

ROS AUTOMATION TESTING FOR WHEELED AND LEGGED MOBILE ROBOTS

**Lyubomyr Demkiv,
Yuriy Fedyuk, Taras Borovets**

softserve

The Case for Robot Software Automation Testing

Building and testing applications for robots has historically been a complicated and time-consuming process. With AWS RoboMaker's cutting-edge automated testing for virtual environments, integrating new features to your robots and confirming that they work is easier than ever.

Testing in robotics is as important as in other software development domains. The sooner you find an issue, the easier it can be fixed. We can divide software testing into two types: manual and automated tests.



In manual robotics testing, challenges include:

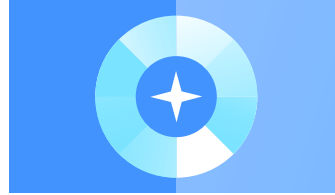


Test execution takes a lot of time

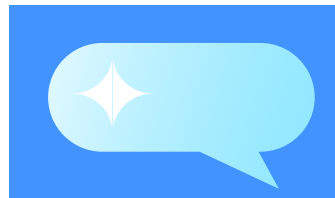


Manual tests cannot be easily scaled to different environments and require expensive hardware

Automated testing in a simulated virtual environment allows you to:



Check robot behavior in a large number of environments with different scenarios, so one can find more bugs compare to a manual test



Provide software developers with more immediate feedback about their applications and if they match with correct robot behavior

There are multiple ways to classify the automation tests. In general, the tests can be split into two groups:

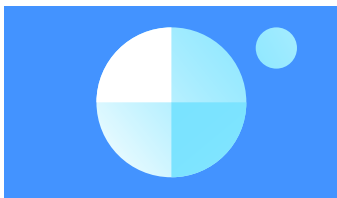


Functional tests. Refer to the ability of the robot to accomplish the task according to the business logic that has been implemented in the solution

Automated tests may be split into the following groups:



Smoke tests that cover the most crucial robot functionality that can be promptly tested to ensure the robot's stable behavior during further tests



Non-functional tests. Validate the robot's resource consumption, data storage, data streaming capabilities, and other



Integration tests that validate the functionality of the solution when all distributed robot components are executed together



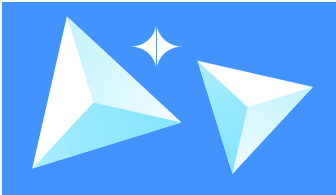
Regression tests that ensure the robot's both functional and non-functional characteristics are not negatively affected by the most recent changes



Security tests aim to reveal the vulnerability of the software during robot operations under various conditions



Performance tests are usually devoted to non-functional robot performance ensuring stable, and responsive behavior of the robot



Acceptance tests are final tests of robot's functional behavior prior to final approval and release

Automated tests may be split into the following groups:

The test phases are generally split into three stages:

1. Unit tests when each individual component is tested
2. Hardware integration tests
3. Business logic conformity tests

Using automated tests in simulated virtual environments allows you to decrease development and maintenance costs as well as save costs during hardware tests.

Benefits of AWS RoboMaker

With multiple ways of orchestrating automated tests, but AWS RoboMaker is one of the most convenient for implementation. It is a fully managed simulation service for running robot applications in a simulated virtual environment. You can seamlessly integrate your robot automation tests with AWS CodePipeline (or to other AWS services) and run them during the CI/CD process.

Types of Tests

AWS RoboMaker supports large-scale and parallel simulations, so you can easily run a wide variety of simulations and receive results much faster than using own hardware for that.

And with its pay-as-you-go pricing model, you only pay for resources used, without the need to think about hardware and how to scale it. AWS RoboMaker provides all the resources needed for a single simulation or for dozens of simulations in the same way, using a single API call.

Depending on your robot type, it could require a unique set of tests. For example, tests for drones are not as efficient when applied to ground mobile platforms, and vice versa. Best practice is to divide all tests into several parts, with each part responsible for testing specific robot components such as its planner, localization, and others. Let's take a closer look at each one of these tests.

