ÇANKAYA UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

Test Plan Document

CENG 408

Sentinel: Autonomous Discovery Vehicle

Team Members:

Turgut Utku ALTINKAYA

Burak ATEŞ

Yunus Emre DİNÇEL

Bayram Alper KILIÇ

İlteriş SAMUR

Table of Contents

1. Introduction

1.1. Version Control

Version Number	Description of Changes	Date
sentinel@1.0.0	First version including the autonomous movement and mapping in the simulation	27 March 2025

1.2. Overview

Sentinel: Autonomous Discovery Vehicle is an autonomous discovery vehicle that maps its environment in 2D and 3D using SLAM algorithms. Use cases and software design are detailed in the Software Requirements Specification (SRS) and Software Design Description (SDD) documents. This test plan outlines procedures for evaluating the vehicle's performance against these specified specifications in an unknown environment. It defines the test methodology, success and exit criteria, and expected results to ensure the system meets intended functionality and reliability standards.

1.3. Scope of The Document

This document will provide a detailed explanation of the test cases. Additionally, we will outline the features that are not to be tested, as well as our success, failure, and exit criteria. Finally, we will present the test results to assess the overall performance of the system.

1.4. Terminology

Acronym	Definition
SRS	Software Requirements Specification
SDD	Software Design Document
UI	User Interface
2D	Two-dimensional
3D	Three-dimensional

ROS	Robot Operating System
YOLO	You Look Only Once (Object Detection Framework)
LIDAR (Light detection and Ranging) Sensor	A sensor for determining ranges by targeting an object or a surface with a laser and measuring the time for the reflected to return to the receiver.
MUI	Material UI

2. Features To Be Tested:

2.1. Web UI

Web UI processes while The Sentinel activities showing on the web application as an Admin dashboard format.

TC_ID	Requirements	Priority	Scenario
WUI.01	Node.js, NPM, Wi-Fi connection, Joystick	Low	The user can control the Sentinel with a joystick and see its movements simultaneously in the web browser.
WUI.02	Node.js, Wi-Fi connection, Keyboard	Low	The user can control the Sentinel with a keyboard and see the pressed keys simultaneously in the web browser.
WUI.03	Node.js, NPM, Wi-Fi connection	Low	The user can view the real-time video captured by the camera on the web UI, pause it, and switch to full-screen mode.
WUI.04	Node.js, NPM, Wi-Fi connection	Low	The user can see the real-time generation of the 2D map on the web UI, switch to full-screen mode, and download the generated map.
WUI.05	Node.js, NPM, Wi-Fi connection	Low	The user can view the vehicle's movement direction through the 3D vehicle model in the web browser.
WUI.06	Node.js, NPM, Wi-Fi connection	Low	The user can see the 3D Map after it is completed.

WUI.07	Node.js, NPM, Wi-Fi connection	Low	The user can display the system logs by clicking the hamburger menu button located at the top left of the screen.

2.2. Hardware

Hardware processes while The Sentinel is prepared for real life applications.

TC_ID	Requirements	Priority	Scenario
H.01	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi	Low	The user can control the Sentinel without any loose contact in cables. The Sentinel must move to the desired direction instantly.
H.02	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi	Low	The motor driver can deliver the required voltage to motors for the movement with respect to given direction.
H.03	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi	Low	All of the motors can be turned to the desired way while Sentinel is controlling. (forward or backward)
H.04	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi	Low	The Sentinel's Lithium-Ion battery can deliver 1.5 hours of continuous operation at maximum power.
H.05	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi	Low	The Sentinel's powerbank can deliver 8 hours of continuous operation at maximum power

2.3. Manual Movement

Manual Movement processes while The Sentinel is controlled by a user from a joystick or keyboard in a simulation or real life.

TC_ID	Requirements	Priority	Scenario
MMV.01	Joystick (or Keyboard), Hardware Components	High	The Manual Movement will be processed while Sentinel is moving with controlled by the user, using the joystick (or keyboard) in real life.
MMV.02	Joystick (or Keyboard), Car Model, ROS Visualization Apps	Medium	The Manual Movement will be processed while Sentinel is moving with controlled by the user, using the joystick (or keyboard) in the simulation.

