for new goals after completing the previous action.

**Expected Result:** The TurtleBot updates its goal if a new one is provided before reaching the previous goal and listens for new goals after task completion.

* First code attempt: Turtlebot moved to goals published in the terminal, and waited for a second terminal prompt to confirm it can begin movement to goals. However, goal publishing was interrupting, it did not add to a queue
* Second code attempt: Turtlebot did save goals in queue, however, it lost the functionality to return to the drinks table and wait for the “go” command and travelled directly to each goal in the saved queue.

**DISTINCTION: Continuous Goal Execution**

Ensure the TurtleBot continuously listens for new goals, the goals are not interrupting, it executes them in order, and returns to the drinks stand between each goal.

Steps:

1. Deploy the TurtleBot in gazebo
2. Provide multiple sequential goals from the host during any point in TurtleBot movement
3. Observe if the TurtleBot continuous to navigate to each goal and returns to the drinks stand after each time
4. Verify that the process repeats indefinitely for infinite goals

**Expected Result:** The TurtleBot listens and executes new goals while always returning to the drinks stand.

* Combined functionality of the two codes from “credit”, so TurtleBot travels to the drinks stand first, waits for terminal data (aka host data), and goes to goal after “go” command aka UR3 bartender data), delivers drink, waits an appropriate time, and always returns.
* Services set up for goal trigger and go trigger, not terminal commands

**HIGH DISTINCTION: Multi-TurtleBot Scalability**

Ensure multiple TurtleBot can navigate to different goals and return without conflict

Steps:

1. Deploy multiple TurtleBots in the environment
2. Assign separate goals to each TurtleBot from the host
3. Observe if all TurtleBots reach their respective goals and return to the drinks stand
4. Check for potential navigation conflicts, goal publishing blocking, or coinciding goals

**Expected Result:** Multiple TurtleBot’s can travel to their goals without interfering with each other.

To Be Done:

* Change code so node listens for services instead of receiving info from directly publishing to terminal for goals and when to go info
* Scalable logic for multiple TurtleBot’s, such as ensuring when to stop/start/go triggers/wait times are assigned to the correct TurtleBot
* Status updates come from both TurtleBot’s and is distinguishable
* Set up node/function that can be called for each TurtleBot, integrate with goal management and assignment logic

**EXTENSION: Dynamic Environment Handling**

Validate that the TurtleBot can handle dynamic obstacles while still reaching its goals.

1. Deploy the TurtleBot in a gazebo environment
2. Provide a goal location and start movement
3. Introduce dynamic obstacles (e.g., moving objects, people walking across the path)
4. Observe if the TurtleBot adjusts its path while still targeting the goal.
5. Confirm that the TurtleBot does not lose its ability to navigate, moving obstacle is not interrupting to travel logic, and TurtleBot appropriately waits for a clear path

**Expected Result:** Multiple TurtleBot successfully avoid obstacles and reaches its assigned goals without failure, and returns to drinks stand

To be Done

* Scale code and add additional logic for dynamic environments