SLIPPAGE AND MOTOR SATURATION TESTS	PLANNING TESTS	BEHAVIOR TESTS	LOCALIZATION TESTS
Such tests are conducted to check the work of motor drivers and algorithms. These tests could check if the robot does not move without commands or whether the robot's movements are precise enough.	With planning tests, we can check the work of the robot navigation planner and easily fix a bug when the planner creates a new route when we ask it to go to a certain point where we are located.	With behavior tests, we check the robot's behavior under different circumstances. Such tests include, but not limited to, obstacle avoidance testing, coverage testing, and moving from point A to point B.	Localization tests help us check whether the robot can successfully localize itself on a known map, allowing us to control the robot localization stack. We will show what tests are useful for ground mobile robots. In our examples we use scenario-based testing , as it allows us to easily parametrize each test and integrate robot simulation tests into CI/CD process.

How to Run the Tests for Boston Dynamics' Spot

SoftServe's robotics group developed [a list of ROS1 tests](#) suitable for any mobile robot running in 2D. Here's [how to run the tests for the quadruped robot Spot](#) from [Boston Dynamics](#). The control of multi-legged robots is significantly different from the control of robots with wheels or other types of locomotion. However, the above-mentioned types of tests are quite generic and relate to high-level control testing or robot-environment testing. Therefore, the

tests can be easily adapted to the robot regardless of the type of its locomotion.

Boston Dynamics' Spot is one of the most advanced legged robots available on the market. It can perform navigation missions, move on difficult terrains, climb stairs, carry payloads, and other. You can find out more about the full functionality of Boston Dynamics' Spot on their [official website](#) or read our [blog](#).

Today, the robot is used in such areas as pipeline inspection, construction inspection, healthcare, and others. To ensure Spot's reliability for your specific mission, it must be tested beforehand. For that, AWS RoboMaker can be used.

Parts of the Code

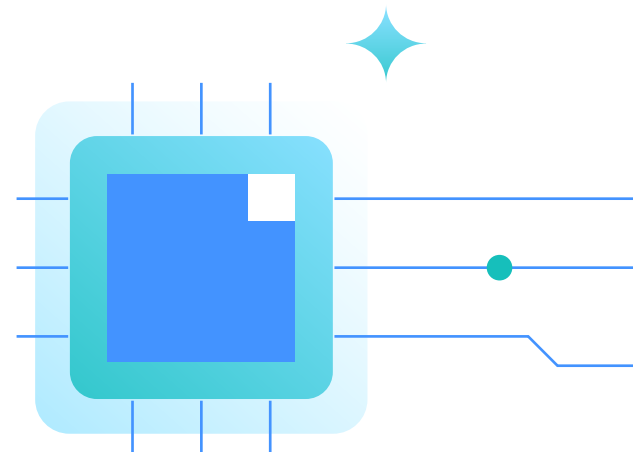
To test the robot on the AWS RoboMaker, you need to have:

- Robot simulation package with a control system and navigation stack.
- Test package
- A package that contains scripts that create the cloud formation
- Test launcher that runs test scripts with the required launch files

SoftServe's robotics group repository [Scenario-based Tests with AWS RoboMaker for Boston Dynamics' Spot](#) allows you to run tests with just one click. The repository consists of the simulation tool, package with tests, cloud formation setup scripts, and test launch scripts. Based on this example, you can easily adopt scripts for your application because of the repo's modular structure.

The simulation for Spot is implemented on the basis of the open-source package [champ](#), which is designed to develop the control of the legged robots and contains a preconfigured package for the Spot simulation. The repository includes navigation and localization stacks, allows you to control the walk of the robot and the positioning of its body relative to the limbs. We have made the necessary improvements to be able to run the champ on AWS RoboMaker.

The cloud formation scripts set up all the AWS resources needed for automated testing on your behalf.