2.4. Autonomous Movement

Autonomous Movement processes while The Sentinel is controlled by exploration and decision-making algorithms, in a simulation or in real life.

TC_ID	Requirements	Priority	Scenario
AMV.01	Lidar, Hardware Components, Exploring Algorithms	Medium	The Autonomous Movement will be processed while Sentinel is moving without being controlled by the user, using the exploration of the objects, and deciding the path around the real-life environment.
AMV.02	Lidar, Exploring Algorithms, Car Model, ROS Visualization Apps	Medium	The Autonomous Movement will be processed while Sentinel is moving without being controlled by the user, using the exploration of the objects, and deciding the path around the simulation environment.

2.5. Manual Mapping

2D and 3D mapping process while the user drives the Sentinel manually from the joystick.

TC_ID	Requirements	Priority	Scenario
MM.2D.01	Joystick, Lidar	Low	The 2D map will be created while Sentinel is moving with user control in a straight line without objects in simulation.

MM.2D.02	Joystick, Lidar	Medium	The 2D map will be created while Sentinel is moving with user control in different directions without objects in simulation.
MM.2D.03	Joystick, Lidar	Medium	The 2D map will be created while Sentinel is moving with user control in a straight line with an object in simulation.
MM.2D.04	Joystick, Lidar	High	The 2D map will be created while Sentinel is moving around with user control with different objects in simulation.
MM.2D.05	Joystick, Lidar	High	The 2D map will be created while Sentinel is moving around with user control with different objects in the real room.

TC_ID	Requirements	Priority	Scenario
MM.3D.01	Joystick, Lidar, Camera	Medium	The 3D map will be created while Sentinel is moving around with user control with different objects in simulation.
MM.3D.02	Joystick, Lidar, Camera	High	The 3D map will be created while Sentinel is moving around with user control with different objects in the real room.

2.6. Autonomous Mapping

2D and 3D mapping process while the Sentinel is controlled by the autonomous algorithms.

TC_ID	Requirements	Priority	Scenario
AM.2D.01	Joystick, Lidar, Autonomous Movement	High	The 2D map will be created while Sentinel is moving, using an autonomous movement algorithm in a simulation.
AM.2D.02	Joystick, Lidar, Autonomous Movement	High	The 2D map will be created while Sentinel is moving, using an autonomous movement algorithm in a real room.

TC_ID	Requirements	Priority	Scenario
AM.3D.01	Joystick, Lidar, Camera, Autonomous Movement	High	The 3D map will be created while Sentinel is moving, using an autonomous movement algorithm in a simulation.
AM.3D.02	Joystick, Lidar, Camera,	High	The 3D map will be created while Sentinel is moving, using an autonomous movement

	Autonomous Movement		algorithm in a real room.

2.7. Object Detection

Object Detection processes with using real-time camera data.

TC_ID	Requirements	Priority	Scenario
OD.01	Camera, Movement, YOLO model	Medium	Object detection algorithms will be actively performed to identify and detect objects in the simulation world.
OD.02	Camera, Movement, YOLO model.	Medium	Object detection algorithms will be actively performed to identify and detect objects in the real world.
OD.03	Camera, Movement, YOLO model.	Medium	The system will identify and distinguish multiple objects in an environment, each with unique characteristics like shape, size, or color, and track them as they interact.
OD.04	Camera, Movement, YOLO model.	Medium	The system will detect objects at varying distances, adjusting its focus to accurately identify them.
OD.05	Camera, Movement, YOLO model.	Medium	Evaluate the accuracy of detected objects.
OD.06	Camera, Movement, YOLO model.	Medium	Object detection algorithm meets real-time performance requirements during movement

2.8. Simulation

TC_ID	Requirements	Priority	Scenario
S.01	Remote Computer, ROS2 Jazzy Harmonic	High	The movement of the Sentinel must be the same on RViz and Gazebo.
S.02	Remote Computer, ROS2 Jazzy Harmonic	High	The Sentinel can create a 2D map of the Gazebo simulation world.
S.03	Remote	High	The Sentinel can create a 3D map of the Gazebo

	Computer, ROS2 Jazzy, Gazebo Harmonic		simulation world.
S.04	Remote Computer, ROS2 Jazzy, Gazebo Harmonic	High	The Sentinel can move by avoiding objects on the Gazebo simulation world
S.05	Remote Computer, ROS2 Jazzy, Gazebo Harmonic	High	The Sentinel can move autonomously to a published point on the Gazebo simulation world.

3. Detailed Test Cases

3.1 Web UI

TC_ID	WUI.01
Purpose	Viewing real-time joystick controls on the web dashboard.
Requirements	Node.js, NPM, Wi-Fi connection, Joystick
Priority	Low
Estimated Time Needed	1 hour
Dependency	Movement data from the ROS topic
Setup	Run the React Application, have Wi-Fi Connection, Connect to Rosbridge server

Procedure	1. Open the web dashboard 2. Use the Joystick to publish data on movement topic 3. Observe whether the joystick on the web moves exactly the same as the physical one.
Cleanup	Close the web dashboard

TC_ID	WUI.02
Purpose	Viewing real-time keyboard controls on the web dashboard.
Requirements	Node.js, NPM, Wi-Fi connection, Keyboard
Priority	Low
Estimated Time Needed	1 hour
Dependency	Movement data from the ROS topic
Setup	Run the React Application, have Wi-Fi Connection, Connect to Rosbridge server
Procedure	1. Open the web dashboard 2. Use the keyboard to publish data on movement topic 3. Observe whether the keyboard on the web moves exactly the same as the physical one.
Cleanup	Close the web dashboard

TC_ID	WUI.03
Purpose	Viewing the real-time video footage on the web dashboard

Requirements	Node.js, NPM, Wi-Fi connection
Priority	Low
Estimated Time Needed	1 hour
Dependency	Camera data from the ROS topic, a stable internet connection
Setup	Run the React Application, have Wi-Fi Connection, Connect to Rosbridge server
Procedure	1. Open the web dashboard 2. Publish the camera data to camera topic from Sentinel 3. Click on the play button of the camera container on web dashboard 4. Observe whether the camera data arrives in real-time. 5. Click on the expand button to see the camera in full-screen. 6. Pause the camera
Cleanup	Close the web dashboard

TC_ID	WUI.04
Purpose	Viewing the generated 2D Map on the web dashboard
Requirements	Node.js, NPM, Wi-Fi connection
Priority	Low
Estimated Time Needed	2 hour
Dependency	Map data from the ROS topic

Setup	Run the React Application, have Wi-Fi Connection, Connect to Rosbridge server
Procedure	1. Open the web dashboard 2. Publish the map data to map topic from remote computer system 3. Click on the play button of the 2D Map container on web dashboard 4. Observe whether the 2D Map data arrives in real-time. 5. Click on the expand button to see the 2D Map in full-screen. 6. Pause the 2D Map 7. Download the generated 2D Map
Cleanup	Close the web dashboard

TC_ID	WUI.05
Purpose	Viewing the Sentinel’s movement direction on the web dashboard in real-time
Requirements	Node.js, NPM, Wi-Fi connection
Priority	Low
Estimated Time Needed	2 hours
Dependency	Movement data from the ROS topic, (Keyboard, Joystick, and autonomous Movement)
Setup	Run the React Application, have Wi-Fi Connection, Connect to Rosbridge server

Procedure	1. Open the web dashboard 2. Publish the movement data to map topic from remote computer system 3. Observe whether the 3D Model moves at the same direction as the Joystick or Keyboard movement.
Cleanup	Close the web dashboard

TC_ID	WUI.06
Purpose	Viewing the 3D Map after the mapping is completed
Requirements	Node.js, NPM, Strong Wi-Fi connection
Priority	Low
Estimated Time Needed	2 hours
Dependency	Generated 3D Map from Rtabmap.
Setup	Run the React Application, have Wi-Fi Connection, Connect to Rosbridge server
Procedure	1. Open the web dashboard 2. Click on the generate 3D Map button in the 3D Map container section 3. Observe whether the 3D Map is generated
Cleanup	Close the web dashboard