The test launcher aims to launch all needed nodes for test running and launching the test as well. The simulation tool is split into the simulation environment `simulation_ws` and robot environment `robot_ws`. In order to launch a test, you should launch the required package for testing from both environments. As such, we created directories which include launch files to do that. For example, in order to run an obstacle avoidance test, you need to run Gazebo with a robot model and test script in a simulation environment, while in a robot environment, you should run navigation and localization stacks. Examples of such launch files are given below.

```
<launch>
```

```
  <arg name="gui" default="false"/>
```

```
  <arg name="path_topic" default="/  
move_base/GlobalPlanner/plan"/>
```

```
  <param name="use_sim_  
time" value="true" />
```

```
  <include file="$(find rs_config)/  
launch/gazebo.launch" >
```

```
    <arg name="gui" default="$(arg gui)" />
```

```
  </include>
```

```
  <arg name="mode" default="dynamic"/>
```

```
  <node pkg="softserve_simulation_  
common" type="move_base_route_  
manager.py" name="move_base_  
route_manager" output="screen">
```

```
  </node>
```

```
  <include file="$(find mp_  
behaviour_tests)/launch/obstacle_  
avoidance_test.launch" >
```

```
    <arg name="path_topic"  
value="$(arg path_topic)"/>
```

```
  </include>
```

```
</launch>
```

Launch File in the Simulation Environment

```
<launch>

  <arg name="gui" default="false"/>

  <param name="use_sim_
time" value="true" />

  <include file="$(find rs_navigation)/
launch/navigate.launch" >

    <arg name="map_file" value="$(find
aws_robomaker_small_house_world)/
maps/turtlebot3_waffle_pi/map.yaml"/>

    <arg name="rviz" value="$(arg gui)" />

  </include>

</launch>
```

Launch File in the Robot Environment

When the launch files are ready, you can run your test on the AWS RoboMaker platform. How can you run many different tests in different environment conditions at scale? For that, you should write test scenarios.

Test Scenarios

The scenario consists of a set of parameters that define environment conditions, robot behaviors, and expected outcomes. AWS RoboMaker allows you to run hundreds of simulation tests with different scenarios at scale. Take a look at the JSON file to configure the coverage test given below.

The coverage test shows the efficiency of the robot's ability to cover the floor of the environment. Here, the test-related parameter is `ROBOT_COVERAGE_TEST_COVERAGE_GOAL` which determines the coverage goal. You can set different goal values for different scenarios.

```

{
  "scenarios":{
    "Scenario1":{
      "simEnvironmentVariables":{
        "MODEL_NAME":"/",
        "START_X":"0",
        "START_Y":"0",
        "START_Z":"0.0",
        "START_YAW":"0",
        "ROBOT_COVERAGE_TEST_COVERAGE_GOAL":"80"
      },
      "robotEnvironmentVariables":{
        "MODEL_NAME":"/",
        "START_X":"0",
        "START_Y":"0",
        "START_Z":"0.0",
        "START_YAW":"0"
      }
    },
    "Scenario2":{
      "simEnvironmentVariables":{
        "MODEL_NAME":"/",
        "START_X":"0",
        "START_Y":"0",
        "START_Z":"0.0",
        "START_YAW":"0",
        "ROBOT_COVERAGE_TEST_COVERAGE_GOAL":"10"
      },
      "robotEnvironmentVariables":{
        "MODEL_NAME":"/",
        "START_X":"0",
        "START_Y":"0",
        "START_Z":"0.0",
        "START_YAW":"0"
      }
    }
  },
  "simulations":[
    {
      "scenarios":[
        "Scenario1",
        "Scenario2"
      ],
      "params":{
        "failureBehavior": "Fail",
        "maxJobDurationInSeconds": 600,
        "simulationApplications":[
          {
            "applicationVersion": "$LATEST",
            "launchConfig":{
              "launchFile":"coverage_test.launch",
              "packageName":"rs_tests"
            }
          }
        ],
        "robotApplications":[
          {
            "applicationVersion": "$LATEST",
            "launchConfig":{
              "launchFile":"coverage_test.launch",
              "packageName":"rs_robot_tests"
            }
          }
        ],
        "vpcConfig": {
          "assignPublicIp": true
        }
      }
    }
  ]
}