TC_ID	WUI.07
Purpose	Viewing the active system logs

Requirements	Node.js, NPM, Wi-Fi connection
Priority	Low
Estimated Time Needed	1 hour
Dependency	No dependencies
Setup	Run the React Application, have Wi-Fi Connection, Connect to Rosbridge server
Procedure	1. Open the web dashboard 2. Click on the generate hamburger menu button located at the top left of the screen. 3. Do an operation (Start Camera, Move the vehicle, Download 2D Map) or see whether an object is detected and printed in the logs
Cleanup	Close the web dashboard

3.2. Hardware

TC_ID	H.01
Purpose	Controlling the Sentinel
Requirements	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi
Priority	Low
Estimated Time Needed	5 minutes
Dependency	Well connected wires.
Setup	Give power to motors and Raspberry Pi. Connect the joystick.
Procedure	1. Run the manual movement package on both Raspberry Pi and Remote Computer. 2. Control the car via joystick or keyboard 3. Check the movement of the car

Cleanup	Interrupt both packages. Power off the battery and Raspberry Pi.

TC_ID	**H.02**
Purpose	Controlling the Sentinel
Requirements	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi
Priority	Low
Estimated Time Needed	5 minutes
Dependency	Well connected wires.
Setup	Give power to motors and Raspberry Pi. Connect the joystick.
Procedure	1. Run the manual movement package on both Raspberry Pi and Remote Computer. 2. Drive the Sentinel via joystick or keyboard 3. Check the voltage values on motor driver via voltmeter
Cleanup	Interrupt both packages. Power off the battery and Raspberry Pi.

TC_ID	**H.03**
Purpose	Controlling the Sentinel
Requirements	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi
Priority	Low
Estimated Time Needed	5 minutes
Dependency	Well connected wires.
Setup	Give power to motors and Raspberry Pi. Connect the joystick.
Procedure	1. Run the manual movement package on both Raspberry Pi and Remote Compuer. 2. Drive the car in backward and forward direction. 3. Check the movement of all motors
Cleanup	Interrupt both packages. Power off the battery and Raspberry Pi.

TC_ID	**H.04**
Purpose	Powering the motors

Requirements	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi
Priority	Low
Estimated Time Needed	1.5 Hours
Dependency	Fully charged battery.
Setup	Give power to motors and Raspberry Pi.
Procedure	1. Run the desired package that uses motors on the system. 2. Check the time
Cleanup	Interrupt packages. Power off the battery and Raspberry Pi.

TC_ID	**H.05**
Purpose	Powering the Raspberry Pi
Requirements	Wires, Tires, Motors, Car Chassis, Motor Driver, Battery, Powerbank, Raspberry Pi
Priority	Low
Estimated Time Needed	8 Hours
Dependency	Fully charged powerbank.
Setup	Give power to the Raspberry Pi.
Procedure	1. Open the Raspberry Pi. 2. Check the time
Cleanup	Power off the Raspberry Pi.

3.3. Manual Movement

TC_ID	**MMV.01**
Purpose	The Sentinel is manually moved in real life.
Requirements	Joystick (or Keyboard), Hardware Components.
Priority	High
Estimated Time Needed	5-10 seconds

Dependency	The Hardware Components should be fully working.
Setup	Activate the motors using the motor driver and supply energy from battery, connect the Raspberry Pi, and connect a joystick or keyboard.
Procedure	1. Activate Motors. 2. Connect the Raspberry Pi. 3. Run the movement code. 4. Control the joystick or keyboard from the Remote Computer. 5. Move around real life and explore the environment.
Cleanup	Close the Sentinel.

TC_ID	**MMV.02**
Purpose	The Sentinel is manually moved in simulation.
Requirements	Joystick (or Keyboard), Car Model, ROS Visualization Apps
Priority	Medium
Estimated Time Needed	15-20 seconds
Dependency	Car Model with the real car features.
Setup	Enable the Car model, and connect the joystick or keyboard. Activate visulization app.
Procedure	1. Open the ROS Visualization App. (RViz, Gazebo) 2. Add the Car Model. 3. Connect the joystick or keyboard to the computer. 4. Run the Movement code for simulation. 5. Move around the simulation world.
Cleanup	Close the simulation.

3.4. Autonomous Movement

TC_ID	**AMV.01**
Purpose	The Sentinel is autonomously moved in real life.
Requirements	Lidar, Hardware Components, Exploring Algorithms
Priority	Medium
Estimated Time Needed	1-2 minutes.
Dependency	The Hardware Components and Exploring and Deciding algorithms should be fully working.
Setup	Activate the motors using the motor driver and supply energy from battery, connect the Raspberry Pi, and run the exploring and deciding path algorithms.
Procedure	1. Activate Motors. 2. Connect the Raspberry Pi. 3. Run the autonomous movement codes. 4. Wait the explore the area, and decide the path to move. 5. After exploring the whole room area, stop the movement.
Cleanup	Close the Sentinel.

TC_ID	AMV.02
Purpose	The Sentinel is autonomously moved in simulation.
Requirements	Lidar, Exploring Algorithms, Car Model, ROS Visualization Apps.
Priority	Medium
Estimated Time Needed	1-2 minutes.
Dependency	The Car Model should be added to the ROS Visualization App, and the Exploring and Deciding algorithms should be fully working.
Setup	Enable the Car Model in the ROS Visualization App, and run the exploration and decision path algorithms.
Procedure	1. Open the ROS Visualization App. (Gazebo) 2. Add the Car Model. 3. Run the autonomous movement codes. 4. Exploring the simulation world. 5. Move around the simulation world.
Cleanup	Close the simulation.

3.5. Manual Mapping

TC_ID	MM.2D.01
Purpose	2D mapping a single line
Requirements	Joystick, Lidar
Priority	Low
Estimated Time Needed	5-10 seconds
Dependency	Accurate lidar data
Setup	Active Lidar sensor, connect joystick
Procedure	4. Open simulation world 5. Clear all the objects 6. Run the movement code 7. Move the Sentinel forward and backward
Cleanup	Close the simulation

TC_ID	MM.2D.02

Purpose	2D mapping an empty virtual room
Requirements	Joystick, Lidar
Priority	Medium
Estimated Time Needed	60-120 seconds
Dependency	Accurate lidar data
Setup	Active Lidar sensor, connect joystick
Procedure	1. Open simulation world 2. Clear all the objects 3. Run the movement code 4. Move the Sentinel in four directions
Cleanup	Close the simulation

TC_ID	**MM.2D.03**
Purpose	2D mapping an object over a line
Requirements	Joystick, Lidar
Priority	Medium
Estimated Time Needed	5-10 seconds
Dependency	Accurate lidar data
Setup	Active Lidar sensor, connect joystick
Procedure	1. Open simulation world 2. Place one object to world 3. Run the movement code 4. Move the Sentinel forward and backward 5. Check the map for an object appearance from Lidar data
Cleanup	Close the simulation

TC_ID	**MM.2D.04**
Purpose	2D mapping a virtual room with many objects
Requirements	Joystick, Lidar
Priority	High
Estimated Time Needed	3 minutes

Dependency	Accurate lidar data
Setup	Active Lidar sensor, connect joystick
Procedure	1. Open simulation world 2. Place objects to different locations 3. Run the movement code 4. Move the Sentinel to recognize objects and the world 5. Check the created 2D map with respect to created world
Cleanup	Close the simulation

TC_ID	**MM.2D.05**
Purpose	2D mapping in a real room
Requirements	Joystick, Lidar
Priority	High
Estimated Time Needed	4 minutes
Dependency	Accurate lidar data
Setup	Active Lidar sensor, connect joystick, activate motors
Procedure	1. Connect raspberry pi to power 2. Open motors’ switch 3. Run the movement code from Sentinel 4. Run the movement code from Server 5. Launch Rviz2 and SlamToolBox 6. Move the Sentinel inside the room 7. Check if 2D map matches with real room
Cleanup	Close the Sentinel and other code blocks

TC_ID	**MM.3D.01**
Purpose	Create 3D map of simulated world
Requirements	Joystick, Lidar, Camera
Priority	Medium