```

JSON file to configure the coverage test

Run Test

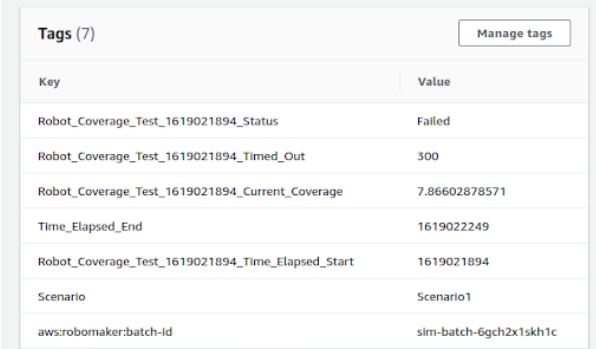
In order to run a test, you should execute the following command in an IDE terminal:

```
cd ~/environment/aws_ros_tests./run.sh test_launch_json/<json_file_name>.json
```

For example, run the coverage test:

```
cd ~/environment/aws_ros_tests./run.sh test_launch_json/coverage_test.json
```

This command launches two Simulation Jobs picking parameters determined in `coverage_test.json` file for two scenarios. When the jobs are complete, you will see the test results tagged to the simulation jobs. For example, the coverage test is tagged with the reasons for the end of the test, test results, current coverage value, start and end simulation time and to which test scenario it corresponds.



Key	Value
Robot_Coverage_Test_1619021894_Status	Failed
Robot_Coverage_Test_1619021894_Timed_Out	300
Robot_Coverage_Test_1619021894_Current_Coverage	7.86602878571
Time_Elapsed_End	1619022249
Robot_Coverage_Test_1619021894_Time_Elapsed_Start	1619021894
Scenario	Scenario1
aws:robomaker:batch-id	sim-batch-6gch2x1skh1c

Tags can carry additional information, which will allow you to quickly find the cause of bugs. Tag assignments are performed by a simple command:

```
self.utils.set_tag(name=self.test_name + "_Status", value="Failed")
```

Examples of Other Automation Tests' Execution

Let us consider several tests from the available list in our [test repository](#).

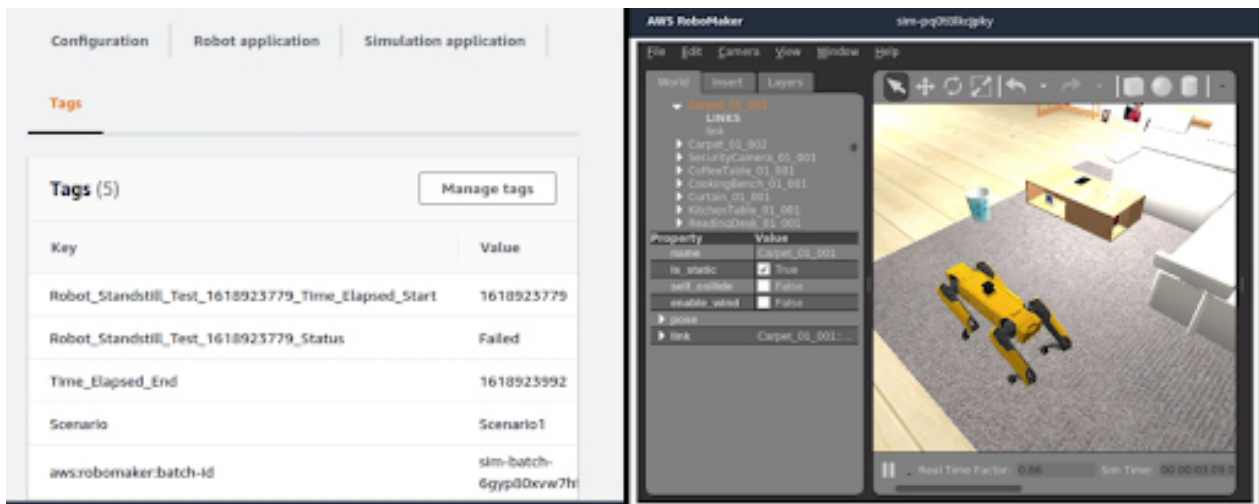
Slippage and Motor Saturation Tests: Standstill Test

Video 1 depicts the execution of a standstill test that has to detect whether the slippage is set to a realistic value. When standing on the flat surface, the robot's 6DOF position should not change as time passes by.

The test is marked as Failed, if a robot's 6DOF position changes during a predefined period of time. The test

is marked as Passed, if a robot's 6DOF position is unchanged within time, less than the predefined period of time.