Estimated Time Needed	3 minutes
Dependency	Accurate lidar data and camera data
Setup	Active Lidar sensor, activate camera, connect joystick
Procedure	1. Open simulation world 2. Place the objects into desired locations 3. Run the movement code 4. Move the Sentinel to recognize objects and the world 5. Put images and lidar data together 6. Create 3D map using Rtabmap 7. Check if the generated map is matches the simulated world
Cleanup	Close the simulation

TC_ID	**MM.3D.02**
Purpose	Create 3D map of real life room
Requirements	Joystick, Lidar, Camera
Priority	High
Estimated Time Needed	4 minutes
Dependency	Accurate lidar data and camera data
Setup	Active Lidar sensor, activate camera , connect joystick, activate motors
Procedure	1. Connect raspberry pi to power 2. Open motors' switch 3. Run the movement code from Sentinel 4. Run the movement code from Server 5. Launch Rtabmap 6. Move Sentinel to explore the room 7. Put images and lidar data together 8. Create 3D map using Rtabmap 9. Check if the generated map is matches the simulated world
Cleanup	Close the Sentinel and code blocks

3.6. Autonomous Mapping

TC_ID	**AM.2D.01**

Purpose	2D mapping in simulation autonomously
Requirements	Joystick, Lidar
Priority	High
Estimated Time Needed	3 minutes
Dependency	Accurate lidar data
Setup	Active Lidar sensor, connect joystick
Procedure	1. Open simulation world 2. Place the objects into desired locations 3. Run Rviz2 4. Run the autonomous movement code 5. Wait for the Sentinel to move around and recognize objects 6. Check if the generated 2D map is matches the simulated world 7. Check for possible crashes
Cleanup	Close the Simulation and other code blocks

TC_ID	**AM.2D.02**
Purpose	2D mapping in a real room autonomously
Requirements	Joystick, Lidar
Priority	High
Estimated Time Needed	4 minutes
Dependency	Accurate lidar data
Setup	Active Lidar sensor, connect joystick, activate motors
Procedure	1. Connect raspberry pi to power 2. Open motors' switch 3. Run the autonomous movement code from Sentinel 4. Run the autonomous movement code from Server 5. Run Rviz2 6. Wait for the Sentinel to move around and recognize objects 7. Check if the generated 2D map is matches the simulated world 8. Check for possible crashes
Cleanup	Close the Sentinel and other code blocks

TC_ID	**AM.3D.01**

Purpose	3D mapping in a simulation autonomously
Requirements	Joystick, Lidar, Camera
Priority	High
Estimated Time Needed	3 minutes
Dependency	Accurate lidar data and camera data
Setup	Active Lidar sensor, activate camera , connect joystick
Procedure	1. Open simulation world 2. Place the objects into desired locations 3. Run Rviz2, Rtabmap 4. Run the autonomous movement code 5. Wait for the Sentinel to move around and recognize objects 6. Check if the generated 3D map is matches the simulated world
Cleanup	Close the simulation and other code blocks

TC_ID	**AM.3D.02**
Purpose	3D Mapping in real room autonomously
Requirements	Joystick, Lidar,Camera
Priority	High
Estimated Time Needed	4 minutes
Dependency	Accurate lidar data and camera data
Setup	Active Lidar sensor, activate camera , connect joystick, activate motors
Procedure	1. Connect raspberry pi to power 2. Open motors' switch 3. Run the autonomous movement code from Sentinel 4. Run the autonomous movement code from Server 5. Run Rviz2, Rtabmap 6. Wait for the Sentinel to move around and recognize objects 7. Check if the generated 3D map is matches the simulated world
Cleanup	Close the Sentinel and other code blocks

3.7. Object Detection

TC_ID	OD.01
Purpose	Detect the objects in the simulation world.
Requirements	Camera, Movement, YOLO model.
Priority	Medium
Estimated Time Needed	3 minutes
Dependency	Camera data
Setup	Active camera, download object detection model.
Procedure	1. Open simulation world. 2. Listen camera and start object detection model. 3. Place the objects into the desired location in the simulation world. 4. Turn Sentinel’s camera to the object. 5. Wait model to detect objects. 6. Check the detected object.
Cleanup	Close the Sentinel and other code blocks

TC_ID	OD.02
Purpose	Detect the objects in the real world.
Requirements	Camera, Movement, YOLO model.
Priority	Medium
Estimated Time Needed	15 minutes
Dependency	Camera data
Setup	Active camera, download object detection model.
Procedure	1. Connect Raspberry Pi to power 2. Open motors’ switch 3. Run the movement and camera code from Sentinel 4. Run the movement code from Server 5. Listen camera and start object detection model 6. Turn Sentinel’s camera to the object 7. Wait model to detect objects.

	8. Check the detected object
Cleanup	Close the Sentinel and other code blocks

TC_ID	**OD.03**
Purpose	Distinguish multiple object from each other.
Requirements	Camera, Movement, YOLO model.
Priority	Medium
Estimated Time Needed	15 minutes
Dependency	Camera data
Setup	Active camera, download object detection model.
Procedure	1. Open Simulation or Start Sentinel. 2. Listen camera and start object detection model. 3. Place the different objects in the desired locations with close distances. 4. Turn Sentinel's camera to the objects. 5. Wait model to detect objects. 6. Check each object detected properly.
Cleanup	Close the Sentinel and other code blocks

TC_ID	**OD.04**
Purpose	Detect objects from different distances.
Requirements	Camera, Movement, YOLO model.
Priority	Medium
Estimated Time Needed	8 minutes
Dependency	Camera data
Setup	Active camera, download object detection model.
Procedure	1. Open Simulation or Start Sentinel 2. Listen camera and start object detection model 3. Place objects at multiple known ranges like close, mid-range or far. 4. Turn Sentinel's camera to the objects 5. Wait model to detect objects. 6. Check the detected object for each distance.
Cleanup	Close the Sentinel and other code blocks

TC_ID	**OD.05**
Purpose	Evaluate the reliability of detection by evaluating the rate of false positives and negatives.
Requirements	Camera, Movement, YOLO model.
Priority	Medium
Estimated Time Needed	20 minutes
Dependency	Camera
Setup	Active camera, download object detection model.
Procedure	1. Open Simulation or Start Sentinel 2. Listen camera and start object detection model 3. Place objects in a patterned background or with reflections. 4. Turn Sentinel’s camera to the objects. 5. Wait model to detect objects. 6. Log instances of false detection and missed detections. 7. Evaluate the model performance.
Cleanup	Close the Sentinel and other code blocks

TC_ID	**OD.06**
Purpose	Ensure the detection algorithm meets real-time performance requirements during both manual and autonomous operation.
Requirements	Camera, Movement, YOLO model.
Priority	Medium
Estimated Time Needed	1 hour
Dependency	Camera
Setup	Active camera, download object detection model.
Procedure	1. Open Simulation or Start Sentinel 2. Listen camera and start object detection model 3. Place the different objects in the desired and distinct locations. 4. Move Sentinel around the environment. 5. Monitor the latency between object appearance and detection reporting.

Cleanup	Close the Sentinel and other code blocks

3.8 Simulation

TC_ID	S.01
Purpose	Movement of the Sentinel on simulation world
Requirements	Remote Computer, ROS2 Jazzy, Gazebo Harmonic
Priority	High
Estimated Time Needed	10 Minutes
Dependency	Installed RViz and Gazebo Harmonic
Setup	The accurate URDF model of Sentinel
Procedure	1. Run simulation package 2. Compare the movement of Sentinel on RViz and Gazebo it must be same
Cleanup	Interrupt the package

TC_ID	S.02
Purpose	2D mapping
Requirements	Remote Computer, ROS2 Jazzy, Gazebo Harmonic
Priority	High
Estimated Time Needed	10 Minutes
Dependency	Installed RViz, Gazebo Harmonic and SLAM Toolbox
Setup	The accurate URDF model of Sentinel
Procedure	1. Run the simulation package 2. Start to mapping 3. Check the created map on RViz
Cleanup	Interrupt the package