Video 1 also demonstrates that when the robot's position was changed, the test was tagged as failed. You can run different scenarios in which the robot is spawned on different floor materials (wood, betony etc.), and also redefine its position and orientation tolerance.



Video 1. Slippage and motor saturation tests: Standstill test.

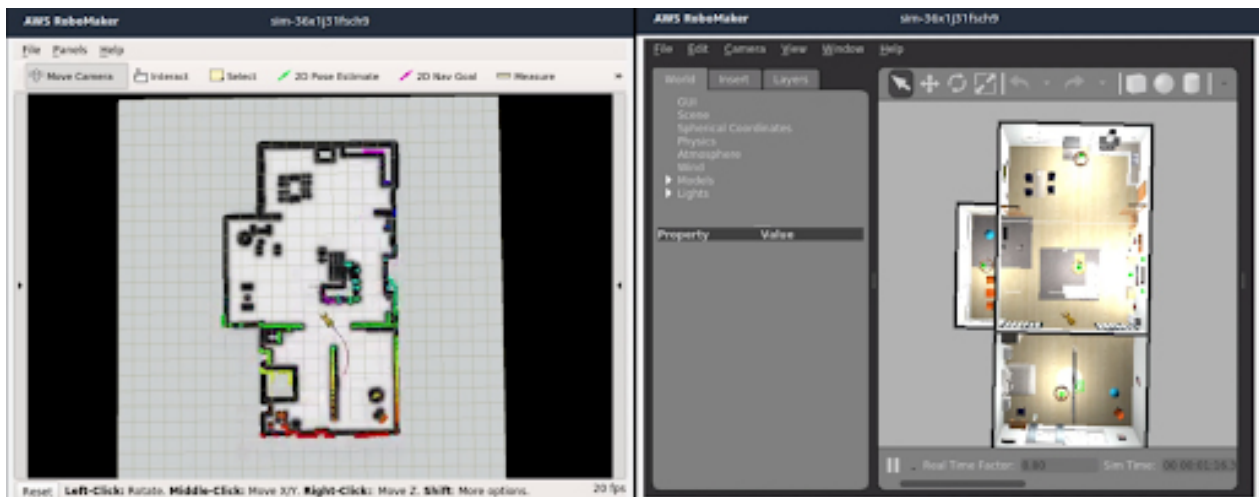
Planning Test: Navigation from Point A to B with Predefined Position and Orientation Tolerances

When you have tested how the robot interacts with different environments, you next need to test a planning algorithm. In video 2, you can see how the robot navigates in an apartment to a random goal.

The test verifies whether the robot navigates from point A to point B with

predefined accuracy while considering a goal yaw orientation. The test is marked as Failed, if a timeout occurred and the robot did not reach the goal within the predefined tolerance.

The test is marked as Passed, if the robot navigated to point B with predefined accuracy within time (less than the timeout). You can run many tests at scale to test how the robot navigates to a random point or set predefined points on the map.