TC_ID	S.03

Purpose	3D mapping
Requirements	Remote Computer, ROS2 Jazzy, Gazebo Harmonic
Priority	High
Estimated Time Needed	10 Minutes
Dependency	Installed RViz and Gazebo Harmonic
Setup	The accurate URDF model of Sentinel
Procedure	1. Run simulation package 2. Start to mapping 3. Check the created map on RTAB-Map
Cleanup	Interrupt the package

TC_ID	**S.04**
Purpose	Object Avoidance
Requirements	Remote Computer, ROS2 Jazzy, Gazebo Harmonic
Priority	High
Estimated Time Needed	10 Minutes
Dependency	Installed RViz and Gazebo Harmonic
Setup	The accurate URDF model of Sentinel
Procedure	1. Run the simulation package 2. Put obstacle objects on the Gazebo Simulation World 3. Start autonomous movement 4. Check the movement
Cleanup	Interrupt the package

TC_ID	**S.05**
Purpose	Movement of the Sentinel on simulation world
Requirements	Remote Computer, ROS2 Jazzy, Gazebo Harmonic
Priority	High
Estimated Time Needed	10 Minutes
Dependency	Installed RViz and Gazebo Harmonic
Setup	The accurate URDF model of Sentinel

Procedure	1. Run the simulation package 2. Publish a 2D pose estimation on RViz 3. Check the movement
Cleanup	Interrupt the package

4. Features To Be Not Tested

4.1. Windows Operating System Compatibility

Since all the developments and validations are performed on Ubuntu systems, testing on Windows environment are excluded. This prevents inconsistencies result from operating system-specific behaviors that are out of scope.

4.2. Driving on Flat Surfaces Only

All the test are limited to flat surfaces, since the Sentinel's navigation and performance are only validated in these conditions. Testing on the rough terrain is beyond of the current scope of the Sentinel.

4.3. Well-Lit Environment

Testings are performed in a controlled, well-lit environment to ensure consistent sensor performance and reduce variability. Low or variable lighting conditions are outside the current test parameters.

4.4. Limited to the ROS-Jazzy and Gazebo Harmonic Versions

The system only be evaluated using ROS-Jazzy and Gazebo Harmonic versions. Compatibility with older versions or alternative releases will not be tested.

4.5. Internet Connectivity and Real Time Data

The Sentinel is designed to remain constantly connected to the internet to send and receive real-time data for remote monitoring, control, and updates. Testing without an internet connection or under intermittent network conditions is out of scope for the Sentinel.

5. Pass/Fail Criteria

5.1. Manual Movement

The Sentinel must move in four directions without struggling. Also, the linear speed of the Sentinel must be adjustable.

5.2. Autonomous Movement

The Sentinel will use an autonomous movement algorithm that will handle driving without any crashes and explore the whole room that Sentinel has in it.

5.3. Manual Mapping

The algorithm should create at least 95% accurate 2D maps in a simulation environment and 90% accurate 2D maps in real life conditions while the Sentinel is driven by the user.

The 3D map creating algorithm should properly create objects in the correct location of themselves and place them accordingly with a maximum of 10% error margin in both simulation and real life conditions while the Sentinel is driven by the user.

5.4. Autonomous Mapping

The algorithm should create at least 95% accurate 2D maps in a simulation environment and 90% accurate 2D maps in real life conditions while the Sentinel is driven by the autonomous drive algorithm.

The 3D map creating algorithm should properly create objects in the correct location of themselves and place them accordingly with a maximum of 10% error margin in both simulation and real life conditions while the Sentinel is driven by the autonomous drive algorithm.

5.5. Object Detection

The object detection algorithm must process frames fast enough to catch the camera stream, and must classify objects with respect to these images. The accuracy of classifying objects should have accuracy higher than 75%.

5.6. Simulation

The simulation must mimic real world scenarios such as friction, objects, lidar data and camera view. Simulation and real-world movement must be identical to each other. (For example, if Sentinel turns with 45 degrees angle, the simulated car must also be turned with 45 degrees angle.

6. Exit Criteria

The testing of the Sentinel is considered successful under the following conditions:

- 100% of the test cases are executed.
- 90% of the test cases passed.
- All High and Medium Priority test cases passed.