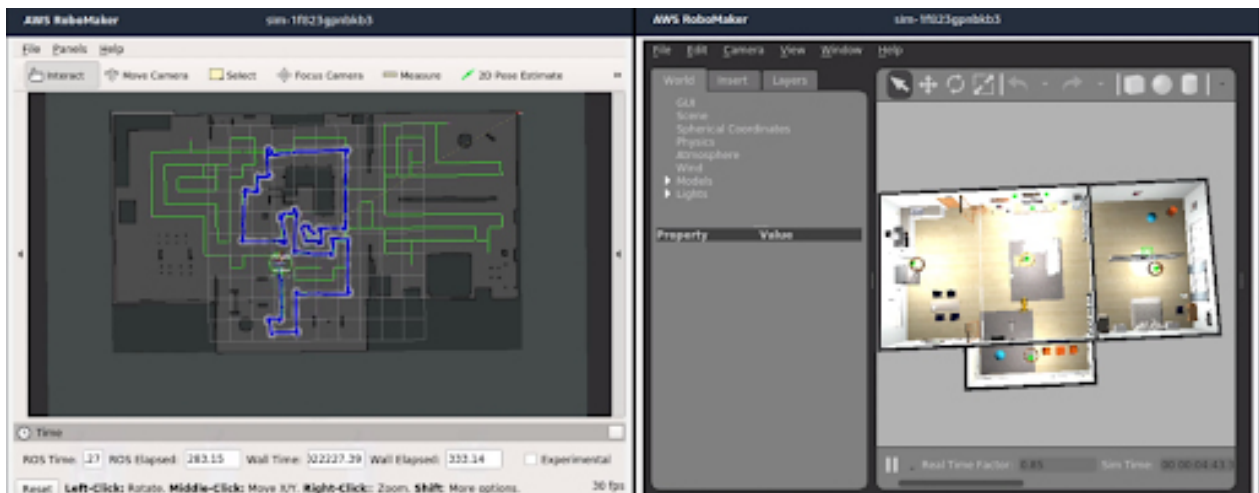


Video 2. Planning test: Navigation from point A to B with predefined position and orientation tolerances.

Behavior Tests: Coverage Test

The next type of testing is robot behavior tests. The most often required for mobile robots is the coverage test (see Video 3) that shows the efficiency of the robot's ability to cover the floor of the environment. The test monitors the robot's movement and creates a coverage grid with places where the robot already was.

Based on this map and parameters, like robot tool radius and coverage area offset, we can calculate coverage progress and compare it with the coverage goal. The test is marked as Failed, if a timeout occurred and the coverage progress is less than the coverage goal. The test is marked as Passed, if the coverage goal is reached within time, less than the timeout.



Video 3. Behavior tests: Coverage test.

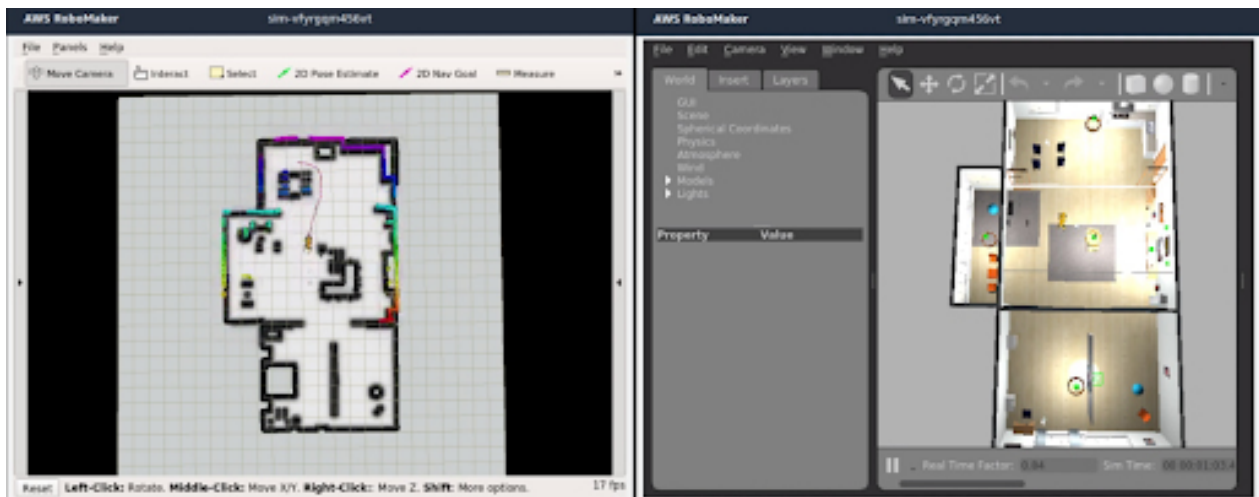
Localization Test

To satisfy the reliability of autonomous missions, the localization algorithm should work with the desired accuracy under different world disturbances. Thus, the developer should test the localization stack in a changeable world. For example, in video 4, you can see how the robot navigates and localizes itself when the objects disappeared from their initial position.

The test verifies whether a localization tool locates the robot pose correctly when the predefined amount of the world's objects is deleted or moved to predefined poses.

You can specify which object should be moved/deleted and their final destination when the Move option is activated.

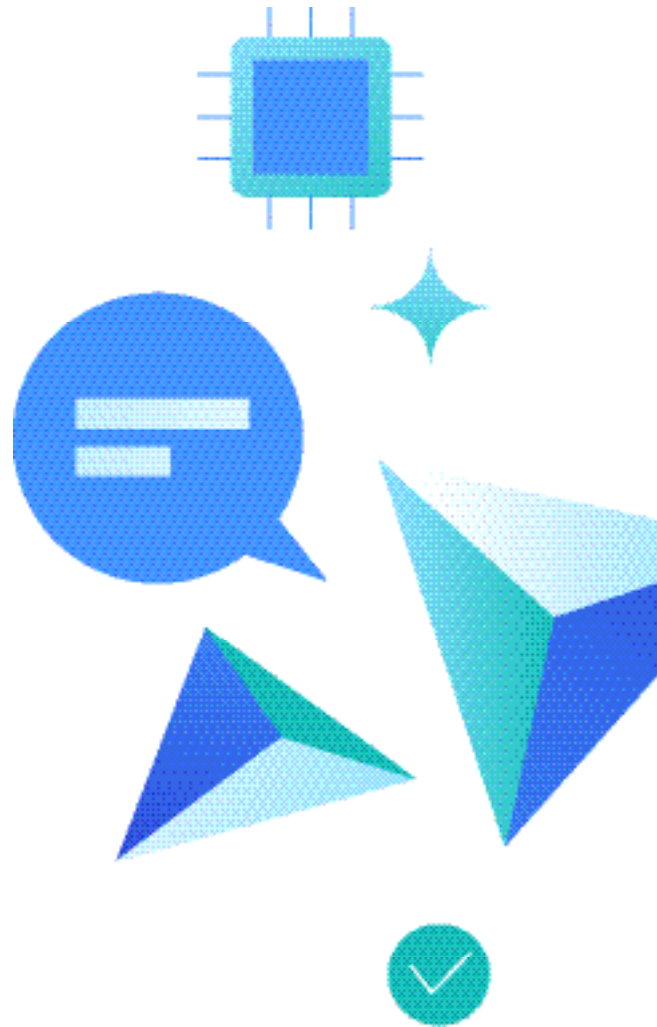
During the test, the robot navigates through the predefined or random points on the map until the robot has reached all points or timeout occurred. The test is marked as Failed, if a timeout has occurred or predefined points were reached, and the robot did not localize itself with desired tolerance. The test is marked as Passed, if a robot localizes itself with predefined accuracy before timeout or all points were reached, and the robot localizes itself correctly.



Video 4. Localization test.

More tests are available in the [ROS1 tests for AWS RoboMaker repository](#). They cover some basic scenarios and thus can decrease the time-to-market for robotics projects. We are working on adapting these tests for other types of robots: drones, robotic arms, and other.

LET'S TALK about your robotic software testing and how SoftServe's global team of experts may help you on your journey.



ABOUT US

SoftServe is a digital authority that advises and provides at the cutting-edge of technology. We reveal, transform, accelerate, and optimize the way enterprises and software companies do business. With expertise across healthcare, retail, energy, financial services, and more, we implement end-to-end solutions to deliver the innovation, quality, and speed that our clients' users expect.

SoftServe delivers open innovation, from generating compelling new ideas, to developing and implementing transformational products and services.

Our work and client experience is built on a foundation of empathetic, human-focused experience design that ensures continuity from concept to release.

We empower enterprises and software companies to (re)identify differentiation, accelerate solution development, and vigorously compete in today's digital economy-no matter where you are in your journey.

Visit our [website](#), [blog](#), [LinkedIn](#), [Facebook](#), and [Twitter](#) pages.

NORTH AMERICAN HQ

201 W 5th Street, Suite 1550
Austin, TX 78701 USA
+1 866 687 3588 (USA)
+1 647 948 7638 (Canada)

EUROPEAN HQ

30 Cannon Street
London EC4M 6XH
United Kingdom
+44 333 006 4341

APAC HQ

6 Raffles Quay
#14-07
Singapore 048580
+65 31 656 887

info@softserveinc.com
www.softserveinc.com

softserve